REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

École supérieure en génie électrique et énergétique d'Oran

Département du Second Cycle

Polycopié de Cours

Électronique Numérique

Présenté par :

Dr. SELAMI Nada.

Année Universitaire

2021 – 2022

Avant-Propos :

*Ce polycopié est destiné particulièrement aux étudiants de la première année (deuxième cycle) de l'**E**cole **S**upérieure en **G**énie **E**lectrique et **E**nergétique (**ESGEE**) d'Oran. Il constitue un manuel de cours qui traite des principes de base de l'électronique numérique et permet aux étudiants de comprendre les différentes notions des circuits numériques, des portes logiques et des machines d'états.*

Ce manuel introduit les techniques permettant d'établir de manière simple et concise les équations logiques, ainsi que les méthodes d'analyse et de conception des circuits numériques. Il présente les aspects liés à la logique combinatoire et à la logique séquentielle et met particulièrement l'accent sur les approches de conception, qui peuvent être utilisées pour assurer un fonctionnement fiable des machines d'états. Chaque chapitre comprend des exemples pratiques bien conçus.

Ce polycopié se compose de six chapitres, consacrés aux principes de la logique combinatoire et séquentielle, ainsi que les techniques sous-jacentes pour l'analyse et la conception des circuits numériques.

Chapitre I Systèmes de numération et Opérations arithmétiques

Chapitre II Algèbre de Boole et fonctions logiques

Electronique numérique I

-Logique Combinatoire-

Chapitre I

Systèmes de numération et

Opérations arithmétiques

1 Introduction

L'étude des systèmes numériques est importante du point de vue de la compréhension de la représentation des données avant de pouvoir être traitées par tout système numérique. C'est l'un des sujets les plus fondamentaux de l'électronique numérique.

Ce chapitre présente les différents systèmes de numération et leurs conversions adaptés à la représentation de l'information. Des exemples sont donnés pour l'addition, la soustraction de nombres binaires (signés et non signés) et de nombres décimaux au format BCD ainsi que la multiplication et la division binaire.

2 Systèmes de numération et Codage de l'information

Les systèmes numériques sont importants pour la compréhension des modes de représentation des données avant de pouvoir les traiter par n'importe quel système numérique.

Dans cette section, nous abordons les différents systèmes numériques couramment utilisés pour représenter les données en décrivant brièvement les paramètres qui sont communs à tous les systèmes numériques.

2.1 Système de Numération

Les différentes caractéristiques qui définissent un système de numération comprennent : le nombre de chiffres utilisés dans le système de numération, le poids des différents chiffres qui constituent le nombre et les nombres maximums qui peuvent être écrits avec un nombre donné de chiffres. Parmi les trois paramètres caractéristiques, le plus fondamental est celui du nombre de symboles utilisés dans le système numérique. Il est connu sous le nom de base du système numérique.

Les positions des différents chiffres du nombre entier est donnée par b^0, b^1, b^2, b^3 et ainsi de suite. Pour la partie fractionnaire, elles sont b^{-1}, b^{-2}, b^{-3} et ainsi de suite. Ici, b est la base du système numérique. Aussi, le nombre maximal qui peut être écrit avec n chiffres dans un système de numération donné est égal à b^n.

Exemple :

A = {0, 1, 2,....., b-1}, soit b caractères qui quantifient le nombre d'unité d'une base d'un système de numération quelconque.

• La représentation polynomiale d'un nombre de base b est :

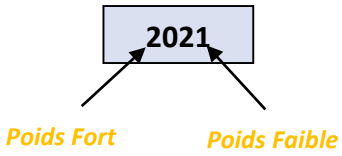
$$Nb = a_{n-1}\, a_{n-2} \ldots a_2\, a_1\, a_0, a_{-1}\, a_{-2} \ldots a_{-m} = a_{n-1}*b^{n-1} + a_{n-2}*b^{n-2} + \cdots + a_2*b^2 + a_1*b^1 + a_0*b^0 + a_{-1}*b^{-1} + a_{-2}*b^{-2} + \cdots + a_{-m}*b^{-m}$$

Si la base 10 nous est familière, d'autres bases existent. Les bases les plus utilisées en informatique sont les bases 10, 2, 8 et 16 appelées respectivement «décimale», «binaire», «octale» et «hexadécimale».

a. Système Décimal :

Le système numérique décimal est un système numérique de base 10 et comporte donc 10 chiffres ou symboles différents. Il s'agit de 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

Exemple: 2021 est un chiffre de 4 caractères :



La valeur ou la grandeur d'un nombre décimal donné peut être exprimée comme la somme des différents chiffres multipliés par leur base et leur poids.

La représentation polynomiale de ce chiffre est sous la forme:

$$2021 = 2 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$$

Poids

Base

b. Système Binaire :

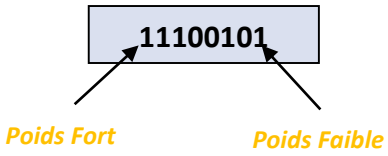
Le système de nombres binaires est un système de base 2 avec '0' et '1' comme deux chiffres indépendants.

La procédure d'écriture des nombres binaires d'ordre supérieur à "1" est similaire à celle expliquée dans le cas du système décimal.

Par exemple, les 16 premiers chiffres du système numérique binaire sont 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 et 1111. Le nombre suivant après 1111 est 10000, qui est le plus petit nombre binaire à cinq chiffres.

En partant du point binaire, les positions des différents chiffres d'un nombre binaire sont 2^0, 2^1, 2^2 et ainsi de suite (pour la partie entière) et 2^{-1}, 2^{-2}, 2^{-3} et ainsi de suite (pour la partie fractionnaire).

Exemple: 11100101 est un chiffre de 8 caractères



La représentation polynomiale de ce chiffre est sous la forme:

$$(11100101)_2 = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

Poids

Base

c. Système Octal

Le système numérique octal a une base de 8 et comporte huit chiffres distincts. Tous les nombres d'ordre supérieurs sont exprimés en combinant ces chiffres selon le modèle du système binaire et décimal. Les chiffres indépendants sont 0, 1, 2, 3, 4, 5, 6 et 7. Les 10 chiffres qui suivent '7', sont les suivants : 10, 11, 12, 13, 14, 15, 16, 17, 20 et 21.

Les positions des caractères du système octal sont 8^0, 8^1, 8^2 et ainsi de suite (pour la partie entière) et 8^{-1}, 8^{-2}, 8^{-3} et ainsi de suite (pour la partie fractionnaire).

Un Caractère Octal est codé sur 3 bits binaires (000 —— 0_8, 111 ——— 7_8)

d. Système Hexadécimal

Le système numérique hexadécimal est un système de base 16 et ses chiffres sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F. Le poids des différents chiffres d'un nombre hexadécimal est de 16^0, 16^1, 16^2, etc (pour la partie entière) et 16^{-1}, 16^{-2}, 16^{-3}, etc (pour la partie fractionnaire).

Le système numérique hexadécimal permet de représenter de manière plus courte les grands nombres binaires stockés et traités dans l'ordinateur. La représentation des adresses des différents emplacements de mémoire en est un exemple.

Un Caractère Hexa est codé sur 4 bits binaires (0000 —— 0_{16}, 1111 ——— F_{16})

Exemple:

$$(AC53)_{16} = (AC53)_H = A \times 16^3 + C \times 16^2 + 5 \times 16^1 + 3 \times 16^0$$

Le principal avantage de ce code est de pouvoir coder sur un mot court, un chiffre binaire important:

$$(11100101)_2 = (E5)_{16}$$

2.2 Changement de base

A. Calcul des équivalences décimales

L'équivalent décimal d'un nombre exprimé dans un autre système de numération X est donné par la somme du développement polynômial du nombre dans la base X.

Les parties entières et fractionnelles d'un nombre donné doivent être traitées séparément. Les conversions binaire-décimal, octal-décimal et hexadécimal-décimal sont illustrées ci-dessous à l'aide d'exemples.

Exemple1 : Conversion Binaire → Décimal

$$(10010101)_2 = 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (149)_{10}$$

$$(111{,}0011)_2 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} = (7{,}1875)_{10}$$

Exemple2 : Conversion Octal → Décimal

$$(5274)_8 = 5 \times 8^3 + 2 \times 8^2 + 7 \times 8^1 + 4 \times 8^0 = (2748)_{10}$$

Exemple3 : Conversion Hexadécimal → Décimal

$$(ABC)_{16} = 10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0 = (2748)_{10}$$

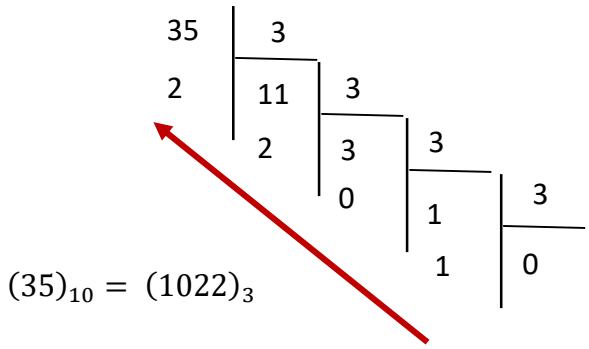
B. Conversion du décimal à une base X

Comme indiqué dans la section ci-dessus, les parties entières et fractionnaires sont calculées séparément. Pour la partie entière, l'équivalent en base X peut être calculé en divisant successivement la partie entière par X et en multipliant la partie fractionnaire fois 2, en conservant les restes jusqu'à ce que le résultat soit égal à "0". Les restes sont exprimés dans l'ordre inverse et constituent l'équivalent en base X.

Les conversions décimal →base 3, décimal-binaire, décimal-octal et décimal-hexadécimal sont illustrées ci-dessous à l'aide d'exemples.

Exemple 1: Conversion du décimal →base 3

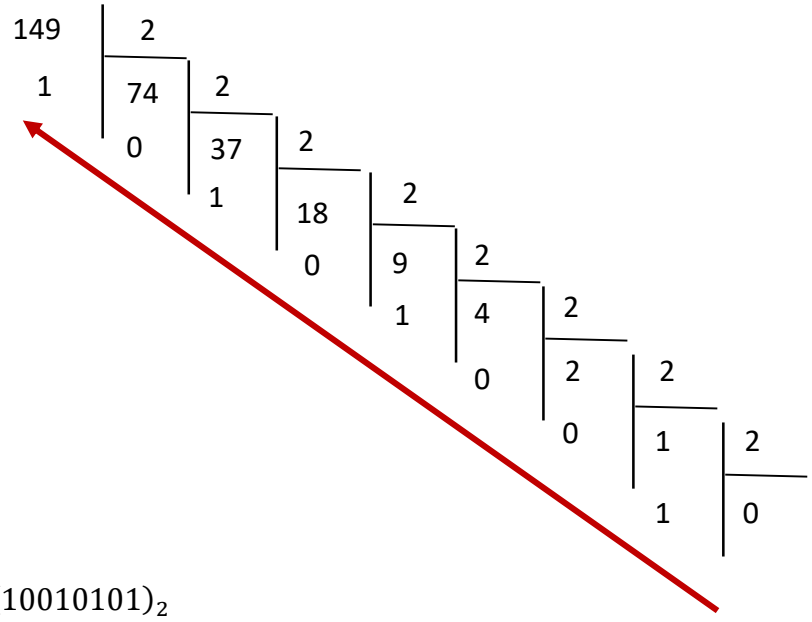
$(\mathbf{35})_{\mathbf{10}} = (?)_{\mathbf{3}}$



$(35)_{10} = (1022)_3$

Exemple 2: Conversion Décimal→ Binaire

$(149)_{10} = (?)_2$

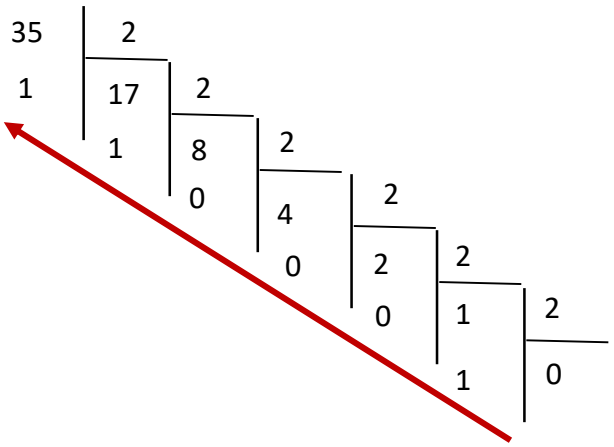


$(149)_{10} = (10010101)_2$

Exemple 3: Conversion Décimal→ Binaire (Cas d'un nombre réel)

$(35{,}625)_{10} = (?)_2$

- La partie entière est transformée en effectuant des divisions successives.
- La partie fractionnelle est transformée en effectuant des multiplications successives par 2.



$0.625 * 2 = \mathbf{1}.25$

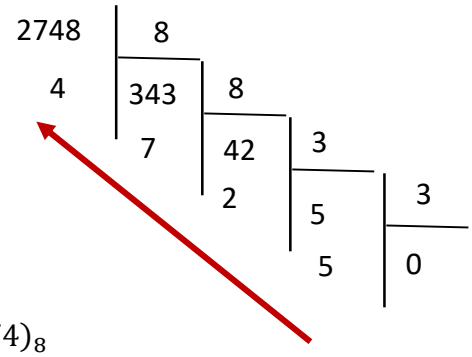
$0.25 * 2 = \mathbf{0}.5$

$0.5 * 2 = \mathbf{1}.0$

$$(35{,}625)_{10} = (100011{,}101)_2$$

Exemple 3: Conversion Décimal→Octal

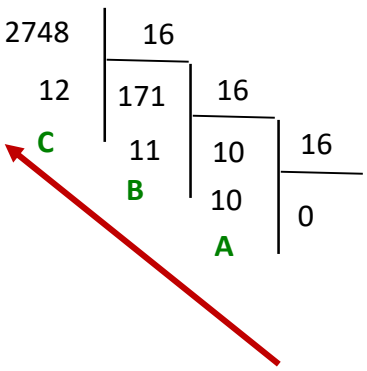
$(2748)_{10} = (?)_8$



$(2748)_{10} = (5274)_8$

Exemple 4: Conversion Décimal→Hexadécimal

$(2748)_{10} = (?)_{16}$



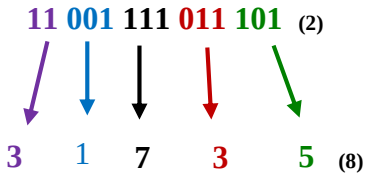
$(2748)_{10} = (ABC)_{16}$

C. Conversions binaire-octal (octal-binaire)

Un nombre octal peut être converti en son équivalent binaire en remplaçant chaque chiffre octal par son équivalent binaire de trois bits. Nous prenons l'équivalent de trois bits parce que la base du système octal est 8 et qu'elle est la troisième puissance de la base du système binaire, c'est-à-dire 2^3.

Un nombre binaire peut être converti en un nombre octal équivalent en divisant les parties entières et fractionnaires en groupes de trois bits, en partant du point binaire des deux côtés. Des 0 peuvent être ajoutés pour compléter les groupements de trois bits.

Exemple :

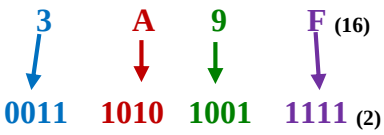


D. Conversions hexa-binaire (binaire-hexa)

Un nombre hexadécimal peut être converti en son équivalent binaire en remplaçant chaque chiffre hexadécimal par son équivalent binaire à quatre bits. Nous prenons l'équivalent de quatre bits parce que la base du système numérique hexadécimal est 16 et qu'elle est la quatrième puissance de la base du système numérique binaire, c'est-à-dire 2^4**.**

Un nombre binaire donné peut être converti en un nombre hexadécimal équivalent en divisant les parties entières et fractionnaires en groupes de quatre bits, en partant du premier point binaire de chaque côté. Les 0 peuvent être ajoutés si nécessaire.

Exemple 1 : Prenons la valeur hexadécimale $(3A9F)_{16}$ nous souhaitons la convertir en binaire.



E. Conversions hexa-octal (octal-hexa)

Pour la conversion hexadécimale-octale (octal-hexadécimal), le nombre hexadécimal (octal) donné est d'abord converti en son équivalent binaire, qui est ensuite converti en son équivalent octal (hexadécimal). Une autre approche consiste à convertir d'abord le nombre hexadécimal (octal) donné en son équivalent décimal, puis à convertir le nombre décimal en un nombre octal (hexadécimal) équivalent.

La première approche est sans aucun doute la meilleure. Les deux types de conversion sont illustrés dans l'exemple suivant.

Exemple 1 :

$(3\ 1\ 7\ 3\ 5)_8 = (011\ 001\ 111\ 011\ 101)_2 = (3\ 3\ D\ D)_{16}$

Exemple 2 :

$(3\ A\ 9\ F)_{16} = (0011\ 1010\ 1001\ 1111)_2 = (3\ 5\ 2\ 3\ 7)_8$

2.3 Décimal codé binaire (BCD)

Le code binaire décimal (BCD) est un type de code binaire utilisé pour représenter un nombre décimal donné sous une forme binaire équivalente. Les conversions BCD-décimal et décimal-BCD sont très faciles et directes. Il est également beaucoup moins fastidieux de représenter un nombre décimal donné dans un code BCD équivalent que de le représenter sous la forme binaire directe équivalente.

L'équivalent BCD d'un nombre décimal s'écrit en remplaçant chaque chiffre décimal dans les parties entières et fractionnaires par son équivalent binaire à quatre bits.

Exemple: Conversion Décimal→ BCD

$$(137{,}15)_{10} = (0001\ 0011\ 0111{,}0001\ 0101)_{DCB}.$$

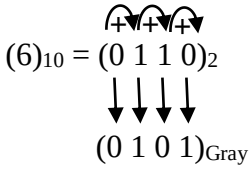
2.4 Code GRAY (Binaire réfléchi, BR)

Le code Gray est un code binaire non pondéré dans lequel deux valeurs successives ne diffèrent que d'un bit. Grâce à cette caractéristique, l'erreur maximale qui peut se glisser dans un système

utilisant le code Gray binaire pour coder des données est bien inférieure à celle correspondant d'un codage binaire direct.

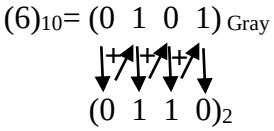
A. Conversion binaire – Gray :

Un nombre binaire peut être converti en équivalent Gray en suivant les étapes suivantes :



B. Conversion Gray-binaire :

Un code Gray peut être converti en équivalent binaire en suivant les étapes suivantes :



3 Opérations arithmétiques

Après avoir abordé les différentes méthodes de représentation des données numériques, cette section est dédiée à l'étude des règles de manipulation des données. Deux types d'opérations sont appliqués aux données numériques : les opérations arithmétiques et les opérations logiques. Les opérations arithmétiques de base comprennent l'addition, la soustraction, la multiplication et la division. Les fonctions logiques de base sont AND, OR et NOT. Alors que les règles des opérations arithmétiques sont traitées dans le présent chapitre, celles relatives aux opérations logiques seront abordées dans le chapitre suivant.

3.1 Addition

L'addition de deux nombres binaires est réalisée de la même façon que l'addition décimale.

Par exemple, 6 + 1 en décimal est égal à "7" car "7" suit immédiatement "6" dans le système numérique décimal. De même, 7 + 1 en octal est égal à "10" car, dans le système octal, le nombre supérieur adjacent après "7" est "10". De même, F + 1 dans le système hexadécimal est '10'. Dans ce contexte, nous pouvons écrire les règles de base de l'addition binaire comme suit :

Tableau 1.1: addition binaire

Opération	Résultat	Retenue
0 + 0	0	
0 + 1	1	
1 + 0	1	
1 + 1	0	1
1+ 1 + 1	1	1
1 + 1 + 1 + 1	0	10

Exemple 1 : Addition binaire

Retenue				1	1		1	1	1	1	1	1	1	1		
A		1	0	1	0	1	1	1	0	1	0	0	1	0	0	1	1
B	+	0	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1
Somme		1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0

Exemple 2 : Addition binaire

Retenue		1			1	1		1	1	1	1	1	1	1	1	
A		1	1	1	0	1	1	1	0	1	0	0	1	0	0	1	1
B	+	0	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1
Somme		~~1~~	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0

Le résultat de cette addition a nécessité 17 bits. Or, si nous limitons la longueur des mots à 16 bits, cette addition ne serait pas possible. En admettant un retranchement de la retenue supplémentaire, on obtient un résultat Faux.

Exemple 3: Addition Octal

Retenue		1	1		
A		4	3	6	5
B	+		4	5	1
Somme		5	8	11	6

En octal 11 s'écrit 13 et 8 s'écrit 10

Le résultat final: $(5036)_8$.

Exemple 2 : Addition Hexadécimal

Retenue		**1**			
A		**4**	**8**	**6**	**5**
B	**+**	**7**	**A**	**5**	**1**
Somme		**12**	**18**	**11**	**6**

En Hexadécimal 11 s'écrit B, 18 s'écrit 12 et 12 s'écrit C .

Le résultat final: $(C2B6)_{16}$.

3.2 Addition en BCD

Deux cas peuvent survenir quand on additionne deux chiffres décimaux.

- ***La somme de deux digits est inférieure ou égale à 9:***

Dans ce cas, aucune somme de deux chiffres décimaux ne dépasse 9

Exemple:

	2	**5**	**3**			**0010**	**0101**	**0011**
+	**4**	**2**	**1**		**+**	**0100**	**0010**	**0001**
=	**6**	**7**	**4**		**=**	**0110**	**0111**	**0100**
						6	**7**	**4**

- ***La somme de deux digits est supérieure 9:***

Dans un tel cas, il faut corriger la somme en additionnant 6 (0110) afin de prendre en considération le fait qu'on saute six présentations codées non valides.

Exemple:

	4	**7**			**0100**	**0111**	
+	**3**	**5**		**+**	**0011**	**0101**	
=	**8**	**2**		**=**	**0111**	**1100**	Le 1er chiffre est non valide car il est supérieur à 0
				+		**0110**	
				=	**1000**	**0010**	

3.3 Soustraction :

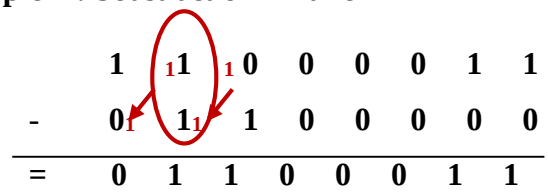
3.3.1 Soustraction directe

Les principes de base de la soustraction sont similaires à ceux que nous connaissons si bien dans les systèmes décimaux. Quand la quantité à soustraire est supérieure à la quantité dont on soustrait, on emprunte 1 au voisin de gauche.

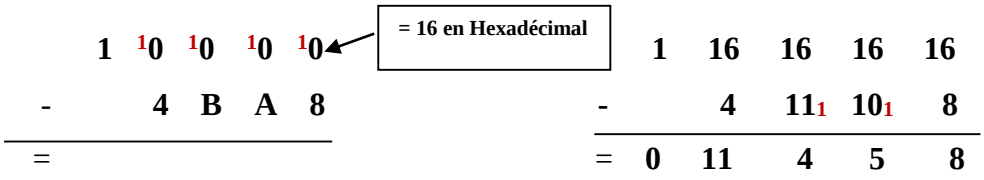
Tableau 1.2:soustraction binaire

Opération	Soustraction	Retenue
0 - 0	0	0
0 - 1	1	1
1 - 0	1	0
1 - 1	0	0

Exemple 1 : Soustraction Binaire



Exemple 2 : Soustraction Hexadécimal



Le résultat final: $(B458)_{16}$.

Remarque :

Une soustraction peut toujours, si on rend négatif son second terme, se ramener à une addition, ainsi : $[A - B] = [A + (-B)]$

3.3.2 Soustraction d'un nombre binaire signé :

Les nombres véhiculés dans la plus part des systèmes numériques (ordinateur) sont précédés (à la place habituelle du bit de poids fort) par un bit de signe: par conversion "0" représente un nombre positif et "1" représente un nombre négatif.

Exemple:

$$+13 \rightarrow 13 = (1101)_2 = \mathbf{0} : 1101$$

$$-13 \rightarrow 13 = (1101)_2 = \mathbf{1} : 1101$$

- **Le complément à 1 d'un nombre binaire**

Complémenter un nombre binaire à 1 consiste à changer tous les 0 par1 et tous les 1 par les 0.

Exemple: Le complément à 1 de:

- 10111 est 01000

- 0101101 est 1010010

La méthode la plus utilisée pour rendre négatif un nombre binaire est la méthode du complément à 2.

- **Le complément à 2 d'un nombre binaire**

Le complément à 2 d'un nombre binaire revient à trouver son complément à 1 puis additionner 1.

Exemple1:

Complément à 2 de $(-96)_{10} \rightarrow (96)_{10} = (001100000)_2$

Complément à 1 est : 1 1 0 0 1 1 1 1 1

\+ 1

Complément à 2 de (-96) est = 1 1 0 1 0 0 0 0 0

- **Le complément à 16 (8) d'un nombre Hexadécimal (Octal)**

En Hexadécimal (Octal) il faut faire le complément à 15 (7) puis additionner 1.

Exemple2: complément à 8 de $(-6230)_8$

Complément à 7 est : 1547

\+ 1

Complément à 8 est : 1550

Exemple3: Complément à 16 de $(-2E)_{16}$

Complément à 15 est : D 1

\+ 1

Complément à 16 est : D 2

Exemple4: soustraction d'un nombre binaire signé.

		1	9	5			0	1	1	0	0	0	0	1	1
+		(-	9	6)	+		1	1	0	1	0	0	0	0	0
=			9	9	=	~~1~~	0	0	1	1	0	0	0	1	1

Exemple5: soustraction d'un nombre hexadécimal signé.

		F	7			F	7
+	-	2	E	+		D	2
=				=	~~1~~	C	9

3.4 Multiplication

Les règles de base de la multiplication binaire sont régies par le fonctionnement d'une porte ET lorsque les deux bits à multiplier lui sont fournis en entrée. Les portes logiques sont abordées en détail dans le chapitre II. Pour l'instant, il suffit de dire que le résultat de la multiplication de deux bits est le même que la sortie de la porte ET avec les deux bits appliqués comme entrées à la porte. Les règles de base de la multiplication sont énumérées ci-dessous :

Tableau 1.3: multiplication binaire

Opération	Multiplication
0 * 0	0
0 * 1	0
1 * 0	0
1 * 1	1

La multiplication de nombres binaires de grande taille est similaire à celle que nous connaissons dans le cas des nombres décimaux. Il s'agit de l'algorithme "décalage à gauche et addition répétés".

Exemple :

$$\begin{array}{rr} & 5 \\ * & 2 \\ \hline = & 10 \end{array} \qquad \begin{array}{rr} & 0\ 1\ 0\ 1 \\ * & 0\ 0\ 1\ 0 \\ \hline & 0\ 0\ 0\ 0 \\ & 0\ 1\ 0\ 1\ . \\ 0\ 0\ 0\ & 0\ .\ . \\ \hline = \quad 0\ 0\ 1\ & 0\ 1\ 0 \end{array}$$

3.5 Division

Alors que la multiplication binaire est un processus d'addition répétée, la division binaire est un processus de soustraction répétée.

Tableau 1.4: division binaire

Opération	Division
0 ÷ 0	Impossible
0 ÷ 1	0
1 ÷ 0	Impossible
1 ÷ 1	1

Exemple :

```
1159 | 11
   4 | 105

  1 0 0 1 0 0 0 0 1 1 1  |  1 0 1 1
-     1 0 1 1            |  ---------------
  0 1 1 1 0              |  0 1 1 0 1 0 0 1
-   1 0 1 1
  0 0 1 1 0 0
-   1 0 1 1
  0 0 0 1 1 1 1
-     1 0 1 1
  0 1 0 0
```

Chapitre II

Algèbre de Boole et fonctions logiques

Introduction

Ce chapitre fournit un vocabulaire de base et de brèves notions d'algèbre de Boole qui permettent d'optimiser des circuits simples et de comprendre et optimiser des circuits complexes impliquant plusieurs portes logiques.

L'algèbre de Boole, comme tout autre système mathématique déductif, peut se définir avec un ensemble d'éléments, d'opérateurs et de variables à deux états (0 et 1).

1 Portes logiques

L'algèbre booléenne est munie de trois opérateurs élémentaires:

• La multiplication logique, dite aussi ET. Le symbole de cette opération est (.).

• L'addition logique, dite aussi OU. Le symbole de cette opération est (+).

• L'inverse logique, dite aussi NON. Le symbole de cette opération est (¯).

1.1 Opérateur ET

Une porte ET est un circuit logique comportant deux ou plusieurs entrées et une sortie. La sortie d'une porte ET est égale à 1 uniquement lorsque toutes ses entrées sont à l'état 1. Dans tous les autres cas, la sortie est égale à 0.

Si deus variables A et B sont combinées par la multiplication logique (ET), le résultat s'exprime ainsi :

$$X = A \,.\, B$$

Cela se traduit par l'expression suivante :

Si A est vraie ET B est vraie alors X est vraie.

Exemple :

Si la porte d'un ascenseur est fermée ET quelqu'un appuie sur le bouton 4, alors l'ascenseur monte au quatrième étage.

Les propositions caractérisées par les variables logiques A, B et X sont :

(A) : porte fermée.

(B) : quelqu'un appuie sur le bouton 4.

(X) : l'ascenseur monte au 4eme étage.

On a alors : (X)= (A) ET (B) ↔ $X = A \,.\, B$

La liste complète des valeurs que peut prendre la fonction booléenne X en fonction de toutes les combinaisons possibles des valeurs A et B, peuvent être résumées dans un tableau appelé table de vérité. La table de vérité (Tableau 2.1) et le symbole logique (Fig. 2.1) de l'opérateur ET est donnée par le tableau suivant.

Tableau 2.1: Table de vérité ET.

A	B	X=A.B
0	0	0
0	1	0
1	0	0
1	1	1

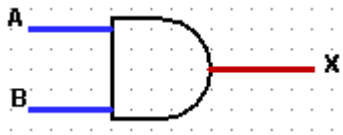


Figure 2.1 : Symbole logique de la porte ET.

1.2 Opérateur OU

La sortie d'une porte OU est égale à 0 uniquement lorsque toutes ses entrées sont égales à 0. Pour toutes les autres combinaisons d'entrées possibles, la sortie est égale à 1.

L'addition logique de deux variables A et B donne :

$$X = A + B$$

Cela se traduit par l'expression suivante : X est vraie si au moins A ou B est vrai.

Exemple :

On veut recruter un interprète connaissant l'allemand **OU** l'anglais. Cette expression peut se décomposer ainsi :

(A) : connaissance de l'allemand
(B) : connaissance de l'anglais
(X) : recrutement de l'interprète.

On a alors : (X) = (A) **OU** (B) $\leftrightarrow X = A + B$

Le recrutement de l'interprète se fait s'il connait : soit l'allemand soit l'anglais soit les deux à la fois.

Le symbole du circuit (Fig. 2.2) et la table de vérité (Tableau 2.2) d'une porte OU à deux entrées sont montrés ci-dessous.

Tableau 2.2: Table de vérité OU.

A	B	X=A+B
0	0	0
0	1	1
1	0	1
1	1	1

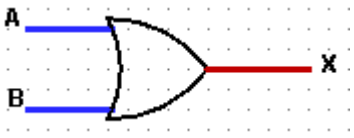


Figure 2.2 : Symbole logique de la porte OU.

1.3 Opérateur NOT

Une porte NOT, contrairement aux opérateurs précédents, est un circuit logique à une seule entrée et une seule sortie dont la sortie est toujours le complément de l'entrée.

Le résultat X de l'opérateur NOT sur une variable booléenne A donne son complément. Lorsqu'il est interprété pour un système logique positif, un "0" logique à l'entrée produit un "1" logique à la sortie, et vice versa. On note :

$$X = \bar{A}$$

Le symbole du circuit (Fig.2.3) ainsi que la table de vérité (Tableau 2.3) de l'opérateur NOT sont donnés ci-dessous :

Tableau 2.3: Table de vérité NOT.

A	$X = \bar{A}$
0	1
1	0

Figure 2.3 : Symbole logique de la porte NOT.

2 Autres portes logiques

2.1 Opérateur NAND

NAND est l'abréviation de NOT AND. Une porte ET suivie d'un circuit NON constituent une porte NON-ET.

Ci-dessous sont montrés le symbole de circuit d'une porte NAND (Fig. 2.4) à deux entrées ainsi que sa table de vérité (Tableau 2.4). Cette dernière est obtenue à partir de la table de vérité d'une porte ET en complétant les sorties. La sortie d'une porte NAND est "0" logique lorsque toutes ses entrées sont un "1" logique. Pour toutes les autres combinaisons d'entrées, la sortie est égale à '1' logique. On le note ↑ : A↑ B =$\overline{A.B}$.

Tableau 2.4: *Table de vérité NAND.*

A	**B**	**X=A↑B**
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

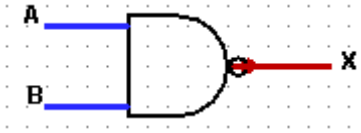


Figure 2.4 : Symbole logique de la porte NAND.

2.2 Opérateur NOR

NOR est l'abréviation de NOT OR. Une porte OU suivie d'un circuit NON font de cette porte une porte NOR. La table de vérité d'une porte NOR est obtenue à partir de la table de vérité d'une porte OU en complétant les sorties. La sortie d'une porte NOR est un '1' logique lorsque toutes ses entrées sont des '0' logiques. Pour toutes les autres combinaisons d'entrées, la sortie est un '0' logique. On le note ↓ : A↓ B =$\overline{A+B}$.

La sortie d'une porte NOR à deux entrées est exprimée logiquement comme suit :

Tableau 5: Table de vérité NOR.

A	B	X=A↓B
0	0	1
0	1	0
1	0	0
1	1	0

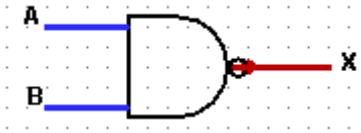


Figure 2.5 : Symbole logique de la porte NOR.

2.3 Opérateur XOR

La porte OU-EXCLUSIF, communément appelée porte XOR, est une porte à deux entrées et une sortie. La figure 2.6 et le tableau 2.6 montrent le symbole logique et la table de vérité d'une porte XOR à deux entrées. Comme le montre la table de vérité, la sortie d'une porte OU-exclusif est un " 1 " logique lorsque les entrées sont différentes et un " 0 " logique lorsque les entrées

sont égaux. C'est-à-dire qu'une séquence d'entrée entièrement à 0 produit un " 0 " logique à la sortie. On le note $\oplus$: A $\oplus$B $= A.\bar{B} + \bar{A}.B$.

Tableau 2.6: Table de vérité XOR.

A	B	X=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

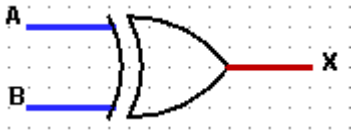


Figure 2.6 : Symbole logique de la porte XOR.

2.4 Opérateur XNOR

EXCLUSIF-NOR (couramment écrit XNOR) correspond au NON de XOR, c'est-à-dire la porte logique que l'on obtient en complétant la sortie d'une porte XOR.

La figure 2.7 et le tableau 2.7 montrent le symbole de son circuit ainsi que sa table de vérité. La table de vérité d'une porte XNOR est obtenue à partir de la table de vérité d'une porte XOR en complétant les entrées de sortie.

Tableau 2.7: Table de vérité XNOR.

A	B	X=AΘB
0	0	1
0	1	0
1	0	0
1	1	1

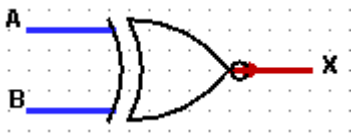


Figure 2.7 : Symbole logique de la porte XOR.

3 Opérateurs Universelles (système complet)

Les NAND et les NOR sont des portes universelles car elles permettent de réaliser toutes les opérations logiques élémentaires. Par exemple avec des NAND, on peut réaliser les opérations suivantes:

a) Implémentation de portes logiques élémentaires en utilisant uniquement des portes NAND.

1) $\bar{A} = \overline{A.A}$ (selon le théorème d'idempotence section 4.1)

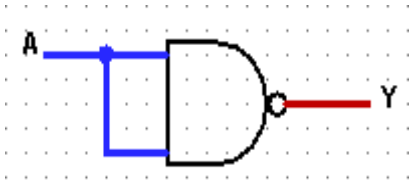


Figure 2.8 : Porte NOT

2) $A.B = \overline{\overline{A.B}}$ (selon le théorème d'involution section 4.1)

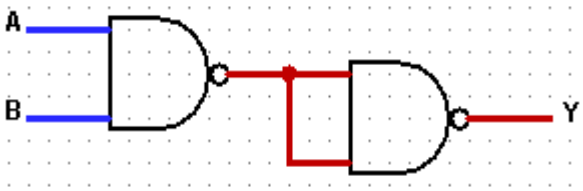


Figure 2.9 : Porte AND

3) $A + B = \overline{\overline{A + B}} = \overline{\bar{A}\,.\,\bar{B}}$ (selon le théorème d'involution et de Morgan section 4.1)

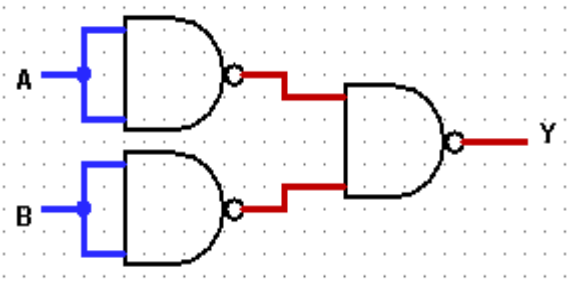


Figure 2.9 : Porte OR

b) Implémentation de portes logiques élémentaires en utilisant uniquement des portes NOR.

1) $\bar{A} = \overline{A + A}$

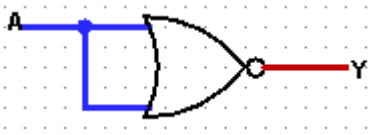


Figure 2.10: Porte NOT

2) $A.B = \overline{\bar{A} + \bar{B}}$

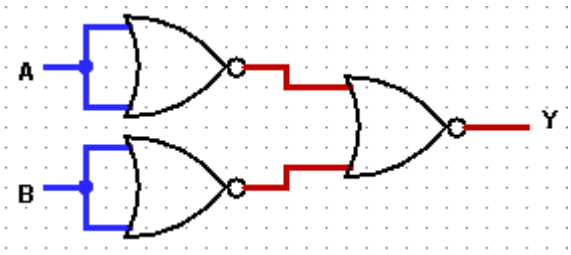


Figure 2.11 : Porte AND

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

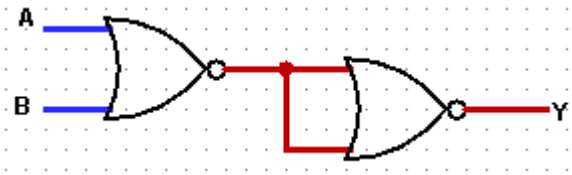


Figure 2.12: Porte OR

4 Algèbre de Boole

L'algèbre de Boole est l'un des outils les plus fondamentaux dont dispose le concepteur de logique qui peut donc être utilisé efficacement pour simplifier des expressions logiques complexes. D'autres techniques intéressantes et largement utilisées, basées sur les théorèmes booléens, comprennent l'utilisation du tableau de Karnaugh.

Dans ce chapitre, nous examinerons de plus près les différents propriétés et théorèmes de l'algèbre de Boole et leurs applications dans la minimisation des expressions booléennes.

Nous aborderons aussi en détail les méthodes des tableaux de Karnaugh permettant de minimiser des expressions logiques assez complexes et de grande taille.

4.1 Théorèmes de l'algèbre de Boole

Les propriétés (Tableau 2.8) et les théorèmes (Tableau 2.9) de l'algèbre de Boole, peuvent être utilisés pour simplifier de nombreuses expressions booléennes complexes, et pour transformer une expression donnée en une expression équivalente plus utile et plus significative. Également pour transformer l'expression donnée en une expression équivalente plus utile et plus significative.

Tableau 2.8: Propriétés de l'algèbre de Boole.

Propriétés	OU (OR)	ET (AND)
Commutativité	$x + y = y + x$	$x.y = y.x$
Associativité	$x + (y + z) = (x + y) + z$	$x.(y.z) = (x.y).z$
Distributivité	$x.(y + z) = (x.y) + (x.z)$	$x + (y.z) = (x + y).(x + z)$
Complémentation	$x + \bar{x} = 1$	$x.\bar{x} = 0$
Eléments neutres	$x + 0 = x$	$x.1 = x$

Tableau 2.9: Théorèmes de l'algèbre de Boole

Théorèmes	OU (OR)	ET (AND)
Idempotence	$x + x = x$	$x.x = x$
Absorption	$x + (x.y) = x$ $x.(\bar{x} + y) = x.y$	$x.(x + y) = x$ $x + \bar{x}\,.y = x + y$
Elément neutre	$x + 1 = 1$	$x.0 = 0$
Involution	$\bar{\bar{x}} = x$	
Morgan	$\overline{x.y} = \bar{x} + \bar{y}$	$\overline{x + y} = \bar{x}\,.\bar{y}$
Consensus	$a.x + b.\bar{x} + a.b = a.x + b.\bar{x}$	$(a + x).(b + \bar{x}).(a + b) = (a + x).(b + \bar{x})$

4.2 Simplification des fonctions logiques

Bien que la représentation de la table de vérité d'une fonction soit unique, lorsqu'elle est exprimée algébriquement, elle peut apparaître sous de nombreuses formes différentes, mais équivalentes.

La complexité des portes logiques numériques qui implémentent une fonction booléenne est directement liée à la complexité de l'expression algébrique à partir de laquelle la fonction est implémentée.

L'objectif de la simplification des fonctions logiques est de :

– réduire le nombre de termes dans une fonction

– réduire le nombre de variables dans un terme

•Cela afin de réduire le nombre de portes logiques utilisées et réduire le coût du circuit

• Plusieurs méthodes existent pour la simplification :

– La Méthode algébrique

– Les Méthodes graphiques : (ex : table de karnaugh)

- **Méthode algébrique**

Le principe consiste à appliquer les règles de l'algèbre de Boole (sans démarche bien spécifique) afin d'éliminer des variables ou des termes.

Exemple :

$F_1 = x.(\bar{x} + x.y) = x.\bar{x} + x.x.y = 0 + x.y = x.y$

$F_2 = x.y + \bar{x}.z + y.z = x.y + \bar{x}.z + x.y.z + \bar{x}.y.z = x.y.(1 + z) + \bar{x}.z.(1 + y) = x.y + \bar{x}.z$

$F_3 = \overline{\left(x.(\bar{y}.\bar{z} + y.z)\right)} = \bar{x} + \overline{(\bar{y}.\bar{z} + y.z)} = \bar{x} + (y + z).(\bar{y} + \bar{z}) = \bar{x} + y.\bar{z} + \bar{y}.z$

La méthode algébrique est efficace, mais il est facile de rater une simplification, notamment quand la fonction est compliquée.

La méthode graphique des tableaux de KARNAUGH, est la méthode la plus efficace car elle garantit un bon résultat.

- **Méthode graphique (table de Karnaugh)**

Un tableau de Karnaugh est une représentation graphique (Fig. 2.13) du système logique qui fournit une procédure simple et directe pour minimiser les fonctions booléennes. Il peut être

établi directement à partir des expressions booléennes mintermes (somme de produits) ou maxtermes (produit de sommes). Le tableau de Karnaugh est établi à partir de la table de vérité, et doit vérifier les points suivants :

1) Un tableau de Karnaugh à n-variables a 2^n cases, et à chaque entrée possible est attribuée une case.

2) Tous les termes sont adjacents et ne différent que par l'état d'une seule variable.

3) Un tableau de Karnaugh peut s'appliquer aux fonctions logiques de 2, 3, 4, 5 et 6 variables.

AB / C	0	1
0		
1		

2 Variables

AB / C	00	01	11	10
0				
1				

3 Variables

AB / CD	00	01	11	10
00				
01				
11				
10				

4 Variables

ABC / DE	**000**	**001**	**011**	**010**	**110**	**111**	**101**	**100**
00								
01								
11								
10								

5 Variables

Figure 2.13: Tableaux de Karnaugh à 2, 3, 4 et 5 variables

Dans un tableau de karnaugh, chaque case possède un certain nombre de cases adjacentes (Fig 2.14).

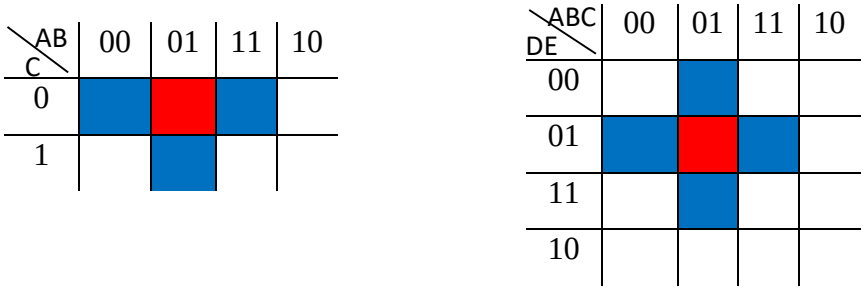


Figure 2.14: Cases adjacentes du tableau de karnaugh

Après avoir dessiné la carte de Karnaugh, l'étape suivante consiste à former des groupes de 1 selon les directives suivantes :

1. Chaque carré contenant un "1" doit être considéré au moins une fois, bien qu'il puisse être considéré aussi souvent que souhaité.
2. L'objectif doit être de prendre en compte toutes les cases marquées dans le nombre minimum de groupes.
3. Le nombre de carrés dans un groupe doit toujours être une puissance de 2, c'est-à-dire que les groupes peuvent avoir 1, 2, 4 8, 16, carrés.

4.3 Expression d'une fonction logique

A un nombre fini N de variables d'entrée correspond 2^N combinaisons possibles. La table de vérité va donc indiquer la valeur de la fonction pour les 2^N valeurs possibles. Si par exemple S est une fonction de trois variables (Tableau 2.10), il y aura 2^3 soit 8 combinaisons possibles.

Tableau 2.10: table de vérité de 3 entrées et les fonctions correspondantes.

Entrées					**Sorties**
Valeurs Entières	**A**	**B**	**C**	**Combinaison**	**S**
0	0	0	0	$\bar{A}.\bar{B}.\bar{C}$	0
1	0	0	1	$\bar{A}.\bar{B}.C$	1
2	0	1	0	$\bar{A}.B.\bar{C}$	1
3	0	1	1	$\bar{A}.B.C$	1
4	1	0	0	$A.\bar{B}.\bar{C}$	0
5	1	0	1	$A.\bar{B}.C$	1
6	1	1	0	$A.B.\bar{C}$	0
7	1	1	1	$A.B.C$	0

Sous sa forme complète, l'équation logique de S se lit directement. C'est la somme du ET logique de chaque combinaison avec l'état de S correspondant.

Dans notre exemple on obtient la forme canonique complète :

$$S = \bar{A}.\bar{B}.\bar{C}.0 + \bar{A}.\bar{B}.C.1 + \bar{A}.B.\bar{C}.1 + \bar{A}.B.C.1 + A.\bar{B}.\bar{C}.0 + A.\bar{B}.C.1 + A.B.\bar{C}.0 + A.B.C.0$$

On sait que X.0 = 0, donc on peut éliminer les termes qui valent 0. Cela nous donne :

$$S = \bar{A}.\bar{B}.C.1 + \bar{A}.B.\bar{C}.1 + \bar{A}.B.C.1 + A.\bar{B}.C.1$$

On sait aussi que X.1 = X, donc on peut écrire la forme canonique abrégée :

$$S = \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + \bar{A}.B.C + A.\bar{B}.C$$

- ***Somme Canonique (Σ.Π):***

Une fonction logique est écrite sous forme de somme canonique, si toutes les variables de chaque termes sont reliées entre elles par l'opérateur ET. Ces termes se désignent sous le nom de mintermes.

Exemple: $X = \overline{A}.B.C + \overline{A}.\overline{B}.C$

- ***Produit Canonique (Π.Σ):***

Une fonction logique est écrite sous forme de produit canonique, si toutes les variables de chaque termes sont reliées entre elles par l'opérateur OU. Ces termes se désignent sous le nom de maxtermes.

Exemple: $X = (A + \overline{B} + C + D).(\overline{A} + B + \overline{C} + \overline{D})$

Exemple : Passage de la forme canonique à la table de Karnaugh

Soit X (A, B, C) une fonction logique à trois variables, représentée par la table de vérité suivante:

N°	A	B	C	X
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

L'expression algébrique de la fonction X peut être déduite de la table de vérité sous l'une des deux formes canoniques :

- 1ere forme canonique : Somme canonique (∑.Π)

Si la fonction logique est donnée sous la première forme canonique, alors sa représentation est directe :

$$X(A,B,C) = \sum(0,2,5,7) = \bar{A}.\bar{B}.\bar{C} + \bar{A}.B.\bar{C} + A.\bar{B}.C + A.B.C$$

Chaque terme lui correspond une seule case qui doit être mise à 1.

A \ BC	00	01	11	10
0	1 (0)	(1)	(3)	1 (2)
1	(4)	1 (5)	1 (7)	(6)

$\bar{A}.\bar{C}$

$A.C$

En effectuant une lecture par « intersection » en recherchant la ou les variables ne changent pas par paquet, la fonction simplifiée devient :

$$X(A,B,C) = \bar{A}.\bar{C} + A.C = \overline{A \oplus C}$$

- 2eme forme canonique : Somme canonique (Π .∑)

Si la fonction logique est donnée sous la deuxième forme canonique (conjonctive), alors sa représentation est directe :

$$X(A,B,C) = \prod(1,3,4,6) = (\bar{A} + \bar{B} + C).(A + B + \bar{C}).(\bar{A} + B + C).(A + \bar{B} + \bar{C})$$

Chaque terme lui correspond une seule case qui doit être mise à 0.

A \ BC	00	01	11	10
0	(0)	0 (1)	0 (3)	(2)
1	0 (4)	(5)	(7)	0 (6)

$A + \bar{C}$

$\bar{A} + C$

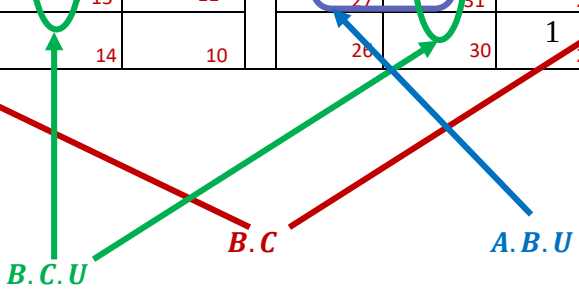
La fonction simplifiée devient :

$$X(A,B,C) = (A + \bar{C}).(\bar{A} + C) = \overline{A \oplus C}$$

Exemple : Table de karnaugh à 5 variables

$$F(A,B,C,D,U) = \sum(0,1,2,3,13,15,16,17,18,19,22,25,27,29,31)$$

DU \ ABC	000	001	011	010		110	111	101	100
00	1 0	4	12	8		24	28	20	1 16
01	1 1	5	1 13	9		1 25	1 29	21	1 17
11	1 3	7	1 15	11		1 27	1 31	23	1 19
10	1 2	6	14	10		26	30	1 22	1 18



$$F(A,B,C,D,U) = B.C.U + B.C + A.B.U$$

Chapitre III

Circuits Logiques Combinatoires

Introduction

Un circuit combinatoire permet de produire une sortie qui dépend uniquement de la combinaison actuelle des entrées. La porte logique est l'élément de base de la logique combinatoire. La fonction logique exécutée par un circuit combinatoire est entièrement définie par un ensemble d'expressions booléennes.

L'autre catégorie de circuits logiques, appelée circuits logiques séquentiels, comprend à la fois des portes logiques et des éléments de mémoire tels que les bascules. En raison de la présence d'éléments de mémoire, la sortie d'un circuit séquentiel dépend non seulement de l'état actuel mais aussi de l'état passé des entrées. Les éléments de base des circuits logiques séquentiels sont décrits en détail dans les chapitres suivants.

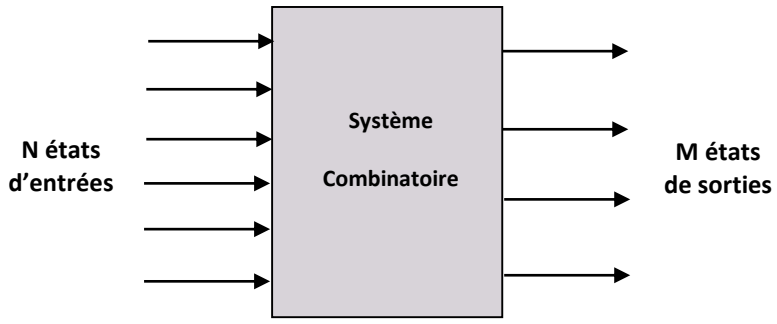


Figure 3.1 : Système logique combinatoire

La figure 3.1, montre un bloc de circuits combinatoires généralisés ayant N variables d'entrée et M variables de sortie. Comme le nombre de variables d'entrée est N, il y a 2^N combinaisons possibles de bits à l'entrée. Chaque sortie peut être exprimée en termes de variables d'entrée par une expression booléenne, de sorte que le système généralisé de la figure précédente peut être exprimé par M expressions booléennes.

1 Implémentation de la logique combinatoire

Les différentes étapes de la conception d'un circuit logique combinatoire sont les suivantes :

1. Définition de la problématique.

2. Identification des variables d'entrée et de sortie.

3. Expression du rapport entre les variables d'entrée et de sortie.

4. Mise en place d'une table de vérité correspondant aux conditions d'entrée-sortie.

5. Exprimer sous forme d'expressions booléennes les variables de sortie en fonction des variables d'entrée.

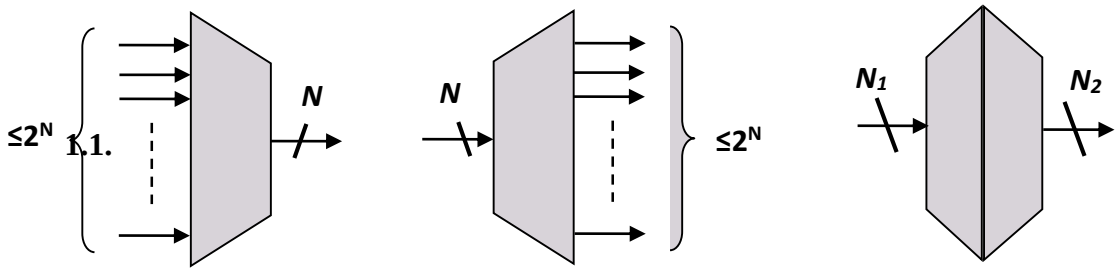
6. Réduction des expressions booléennes.

7. Implémentation d'expressions booléennes minimisées.

Ces différentes étapes sont explicites. Il existe différentes techniques de simplification pour minimiser les expressions booléennes, qui ont été abordées dans le chapitre précédent.

2 Circuits de Transcodage

Ces circuits de transcodages (Fig. 3.2) transforment une information présente à leurs entrées sous une forme donnée (code 1) en la même information présente à leurs sorties, sous une forme différente (code 2). Ils se répartissent en 3 catégories:



Codeur (Encodeur) Décodeur Transcodeur

Figure 3.2: Circuit de transcodage

2.1 Encodeurs

Un codeur a un certain nombre de voie d'entrée 2N, dont une seule peut être activée à la fois. A chaque activation d'entrée, il apparait un unique code binaire de sortie à N bits. Prenons le cas d'un codeur 4X2. Un tel codeur aurait 4 lignes d'entrée et 2 lignes de sortie (Fig 3.3) représentant l'équivalent binaire. La table de vérité d'un tel codeur est donnée dans le tableau 3.1. Dans cette table de vérité, E0 à E3 représentent les chiffres 0 à 3. S0, S1 représentent les chiffres binaires.

Tableau 3.1 : Codeur élémentaire 4 vers 2

E_3	E_2	E_1	E_0	S_0	S_1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

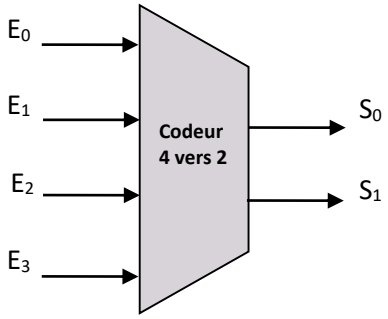


Figure 3.3: Codeur 4 vers 2

- **Encodeur de priorité**

Un codeur de priorité est une version pratique du codeur. Dans ce type d'encodeur, une priorité est attribuée à chaque entrée de sorte que, lorsque plusieurs entrées sont simultanément actives (à 1), c'est la priorité du codage le plus élevé qui est retenue (Tableau 3.2). Il y a donc priorité aux entrées de rang le plus élevé.

Tableau 3.2 : Codeur Prioritaire 4 vers 2

E_3	E_2	E_1	E_0	S_0	S_1
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

2.2 Décodeur

Un décodeur est un circuit combinatoire qui décode n entrées vers 2^n sorties uniques. Avec un code binaire d'entrée de n bits, un décodeur peut générer un maximum de 2n mintermes possibles.

Afin d'illustrer davantage le fonctionnement d'un décodeur, considérons le schéma de la figure 4.3. Ce bloc logique, met en œuvre une fonction de décodage de 3 vers 8. Ce décodeur possède trois entrées désignées par a_0, a_1 et a_2 et huit sorties désignées par s_0, s_1, s_2, s_3, s_4, s_5, s_6 et s_7. D'après le tableau 3.3, il est clair que, pour toute combinaison d'entrées donnée, une seule des huit sorties est à l'état logique "1". Ainsi, chaque sortie produit un minterme qui correspond au nombre binaire actif à l'entrée.

Tableau 3.3: table de vérité d'un décodeur 3 vers 8 et les fonctions correspondantes.

N°	a_2	a_1	a_0	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0	mintermes
0	0	0	0	0	0	0	0	0	0	0	1	$s_0 = \overline{a_2}.\overline{a_1}.\overline{a_0}$
1	0	0	1	0	0	0	0	0	0	1	0	$s_1 = \overline{a_2}.\overline{a_1}.a_0$
2	0	1	0	0	0	0	0	0	1	0	0	$s_2 = \overline{a_2}.a_1.\overline{a_0}$
3	0	1	1	0	0	0	0	1	0	0	0	$s_3 = \overline{a_2}.a_1.a_0$
4	1	0	0	0	0	0	1	0	0	0	0	$s_4 = a_2.\overline{a_1}.\overline{a_0}$
5	1	0	1	0	0	1	0	0	0	0	0	$s_5 = a_2.\overline{a_1}.a_0$
6	1	1	0	0	1	0	0	0	0	0	0	$s_6 = a_2.a_1.\overline{a_0}$
7	1	1	1	1	0	0	0	0	0	0	0	$s_7 = a_2.a_1.a_0$

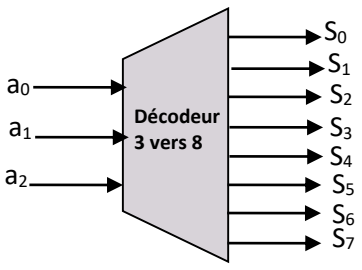


Figure 3.4: Décodeur 3 vers 8

Exemple: Implémentation d'une fonction F= Σ (1, 3, 4, 6) en utilisant un multiplexeur 3vers8.

F= Σ (1, 3, 4, 6)= $\overline{a_2}.\overline{a_1}.a_0 + \overline{a_2}.a_1.a_0 + a_2.\overline{a_1}.\overline{a_0} + a_2.a_1.\overline{a_0}$

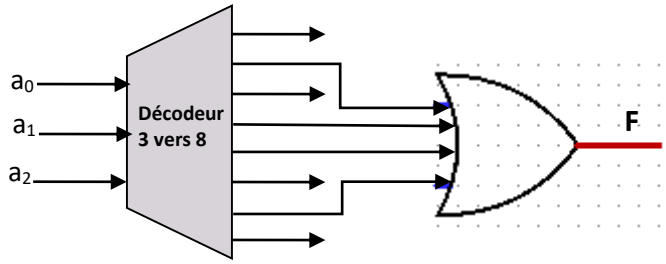


Figure 3.5: Implémentation de la fonction F à l'aide d'un décodeur 3 vers 8

- **Association de décodeurs**

Il est possible d'associer plusieurs décodeurs afin d'augmenter les capacités d'un système.

Exemple: deux décodeurs 3 vers 8 →décodeur de 3 vers 16

Si l'entrée a_3 vaut 0 (1), les entrées $\bar{X}$ et $\bar{Y}$ du décodeur 1 (2) sont actives, celles du décodeur 2 (1) inactives.

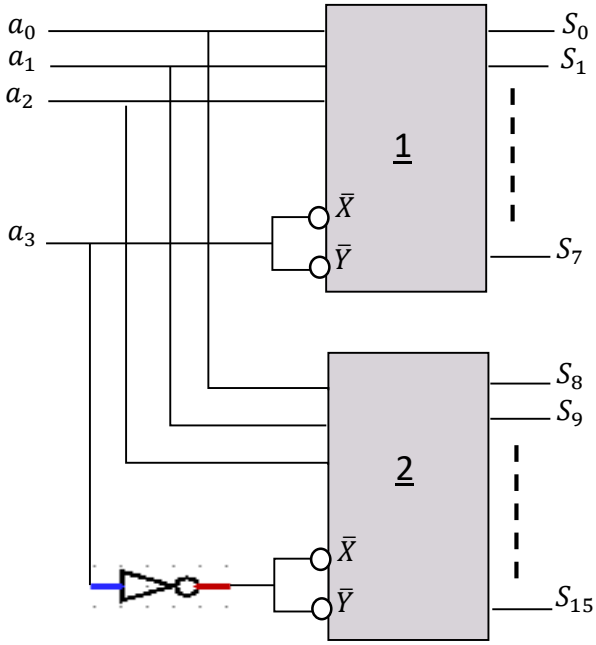


Figure 3.6: Association deux décodeurs 3 vers 8.

Quand le décodeur 1 (2) est validé, et le décodeur 2 (1) inhibé. Les sorties S_0 à S_7 (S_8 à S_{15}) sont alors représentatives des entrées b c d e.

2.3 Transcodeurs

Un transcodeur est un circuit logique permettant de passer d'une information écrite dans un code C1 à un autre code C2.

L'information s'exprimera par exemple à l'aide des variables A, B, C, D dans le code C1, et à l'aide des variables X, Y, Z dans le code C2.

Les deux importantes applications de transcodeurs sont:

- La conversion de code
- L'affichage par segment

Exemple : Application des transcodeurs [binaire pur – binaire réfléchi (code Gray) sur 3 bits]

Code binaire			**Code Gray**		
A	**B**	**C**	**X**	**Y**	**Z**
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Equations des sorties :

C \ AB	00	01	11	10
0	0	0	1	1
1	0	0	1	1

$X = A$

C \ AB	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$Y = \bar{A}B + A\bar{B} = A \oplus B$

C \ AB	00	01	11	10
0	0	1	1	0
1	1	0	0	1

$Z = B\bar{C} + \bar{B}C = B \oplus C$

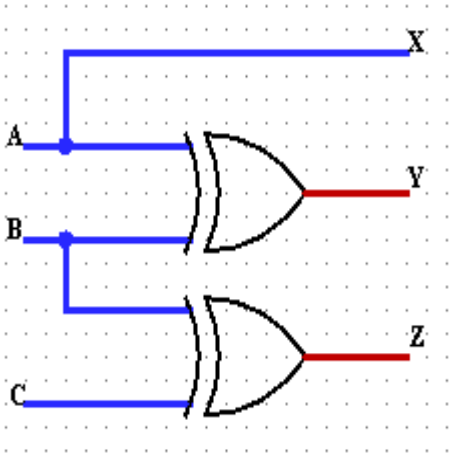


Figure 3.7: Transcodeur binaire - Gray.

3 Circuit d'aiguillage

Cette section traite des éléments de base tels que les multiplexeurs et les démultiplexeurs. Un accent particulier est mis sur les bases opérationnelles et l'utilisation de ces dispositifs pour concevoir des circuits combinatoires plus complexes. Les multiplexeurs et démultiplexeurs sont généralement utilisés pour assurer la répartition des ressources. Les multiplexeurs acheminent les données de plusieurs sources vers une destination et le démultiplexeur redistribue les données d'une source vers plusieurs destinations.

3.1 Multiplexeurs

Un multiplexeur ou MUX, également appelé sélecteur de données, est un circuit combinatoire comportant plus d'une ligne d'entrée, une ligne de sortie et plus d'une ligne de sélection. Un multiplexeur sélectionne les informations binaires présentes sur une des lignes d'entrée, en fonction de l'état logique des entrées de sélection, et les transmet à la ligne de sortie (Tableau 3.4, Fig. 3.8 et 3.9).

Si il y a n lignes de sélection, alors le nombre maximal de lignes d'entrée possibles est 2^n et le multiplexeur est appelé multiplexeur 2^n vers 1 ou $2^n \times 1$.

Tableau 3.4 : Table de vérité d'un multiplexeur 4 vers 1

a_0	a_1	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

$$S = \overline{a_0}.\overline{a_1}.E_0 + \overline{a_0}.a_1.E_1 + a_0.\overline{a_1}.E_2 + a_0.a_1.E_3$$

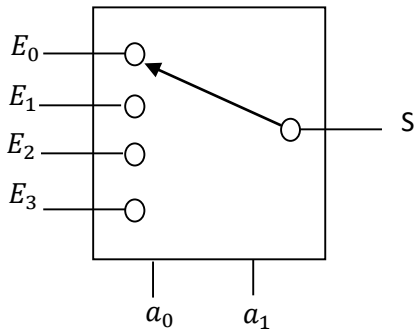


Figure 3.8: Symbole d'un multiplexeur 4 vers 1.

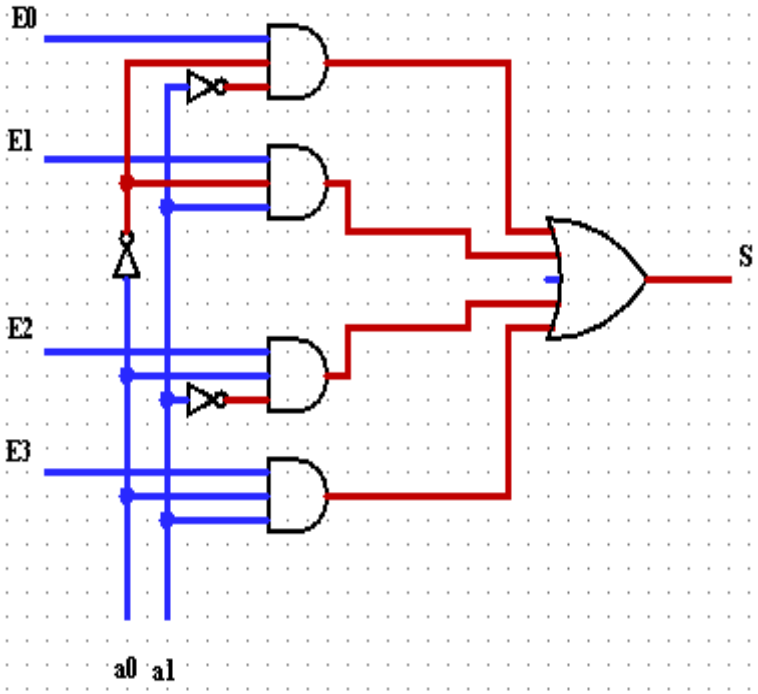


Figure 3.9: Schéma fondamental d'un multiplexeur 4 vers 1.

Exemple: Implémentation d'une fonction F= Σ (1, 3, 4, 11, 12, 13, 14, 15) en utilisant un multiplexeur 8x1

A	B	C	D	F	
0	0	0	0	0	**F=D**
0	0	0	1	1	
0	0	1	0	0	**F=D**
0	0	1	1	1	
0	1	0	0	1	**F=$\overline{D}$**
0	1	0	1	0	
0	1	1	0	0	**F=0**
0	1	1	1	0	
1	0	0	0	0	**F=0**
1	0	0	1	0	
1	0	1	0	0	**F=D**
1	0	1	1	1	
1	1	0	0	1	**F=1**
1	1	0	1	1	
1	1	1	0	1	**F=1**
1	1	1	1	1	

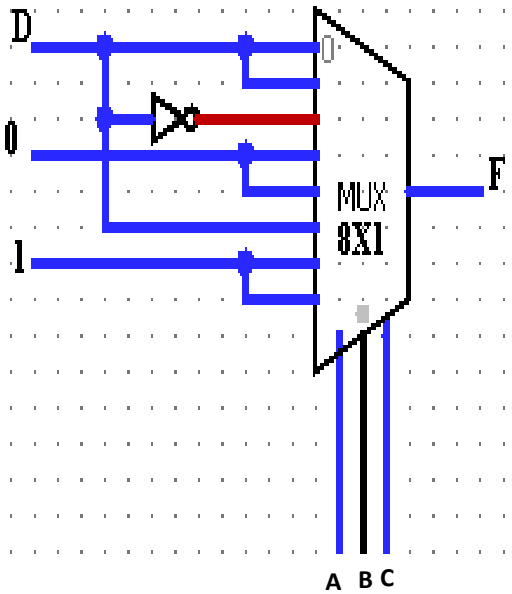


Figure 3.10: Implémentation de la fonction F à l'aide d'un multiplexeur 8 vers 1.

- **Association de Multiplexeurs**

En général, les opérateurs de multiplexage disposent d'une entrée supplémentaire de validation V qui permet de valider ou d'inhiber la sortie S (Fig. 3.11). Cette entrée est souvent utilisée afin d'augmenter les capacités de multiplexage.

Exemple : multiplexeur 8 vers 1 à partir de deux multiplexeurs 4 vers 1.

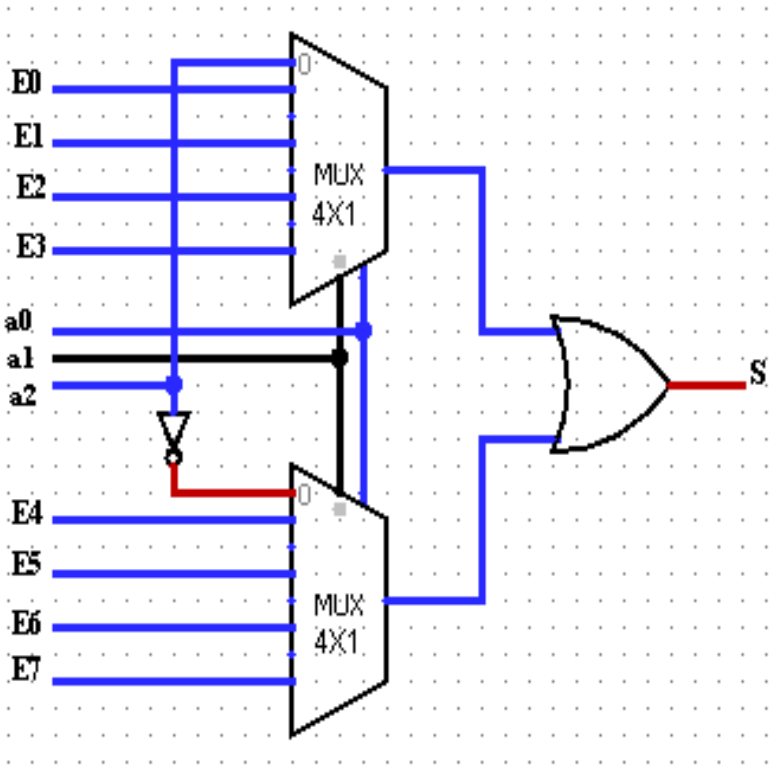


Figure 3.11: Association deux multiplexeurs 4 vers 1.

3.2 Démultiplexeurs

Le rôle du démultiplexeur est aiguiller (ou commuter) une entrée vers 2^N sortie à l'aide de N bits de commande binaires. Il achemine l'information présente sur la ligne d'entrée vers l'une des lignes de sortie. La sortie qui reçoit l'information présentée en entrée est déterminée par l'état des bits de sélection (Tableau 3.5, Fig. 3.12 et 3.13).

Tableau 3.5 : Table de vérité d'un démultiplexeur 1 vers 4

N	a_0	a_1	S_0	S_1	S_2	S_3	Fonction
0	0	0	E	0	0	0	$S_0 = E.\overline{a_0}.\overline{a_1}$
1	0	1	0	E	0	0	$S_1 = E.\overline{a_0}.a_1$
2	1	0	0	0	E	0	$S_2 = E.a_0.\overline{a_1}$
3	1	1	0	0	0	E	$S_3 = E.a_0.a_1$

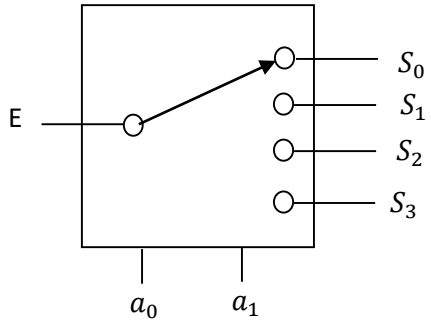


Figure 3.12: Symbole d'un démultiplexeur 1 vers 4.

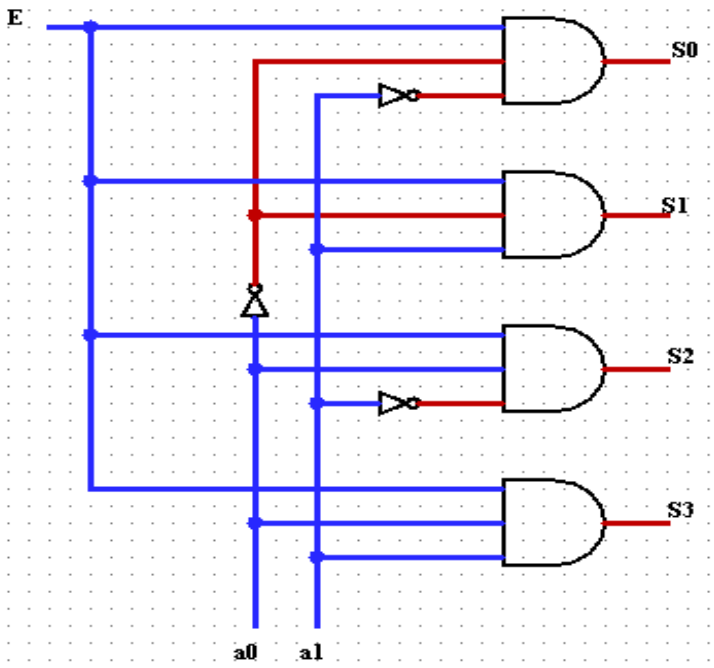


Figure 3.13: Schéma fondamental d'un démultiplexeur 1 vers 4.

4 Circuit Arithmétique

Dans cette section, nous abordons les différents modules de logique combinatoire qui peuvent être utilisés pour effectuer les opérations d'addition et de soustraction sur des nombres binaires. L'addition et la soustraction sont les deux opérations arithmétiques les plus couramment utilisées, car les deux autres, à savoir la multiplication et la division, sont respectivement des processus d'addition répétée et de soustraction répétée.

4.1 Additionneur

4.1.1 Demi additionneur (DA)

Le demi additionneur additionne deux nombres binaires codes sur un bit chacun. Un tel circuit possède donc deux entrées qui représentent les deux bits à additionner et deux sorties, le bit de poids faible produit la sortie S et tandis que le bit de poids fort est la retenue R_0.

Le tableau 3.6 et la figure 3.14, représentent respectivement la table de vérité et le circuit logique d'un demi-additionneur, avec toutes les combinaisons d'entrées possibles et les sorties correspondantes.

Tableau 3.6 : table de vérité d'un demi-additionneur

A	B	S	R_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = A \oplus B \ .$$

$$R_0 = A.B \ .$$

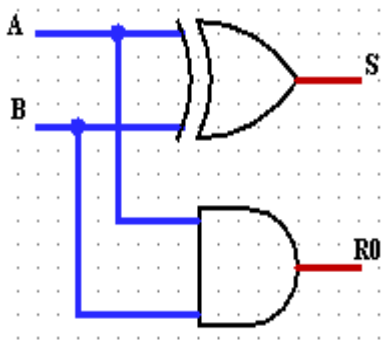


Figure 3.14: Schéma fondamental d'un demi-additionneur

4.1.2 Additionneur complet (AC)

Le principe précédent peut être généralisé pour décrire l'addition de deux nombres binaires A et B de taille supérieure à un bit. Il faut pour cela répercuter à l'étape i+1 l'éventuelle retenue provenant de l'addition de Ai et Bi. Une nouvelle variable Ri représentant la retenue entrante est alors introduite.

Le tableau 3.7 et la figure 3.15, représentent respectivement la table de vérité et le circuit logique d'un additionneur complet.

Tableau 3.7: table de vérité d'un additionneur complet

A	B	Ri	S	R_0
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

$$S = A \oplus B \oplus R_i$$

$$R_0 = A.B + A.R_i + B.R_i$$

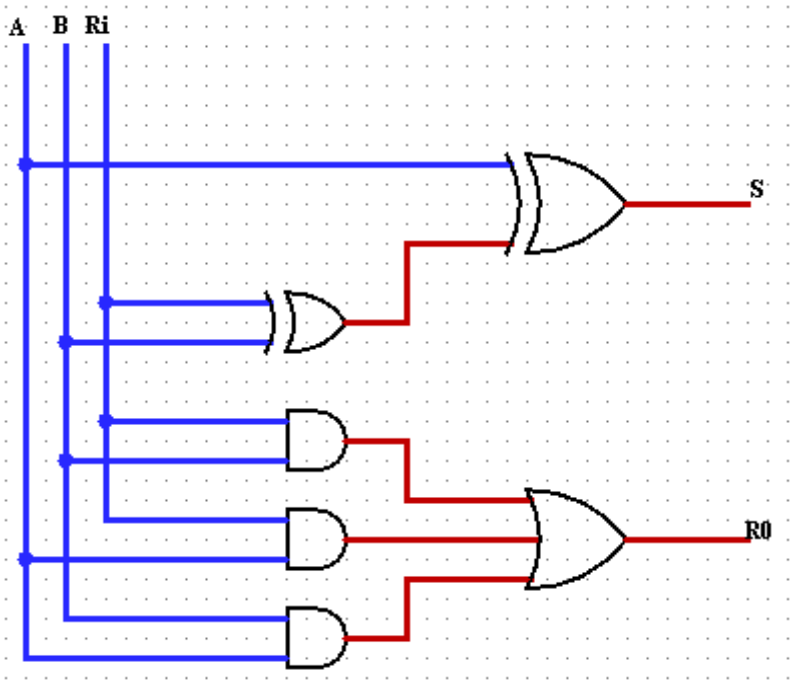


Figure 3.15: Schéma fondamental d'un additionneur complet.

Ainsi, nous pouvons implémenter un circuit d'additionneur complet à l'aide de deux circuits de demi-additionneurs. Le premier demi-additionneur sera utilisé pour ajouter A et B, afin de produire une somme partielle. La deuxième partie du demi-additionneur peut être utilisée pour ajouter R_i à la somme produite par le premier demi -additionneur, pour obtenir la sortie finale S. Si un des demi-additionneurs produit une retenue, il y aura une retenue en sortie. Ainsi, R_{i+1} est une fonction OU des sorties Carry des deux demi-additionneurs.

Dans la figure 3.16 est présenté un aperçu de l'implémentation d'un circuit d'additionneur complet:

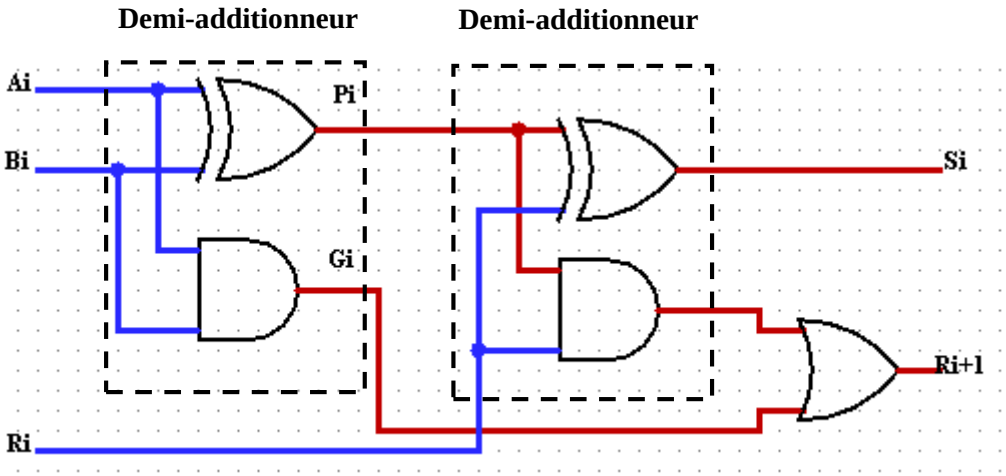


Figure 3.16: Schéma fondamental d'un (AC) utilisant deux (DA).

Même si la mise en œuvre de schémas logiques plus grands est possible avec les additionneurs complets, un symbole plus simple est généralement utilisé pour représenter l'opération (Fig. 3.17).

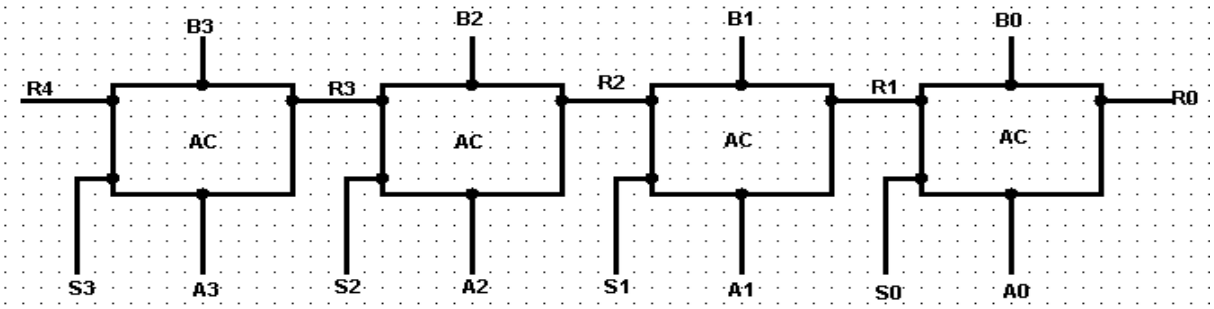


Figure 3.17: Schéma bloc d'un additionneur complet de 4 bits.

La retenue initiale Ro est bien évidemment égale à 0 puisqu'il n' y a pas de rang précèdent au rang 0.

4.2 Soustracteur

La soustraction d'un nombre binaire B de l'autre nombre binaire A se ramène à l'addition de A et du complément à 2 de B.

A – B = A + (-B) = A + C2 (B) = A + (B' + 1)

Cela nous permet d'effectuer une opération de soustraction avec des circuits additionneurs. Nous allons examiner l'utilisation des circuits additionneurs pour les opérations de soustraction. Avant cela, nous allons aborder les circuits demi-soustracteur et le soustracteur complet, pour l'implémentation directe des opérations de soustraction à l'aide de portes logiques.

4.2.1 Demi-soustracteur (DS)

Le demi-soustracteur réalise la soustraction d'un nombre binaire A à un autre nombre binaire B. Il possède donc deux entrées qui sont A et B, et deux sorties Di et R_0, qui sont respectivement le bit de différence et la retenue sortante. La sortie R_0 indique si un 1 est emprunté pour effectuer la soustraction.

Dans le tableau 3.8 et la figure 3.18, sont présentés respectivement la table de vérité et l'implémentation d'un circuit demi-soustracteur :

Tableau 3.8: table de vérité d'un demi-additionneur

A	B	Di	R_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$D_i = \bar{A}.B + A.\bar{B} = A \oplus B \ ,$$

$$R_0 = \bar{A}.B$$

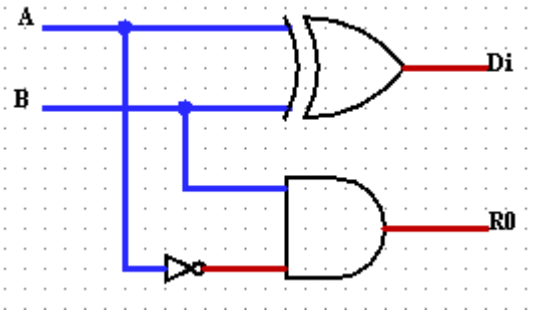


Figure 3.18: Schéma fondamental d'un demi-soustracteur.

4.2.2 Soustracteur complet (SC)

Nous pouvons généraliser la structure précédente pour décrire la soustraction des nombres binaires de taille supérieure à un bit. Pour cela, il faut introduire une nouvelle variable Ri, qui représente la retenue entrante. Par conséquence, un soustracteur complet effectue une opération de soustraction sur deux bits, un premier et un second, en prenant en compte les éléments tels que la prise en compte du 1 emprunté par le bit précédent.

La table de vérité et le circuit logique d'un soustracteur complet sont donnés dans le tableau 3.9 et la figure 3.19.

Tableau 3.9: table de vérité d'un soustracteur complet

A	B	Ri	Di	R_0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D_i = A \oplus B \oplus R_i$$

$$R_0 = \bar{A}.B + \bar{A}.R_i + B.R_i$$

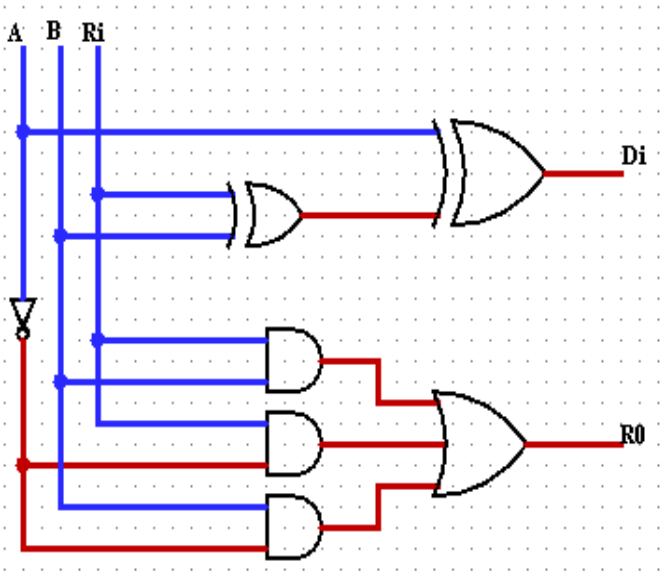


Figure 3.19: Schéma fondamental d'un soustracteur complet.

Si nous comparons ces expressions avec celles obtenues précédemment dans le cas d'un additionneur complet, nous constatons que l'expression de la sortie Somme est la même que celle de la sortie Différence. De même, les expressions des retenues pour les opérations de soustraction et d'addition sont similaires. Dans le cas d'un demi-soustracteur, l'entrée A est complétée. Par une analyse similaire, on peut montrer qu'un soustracteur complet peut être implémenté avec des demi-soustracteurs de la même manière que la construction d'un additionneur complet avec des demi-additionneurs.

Dans la figure 3.20, est présenté un aperçu de l'implémentation d'un circuit soustracteur complet :

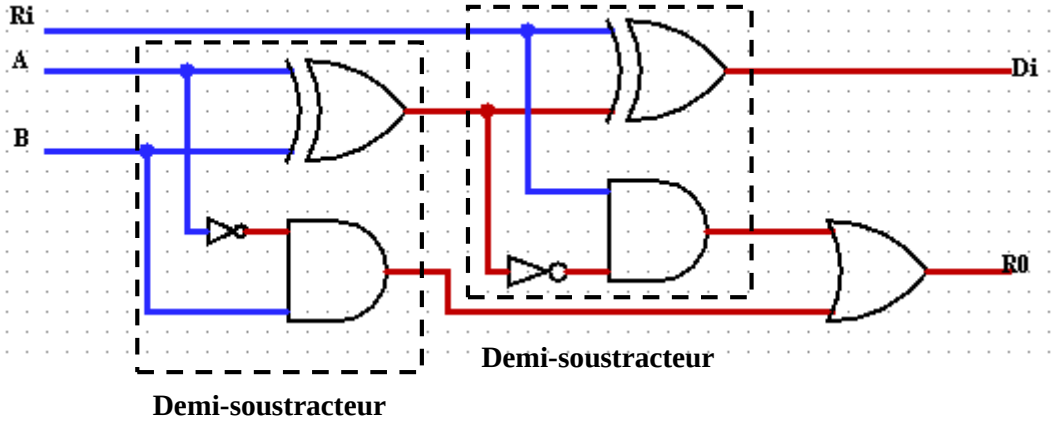


Figure 3.20: Schéma fondamental d'un (SC) utilisant deux (DS).

De même, plusieurs soustracteurs complets peuvent être connectés en cascade, pour effectuer une soustraction sur des nombres binaires plus grands (Fig.3.21).

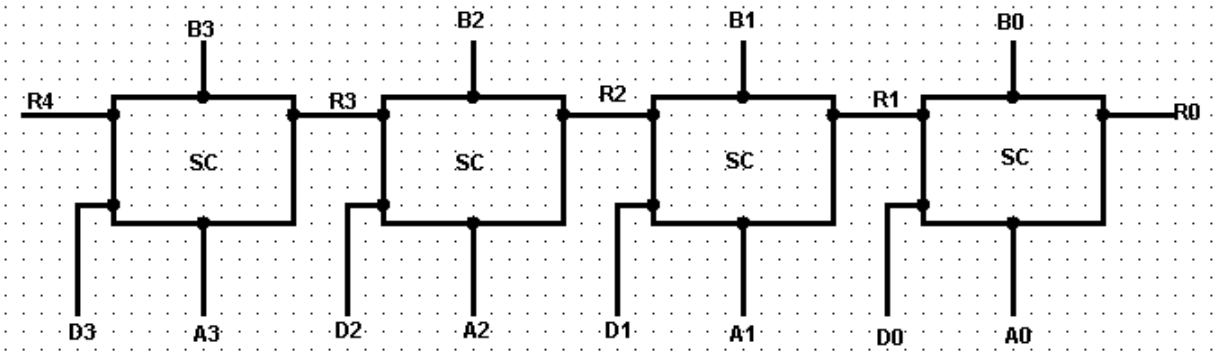


Figure 3.21: Schéma bloc d'un soustracteur complet de 4 bits.

4.3 Additionneur-soustracteur binaire

Comme nous l'avons souligné au début de la section 4.2, la soustraction n'est rien d'autre qu'une addition à complémentaire à 2. Ainsi, le circuit permettant de soustraire A – B, est constitué d'un additionneur avec des inverseurs placés dans chaque entrée de données B. La retenue d'entrée Rin doit être égale à 1 lorsque la soustraction est effectuée. L'opération ainsi réalisée devient A, plus le complément à 1 de B, plus 1. Ceci est égal à A plus le complément à 2 de B. Les opérations d'addition et de soustraction peuvent être combinées en un seul circuit avec un seul additionneur binaire commun, en incluant une porte OU exclusif avec chaque additionneur complet. Un circuit additionneur-soustracteur de quatre bits est illustré à la figure 3.22. L'entrée de mode Ctl contrôle le fonctionnement. Lorsque Ctl = 0, le circuit est un additionneur, et lorsque M = 1, le circuit devient un soustracteur (Tableau 3.10).

Tableau 3.10: fonctionnement de l'additionneur-soustracteur:

Ctl	Opération	Xi	Yi	Rin
0	Add (A+B)	Ai	Bi	0
1	Sous $(A + \overline{B} + 1)$	Ai	$\overline{B_i}$	1

$$X_i = A_i$$

$$Y_i = B_i.\overline{Ctl} + \overline{B_i}.Ctl = B_i \oplus Ctl$$

$$R_{in} = Ctl$$

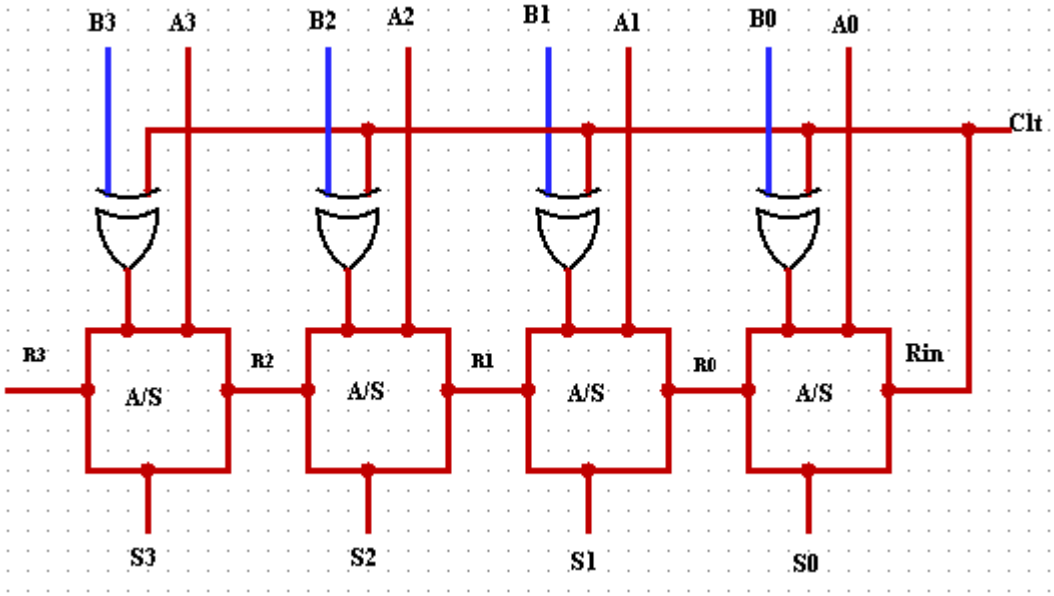


Figure 3.22: Schéma bloc d'un additionneur-soustracteur complet de 4 bits.

4.4 Additionneur Décimal (BCD)

Un additionneur BCD est utilisé pour faire une addition des chiffres BCD. Un chiffre BCD peut avoir l'une des dix représentations binaires de 4-bit suivantes : 0000, 0001….1001, ce qui correspond aux nombres décimaux 0, 1…. 9. Lorsque nous décidons d'additionner deux chiffres

BCD et en supposant qu'il y a également une retenue à l'entrée, le plus grand nombre binaire que nous pouvons obtenir est l'équivalent au nombre décimal 19 (9+9+1) égal à $(10011)_2$ en binaire. Cependant, si nous effectuons une addition BCD, nous devons nous attendre à ce que la réponse soit $(0001\ 1001)_{BCD}$. Le tableau 3.11, présente les résultats possibles en binaire et les résultats attendus en BCD, lorsque nous utilisons un additionneur binaire de quatre bits, pour effectuer l'addition de deux chiffres BCD.

Tableau 3.11 : comparaison entre les résultats possibles en binaire et attendus en BCD

Somme binaire					**Somme BCD**					**Déc.**
K	**X3**	**X2**	**X1**	**X0**	**C**	**S3**	**S2**	**S1**	**S0**	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Il ressort clairement du tableau que, tant que la somme des deux chiffres BCD reste égale ou inférieure à 9, l'additionneur quatre bits produit la sortie BCD correcte. Dans ce cas, la somme binaire et la somme BCD sont identiques. La différence entre les deux résultats se produit lorsque la somme est supérieure à 9. Le tableau montre également que, pour une somme décimale supérieure à 9 (ou la somme binaire équivalente supérieure à 1001), si on ajoute 0110 à la somme binaire, on peut obtenir la somme BCD correcte et la sortie de retenue exacte

(Tableau 3.12). L'expression booléenne qui permet de faire la correction nécessaire s'écrit selon l'équation de 'C'.

Table de Karnaugh de l'équation 'C'

$X_1 X_0$ \ $K X_3 X_2$	000	001	011	010	110	111	101	100
00	0	0	1	0	X	X	X	1
01	0	0	1	0	X	X	X	1
11	0	0	1	1	X	X	X	1
10	0	0	1	1	X	X	X	1

$$C = K + X1.X3 + X2.X3$$

Tableau 3.12 : Table de fonctionnement de 'C'

C	a	b	c	d	Action
0	0	0	0	0	Aucune correction
1	0	1	1	0	Correction

Le circuit logique d'un additionneur BCD de 4 bits est donné dans la figure 3.23

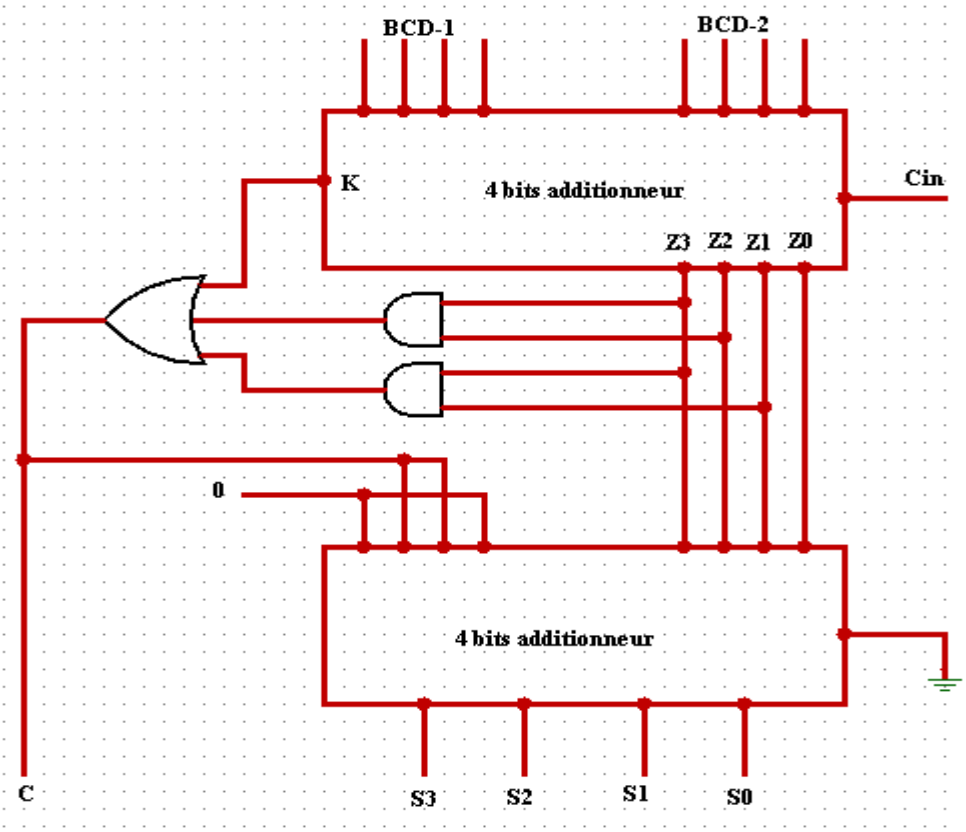


Figure 3.23: Schéma logique d'un additionneur BCD de 4 bits.

4.5 Comparateurs

Un comparateur est un circuit combinatoire qui compare deux nombres donnés, et détermine si l'un est égal, inférieur ou supérieur à l'autre. La sortie se présente sous la forme de trois variables binaires représentant les conditions A = B, A>B et A<B (Tableau 3.12, Fig. 3.24).

Tableau 3.13 : Comparateur binaire

A	B	S_1(A> B)	S_2(A=B)	S_3(A< B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

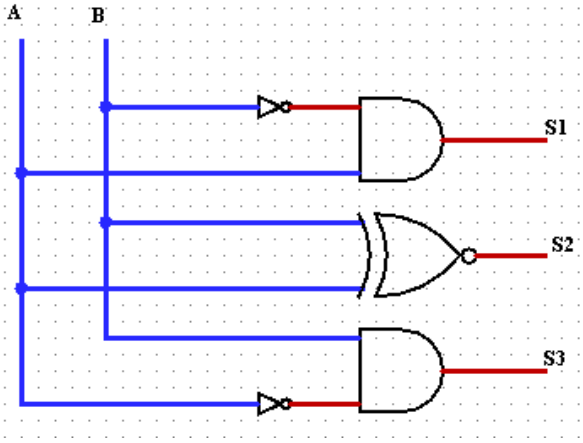


Figure 3.24: Circuit logique d'un comparateur 2 bits.

Si A et B (deux nombres à comparer) sont des nombres binaires de quatre bits désignés par (A3 A2 A1 A0) et (B3 B2 B1 B0): (i) les deux nombres seront égaux si toutes les paires de chiffres significatifs sont égales, c'est-à-dire A3= B3, A2= B2, A1= B1 et A0= B0, (ii) Pour déterminer si A est supérieur ou inférieur à B, nous examinons les paires de chiffres significatifs, en commençant par la position la plus élevée.

Exemple : comparaison de 2 nombres égaux
Deux nombres A et B sont égaux si tous les bits de même rang, Ai et Bi, sont égaux.

$$S = \overline{A_0 \oplus B_0}. \ldots . \overline{A_i \oplus B_i}. \ldots . \overline{A_{n-1} \oplus B_{n-1}} = \prod_{i=0}^{n-1} \overline{A_i \oplus B_i}$$

Soit encore :

$$S = \overline{(A_0 \oplus B_0) + \cdots + (A_i \oplus B_i) + \cdots + (A_{n-1} \oplus B_{n-1})} = \overline{\sum_{\iota=0}^{n-1} A_\iota \oplus B_\iota}$$

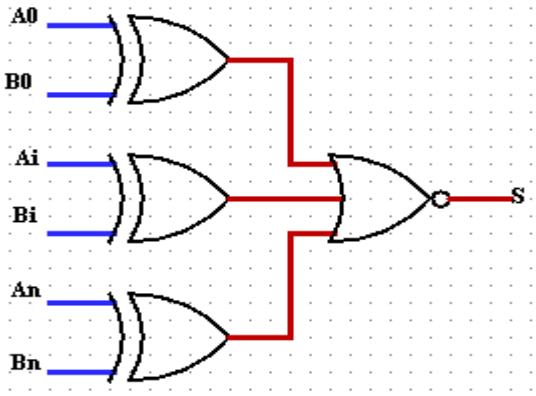


Figure 3.25: Circuit logique d'un comparateur d'égalité 4 bits.

Electronique numérique II

-Logique Séquentielle-

Chapitre IV

Bascules Usuelles

Introduction

Les circuits numériques considérés jusqu'à présent étaient combinatoires - leur sortie dépend uniquement et immédiatement de leurs entrées - ils n'ont pas de mémoire, c'est-à-dire, qu'ils dépendent des valeurs de leurs entrées. Les circuits séquentiels, en revanche, agissent comme des éléments de stockage et possèdent une mémoire. Ils peuvent stocker, conserver, puis récupérer des informations en cas de besoin.

Les figures 4.1 et 4.2, présentent le schéma fonctionnel d'un circuit séquentiel. Il se compose d'un circuit combinatoire, auquel des éléments de stockage sont connectés, pour former un circuit de retour. Le circuit séquentiel reçoit des données binaires provenant d'entrées externes qui, avec l'état actuel des éléments de stockage, déterminent la valeur binaire des sorties. Ainsi, un circuit séquentiel est spécifié par une séquence temporelle d'entrées, de sorties et d'états internes. En revanche, les sorties de la logique combinatoire dépendent uniquement des valeurs actuelles des entrées.

Il existe deux grands types de circuits séquentiels, et leur classification est fonction de la synchronisation de leurs signaux.

1 Circuit séquentiel

1.1 Circuit séquentiel asynchrone

Le fonctionnement d'un circuit séquentiel asynchrone dépend des signaux d'entrée à un instant donné et de la séquence dans laquelle les entrées changent (Fig. 4.1).

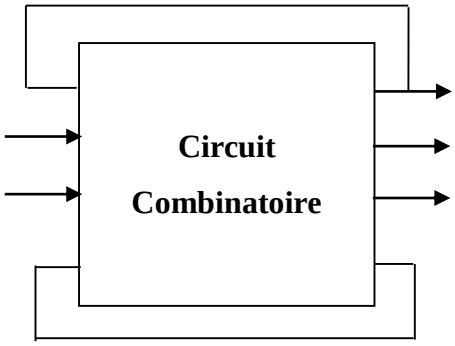


Figure 4.1 : Schéma d'un circuit séquentiel asynchrone

1.2 Circuit séquentiel synchrone

Un circuit séquentiel synchrone est un système dont le comportement peut être défini à partir de la connaissance de ses signaux à des instants discrets.

La synchronisation est assurée par un dispositif de synchronisation appelé générateur d'horloge (Fig. 4.2), qui fournit un signal d'horloge sous la forme d'un train périodique d'impulsions d'horloge. Ainsi, la transition d'un état à l'autre ne se produit qu'à des intervalles prédéterminés dictés par les impulsions d'horloge.

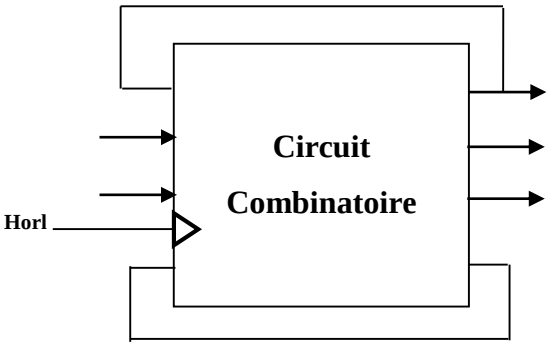


Figure 4.2 : Schéma d'un circuit séquentiel synchrone

Les éléments de stockage (mémoire), utilisés dans les circuits séquentiels synchronisés, sont appelés flip-flops.

2 Bascule RS

La bascule SR est un circuit comportant deux portes NOR (Fig. 4.3) ou deux portes NAND (Fig. 4.5), couplées en croisement, avec deux entrées marquées S pour set et R pour reset.

Lorsque la sortie $Q = 1$ et $\bar{Q} = 0$, on dit que la bascule est dans l'état 'set'. Lorsque $Q = 0$ et $\bar{Q} = 1$, elle est dans l'état de reset . Les sorties Q et $\bar{Q}$ sont en principe complémentaires.

Comme le montre le tableau de fonction (Tableau 4.1) de la figure 4.3, deux conditions d'entrée amènent le circuit à l'état actif.

C**ondition 1 (S = 1, R = 0) :** le circuit passe à l'état 'set'. Lorsqu'on remet l'entrée S à 0, le circuit reste dans le même état (mémoire).

C**ondition 2 (S = 0, R = 1) :** le circuit passe à l'état 'reset'. Lorsqu'on remet l'entrée R à 0, le circuit reste dans le même état (mémoire).

C**ondition 3 (S = 1, R = 1) :** les deux sorties passent à 0. Cette action produit un état indéfini. En fonctionnement normal, cette condition est éliminée en vérifiant que les 1 ne soient pas appliqués aux deux entrées en même temps.

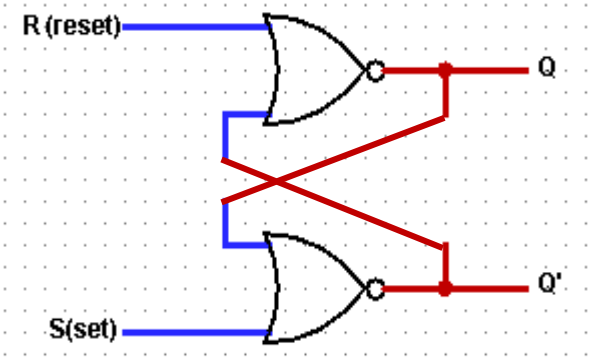


Figure 4.3 : Circuit logique de la bascule SR (NOR).

Tableau 4.1 : Table de vérité de la bascule SR (NOR)

S	R	Q	$\bar{Q}$	Remarques
1	0	1	0	**Mise à 1**
0	0	1	0	**Mémoire**
0	1	0	1	**Mise à 0**
0	0	0	1	**Mémoire**
1	1	0	0	**Interdit**

Le graphe de l'évolution de la bascule SR (NOR) est donné dans la figure 4.4 :

Figure 4.4: Diagramme d'état de la bascule SR (NOR).

La bascule SR utilisant deux portes NAND est illustrée dans la figure 4.5 et le tableau 4.2.

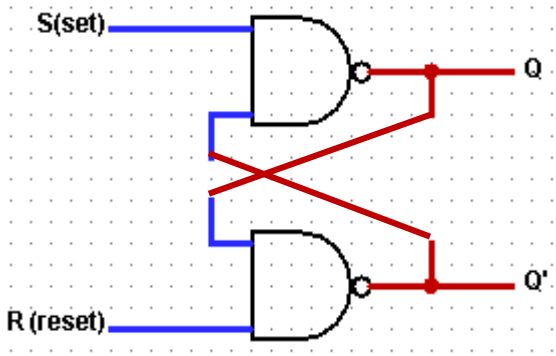


Figure 4.5 : Circuit logique de la bascule SR (NAND).

Tableau 4.2 : Table de vérité de la bascule SR (NAND)

S	R	Q	$\bar{Q}$	Remarques
1	0	0	1	**Mise à 0**
1	1	0	1	**Mémoire**
0	1	1	0	**Mise à 1**
1	1	1	0	**Mémoire**
0	0	1	1	**Interdit**

Condition 1 (S = 0, R = 1): En mettant S=0 à l'entrée, la sortie Q passe à 1, ce qui place la bascule dans l'état set. Lorsque l'entrée S repasse à 1, le circuit reste à l'état précédant (mémoire).

Condition 2 (S = 1, R = 0): En plaçant R=0, le circuit passe à l'état de reset et y reste même si les deux entrées reviennent à 1.

Condition 3 (S = 0, R = 0) : La condition interdite pour la bascule NON-ET, est que les deux entrées soient égales à 0 en même temps, une combinaison d'entrée qui doit être écartée.

En comparant la bascule NAND avec la bascule NOR, on note que les signaux d'entrée de la bascule NAND, nécessitent le complément des valeurs utilisées, pour la bascule NOR.

Le fonctionnement de la bascule SR (NAND) de base peut être modifié. La figure 4.6, présente une bascule SR avec une entrée de commande. Ainsi, l'entrée de contrôle ‘En’ agit comme un

signal de validation pour les 2 autres entrées. Les sorties des portes NAND restent au niveau logique 1 tant que le signal de validation reste à 0. C'est l'état de mémoire de la bascule SR. Lorsque l'entrée de validation passe à 1, l'information provenant de l'entrée S ou R est autorisée à affecter le changement. De plus, lorsque En = 1 et que les deux entrées S et R sont égales à 0, l'état du circuit ne change pas. Ces conditions sont énumérées dans le tableau 4.3, des fonctions accompagnant le schéma suivant.

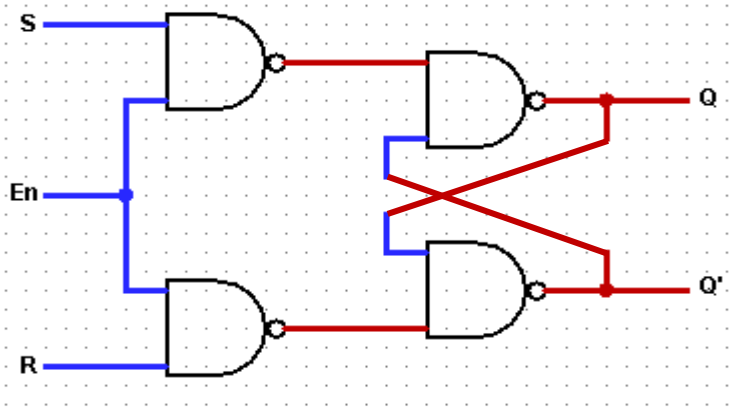


Figure 4.6: Circuit logique de la bascule SR (NAND) avec commande En.

Tableau 4.3 : Table de fonctionnement de la bascule SR avec commande En.

En	S	R	Q	$\bar{Q}$	Remarques
0	X	X	Q	$\bar{Q}$	**Mémoire**
1	0	0	Q	$\bar{Q}$	**Mémoire**
1	0	1	0	1	**Mise à 0**
1	1	0	1	0	**Mise à 1**
1	1	1	0	0	**Interdit**

Néanmoins, la bascule SR est un circuit important, car d'autres latch et flip-flop utiles sont construits à partir de celle-ci.

3 Bascule D

Un moyen de remédier à la condition de l'état interdit dans la bascule SR est de s'assurer que les entrées S et R ne soient jamais égaux à 1 en même temps. Cette opération est réalisée par la bascule D, illustrée à la figure 4.7.

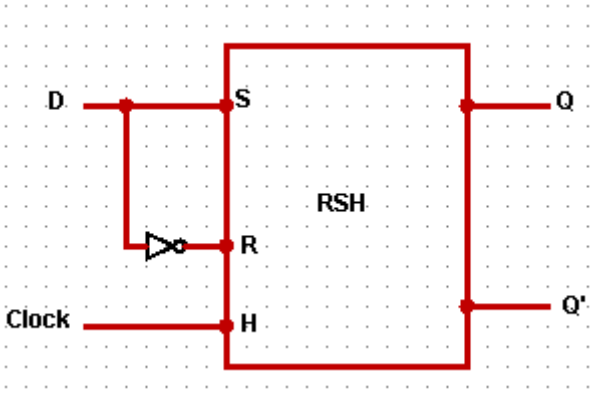


Figure 4.7: Schéma logique de la bascule D.

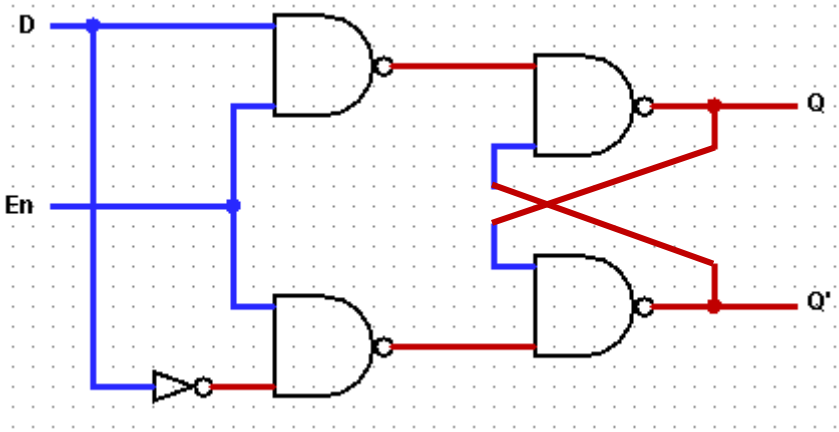


Figure 4.8: Circuit logique de la bascule D.

Tableau 4.4 : Table de fonctionnement de la bascule D.

En	D	Q	$\bar{Q}$	Remarque
0	X	Q	$\bar{Q}$	**Mémoire**
1	0	0	1	**Mise à 0**
1	1	1	0	**Mise à 1**

Cette bascule ne possède que deux entrées : D (data) et En (enable). L'entrée D va directement à l'entrée S, et son complément est appliqué à l'entrée R (Fig. 4.8). Lorsque l'entrée de validation

est à 0, les deux entrées de la bascule SR sont au niveau 1 et le circuit ne peut pas changer d'état quelle que soit la valeur de D. Si En = 1, alors l'entrée D est copiée à la sortie (Tableau 4.4).

Le latch D, doit sa qualification à sa faculté de mémoriser des données dans la mémoire interne.

Le graphe de l'évolution de la bascule D, est donné dans la figure 4.9 :

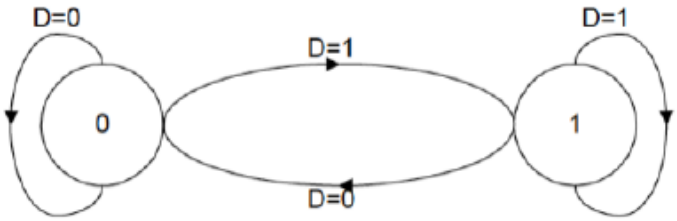


Figure 4.9: Diagramme d'état de la bascule D.

3.1 Flip-Flop D

L'état d'un latch ou d'un flip-flop est commuté par un changement de l'entrée de commande. Ce changement temporaire est appelé déclencheur. Le latch D, avec une entrée à impulsion est en fait un flip-flop, qui est déclenché chaque fois que l'impulsion passe au niveau logique 1. Si le niveau de l'entrée d'impulsion reste inchangé, tout changement en entrée modifiera la sortie et l'état de la bascule.

Il existe deux façons de modifier une bascule pour former un flip-flop. La première consiste à utiliser deux latchs dans une configuration spéciale, qui isole la sortie du flip-flop et l'empêche d'être affectée, lorsque l'entrée du flip-flop change. Une autre façon est de produire une bascule, qui ne se déclenche que pendant une transition de signal (de 0 à 1 ou de 1 à 0) du signal de synchronisation (horloge), et qui est désactivée pendant le reste de l'impulsion d'horloge. Nous allons maintenant montrer l'implémentation de ces deux types de bascules.

3.1.1 Bascule Maitre - esclave

La figure 4.10, montre un flip-flop D, composé de deux latchs D et d'un inverseur. La première bascule est appelée maître et la seconde esclave. Le circuit échantillonne l'entrée D et modifie sa sortie Q uniquement sur le niveau bas de l'horloge de synchronisation (désignée par Clk). Lorsque l'horloge est à 0, la sortie de l'inverseur est à 1. La bascule esclave est activée, et sa sortie Q est égale à la sortie maître Y. Lorsque l'horloge passe au niveau logique 1, les données

venant de l'entrée D sont transférées vers la sortie maitre. La sortie de la bascule correspond à la valeur stockée dans le bloc maître juste avant le passage au niveau négatif.

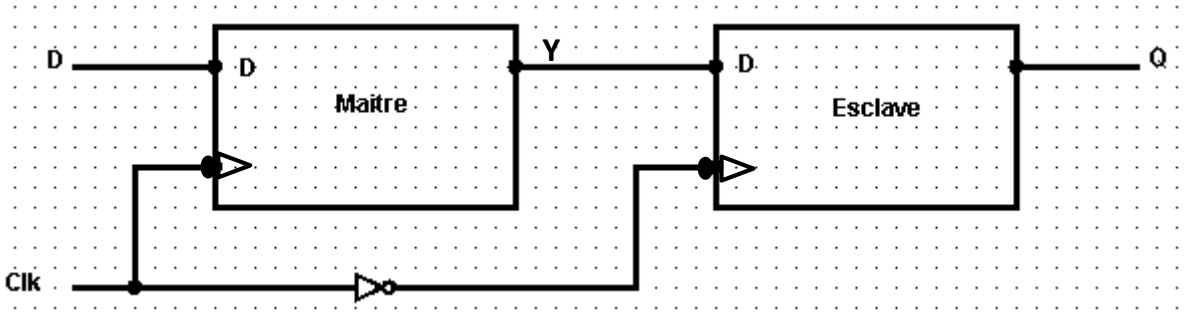


Figure 4.10 : Schéma logique Maitre-Esclave.

Le comportement de la bascule maître-esclave décrite ci-dessus, impose que, (1) la sortie ne peut changer qu'une seule fois, (2) un changement de la sortie est déclenché par le niveau bas de l'horloge et le changement ne peut se produire que pendant ce niveau d'horloge. Il est également possible de concevoir le même circuit de sorte que la sortie de la bascule change sur le niveau haut de l'horloge.

3.1.2 Bascule D à front d'horloge

Une autre structure de bascule D fonctionnant à front d'horloge et utilisant trois bascules SR est illustrée à la Fig. 4.11.

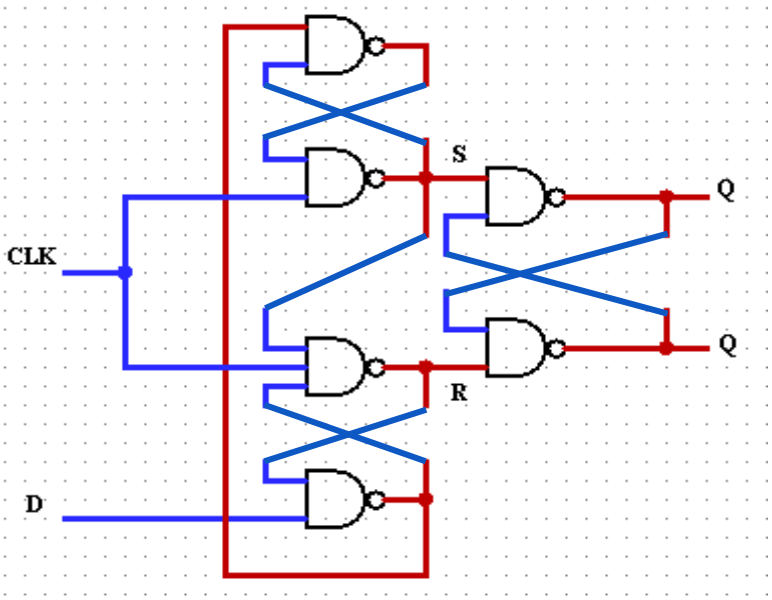


Figure 4.11 : Bascule D à front d'horloge montant.

1- Les entrées S et R de la bascule de sortie sont maintenues au niveau logique-1, lorsque Clk = 0, ce qui permet de garder la sortie à l'état de mémoire. L'entrée D peut prendre la valeur de 0 ou 1.
2- Si D = 0 lorsque Clk passe à 1, R passe à 0. Cela fait passer la bascule à l'état reset, ce qui fait que Q = 0.
3- Si D change d'état quand Clk = 1, la sortie R reste à 0 ainsi que l'état reset (Q =0). Ainsi, la bascule est verrouillée et ne répond plus aux changements de l'entrée D.
4- Lorsque Clk passe à 0, R passe à 1, plaçant la bascule de sortie à l'état de mémoire.

Les symboles graphiques de la bascule D à front montant et descendant sont représentés dans la figure 4.12.

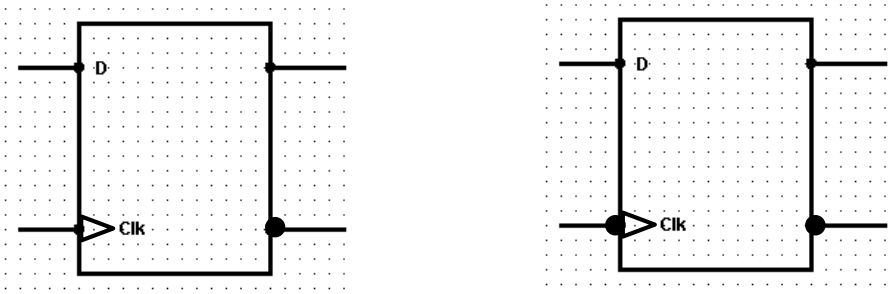


Figure 4.12 : Symbole logique de la bascule D à front montant (gauche) et front descendant (droite).

4 Bascule JK

Une bascule J-K se comporte de la même manière qu'une bascule R-S, à l'exception d'une entrée de la table de vérité. Dans le cas d'une bascule R-S, la combinaison d'entrée S = R = 1 est interdite. Dans le cas d'une bascule J-K, la sortie de la bascule est inversée quand J=K=1. Ainsi, une bascule J-K, permet de résoudre le problème de la combinaison d'entrées interdite de la bascule R-S.

La figure 4.13, montre le circuit de la bascule J-K en fonction de la bascule SR, ainsi que les fonctions et la table de vérité (Tableau 4.5).

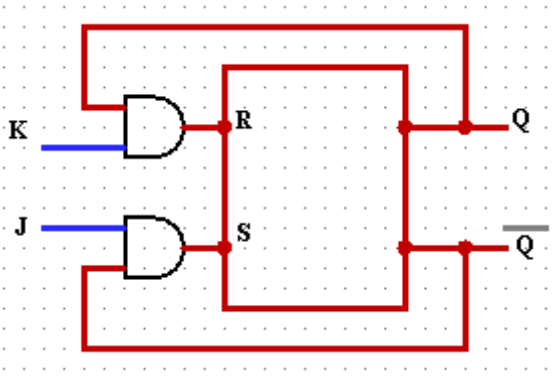


Figure 4.13 : Schéma logique de la bascule JK.

L'équation caractéristique de la bascule JK :

$$Q^+ = J.\bar{Q} + \bar{K}.Q$$

Tableau 4.5: Table de fonctionnement de la bascule JK.

J	K	Q^+	Fonction
0	0	Q	Mémoire
0	1	0	Mise à 0
1	0	1	Mise à 1
1	1	$\bar{Q}$	Inversion

Le graphe de l'évolution de la bascule JK est donné dans la figure 4.14.

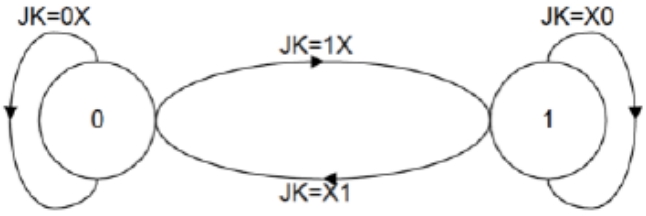


Figure 4.14: Diagramme d'état de la bascule JK.

5 Bascule T (Toggle)

La sortie d'un flip-flop toggle, également appelée flip-flop T, change d'état à chaque fois qu'il est activé à son entrée T (Tableau 4.6), C'est-à-dire que la sortie devient '1' si elle était '0' et '0' si elle était '1'. Le symbole graphique de cette bascule comporte un symbole T en entrée.

Tableau 4.6: Table de fonctionnement de la bascule T.

T	H	Q_{n+1}	Fonction
0	↑	Q_n	Mémoire
1	↑	$\overline{Q_n}$	$\overline{\text{Mémoire}}$

Son graphe d'évolution est présenté dans la figure 4.15.

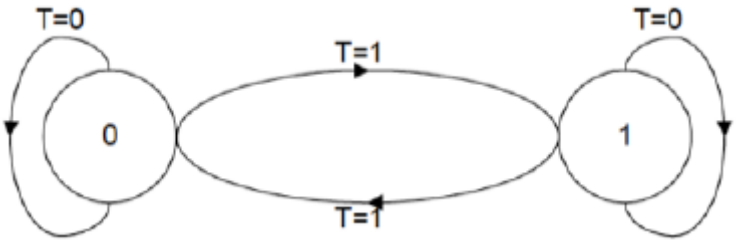


Figure 4.15: Diagramme d'état de la bascule T.

5.1 Flip-Flop T en fonction de J-K

En rappelant la table de vérité de la bascule J-K (voir le tableau 4.5), nous constatons qu'elle correspond à la bascule T, lorsque les deux entrées J et K de la bascule, sont liées à leur niveau actif.

La figure 4.16, présente l'utilisation d'une bascule T en fonction de la bascule JK.

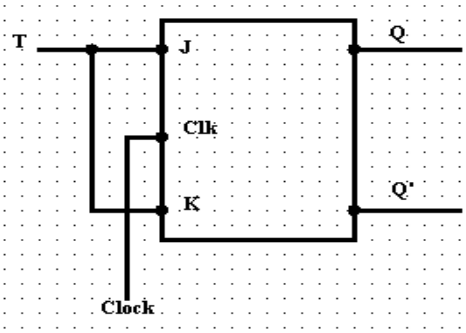


Figure 4.16: Diagramme d'état de la bascule T en fonction de JK.

5.2 Flip-Flop T en fonction de D

Le flip-flop T peut être construit avec un flip-flop D et une porte OU exclusif. L'expression d'entrée D est présentée ci-dessous:

$$D = T \oplus Q = T.\bar{Q} + \bar{T}.Q$$

Lorsque T = 0, D = Q et il n'y a pas de changement dans la sortie. Lorsque T = 1, D = Q et la sortie est inversée.

Le schéma graphique est montré dans la figure suivante:

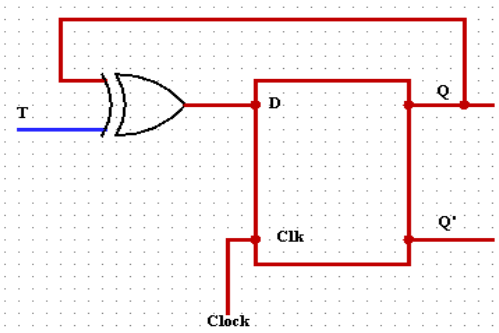


Figure 4.17: Diagramme d'état de la bascule T en fonction de D

Chapitre V

Analyse des circuits séquentiels

Introduction

Le comportement d'un circuit séquentiel synchrone est déterminé par les entrées, les sorties et l'état de ses bascules. Les sorties et l'état suivant sont tous deux fonction des entrées et de l'état présent. L'analyse d'un circuit séquentiel consiste à obtenir un tableau ou un diagramme de la séquence temporelle des entrées, des sorties et des états internes.

Dans ce chapitre, on présente une description algébrique des conditions d'état suivant en fonction de l'état actuel et des entrées. Une table d'état et un diagramme d'état sont ensuite présentés pour décrire le comportement du circuit séquentiel.

1 Machines d'états

Le modèle le plus général d'un circuit séquentiel comporte des entrées, des sorties et des états internes.

1.1 Équations d'État

Le fonctionnement d'un circuit séquentiel est décrit algébriquement par les équations d'état. Une équation d'état spécifie l'état suivant en fonction de l'état présent et des entrées.

On considère le circuit séquentiel suivant (Fig. 5.1), constitué de deux bascules D_A et D_B, d'une entrée x et d'une sortie y.

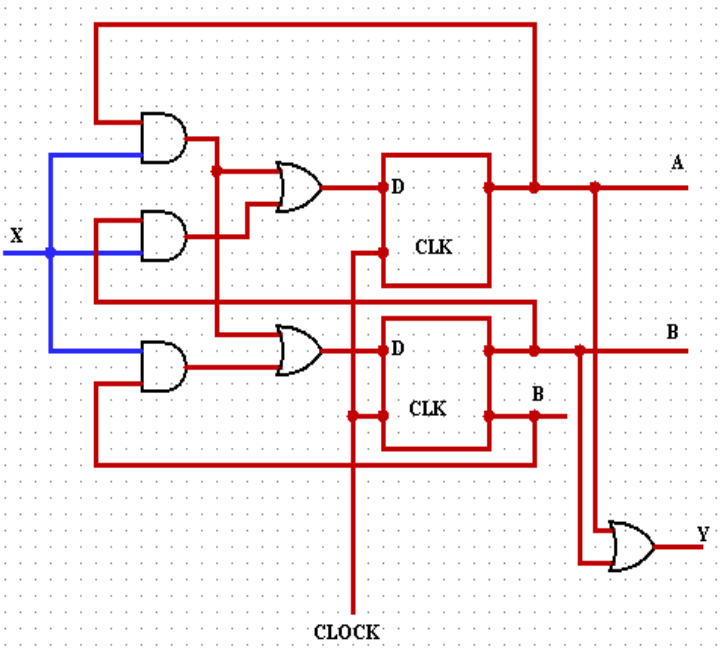


Figure 5.1 : Circuit séquentiel de deux bascules D.

Il est possible d'écrire un ensemble d'équations d'état pour le circuit, tel que :

- **Equations d'entrée de chaque bascule:**
  $D_A = A(t).X + B(t).X,$
  $D_B = A(t).X + \bar{B}(t).X,$
- **Equations caractéristique de la bascule D** (voir chapitre IV, section 3)**:**
  $A(t+1) = D_A$,
  $B(t+1) = D_B$
- **Equations de sortie** (état suivant des bascules)**:**
  $A(t+1) = A(t).X + B(t).X$,
  $B(t+1) = A(t).X + \bar{B}(t).X,$
  $Y = A(t) + B(t).$

1.2 Table d'État

La séquence des entrées, sorties et états des bascules peut être décrits dans une table d'état (Tableau 5.1).

Tableaux 5.1 : La table d'état de la figure 5.1.

Etat Présent		X	Etat Suivant		Y
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	1	1	1

Etat Présent		Etat Suivant				Output	
		X=0		X=1		X=0	X=1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	0	1	1
1	0	0	0	1	1	1	1
1	1	0	0	1	1	1	1

Le tableau se compose de quatre sections intitulées, état présent, entrée, état suivant et sortie. La partie d'état présent, montre les états des bascules A et B à un instant t donné. La partie entrée, donne la valeur de X pour chaque état présent possible. La partie d'état suivant, montre les états de bascules après un cycle d'horloge, au temps t + 1. Les valeurs de l'état suivant sont déterminées à partir des équations d'état. L'état suivant de la bascule A, doit satisfaire l'équation

d'état suivante A(t+1). La partie sortie indique la valeur de Y au temps t pour chaque état présent et chaque condition d'entrée.

1.3 Diagramme d'état

Les informations disponibles dans une table d'état peuvent être représentées graphiquement sous la forme d'un diagramme d'état. Dans ce type de diagramme, un état est représenté par un cercle, et les transitions (déclenchées par l'horloge) entre les états sont indiquées par des lignes directes reliant les cercles. Le diagramme d'état fournit les mêmes informations que le tableau d'état et s'obtient, dans notre exemple, directement à partir du tableau 5.1 ou du tableau 5.1. Le nombre binaire à l'intérieur de chaque cercle identifie l'état des flip-flops en numérateur et la sortie en dénominateur. Les lignes orientées sont étiquetées avec la valeur d'entrée pendant l'état présent.

Par exemple (Fig. 5.2), la ligne orientée de l'état 00 à 01 est étiquetée par une entrée 1 et une sortie 0, ce qui signifie que lorsque le circuit séquentiel est dans l'état actuel 00 et que l'entrée est égale à 1, la sortie est égale à 0. Après le cycle d'horloge suivant, le circuit passe à l'état suivant, 01.

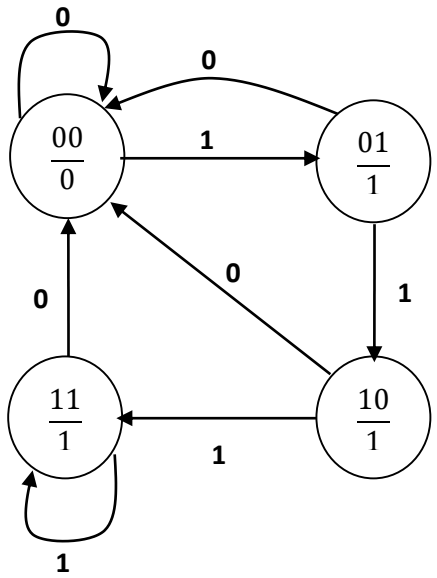


Figure 5.2 : Diagramme d'état de la figure 5.1.

Les étapes présentées dans cet exemple sont résumées ci-dessous :

Circuit logique → Equations et Table d'état → Diagramme d'état

Il est usuel de distinguer deux modèles de circuits séquentiels : le modèle de Moore et le modèle de Mealy. Les deux modèles de circuit séquentiel sont communément appelés machines d'états finis.

1.4 Machine de Moore

Dans la machine de Moore, l'état de sortie futur n+1 est déterminé uniquement à partir de l'état présent n. Les sorties '**Si**' sont une combinaison logique de la sortie des bascules et changent de manière synchrone avec le front actif de l'horloge (Fig. 5.3). La machine de Moore est plus adaptée pour réaliser une machine d'état totalement synchrone.

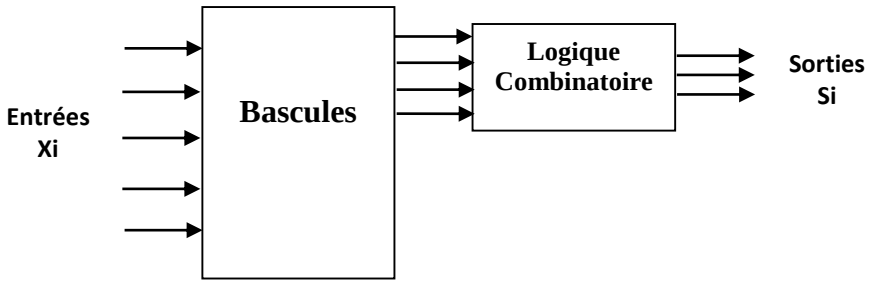


Figure 5.3: machine de Moore.

Diagramme et la table d'état d'une machine de MOORE est présenté dans la figure 5.4 et le tableau 5.2.

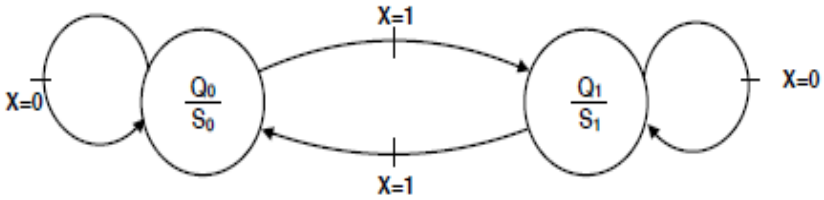


Figure 5.4: Diagramme d'état de la machine de Moore.

Tableau 5.2 : Table d'état d'une machine de MOORE:

Etats Présent	Entrée X	Etats suivant	Sorties Si
Q_0	0	Q_0	S_0
Q_0	1	Q_1	S_0
Q_1	0	Q_1	S_1
Q_1	1	Q_0	S_1

Exemple : Analyse avec un T flip-flop

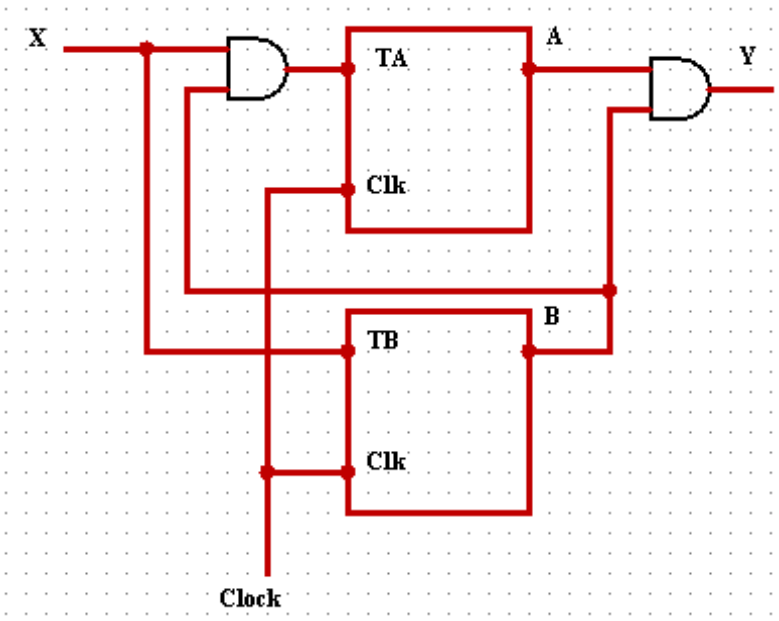


Figure 5.5 : Circuit séquentiel de deux bascules T.

a. Equations d'entrée de chaque bascule:

$T_A = B(t).X,$

$T_B = X,$

b. Equations caractéristique de la bascule T (voir chapitre IV, section 5)**:**

$A(t+1) = T_A \oplus A$,

$B(t+1) = T_B \oplus B$

c. Equations de sortie:

$A(t+1) = B(t).X \oplus A(t)$,

$B(t+1) = B(t) \oplus X,$

$Y = A(t).B(t).$

d. Table d'état:

Pour chaque état présent A et B et pour chaque entrée X = 0 ou 1 on, applique les équations de sorties pour obtenir les états suivants et sortie Y (Tableau 5.3).

Tableau 5.3 : Table d'état de la figure 5.5.

Etats Présent		Entrées	Etats Suivant		Sorties
A	**B**	**X**	**A**	**B**	**Y**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

e. Diagramme d'état:

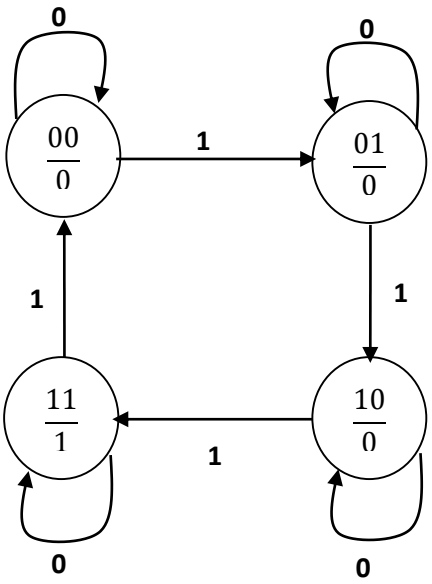


Figure 5.6: Diagramme d'état de la figure 5.5.

1.5 Machine de Mealy

Les sorties dépendent de l'état présent n et de la valeur des entrées (Fig. 5.7). La sortie peut donc changer de manière asynchrone en fonction de la valeur des entrées.

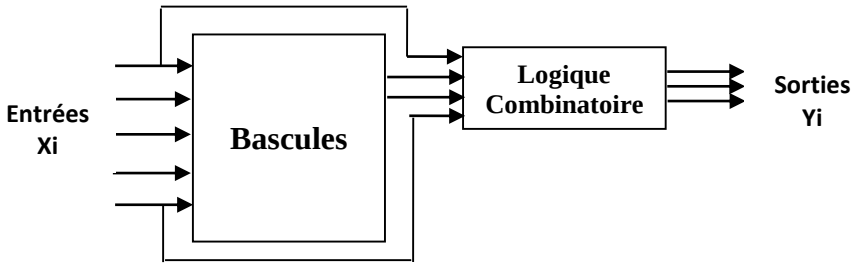


Figure 5.7: machine de Mealy.

Diagramme et la table d'état d'une machine de MEALY est présenté dans la figure 5.8 et le tableau 5.4.

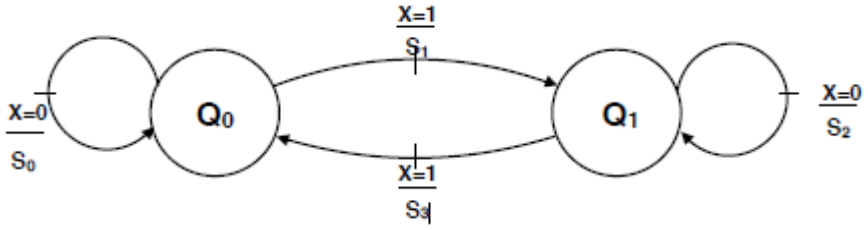


Figure 5.8: Diagramme d'état de la machine de Mealy.

Tableau 5.4 : Table d'état d'une machine de MEALY :

Etats Présent	Entrée X	Etats suivant	Sorties Si
Q_0	0	Q_0	S_0
Q_0	1	Q_1	S_1
Q_1	0	Q_1	S_2
Q_1	1	Q_0	S_3

Exemple : Analyse avec un D flip-flop

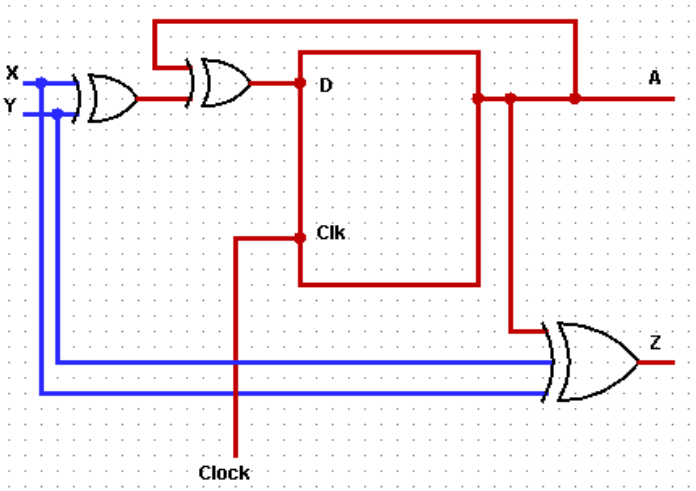


Figure 5.9 : Circuit séquentiel d'une bascule D

a. Equations d'entrée de la bascule D:

$D = A(t) \oplus X \oplus Y$,

b. Equations caractéristique de la bascule D (voir chapitre IV, section 3)**:**

$A(t+1) = D$,

c. Equations de sortie:

$A(t+1) = A(t) \oplus X \oplus Y$,

$Z = A(t) \oplus X \oplus Y$.

d. Table d'état:

Pour chaque état présent A et pour chaque entrée X et Y, on applique les équations de sorties pour obtenir l'état suivant A(t+1) et la sortie Z

Tableau 5.5 : Table d'état de la figure 5.9.

Etats Présent		Entrées	Etats Suivant	Sorties
A	**X**	**Y**	**A**	**Z**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

e. Diagramme d'état:

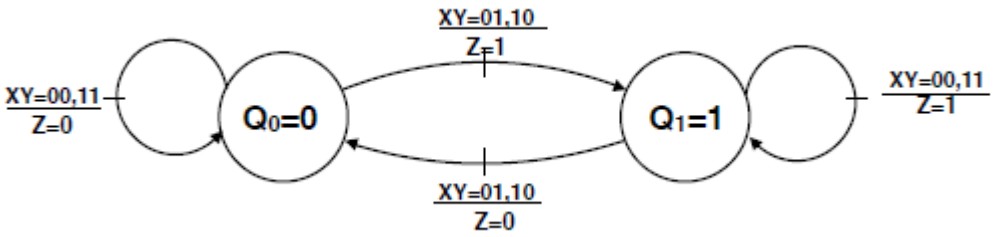


Figure 5.10: Diagramme d'état de la figure 5.9.

2 Réduction et affectations de l'état

Dans la conception de machine d'état, on retrouve souvent des états redondants, c-à-dire, deux ou plusieurs états présents donnent les mêmes états suivants et les mêmes sorties.

Pour réduire la table d'état et simplifier le circuit logique, il est donc important de supprimer les états redondants.

Soit le diagramme d'état suivant (Fig. 5.11) et sa table d'état (Tableau 5.6):

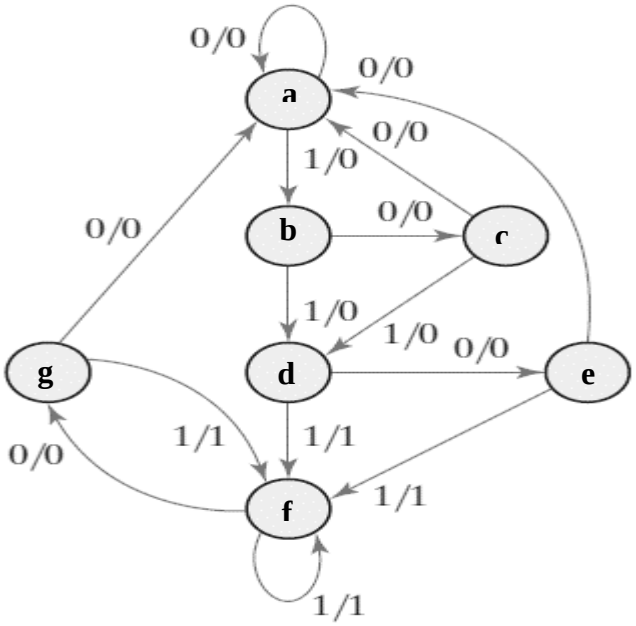


Figure 5.11 : Diagramme d'états

Tableau 5.6 : Table d'état de machine étape 1**:**

Etats Présent	Etats Suivant		Sorties	
	x=0	x=1	x=0	x=1
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*g*	*f*	0	1
g	*a*	*f*	0	1

Les états ‘e’ et ‘g’ donnent les mêmes états suivants (a quand x=0 et f quand x=1) et les mêmes sorties (0 quand x=0 et 1 quand x=1).

Donc g=e, on remplace l’état g par l’état e et on obtient la table d’état ‘2’ (Tableau 5.7).

Tableau 5.7 : Table d’état de machine étape 2

Etats Présent	Etats Suivant		Sorties	
	x=0	x=1	x=0	x=1
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*f*	0	1
e	*a*	*f*	0	1
f	*e*	*f*	0	1

Les états ‘d’ et ‘f’, donnent les mêmes états suivants et les mêmes sorties. Donc d=f, on remplace l’état f par l’état d, et on obtient la table d’état ‘3’.

Tableau 5.8 : Table d’état de machine étape 3**:**

Etats Présent	Etats Suivant		Sorties	
	x=0	x=1	x=0	x=1
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*d*	0	1
e	*a*	*d*	0	1

La table d’état ‘3’, ne comporte plus d’états redondants. On obtient alors le diagramme d’état réduit présenté dans la figure 5.12.

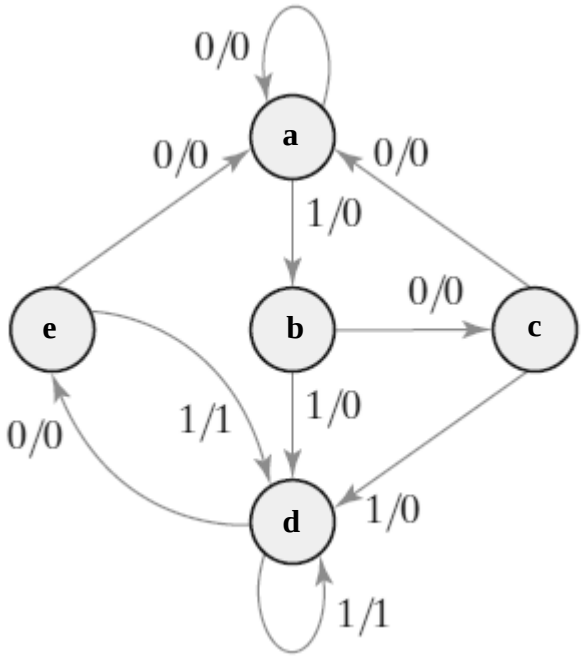


Figure 5.12: Diagramme d'états de la figure 5.11 réduit.

3 Procédure de conception

La conception d'un circuit séquentiel synchrone commence par un ensemble de caractéristiques, et se conclut par un diagramme logique ou une liste de fonctions booléennes, à partir desquelles, le diagramme logique peut être obtenu. La procédure de conception d'un circuit logique séquentiel est la suivante:

a) Diagramme d'état

b) Table d'état

c) Réduction des états

d) Codage des états

e) Table d'excitation

f) Equations logiques simplifiées

g) Circuit logique séquentiel

Exemple 1: Détection de trois '1' successifs avec chevauchement

1. Solution en utilisant la machine de MOORE:

a) Diagramme d'état:

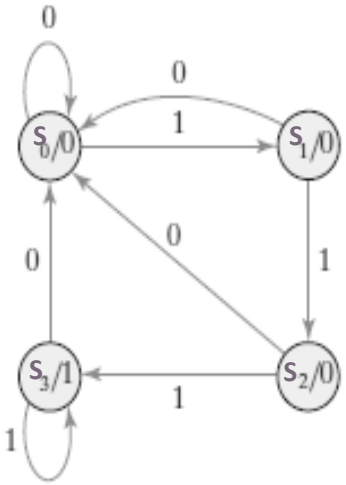


Figure 5.13: Diagramme d'états d'un détecteur de séquences (Machine Moore)

Explications:

- A chaque fois qu'un '0' est détecté la séquence retourne à l'état initial S_0.
- La machine de Moore est synchrone, alors la sortie Y =1 est observée dans l'état de machine suivant (front d'horloge actif suivant) la détection des trois '1' successifs.

b) Table d'état: les changements des 4 états de machine en fonction de l'entrée X sont résumés dans le tableau 5.9.

Tableau 5.9 : table d'état réalisée à partir du diagramme d'état:

Etats Présent	Etats Suivant		Y
	X=0	X=1	
S_0	S_0	S_1	0
S_1	S_0	S_2	0
S_2	S_0	S_3	0
S_3	S_0	S_3	1

c) Réduction des états:

Dans cet exemple, les états présents S2 et S3 donnent les mêmes états suivants S0 à X = 0 et S3 à X= 1 mais leurs sorties sont différentes. Alors, il n'y a pas de réduction dans la table d'état

d) Codage des états:

2 bits sont suffisants pour coder les 4 états de machine. S0 = 00, S1 = 01, S2 = 10, S3 = 11. Ce codage donne la table d'excitation suivante (Tableau 5.10)

e) Table d'excitation:

Tableau 5.10 : Table d'excitation utilisant la bascule D.

Etats Présent			**Etats Suivant**			**Bascules D**	
A	**B**	**x**	**A**	**B**	**y**	**D_A**	**D_B**
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	0	1	0	0
1	1	1	1	1	1	1	1

NB: Si 2^N est le nombre d'états de machine, alors N est le nombre de bascules. Dans cet exemple, il y a 4 (2^2) états de machine, donc nous avons besoin de 2 bascules D_A et D_B.

f) Equations logiques simplifiées

Bx / A	00	01	11	10
0			1	
1		1	1	

$D_A = A.x + B.x$

Bx / A	00	01	11	10
0		1		
1		1	1	

$D_B = A.x + \bar{B}.x$

Bx / A	00	01	11	10
0				
1			1	1

$y = A.B$

g) Circuit logique séquentiel

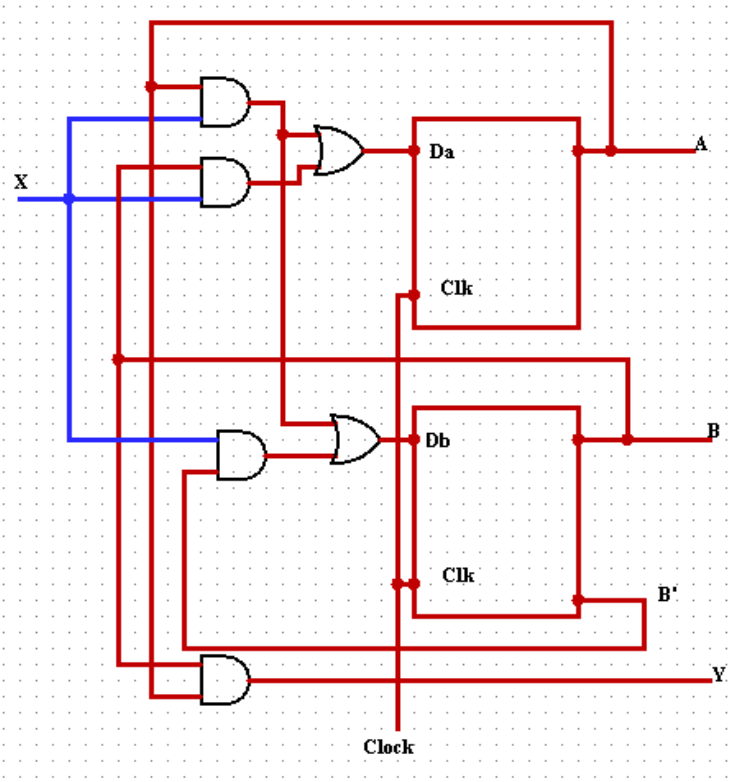


Figure 5.14 : Circuit séquentiel de la figure 5.13

2. Solution en utilisant la machine de MEALY:

a) Diagramme d'état:

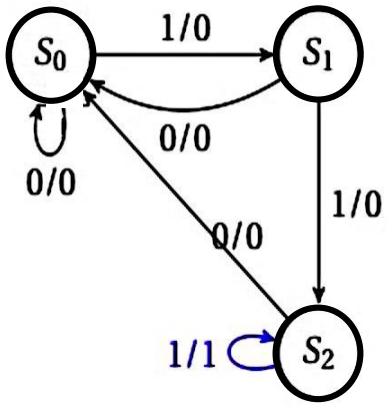


Figure 5.15 : Diagramme d'états d'un détecteur de séquences (Machine Mealy)

Explications:

- Comme pour la machine de Moore, à chaque fois qu'un '0' est détecté la séquence retourne à l'état initial S0.

- La machine de Mealy est asynchrone (contrairement à la machine de Moore). Dans ce cas, la sortie Y change directement avec le changement de l'entrée X et Y est = 1 directement après la détection du 3éme '1' successifs, sans attendre l'état de machine suivant (sans attendre le front d'horloge actif suivant).

b) Table d'état : les changements des 3 états de machine en fonction de l'entrée X, sont résumés dans le tableau 5.11.

Tableau 5.11 : table d'état réalisée à partir du diagramme d'état:

Etats Présent	Etats Suivants		Sorties Y	
	x=0	x=1	x=0	x=1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_0	S_2	0	1

c) Réduction des d'états:

Les états présents S1 et S2 donnent les mêmes états suivants S0 à X=0 et S2 à X=1 mais leurs sorties sont différentes à X=1. Alors, il n'y a pas de réduction d'états de machine.

d) Codage des états:

2 bits sont suffisants pour coder les 3 états de machine. S0 = 00, S1 = 01, S2 = 10. Ce codage donne la table d'excitation suivante (Tableau 5.12).

e) Table d'excitation:

Tableau 5.12 : Table d'excitation en utilisant la bascule JK.

Etats Présent		Entrées	Etats Suivant		Sorties	Bascule A		Bascule B	
A	B	x	A	B	Y	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	X
0	1	0	0	0	0	0	X	X	1
0	1	1	1	0	0	1	X	X	1
1	0	0	0	0	0	X	1	0	X
1	0	1	1	0	1	X	0	0	X

f) Equations logiques simplifiées

x \ AB	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_A = B.x$

x \ AB	00	01	11	10
0	X	X	X	1
1	X	X	X	0

$K_A = \bar{x}$

x \ AB	00	01	11	10
0	0	X	X	0
1	1	X	X	0

$J_B = \bar{A}.x$

x \ AB	00	01	11	10
0	X	1	X	X
1	X	1	X	X

$K_B = 1$

x \ AB	00	01	11	10
0	0	0	X	0
1	0	0	X	1

$Y = A.x$

g) Circuit logique séquentiel

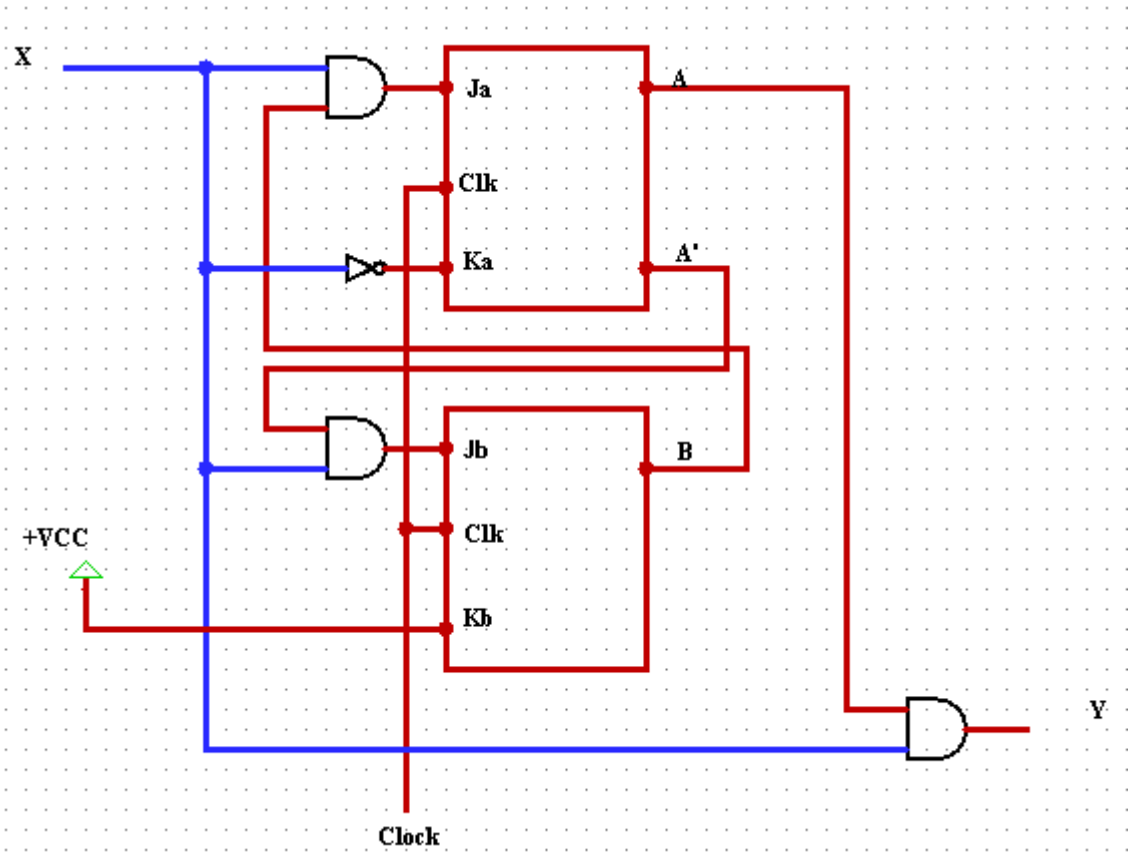
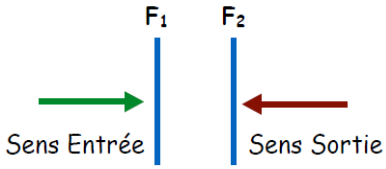


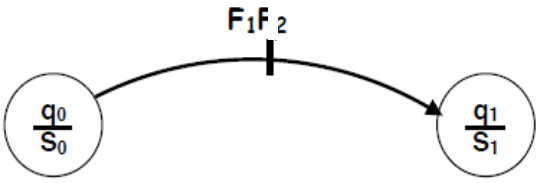
Figure 5.16 : Circuit séquentiel de la figure 5.15.

Exemple 2: Détection de personnes qui entrent dans un musée en utilisant la machine de Moore.

Pour que la personne soit détectée (S=1), il faut que les 2 faisceaux F_1 et F_2 soient coupés simultanément (F_1F_2 = 11).



Et le diagramme d'état doit suivre la représentation suivante:



a) Diagramme d'état:

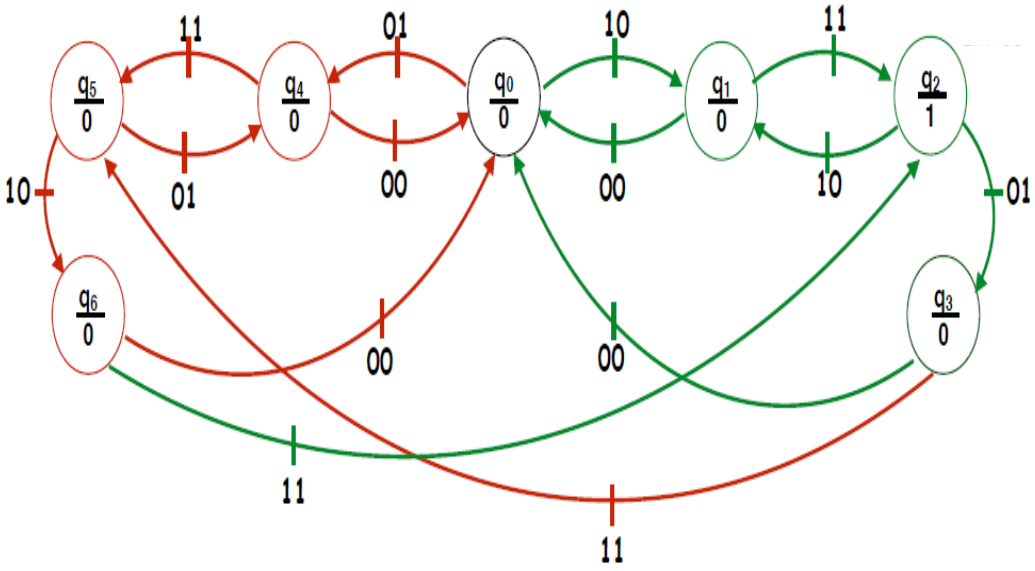


Figure 5.17 : Diagramme d'états détecteur d'entrée de personnes.

q_0: état initial

q_1: la personne franchit le premier faisceau F_1.

q_2: les deux faisceaux F_1F_2 sont franchis et sortie S= 1

q_3: la personne dépasse le premier faisceau F1, elle peut rentrer complètement dans le musée (c'est l'état initial q_0), comme elle peut faire marche arrière et ressortir du musée (c'est l'état q_5).

q_4 et **q_5**: la personne est dans le musée, elle franchit respectivement F_2 (q_4), ensuite F_1F_2 (q_5) et S=0, puisque seules les personnes qui entrent sont comptées.

q_6: La personne dépasse le faisceau F_2, elle peut sortir complètement dans le musée (c'est l'état initial q_0), comme elle peut faire marche arrière et re-entrer au musée (c'est l'état q_2).

b) Table d'état :

Tableau 5.13 : Table d'état du diagramme de la figure 5.17

Etats Présent	**Etats suivant**				**Sorties S**
	F_1F_2=00	**F_1F_2=01**	**F_1F_2=11**	**F_1F_2=10**	
q_0	q_0	q_4	-	q_1	0
q_1	q_0	-	q_2	q_1	0
q_2	-	q_3	q_2	q_1	1
q_3	q_0	q_3	q_5	-	0
q_4	q_0	q_4	q_5	-	0
q_5	-	q_4	q_5	q_6	0
q_6	q_0	-	q_2	q_6	0

c) Réduction des d'états:

- Les états présents q_0 et q_1 donnent les mêmes états suivants et les mêmes sorties, alors on enlève q_1 et on remplace q_1 par q_0.
- Les états présents q_4 et q_5 donnent les mêmes états suivants et les mêmes sorties, alors on enlève q_5 et on remplace q_5 par q_4.

On obtient le tableau suivant:

Tableau 5.14 : table d'états réduite étape 1.

Etats Présent	Etats suivant				Sorties S
	F1F2=00	F1F2=01	F1F2=11	F1F2=10	
q_0	q_0	q_4	q_2	q_0	0
q_2	-	q_3	q_2	q_0	1
q_3	q_0	q_3	q_4	-	0
q_4	q_0	q_4	q_4	q_6	0
q_6	q_0	-	q_2	q_6	0

D'après le tableau 5.14, si on prend $q_0 = q_6$ et si on prend $q_3 = q_4$, on obtient la table d'état réduite présentée dans le tableau 5.15.

Tableau 5.15 : table d'états réduite étape 2.

Etats Présent	Etats suivant				Sorties S
	F1F2=00	F1F2=01	F1F2=11	F1F2=10	
q_0	q_0	q_3	q_2	q_0	0
q_2	-	q_3	q_2	q_0	1
q_3	q_0	q_3	q_3	q_0	0

Les états présents q_0 et q_2 ne peuvent pas être compatibles, ils ont les mêmes états suivants mais leurs sorties sont différentes.

d) Codage des états:

2 bits sont suffisants pour coder les 3 états de machine. q0 = 00, q2 = 01, q3 = 10. Ce codage donne la table d'excitation suivante (Tableau 5.16)

e) Table d'excitation:

Tableau 5.16 : table d'excitation en utilisant la bascule T.

Etats Présent		Entrées		Etats Suivant		Sorties	Bascules T	
A	B	F_1	F_2	A	B	S	T_A	T_B
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	1	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	1	0	0	1
0	1	0	0	X	X	1	X	X
0	1	0	1	1	0	1	1	1
0	1	1	0	0	0	1	0	1
0	1	1	1	0	1	1	0	0
1	0	0	0	0	0	0	1	0
1	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	1	0
1	0	1	1	1	0	0	0	0

f) Equations logiques simplifiées :

AB / F1F2	00	01	11	10
00	0	X	X	1
01	1	1	X	0
11	0	0	X	0
10	0	0	X	1

$$T_A = A.\overline{F_2} + \bar{A}.\overline{F_1}.F_2$$

AB / F1F2	00	01	11	10
00	0	X	X	0
01	0	1	X	0
11	1	0	X	0
10	0	1	X	0

$$T_B = B.(\overline{F_1} + \overline{F_2}) + \bar{A}.\bar{B}.(F_1.F_2)$$

AB / F1F2	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	0	1	X	0
10	0	1	X	0

$$S = B$$

g) Circuit logique séquentiel

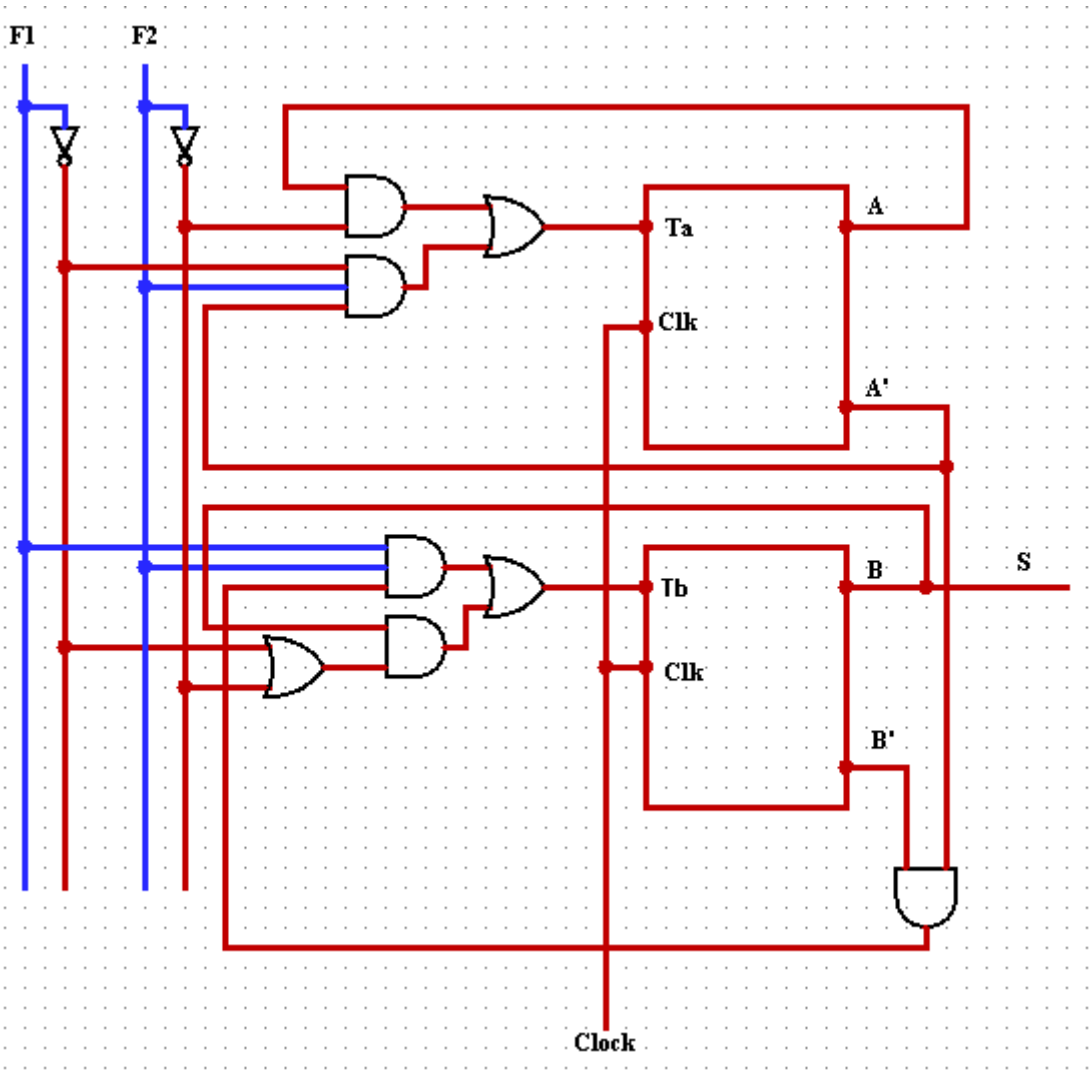


Figure 5.18: Circuit séquentiel de la figure 5.17.

Chapitre VI

Compteurs et registres

Introduction

Les compteurs et les registres comprennent une disposition en cascade de plusieurs bascules, avec ou sans composants de logique combinatoire. Tous deux constituent des éléments essentiels de la logique séquentielle. Les compteurs sont principalement utilisés dans les applications de comptage, où ils mesurent l'intervalle de temps entre deux instants inconnus, où la fréquence d'un signal donné. Les registres sont principalement utilisés pour le stockage temporaire des données présentes à la sortie d'un circuit numérique, avant qu'elles ne soient transmises à un autre circuit numérique.

Ce chapitre traite des différents types de compteurs et de registres en termes de fonctionnement de base, de méthodologie de conception et d'aspects liés aux applications.

1 Compteurs

Les compteurs sont classés en deux catégories selon leur mode de fonctionnement. On distingue les compteurs asynchrones et les compteurs synchrones.

1.1 Compteurs asynchrones

Dans un compteur asynchrone ou un compteur série, l'entrée d'horloge est appliquée uniquement à la première bascule, aussi appelée bascule d'entrée. L'entrée d'horloge de toute bascule suivante provient de la sortie de sa bascule précédente.

La figure 6.1, montre le schéma de connexion d'horloge pour obtenir un compteur asynchrone avec propagation de l'ordre de changement d'état en série.

La première bascule (A) est synchronisée par une horloge externe mais le déclenchement des autres bascules (B et C) est déterminé par une combinaison logique des sorties des bascules précédentes (QA et QB).

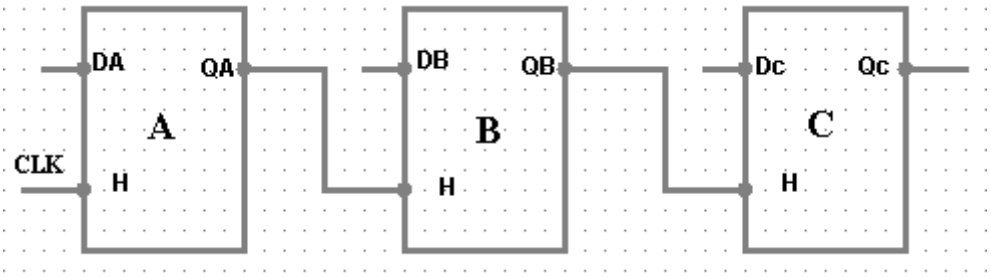


Figure 6.1 : Schéma logique d'un compteur asynchrone.

La propagation de l'ordre de changement d'état se fait en cascade. Les sorties des bascules ne changent pas d'état exactement en même temps, car elles ne sont pas reliées au même signal d'horloge.

Exemple : Le schéma de la figure 6.2, représente un exemple de compteur à 3 bascules JK.

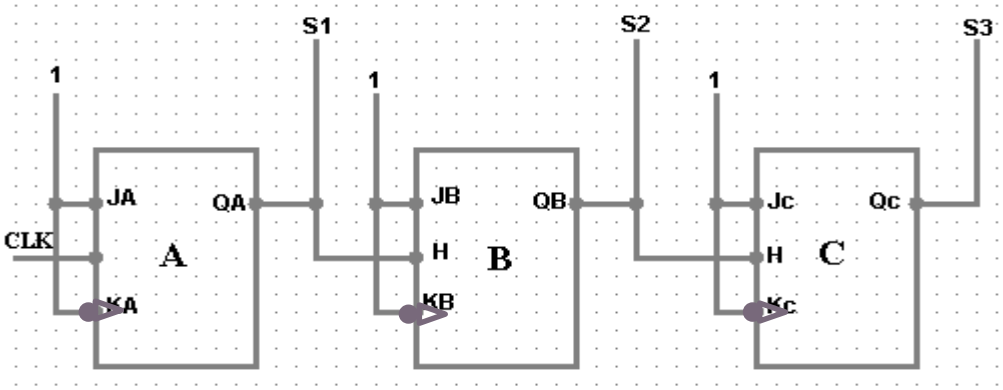


Figure 6.2 : Compteur JK asynchrone.

Les entrées J et K sont à 1 (état d'inversion, voir la section 4 du chapitre IV), donc les bascules changent d'état à chaque front actif de leur horloge (front descendant). La bascule S_1 change d'état à chaque front descendant de l'horloge externe, la bascule S_2 à chaque front descendant de S_1 et la bascule S_3 à chaque front descendant de S_2, comme le montre le chronogramme suivant.

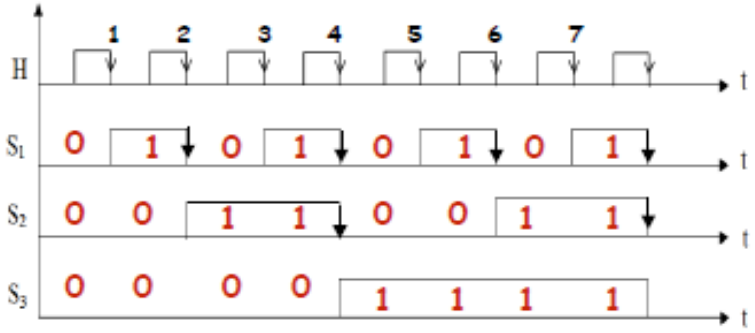


Figure 6.3 : chronogramme d'un compteur JK asynchrone (3bits).

1.2 Compteurs synchrones

Le compteur synchrone, également appelé compteur parallèle, permet à toutes les bascules du compteur de changer d'état en même temps, en synchronisation avec le signal d'horloge d'entrée. Dans un tel cas, le signal d'horloge est appliqué simultanément aux entrées d'horloge de toutes les bascules. Le retard impliqué dans ce cas est égal au délai de propagation d'une seule bascule,

quel que soit le nombre de bascules utilisées pour construire le compteur. En outre, le retard est indépendant de la taille du compteur.

La figure 6.4, montre le schéma de connexion d'horloge pour obtenir un compteur synchrone avec propagation de l'ordre de changement d'état en parallèle.

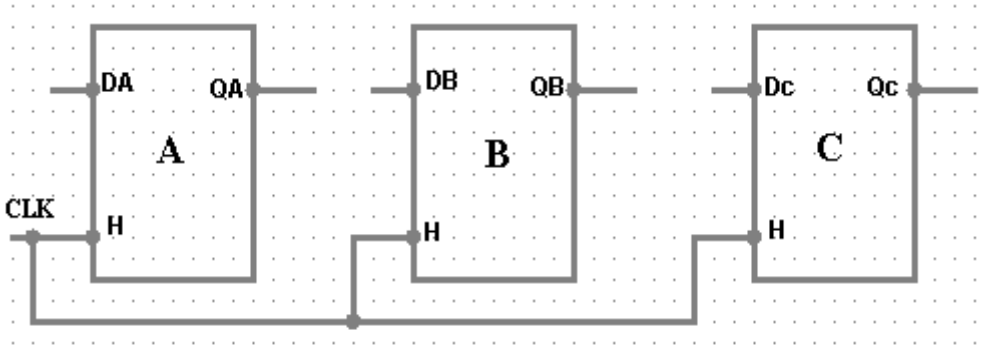


Figure 6.4 : Compteur D synchrone.

- ***Compteur synchrone modulo N***

On appelle compteur modulo N, un compteur qui compte de 0 à N-1.

Exemple 1: Conception d'un circuit qui compte de 0 à 7 (modulo 8)

1) **Diagramme d'état**

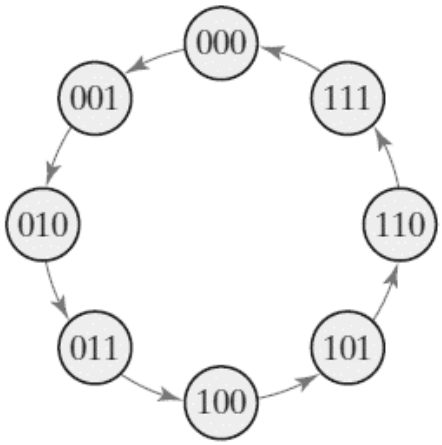


Figure 6.5 : diagramme d'états d'un compteur modulo 8.

a) Table d'état et d'excitation :

Tableau 6.1 : Table d'état du diagramme de la figue 6.5

Etats Présent			Etats Suivant			Bascules T		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

b) Equations logiques simplifiées

A_2 \ A_1A_0	00	01	11	10
0	0	0	1	0
1	0	0	1	0

A_2 \ A_1A_0	00	01	11	10
0	0	1	1	0
1	0	1	1	0

A_2 \ A_1A_0	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$$T_{A2} = A_1.A_0 \qquad T_{A1} = A_0 \qquad T_{A0} = 1$$

c) Circuit logique séquentiel

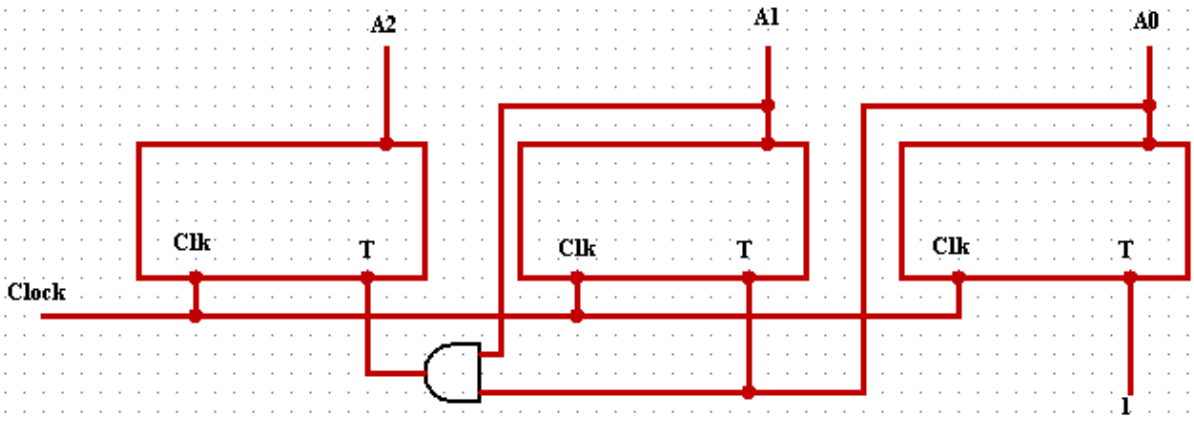


Figure 6.6 : Circuit séquentiel de la figure 6.5.

Exemple 2: Conception d'un compteur modulo 5 (qui compte de 0 à 4)

1) Diagramme d'état

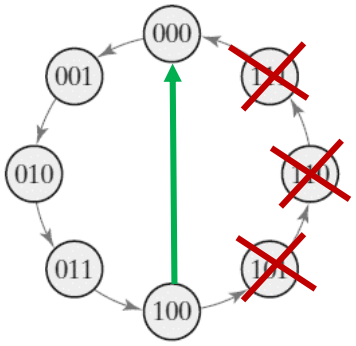


Figure 6.7 : Diagramme d'états d'un compteur modulo 5.

a) Table d'état et d'excitation :

Tableau 6.2 : Table d'état du diagramme de la figure 6.7.

Etats Présent			Etats Suivant			Bascules T		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	0	0	0	1	0	0

b) Equations logiques simplifiées

A_2 \ A_1A_0	00	01	11	10
0	0	0	1	0
1	1	X	X	X

$$T_{A2} = A_2 + A_1.A_0$$

A_2 \ A_1A_0	00	01	11	10
0	0	1	1	0
1	0	X	X	X

$$T_{A1} = A_0$$

A_2 \ A_1A_0	00	01	11	10
0	1	1	1	0
1	0	X	X	X

$$T_{A0} = \overline{A_2}$$

c) Circuit logique séquentiel

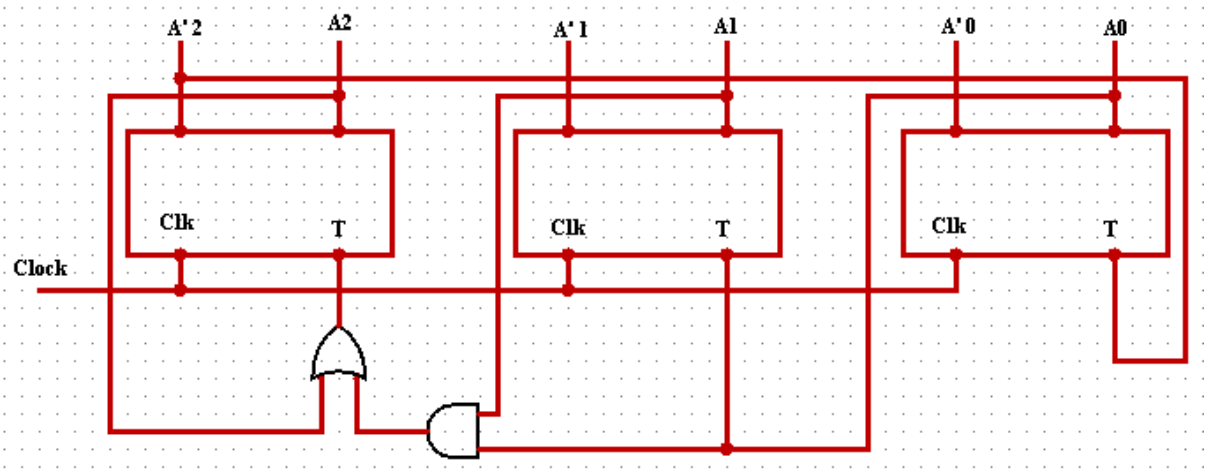


Figure 6.8 : Circuit séquentiel de la figure 6.7.

d) Analyse des états non utilisés:

Dans cet exemple (compteur de 0 à 4), les états sont codés sur 3 bits. En cas d'erreur, des états comme 5 (101), 6 (110) ou 7 (111), peuvent se produire. Pour cela, il est important de vérifier ces cas d'erreurs:

Par exemple en appliquant les équations logiques simplifiées pour le cas 101, on obtient : $T_{A2}=1$, $T_{A1}=1$ et $T_{A0}=0$

• Si l'état présent $A_2=1$ et $T_{A2}=1$ (inversion) alors l'état suivant $A_2=0$

• Si l'état présent $A_1=0$ et $T_{A1}=1$ (inversion) alors l'état suivant $A_1=1$

• Si l'état présent $A_0=1$ et $T_{A0}=0$ (mémoire) alors l'état suivant $A_0=1$

Tableau 6.3 : vérification des 3 cas d'erreurs

Etats Présent			Bascules T			Etats Suivant		
A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}	A_2	A_1	A_0
1	0	1	1	1	0	0	1	1
1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	0	0	1

D'après le tableau de vérification, les 3 cas d'erreurs donnent un état suivant stable.

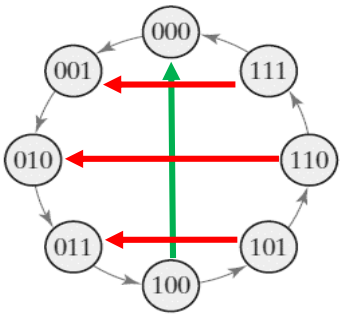


Figure 6.9 : Diagramme d'états d'un compteur modulo 5 après vérification des cas d'erreurs.

2 Registre

Un registre est un ensemble de bascules, qui partagent une horloge commune et qui sont capables de stocker des bits d'informations. Un registre à n bits est constitué d'un ensemble de n bascules capables de stocker n bits d'information binaire. En plus des bascules, un registre peut comporter des portes combinatoires qui effectuent certaines tâches de traitement de données. Différents types de registres sont disponibles mais les plus simples sont ceux qui se composent uniquement de bascules, sans portes. La classification des modes de fonctionnement des registres est la suivante :

2.1 Registres entrée série et sortie parallèle

Ce registre possède une entrée E, n sorties (Q0, Q1, Q2, Q3, …) et n bascules (Fig. 6.10). Les données binaires d'entrée sont introduites bit après bit. Les sorties sont toutes disponibles en même temps. Ces registres peuvent être utilisés pour faire une transformation série-parallèle des données.

Exemple : Registre à décalage 4 bits

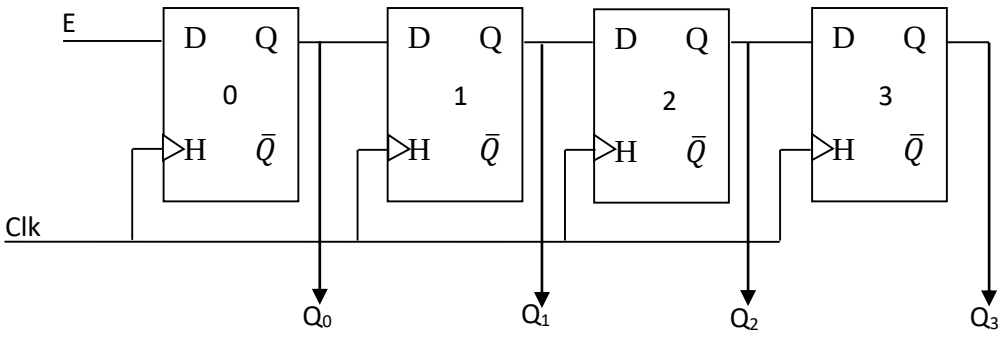


Figure 6.10 : Registre à décalage entrée série et sortie parallèle**.**

Tableau 6.4 : table de vérité d'un registre à décalage entrée série et sortie parallèle à 4 bits.

Entrée série	Clk	Q_0	Q_1	Q_2	Q_3
E_0	↑	b_0	0	0	0
E_1	↑	b_1	b_0	0	0
E_2	↑	b_2	b_1	b_0	0
E_3	↑	b_3	b_2	b_1	b_0
E_4	↑	b_4	b_3	b_2	b_1

2.2 Registres entrée parallèle - sortie série

Toutes les entrées (E_0, E_1, E_2, E_3), sont introduites en même temps dans le registre (Fig. 6.11). Les informations en sortie sur S, sont disponibles les unes après les autres au rythme de l'horloge.

Exemple : registre à décalage 4 bits

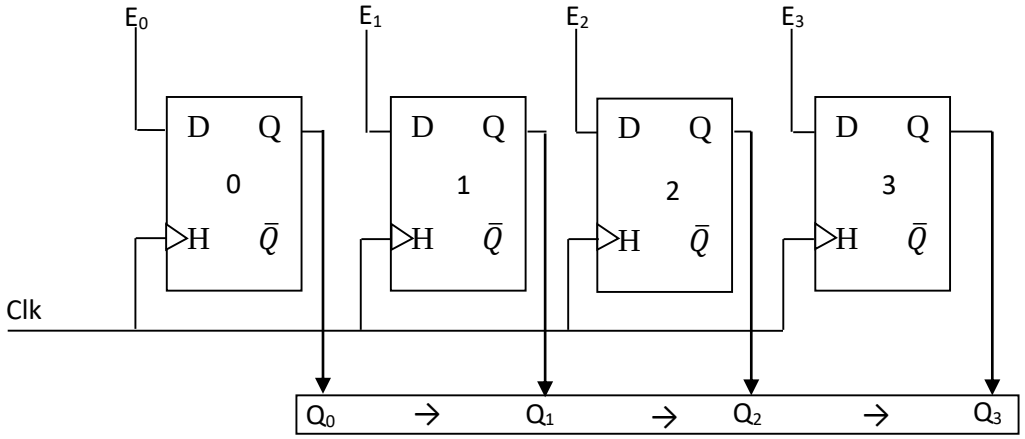


Figure 6.11 : Registre à décalage entrée parallèle et sortie série.

Tableau 6.5 : table de vérité d'un registre à décalage entrée parallèle et sortie série à 4 bits.

Chargement	Clk	Q_0	Q_1	Q_2	Q_3
1	↑	E_0	E_1	E_2	E_3
0	↑	0	E_0	E_1	E_2
0	↑	0	0	E_0	E_1
0	↑	0	0	0	E_0

2.3 Registres entrée série - sortie série

Ce registre possède une seule entrée E, une seule sortie S et n bascules (Fig. 6.12). Les données binaires d'entrée sont introduites bit après bit. Elles sont également disponibles les unes après les autres au rythme de l'horloge en sortie. Ce type de registre est utilisé pour effectuer des décalages.

Exemple : registre à décalage 4 bits

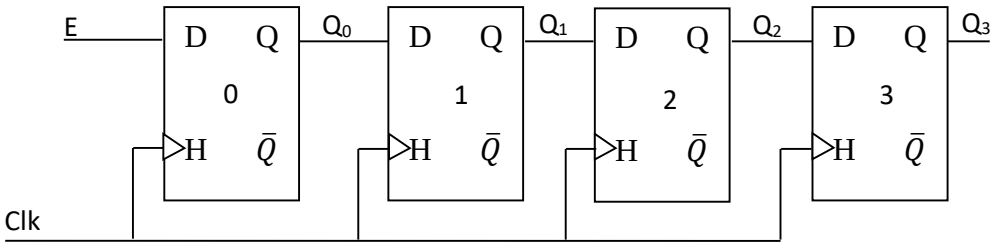


Figure 6.12 : Registre à décalage entrée série et sortie série.

Les bascules D sont les plus utilisées dans ce type de registre car elles demandent moins de connexions.

Tableau 6.6 : table de vérité d'un registre à décalage entrée série et sortie série à 4 bits.

E	Clk	Q_0	Q_1	Q_2	Q_3
1	↑	Ø	Ø	Ø	Ø
1	↑	1	Ø	Ø	Ø
0	↑	1	1	Ø	Ø
1	↑	0	1	1	Ø
/	↑	1	0	1	1

2.4 Registres entrée parallèle - sortie parallèle

Toutes les entrées (E_1, E_2, E_3, E_4), sont introduites en même temps dans le registre (Fig. 6.12). Toutes les sorties (S_0, S_1, S_2, S_3), sont disponibles au même instant. On considère un registre de quatre bits.

Exemple 1 : registre à décalage 4 bits

Les bascules utilisées dans cet exemple sont des bascules D, mais un registre peut également être réalisé à partir de bascules JK.

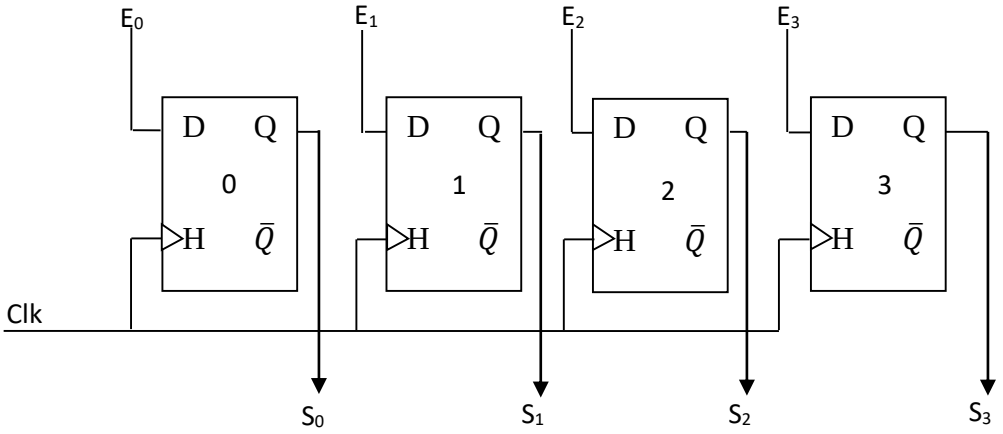


Figure 6.12 : Registre entrée parallèle et sortie parallèle.

Exemple 2: Application d'un registre entrée parallèle - sortie parallèle

Soit le registre de la figure 6.13, à Entrées (Ii) parallèles et à sorties (Ai) parallèles.

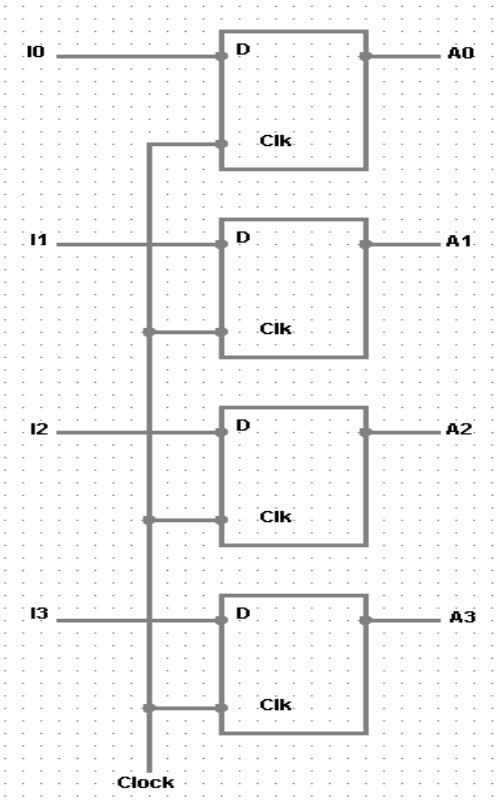


Figure 6.13 : Registre entrée parallèle et sortie parallèle

On désire ajouter une commande (Load) qui lorsque :

1) Load = 0 ——>le registre mémorise l'ancienne sortie (Ai) ——> Ai (t+1) = Ai
2) Load = 1——> les nouvelles entrées (Ii) sont chargées ——> Ai (t+1) = Ii

Tableau 6.7 : Table de fonctionnement de la commande Load

L	Fonction	$A_i(t+1)$	D
0	No Load	A_i	A_i
1	Load	I_i	I_i

Fonction logique correspondant le tableau 6.7 est la suivante :

$$D_i = \bar{L}.A_i + L.I_i$$

En utilisant cette fonction logique on obtient la figure 6.14.

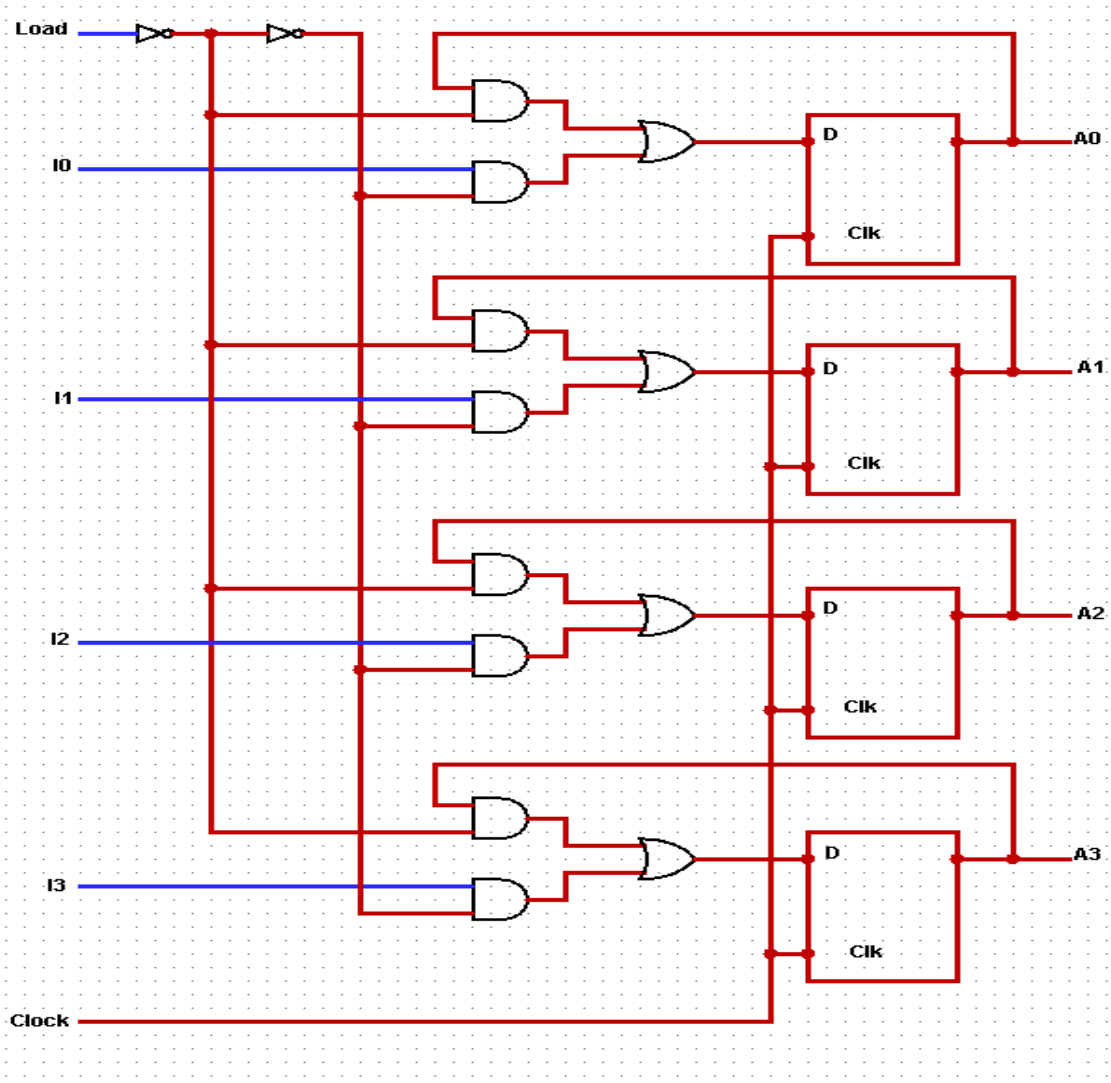


Figure 6.14 : Circuit séquentiel d'un registre 4 bits entrée parallèle - sortie parallèle

2.5 Registre à décalage

Ce type de registre est principalement utilisé comme mémoire d'information dynamique, la fonction de décalage consiste de faire glisser l'information de chaque cellule élémentaire dans une autre cellule élémentaire adjacente.

Soit un mot de huit éléments binaires (constitué de huit bascules), schématisé par :

D7	D6	D5	D4	D3	D2	D1	D0

On peut effectuer différents types de décalages sur ce mot.

2.5.1 Décalage à droite

Tous les éléments binaires sont décalés d'un rang vers la droite ; il apparaît un 0 (ou un 1) sur l'élément binaire de poids fort (bit le plus à gauche). L'élément binaire de poids faible (le bit le plus à droite) est perdu (Fig.6.15)**.**

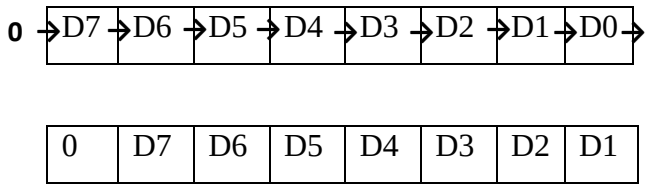


Figure 6.15 : décalage à droite des éléments binaires.

Si la valeur 0 est entrée sur l'élément binaire de poids fort, on obtient une division par deux du nombre initial.

2.5.2 Décalage à gauche

Tous les éléments binaires sont décalés d'un rang vers la gauche ; il apparaît un 0 (ou un 1) sur l'élément binaire de poids faible (bit le plus à droite). L'élément binaire de poids fort (le bit le plus à droite) est perdu (Fig.6.16)**.**

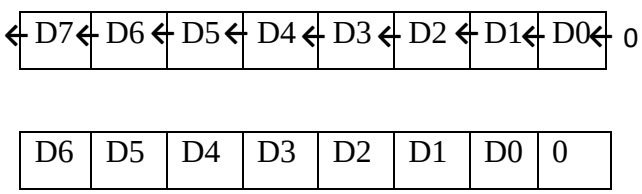


Figure 6.16 : décalage à gauche des éléments binaires.

Si la valeur 0 est entrée sur l'élément binaire de poids faible, on obtient une multiplication par deux du nombre initial.

2.6 Registre à rotation

2.6.1 Rotation à droite

Tous les éléments binaires sont décalés vers la droite et le bit de poids fort prend la valeur du bit de poids faible (Fig.6.17)**.**

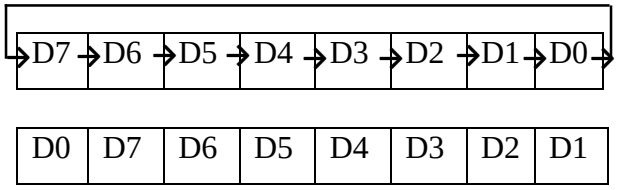


Figure 6.17 : rotation à droite des éléments binaires.

2.6.2 Rotation à gauche

Tous les éléments binaires sont décalés vers la gauche et le bit de poids faible prend la valeur du bit de poids fort (Fig.6.18).

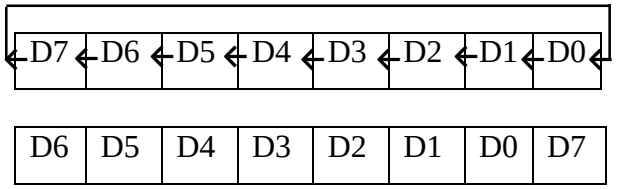


Figure 6.18 : rotation à gauche de huit éléments binaires.

Références :

A. BESSAID. « ELECTRONIQUE NUMERIQUE : LOGIQUE COMBINATOIRE, EXERCICES ET PROBLEMES RESOLUS (TOME1) ». *OFFICE DES PUBLICATIONS UNIVERSITAIRES*. 2015.

A. BESSAID. « ELECTRONIQUE NUMERIQUE : LOGIQUE SEQUENTIELLE, EXERCICES ET PROBLEMES RESOLUS (TOME2) ». *OFFICE DES PUBLICATIONS UNIVERSITAIRES*. 2015.

A- K. MAINI. « DIGITAL ELECTRONICS: PRINCIPLES, DEVICES AND APPLICATIONS ». *JOHN WILEY & SONS, LTD*. 2007. DOI: HTTPS://WWW.SHAHUCOLLEGELATUR.ORG.IN/DEPARTMENT/ STUDYMATERIAL/ SCI/IT/ BCA/ FY/DIGIELEC.PDF

C. ALEXANDRE. « CIRCUITS NUMERIQUES : 1ERE PARTIE ». *CONSERVATOIRE NATIONAL DES ARTS ET METIERS*. 2004. DOI: HTTPS://EASYTP.CNAM.FR/ALEXANDRE/INDEX_FICHIERS/SUPPORT/ ELE015 COUR _ VOL1.PDF

C. DIOU. « COURS D'ELECTRONIQUE NUMERIQUE ». *UNIVERSITE PAUL VERLAINE–METZ*. 2009. DOI : HTTPS://DOCPLAYER.FR/3943973-COURS-D-ELECTRONIQUE -NUMERIQUE.HTML.

C. DOUILLARD., A. THEPAUT. « ELECTRONIQUE NUMERIQUE ». *TELECOM BRETAGNE*. 2008. DOI : HTTPS://DOCPLAYER.FR/8037265-ELP304-203-ELECTRONIQUE-NUMERIQUE.HTML.

D. ETIEMBLE. « COMPOSANTS LOGIQUES ET OPERATEURS MATERIELS ». *UNIVERSITE PARIS SUD*. 2004. DOI : HTTPS://WWW.LRI.FR/~DE/POLY-CLM -MATERIEL.PDF

D. MANGE. « ANALYSE ET SYNTHESE DES SYSTEMES LOGIQUES ». *PRESSE POLYTECHNIQUES ET UNIVERSITAIRES ROMANDES*. 1995.

E. CARIOU. « CIRCUITS LOGIQUES ». *UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR*. DOI : HTTPS://ECARIOU.PERSO.UNIV-PAU.FR/COURS/ARCHI/COURS-3-CIRCUITS-LOGIQUES.PDF

E. MESSERLI., Y. MEYER. « ELECTRONIQUE NUMERIQUE 1ER TOME SYSTEMES COMBINATOIRES ». *HAUTE ECOLE SPECIALISEE DE SUISSE OCCIDENTALE*. 2010. DOI : HTTP://WWW.GECIF.NET/ARTICLES/GENIE_ELECTRIQUE/COURS/LIVRE_ELECTRONIQUE_NUMERIQUE.PDF

E. MESSERLI., Y. MEYER. « ELECTRONIQUE NUMERIQUE 2EME TOME SYSTEMES SEQUENTIELS ». *HAUTE ECOLE SPECIALISEE DE SUISSE OCCIDENTALE*. 2004. DOI : HTTP://REDS.HEIG-VD.CH/SHARE/COURS/MANUEL/ELECNUM_T2.PDF

G. CORMIER. « CIRCUITS SEQUENTIELS ». UNIVERSITE DE MONCTON. 2015. DOI: HTTP://WWW8.UMONCTON.CA/UMCMCORMIER_GABRIEL/CIRCUITSLOGIQUES/GELE2442_CHAPITRE6.PDF

J. BOUQUET., P. MAYE. « ÉLECTRONIQUE NUMERIQUE EN 26 FICHES ». *DUNOD, PARIS*. 2010.

J. HARVEY., M. SAWAN. « SYSTEMES LOGIQUE I ». *ECOLE POLYTECHNIQUE DE MONTREAL*. 1999.

J. WHITESITT. « BOOLEAN ALGEBRA AND ITS APPLICATIONS ». *DOVER PUBLICATION*. 2012.

M. BILLAUD. « CIRCUITS LOGIQUES ET ALGEBRE DE BOOLE ». *UNIVERSITE BORDEAUX*. 2015.

M. SIADAT., C. DIOU. « COURS D'ELECTRONIQUE NUMERIQUE ». 2019. DOI : HTTPS://F2SCHOOL.COM/ WP-CONTENT/UPLOADS/2019/09/COURS_ELECTRONIQUE_NUMERIQUE-3.PDF

M. MANO, M. CILETTI. « DIGITAL DESIGN, 5TH EDITION ». *PEARSON EDUCATION*. 2012. DOI: HTTPS://WWW.PEARSONHIGHERED.COM/ASSETS/PREFACE/0/1/3/2/0132774208.PDF

N. SOUAG. « ELECTRONIQUE NUMERIQUE : COURS ET EXERCICES CORRIGES ». *OFFICE DES PUBLICATIONS UNIVERSITAIRES*. 2013.

R. FREY. « LECTURE NOTES FOR DIGITAL ELECTRONICS ». *UNIVERSITY OF OREGON EUGENE USA*. 2000. DOI : HTTPS://PAGES.UOREGON.EDU/RAYFREY/DIGITALNOTES.PDF

W. HAMMER. « SYSTEMES LOGIQUES ». *ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE*. 2001. DOI : HTTPS://LSPWWW.EPFL.CH/TEACHING/COURS-WH.PDF