Electronique Numérique

Chapitre 2 : Logique combinatoire

Pr . YOUSSEF MOUZOUNA

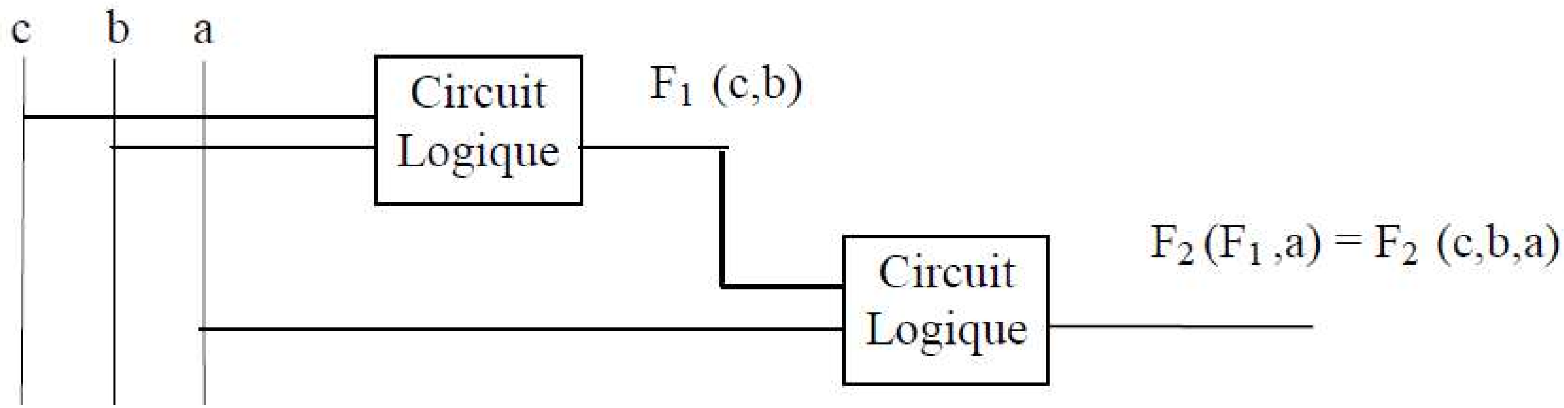
Chapitre2:Logique combinatoire

I. Les fonctions logiques:

❑ Le fonctionnement d'un système logique est décrit par une ou plusieurs **propositions logiques** simples qui présentent le caractère binaire « VRAI » ou « FAUX ».

❑ La relation sorties-entrées appelée fonction de transfert du système est décrite par une ou plusieurs fonctions logiques qui traduisent algébriquement les propositions logiques.

❑ Une fonction logique qui prend les valeurs 0 ou 1 peut être considérée comme une variable binaire pour une autre fonction logique.

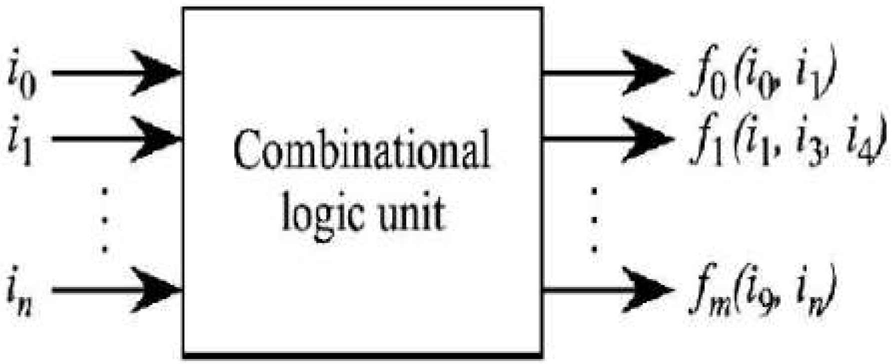


❑ Pour décrire le fonctionnement d'un système en cherchant l'état que doit prendre la sortie pour toutes les combinaisons possibles des entrées, on utilisera ce qu'on appelle « la table de vérité »

Chapitre2:Logique combinatoire

I. Les fonctions logiques:

- ❑ Pour décrire le fonctionnement d'un système en cherchant l'état que doit prendre la sortie pour toutes les combinaisons possibles des entrées, on utilisera ce qu'on appelle « la table de vérité »
- ❑La table de vérité d'une fonction de n variables a 2n lignes - états d'entrée
- ❑Algèbre de Boole et les fonctions logiques sont le support théorique des circuits combinatoires



i_0	i_1	$F_0(i_0, i_1)$
0	0	
0	1	
1	0	
1	1	

i_1	i_3	i_4	$F_1(i_1, i_3, i_4)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
.	.	.	
1	1	1	

Table de vérité

II. Les Portes logiques:

- En électronique les deux états d'une variable booléenne sont associés à deux niveaux de tension : V(0) et V(1) pour les états 0 et 1 respectivement.
- On distingue les logiques positive et négative selon que V(1) > V(0) ou V(1) < V(0)

Niveau	Logique positive	Logique négative
Haut	1	0
Bas	0	1

- Toute fonction logique peut être réalisée à l'aide d'un nombre de fonctions logiques de base appelées portes
- Un circuit se représente par un logigramme

II. Les Portes logiques:

Les portes logiques sont des circuits électroniques dont les fonctions de transfert (relation entre les entrées et les sorties) matérialisent les opérations de base appliquées à des variables électriques.

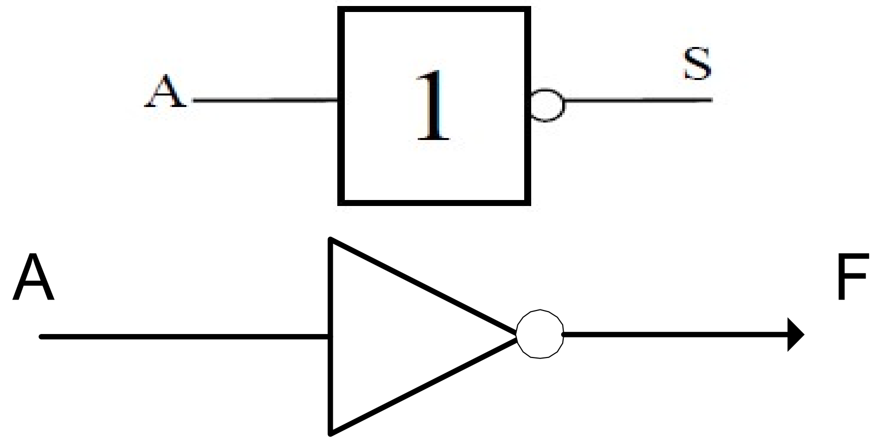
- La porte **Non (**en anglais **not)**

Représentation:

$$S = \overline{A}$$

Table de vérité

Entrée	*Sortie*
A	F
0	1
1	0



Symbole graphique

II. Les Portes logiques:

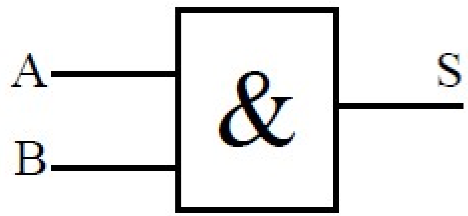
- La porte **ET (**en anglais **and)**

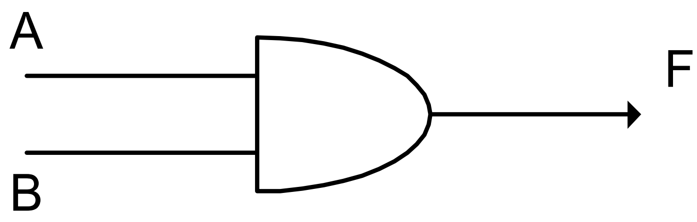
Représentation:

$$S = A * B$$

Table de vérité

Entrée		***Sortie***
B	A	F
0	0	0
0	1	0
1	0	0
1	1	1





Symbole graphique

II. Les portes logiques:

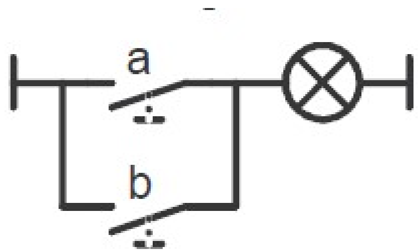
- La porte **OU**;(en anglais **OR)**

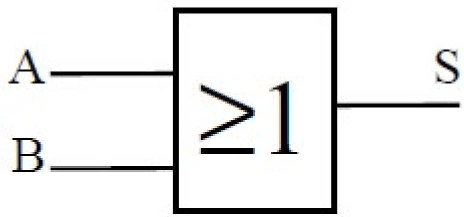
Représentation:

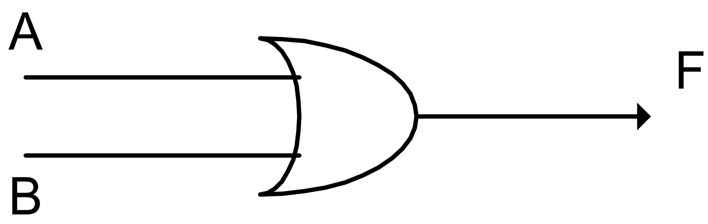
$$S = A + B$$

Table de vérité

Entrée		*Sortie*
B	A	F
0	0	0
0	1	1
1	0	1
1	1	1







Symbole graphique

II. Les portes logiques:

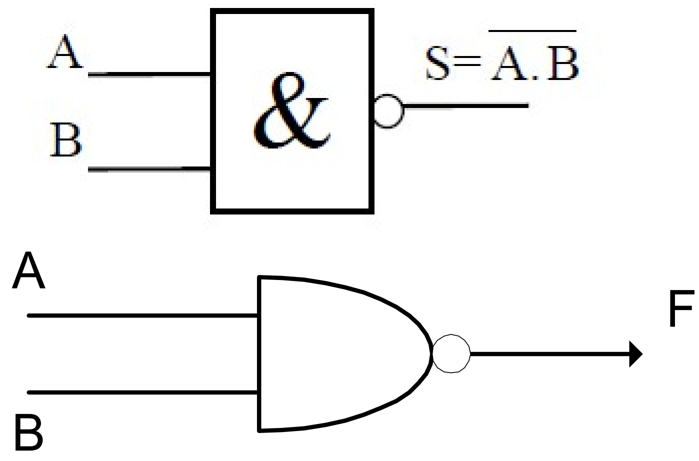
- La porte**NAND**

Représentation:

$$S = \overline{A * B}$$

Table de vérité

Entrée		***Sortie***
B	A	F
0	0	1
0	1	1
1	0	1
1	1	0



Symbole graphique

II. Les portes logiques:

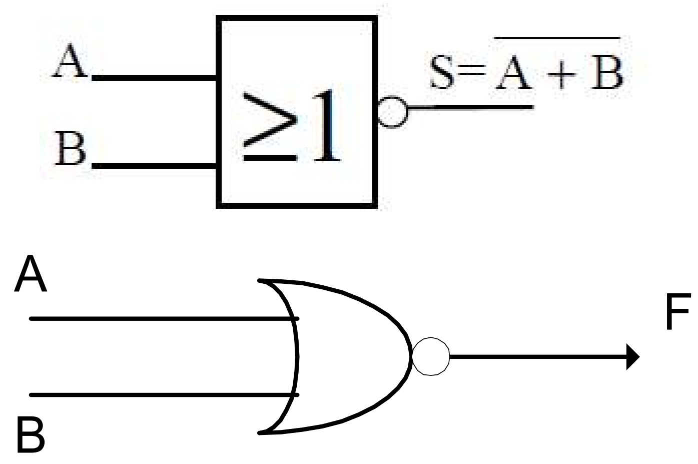
- La porte **NOR**

Représentation:

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

Table de vérité

Entrée		***Sortie***
B	A	F
0	0	1
0	1	0
1	0	0
1	1	0



Symbole graphique

II. Les portes logiques:

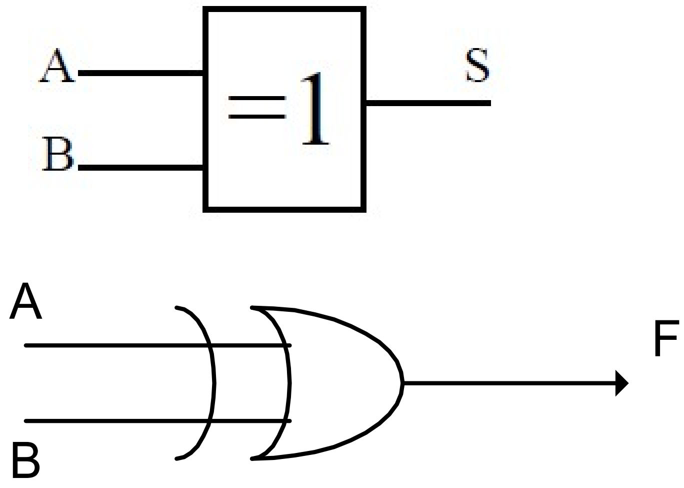
- La porte **XOR** , OU Exclusif

Représentation:

$$S = A \oplus B = \overline{A}B + A\overline{B}$$

Table de vérité

Entrée		***Sortie***
B	A	F
0	0	0
0	1	1
1	0	1
1	1	0



Symbole graphique

Exercice: Représenter les fonctions f et g avec des portes logique:

$$f = AB + C\overline{D}E + \overline{F}$$

$$g = (A + \overline{B} + C)(B + D)\overline{E}$$

III. Réalisation d´une fonction booléenne:

- Toute fonction logique peut être réalisée à l’aide des portes logiques

- Réalisation d’une fonction booléenne:

  - ✓ Écrire l’équation de la fonction à partir de sa table de vérité
  - ✓ Simplifier l’équation
  - ✓ Réaliser l’équation à l’aide des portes disponibles

III. Réalisation d´une fonction booléenne:

❑ *Comment rendre une table de vérité une fonction booléenne*

• À partir de la table de vérité, nous pouvons avoir deux formes analytiques, dénommées formes canoniques

✓ somme canonique de produits (SOP)

(forme disjonctive) c'est la somme logique(ou réunion) des ***mintermes*** associés aux combinaisons pour lesquelles la fonction vaut 1 (somme de produits).

✓ produit canonique de sommes (POS)

(forme conjonctive): c'est le produit logique (ou intersection) des ***maxtermes*** associés aux combinaisons pour lesquelles la fonction vaut 0 (produit de sommes).

III. Réalisation d´une fonction booléenne:

❑ *Comment rendre une table de vérité une fonction booléenne*

- **Mintermes**

Minterme : il est défini comme étant le produit logique des variables booléennes considérées avec la convention suivante :

- Si la variable est égale à 1 alors inscrire la variable elle-même.
- Si la variable est égale à 0 alors inscrire son complément.

- **Maxtermes**

Maxterme : il est défini comme étant la somme logique des variables booléennes considérées avec la convention suivante :

- Si la variable est égale à 0 alors inscrire la variable elle-même.
- Si la variable est égale à 1 alors inscrire son complément.

III. Réalisation d´une fonction booléenne:

❑ *Comment rendre une table de vérité une fonction booléenne*

Combinaison	A	B	C	F	Minterme	Maxterme
0	0	0	0	0	$\bar{A}\bar{B}\bar{C}$	$A + B + C$
1	0	0	1	0	$\bar{A}\bar{B}C$	$A + B + \bar{C}$
2	0	1	0	0	$\bar{A}B\bar{C}$	$A + \bar{B} + C$
3	0	1	1	1	$\bar{A}BC$	$A + \bar{B} + \bar{C}$
4	1	0	0	0	$A\bar{B}\bar{C}$	$\bar{A} + B + C$
5	1	0	1	1	$A\bar{B}C$	$\bar{A} + B + \bar{C}$
6	1	1	0	1	$AB\bar{C}$	$\bar{A} + \bar{B} + C$
7	1	1	1	1	ABC	$\bar{A} + \bar{B} + C$

III. Réalisation d´une fonction booléenne:

❑ *Comment rendre une table de vérité une fonction booléenne*

- la fonction **F(A,B,C) est à l'état 1 pour les combinaisons 3,5, 6 et 7. On l'écrit sous une** forme dite numérique : **F(A,B,C) =R(3,5,6,7) (c'est-à-dire réunion des combinaisons** 3, 5, 6 et7). D'où la première forme canonique de F :

$$F = \overline{A}BC + ABC + A\overline{B}C + AB\overline{C})$$

- la fonction **F(A,B,C) est à l'état 0 pour les combinaisons 0, 1, 2 et 4. On l'écrit sous la** forme numérique : **F(A,B,C) =I(0,1,2,4) (c'est-à-dire intersection des combinaisons 0,**1, 2 et 4). D'où la deuxième forme canonique de F :

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

III. Réalisation d´une fonction booléenne:

- ***Ecriture canoniques, SOP:*** ***on fait la sélections des uns***

A	B	C	F	$P_3 + P_5 + P_6 + P_7$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Cette façon, très générale, d'écrire une fonction booléenne est appelée somme canonique de produits (SOP)

$$F(A, B, C) = P_3 + P_5 + P_6 + P_7$$

$$F(A,B,C) = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = \sum(3,5,6,7)$$

III. Réalisation d´une fonction booléenne:

❑ *Ecriture canoniques, POS: on fait la sélection des zéros*

X	Y	Z	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$F(X, Y, Z) = S_0 \cdot S_1 \cdot S_2 \cdot S_4$$

$$F(X,Y,Z) = (\mathbf{X} + \mathbf{Y} + \mathbf{Z})(\bar{Z} + \mathbf{Y} + \mathbf{X})(\bar{Y} + \mathbf{X} + Z)(\mathbf{Z} + \mathbf{Y} + \bar{X})$$

III. Réalisation d´une fonction booléenne

❑ *Soucis majeurs des concepteurs*

- Réduire le nombre de portes nécessaires à la réalisation des systèmes
  - ✓ Minimiser le coût en nombre de boîtiers
  - ✓ La consommation électrique
- Minimiser la complexité
  - ✓ Créer un système équivalent avec certains paramètres optimisés
- Recherche d'équivalence
  - ✓ Utiliser les lois et théorèmes de l'algèbre de Boole

IV. Simplification des fonctions logiques par la méthode algébrique

1. Les postulats

Pour tout a, b, c ε E2, les propriétés suivantes sont vérifiées :

❑ Sur plusieurs variables

- **0 est l'élément minimum, 1 est l'élément maximum**
- **a.1 = a car min (a,1) = a**
- **a+0 = a car max (a,0) = a**
- **a.0 = 0**
- **a+1 = 1**

- **complément :**
- **a. /a = 0 car min (0,1) = 0**
- **a+ /a = 1 car max (0,1) = 1**

IV. Simplification des fonctions logiques par la méthode algébrique

1. Les postulats

❑ Sur une seule variable

Opérateur OU	Opérateur ET
$A+A = A$	$A.A = A$
$A+1 = 1$	$A.1 = A$
$A+0 = A$	$A.0 = 0$
$A+\overline{A} = 1$	$A.\overline{A} = 0$
Elément neutre = 0	Elément neutre = 1

IV. Simplification des fonctions logiques par la méthode algébrique

1. Les postulats

❑ Sur plusieurs variables

Pour tout a, b, c ε E2, les propriétés suivantes sont vérifiées :

- **Commutativité:** $A+B = B+A$

  $A.B = B.A$

- **Associativité :** $(A+B)+C = A+(B+C) = A+B+C$

  $(A.B).C = A.(B.C) = A.B.C$

- **Distributivité :** $A.(B+C) = A.B + A.C$

  $A+B.C = (A+B).(A+C)$

IV. Simplification des fonctions logiques par la méthode algébrique

2. Les Théorèmes:

❑ Théorème de Morgan:

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

❑ Divers:

- A+A.B = A

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

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

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

IV. Simplification des fonctions logiques par la méthode algébrique

Exercice: Simplifier les fonction suivante et représenter les avec des portes logique

1) F1= A(A+B)

2) $F2=(A+B)(\overline{A}+b)$

3) $F3 = AB + \overline{C} + C(\overline{A} + \overline{B})$

4) $F4 = (X\overline{Y} + Z)(X + \overline{Y})Z$

5) $F5=(X+Y)Z+\overline{X}(\overline{Y}+Z)+\overline{Y}$

V. Simplification des fonctions logiques par la méthode karnaugh

- Cette méthode de simplification consiste à mettre en évidence graphiquement les groupements de termes produits qui ne diffèrent que par l'état d'une seule variable d'entrée (termes adjacents).

- Un tableau de Karnaugh est une autre forme de la table de vérité. Il est organisé en colonnes et lignes dont les intersections donnent une case qui représente un des termes produits de la fonction. Pour une fonction de *n variables, le tableau* Comportera 2*n* cases. On écrit dans chaque case la valeur correspondante de la fonction : 0 ou 1, et si cette valeur est indéterminée, Ø ou X.

Exemples des tableaux de Karnaugh pour 2, 3, 4 variables

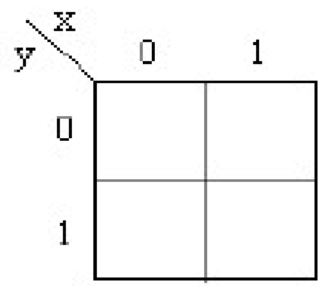


Tableau à 2 variables

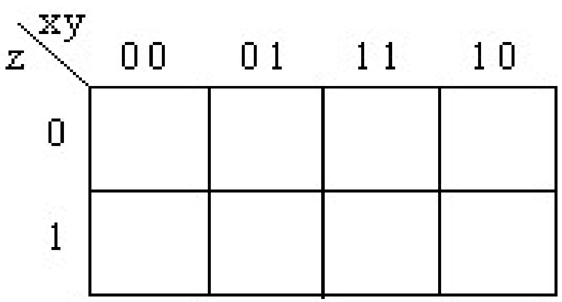


Tableau à 3 variables

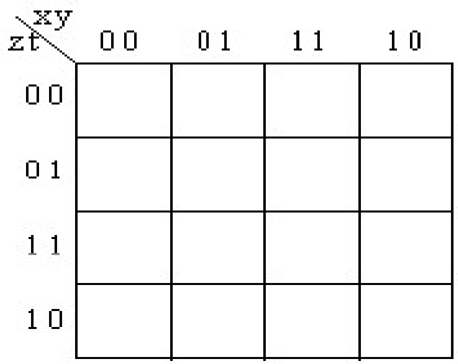
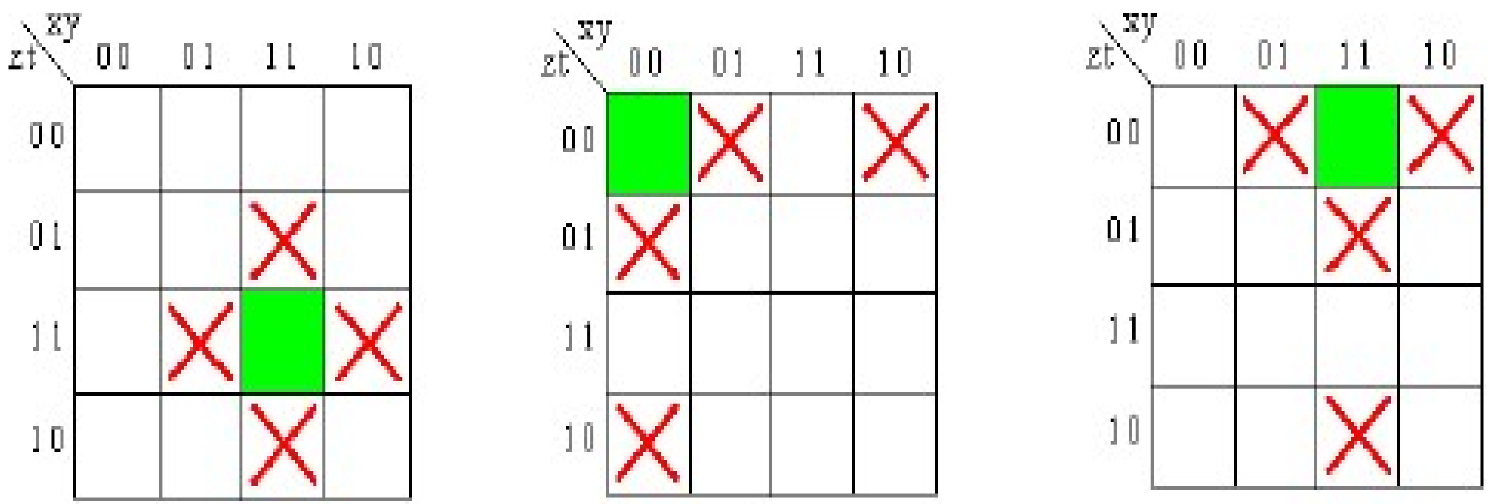


Tableau à 4 variables

V. Simplification des fonctions logiques par la méthode karnaugh

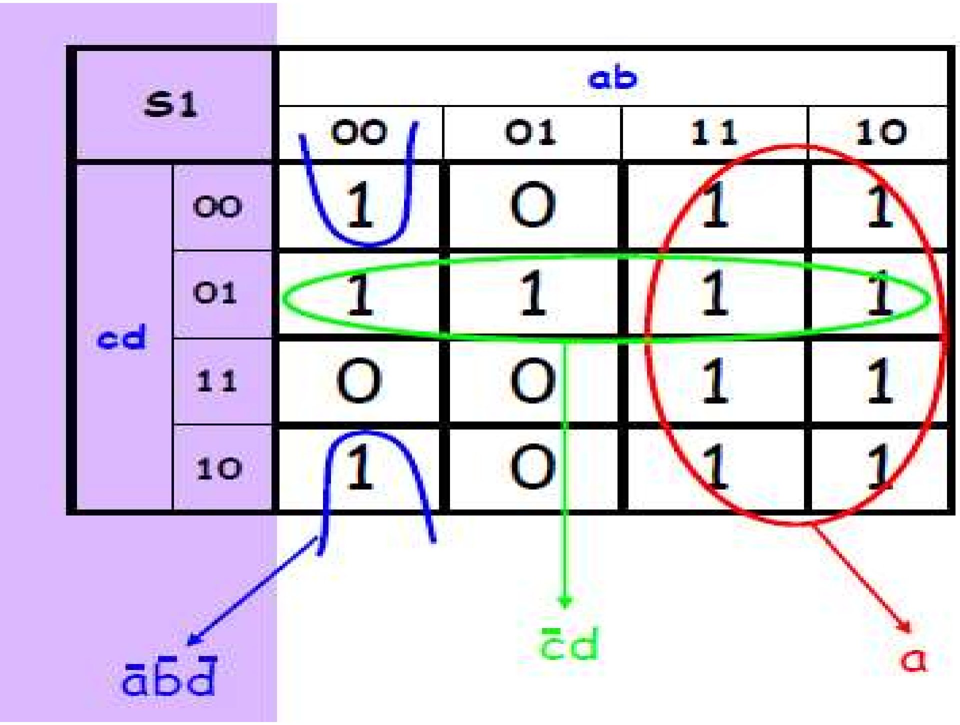
Le tableau de Karnaugh consiste à rassembler les cases adjacentes contenant des 1 par groupes de 2, 4 ou 8 termes.

Cases adjacentes



V. Simplification des fonctions logiques par la méthode karnaugh

- Dans un groupement de deux termes on élimine donc la (1) variable qui change d'état et on conserve le produit des variables qui ne changent pas.
- Dans un groupement de quatre on élimine les deux variables qui changent d'état.
- Dans un groupement de huit on élimine trois variables.



V. Simplification des fonctions logiques par la méthode karnaugh

- **Passage de la table de vérité au tableau de Karnaugh :** Il suffit de reporter dans chaque case du tableau de Karnaugh la valeur de la variable de sortie correspondant à chaque combinaison des variables d'entrée

	a	b	c	S1	S2
0	0	0	0	1	1
1	0	0	1	1	1
2	0	1	0	0	1
3	0	1	1	0	1
4	1	0	0	0	0
5	1	0	1	1	0
6	1	1	0	0	0
7	1	1	1	0	1

S1		ab			
		00	01	11	10
c	0	1	0	0	0
	1	1	0	0	1

$$S1 = \overline{a} \cdot \overline{b} + \overline{b} \cdot c = \overline{b}(\overline{a} + c)$$

S2		ab			
		00	01	11	10
c	0	1	1	0	0
	1	1	1	1	0

$$S2 = \overline{a} + b \cdot c$$

VI. Fonction combinatoire avancées

La logique combinatoire concerne l'étude des fonctions dont la valeur de sortie ne dépend que de l'état logique des entrées se traduisant par une modification de la valeur des sorties et non pas non plus de ses états antérieurs : à chaque combinaison des variables d'entrée correspond toujours une seule combinaison des fonctions de sortie.

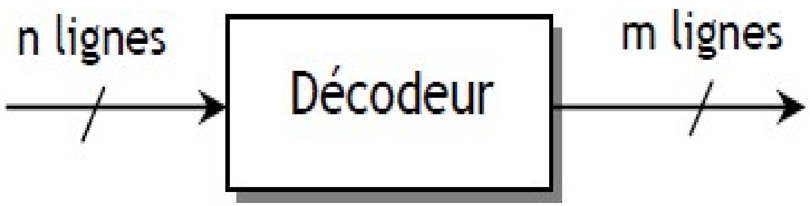
On note en particulier les décodeurs, les multiplexeurs, les démultiplexeurs et les circuits arithmétiques. Bien qu'ils soient plus ou moins remplacés actuellement par les systèmes programmables (circuits logiques programmables et microprocesseur), ils sont encore utilisés.

IV. Fonction combinatoire avancées

1. Décodeur

La fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes, un autre code en sortie sur m lignes avec en général m # n :

Un décodeur est un circuit logique combinatoire qui a une entrée binaire de n bits permettant 2n combinaisons et M sorties telles que 2n ≥M



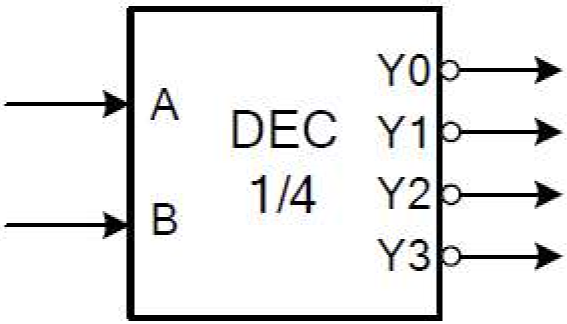
IV. Fonction combinatoire avancées

1. Décodeur

a. Décodeur un parmi

Ce type de décodeur permet de faire correspondre à un code présent en entrée sur n lignes une sortie et une seule active parmi les N = 2n sorties possibles. On le désigne aussi par décodeur m lignes vers n lignes.

Pour comprendre le principe d'un tel décodeur, étudions le décodeur 1 parmi 4 ou 2 vers 4, donné à la figure 1 ; le niveau active des sorties est le 0, car c'est souvent le cas :



ENTREES		SORTIES			
B	A	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

IV. Fonction combinatoire avancées

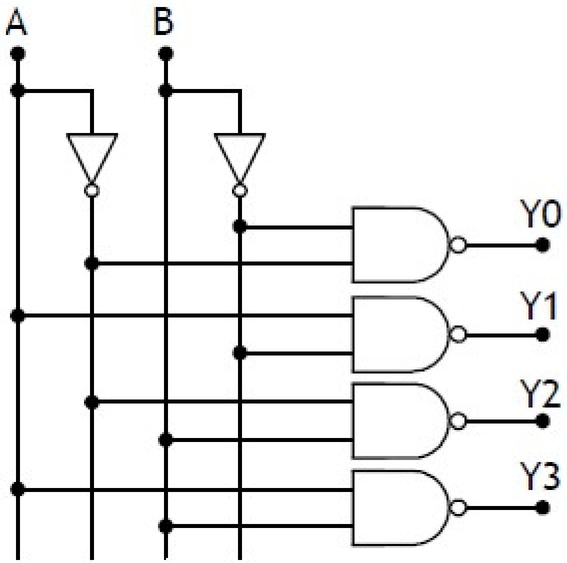
1. Décodeur

a. Décodeur un parmi

Directement on détermine les équations de sortie:

$$\overline{Y_0} = \overline{B}.\overline{A}$$
$$\overline{Y_1} = \overline{B}.A$$
$$\overline{Y_2} = B.\overline{A}$$
$$\overline{Y_3} = A.B$$

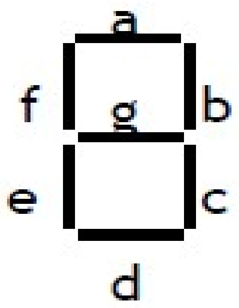
Le logigramme du décodeur est:



IV. Fonction combinatoire avancées
1. Décodeur
b. Décodeur 7 segments

Ce type de décodeur permet de convertir le code BCD 4bits à l'entrée pour obtenir à la sortie un code 7 segments permettant de commander un afficheur 7 segments permettant l'écriture de tous les chiffres et aussi d'autres symboles:

Afficheur 7 segments



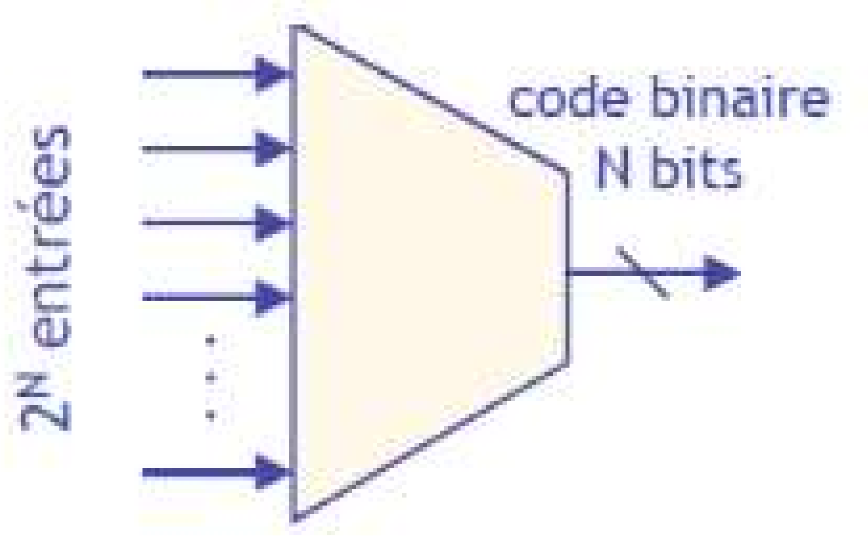
En dressant la table de vérité de ce décodeur, et en utilisant la table de karnaugh pour simplifier chaque sortie on peux déterminer l’expression des sortie a,b,c,d,e,f,g. (voir TD)

exemple de ce décodeur on peut citer le circuit intégré TTL 7447/ 7448

IV. Fonction combinatoire avancées

1. Codeur

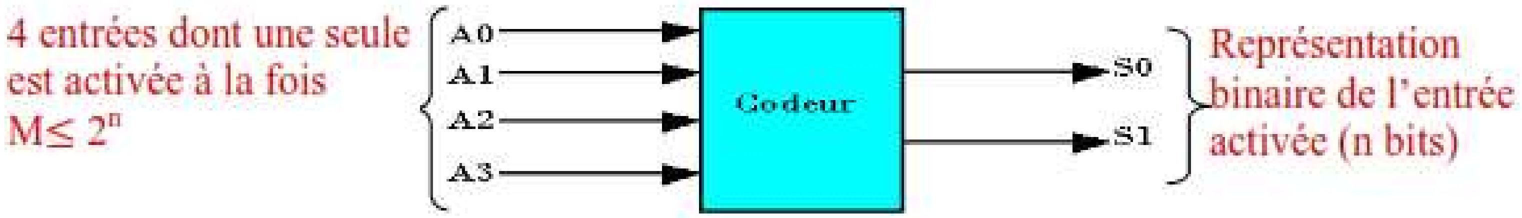
Un codeur est un circuit logique qui possède 2´n voies entrées , dont une seule est activée et N voies sorties. Il fournit en sortie le code binaire correspondant



IV. Fonction combinatoire avancées

1. Codeur

Pour comprendre le principe d'un tel **codeur**, étudions le codeur 4 vers 2 qui est un codeur a 4 voies entrées et 2 vois de sorties, donné à la figure 2 ; le niveau active des sorties est le 1:



Entées				Sorties	
Codage 1 parmi 2^n				Nombre binaire de n bits	
A_3	A_2	A_1	A_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

IV. Fonction combinatoire avancées

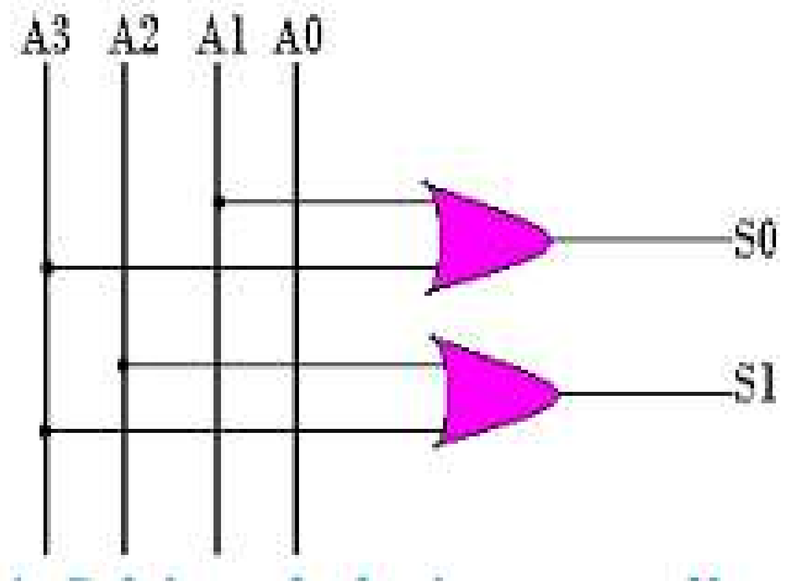
1. Codeur

Les équations des sorties:

S_1=1 si (A_2= 1) ou (A_3=1) ; S1= A_2+A_3

S_2=1 si (A_3= 1) ou (A_1=1) ; S1= A_3+A_1

Logigramme:

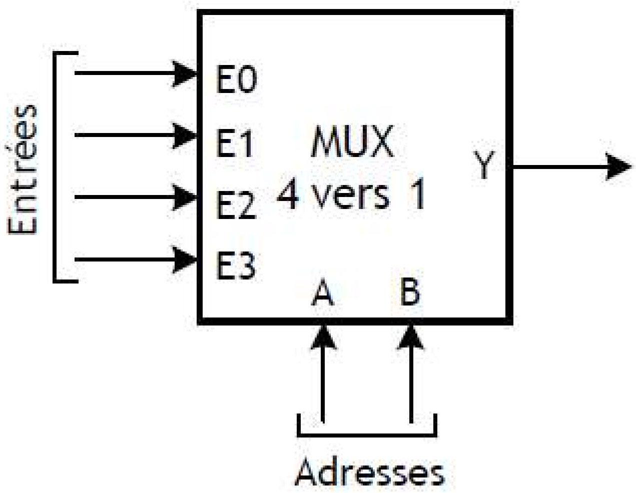


IV. Fonction combinatoire avancées

2. Multiplexeur

Un multiplexeur permet de sélectionner une entrée parmi 2n pour transmettre l'information portée par cette ligne à un seul canal de sortie. La sélection de l'entrée se fait alors à l'aide de n lignes d'adressage.

Pour comprendre le principe, considérons un multiplexeur à quatre entrées , donc deux lignes d'adressage et une ligne de sortie :



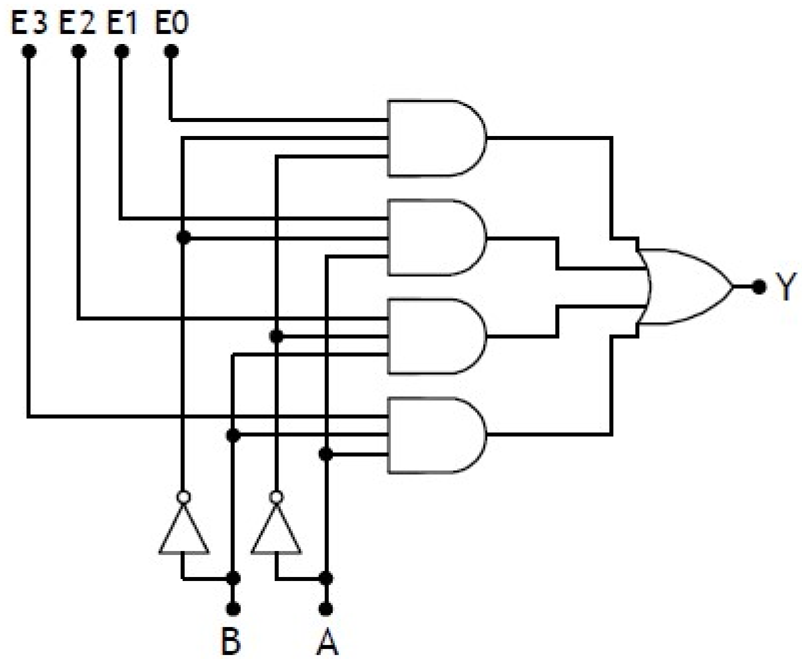
ADRESSES		SORTIE
B	A	Y
0	0	E0
0	1	E1
1	0	E2
1	1	E3

IV. Fonction combinatoire avancées

2. Multiplexeur

on déduit l'expression logique de la sortie Y : $Y = \overline{A}.\overline{B}.E0 + A.\overline{B}.E1 + \overline{A}.B.E2 + A.B.E3$

Le schéma d'implantation du multiplexeur 4 vers 1 sera

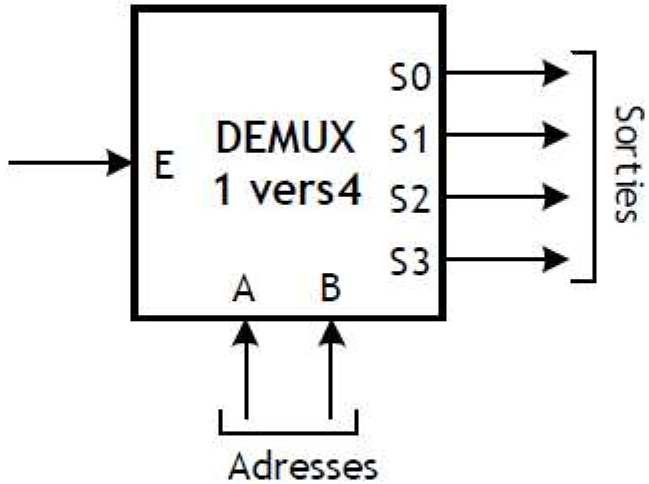


IV. Fonction combinatoire avancées

3. Démultiplexeur

Le démultiplexeur effectue l'opération inverse d'un multiplexeur à savoir il permet de distribuer l'information présente à l'entrée vers l'une des 2n sorties. La sélection de la sortie se fait à l'aide de n lignes d'adressage.

Pour comprendre le principe, considérons un démultiplexeur à quatre sorties, donc deux lignes d'adressage et une ligne d'entrée :



ADRESSES		SORTIES			
B	A	S0	S1	S2	S3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

IV. Fonction combinatoire avancées

3. Démultiplexeur

A partir de la table de vérité, on détermine les équations de sortie suivantes :

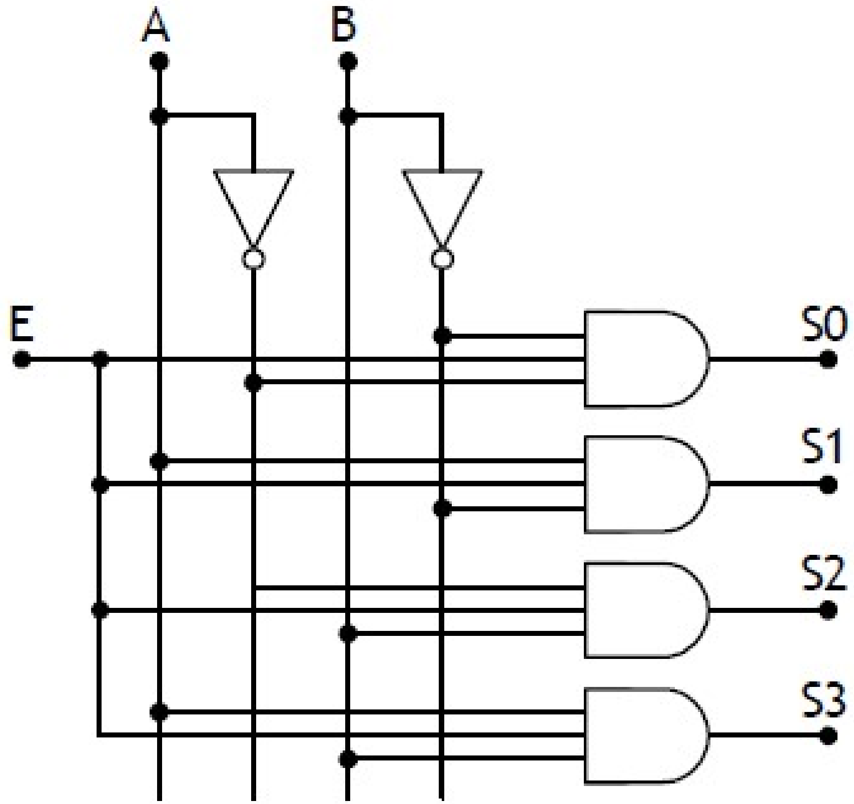
$$S_0 = E.\overline{B}.\overline{A}$$

$$S_1 = E.\overline{B}.A$$

$$S_2 = E.B.\overline{A}$$

$$S_3 = E.A.B$$

Equations

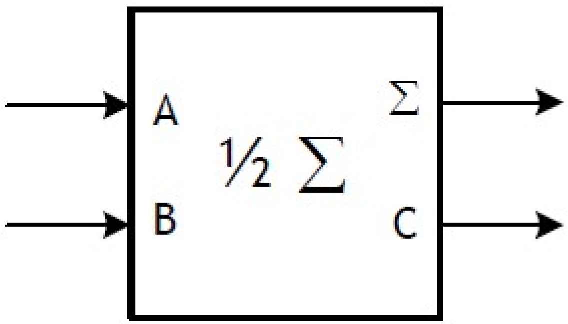


Logigramme

IV. Fonction combinatoire avancées

4. Demi-Additionneur

C'est un circuit permettant d'effectuer l'addition de deux bits A et B pour générer leur somme ∑ et leur retenue C (Carry) comme le montre le schéma et la table de vérité de la figure:



ENTREES		SORTIES	
B	A	Σ	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

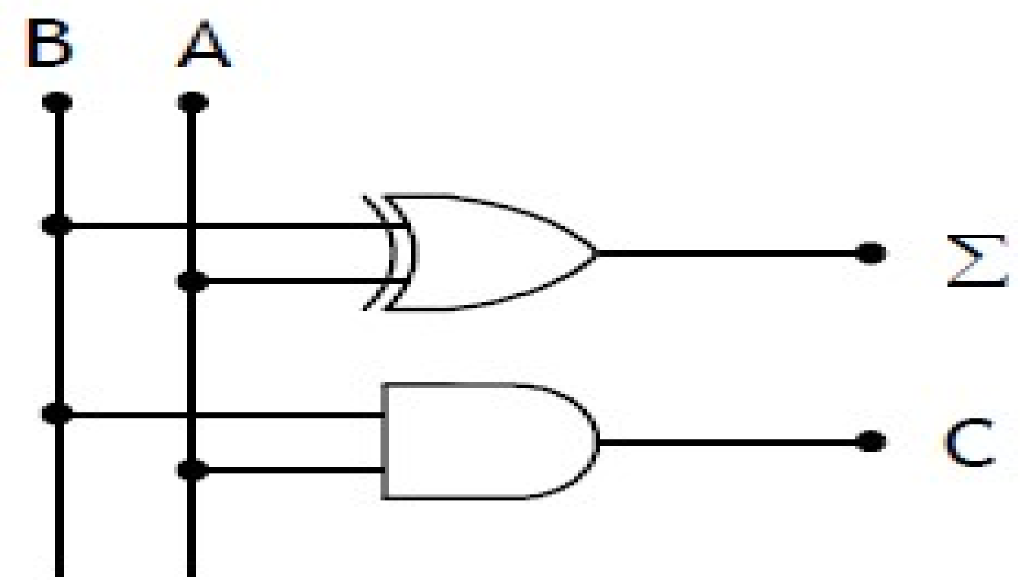
IV. Fonction combinatoire avancées

4. Demi-Additionneur

A partir de la table de vérité, on peut écrire les deux fonctions sous la forme suivante :

$$\Sigma = A.\overline{B} + \overline{A}.B = A \oplus B \qquad\qquad C = A.B$$

Equations

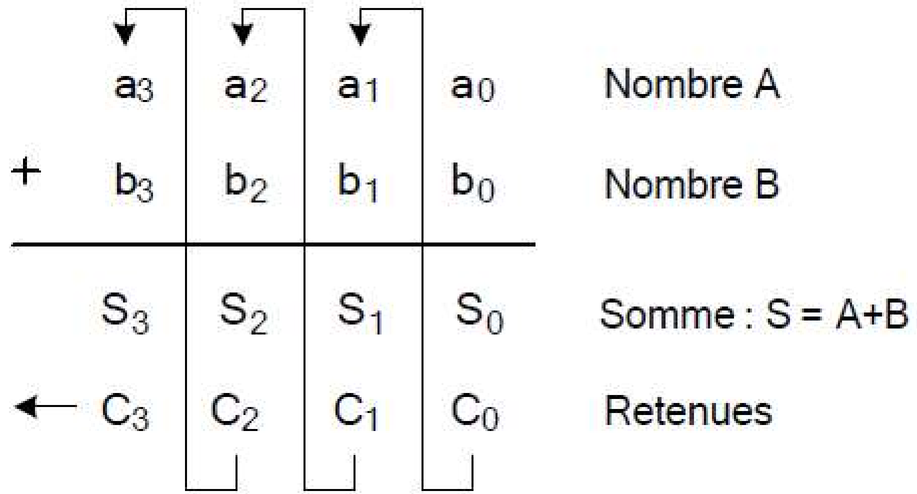


Logigramme

IV. Fonction combinatoire avancées

5. Additionneur-Complet

Pour effectuer une addition de deux nombres binaires de n bits, on additionne successivement les bits du même poids en tenant compte de la retenue de l'addition précédente comme le montre l'exemple suivant :



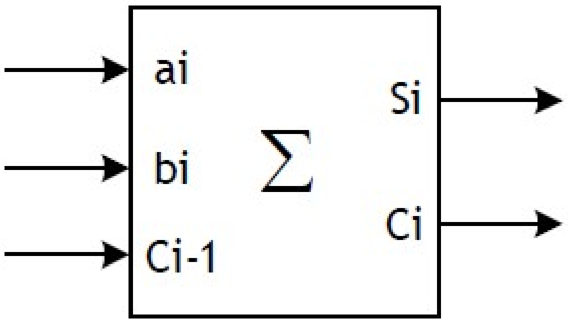
PROBLEME: le Demi-Additionneur ne tien pas compte de la retenue de l'addition précédente c'est-à-dire du rang inférieure

IV. Fonction combinatoire avancées

5. Complet-Additionneur

SOLUTION: Il faut donc concevoir une cellule élémentaire appelée additionneur complet et qui permet de réaliser l'addition des bits ai et bi en plus de la retenue Ci-1 de l'addition précédente.

De tel circuit est défini par le schéma et la table de vérité suivante:



ENTREES			SORTIES	
ai	bi	Ci-1	Si	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

IV. Fonction combinatoire avancées

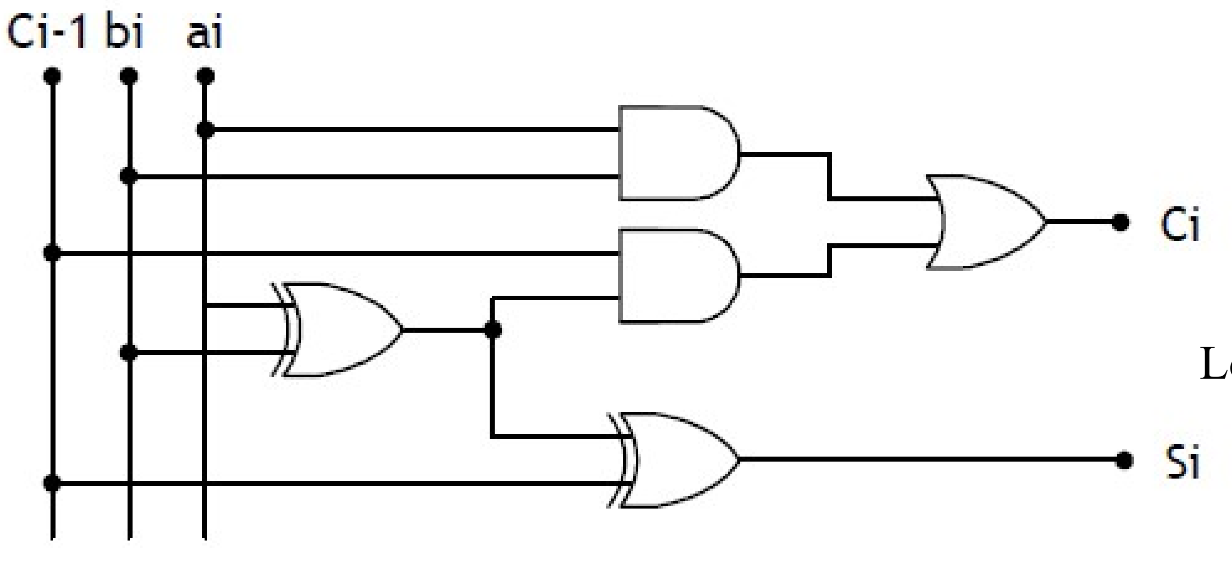
5. Complet-Additionneur

Equations

$$S_i = a_i.\overline{b_i}.\overline{C_{i-1}} + \overline{a_i}.b_i.\overline{C_{i-1}} + \overline{a_i}.\overline{b_i}.C_{i-1} + a_i.b_i.C_{i-1} = a_i \oplus b_i \oplus C_{i-1}$$

$$C_i = a_i.b_i + a_i.\overline{b_i}.C_{i-1} + \overline{a_i}.b_i.C_{i-1} = a_i.b_i + (a_i \oplus b_i).C_{i-1}$$

La retenue

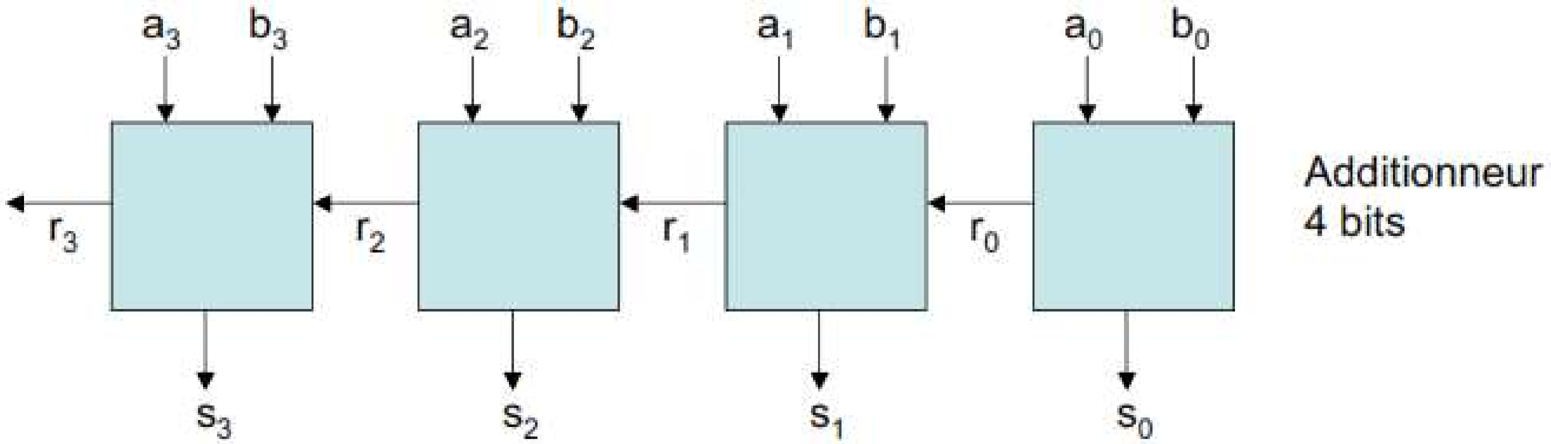


Logigramme

IV. Fonction combinatoire avancées

5. Additionneur sur n bit

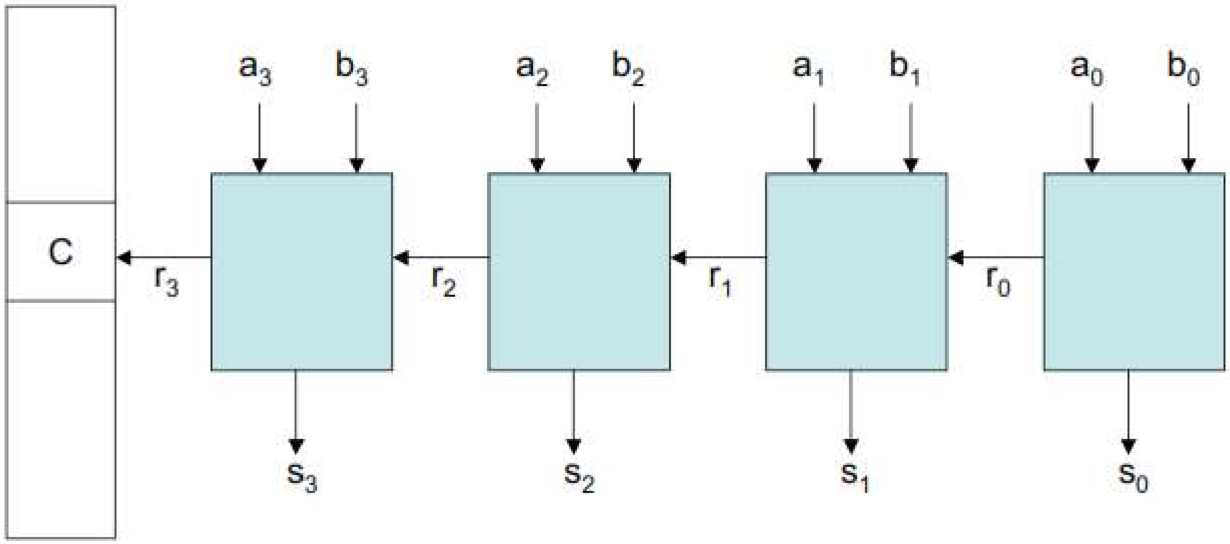
- L'additionneur n bits est obtenu en chaînant entre eux un demi-additionneur et n-1 additionneurs 1 bit complets
- Le chaînage s'effectue par le biais des retenues propagées



IV. Fonction combinatoire avancées

5. Indicateur de carry

- Lors d'une opération arithmétique effectuée sur des nombres de n bits un n+1e bit, un bit de carry peut être généré
- Ce bit de carry mémorisé par l'indicateur C du registre d'état du processeur, correspond au niveau de l'additionneur n bits, à une retenue rn-1 égale à 1pour l'additionneur complet 1 bit



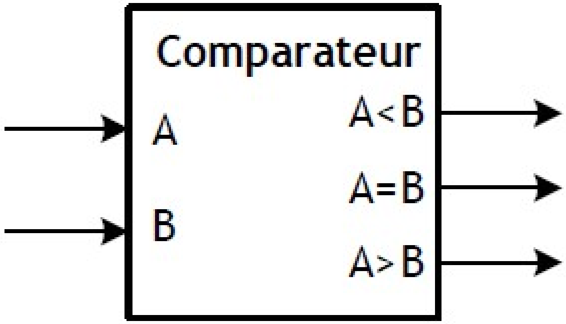
Registre d´etat

IV. Fonction combinatoire avancées

6. Comparateur

Un comparateur est un circuit permettant de détecter l'égalité de deux nombres et éventuellement d'indiquer le nombre le plus grand ou le plus petit.

Pour comprendre le principe, on va réaliser un comparateur simple permettant de comparer deux mots de 1 bit.



ENTREES		SORTIES		
B	A	S1:A<B	S2:A=B	S3:A>B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

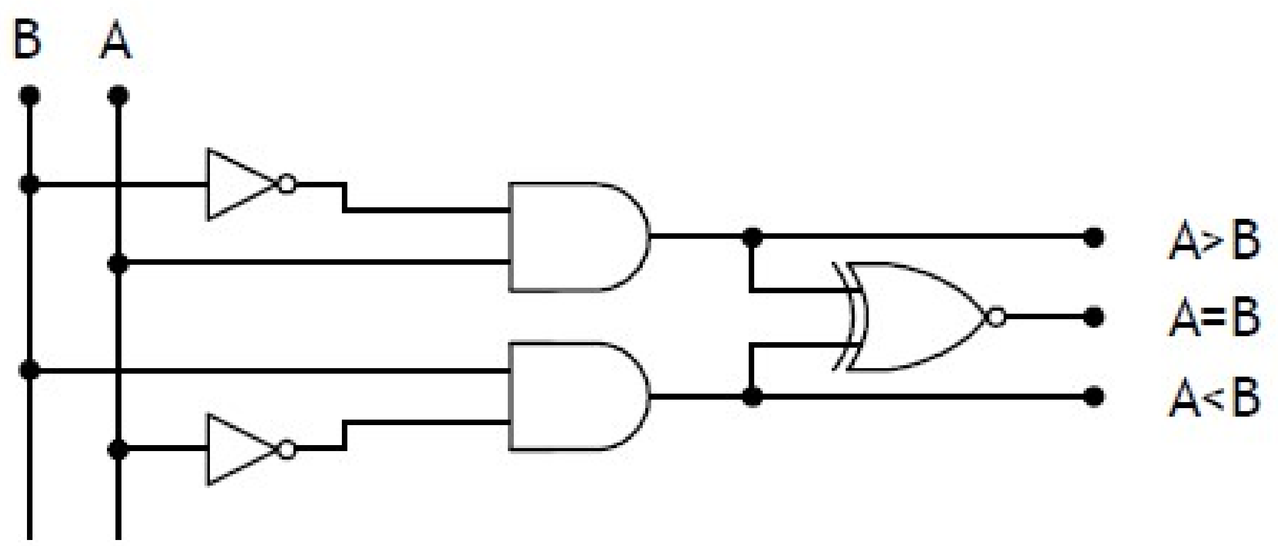
Table de vérité

IV. Fonction combinatoire avancées

6. Comparateur

$$S1 = \overline{A}.B \qquad S2 = A.\overline{B} \qquad S3 = \overline{S1 \oplus S3}$$

Equations



Logigramme