REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Saïda – Dr. Moulay Tahar

Faculté de Technologie

Département d'Electronique

Polycopié Pédagogique

- Intitulé :

  Logique combinatoire et séquentielle

- Présenté par :

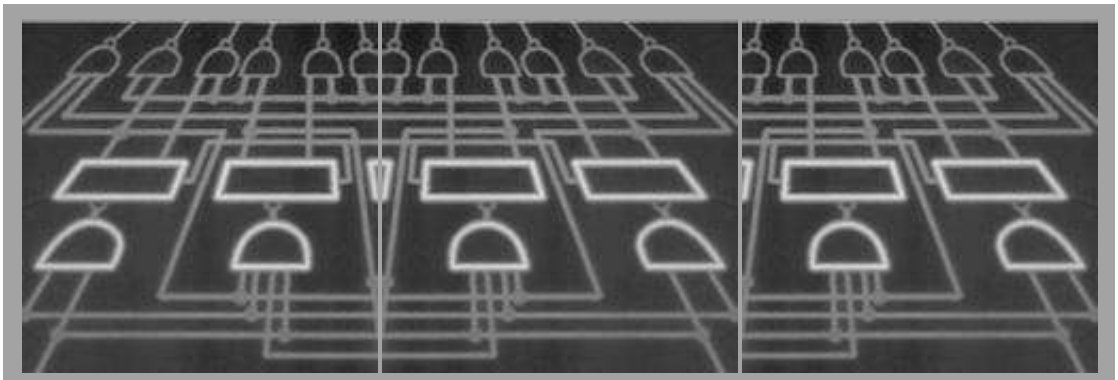
Dr. BOUDKHIL Abdelhakim

Maître de Conférence B (MCB)

- Pour les étudiants de la 2ème année Licence

  Electronique, Télécommunications et Génie Biomédical

  Semestre 4



Année universitaire : 2021-2022

PREFACE

Le cours "Logique combinatoire et séquentielle" est destiné aux étudiants de la deuxième année Licence du déparetement d'Electronique avec ses filières Electronique, Télécommunications et Génie Biomédical.

Cette matière permet à l'étudiant d'apprender les notions de base sur la synthèse et l'analyse des circuits logiques en utilisant des outils standarisés. A l'issue de cette matière, il devient capable à concevoir et réaliser les différents circuits logiques combinatoires et séquentiels utilisés aujourd'hui ce qui lui servira par la suite dans son domaine de spécialité (commande de processus, systèmes de communication, contrôle industriel, etc.), à sa voir la table de vérité, le tableau de Karnaugh et les différentes mémoires comme les bascules, les compteurs et les registres.

L'étudiant à travers ce support va aborder :

- au premier temps à travers les deux premiers chapitres, les principe de l'Algèbre de Bool et les différentes opérations logiques et leur simplification en utilisant l'algèbre boooléènne. Aprés, les bases des systèmes de numération y cmporis le système binaire ainsi que le codage de l'information.

- Ensuite étape par étape, à travers les chapitres qui suivent, les circuits logiques combinatoires en premier ordre tels que : additionneur, soustracteur, comparateur, encodeur, décodeur, multiplexeur, démultiplexeur… et par la suite les circuits logiques séquentiels tels que : bascules, compteur et registres jusqu'à qu'il devient capable de réaliser tout circuit combinatoire ou séquentiel relatif à son domaine de spécialité.

Aucune conaissance préalable n'est indispensable pour entamer cette matière. L'étudiant entamera étape par étape toutes les notions théoriques constituant le support faondamental pour la conception des systèmes digitaux de nos jours.

Table des Matières

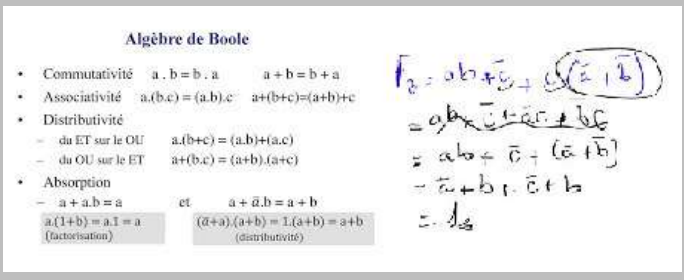
Liste des Figures

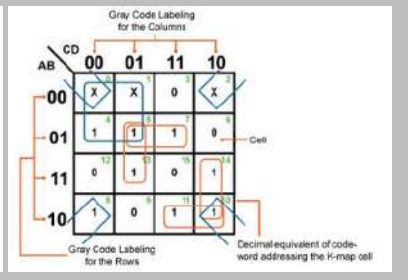
Liste des Tables

Chapitre 1

Algèbre de Boole et simplification des fonctions logiques

- **Objectifs du chapitre :**
  - Savoir simplifier une fonction logique par la méthode algébrique utilisant les postulas et théorèmes de l'Algèbre booléenne) ou par la méthode graphique de Karnaugh.
- **Dans ce chapitre :**
  - ✓ Bases de l'Algèbre de Boole
  - ✓ Fonctions logiques binaires
  - ✓ Présentations des fonctions logiques
  - ✓ Simplification de fonctions logiques





1.1. Algèbre de Boole « algèbre logique »

La logique binaire [1-2] consiste en des variables binaires qui ne peuvent prendre que deux valeurs possibles et des opérations binaires élémentaires fondées à la base de l'Algèbre de Boole qui constitue l'outil théorique pour étudier le système logique binaire dont les bases sont établies par George Boole.

En effet, pour décrire, analyser et concevoir des circuits digitaux logiques, il est d'abord nécessaire de se familiariser avec les mathématiques logique de base relative à ce sujet. C'est en 1854 que Boole introduisit une notion symbolique pour présenter des situations qui ne pouvaient être que vraies ou fausses. En 1938 Claude Shannon a utilisé cette notion pour analyser des fonctions logiques connue depuis sous le nom de l'Algèbre Booléienne ou logique.

De point de vu logique général, une proposition logique « vrai / faux » pour les variables de système logique étant essentiellement binaire due à la correspondance de sa véracité au chiffre binaire 1 et de sa fausseté au chiffre binaire 0.

L'algèbre de Boole, comme tout autre système mathématique inductif peut être définie comme un ensemble d'éléments et d'opérations, un nombre d'axiomes et de postulas crée pour traiter les fonctions logiques, dont l'algèbre binaire constitue un cas particulier de l'algèbre booléienne.

L'algèbre de Boole est une structure algébrique définit sur un nombre d'éléments E muni deux lois de composition ; addition booléenne (+, $\bigcup$, Ou), produit booléen (., $\bigcap$, Et) et d'une loi de complémentation ($\overline{....}$, Non) basée sur le postula qu'il existe au moins deux éléments distincts dans E ce qui est le cas particulier de l'algèbre binaire où $E = \{0, 1\}$, dont 1 et 0 sont compléments uniques $\bar{1} = 0$. Les postulas de Huntington et les théorèmes les plus communément utilisés pour formuler la structure algébrique du système logique, sont donnés par :

Fermeture	$\forall x, y \in E : \begin{cases} x+y \in E \\ x.y \in E \end{cases}$
Commutativité	$\forall x, y \in E : \begin{cases} x+y = y+x \\ x.y = y.x \end{cases}$
Associativité	$\forall x, y, z \in E : \begin{cases} x+(y+z) = (x+y)+z \\ x.(y.z) = (x.y).z \end{cases}$
Distributivité	$\forall x, y, z \in E : \begin{cases} x+(y.z) = (x+y).(x+z) \\ x.(y+z) = (x.y)+(x.z) \end{cases}$
Elément neutre	Pour l'addition logique, le 0 est l'élément neutre : $x+0 = x$ Pour la multiplication logique, le 1 est l'élément neutre : $x*1 = x$
Complément	$\forall x \in E : \begin{cases} x+\bar{x} = 1 \\ x.\bar{x} = 0 \end{cases}$

Table 1.1 - Postulas de l'algèbre booléenne

Idem	$\forall x \in E : \begin{cases} x+x = x \\ x.x = x \end{cases}$
Double négation « Involution »	$\forall x \in E : \bar{\bar{x}} = x$
Loi de Morgan	$\forall x, y \in E : \begin{cases} \overline{x+y} = \bar{x}.\bar{y} \\ \overline{x.y} = \bar{x}+\bar{y} \end{cases}$
Absorption I	$\forall x, y \in E : \begin{cases} x+xy = x \\ x.(x+y) = x \end{cases}$
Absorption II	$\forall x, y \in E : \begin{cases} x+\bar{x}.y = x+y \\ x.(\bar{x}+y) = x.y \end{cases}$
Consensus	$\forall x, y, z \in E : \begin{cases} x.y+\bar{x}.z+y.z = x.y+\bar{x}.z \\ (x+y).(\bar{x}+z).(y+z) = (x+y).(\bar{x}+z) \end{cases}$

Table 1.2 - Théorèmes fondamentaux de l'algèbre booléenne

1.2. Variables logiques binaires

Une variable logique [3] est une variable qui ne peut prendre que deux valeurs conventionnellement repérées par 0 et 1. On parle ici de variable binaire ont chacune de ces deux valeurs est associée à une grandeur physique, par exemple la tension collecteur d'un transistor, ce qui permet de faire le lien entre une étude théorique utilisant l'algèbre de Boole et un circuit électronique.

Dans la logique positive, la variable 0 sera associée à un niveau bas (typiquement une tension nulle). La variable 1 sera associée à un niveau haut (une tension positive de +5V par exemple dans le cas des circuits électroniques réalisés en technologie TTL1).

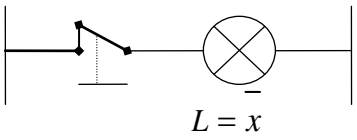
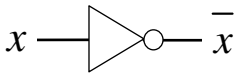
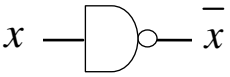
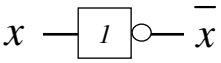
	Logique positive	Logique négative
0	Valeur algébrique minimale	Valeur algébrique maximale
1	Valeur algébrique maximale	Valeur algébrique minimale

Table 1.3 - Variables logiques binaires

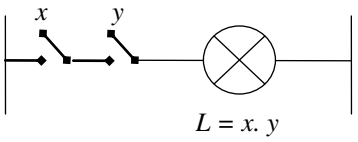
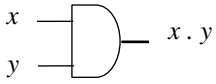
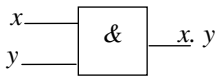
1.3. Fonctions logiques binaires

Un circuit logique est formé par l'interconnexion de plusieurs circuits élémentaires appelés « portes logiques » conçu pour réaliser une fonction désirée. Cette fonction est binaire du moment qu'elle ne peut prendre que deux valeurs 0 ou 1. Elle est en fonction d'une ou plusieurs variables binaires. Le circuit logique peut avoir une ou plusieurs entrées ou ue ou plusieurs fonctions de sortie. Une porte logique contient des éléments logiques élémentaires qui peuvent avoir plusieurs entrées et une seule sortie. Elle traite des signaux appliqués à ses entrées pour produire des signaux en sa sortie. La porte logique peut être appelée aussi par un circuit commutateur, digital ou logique. Il existe plusieurs portes logiques dont trois sont élémentaires :

Inversion logique : NON / NOT

Définition	Table de vérité	Schéma électrique	Symbole graphique
La seule opération qu'on peut effectuer sur une seule variable binaire. C'est une fonction complémentaire d'inversion ou de négation.	x | $\bar{x}$ 0 | 1 1 | 0		Norme américaine  Norme européenne  Nouvelle norme 

Produit logique : ET / AND

Définition	Table de vérité	Schéma électrique	Symbole graphique
C'est la fonction d'intersection logique. Le résultat de l'operateur est vrai si les deux variables sont simultanément vraies.	x | y | $x.y$ 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1		Norme américaine  Norme européenne 

Addition logique : OU / OR

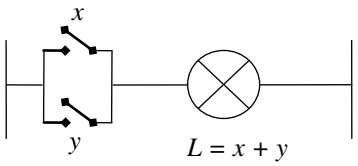
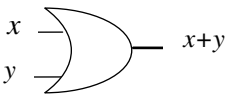
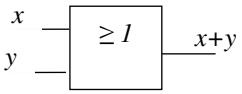
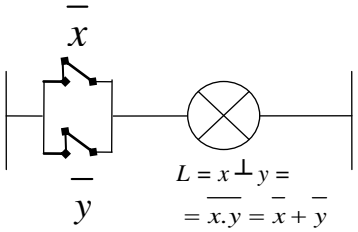
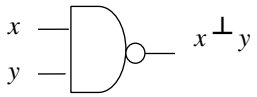
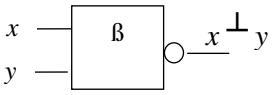
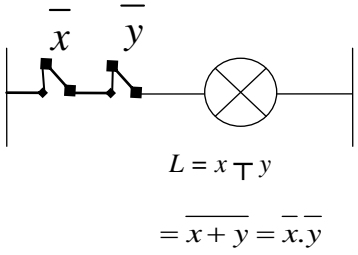
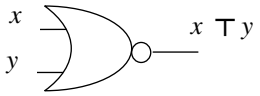
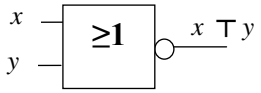
Définition	Table de vérité	Schéma électrique	Symbole graphique
C'est la fonction de réunion logique. Le résultat de l'operateur est faux si les deux variables sont simultanément fausses.	x | y | $x+y$ 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1		Norme européenne  Nouvelle norme 

Table 1.4 - Portes logiques de base

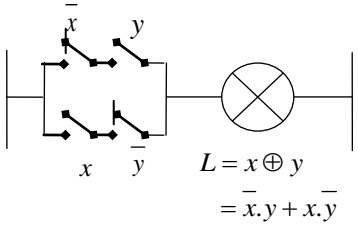
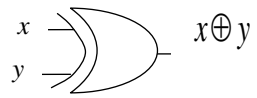
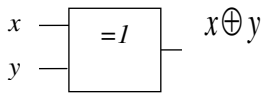
NON-ET / NAND

Définition	Table de vérité	Schéma électrique	Symbole graphique
Le résultat de l'operateur est faux si les deux variables sont simultanément vraies.	x y $x \perp y$ 0 0 1 0 1 1 1 0 1 1 1 0		Norme européenne  Nouvelle norme 

NON-OU / NOR

Définition	Table de vérité	Schéma électrique	Symbole graphique
Le résultat de l'operateur est vrai si les deux variables sont simultanément fausses.	x y $x \top y$ 0 0 1 0 1 0 1 0 0 1 1 0		Norme européenne  Nouvelle norme 

Comparateur de différence : OU-EXCLUSIF / XOR

Définition	Table de vérité	Schéma électrique	Symbole graphique
Le résultat de l'operateur est vrai si les deux variables sont différentes.	x y $x \oplus y$ 0 0 0 0 1 1 1 0 1 1 1 0		Norme européenne  ouvelle norme 

Equivalence : Complément du OU-EXCLUSIF / XOR

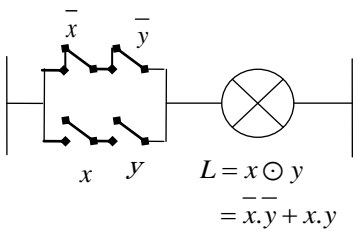
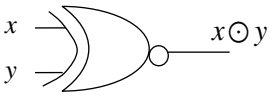
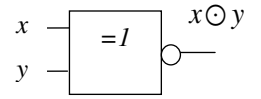
Définition	Table de vérité	Schéma électrique	Symbole graphique
Le résultat de l'operateur est vrai si les deux variables sont identiques.	x y $x \odot y$ 0 0 1 0 1 0 1 0 0 1 1 1		Norme européenne  Nouvelle norme 

Table 1.5 - Autres portes logiques

1.4. Présentations des fonctions logiques

Les fonctions logiques [2] sont représentées par leur écriture algébrique en utilisant les signes opératoires de l'algèbre logique. Elle peut être aussi représentée en utilisant :

1) Table de vérité

Les fonctions logiques peuvent être représentées sous forme de tables, appelées « tables de vérité » qui donnent la valeur de la fonction pour chaque combinaison des deux variables binaire 0 et 1 où il existe $2^2 = 4$ combinaisons possibles pour ces deux variables. La table de vérité de dimensions 2^n lignes et (n+1) colonne donne alors la valeur de sortie pour chacune de ces combinaisons.

L'écriture de la table de vérité fait partie de l'analyse d'un système (circuit) donné. A l'inverse une fois la table de vérité connue, il faut pouvoir déterminer le schéma électronique permettant de réaliser cette table : c'est la phase de synthèse.

- **Exemple**

$$F = x + \overline{y}.z$$

x	Y	$\overline{y}$	z	$\overline{y}.z$	F
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	1

2) Image d'une fonction logique

Une seule colonne reste très importante dans la table de vérité ; c'est la colonne où figure les valeurs de sorties de la fonction logique. Une fois

l'ordre des lignes fixe cette colonne, il suffit de caractériser la fonction par une suite de 2^n chiffres binaire. La colonne sera transposée en ligne où on obtient pour l'exemple précédent l'image de la fonction : ÷F = 01001111.

3) Expression numérique

La fonction logique peut être représentée à partir de son image par une expression numérique donnée par le symbole R qui signifie une réunion des nombres décimaux indiquant l'ordre des variables qui valent 1 : F = R(1, 4,5, 6, 7) et $\bar{F} = R(0, 2, 3)$.

4) Forme canonique

Il existe deux formes canoniques pour chaque fonction logique : forme disjonctive (réunion des produits) et forme conjonctive (intersection des réunions). Les deux formes sont basées respectivement sur la notion de minterme et maxterme qui est définie comme :

- ✓ Le minterme est un produit booléen de n littéraux. Le nombre total des mintermes pour n variable est égale à 2^n.
- ✓ Le maxterme est par dualité la somme booléenne de n littéraux. Le nombre total des maxtermes pour n variable est aussi égale à 2^n.

F (x ,y ,z)

x	*y*	*z*	*Minterme*	*Maxterme*
0	0	0	$m_0 = \bar{x}.\bar{y}.\bar{z}$	$M_0 = x+y+z$
0	0	1	$m_1 = \bar{x}.\bar{y}.z$	$M_1 = x+y+\bar{z}$
0	1	0	$m_2 = \bar{x}.y.\bar{z}$	$M_2 = x+\bar{y}+z$
0	1	1	$m_3 = \bar{x}.y.z$	$M_3 = x+\bar{y}+\bar{z}$
1	0	0	$m_4 = x.\bar{y}.\bar{z}$	$M_4 = \bar{x}+y+z$
1	0	1	$m_5 = x.\bar{y}.z$	$M_5 = \bar{x}+y+\bar{z}$
1	1	0	$m_6 = x.y.\bar{z}$	$M_6 = \bar{x}+\bar{y}+z$
1	1	1	$m_7 = x.y.z$	$M_7 = \bar{x}+\bar{y}+\bar{z}$

La fonction logique est égale à la somme des mintermes (produit des maxtermes) pour lesquels la fonction vaut 1. Pour l'exemple précédent, on tire :

Forme disjonctive → $F = \sum(1,4,5,6,7) = (\bar{x}.\bar{y}.z) + (x.\bar{y}.\bar{z}) + (x.\bar{y}.z) + (x.y.\bar{z}) + (x.y.z)$

Forme conjonctive → $F = \prod(0,2,3) = (x+y+z).(x+\bar{y}+z).(x+\bar{y}+\bar{z})$

À l'aide d'une double négation on peut aussi obtenir la forme conjonctive à partir de la forme disjonctive : $F_{Conjctive} = \bar{\bar{F}}_{Disjonctive}$

5) Matrice de combinaison « table de Karnaugh »

La table de Karnaugh est une table à double entrée dont les variables logiques seront partagées en deux groupes. Dans chaque groupe les combinaisons de variables se succèdent selon le code binaire réfléchi ; en conséquence, deux cases voisines sur l'horizontale ou la verticale font des combinaisons adjacentes. L'équivalent décimal de chaque combinaison et le nombre de la case correspondant toujours au code binaire naturel. Pour représenter la fonction logique en utilisant la table de Karnaugh, il suffit de placer chaque valeur dans la case correspondante à la combinaison considérée :

F (x, y, z)

x \ yz	00	01	11	10
0	0 (0)	1 (1)	0 (3)	0 (2)
1	1 (4)	1 (5)	1 (7)	1 (6)

6) Logigramme (schéma logique)

Une fonction logique peut être transformée de l'expression algébrique en un diagramme logique appelé « logigramme » composé de portes logiques élémentaires. L'exemple étudié peut être présenté par le logigramme suivant qui est construit par les portes NOT, AND et OR :

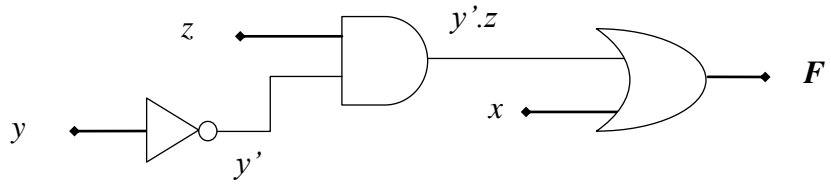


Figure 1.1 – Exemple d'un logigramme (schéma logique)

7) Schéma électrique

Une fonction logique peut être représentée par un schéma électrique équivalent et réalisée par un circuit électrique où les variables Non barrées sont données par des boutons poussoirs ouverts ou en repos et les variables barrées sont données par des boutons poussoirs fermés ou hors repos. Les boutons sont connectés en série en cas de produit entre les variables ou en parallèle en cas de sommation. La sortie de la fonction correspond à un témoin présenté généralement par une lampe qui s'éteint ou s'allume suivant chaque état de sortie. La fonction de l'exemple considéré est donnée par le schéma électrique suivant :

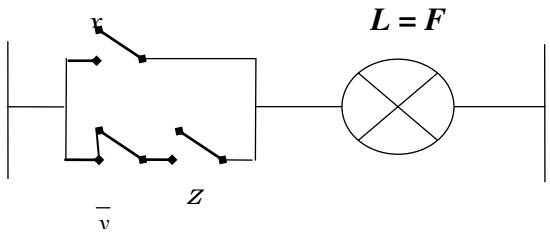


Figure 1.2 – Exemple d'un schéma électrique

1.5. Simplification des fonctions logiques

Une fonction logique doit être simplifiée pour procéder à sa réalisation en exploitant le minimum nombre possible de portes et d'interconnexions. La méthode de simplification permet donc de réduire le matériel logique utilisé pour obtenir des réalisations les moins couteuses possibles en minimum de temps. Plusieurs méthodes sont définies pour la simplification des fonctions logiques :

1) Méthode algébrique « méthode directe »

Cette méthode étant directe et basique consiste à appliquer les postulas et théorèmes de l'algèbre booléenne pour simplifier l'expression algébrique représentant la fonction logique. Cette méthode reste plus compliquée et donne des résultats rapides seulement aux cas de fonctions simples.

- **Exemple**

$$
\begin{aligned}
F(x,w,y,z) &= x.y + x.y.\overline{w} + x.y.z + x.\overline{z} \\
&= x.y + x.y.\overline{w} + x\underbrace{\left(y.z + \overline{z}\right)}_{T_1} \rightarrow \text{THM Absorption II sur le terme } T_1 \\
&= x.y + x.y.\overline{w} + x\left(y + \overline{z}\right) \\
&= \underbrace{x.y}_{T_2} + x.y.\overline{w} + \underbrace{x.y}_{T_3} + x.\overline{z} \rightarrow \text{THM Idem sur les termes } T_2 \text{ et } T_3 \\
&= x.y + x.y.\overline{w} + x.\overline{z} \\
&= x.y\underbrace{\left(1+\overline{w}\right)}_{\text{égale à } 1} + x.\overline{z}
\end{aligned}
\Rightarrow \boxed{F(x,w,y,z) = x.y + x.\overline{z}}
$$

2) Méthode de Karnaugh « méthode graphique »

La méthode de Karnaugh proposée par *Veitch* et modifiée par *Karnaugh*, est une méthode graphique manuelle très efficace pour les fonctions de deux à six variables. Elle offre une procédure simple de minimisation des fonctions logiques ; considérée comme une forme schématique de la table de vérité ou une extension du diagramme de *Venn*.

Cette méthode repose sur l'utilisation de la propriété d'adjacente des cases voisines de la table de Karnaugh qui est formée de carrés dont chaque carré représente un minterme (dont une seule variable change de valeur quand on passe de case à l'autre) ; donc toute fonction peut être représentée graphiquement sur la table en marquant les carrés correspondants aux mintermes présentant sa forme canonique disjonctive.

- ✓ **Table à deux variables :** Pour deux variables, il existe quatre mintermes. La table de Karnaugh consiste donc de quatre carrés (cases), un pour chaque minterme comme suit :

F (x, y)

y \ x	0	1
0	$m_0 = \bar{x}.\bar{y}$	$m_1 = x.\bar{y}$
1	$m_2 = \bar{x}.y$	$m_3 = x.y$

- **Exemple**

$$F(x,y) = x.y + \bar{x}.y + x.\bar{y}$$

y \ x	0	1
0	0	1
1	1	1

Le résultat est la somme des logiques mintermes correspondants à chques deux cases adjacentes en éliminant la variable qui change de valeur lors du passage d'une case à sa voisine. On tire donc :

$$F(x,y) = x.y + \bar{x}.y + x.\bar{y} = \boxed{y + x}$$

✓ **Table à trois variables :** Il existe huit mintermes pour trois variables ; la table de Karnaugh peut être schématisée comme suit :

$$F(x,y,z)$$

x \ yz	00	01	11	10
0	$m_0 = \bar{x}.\bar{y}.\bar{z}$ 0	$m_1 = \bar{x}.\bar{y}.z$ 1	$m_3 = \bar{x}.y.z$ 3	$m_2 = \bar{x}.y.\bar{z}$ 2
1	$m_4 = x.\bar{y}.\bar{z}$ 4	$m_5 = x.\bar{y}.z$ 5	$m_7 = x.y.z$ 7	$m_6 = x.y.\bar{z}$ 6

On rappelle que les mintermes ne sont pas rangés dans une séquence binaire ordonnée mais plutôt dans une autre séquence. La caractéristique de cette séquence est que, un seul bit change de valeur ce qui donne des carrés adjacents.

- **Exemple**

$$F(x,y,z) = \bar{x}.y.z + x.y.z + \bar{x}.y.\bar{z} + x.y.\bar{z}$$

x \ yz	00	01	11	10
0	0 0	0 1	1 3	1 2
1	0 4	0 5	1 7	1 6

Le résultat correspondant à la somme des mintermes dans les groupements contenants un 1 en éliminant les variables qui change de valeur. On tire donc :

$$F(x,y) = \bar{x}.y.z + x.y.z + \bar{x}.y.\bar{z} + x.y.\bar{z} = \boxed{y}$$

✓ **Table à quatre variables :** Il existe seize mintermes pour quatre variables et la table de Karnaugh est configurée comme suit :

F (w, x, y, z)

wx \ yz	00	01	11	10
00	$m_0 = \bar{x}.\bar{w}.\bar{y}.\bar{z}$ 0	$m_1 = \bar{x}.\bar{w}.\bar{y}.z$ 1	$m_3 = \bar{x}.\bar{w}.y.z$ 3	$m_2 = \bar{x}.\bar{w}.y.\bar{z}$ 2
01	$m_4 = x.\bar{w}.\bar{y}.\bar{z}$ 4	$m_5 = x.\bar{w}.\bar{y}.z$ 5	$m_7 = x.\bar{w}.y.z$ 7	$m_6 = x.\bar{w}.y.\bar{z}$ 6
11	$m_{12} = x.w.\bar{y}.\bar{z}$ 12	$m_{13} = x.w.\bar{y}.z$ 13	$m_{15} = x.w.y.z$ 15	$m_{14} = x.w.y.\bar{z}$ 14
10	$m_8 = \bar{x}.w.\bar{y}.\bar{z}$ 8	$m_9 = \bar{x}.w.\bar{y}.z$ 9	$m_{11} = \bar{x}.w.y.z$ 11	$m_{10} = \bar{x}.w.y.\bar{z}$ 10

L'indice du minterme d'un carré est obtenu en concaténant le numéro de la ligne avec celui de la colonne du carré. Par exemple, le binaire correspondant à la 3ème ligne 11, concaténé avec celui de la 2ème colonne 01 donne le binaire 1101 qui représente $13_{décimal}$. Par conséquent, le carré de l'intersection de la 3ème ligne et de la 2ème colonne représente le minterme m_{13}.

- **Exemple**

$$F(x,w,y,z) = \bar{x}.w.y.z + \bar{x}.\bar{w}.y.\bar{z} + \bar{x}.\bar{w}.\bar{y}.\bar{z} + x.\bar{w}.\bar{y}.\bar{z} + \bar{x}.w.\bar{y}.z + x.\bar{w}.y.\bar{z}$$

wx \ yz	00	01	11	10
00	1 0	0 1	0 3	1 2
01	1 4	0 5	0 7	1 6
11	0 12	0 13	0 15	0 14
10	0 8	1 9	1 11	0 10

La table à quatre variables peut être vue comme une boule avec les extrémités gauche et droite collées ensemble et celle du haut et du bas collées aussi ensemble. On tire :

$$F(x,y) = \bar{x}.w.y.z + \bar{x}.\bar{w}.y.\bar{z} + \bar{x}.\bar{w}.y.\bar{z} + x.\bar{w}.y.\bar{z} + \bar{x}.w.\bar{y}.w + x.\bar{w}.y.\bar{z} = \boxed{\overline{w.z} + w.z}$$

✓ **Table à cinq et six variables :** Les tables à plus de quatre variables ne sont pas simples à utiliser. Le nombre de carré devient excessivement grand. Il est égal à 32 pour cinq variables et 64 pour six variables, donc elles sont mois praticables.

F (v, w, x, y, z)

	v = 0				*v = 1*				
yz / wx	00	01	11	10	00	01	11	10	*yz / wx*
00	0	1	3	2	16	17	19	18	00
01	4	5	7	6	20	21	23	22	01
11	12	13	15	14	28	29	31	30	11
10	8	9	11	10	24	25	27	26	10

F (u, v, w, x, y, z)

		v = 0				*v = 1*				
	yz / wx	00	01	11	10	00	01	11	10	*yz / wx*
u = 0	00	0	1	3	2	16	17	19	18	00
	01	4	5	7	6	20	21	23	22	01
	11	12	13	15	14	28	29	31	30	11
	10	8	9	11	10	24	25	27	26	10
u = 1	00	32	33	35	34	48	49	51	50	00
	01	36	37	39	38	52	53	55	54	01
	11	44	45	47	46	60	61	63	62	11
	10	40	41	43	42	56	57	59	58	10
	yz / wx	00	01	11	10	00	01	11	10	*yz / wx*

✓ **Forme réduite conjonctive :** Si au lieu d'une réunion des produits, on désire obtenir une forme réduite conjonctive c'est-à-dire un produit des réunions, il suffit de chercher la forme réduite disjonctive de la fonction complémentaire : Pour cette forme on fait grouper les zéros (au lieu des 1) selon les mêmes règles (exemple précédent table à quatre variables).

$$F(x,w,y,z) = \bar{x}.w.y.z + \bar{x}.\bar{w}.y.\bar{z} + \bar{x}.\bar{w}.\bar{y}.\bar{z} + x.\bar{w}.\bar{y}.\bar{z} + \bar{x}.w.\bar{y}.z + x.\bar{w}.y.\bar{z}$$

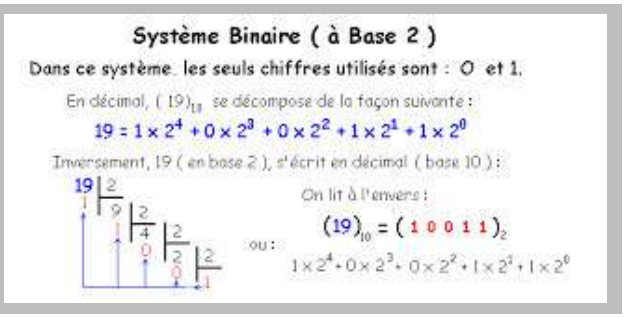
$$= (w+\bar{z})(\bar{w}+\bar{x})(\bar{w}+z)$$

yz / wx	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	1 (4)	0 (5)	0 (7)	1 (6)
11	0 (12)	0 (13)	0 (15)	0 (14)
10	0 (8)	1 (9)	1 (11)	0 (10)

Chapitre 2

Systèmes de Numération et Codage de l'Information

- **Objectifs du chapitre :**
  - Savoir manipuler les opérations arithmétiques et de conversion dans les systèmes de numération. Représenter l'information et décrire les symboles en utilisant les codes binaires.
- **Dans ce chapitre :**
  - ✓ Définition d'un système de numération
  - ✓ Forme polynomiale et conversion entre bases
  - ✓ Intérêt du système binaire
  - ✓ Arithmétique binaire et codes binaires



2.1. Définition d'un système de numération

L'information est toute sorte de connaissance communiquée entre personnes et transférée par l'ensemble des symboles conventionnels (lettres, chiffres …) formant un système facilitant la valorisation sa qualité et le dénombrement sa quantité [4].

Système décimal b = 10	Système binaire b = 2	Système octal b = 8	Système hexadécimal b = 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table 2.1 - Systèmes de numération (B = 10, 2, 8, 16)

Par exemple pour manipuler, afficher ou transmettre des nombres (digits) en utilisant des circuits électroniques traitant automatiquement ces informations sur un certain nombre de données, il faut représenter chaque

nombre par un état différent du circuit. Suivant ce raisonnement, un circuit à dix états permet donc de représenter les chiffres décimaux de 0 à 9 basant sur une décision logique.

Un système de numération (arithmétique) est un système de symboles qui permet de compter le nombre d'occurrence d'un phénomène, autrement dit une correspondance arbitraire entre un système de symboles et un nombre d'objets. En effet, un système de numération est dit de base "B" s'il est formé par B symboles distincts et permet de compter de 0 jusqu'à B-1.

2.2. Forme polynomiale

Généralement, un nombre décrit dans un système de numération de base « b » quelconque peut être exprimé sous la forme polynomiale suivante :

$$N_b = \sum_{i=0}^{n-1} a_i r^i = a_0 + a_1.b + a_2.b^2 + ... + a_{n-1}.b^n$$

n est le plus grand poids (rang à droite)

Le poids d'un chiffre est défini comme étant l'exposant de la base associé à ce chiffre lorsque le nombre est représenté sous sa forme polynomiale. On parle de bit de poids fort (ou plus significatif) et de bit de poids faible (ou mois significatif) lorsqu'il s'agit d'un nombre binaire [5].

Exemples

- $\underset{3\,2\,1\,0}{(6329)_{10}} = 9.10^0 + 2.10^1 + 3.10^2 + 6.10^3$
- $\underset{2\,1\,0}{(121)_3} = 3^0 + 2.3 + 3^2$
- $\underset{3\,2\,1\,0}{(1100)_2} = 2^2 + 2^3$

2.3. Conversion entre bases

Il est possible de convertir un chiffre écrit dans une base « b » vers une autre base « b' »en appliquant un certain nombre de règle :

1) Passage de base « b » vers le décimal « 10 »

C'est simple de trouver l'équivalent décimal d'un nombre écrit dans une base « b = 2, 3, 4, …, 8, ..,16.. » en décomposant ce nombre sous la représentation polynomiale puis additionner les coefficients résultants.

Exemples

- $(11100101)_2 = 1 + 2^2 + 2^5 + 2^6 + 2^7 = 1 + 4 + 32 + 64 + 128 = (229)_{10}$
- $(AF21C)_{16} = 12 + 16 + 2.16^2 + 15.16^3 + 10.16^4 = (717340)_{10}$

2) Passage du décimal « 10 » vers une base « b »

Une méthode très rapide est recommandée pour convertir un chiffre décimal vers une autre base « b » différente à 10 ; consiste à utiliser la séquence de divisions du nombre décimal sur b ($N|\underline{b} \rightarrow r$) jusqu'à ce que le quotient soit nul. Le nombre équivalent est donné par les restes des divisions pris de bas en haut.

Exemples

- $\left.\begin{array}{l} 1989|\underline{16} \rightarrow 5 \\ 124|\underline{16} \rightarrow 12 \\ 7|\underline{16} \rightarrow 7 \\ \boxed{0} \end{array}\right|$ lecture de bas en haut $\Rightarrow (1989)_{10} = (7C5)_{16}$
- $\left.\begin{array}{l} 19|\underline{2} \rightarrow 1 \\ 9|\underline{2} \rightarrow 1 \\ 4|\underline{2} \rightarrow 0 \\ 2|\underline{2} \rightarrow 0 \\ 1|\underline{2} \rightarrow 1 \\ \boxed{0} \end{array}\right| \Rightarrow (19)_{10} = (10011)_2$

3) Passage de l'octal/hexadécimal vers le binaire ou l'inverse

La conversion de l'octal (ou l'hexadécimal) vers le binaire et vice versa joue un rôle important dans les systèmes digitaux. Puisque 8 (16) est une

puissance 3 (4) de la base binaire 2. Chaque chiffre composant le nombre octal (hexadécimal) doit être remplacé par la combinaison de 3 (4) bits correspondante. Ceci est possible pour toute autre système de numération admettant une base de la forme générale 2^n.

Exemples

- $(261)_8 = (\underline{010}\ \underline{110}\ \underline{001})_2$
- $(F06)_{16} = (\underline{1111}\ \underline{0000}\ \underline{0110})_2$
- $(13)_4 = (\underline{01}\ \underline{11})_2$
- $(3617)_8 = (\underline{011}\ \underline{110}\ \underline{001}\ \underline{111})_2$
- $(5A7E)_{16} = (\underline{0101}\ \underline{1010}\ \underline{0111}\ \underline{1110})_2$

La conversion du binaire vers l'octal (hexadécimal) est similaire à savoir grouper les bits en groupe de 3 (4) commençant par la droite vers la gauche (avant la vergule) en complétant les groupes manquants qui ne sont pas multiples de 3 (4) par des zéros, puis convertir chaque groupe en chiffre octal (hexadécimal).

Exemples

- $(\underline{110}\ \underline{100})_2 = (64)_8$
- $(\underline{1101}\ \underline{0100}\ \underline{0111})_2 = (D47)_{16}$
- $(\underline{11}\ \underline{01}\ \underline{11}\ \underline{10})_2 = (3132)_4$

4) Nombres fractionnels

Un nombre fractionnel est composé d'une partie entière et d'une autre partie fractionnaire. La conversion de la partie entière d'une base à une autre se fait la même manière précédente pour la partie entière. La partie fractionnaire subit les règles suivantes :

- ✓ En allant d'une base « b » vers le décimal, les poids attribués aux chiffres après la virgule dans la forme polynomiale prennent des valeurs négatives commencent par la puissance -1 :

$$(0,1101)_2 = 2^{-1} + 2^{-2} + 2^{-4} = (0,8125)_{10}$$

- ✓ En allant du décimal vers une autre base « b » différente à 10, les chiffres après la virgule seront multipliés par la base non pas divisés. Le nombre équivalent est donné par les résultats entiers mais cette fois pris de haut en bas :

$$\left.\begin{array}{r} 0.6875\times 2 = 1 \\ 0.375\times 2 = 0 \\ 0.75\times 2 = 1 \\ 0.5\times 2 = 1 \end{array}\right| \Rightarrow (0,6875)_{10} = (0,1101)_2$$

- ✓ La conversion de l'octal (hexadécimal) vers le binaire ou l'inverse, ne change pas de principe en gardant la virgule et complétant les groupes de bits manquants de la partie fractionnaire par des zéros vers la droite :

$$(0,110110011)_2 = (0,\underline{1101}\ \underline{1001}\ \underline{1000})_2 = (0,D98)_{16}$$

$$(0,217)_8 = (0,\underline{010}\ \underline{001}\ \underline{111})_2 = (0,010001111)_2$$

2.4. Intérêt du système binaire

Les personnes ont toujours travaillé avec le système décimal qui reste difficile à adapter aux mécanismes numériques parce qu'il est difficile de concevoir des circuits électroniques fonctionnant sur dix plages de tensions différentes. En effet, les circuits à dix états sont très rares et même s'ils sont spécialement réalisés ils redeviennent très couteux (chers) aux constructeurs. Il est donc naturel de chercher à utiliser un système de numération avec peu de symboles. Le système binaire (ou système à base 2) traite les nombres à base de seulement deux symboles 0 et 1 appelés bits (de

mot binary digit) et peut être matérialisé par plusieurs phénomènes physiques car en électronique le bit 0 peut être représenté par une tension basse ou nulle (low) et le bit 1, par une tension haute (high) [6].

2.5. Arithmétique binaire

Opérations arithmétiques en binaire se font de la même manière qu'au décimal, à la seule différence que le système binaire utilise seulement les deux symboles 0 et 1.

Addition binaire	Soustraction binaire
0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 10	0 - 0 = 0 0 - 1 = 1 retenue 1 1- 0 = 1 1 - 1 = 0
Multiplication binaire	**Division binaire**
0 × 0 = 0 0 × 1 = 0 1 × 0 = 0 1 × 1 = 1	0 ÷ 0 = impossible 0 ÷ 1 = 0 1 ÷ 0 = impossible 1 ÷ 1 = 1

Table 2.2 - Opérations arithmétiques binaires

Exemples

$$\begin{array}{r} 11101101 \\ +10111001 \\ \hline 110100110 \end{array}$$

L'équivalent de : $237_{10} + 185_{10} = 422_{10}$

$$\begin{array}{r} 11001010 \\ -10101001 \\ \hline 00100001 \end{array}$$

L'équivalent de : $202_{10} + 169_{10} = 33_{10}$

$$\begin{array}{r} 11011 \\ \times\ 10101 \\ \hline 11011 \\ 11011.. \\ 11011.... \\ \hline 1000110111 \end{array}$$

$$\begin{array}{r|l} 101101 & 11 \\ \hline 11 & 01111 \\ \hline 0101 & \\ 11 & \\ \hline 0100 & \\ 11 & \\ \hline 011 & \\ \boxed{0} & \end{array}$$

1) Complément à 2

En binaire l'opération d'addition peut également être utilisée pour l'opération de soustraction grâce au complément à 2 qui pourra être déterminé par :

$CA2 = 2^n - N$ avec N est un nombre binaire écrit avec une combinaison de n chiffre

$$N = 10100110 \rightarrow CA2 = 2^8 - N = 01011010$$

2) Complément à 1

L'opération de soustraction peut aussi employer le complément à 1 qui est tiré par inversion des bits 0 en 1 et 1 en 0 ou en soustrayant 1 binaire du CA2 :

$$N = 10100110 \rightarrow CA1 = 01011001$$

$$CA2 = CA1 + 1$$

3) Nombres entiers négatifs

Il existe trois types de représentation pour les nombres entiers négatifs :

- ✓ **Représentation par un bit de signe et une valeur absolue :** Le premier bit indique le signe (0 pour le signe + et 1 pour le signe –) et le reste des bits représente la valeur absolue du nombre en base 2.

Exemple

- $(+ 4)_{10} = (0100)_2$ où 0 → signe + et 100 → | 4 |
- $(- 2)_{10} = (110)_2$ o`u 1→ signe − et 10 → | 2 |
- − 9 = 11001
- + 9 = 01001

- ✓ **Représentation par le complément à 1 :** Un nombre négatif est obtenu en complémentant tout ses bits.

Exemple

- $(+ 6)_{10}$ = 110 et en ajoutant le bit de signe, on obtient : **0**110. Par inversion $(- 6)_{10}$ = 001 et en tenant compte du bit de signe : **1**001 en CA1.

- ✓ **Représentation par le complément à 2 :** Un nombre négatif est obtenu en complémentant tout ses bits et en lui ajoutant 1 suivant la règle CA2 = CA1 + 1.

Exemple

- $(- 6)_{10}$ = **1**001 en appliquant le CA1 et ajoutant le bit 1, on obtient : **1**010 en CA2.

2.6. Codes binaires

Le monde extérieur utilise des caractères alphanumériques qui ne peuvent pas réaliser les opérations arithmétiques ou numériques qui réalisent les opérations arithmétiques et peuvent exister en plusieurs types tels que les entiers non signés (5, 8,...), entiers signés (+5, -8,…), chiffres réels (5.25, 6.75,…), … pour représenter l'information. Dans les systèmes digitaux, ces caractères sont représentés sous forme binaire au niveau de la machine (pour le traitement) à l'aide des codes binaires en affectant une combinaison binaire unique de 0 et 1 pour chaque caractère.

Avec n bits, on peut construire 2^n mots codes différents parce qu'il existe 2^n arrangements formés par 0 et 1 possibles distincts. Par exemple, un groupe de huit éléments requiert un code de trois bits dont chaque élément aura un code unique parmi 000, 001, 010, 011, 100, 101, 110 et 111. A ce regard, de nombreuses possibilités de codes peuvent être conçues pour

représenter les dix chiffres décimaux qui requièrent un minimum de quatre bits [7].

1) Codes pondérés et non pondérés

Le code est pondéré si la position de chaque symbole dans chaque mot code correspond à un poids fixe : 1 (20), 10 (21), 100 (22), 1000 (23) pour 1, 2,1 4, 8 pour la numération binaire.

	Codes pondérés		Codes non pondérés		
Chiffre décimal	BCD 8421	Aiken 2421	Excédant 3 (BCD +3)	Chiffre décimal	Gray
0	0000	0000	0011	0	0000
1	0001	0001	0100	1	0001
2	0010	0010	0101	2	0011
3	0011	0011	0110	3	0010
4	0100	0100	0111	4	0110
5	0101	1011	1000	5	0111
6	0110	1100	1001	6	0101
7	0111	1101	1010	7	0100
8	1000	1110	1011	8	1100
9	1001	1111	1100	9	1101
				10	1111
				11	1110
				12	1010
				13	1011
				14	1001
				15	1000

Table 2.3 – Codes pondérés et non pondérés

- ✓ **Code BCD (Binary Coded Decimal) :** Le code BCD code qui semble le plus naturel. C'est un code pondéré 8421 : à chaque position de bit on associe un poids. Le chiffre qui correspond au code $a_0a_1a_2a_3$ est obtenu par l'opération : $8.a_3 + 4.a_2 + 2.a_1 + 1.a_0$ (Exemple : $8.0 + 4.1 + 2.0 + 1.1 = 5_{10}$).
- ✓ **Code Aiken :** Le code Aiken est un code 2421 pondéré. Pour les chiffres décimaux 0, 1, 2, 3 et 4, il est identique au code BCD par contre il termine les cinq restants chiffres décimaux 5, 6, 7, 8 et 9 par les concordances aux chiffres décimaux 11, 12, 13, 14 et 15 en codes binaire naturel. Ces derniers chiffres (5-9) sont compléments aux premiers (0-4) par rapport à l'axe. 0100/1011, 0001/1000 …
- ✓ **Code excédant-3 :** Le code excédant-3 (Ex-3) s'obtient en ajoutant 3 (0011) au mot code BCD. Il est utile pour faire l'arithmétique en utilisant la méthode de complément. Ce code présente l'avantage qu'il est auto-complémentaire où chaque chiffre est obtenu par la simple inversion de tous les bits.
- ✓ **Code Gray :** Le code Gray est un code cyclique où le passage d'un chiffre au chiffre immédiatement suivant ne diffère que d'un seul bit. Il contient donc deux combinaisons successives adjacentes. Pour passer du BCD au binaire réfléchi, on laisse les chiffres tels qu'ils sont s'ils sont précédés par 0 sinon on les change s'ils sont précédés par 1 de droite à gauche. La dernière combinaison représentant le chiffre 15 est adjacente à la première combinaison représentant le 0. L'utilisation de ce code réduira les erreurs dans les conversions analogiques/numériques, comme il est recommandé pour les tableaux de Karnaugh dans la conception des circuits logique, et la capture angulaire ou de positionnement.

0	00	000	0000
1	01	001	0001
		011	0011
		010	0010
		110	0110

Génération du code Gray par réflexion

111 0111
101 0101
100 0100
1100
1101
1111
1110
1010
1011
1001
1000

2) Codes détecteurs d'erreurs

Il existe aussi d'autres codes binaires appelés codes détecteurs d'erreurs utilisés pour les transmissions des messages d'une station à l'autre dans un réseau téléinformatique afin de réduire le risque d'erreurs :

- ✓ **Code P parmi n :** A chaque chiffre décimal correspondent n bits dont P sont égaux à 1 et (n - P) sont égaux à 0. Par exemple, dans le code 2 parmi 5, chaque nombre décimal est représenté par 2 bits 1 et trois bits 0 en utilisant le poids 74210 ou bien 84210. Ces codes permettent de détecter 0 ou 1 erreur mais pas 2, 3, 4 ou 5 sans permettre les corriger.

Chiffre décimal	74210	84210
0	11000 (*)	11000 (*)
1	00011	00011
2	00101	00101
3	00110	00110
4	01001	01001
5	01010	01010
6	01100	01100
7	10001	10100 (*)
8	10010	10001
9	10100	10010
(*) : Anomalie		

Table 2.4 - Code 2 parmi 5

- ✓ **Code biquinaire :** Il est composé de deux groupes de bits chacun contenant un 1, il permet de détecter 2 erreurs à condition qu'elles ne soient pas dans le même groupe sans permettre localiser ou corriger ces erreurs.

Chiffre décimal	Groupe 1 Poids : 50	Groupe 2 43210
0	01	00001
1	01	00010
2	01	00100
3	01	01000
4	01	10000
5	10	00001
6	10	00010
7	10	00100
8	10	01000
9	10	10000

Table 2.5 - Codes biquinaires

- ✓ **Code avec bit de parité :** Pour détecter les erreurs d'un code qui ne présente pas la propriété du code biquinaire, on associe au code un bit supplémentaire appelé « bit de parité » qui servira à déterminer la parité du code. Ce bit est égal à 1 si le nombre de bits est impair, à 0 dans le cas contraire. A la réception du message, le nombre de bits total (code + bit de parité) est compté. Si ce dernier est impair, une erreur est détectée. Si le message présente un nombre important d'erreur, le message est écarté et on essaye de réparer la cause de l'erreur. Si par contre, on ne remarque que quelques rares erreurs, on essaye de les corriger ou de demander à l'émetteur de renvoyer la partie erronée.
- ✓ **Code alphanumériques :** Il existe d'autres codes appelés alphanumériques utilisés pour coder les données non numériques tels que :
  - □ **Code ASCII :** Ce code est une abréviation de « American Standard Code for Information Interchange », sert à coder 128 caractères possibles se divisant en caractères alphanumériques

(a, b, c, … A, B, C, … 1, 2, 3, …) et caractères spéciaux (@, #, *, ?, +, §, …). Chaque mot, représenté en code ASCII est codé en hexadécimal est l'équivalent d'une séquence binaires contenant 7 (actuellement 8) bits, suivant la table suivante :

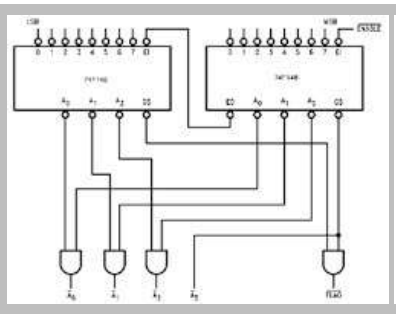
	0	1	2	3	4	5	6	7
0	NUL	DLE	ESP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

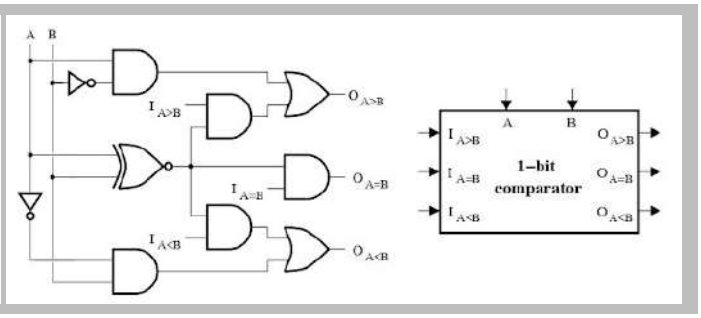
Table 2.6 - Table du codage ASCII (ISO-646)

Chapitre 3

Circuits Logiques Combinatoires

- **Objectifs du chapitre :**
  - Savoir analyser et synthétiser un circuit logique combinatoire. Connaître les différentes fonctions combinatoires standards disponibles en circuits intégrés.
- **Dans ce chapitre :**
  - ✓ Définition d'un circuit combinatoire
  - ✓ Synthèse et analyse d'un circuit logique combinatoire
  - ✓ Circuits combinatoires particuliers
  - ✓ Circuits combinatoires aiguilleurs
  - ✓ Circuits combinatoires transcodeurs





3.1. Définition d'un circuit combinatoire

Les circuits logiques sont à la base de tout matériel électronique, dont on ne peut pas comprendre la structure ni le fonctionnement d'une machine électronique sans comprendre le fonctionnement de ces circuits logiques fondamentaux.

Un circuit logique n'est qu'un dispositif qui traite et exécute des opérations logiques sur des variables logiques. Il est dit combinatoire si ses valeurs de sortie ne dépendent que de ses valeurs d'entrée. En général, l'étude des circuits logiques combinatoires consiste en deux sens [8] :

- **Synthèse :** consiste à réaliser un circuit logique à partir de l'énoncé décrivant les fonctions ou le rôle du circuit.
- **Analyse :** qui constitue le travail inverse qui sert à déterminer le rôle du circuit logique à partir de sa présentation (présentation schématique à l'aide de portes logiques).

Plus précis, un circuit combinatoire consiste en variables d'entrée, de sortie et des portes logiques qui reçoivent des signaux appliqués en entrée et produisent des signaux en sortie.

Ce processus transforme une information binaire donnée à l'entrée en une autre information demandée en sortie. Ce principe permet donc de schématiser un circuit combinatoire comme suit :

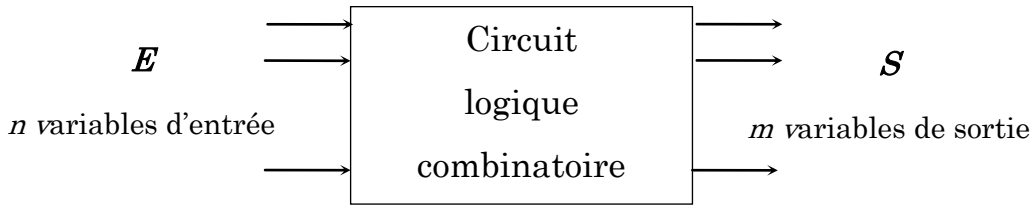


Figure 3.1 – Schéma synoptique d'un circuit combinatoire

3.2. Synthèse d'un circuit logique combinatoire

La synthèse d'un circuit logique combinatoire a pour but la réalisation d'une fonction logique qui remplit un cahier des charges et qui satisfait

également à d'autres critères tels que le coût et l'encombrement minimum par exemple. Le nombre de circuits à produire, le matériel à disposition, le délai de réalisation,… etc. sont d'autres paramètres dont il faut tenir compte lors de la synthèse. De façon générale, la simplification d'un circuit est toujours utile. Concrètement, il s'agit de déterminer le logigramme associé à une fonction logique connaissant la définition de cette dernière. Les étapes essentielles à suivre pour réaliser une synthèse d'un circuit logique combinatoire sont [7] :

1) Établir puis simplifier les équations représentant la fonction logique à traiter.
2) Établir le logigramme du circuit logique puis réaliser le circuit logique.

Autrement dit, avant d'effectuer la synthèse d'un circuit combinatoire, on doit d'abord répondre aux questions suivantes :

1) Quelle représentation de la fonction à choisir ? somme de mintermes ou produit de maxtermes !
2) Quelles portes logiques utilisées pour faire la représentation schématique ?
3) Est-que la fonction est simplifiée ?

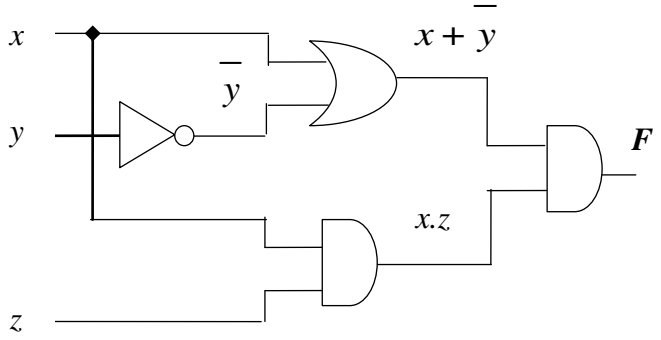
- **Exemple**

Soit la fonction logique suivante : $F(x, y, z) = x.z.(x + \overline{y})$

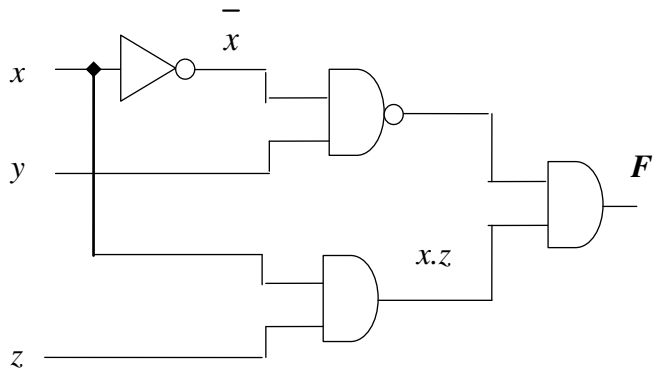
On peut aussi écrire : $F(x, y, z) = x.z.(\overline{\overline{x}.y}) = x.z.\left[\left[(x \perp) \perp y\right]\right]$

On peut aboutir la même sortie de la fonction logique en utilisant des portes logiques de types différents tels que :

- ✓ OR et AND :

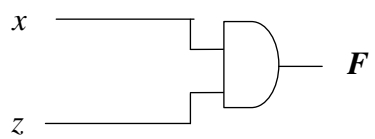


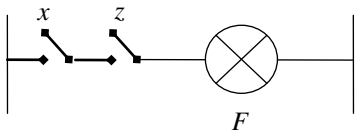
- ✓ NAND :



- ✓ Comme on peut aussi simplifier cette fonction logique et utiliser une seule porte logique basique du type AND :

$$\begin{aligned} F(x,y,z) &= x.z.(x+\bar{y}) \\ &= x.x.z + x.z\,\bar{y} \\ &= x.z + x.z\,\bar{y} \\ &= x.z\left[1+\bar{y}\right] \\ &= x.z \end{aligned}$$





3.3. Analyse d'un circuit logique combinatoire

L'analyse d'un circuit combinatoire consiste à étudier le logigramme du circuit logique dans le but de déterminer son rôle. Elle consiste simplement à trouver à partir d'un logigramme ses fonctions logiques comme suit [6-7] :

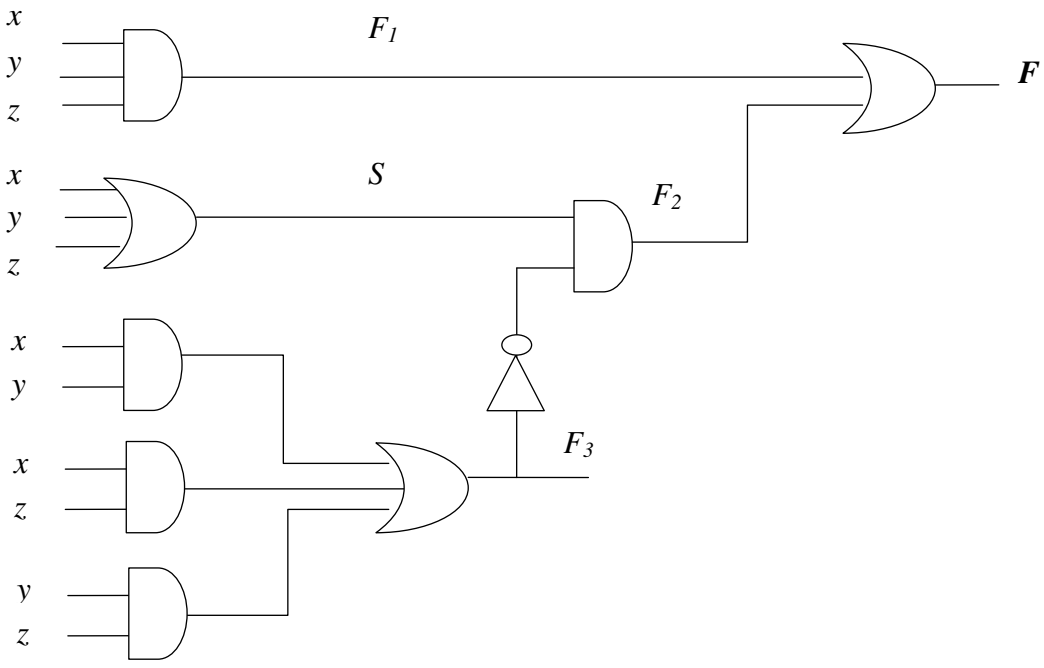
1) Donner l'expression de chaque porte en fonction des valeurs de ses entrées.
2) Déduire au final la ou les fonctions logiques du circuit analysé.
3) On peut ensuite établir la table de vérité du circuit analysé et opéré à une simplification à l'aide des propriétés de l'algèbre de Boole ou de la table de Karnaugh.

De manière générale, la démarche à suivre pour effectuer une analyse d'un circuit combinatoire consiste des étapes essentielles suivantes :

1) Déterminer les expressions logiques des variables de sortie, pour cela on associe à chaque sortie de porte une variable et on détermine la fonction logique correspondante en commençant par les portes directement connectées aux variables d'entrée. A la fin de cette étape, on doit avoir les expressions des variables de sortie en fonction des variables d'entrée, uniquement.
2) Dresser la table de vérité du circuit et la traduire par un énoncé décrivant le rôle du circuit logique à étudier.

- **Exemple**

Soit à analyser le circuit combinatoire suivant :



✓ Détermination de la fonction F :

$$\begin{aligned} F_2 &= S.\overline{F_3} \\ &= (x+y+z)(\overline{xy+xz+yz}) \\ &= (x+y+z)(\bar{x}+\bar{y})(\bar{x}+\bar{z})(\bar{y}+\bar{z}) \\ &= \bar{x}.\bar{y}.z+\bar{x}.y.\bar{z}+x.\bar{y}.\bar{z} \end{aligned}$$

$$\downarrow$$

$$F = F_1 + F_2 = x.y.z+\bar{x}.\bar{y}.z+\bar{x}.y.\bar{z}+x.\bar{y}.\bar{z}$$

✓ Table de vérité :

x	*y*	*z*	F_1	*S*	F_3	F_2	*F*	*Remarque*
0	0	0	0	0	0	0	0	0+0+0=0, La retenue est 0
0	0	1	0	1	0	1	1	0+0+1=1, La retenue est 0
0	1	0	0	1	0	1	1	0+1+0=1, La retenue est 0
0	1	1	0	1	1	0	0	0+1+1=0, La retenue est 1
1	0	0	0	1	0	1	1	1+0+0=1, La retenue est 0
1	0	1	0	1	1	0	0	1+0+1=0, La retenue est 1
1	1	0	0	1	1	0	0	1+1+0=0, La retenue est 1
1	1	1	1	1	1	0	1	1+1+1=1, La retenue est 1

✓ On conclut, que le rôle du circuit logique est d'additionner trois variables x, y et z dont la sortie F indique le résultat et la variable F_3 indique la retenue.

3.4. Circuits combinatoires particuliers

1) Additionneur

Un additionneur est un circuit combinatoire présentant un élément fondamental de toute unité de traitement, son rôle est d'additionner les bits. L'addition de deux nombres binaires s'accomplit en additionnant les bits de même rang en commençant par les bits mois forts en allant vers ceux des rangs les plus forts. Un tel circuit doit présenter :

- ✓ Des entrées (nombres binaires).
- ✓ Une sortie (le résultat).
- ✓ Une retenue.

a) Demi-additionneur

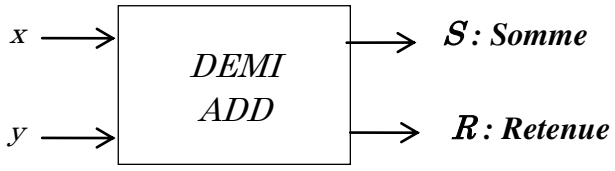
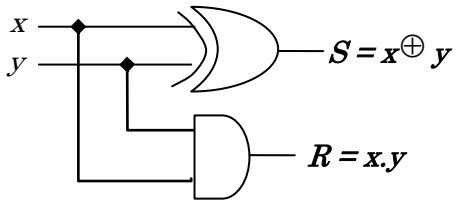


Figure 3.2 – Schéma synoptique d'un demi-additionneur

Ce circuit convient pour effectuer la somme de deux bits de plus faibles poids de deux nombres de *n* bits, mais il ne convient pas pour réaliser la somme des autres bits, car il ne prend pas en compte l'éventuelle retenue provenant de la somme des bits de poids inferieurs.

Un demi-additionneur binaire se porte sur un bit unique et mène aux 4 cas notés dans la table de vérité suivante :

x	y	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



b) Additionneur complet

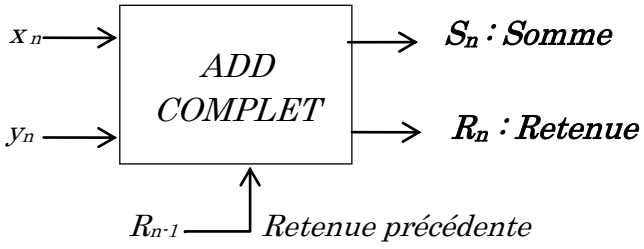
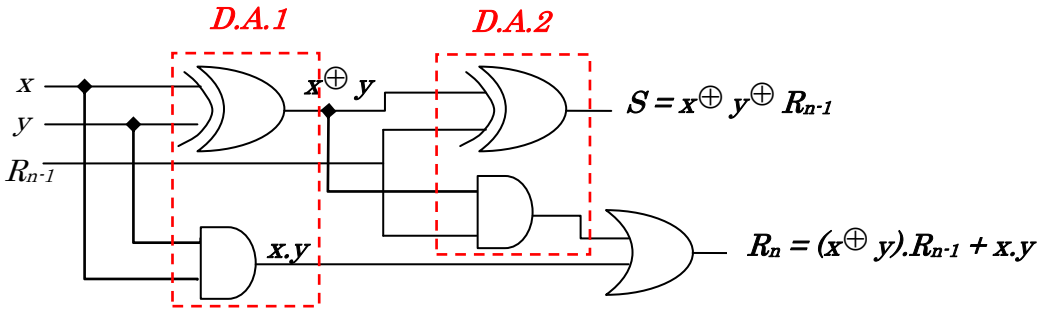


Figure 3.3 – Schéma synoptique d'un additionneur complet

La représentation de l'additionneur complet est donnée par le schéma suivant, où R_n indique la retenue et R_{n-1} la retenue précédente. Le fonctionnement de ce circuit est donné par la table de vérité suivante :

x	y	R_{n-1}	S	R_n
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



2) Comparateur

Un comparateur est un circuit combinatoire qui effectue la comparaison entre deux mots **x** et **y** de n bits présents sur n lignes d'entrée. Un circuit comparateur dispose de trois sorties (en général) qui indiquent les résultats de la comparaison :

- ✓ Une sortie $x = y$.
- ✓ Une sortie $x > y$.
- ✓ Une sortie $x < y$.

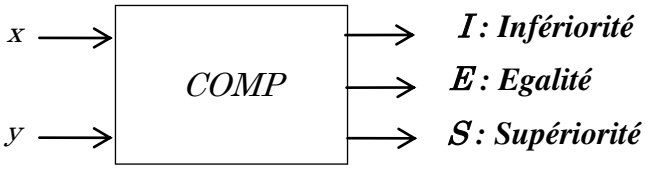
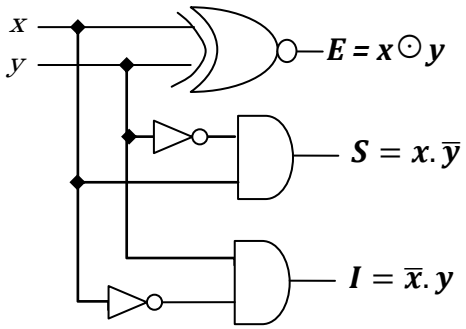


Figure 3.4 – Schéma synoptique d'un comparateur

La table de vérité suivante présente fonctionnement d'un comparateur à 1 bit :

x	*y*	*E*	*S*	*I*
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0



3.5. Circuits combinatoires aiguilleurs

Les circuits d'aiguillage sont des circuits combinatoires permettent de faire passer plusieurs informations par une même ligne de transmission exploitées en commun, c'est le cas du multiplexage et le démultiplexage appliqués à la centrale du téléphone et d'autres applications digitales [5].

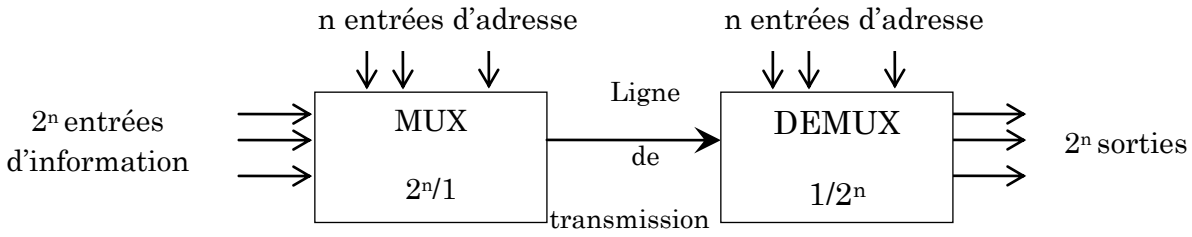


Figure 3.5 – Principe multiplexage-démultiplexage

1) Multiplexeur

Le multiplexeur (MUX) peut être regardé comme un interrupteur électronique à plusieurs positions qui peut relier une de ses 2^n lignes d'entrée à une seule ligne de sortie selon la combinaison des n lignes de commande.

Un multiplexeur digital est un circuit logique combinatoire qui fait transiter un grand nombre d'unités discrètes d'information sur un petit nombre de canaux ou de lignes. Il est utilisé comme un sélecteur de données le plus souvent dans les systèmes de communication où la sélection des chemins de transferts entrées-sortie est réalisée par des portes logiques et commandée par des lignes de contrôle :

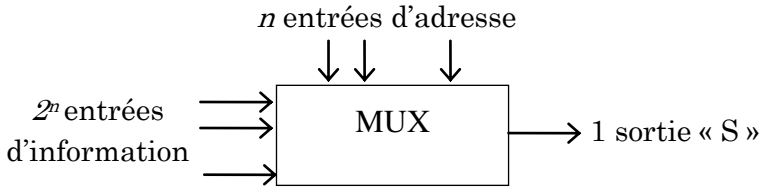
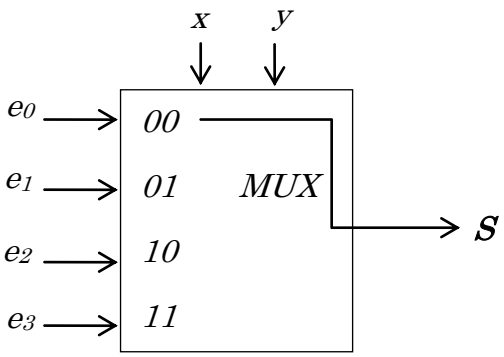


Figure 3.6 – Schéma synoptique d'un multiplexeur

- **Exemple**

Un exemple de multiplexeur à deux entrées d'adresse ou de sélection ($n = 2$) va être présenté.

Ce circuit logique se composant de deux entrées de commande x et y contient quatre entrées d'information et une sortie (MUX 4 à 1). Il permet de commuter les données présentes à l'une de ses entrées vers sa sortie unique.



- ✓ x et y sont appelées lignes de sélection (entrées d'adresse)
- ✓ e_0, e_1, e_2 et e_3 sont les lignes de données (entrées d'information)
- ✓ S est le résultat (sortie)

Les lignes de commande déterminent quelle entrée se retrouve en sortie ; de ce fait, on dira qu'un multiplexeur est un sélecteur de données.

x	y	S
0	0	$e_0.\bar{x}.\bar{y}$
0	1	$e_1.\bar{x}.y$
1	0	$e_1.x.\bar{y}$
1	1	$e_3.x.y$

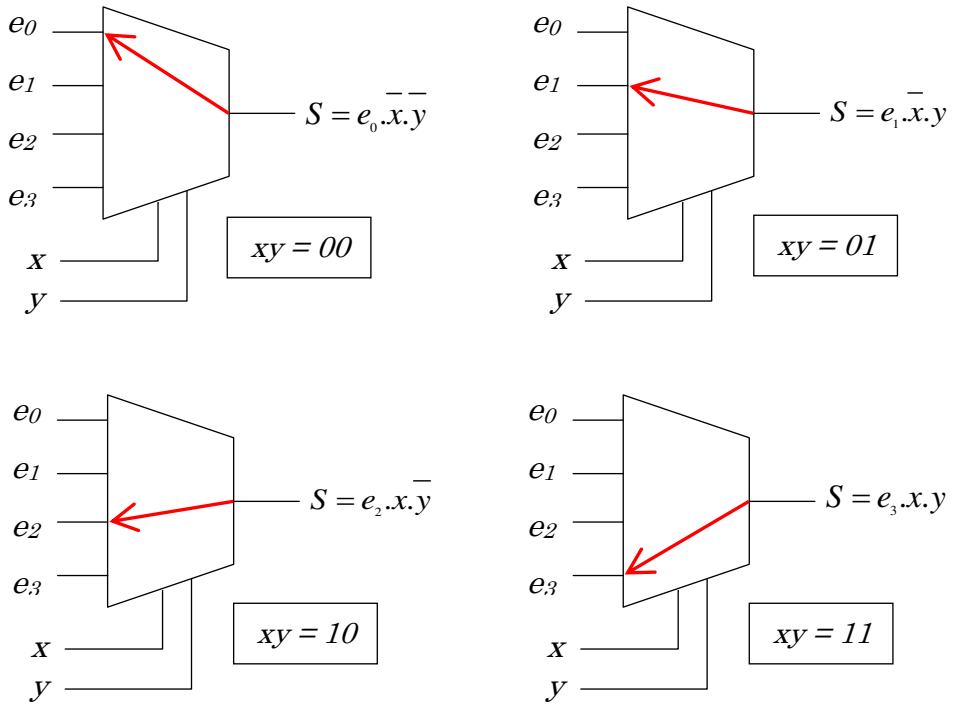


Figure 3.7 – Lignes de commande d'un MUX 4 à 1

Le circuit logique du multiplexeur 4 à 1 est réalisé à partir du logigramme suivant comprenant les portes logiques élémentaires NOT, AND et OR :

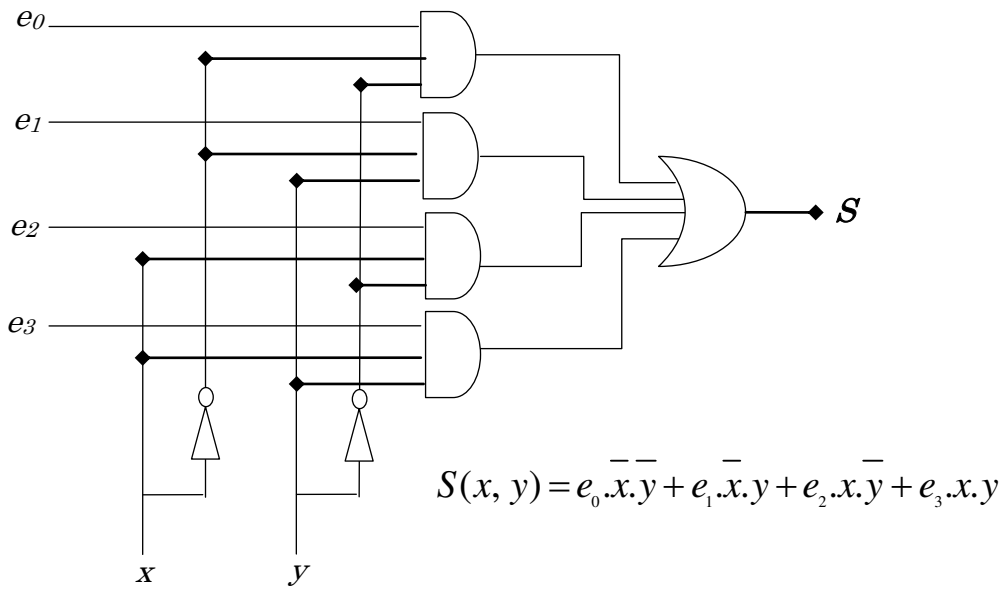


Figure 3.8 – Logigramme d'un MUX 4 à 1

Les lignes de sélection (commande) déterminent quelle entrée se retrouve en sortie ; de ce fait, on dira qu'un multiplexeur est un sélecteur de données. Les multiplexeurs ont de nombreuses applications. De plus leurs applications comme sélecteurs de données, ils peuvent par exemple être utilisés comme convertisseurs parallèle-série : le multiplexeur reçoit en parallèle des données qu'il peut transmettre l'une après l'autre sur sa sortie, ou générateurs de fonctions logiques.

2) Démultiplexeur

Le démultiplexeur réalise l'opération inverse de celle du multiplexeur. Il comporte une seule entrée d'information (données), *n* entrées de sélection (commande) 2^n sorties. Le schéma représentatif du démultiplexeur est illustré comme suit :

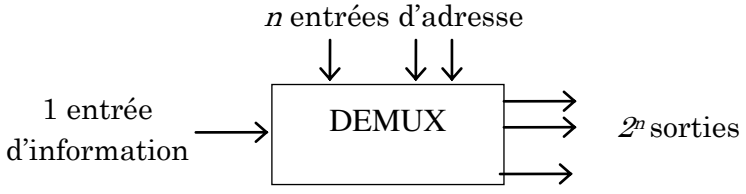


Figure 3.9 – Schéma synoptique d'un démultiplexeur

- **Exemple**

Un exemple de démultiplexeur (DEMUX 1 à 4) à deux entrées de sélection x et y est présenté. Ce circuit logique se compose d'une seule entrée d'information et quatre sorties. Il permet de dispatcher les données présentes sur ses entrées sur ses différentes sorties.

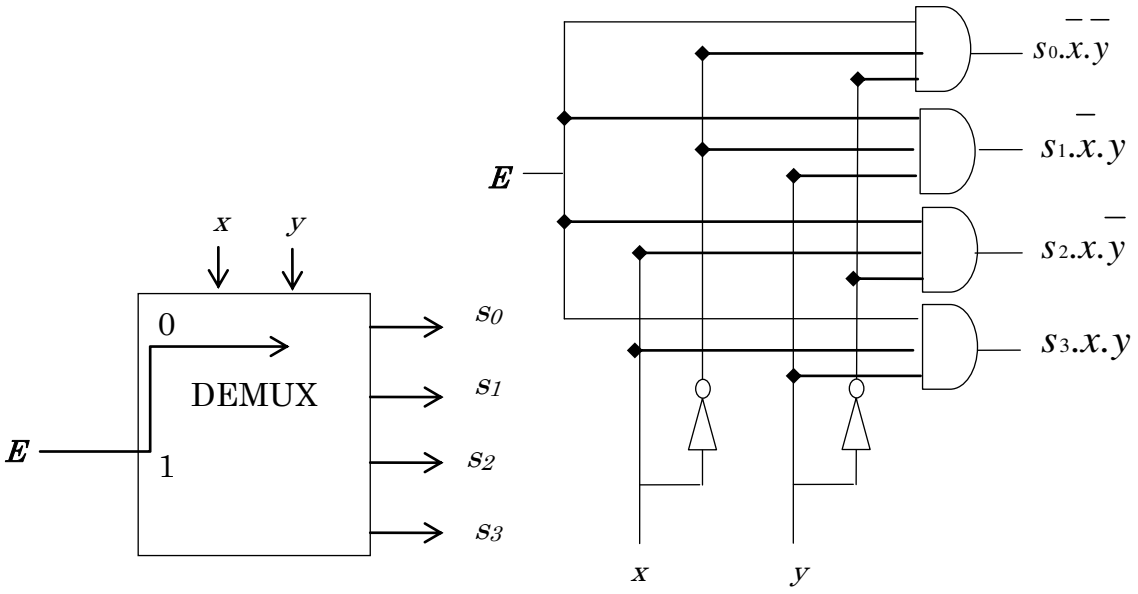


Figure 3.10 – Schéma synoptique et Logigramme d'un DEMUX 1 à 4

3.6. Circuits combinatoires transcodeurs

La disponibilité d'une grande variété de codes pour les mêmes unités discrètes d'information résulte en l'utilisation de différents codes par différents systèmes digitaux. Il est nécessaire quelques fois de connecter la sortie d'un système à l'entrée d'un autre système ; si ces deux systèmes manipulent deux codes différents pour la même information, un circuit de conversion doit être installé entre les deux systèmes [5].

1) Transcodeur

Le transcodeur ou autrement dit convertisseur de codes est circuit combinatoire qui a pour tâche principale d'assurer la compatibilité de deux systèmes traitant la même information avec deux codes différents. Il fait

correspondre à un code X traite en entrée sur n lignes, un code Y en sortie sur m lignes.

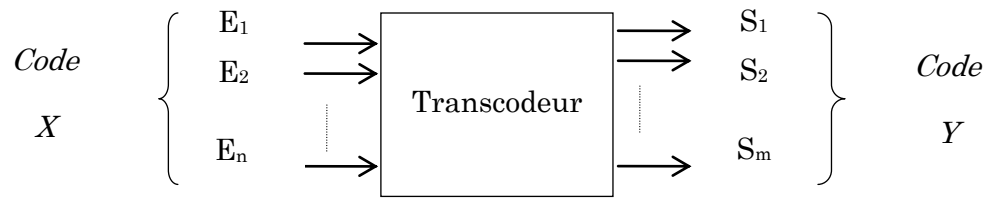


Figure 3.11 – Principe de transcodage

- **Exemple**

La procédure de conversion du code pondéré BCD en code non pondéré Excédent-3 est montrée dans la table suivante :

Chiffre décimal	BCD E_0 E_1 E_2 E_3	Excédant-3 S_0 S_1 S_2 S_3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Table 3.1 – Codes BCD et Ex-3

Puisque les deux codes utilisent 4 bis chacun pour représenter un chiffre décimal. Le nombre de variables en entrée sera égal à 4, celui des variables en sortie sera aussi égal à 4. La table de vérité du transcodeur BCD/EX-3 est comme suit :

A B C D	A' B' C' D
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0

0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 1 1 0	- - - -
. . .	. . .
1 1 1 1	- - - -

Après simplification par table de Karnaugh :

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

$$C' = C \odot D \qquad D' = \overline{D}$$

Un logigramme à deux niveaux peut être obtenu directement à partir des expressions algébriques simplifiées. Il existe plusieurs autres possibilités pour implémenter ce circuit de transcodage :

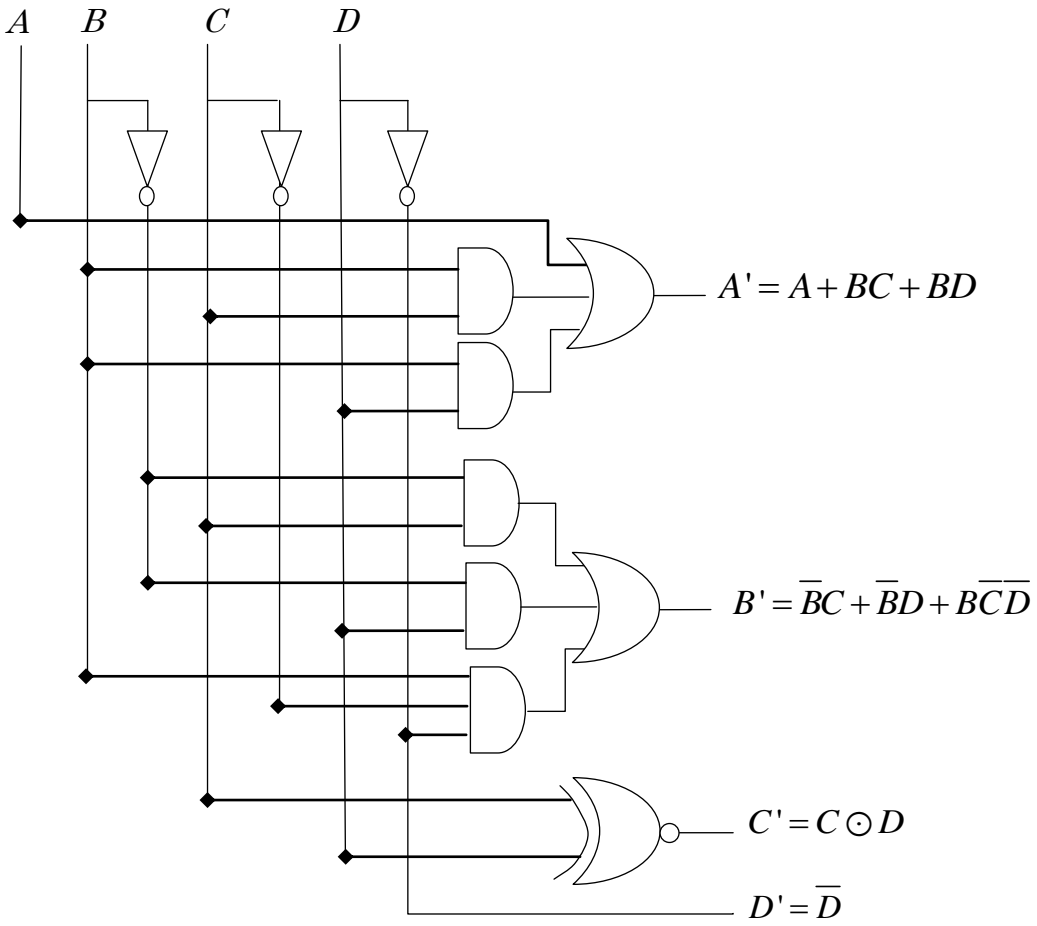


Figure 3.12 – Transcodeur BCD/EX-3

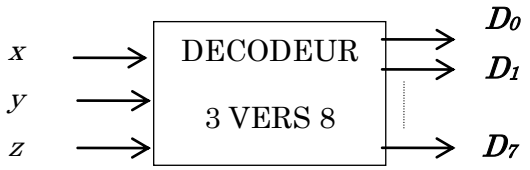
2) Décodeur

Le décodeur est un circuit combinatoire dit de type n vers 2^n, traduit l'information binaire présente sur *n* lignes d'entrées et l'utilise pour mettre à l'état « 1 » l'une seulement de ses 2^n lignes de sortie.

Les décodeurs sont largement utilisés dans de nombreuses applications. Ils sont utilisés pour adresser une case mémoire pour une opération de lecture ou écriture, ou sélectionner une mémoire parmi plusieurs. Ils sont aussi utilisés pour le passage d'un code à l'autre (conversion entre base) comme dans le casdu décodeur BCD/7-segments où les données en BCD à l'entrée du circuit sont converties en décimal pour être affichées sur des afficheurs à 7-segments.

- **Exemple**

Par exemple un décodeur « 1 parmi 8 » dit aussi « 3 vers 8 » consiste simplement en huit portes AND réalisant les huit combinaisons possibles des trois variables d'entrée. Pour chaque combinaison des variables d'entrée, seulement une sortie du décodeur est égale à 1 (activée).



x	*y*	*z*	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

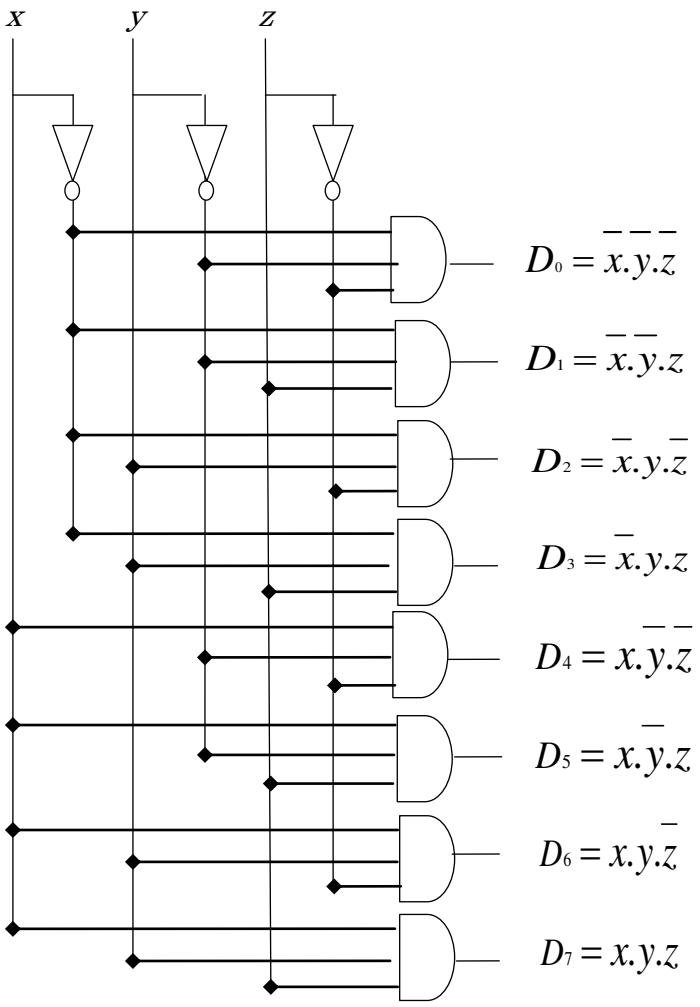


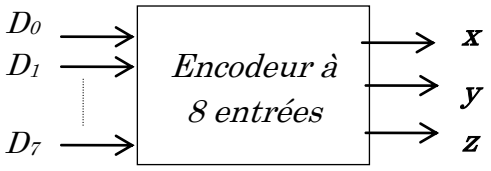
Figure 3.13 –Décodeur 3 vers 8

3) Encodeur

L'encodeur réalise effectivement la fonction inverse du décodeur, c'est-à-dire à une entrée active (état 1), parmi les 2^n entrées, il faut correspondre en sortie un code sur *n* lignes. Il génère un code de sortie différent pour chaque signal d'entrée sélectionné. v

- **Exemple**

L'encodeur de priorités à 8 entrées par exemple génère en sortie un code à trois bits qui indique l'entrée de plus grand priorité sélectionnée. La première ligne dans la table de vérité indique que l'entrée e_0 est activée ; le code de sortie est par conséquent le nombre binaire $S_3S_2S_1$ = 000 qui indique que a la priorité.



D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
*	1	0	0	0	0	0	0	0	0	1
*	*	1	0	0	0	0	0	0	1	0
*	*	*	1	0	0	0	0	0	1	1
*	*	*	*	1	0	0	0	1	0	0
*	*	*	*	*	1	0	0	1	0	1
*	*	*	*	*	*	1	0	1	1	0
*	*	*	*	*	*	*	1	1	1	1

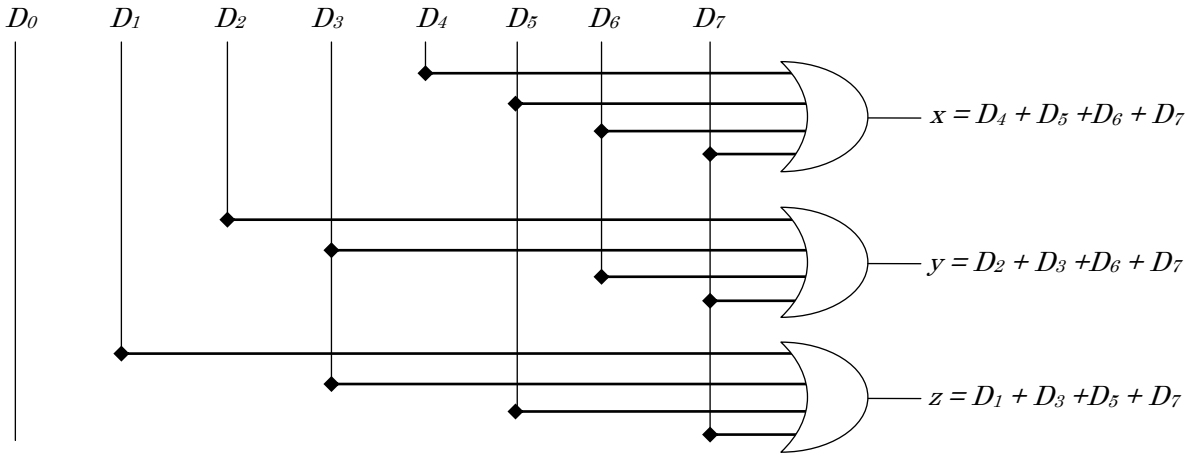


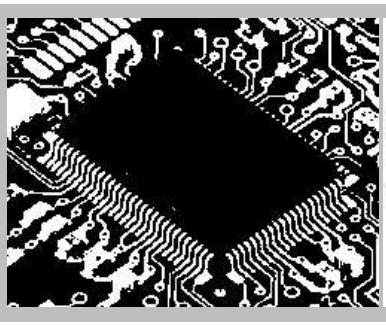
Figure 3.14 – Encodeur à 8 entrées

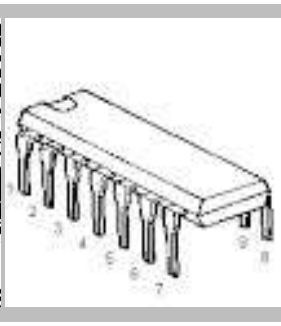
Un tel encodeur est disponible en circuit intégré doté d'une entrée de validation pour la sélection du circuit et deux sorties de validation, utilisées pour la mise en cascade de plusieurs circuits semblables lors de l'extension à plus de huit entrées.

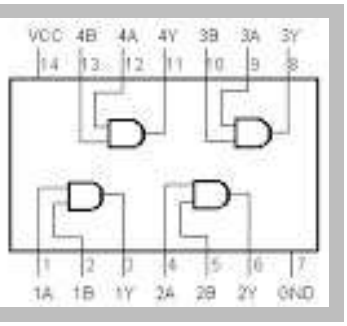
Chapitre 4

Implémentation des circuits combinatoires

- **Objectifs du chapitre :**
  - Savoir synthétiser un circuit combinatoire au moyen des portes et circuits logiques universels.
- **Dans ce chapitre :**
  - ✓ Définition d'un circuit intégré
  - ✓ Critères d'implémentation
  - ✓ Logique NAND/NOR
  - ✓ Synthèse au moyen de multiplexeur / démultiplexeur / PLA.







4.1. Définition d'un circuit intégré

Un circuit intégré appelé aussi « puce » ou « chip en anglais » est un silicium contenant des transistors, des diodes, des résistances ainsi que des condensateurs groupant un certain nombre de portes logiques. Il est enveloppé dans un boitier en plastique ou en métal et possède un certain nombre de broches métalliques servant pour les connexions externes [9].

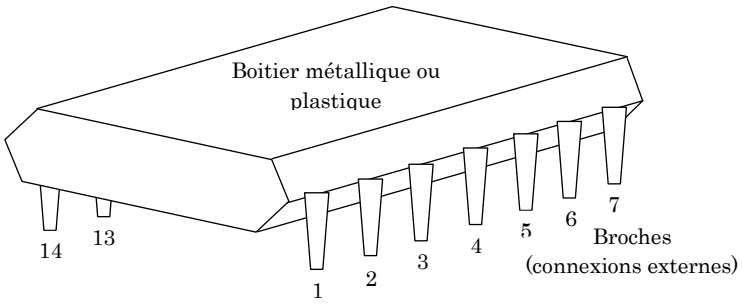


Figure 4.1 – Vue externe d'un exemple de boitier de circuit intégré à 14 broches

La figure ci-dessus illustre un exemple de boitier de circuit intégré enveloppant quatre portes NAND. Les broches 1 et 2 par exemple sont les entrées d'une porte NAND et la broche 3 la sortie de cette porte comme le montre la figure ci-dessous :

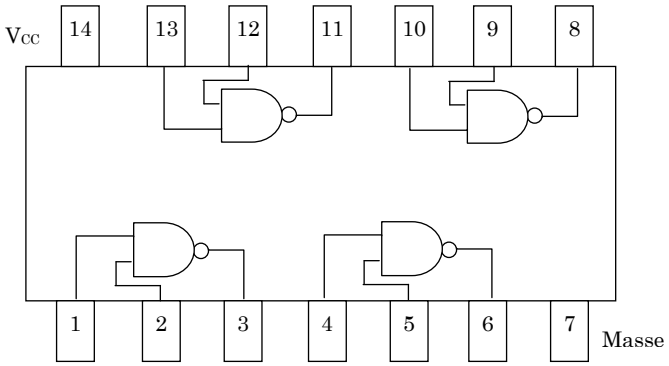


Figure 4.2 – Schéma interne d'un exemple de boitier de circuit intégré à 14 broches

Les circuits intégrés offrent plusieurs avantages tels que :

- ✓ Réduction de la taille ; un circuit intégré peut contenir plusieurs portes logiques dans un volume très restreint ;
- ✓ Réduction substantielle du cout de réalisation ;
- ✓ Fiabilité contre les accidents due à l'enveloppe métallique ou plastique ;
- ✓ Rapidité d'opération et réduction de connexions externes.

Vue tous ces aspects positifs, les circuits intégré ont connu une grande évolution ces dernières années. Ils peuvent être classés dans trois grandes catégories [10] :

1) **Circuits SSI « Small Scale Integration » :** Un exemple de circuit est celui de la figure 4.2 contennant moins de dix portes logiques.
2) **Circuits MSI « Medium Scale Integration » :** C'est un circuit qui comporte dix portes logiques en moyennes comme par exemple les cicruits logiques déjà étudiés dans le chapitre 3 (additionneur, comparateur. Encodeur et décodeur).
3) **Circuits LSI « Large Scale Integration » :** Ce circuit englobe plus d'une centaine de portes logiques. Un calculateur peut être construit à l'aide d'un petit nombre de circuits LSI.

Les circuits MSI et LSI présentent l'avantage d'être très économiques et de permettre une réduction de la consommation d'énergie. En contrepartie, ils sont difficiles à construire. Aussi, une fois construits, il est impossible de modifier ou toucher aux connexions internes et d'introduire d'autres circuits ou portes à l'intérieur de l'enveloppe.

4.2. Critères d'implémentation

Avant de procéder à l'implémentation des circuits logiques combinatoires il est très important de simplifier ces fonctions booléennes afin d'achever un processus de réalisation économique. Ils existent d'autres

facteurs mis à part le coût du circuit, qui restent en jeu dans la simplification des fonctions logiques. Ces critères sont classés dans trois classes essentielles :

1) **Réduction du nombre de composants et connexions :** On s'intéresse ici à la réduction du nombre de portes logiques, du nombre d'entrée à une porte, du nombre de circuits intégrés et aussi au nombre de fils de connexion. Ces critères conduisent à l'obtention d'un circuit économique.
2) **Réduction du délai de propagation :** Un signal traverse un circuit à partir des entrées du circuit jusqu'aux sorties de celui-ci. Cet intervalle de temps est appelé « délai de propagation » du circuit. Pour arriver aux sorties du circuit, le signal passe par un certain nombre de portes logiques mais pas nécessairement par toutes. Le délai de propagation du circuit est égal à la somme des délais de propagation des portes logiques traversées par le signal. Une réduction du délai de propagation entraine une opération rapide. Ce critère est crucial traitement en car très souvent on cherche d'accélérer son exécution. C'est donc un facteur à considérer sérieusement. Une procédure de réduction du délai de propagation d'un circuit consiste tout simplement à réduire le nombre de niveaux ou étages du circuit.
3) **Prise en compte des contraintes de chargement :** l'objectif est de chercher à établir un bon fonctionnement du circuit. Ces critères concernent particulièrement les sorties des portes logiques. La sortie d'une porte logique possède une charge limitée. Si elle est connectée à plusieurs autres portes à la fois, elle peut cesser d'opérer. Dans ce cas l'opération attendue du circuit peut devenir imprévisible.

4.3. Logique NAND/NOR

Les circuits combinatoires sont beaucoup plus construits avec des portes NAND et NOR qu'avec des portes AND, OR et NOT, car du point

technologique, les portes NAND et NOR sont plus faciles à construire [8]. Il est donc intéressant de savoir réaliser des circuits combinatoires avec des portes NAND ou NOR.

Les portes NAND et NOR sont dites universelles car tout circuit combinatoire peut être construit à l'aide de ces portes uniquement. Pour synthétiser des circuits combinatoires au moyen de portes NAND ou NOR, il est recommandé de faire la synthèse avec des portes AND, OR et NOT tout d'abord et dans une deuxième étape procéder à la conversion du circuit AND/OR/NOT en un circuit NAND/NOR et ceci parce qu'il n'est du tout évident de manipuler algébriquement l'opérateur $\perp$ / $\top$. la procédure de conversion est basée sur la substitution aux portes AND, OR et NOT des blocs logiques équivalents constitués uniquement de portes NAND ou NOR :

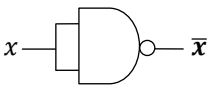
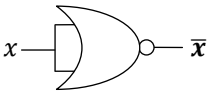
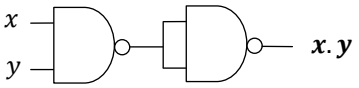
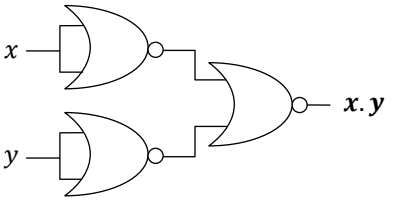
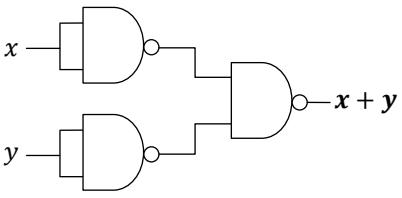
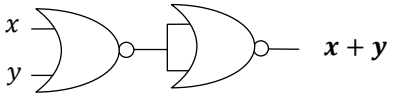
Portes logiques	NAND	NOR
NOT		
AND		
OR		

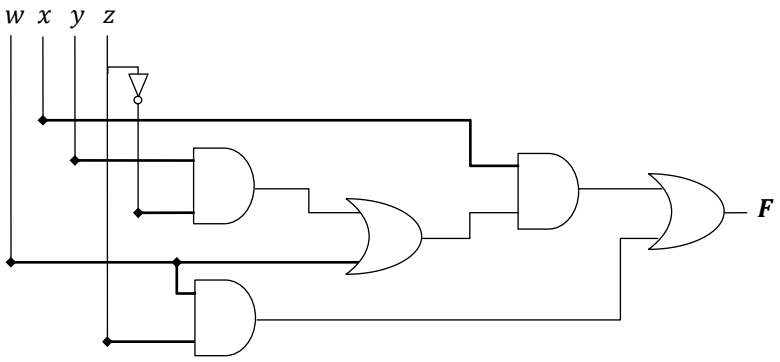
Table 4.1 – Réalisation des portes NOT/AND/OR à l'aide des portes universelles NAND/NOR

1) Procédures de conversion d'un circuit AND/OR/NOT en un circuit NAND

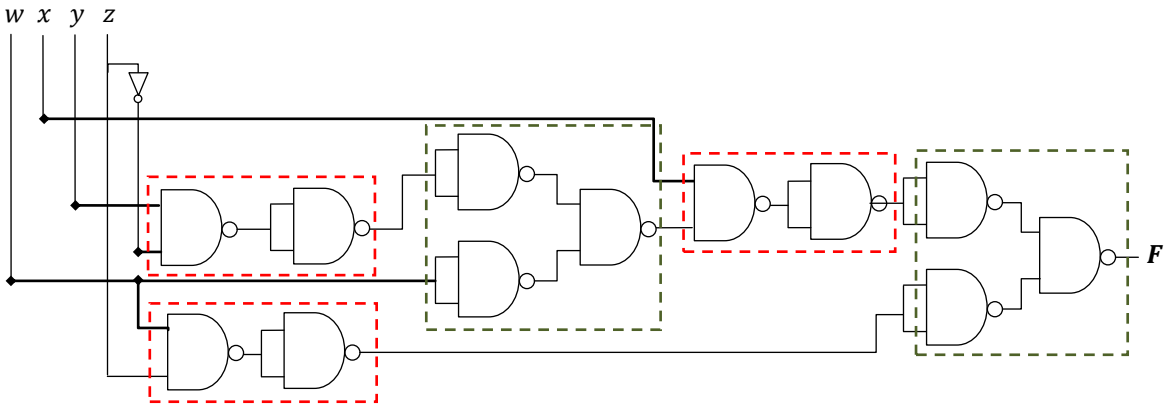
Remplacer chaque porte AND, OR et NOT par son bloc équivalent en portes NAND (Table 4.1). (2)Éliminer deux portes NAND successives à une seule entrée puisque une double inversion n'a aucun effet puis (3)éliminer les

portes NAND à une entrée reliée directement aux entrées du circuit et inverser ces entrées. Un exemple d'application est montré par le circuit donné par le logigramme contenant les portes AND, OR et NOT suivant :

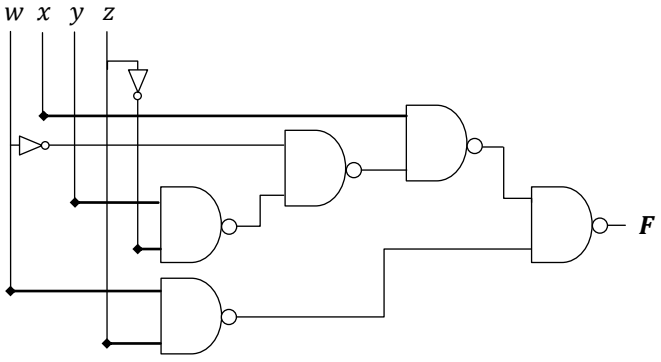
$$F(w,x,y,z) = w.z + x.\left(w + y.\overline{z}\right)$$



L'exécution de la première instruction (1) de la procédure d'implémentation donne le circuit suivant :



Après l'élimination des portes NAND à travers l'instruction (2) et (3), on obtient le circuit final suivant :

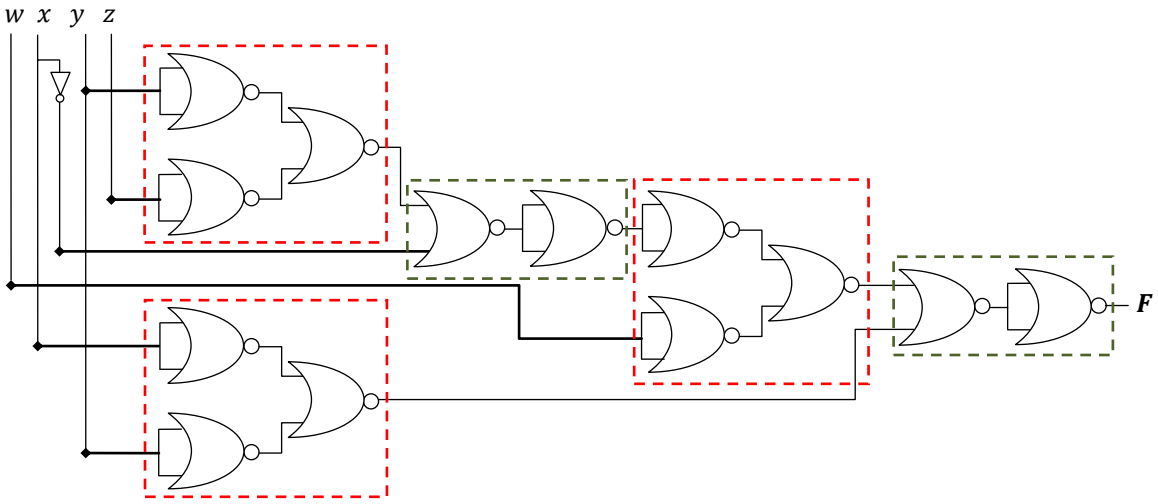


2) Procédures de conversion d'un circuit AND/OR/NOT en un circuit NOR

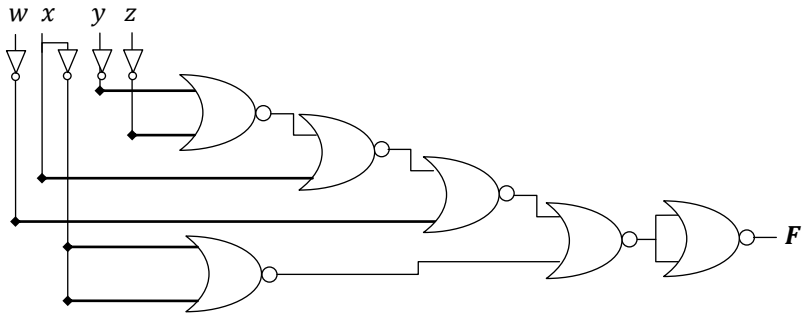
Similairement, (1) remplacer les portes logiques AND, OR et NOT par leurs blocs équivalents en portes NOR exhibés (Table 4.1) puis procéder à la simplification qui constitue à (2)supprimer deux portes NOR successives à une seule entrée consécutive et (3)éliminer les portes NOR à une entrée qui s'applique aux entrées du circuit tout en inversant ces dernières. Un exemple d'application de cette procédure est représenté par la fonction logique contenant les portes AND, OR et NOT suivante [8]:

$$F(w, x, y, z) = x.\overline{y} + w.\left(y.z + \overline{x}\right)$$

Le remplacement des portes AND, OR et NOT par leurs blocs équivalents en portes NOR en exécutant la première instruction (1) donne le circuit suivant :



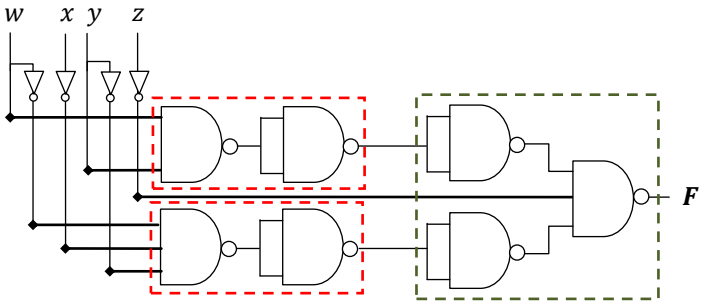
L'exécution des instructions de suppression (2 et 3) donne le circuit final suivant :

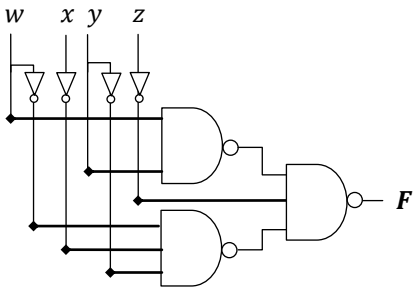


4.4. Implémentation à deux niveaux

Quand une fonction logique est exprimée sous forme de somme de produits, le circuit NAND équivalent est obtenu très facilement. Pour réaliser un circuit NAND à deux niveaux, il suffit de dessiner le logigramme du circuit de la fonction et remplacer les portes AND et la porte OR par des portes NAND. A titre d'exercice, on peut s'entrainer à trouver une implémentation à deux niveaux à partir de l'expression somme de produit de la fonction suivante :

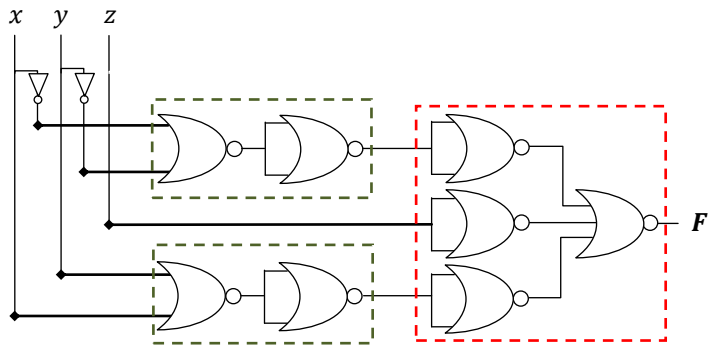
$$F(w,x,y,z) = w.y + \overline{z} + \overline{w}.\overline{x}.\overline{y}$$

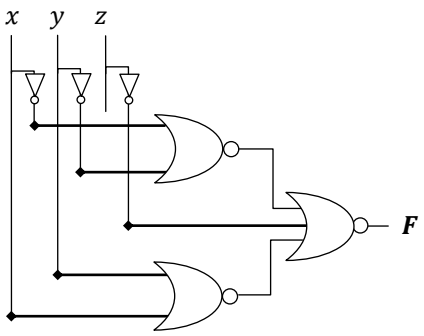




Parallèlement, un circuit NOR à deux niveaux est obtenu à partir d'une expression logique produit de sommes. L'implémentation consiste à dessiner le logigramme du circuit de la fonction en remplaçant les portes OR et AND par des portes NOR puis inverser les entrées du circuit qui s'appliquent au deuxième niveau.

$$F(x, y, z) = (x + y)\, z\left(\overline{x} + \overline{y}\right)$$





4.5. Synthèse au moyen de multiplexeur/démultiplexeur

Les multiplexeurs peuvent êtres aussi utilisés seuls dans les circuits combinatoires. L'exemple précédant peut être synthétisé à l'aide d'un multiplexeur 4 à 1. En considérant la table de vérité de la fonction logique à synthétiser, la même expression à trois variables peut être transformée en une expression à deux variables comme suit [8] :

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

x	*y*	*F*
0	0	0
0	1	z
1	0	1
1	1	$\overline{z}$

Cette égalité permet de synthétiser la fonction logique de la même manière montrée sur la figure suivante :

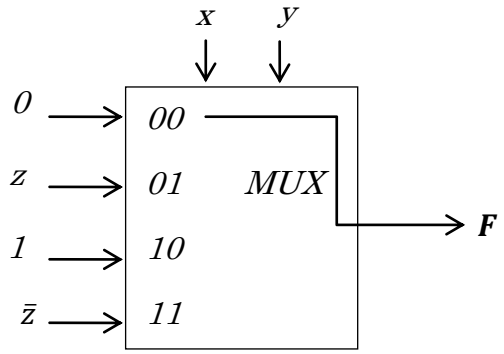


Figure 4.3 – Réalisation d'une fonction logique à l'aide d'un MUX 4 à 1

On peut aussi utiliser des démultiplexeurs en combinaison avec des portes logiques (très souvent des portes NAND) pour réaliser les fonctions logiques. Le multiplexeur sert dans ce cas à engendrer les maxtermes des fonctions booléennes. Le circuit suivant présente un démultiplexeur engendrant les maxtermes de fonctions à trois variables.

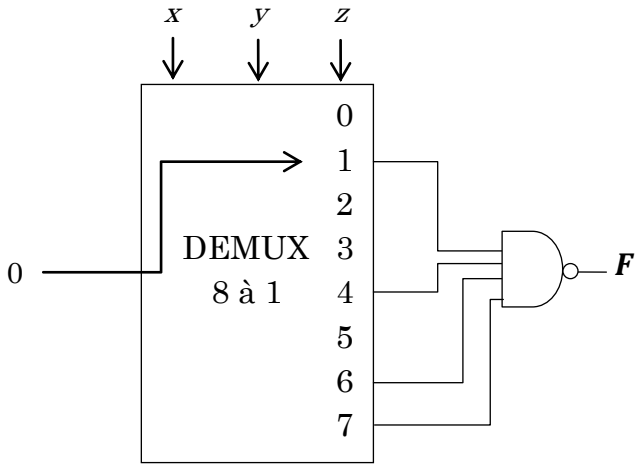


Figure 4.4 – Réalisation d'une fonction logique à l'aide d'un DEMUX 8 à 1

Le circuit ci-dessous montre comment réaliser une autre fonction logique à trois variables à l'aide d'un démultiplexeur 1 à 8 dont les sorties 1, 4, 6 et 7 sont reliées à la porte NAND.

4.6. Synthèse au moyen de PLA

Un PLA « Programmable Logic Array » est un circuit intégré consistant en deux matrices d'éléments programmables permettant l'implémentation de fonctions logiques sous forme de somme de produit. La programmation du PLA est la dernière étape de la fabrication de circuit [9].

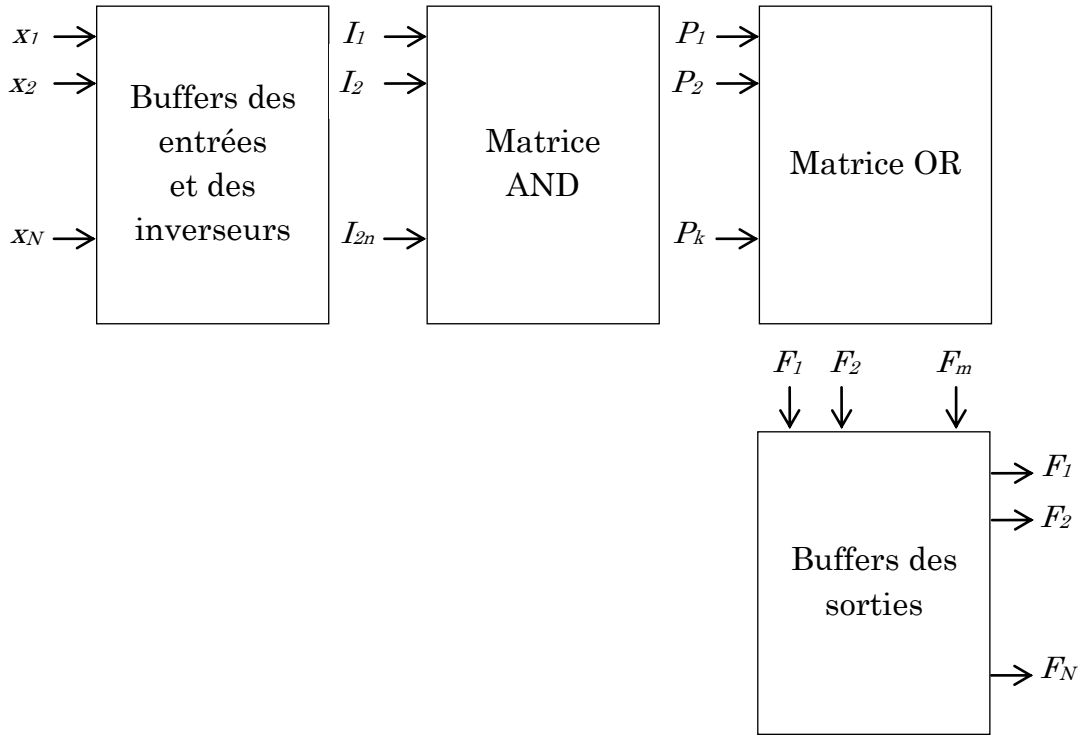


Figure 4.5 – Schéma d'un PLA

Le schéma d'un PLA est montré dans la figure ci-dessus. La matrice AND est un tableau de diodes ou de dispositifs analogues montés en circuits AND. De même, la matrice OR est un tableau de dispositifs montés en portes OR. Un PLA(n,k,m) est un circuit à n entrées permettant de réaliser au plus m fonctions logiques chacune à n variables au plus. Le nombre maximum de portes AND pouvant être formée est égal à k.

Les variables x_1, x_2, …, x_n ainsi que leurs compléments sont présentés au tableau des portes AND pour constituer au plus k termes. Ces termes sont ensuite introduits au tableau des portes OR afin de récupérer en sortie les expressions désirées.

La figure suivante présente un exemple d'un PLA*(4, 2, 2)* programmé pou réaliser les deux fonctions logiques suivantes :

$$F_1(x, y, z) = x.y + x.\overline{z} + \overline{x}.\overline{y}.z$$
$$F_2(x, y, z) = x.y + x.z + x.\overline{y}.\overline{z}$$

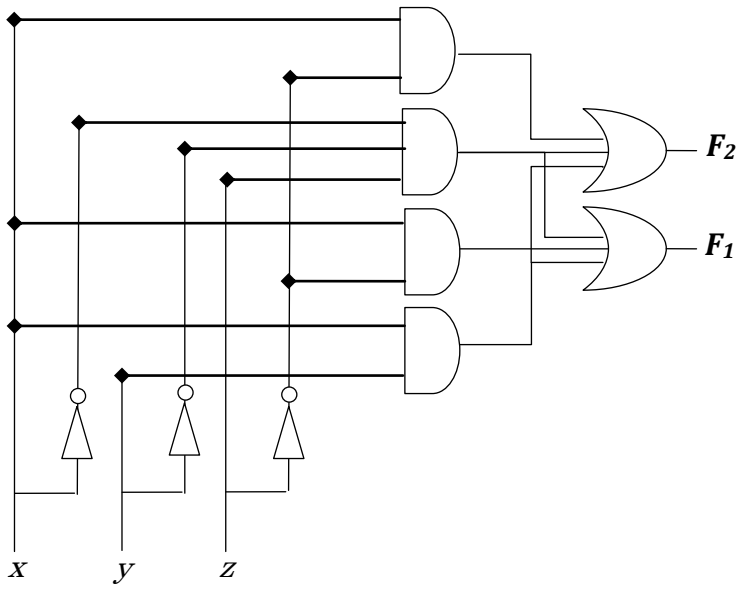
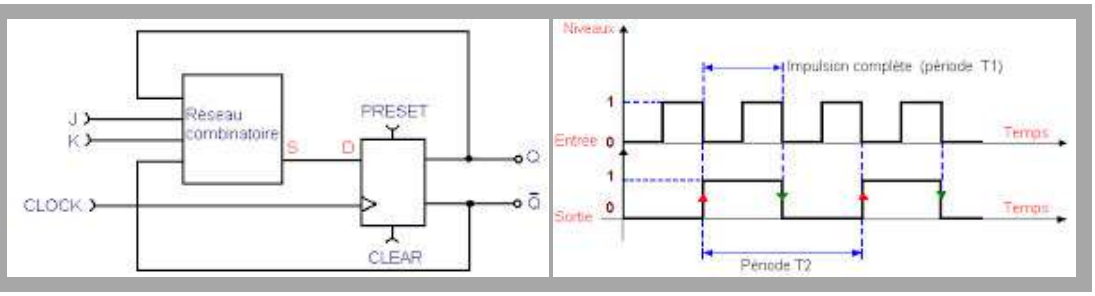


Figure 4.6 – Logigramme d'un PLA*(4,2,2)* réalisant deux fonctions logiques

Chapitre 5

Circuits Logique Séquentiels

- **Objectifs du chapitre :**
  - Savoir le principe de fonctionnement d'un système séquentiel. Connaître les différents circuits logiques séquentiels.
- **Dans ce chapitre :**
  - ✓ Définition d'un circuit séquentiel
  - ✓ Bascules
  - ✓ Compteurs (synchrone et asynchrone)
  - ✓ Registres (de mémorisation et à décalage)
  - ✓ Synthèse des circuits logiques séquentiels



5.1. Définition d'un circuit séquentiel

Souvent, la mémoire d'un ordinateur par exemple stocke les données et les instructions d'un programme en cours d'exécution, à l'aide des circuits de mémorisation capables de souvenir la valeur qu'ils enregistraient. Ces circuits sont appelés des circuits séquentiels. A l'inverse des circuits combinatoires, la valeur de sortie d'un circuit séquentiel ne dépend que des variables logiques d'entrée, mais dépend aussi de la valeur de sortie antérieure.

Un circuit séquentiel est un circuit logique englobant un circuit combinatoire et des éléments de mémoire appelés des Bascules. Le schéma général d'un circuit séquentiel est montré comme suit [1-2] :

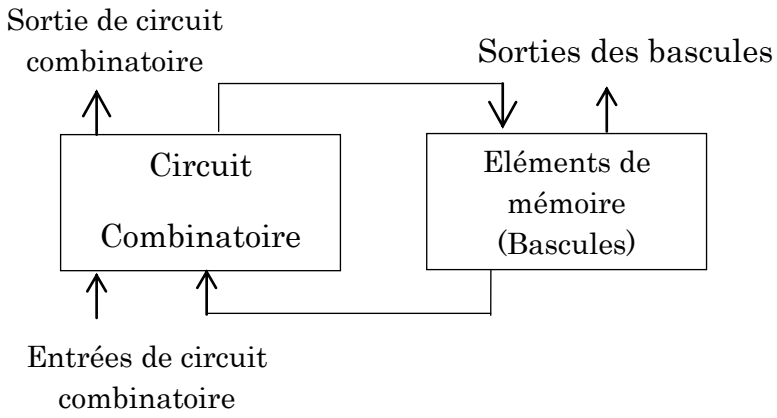


Figure 5.1 – Schéma d'un circuit séquentiel

Le circuit séquentiel accepte des signaux binaires provenant des entrées externes et aussi des unités de mémoire. Un dispositif de mémoire est un organe capable d'emmagasiner un bit d'information. L'information binaire enregistrée dans les unités de mémoire peut être transformée par le circuit combinatoire. Ce processus montre que les sorties externes du circuit séquentiel sont une fonction non seulement des entrées externes mais aussi de l'état présent des unités de mémoire. Un circuit séquentiel est spécifié par une séquence dans

le temps, des entrées, des sorties et des états externes. On désigne deux types de ce genre de circuit [3] :

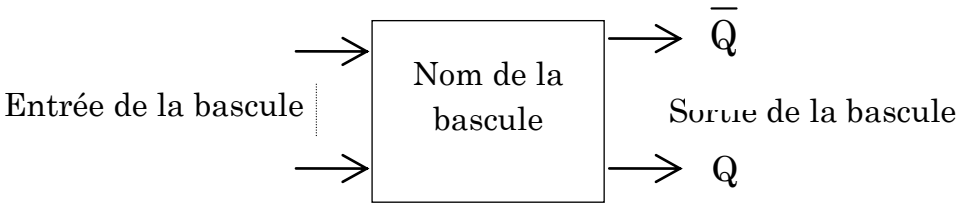
- **Circuits séquentiels synchrones :** ce sont des systèmes dont l'état est défini en fonction des signaux apparaissant en des périodes de temps régulières. Ces circuits sont utilisés généralement dans les horloges, sont donc appelés circuits séquentiels à horloge.

- **Circuits séquentiels asynchrones :** ils ne sont pas pilotés par une horloge l'état d'un tel circuit dépend de l'ordre d'apparitions des signaux en entrée.

Les unités de mémoire employées dans un circuit séquentiel sont connues par les **bascules** ou **flip-flop**.

5.2. Bascules

Les bascules sont des cellules binaires capables de stocker un bit d'information. Donc une bascule est un élément peut mémoriser un bit et le restituer au temps voulu. Les bascules diffèrent selon deux critères majeurs qui sont le nombre d'entrées au circuit de la bascule et la manière dont les entrées transforment l'état de la bascule [10].



Q : Est la variable de sortie de la bascule. Elle fournit l'état de la bascule si Q = 1, on dit que l'état de la bascule est à 1, sinon l'état de la bascule est à 0. $\overline{Q}$: est la variable inverse de la variable Q.

Figure 5.2 - Schéma fonctionnel d'une bascule

On désigne quarte types fondamentaux de bascules de base : R-S, J-K, D et T.

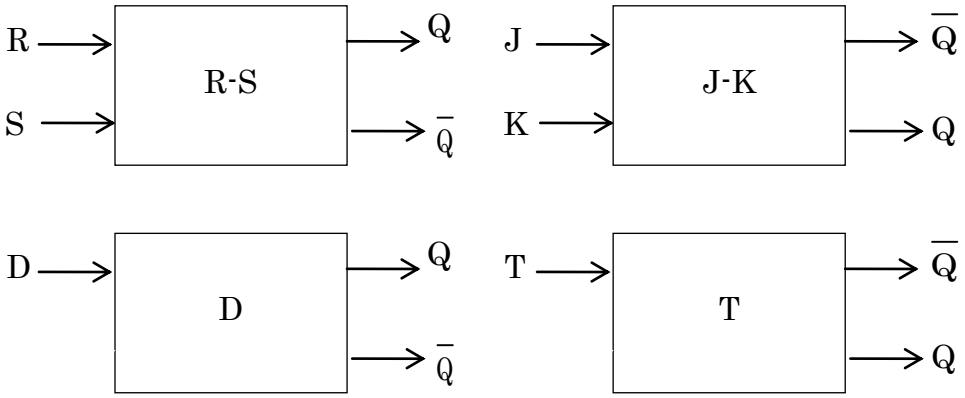


Figure 5.3 – Types de bascules de base

1) Bascule R-S

La bascule R-S est un circuit formé de deux portes logiques NOR ou NAND. Ce circuit possède deux entrées et deux sorties :

- **Entrées :** S pour la mise de 1 de la bascule. R pour la remise de 0 de la bascule.
- **Sorties :** Q et $\overline{Q}$.

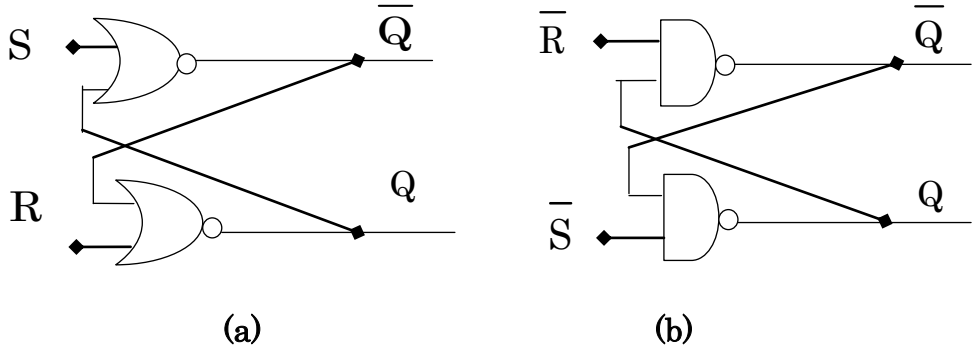


Figure 5.4 - Bascule R-S asynchrone réalisée avec deux portes (a) NOR, (b) NAND

Pour comprendre le fonctionnement de la bascule R-S, il faut étudier le comportement des variables de sortie en fonction des variables d'entrées. Pour cela on désigne :

- Q_t, la variable de la sortie à l'instant t (l'état présent de la bascule).
- Q_{t+1}, la variable de la sortie à l'instant t+1 (l'état futur de la bascule).

La table de vérité suivante résume le comportement de la bascule R-S :

R	S	Etat présent Q_t	Etat future Q_{t+1}	Comportement
0	0	0	0	$Q_{t+1} = Q_t$
0	0	1	1	
0	1	0	1	$Q_{t+1} = 1$ (Mise à 1)
0	1	1	1	
1	0	0	0	$Q_{t+1} = 0$ (Remise à 0)
1	0	1	0	
1	1	0	X	Etats indéterminés
1	1	1	X	

Le problème se pose quand l'une de deux entrées passe à 1, à ce moment là, la bascule peut prendre l'un des états 0 ou 1 selon que se soit S ou R qui change d'état en premier. On dit alors, que l'état de la bascule est indéterminé, il peut être à 1, comme il peut être à 0.

L'équation caractéristique de la bascule R-S est obtenue à l'aide du tableau de Karnaugh est donc :

R \ S Q_t	00	01	11	10
0	0	1	1	1
1	0	0	Φ	Φ

$$Q_{t+1} = \overline{R}.Q_t + S$$

Tout au long de cette étude, le facteur temps de propagation des signaux n'était pas pris en compte. C'est le cas d'une bascule R-S asynchrone. Alors pour une bascule R-S synchrone, l'utilisation d'une

horloge (ou d'une variable quelconque) permet de synchroniser les changements d'état de la bascule. En effet, il est souvent utile de ne faire changer d'état à une bascule qu'à des instants précis.

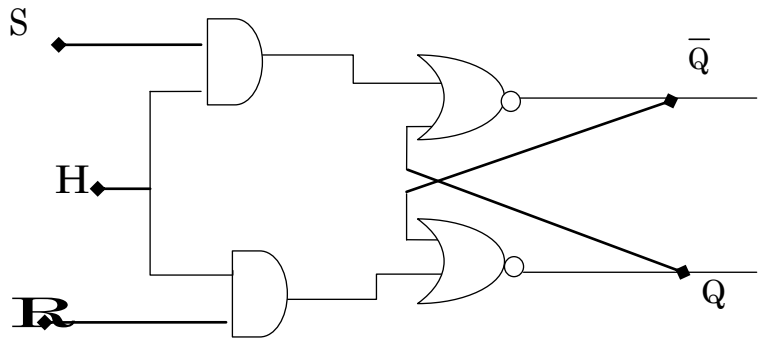


Figure 5.5 - Logigramme d'une bascule R-S synchronisée

Le schéma ci-joint illustre la bascule R-S commandée par une horloge H, à partir de ce schéma, on constate ce qui suit :

- Si H = 0, les valeurs de R et S n'ont aucun effet sur la bascule. Cette dernière garde l'état antérieur.
- Si H = 1, à ce moment-là, la bascule se retrouve sous le contrôle des deux variables R et S.

La table caractéristique de cette bascule synchrone est la suivante :

H	R	S	Q_t	Q_{t+1}	Comportement
0	0	0	0	0	$Q_{t+1} = Q_t$ L'état de la bascule ne change pas. Les valeurs de R et S n'ont aucun effet, tant que H = 0
0	0	0	1	1	
0	0	1	0	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	1	1	
1	0	0	0	0	$Q_{t+1} = Q_t$
1	0	0	1	1	

1	0	1	0	1	Q_{t+1} = 1 (Mise à 1)
1	0	1	1	1	
1	1	0	0	0	Q_{t+1} = 0 (Remise à 0)
1	1	0	1	0	
1	1	1	0	X	Etats indéterminés
1	1	1	1	X	

2) Bascule J-K (Jump Kill)

La bascule J-K diffère de la bascule R-S du fait que, quand les deux variables d'entrée passent simultanément à 1. L'état de la bascule n'est pas indéterminé.

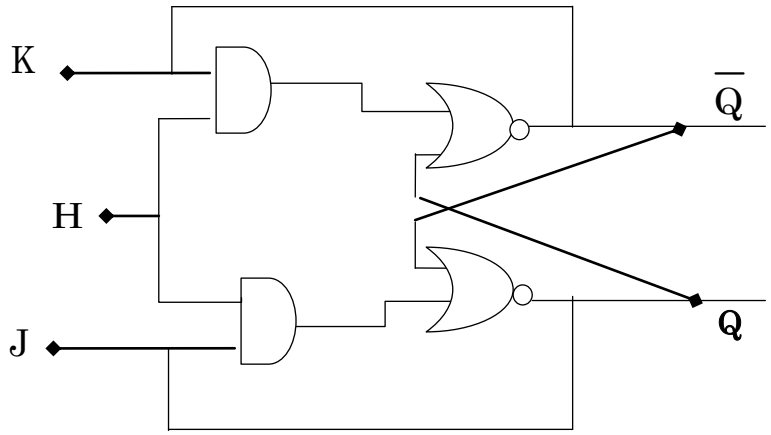


Figure 5.6 - Logigramme d'une bascule J-K synchrone

En effet, quand J = K = 1, la fonction de complémentation devient $Q_{t+1} = \overline{Q_t}$.

Pour une bascule asynchrone, la table caractéristique est la suivante :

J	K	Etat présent Q_t	Etat future Q_{t+1}	Comportement
0	0	0	0	$Q_{t+1} = Q_t$
0	0	1	1	
0	1	0	0	Q_{t+1} = 1 (Mise à 1)
0	1	1	0	

1	0	0	1	$Q_{t+1} = 0$ (Remise à 0)
1	0	1	1	
1	1	0	1	$Q_{t+1} = \overline{Q_t}$ (Complémentation)
1	1	1	0	

L'équation caractéristique de la bascule J-K est obtenue à l'aide du tableau de Karnaugh comme suit :

S Q_t / R	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$$Q_{t+1} = \overline{K}.Q_t + J.\overline{Q_t}$$

A partir de cette expression, on peut élaborer la table suivant qui est d'une très grande utilité lors de la réalisation d'un circuit séquentiel à base de bascules J-K :

Etat présent Q_t	Etat future Q_{t+1}	J	K	Comportement
0	0	0	X	La que peut prendre K n'a aucun effet sur l'état de la bascule.
0	1	1	X	
1	0	X	0	La que peut prendre J n'a aucun effet sur l'état de la bascule.
1	1	X	0	

3) Bascule D (Data)

Une autre manière de résoudre le problème d'ambiguïté rencontrée avec la bascule R-S lorsque R = S = 1, est de faire en sorte que ce cas ne se présente jamais à l'entrée de la bascule.

Pour cela, on utilise qu'une seul variable d'entrée externe D et on parle alors de la bascule D. Elle est illustrée par le schéma suivant :

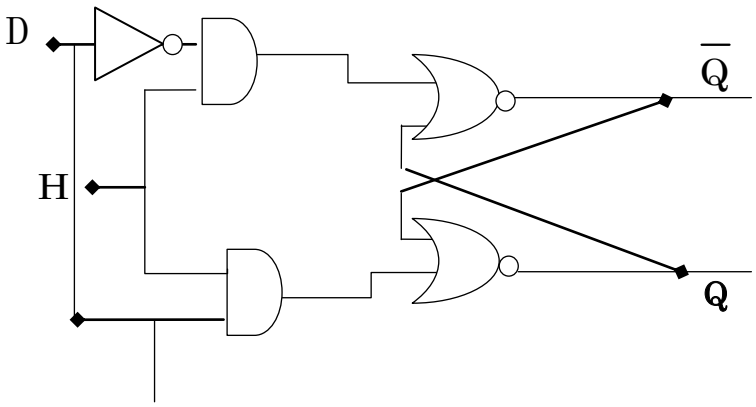


Figure 5.7 - Logigramme d'une bascule D synchrone

La variable d'entrée de la porte AND supérieure étant égale à $\overline{D}$, les entrées internes de la bascule ne peuvent jamais être identiques.

- Si D = 1, alors la bascule passe à l'état 1.
- Si D = 0, alors la bascule passe à l'état 0.

Il faut noter toutefois, que la bascule D n'a d'effet que si H est à l'état 1. Il faut aussi s'ensuit que la bascule D représente bien une mémoire de 1 bit.

La mémorisation du bit présent à l'entrée D prend effet dés qu'une impulsion est appliquée à l'entrée d'horloge H.

La table caractéristique de la bascule D asynchrone est comme suit:

D	H	Etat présent Q_t	Etat future Q_{t+1}	Comportement
0	0	0	0	Q_{t+1} = Q_t, Pas de changement
0	0	1	1	
0	1	0	0	Q_{t+1}= D = 0 (La donnée D passe)
0	1	1	0	
1	0	0	0	Q_{t+1} = Q_t, Pas de changement
1	0	1	1	
1	1	0	1	Q_{t+1}= D = 1 (La donnée D passe)
1	1	1	1	

L'équation caractéristique de la bascule D est obtenue à l'aide du tableau de Karnaugh, à partir de cette table par :

D \ H Q_t	00	01	11	10
0	0	1	0	0
1	0	1	1	1

$$Q_{t+1} = \overline{H}.Q_t + D.H$$

4) Bascule T (Toggle)

La bascule T ressemble à une bascule J-K à une seule entrée.

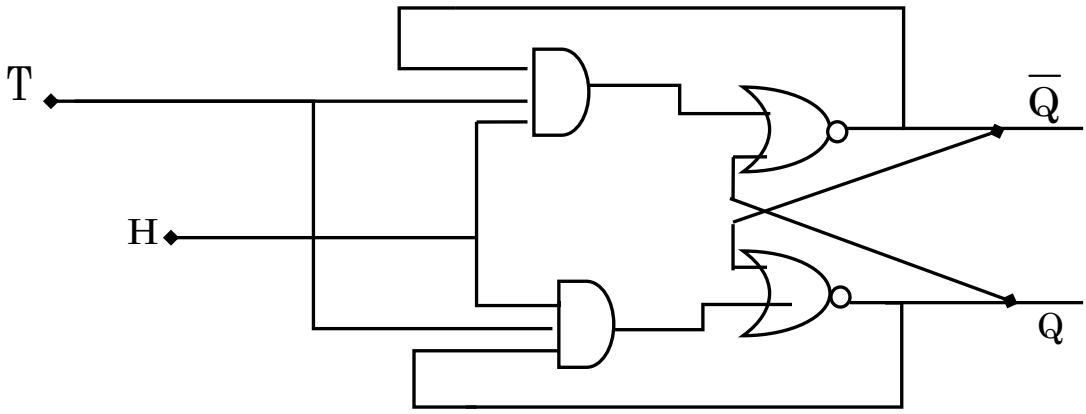


Figure 5.8 - Logigramme d'une bascule T synchronisée

Chaque fois qu'une impulsion arrive, les états de la bascule (Les sorties) sont inversées (Compléments) :

- □ Si T = 0, alors pas de changement.
- □ Si T = 1, à ce moment là, il ya une complémentation des variables de sorties.

La table caractéristique de la bascule T est comme suit :

T	Etat présent Q_t	Etat future Q_{t+1}	Comportement
0	0	0	$Q_{t+1} = Q_t$, Pas de changement
0	1	1	
1	0	1	$Q_{t+1} = \overline{Q_t}$ (Complémentation)
1	1	0	

L'équation caractéristique de la bascule T est obtenue à l'aide du tableau de Karnaugh, à partir de cette table par :

Qt \ T	0	1
0	0	1
1	1	0

$$Q_{t+1} = T.\overline{Q_t} + \overline{T}.Q_t = T \oplus Q_t$$

5.3. Déclenchement d'une bascule

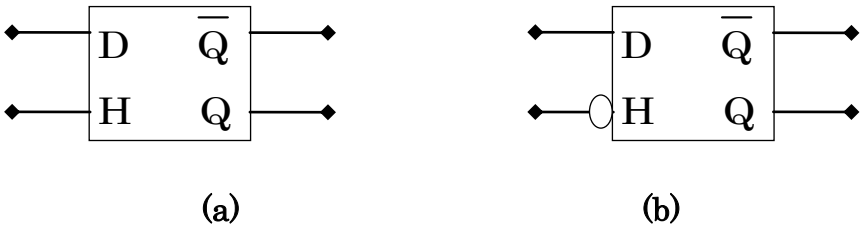
Le déclanchement d'une bascule se traduit par un changement momentané de ses variables d'entrée. Cependant, le déclanchement d'une bascule asynchrone diffère du déclanchement d'une bascule synchrone. En effet, dans le premier type (synchrone), la bascule est déclenchée lorsque les signaux appliqués en entrée changent. Dans le second type (asynchrone), le déclanchement est plus compliqué. Une bascule synchrone est pilotée par une horloge. Le déclanchement d'une telle bascule est provoqué par des impulsions. De ce fait, on distingue deux types de bascules synchrones : Latchs et flip-flops.

1) Bascule Latch

Les bascules Latchs régissent sur niveau d'horloge. Cela revient à dire que ce type de bascule est déclenché quand l'horloge H = 1 (niveau haut) ou quand H = 0.

- **Exemple étudié d'une bascule D-Latch déclenchée au niveau haut (H = 1)**

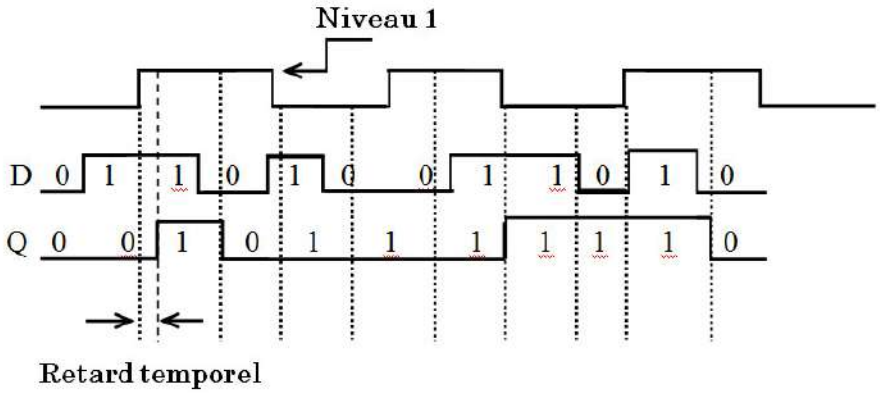
Le schéma standard d'une bascule D-Latch est comme suit :



(a) (b)

Figure 5.9 - Bascule D Latch déclenchée quand (a) H = 1, (b) H = 0

Le but est de visualiser le fonctionnement de la bascule D-Latch déclenchée au niveau haut par un chronogramme : Soit la séquence de bits à se présenter à l'entrée D de la bascule « 01101001101 », quelle sera la séquence de bits produite à la sortie Q de la bascule ?

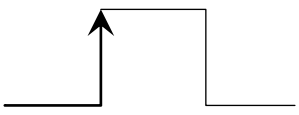


Signal d'horloge

Le signal appliqué à l'entrée D n'est pris en considération que lorsque le signal d'horloge est au niveau 1, sinon, la bascule garde son état antérieur. Il est observé que les signaux produits en sortie subissent un retard temporel Δt.

2) Bascule flip-flop

Ce type de bascule change d'état non pas d'horloge est au niveau **1** (ou **0**), mais, pendant la transition du signal d'horloge de l'état **0** à **1** (front montant) ou lors de la transition de l'état **1** à **0** (front descendant).



Front montant d'un signal horloge

Front descendant d'un signal horloge

✓ **Bascule J-K flip-flops:** Le schéma standard d'une bascule **J-K** flip-flops est comme suit :

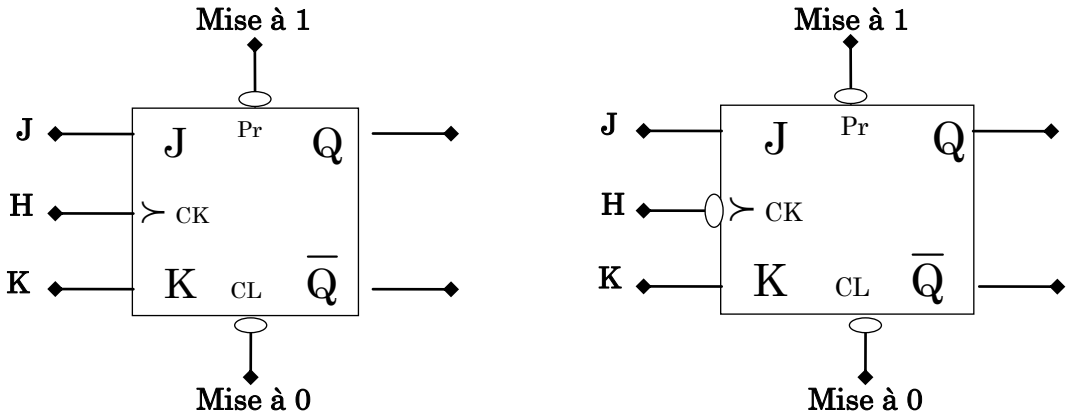


Figure 5.10 - Bascule J-K flip-flop déclenchable sur front (a) montan, (b) descendant

✓ **Bascule D flip-flop :** Le schéma standard d'une bascule **D** flip-flops est comme suit :

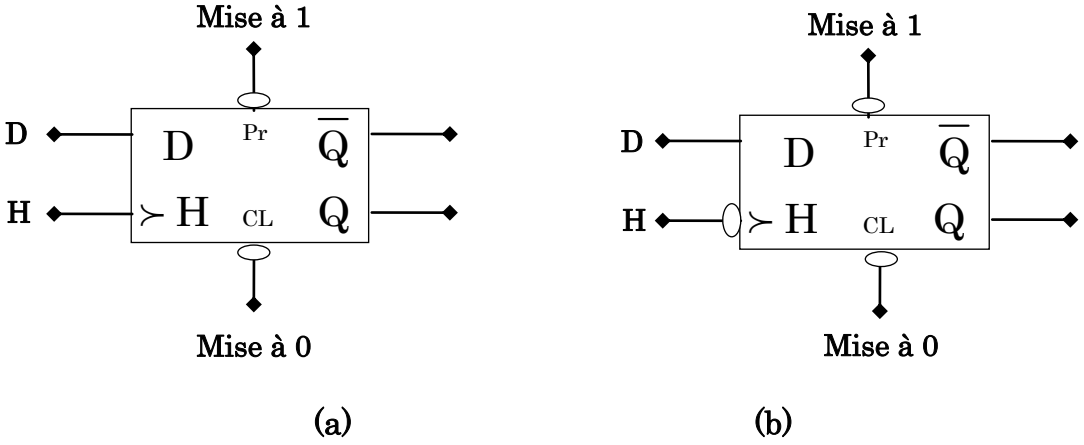
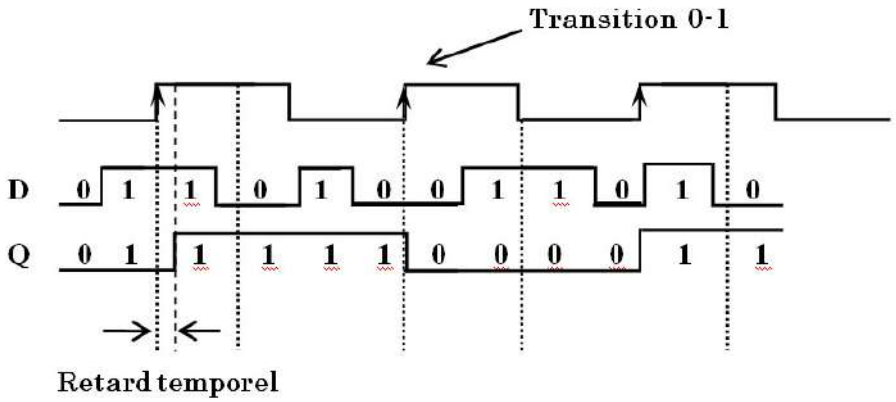


Figure 5.11 - Bascule D flip-flop déclenchable sur front (a) montan, (b) descendant

- **Exemple étudié d'une Bascule D flip-flop déclnchable sur front montant**

Considérant la même séquence 01101001010 de bits à se présenter à l'entrée D de la bascule :



Signal d'horloge

Le signal appliqué à l'entrée **D** n'est pris en considération qu'au moment de la transition du signal d'horloge de **0** à **1**.

5.4. Compteurs

Les compteurs sont des circuits séquentiels existent dans presque tous les équipements basés sur la logique digitale. Ils sont utilisés pour compter le nombre d'occurrence d'un événement et sont utilisés pour engendrer des séquences de temps pour contrôler des opérations dans un système digital. On désigne plusieurs types de compteurs sont [11] :

1) Compteur binaire

Un compteur binaire est un circuit séquentiel composé d'une succession de bascules, capables de sauvegarder le nombre d'impulsions électriques délivrées par une horloge extérieure ou un quelconque signal de comptage, en le transformant en un nombre binaire pour son usage ou affichage intérieur.

- **Exemple**

Pour mesurer la fréquence d'un signal, le nombre d'impulsions de ce signal est compté pendant une durée est égale à une seconde.

2) Compteur progressif

Un compteur binaire est dit progressif, si son contenu passe d'une valeur binaire m à la valeur m+1 (sens croissant) après application d'une impulsion d'horloge.

- **Exemple**

Compteur binaire formé de trois bascules et comptant de 0 jusqu'à 7, la table de vérité des transitions d'un tel compteur après chaque impulsion d'horloge est la suivante :

Valeur	Contenu du compteur
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Table 5.1 - Table de vérité d'un compteur binaire progressif comptant de 0 jusqu'à 7

3) Compteur régressif

Un compteur binaire est dit régressif, si son contenu passe d'une valeur binaire **m** à la valeur **m-1** (sens décroissant) après application d'une impulsion d'horloge.

- **Exemple**

Compteur binaire formé de trois bascules et comptant de 7 jusqu'à 0, la table de vérité des transitions d'un tel compteur après chaque impulsion d'horloge est la suivante :

Valeur	Contenu du compteur
7	111
6	110
5	101
4	100
3	111
2	010
1	001
0	000

Table 5.2 - Table de vérité d'un compteur binaire régressif comptant de 7 jusqu'à 0

4) Compteur modulo N

Un compteur binaire constitué de n bascules est dit modulo N (tel que $N \leq 2^n$), s'il peut compter jusqu'à N-1. La $N^{ième}$ impulsion le remet, obligatoirement, à zéro.

Les n étages constituant un tel compteur permettent de présenter tous les états possibles ($N=2^n$).

Si un certain nombre d'états ne seront jamais utilisés, en fonctionnement normal ($N < 2^n$), on parle d'un compteur incomplet.

- **Exemple**

Compteur binaire formé de trois bascules et comptant de 0 jusqu'à 5 (compteur incomplet), la table de vérité des transitions d'un tel compteur après chaque impulsion d'horloge est:

Valeur	Contenu du compteur
0	000
1	001
2	010
3	011
4	100
5	101
6	X
7	X

X : désigne une valeur indifférente

Table 5.3 - Table de vérité d'un compteur binaire incomplet (modulo 5)

4.5. Registres

Un registre est le deuxième type des circuits séquentiels (synchrone), capable de stoker, temporairement, des informations binaires, en utilisant un ensemble de bascules. Dont, on désigne deux types de registres : registre de mémorisation et registre à décalage [2].

1) Registre de mémorisation

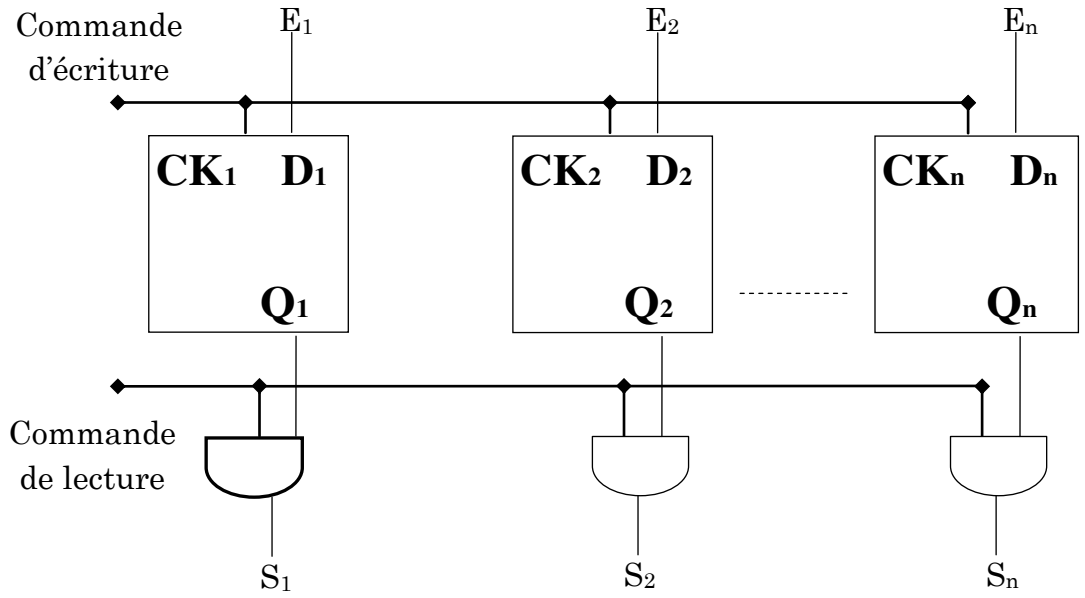


Figure 5.12 - Registre de mémorisation formé de bascules de type D de front montant

Le registre de mémorisation est un registre capable de réaliser la fonction de mémorisation en emmagasinant une information binaire sous forme d'un mot de n bits. Le schéma d'un tel registre, en utilisant des bascules de type D à front montant est montrée par la figure ci-dessus.

2) Registres à décalage

Un registre à décalage est un registre ayant la possibilité de décaler à droite ou à gauche son contenu.

- ✓ **Registre à décalage à droite :** Il est composé de n bascules interconnectées de façon à ce que l'état logique de la sortie Q_i de la i ème bascule soit reproduit à la sortie Q_{i+1} de la (i+1) ème bascule quand un signal d'horloge est appliqué à l'ensemble de bascules. Ce type de registre à décalage possède une seule entrée à gauche E_g et n sorties (Q_1, Q_2,…, Q_n).

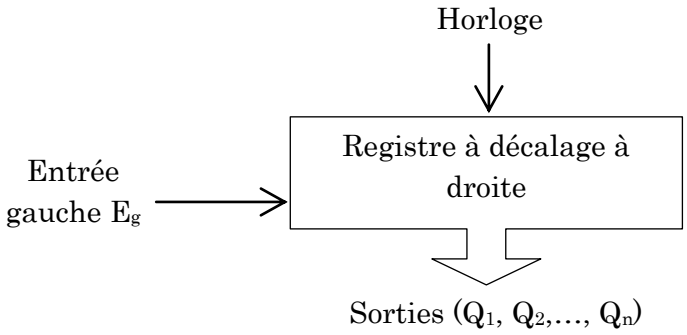


Figure 5.13 - Organisation d'un registre à décalage à droite

- **Exemple**

Registre à décalage à droite formé de quatre bascules de type **D** à front montant. Les expressions algébriques des entrées :

- □ $Q_i = Q_{i+1}^+$ (caractéristique du décalage à droite).
- □ $Q_{i+1}^+ = D_{i+1}$(caractéristique d'une bascule de type D).

A partir de ces deux égalités, on déduit que $D_{i+1} = Q_i$ d'où $D_4 = Q_3$, $D_3 = Q_2$, $D_2 = Q_1$ et $D_1 = E_g$.

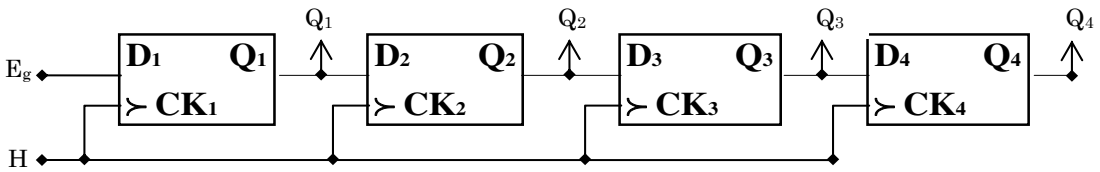


Figure 5.14 - Registre à décalage à droite formé de quatre bascules de type D de front montant

- ✓ **Registre à décalage à gauche :** Il est composé de n bascules interconnectées de façon à ce que l'état logique de la sortie Q_{i+1} de la (i+1) ème bascule soit reproduit à la sortie Q_i de la i ème bascule quand un signal d'horloge est appliqué à l'ensemble de bascules. Ce type de registre à décalage possède une seule entrée à droite E_d et n sorties (Q_1, Q_2,…, Q_n).

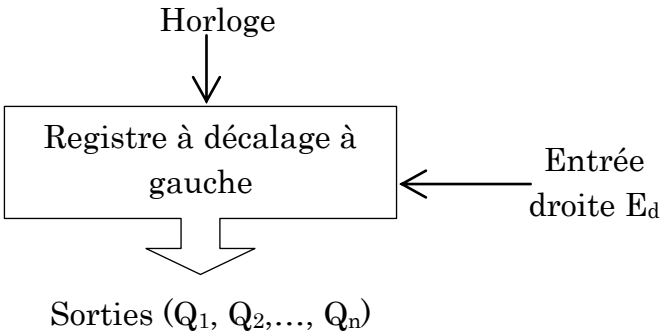


Figure 5.15 - Organisation d'un registre à décalage à gauche

- **Exemple**

Registre à décalage à gauche formé de quatre bascules de type **D** à front montant. Les expressions algébriques des entrées :

- □ $Q_{i+1} = Q_i^+$ (caractéristique du décalage à gauche).
- □ $Q_i^+ = D_i$ (caractéristique d'une bascule de type D).

A partir de ces deux égalités, on déduit que $D_i = Q_{i+1}$ d'où : $D_1 = Q_2$, $D_2 = Q_3$, $D_3 = Q_4$, $D_4 = E_d$.

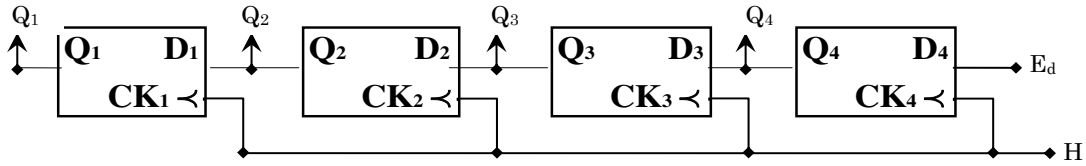


Figure 5.16 - Registre à décalage à gauche formé de quatre bascules de type D de front montant

- ✓ **Registre à décalage à droite ou à gauche :** C'est la composition des deux registres précédents en ajoutant une entrée supplémentaire pour la sélection du sens de décalage. Ce type de registre à décalage possède n sorties (Q_1, Q_2,…, Q_n) et deux entrées E_d et E_g telles que E_d est l'entrée à droite et E_g est l'entrée à gauche.

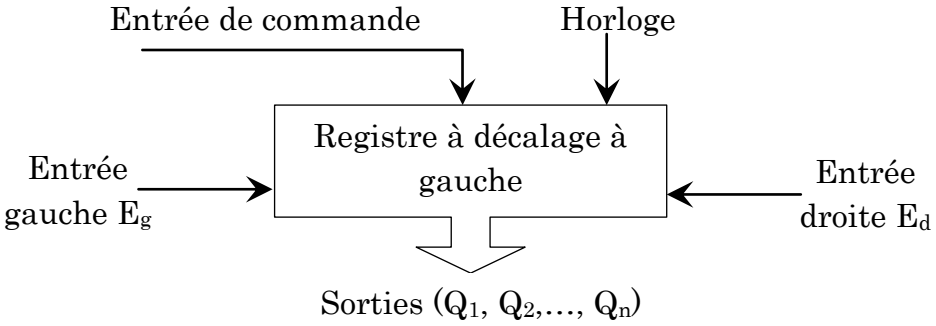


Figure 5.17 - Organisation d'un registre à décalage à droite ou à gauche

3) **Types de registres à décalage :** On désigne quatre types de registres à décalage :

   - □ **Registre à entrée série et sortie série :** Dans ce type de registre l'information est introduite bit par bit et récupérée de la même façon.

- **Registre à entrée série et sortie parallèle :** Dans ce type de registre l'information est introduite bit par bit et récupérée en parallèle en un seul bloc.

- **Registre à entrée parallèle et sortie série :** Dans ce type de registre l'information est introduite en un seul bloc et récupérée mais ne peut être récupérée que bit par bit.

- **Registre à entrée parallèle et sortie parallèle :** Dans ce type de registre l'information est introduite en un seul bloc et récupérée de la même façon.

Références

Bibliographiques

1. GOZIM Djamal et GUESMI Kamel. LOGIQUE COMBINATOIRE ET SEQUENTIELLE. Polycopié Pédagogique. Université ZIANE ACHOUR de Djelfa. Faculté de Science et de la Technologie. Département de Génie Electrique. Algérie. 2019.
2. AMIMEUR Hocine. LOGIQUE COMBINATOIRE ET SEQUENTIELLE. Polycopié Pédagogique. Université ABDERRAHMANE MIRA de Bejaia. Faculté de Technologie. Département de Génie Electrique. Algérie. 2017.
3. M. SBAI. ELECTRONIQUE NUMERIQUE. lOGIQUE COMBINATOIRE ET COMPOSANTS NUMERIQUE, Ellipses Editions Marketing. Paris. France. 2013.
4. M. BELAID et collectif. LOGIQUE COMBINATOIRE ET SEQUENTIELLE, Presses de Mitidja. Baraki. Alger. Algérie. 2010.
   M. K. MOKHTARI et I. CAID. DE L'ALGEBRE DE BOOLE AUX CIRCUITS NUMERIQUES, COURS ET APPLICATIONS. Office des Publications Universitaires. Alger. Algérie. 2010.
5. S. MERZOUK, H. BOUZOURANE, A.

AMAROUCHE et D. HAMOUDI. LOGIQUE COMBINATOIRE ET SEQUENTIELLE. Pages Bleues Internationale. Algérie. 2010.

6. L. MUSEUR, ELECTRONIQUE NUMERIQUE « LOGIQUE COMBINATOIRE ET SEQUENTIELLE ». Université de Paris 13. Institut Galilée. France. 2007.

7. M. BELAID, S. MERZOUK et H. BOUZOURANE. LES CIRCUITS LOGIQUES. Pages Bleues Internationale. Algérie. 2007.

8. H. DRIAS-ZEKAOUI. INTRODUCTION A) L'ARCHITECTURE DES ORDINATEURS. Office des Publications Universitaires. Alger. Algérie. 2006.

9. J. J. MERCIER, BIT PAR BIT : NUMERATION, BINAIRE, LOGIQUE COMBINATOIRE, Ellipses Editions Marketing, Paris, France, 2005.

10. C. ALEXANDRE. CIRCUITS NUMERIQUES, POLYCOPIE DE COURS ELECTRONIQUE. Conservatoire national des arts et métiers, France. 2004.

11. C. AMEUR YAHIA. LOGIQUE ET CALCULATEURS. Edition d'Abeille. Algérie. 2003.

- *La logique combinatoire et séquentielle constitue une matière essentielle aux étudiants de la filière de Télécommunications, Electronique et Génie Biomédical.*
- *Elle peut porter aussi l'intitulé de « Logique et Calculateurs » et peut servir comme introduction nécéssaire à l'architecture des ordinateurs et microprocessuers de plus les autres interdéciplines de l'Electronique Numérique (DSP, FPGA...).Ses concepts théoriques constituent un support faondamental pour la constitution des systèmes digitaux et numériques.*
- *De manière principale, cette matière à pour objectif : En permier ordre, étudier les circuits logiques combiatoires et savoir concevoir quelques applications pour ces circuits en utilisant les outils standards de l'analysse. En deuxième ordre, synthèse des différents circuits logiques séquentiels à savoir les compeuteurs et registres.*

LOGIQUE	**Combinatoire**	**Université de Saîda - Dr. Moulay Tahar -**	Faculté de Téchnologie	**Département d'Electronique**
Séquentielle	**U.E.F 2.2.1** Coeff. 2 – Créd.4 60% EX – 40% C.C	**2ème année TLC-ELN-GBM - Semestre** 4		**Présenté par :** Dr. Abdelhakim BOUDKHIL
		Présentation de la matière		

Description

- *Elle constitue une matière essentielle aux étudiants de la filière de Télécommunications, Electronique et Génie Biomédical.*
- *Elle peut porter aussi l'intitulé de « Logique et Calculateurs » et peut servir comme introduction nécéssaire à l'architecture des ordinateurs et microprocessuers de plus les autres interdéciplines de l'Electronique Numérique (DSP, FPGA...).Ses concepts théoriques constituent un support faondamental pour la constitution des systèmes digitaux et numériques.*

Objectifs

De manière principale, cette matière à pour objectif :

- *Etudier les circuits logiques combiatoires et savoir concevoir quelques applications pour ces circuits en utilisant les outils standards de l'analysse.*
- *Synthèse des circuits logiques séquentiels.*

Définition

La logique (mathématique) est une théorie introduite par Moses Schonfinkel baséée sur l'Algèbre de Boole qui pour objectif de minimiser le nombre d'opérations dans le calcul des prédicats, utilisée par les logiciens pour développer (gérer) les langages de programmation (machine).

En logique combinatoire, les résultats sont fonctions des données actuellement traitées, contrairement à la logique séquentielle où ils relient aux données précédemment traitées. Les électroniciens doivent maitriser la logique com & séq pour savoir concevoir et réaliser les différents circuits numériques.

Contenu général

Le contenu de cette matière peut-être divisé en deux grandes parties :

- *La 1ère partie décrit les concepts théoriques de l'Algèbre de Boole nécessaire à la conception (analyse et synthèse) des circuits logiques à savoir la manipulation algébrique et la simplification logique des différentes opérations et fonctions. Cette partie demande la connaissance préalable des systèmes de numération, arithmétique binaire, codage de l'information et l'Algèbre de Boole.*
- *La 2ème partie étudie les circuits logiques combinatoires en premier temps tels que : additionneur, soustracteur, comparateur, encodeur, décodeur, multiplexeur, démultiplexeur… et par la suite les circuits logiques séquentiels tels que : compteur, registre à décalage….*

En résumé, cette matière va aborder :

1. *Algèbre Booléenne, fonctions logiques binaires et méthodes de simplifications.*
2. *Systèmes de numération et codage de l'information.*
3. *Circuits logiques combinatoires fondamentaux.*
4. *Analyse et Synthèse des circuits logiques séquentiels.*

Références principales

- *Mc. Belaid et collectif, Logique combinatoire et séquentielle, Presses de Mitidja, Baraki, Alger, Algérie, 2010.*
- *M. Sbaı, Electronique numérique, logique combinatoire et composants numérique, Ellipses Editions Marketing, Paris, France, 2013.*
- *MERZOUK S, BOUZOURANE H, AMAROUCHE A, HAMOUDI D, Logique combinatoire séquentielle. Les pages Bleues Internationale 2010.*
- *J. J. Mercier, Bit par bit : numération, binaire, logique combinatoire, Ellipses Editions Marketing, Paris, France, 2005.*
- *M. K. Mokhtari, I. Caid De l'algèbre de Boole aux circuits numériques, cours et applications, Office des Publications Universitaires, Alger, Algérie, 2010.*
- *C. Alexandre, Circuits numériques, Polycopié de cours électronique, Conservatoire national des arts et m´etiers, France, 2004.*
- *Luc MUSEUR, Electronique numérique 'Logique combinatoire et séquentielle', Université Paris 13, Institut Galilée. 2007.*