République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de RELIZANE
Faculté des Sciences et de la Technologie
Département : Informatique

Polycopie

Matière : Structure Machine 2

Niveau : L1

Auteure : Dr. Naima Beladam

Année universitaire : 2022-2023

Sommaire

Liste de figures

Liste de Tableaux

Introduction générale

Introduction générale

Ce polycopié de cours à destination des étudiants de L1 a été préparé à l'université de RELIZANE. Il comporte la matière « Structure Machine 2 » établie selon le canevas officiel du ministère de l'éducation supérieure et de la recherche scientifique Algérien.

L'objectif principal de la matière est d'offrir aux étudiants une partie du savoir nécessaire pour comprendre le fonctionnement et la structure des ordinateurs.

Le polycopié est organisé en trois chapitres

- Conception des circuits logiques combinatoires.
- Conception des circuits logiques séquentiels.
- Caractéristiques de base des circuits intégrés.

Pour s'assurer que l'étudiant assimile bien les connaissances acquises dans le cours, des exercices corrigés en TD viendront consolider ces connaissances.

Chapitre 1 : La logique combinatoire

Introduction

Ce chapitre décrit l'étude théorique de quelques circuits combinatoires de base tels que : additionneurs, soustracteurs, comparateur, multiplexeurs, démultiplexeurs, codeurs, décodeurs et transcodeur. Ces circuits jouent un rôle fondamental dans la conception des différents blocs d'un ordinateur.

1. Définition

Un circuit est dit combinatoire si les sorties dépend uniquement des états des entrées

Figure 1.1: Schéma bloc d'un circuit combinatoire.

2. Analyse et Synthèse d'un circuit combinatoire

L'étude des circuits combinatoires, consiste en deux étapes :

- **Synthèse :** réaliser le circuit combinatoire à partir de l'énoncé décrivant les fonctions ou le rôle du circuit, en question.
- **Analyse :** déterminer le rôle du circuit combinatoire à partir de son logigramme.

2.1. Synthèse d'un circuit combinatoire

D'une façon générale, la démarche est la suivante:

1. Identifier les entrées et les sorties (IN / OUT) de la fonction.
2. Construire la table de vérité.
3. Identifier les fonctions de sortie à partir de la table de vérité.
4. Simplifier les fonctions de sortie.
5. Dessiner le schéma du circuit (logigramme).

Exemple 1.1: On veut réaliser un circuit capable de décrire sur un ensemble de 3 bits s'il y a une majorité de 1 ou une majorité de 0.

- **Les entrées :** a, b et c
- **La sortie:** M

La table de vérité est présentée en Tableau 1.1

a	b	c	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tableau 1.1: Table de vérité.

L'expression logique de sortie M est déduite directement à partir de la table de vérité:

$M = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$

Le tableau de karnaugh de la fonction M est présenté en Tableau 1.2 :

a \ bc	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Tableau 1.2: Table de Karnaugh.

La fonction M après la simplification: $M = ab + ac + bc$

D'après la fonction M le logigramme est :

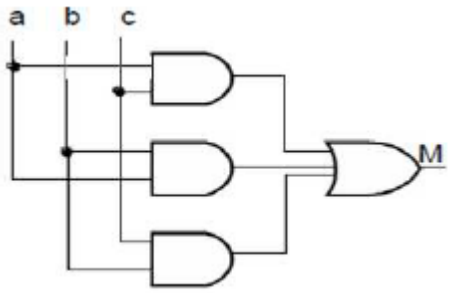


Figure 1.2: Logigramme.

2.2. Analyse d'un circuit combinatoire

Pour analyser un circuit combinatoire, on suit les étapes suivantes :

1. Déterminer les expressions logiques des variables de sortie.
2. Dresser la table de vérité du circuit et la traduire par un énoncé décrivant le rôle du circuit.

Exemple 1.2:

Quel est le rôle de la fonction *f* représentée par le logigramme de la figure 1.3 suivante :

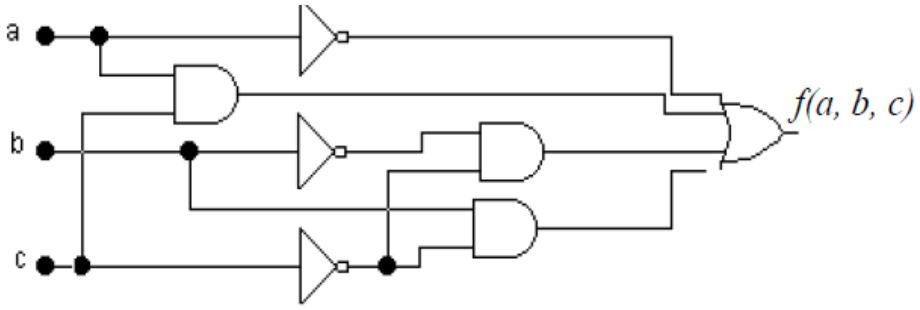


Figure 1.3: Logigramme**.**

● **L'expression logique :**

$$f(a,b,c) = \overline{a} + a.c + \overline{b}.\overline{c} + b.\overline{c}\ .$$

● **La table de vérité :**

a	*b*	*c*	*f(a, b, c)*
0	*0*	*0*	*1*
0	*0*	*1*	*1*
0	*1*	*0*	*1*
0	*1*	*1*	*1*
1	*0*	*0*	*1*
1	*0*	*1*	*1*
1	*1*	*0*	*1*
1	*1*	*1*	*1*

Tableau 1.3: Table de vérité.

● **Le rôle du circuit:** produire la constante 1.

3. Etudes de quelques circuits combinatoires usuels

3.1. Addition

3.1.1. Demi-additionneur

Il s'agit de réaliser un circuit permettant d'additionner 2 nombres à 1 seul bit, et d'obtenir comme sortie le résultat de l'addition S et la retenue R (Figure 1.4).

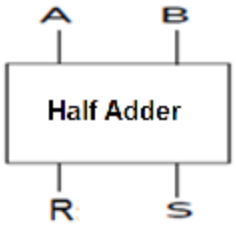


Figure 1.4: Schéma bloc d'un demi- additionneur.

Donc la table de vérité est :

A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tableau 1.4: Table de vérité de demi-additionneur.

Les expressions logiques des sorties sont déduites directement à partir de la table de vérité:

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

$R = AB$

D'après ces équations, un demi-additionneur est alors représenté par le logigramme suivant:

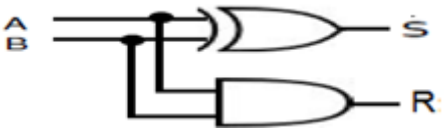


Figure 1.5: Logigramme de demi-additionneur.

3.1.2. Additionneur complet

Il s'agit d'additionner deux nombres A et B en tenant compte d'une retenue antérieure Rn. Il présente deux sorties S_n et R_{n+1} (Figure 1.6).

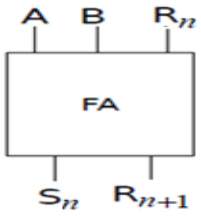


Figure 1.6 : Schéma fonctionnel de l'additionneur complet.

● **La table de vérité :**

A	B	R_n	S_n	R_{n+1}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Tableau 1.5: Table de vérité de l'additionneur complet.

● **Les fonctions de sortie :**

$S_n = A\overline{B}\overline{R_n} + \bar{A}B\overline{R_n} + \bar{A}\bar{B}R_n + ABR_n$

$R_{n+1} = AB\overline{R_n} + \bar{A}BR_n + ABR_n + A\bar{B}R_n$

● **Tableau de Karnaugh pour simplifier R_{n+1}**

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

Tableau 1.6: Table de Karnaugh pour simplifier R_{n+1}.

Donc : $R_{n+1} = AB + AR_n + BR_n$

Ensuite pour simplifier le S_n

$\overline{R_{n+1}} = \bar{A}\bar{B} + \bar{A}\,\overline{R_n} + \bar{B}\overline{R_n}$

$A\overline{R_{n+1}} = A\bar{B}\overline{R_n}$

$B\overline{R_{n+1}} = \bar{A}B\overline{R_n}$

$R\overline{R_{n+1}} = A\bar{B}R_n$

$S_n = A\overline{B}\,\overline{R_n} + \bar{A}B\overline{R_n} + \bar{A}\bar{B}R_n + ABR_n$

$S_n = (A + B + R_n)R_{n+1} + ABR_n$

● **Le logigramme**

D'après ces équations, un additionneur complet est alors représenté par le logigramme suivant:

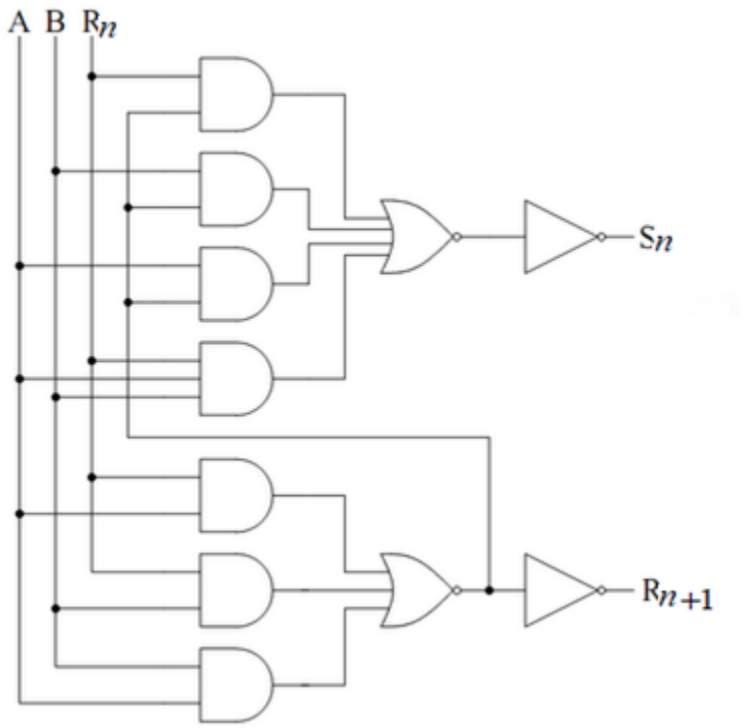


Figure 1.7: Logigramme de l'additionneur complet.

On peut construire un additionneur complet à l'aide de deux demi-additionneurs (Figure 1.8):

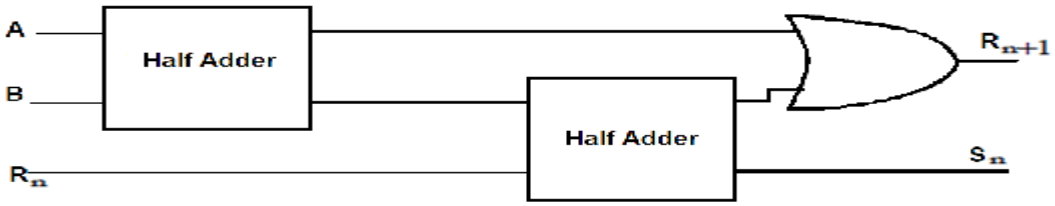


Figure 1.8: Schéma d'Additionneur Complet à l'aide de deux demi-additionneurs.

3.1.3. Additionneur à n bits

➢l'addition de nombres comportant plusieurs bits peut se faire en série (bit après bit)

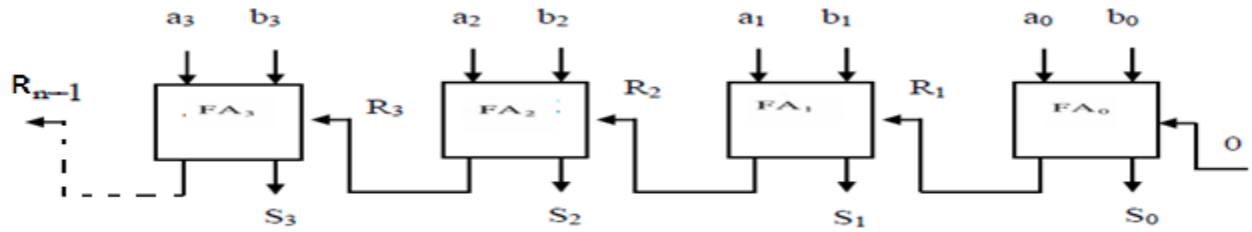


Figure 1.9: Additionneur à n bits.

➢ou (presque) en parallèle (tous les bits simultanément)

Si le dernier bit de retenue est égal à 1, il est signalé par un indicateur de carry mémorisé par un 1 noté C (Carry) dans un registre appelé le Processor Status Word (PSW) ou registre d'etat.

Si le résultat est aussi en dehors de l'intervalle des n bits, ce dépassement de capacité est aussi mémorisé dans le registre d'état du PSW par l'intermédiaire d'un bit 1 noté O (Overflow).

3.2. Soustraction

3.2.1. Demi-soustracteur

C'est un circuit combinatoire qui permet d'effectuer la différence A-B de deux nombres A et B, possède deux sorties: la différence D et un empreint E (Figure 1.10).

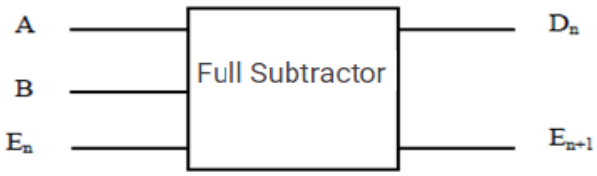


Figure 1.10: Schéma bloc de demi-soustracteur.

Il admet comme table de vérité

A	B	D	E
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tableau 1.7: Table de vérité de demi-soustracteur.

● **Les fonctions de sortie :**

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

$E = \bar{A}B$

D'après ces équations, un demi-soustracteur est alors représenté par le logigramme suivant:

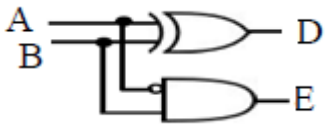


Figure 1.11: Logigramme de demi-soustracteur.

3.2.2. Soustracteur complet

C'est un circuit combinatoire qui permet d'effectuer la différence A-B de deux nombres A et B en tenant compte d'un empreint antérieur En. Il présente deux sorties D_n et E_{n+1}.

A	B	E_n	D_n	E_{n+1}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	1
1	1	0	0	0
0	0	1	1	1
1	0	1	0	0
0	1	1	0	1
1	1	1	1	1

Tableau 1.8: Table de vérité de soustracteur complet.

● **Les fonctions de sortie :**

$$D_n = A\bar{B}\,\overline{E_n} + \bar{A}B\overline{E_n}\bar{A}\bar{B}E_n + ABE_n$$

$$= \overline{E_n}(A\bar{B} + \bar{A}B) + E_n(\bar{A}\bar{B} + AB)$$

$$= \overline{E_n}(A \oplus B) + E_n(\overline{A \oplus B})$$

Ainsi :

$$D_n = E_n \oplus (A \oplus B)$$

$$E_{n+1} = \bar{A}B\overline{E_n} + \bar{A}\bar{B}E_n + \bar{A}BE_n + ABE_n$$

$$= \bar{A}B(\overline{E_n} + E_n) + (\bar{A}\bar{B} + AB)E_n$$

Ainsi : $E_{n+1} = \bar{A}B + \overline{(A \oplus B)}E_n$

D'après ces équations, un soustracteur complet est alors représenté par le logigramme suivant (**Figure 1.12**):

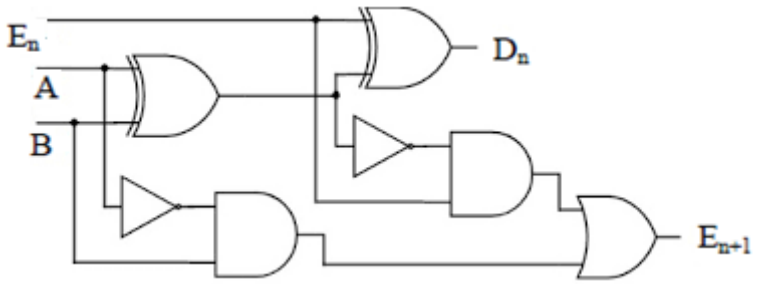


Figure 1.12: Logigramme de soustracteur complet.

Le schéma fonctionnel d'un soustracteur complet à l'aide de deux demi-soustracteurs est alors :

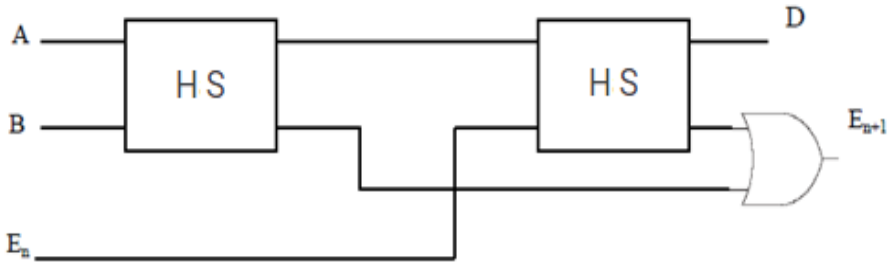


Figure 1.13: Logigramme de soustracteur complet à l'aide de deux demi-soustracteurs.

3.3. Additionneur-soustracteur

Pour effectuer des soustractions on utilise les propriétés du complément à 2. Soit $\bar{Y}$ le nombre binaire obtenu en remplaçant les 1 de Y par un 0 et les 0 par un 1. Effectuer $D = X - Y$ est équivalent à $D = X + \bar{Y} + 1$ en complément à 2.

Soit O un signal de contrôle valant 0 si on veut faire une addition, et 1 si on veut faire une soustraction. On utilise ce signal O comme retenue du bit de poids faible de l'additionneur.

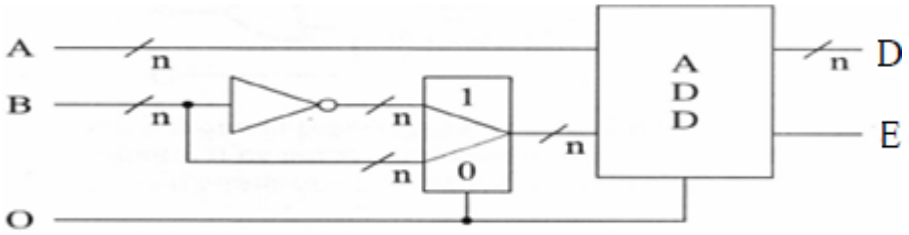


Figure 1.14: Additionneur-soustracteur.

3.4. Comparateur

C'est un circuit permettant de comparer deux nombres binaires A et B de n bits chacun *A=An-1 An-2 …A1 A0* et *B=Bn-1 Bn-2…B1 B0.* en indiquant sur ses sorties S, I ou E si A>B, A<B, A=B

Exemple 1.3: comparateur de deux nombre binaire A et B d'un seul bit

● **La table de vérité :**

A	B	S= (A >B)	I= (A<B)	E= (A=B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Tableau 1.9: Table de vérité de comparateur à un seul bit.

● **Les fonction de sorties :**

- ✓ $S = A\bar{B}$
- ✓ $I = \bar{A}B$
- ✓ $E = AB + \bar{A}\bar{B}$

  $= \overline{S + I}$

● **Le logigramme:**

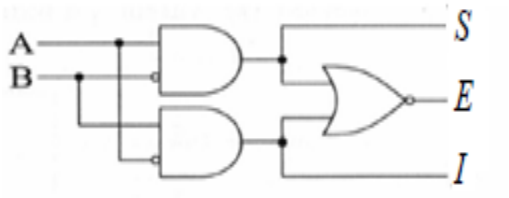


Figure 1.15: Logigramme de comparateur à un seul bit.

Exemple 1.4 : Comparateur de deux nombres binaires de deux bits chacun *A=A1 A0* et *B=B1 B0* (Figure 1.16).

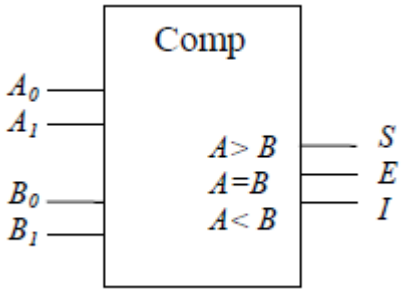


Figure 1.16: Schéma fonctionnel d'un comparateur à 2 bits.

● **Table de Karnaugh générale**

$B_1 B_0$ \ $A_1 A_0$	00	01	11	10
00	*E*	*S*	*S*	*S*
01	*I*	*E*	*S*	*S*
11	*I*	*I*	*E*	*I*
10	*I*	*I*	*S*	*E*

Tableau 1.10: Table Karnaugh générale de comparateur à deux bits.

●**Table de Karnaugh de *S***

$B_1 B_0$ \ $A_1 A_0$	00	01	11	10
00		1	1	1
01			1	1
11				
10			1	

Tableau 1.11: Table Karnaugh de la fonction S.

$S = A_1\overline{B_1} + A_1A_0\overline{B_0} + A_0\overline{B_1}\ \ \overline{B_0}$

●**Table de Karnaugh de *E***

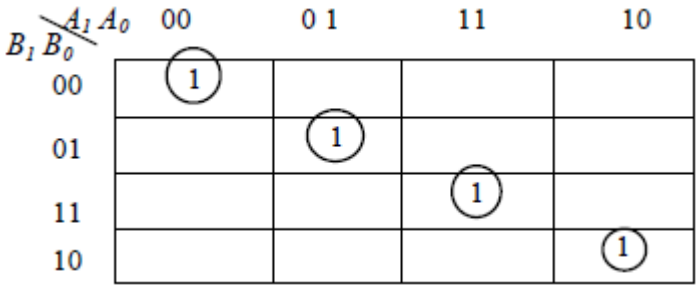


Tableau 1.12: Table Karnaugh de la fonction E.

$E = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1A_0B_1\,B_0 + A_1\overline{A_0}\,B_1\,\overline{B_0}$

$= \overline{(A_1 \oplus B_1)}\,\overline{(A_0 \oplus B_0)}$

●**Table de Karnaugh de *I***

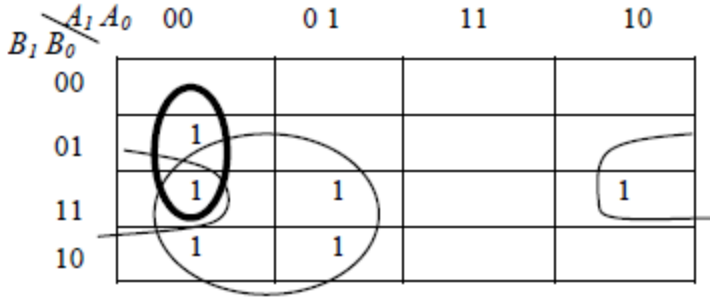


Tableau 1.13: Table Karnaugh de la fonction I.

$I = \overline{A_1}\,B_1 + \overline{A_1}\,\overline{A_0}\,B_0 + \overline{A_0}B_1B_0$

3.5. Multiplexeur

Un multiplexeur (MUX) est un circuit combinatoire sélecteur qui possède 2^n entrées d'information, n entrées de sélection des données (entrées de commande) et une seule sortie. Son rôle consiste à sélectionner, à l'aide de signaux de commande, une des entrées et à la lier à la sortie (). Les Mux sont aussi connus sous le nom de sélecteurs de données.

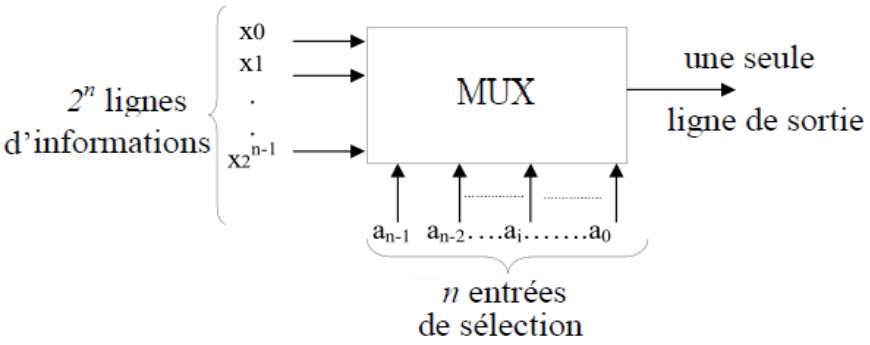


Figure 1.17: Schéma fonctionnel de multiplexeur.

Exemple 1.5: MUX à 4 entrées

Soit la table de vérité suivante :

a	b	Z
0	0	K_0
0	1	K_1
1	0	K_2
1	1	K_3

Tableau 1.14: Table vérité d'un MUX à 4 entrées.

La sortie des données est égale à K_0 seulement si $a = 0$ et $b = 0$: $Z = K_0\bar{a}\bar{b}$.

La sortie des données est égale à K_1 seulement si $a = 0$ et $b = 1$: $Z = K_0\bar{a}b$.

La sortie des données est égale à K_2 seulement si $a = 1$ et $b = 0$: $Z = K_0a\bar{b}$.

La sortie des données est égale à K_3 seulement si $a = 1$ et $b = 1$: $Z = K_0ab$.

D'où la fonction de sortie : $Z = K_0\bar{a}\bar{b} + K_0\bar{a}b + K_0a\bar{b} + K_0ab$

Soit, le logigramme correspondant est :

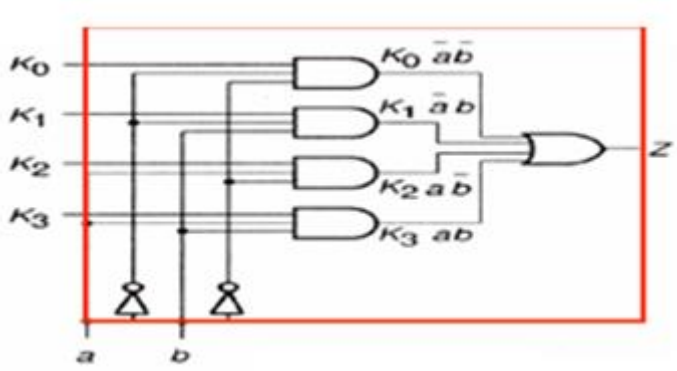


Figure 1.18: Logigramme d'un MUX à 4 entrées.

Et le schéma fonctionnel

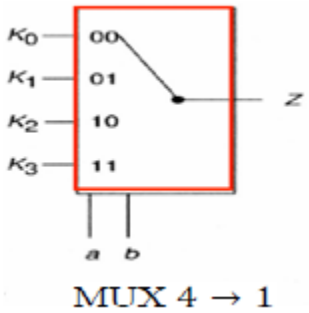


Figure 1.19: Schéma fonctionnel d'un multiplexeur à 4 entrées.

Générateur de fonctions logiques : Une application utile du multiplexeur est de générer des fonctions logiques combinatoires sous forme de sommes de produits. Il permet donc une transcription directe de la table de vérité.

Exemple 1.6: Soit la fonction logique *f* définie par sa table de Karnaugh suivante

c \ *ab*	00	01	11	10
0	*0*	*0*	*1*	*1*
1	*0*	*1*	*1*	*0*

Tableau 1.15: Table Karnaugh de la fonction *f*.

1. Réalisation de *f* par un MUX8---->1.

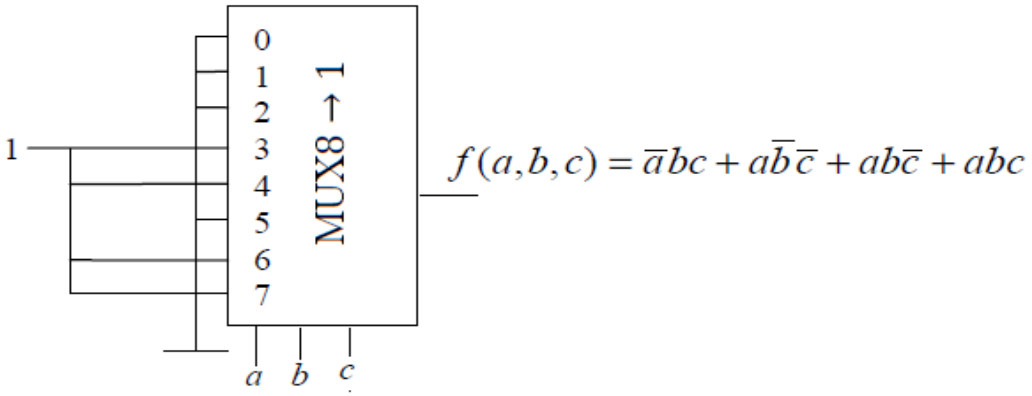


Figure 1.20 : Schéma fonctionnel de *f* réalisée par MUX8---->1.

2. Réalisation de *f* par un MUX4---->1.

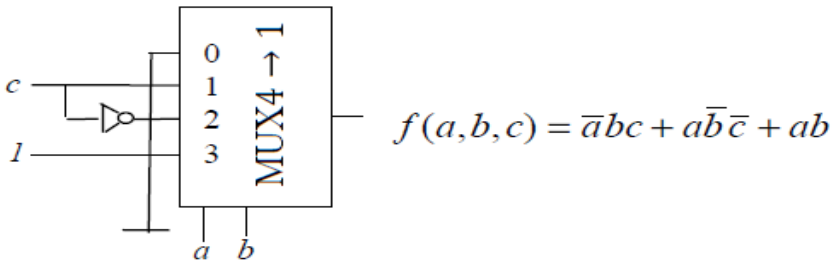


Figure 1.21 : Schéma fonctionnel de *f* réalisée par MUX 4---->1.

3.6. Démultiplexeur

Le démultiplexeur (démux) fonctionne de façon inverse à celle du multiplexeur. Donc c'est un circuit logique ayant *n* entrées de sélection et une seule entrée de donnée à acheminer vers l'une des 2^n sorties désirées par les adresses indiquées par les entrées de sélection. Les autres sorties donnent alors la constante 0. On l'appelle aussi distributeur de données.
Les démultiplexeurs sont surtout utilisés dans les conversions série - parallèle.

RQ : Dans l'application pratique, le démux est peu utilisé dans cette forme. On lui préfère un composant dérivé: le décodeur.

Exemple 1.7: DEMUX à 4 sorties

Faisant la synthèse d'un DEMUX à 4 sorties : S_0 , S_1 , S_2 , S_3 , une entrée *E* et deux entrées de sélection : *a* et *b*

●**Table de vérité :**

a	b	S_0	S_1	S_2	S_3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

Tableau 1.16: Table de vérité d'un DEMUX à 4 sorties.

• **Logigramme**

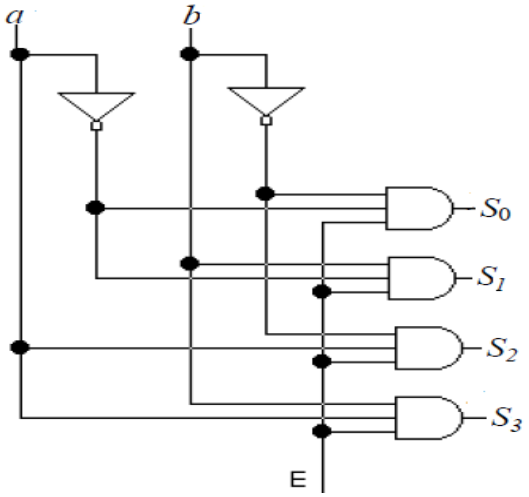


Figure 1.22: Logigramme d'un DEMUX à 4 sorties.

La ligne d'entrée est connectée à toutes les portes ET. Les deux lignes de sélection de données valident une seule porte à la fois. Les données de la ligne d'entrée traversent la porte sélectionnée jusqu'à sa ligne de sortie.

3.7. Codeur

Le **codeur** (ou encodeur) est un circuit logique qui possède 2^n voies entrées, dont une seule est activée et N voies de sorties. Il fournit en sortie le code binaire correspondant. Donc, le codage de 2^n combinaisons en entrée nécessite n bits en sortie.

Exemple 1.8: codeur 4 voies d'entrées et 2 bits de sortie

● **Schéma fonctionnel :**

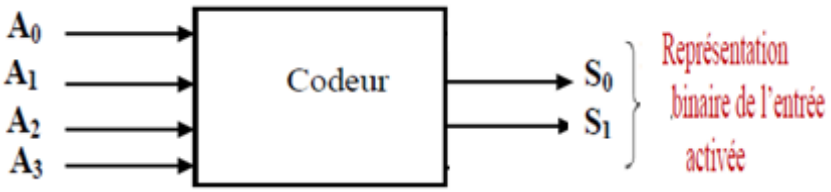


Figure 1.23 : Schéma fonctionnel d'un codeur 4 voies d'entrées et 2 bits de sortie.

- **Table de vérité :**

A_3	A_2	A_1	A_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Tableau 1.17: Table de vérité d'un codeur 4 voies d'entrées et 2 bits de sortie.

- **Equation des sorties :**

S1=1 si (A2=1) ou (A3=1) ; S1=A2+A3 S0=1 si (A1=1) ou (A3=1) ; S0=A1+A3

- **Logigramme :**

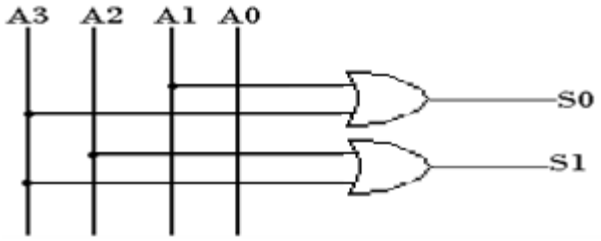


Figure 1.24: Logigramme d'un codeur 4 voies d'entrées et 2 bits de sortie.

Exemple 1.9: conversion décimal-BCD

- Table de vérité d'un codeur BCD (Binary Code Decimal) d'une conversion décimal-BCD

E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Tableau 1.18: Table de vérité d'un convertisseur décimal-BCD.

● **Les expressions logiques**

$A_0 = E_1 + E_3 + E_5 + E_7 + E_9$

$A_1 = E_2 + E_3 + E_6 + E_7$

$A_2 = E_4 + E_5 + E_6 + E_7$

$A_3 = E_8 + E_9$

● **Logigramme**

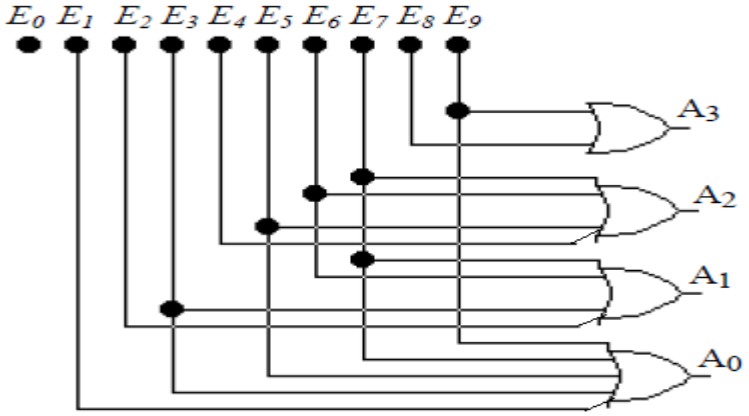


Figure 1.25: Logigramme d'un convertisseur décimal-BCD.

Si nous activons simultanément les entrées E1 et E2 du codeur ci-dessus, les sorties A1A0 présente le nombre 11 qui ne correspond pas au code de l'une ou de l'autre des entrées activés. C'est plutôt le code qui représente l'activation de E3.

Pour résoudre ce problème on utilise un codeur de priorité qui choisit le plus grand nombre lorsque plusieurs entrées sont activées à la fois.
Exemple, lorsqu' E_1 et E_2 sont activées simultanément A1A0 sera égale à 10 qui représentent l'activation de E_2

3.8. Décodeur

Le décodeur réalise la fonction inverse du codeur. Il permet d'identifier quelle combinaison est active. Le décodeur est un circuit logique comportant n entrée, 2^n combinaisons et M sorties tel que $M \leq 2^n$. Lorsque le signal de validation est actif, seule la sortie dont le numéro correspond à la valeur binaire affichée sur l'entrée est active. Toutes les autres sont inactives.

Exemple 1.10: Décodeur 2 entrée et 4 sorties

● **Table de vérité**

a	b	S_0	S_1	S_2	S_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Tableau 1.19: Table de vérité d'un décodeur 2 entrée et 4 sorties.

● **Expressions logiques**

$$\boldsymbol{S_0 = \bar{a}\bar{b}}$$

$$\boldsymbol{S_1 = \bar{a}b}$$

$$\boldsymbol{S_2 = a\bar{b}}$$

$$\boldsymbol{S_3 = ab}$$

● **Logigramme**

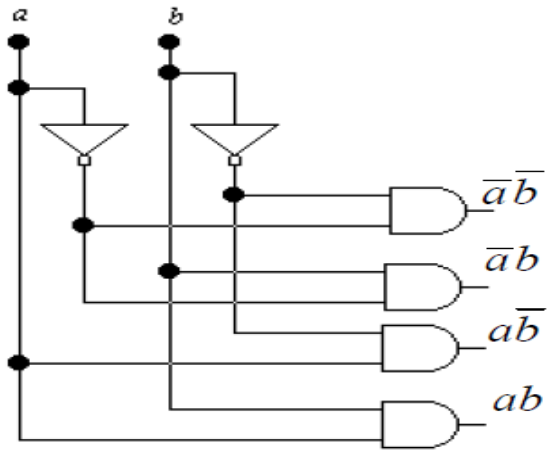


Figure 1.26: Logigramme d'un décodeur 2 entrée et 4 sorties.

Remarque: Certains n'utilisent pas toute la gamme de 2^n combinaisons d'entrées possibles. C'est le cas du décodeur BCD décimal qui a 4 bits d'entrée et 10 sorties donc un seule est actif dans chacune des 10 représentations du BCD.

3.9. Transcodeur

C'est un circuit combinatoire qui permet de transformer un code X (sur n bits) en entrée en un code Y (sur m bits) en sortie. Le lecteur de code barre est un transcodeur. Les deux plus

importantes applications des transcodeurs sont : la conversion de code et l'affichage par segment.

● **Schéma fonctionnel**

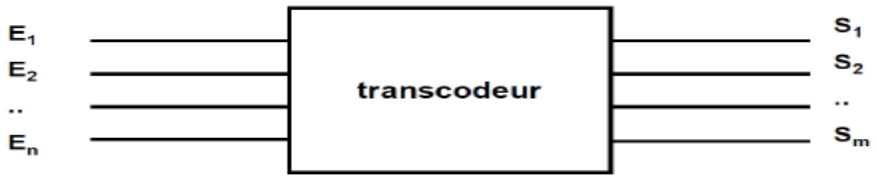


Figure 1.27: Schéma fonctionnel de transcodeur.

Exemple 1.11: Transcodeur BCD -> BCD+3

● **Table de vérité :**

A	B	C	D	X	Y	Z	T
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

Tableau 1.20: Table de vérité d'un transcodeur BCD -> BCD+3.

● **Expressions logiques**

$$X = A + BC + BD$$

$$Y = \overline{B}C + \overline{B}D + B\overline{CD}$$

$$Z = CD + \overline{CD} = \overline{C \oplus D}$$

$$T = \overline{D}$$

Chapitre 2 : La logique séquentielle

Introduction

- Dans les circuits combinatoires, les signaux de sortie ne dépendent que des signaux d'entrée présents au même instant.

- Dans les circuits séquentiels, il y a de la rétroaction : les signaux de sortie ne dépendant pas uniquement des entrées, mais aussi de leur séquence. Le circuit se rappelle des entrées et des états antérieurs. Cela est réalisé, en général, par un bouclage (en anglais : feed-back) (permanent ou périodique) de cette sortie à l'entrée du système, la sortie dépend indirectement d'elle-même.

1. Définition

Un circuit est dit séquentiel si le comportement des sorties dépend des valeurs assignées aux variables d'entrée et selon son histoire. De tels circuits contiennent d'éléments de mémoire – appelés : bascules- à côté d'une partie combinatoire. Ces éléments de mémoire ont pour rôle de conserver l'histoire du circuit

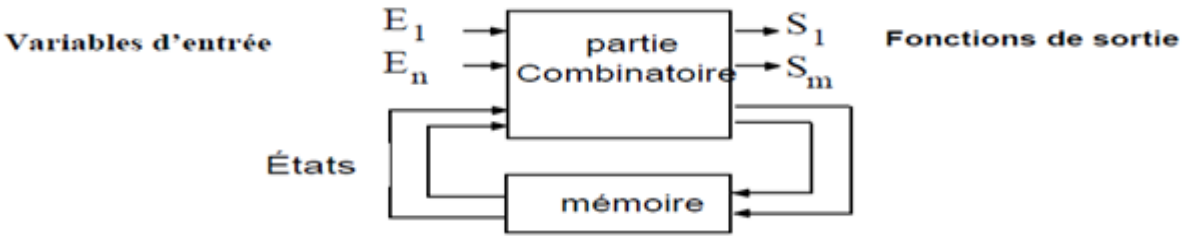


Figure 2.1: Schéma bloc d'un circuit séquentiel.

2. Les bascules

2.1. Définition

Une bascule est un circuit séquentiel élémentaire, capable de mémoriser une variable d'état et donc qui peut prendre 2 états stables (état 0 et état 1). Elle peut basculer d'un état vers l'autre sous l'effet d'une action externe. Chaque bascule possède deux sorties complémentaires Q et $\bar{Q}$. Les deux états stables d'une bascule sont:

- Etat I : $\boldsymbol{Q = 0\ ;\ \bar{Q} = 1}$ c'est l'état "0" ou état de repos
- Etat II : $\boldsymbol{Q = 1\ ;\ \bar{Q} = 0}$: c'est l'état "1" ou état de travail

On l'appelle ainsi « bascule bistable » car elle possède deux états stables. On distingue deux catégories principales de bascules :

• les *bascules asynchrones* que l'on nomme *verrous* (ou *latch* en anglais)
• et les *bascules synchrones* (dépendant d'un signal d'horloge) que l'on nomme simplement bascules (ou *flip-flop* en anglais).

2.2. Les bascules asynchrones

2.2.1. Définition

Une bascule est asynchrone quand ses sorties changent d'états dès qu'Il y a changements des états des entrées. On distingue 4 types de bascules : **RS, D, JK,** et **T**.

2.2.2. Bascule RS

● **Symbole**

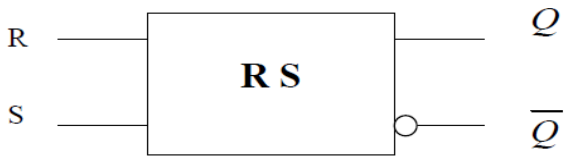


Figure 2.2: Schéma bloc de la bascule RS.

● **Explication**

Une impulsion sur **S (set)** → Mise à 1 de **Q** (marche)
Une impulsion sur **R (Reset)** → Mise à 0 de **Q** (Arrêt)

● **Table de fonctionnement**

Entrées			**Sorties**		**Mode de fonctionnement**
R	**S**	$\mathbf{Q_n}$	$\mathbf{Q_{n+1}}$	$\mathbf{\overline{Q}_{n+1}}$	
0	**0**	**0**	**0**	**1**	Etat précèdent
0	**0**	**1**	**1**	**0**	Etat précèdent
0	**1**	**0**	**1**	**0**	Enclenchement
0	**1**	**1**	**1**	**0**	Maintien à 1
1	**0**	**0**	**0**	**1**	Maintien à 0
1	**0**	**1**	**0**	**1**	Déclenchement
1	**1**	**0**	**-**	**-**	Interdit
1	**1**	**1**	**-**	**-**	interdit

Tableau 2.1: Table de fonctionnement de la bascule RS.

● **Table de karnaugh**

Q_n \ RS	00	01	11	10
0		1	X	
1	1	1	X	

Tableau 2.2: Table de Karnaugt de la bascule RS.

● **L'expression de sortie :** $Q_{n+1} = \bar{R}\, Q_n + S$

● **Logigramme**

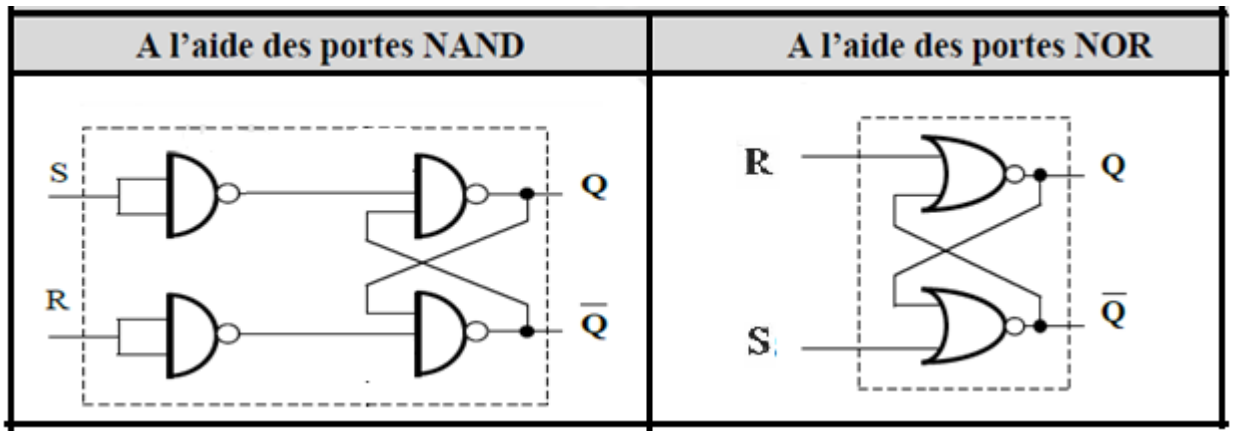


Figure 2.3: Logigramme de la bascule RS à l'aide des portes NAND et NOR.

NB : L'état R=S=1 est un état interdit puisqu'il nous donne le deux sorties complémentaires Q et $\bar{Q}$ au même état ce qui n'est pas logique.

2.2.3. Bascule D

● **Symbole**

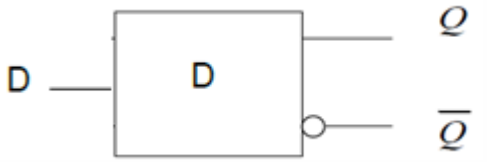


Figure 2.4: Schéma bloc de la bascule D.

● **Explication**

Un appui sur **D** → Mise à 1 de **Q** (marche)

Un relâchement de **D** → Mise à 0 de **Q** (Arrêt)

● **Table de fonctionnement**

Entrées		**Sorties**		**Mode de fonctionnement**
D	Q_n	Q_{n+1}	$\overline{Q}_{n+1}$	
0	**0**	**0**	**1**	Maintien à 0
0	**1**	**0**	**1**	Déclenchement
1	**0**	**1**	**0**	Enclenchement
1	**1**	**1**	**0**	Maintien à 1

Tableau 2.3: Table de fonctionnement de la bascule D.

● **Table de karnaugh**

Q_n \ D	0	1
0	0	1
1	0	1

Tableau 2.4: Table de Karnaugt de la bascule D.

● **L'expression de sortie :** $\boldsymbol{Q_{n+1} = D}$

● **Logigramme**

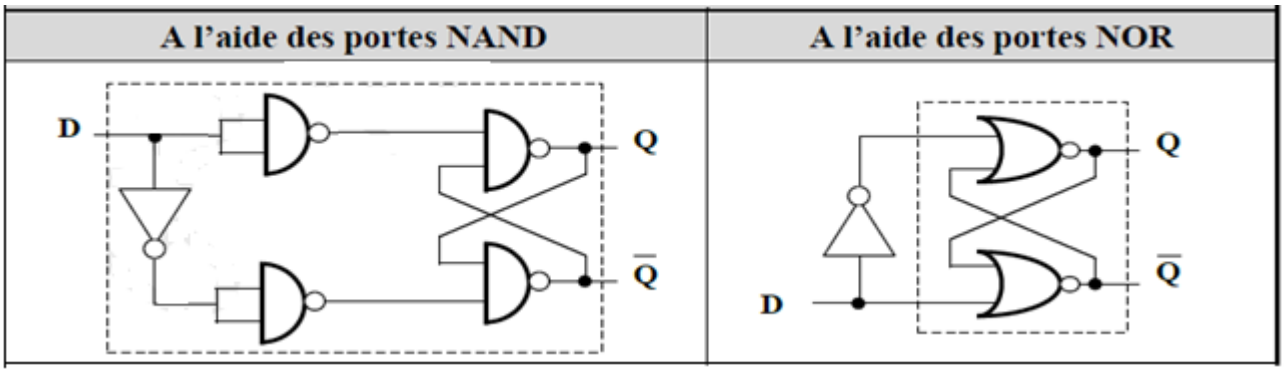


Figure 2.5: Logigramme de la bascule D à l'aide des portes NAND et NOR.

NB : En mettant $\boldsymbol{S = D}$ **et** $\boldsymbol{R = \overline{D}}$ dans l'équation de la bascule ***RS*** on aura

$$\boldsymbol{Q_{n+1} = DQ_n + D = D(1 + Q_n) = D}$$

● **Réalisation de la bascule D à partir d'une Bascule RS**

On obtient une bascule **D** en rajoutant un inverseur entre **S** et **R (Figure 2.6)**.

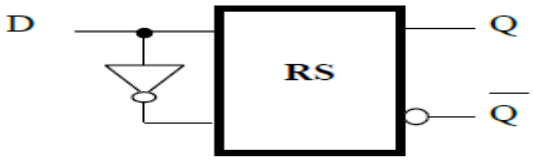


Figure 2.6: Schéma bloc de la bascule D réalisée à partir d'une Bascule RS.

2.2.4. Bascule JK

● **Symbole**

Figure 2.7: Schéma bloc de la bascule JK.

● **Explication**

La bascule JK fonctionne exactement comme la bascule RS, avec J correspondant à S et K correspondant à R, sauf que pour la combinaison J=K=1, ne donne pas lieu à une condition indéterminée, mais par contre la bascule passe à l'état opposé.

● **Table de fonctionnement**

Entrées			**Sorties**		**Mode de fonctionnement**
J	**K**	Q_n	Q_{n+1}	$\overline{Q}_{n+1}$	
0	**0**	**0**	**0**	**1**	Etat précèdent
0	**0**	**1**	**1**	**0**	Etat précèdent
0	**1**	**0**	**0**	**0**	Maintien à 0
0	**1**	**1**	**0**	**0**	Déclenchement
1	**0**	**0**	**1**	**1**	Enclenchement
1	**0**	**1**	**1**	**1**	Maintien à 1
1	**1**	**0**	**1**	**0**	Enclenchement
1	**1**	**1**	**0**	**1**	Déclenchement

Tableau 2.5: Table de fonctionnement de la bascule JK.

● **Table de karnaugh**

K \ J	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Tableau 2.6: Table de Karnaugt de la bascule JK.

● **L'expression de sortie :** $Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$

● **Réalisation de la bascule JK à partir d'une Bascule RS**

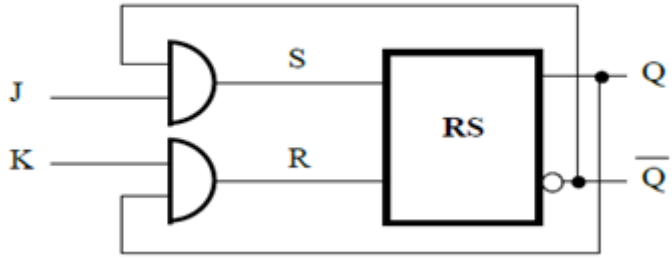


Figure 2.8: Schéma bloc de la bascule JK réalisée à partir d'une Bascule RS.

2.2.5. Bascule T

● **Symbole**

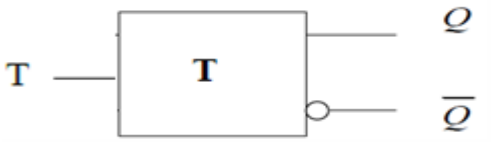


Figure 2.9: Schéma bloc de la bascule T.

● **Explication**

La bascule **T** est obtenue en reliant les entrées **J** et **K** d'une bascule **JK**.

● **Table de fonctionnement**

Entrées		**Sorties**		**Mode de fonctionnement**
T	**Q_n**	**Q_{n+1}**	**$\overline{Q}_{n+1}$**	
0	**0**	**0**	**1**	Maintien à 0
0	**1**	**1**	**0**	Maintien à 1
1	**0**	**1**	**0**	Enclenchement
1	**1**	**0**	**1**	Déclenchement

Tableau 2.7: Table de fonctionnement de la bascule T.

● **Table de karnaugh**

Q_n \ T	0	1
0	0	1
1	1	0

Tableau 2.8: Table de Karnaugt de la bascule T.

● **L'expression de sortie :** $Q_{n+1} = \overline{T}Q_n + T\overline{Q_n} = T \oplus Q_n$

● **Réalisation de la bascule T à partir d'une Bascule JK**

En remplaçant J et K par T dans l'équation d la bascule JK on aura

$$Q_{n+1} = TQ_n + T\overline{Q_n} = T \oplus \overline{Q_n}$$

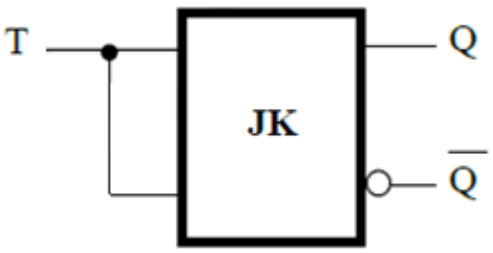


Figure 2.10: Schéma bloc de la bascule T réalisée à partir d'une Bascule JK.

2.3. Forçage des bascules

Certaines bascules sont équipées des entrées particulières :

- Entrée de remise à 1 : PRESET (RA1),
- Entrée de remise à 0 : RESET (RA0).

Exemple 2.1 : forçage de la bascule RS

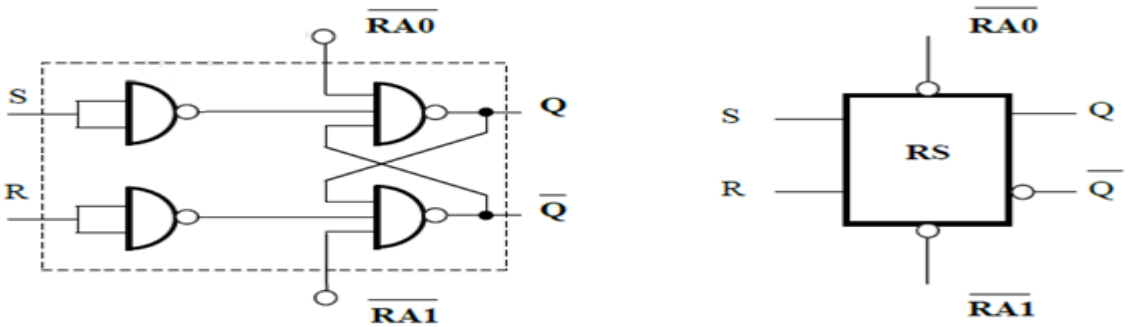


Figure 2.11: Forçage de la bascule RS.

On applique le même résonnement pour les bascules D, T et JK.

Entrées		Sorties		Mode de fonctionnement
Preset	Clear	Q_{n+1}	$\overline{Q}_{n+1}$	
0	**0**	Q_n	$\overline{Q}_n$	Mémorisation
0	**1**	**0**	**1**	Forçage à 1
1	**0**	**1**	**0**	Forçage à 0
1	**1**	**-**	**-**	Interdit

Tableau 2.9: Forçage de bascules.

2.4. Les bascules synchrones

Une bascule est synchrone quand ses sorties ne changent d'état que si un signal supplémentaire est appliquée sur une entrée, dite entrée **d'horloge** (notée **H** ou **CLK**).

Un signal d'horloge est un signal basculant d'un état à son inverse périodiquement. La caractéristique fondamentale de ce signal est sa fréquence et sa période. Sa fréquence est l'inverse de sa période. La période d'horloge se divise en deux parties : une période haute et une période basse

Schématiquement le signal d'horloge est représenté comme suit :

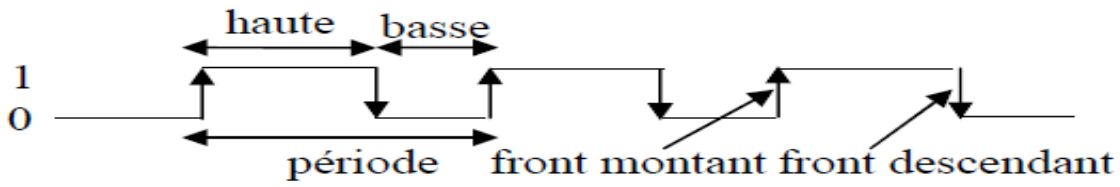


Figure 2.12 : Signale d'horloge.

Il existe deux types de synchronisation : par niveau ou par front d'impulsion.

Dans ***la synchronisation par niveau***, tous les changements d'état ont lieu pendant que l'horloge est à un niveau fixe (haut ou bas).

Dans ***la synchronisation par front d'impulsion***, tous les changements d'état se font sur un front d'impulsion fixé (montant ou descendant) d'horloge.

Exemple 2.2 : Bascule RSH

● **Symbole**

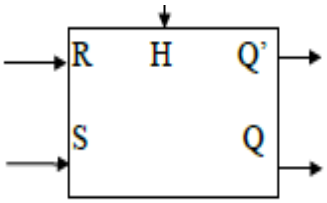


Figure 2.13: Schéma bloc de la bascule RSH.

● **Logigramme**

On représente la bascule RSH comme suit:

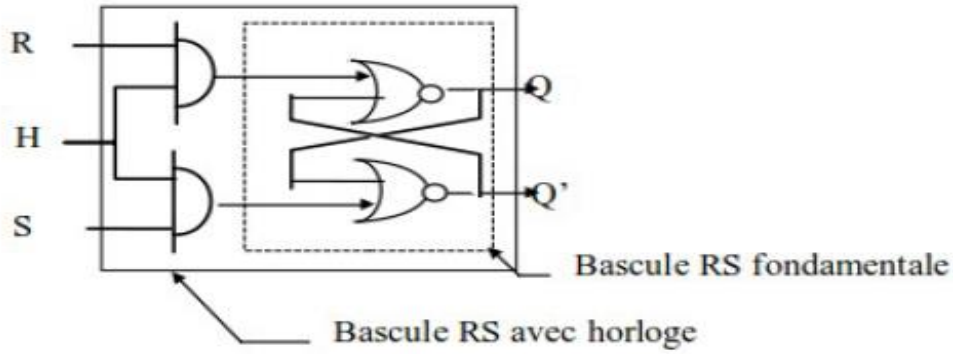


Figure 2.14: Logigramme de la bascule RSH.

Il suffit de faire un ET logique entre ses entrées et le signal d'horloge, pour imposer à ce que la prise en considération de ses entrées soit rythmée par le signal d'horloge.

Si *H=0* : la bascule conserve son état $Q_{n+1} = Q_n$: on dit que la bascule est bloquée, la sortie Q_n est indépendante des éventuels changements de R et S

Si *H=1* : Fonctionnement normal de la bascule. Donc la bascule **RS** ne fonctionne normal que si **H=1** (Niveau Haut). Même chose pour les autres bascules.

3. Les registres

3.1. Définition

Un registre est un ensemble de bascules synchronisées par le même signal d'horloge.

Les registres sont largement utilisés dans les machines numériques pour stocker un nombre limité de bits (8, 16, 32 et 64 bits sont souvent utilisés)

Pour mémoriser un ensemble de n bits a_{n-1}, a_{n-2}, .., a_0, ou mot de n bits, on utilise n bascules qui sont activées en même temps, donc par le même signal

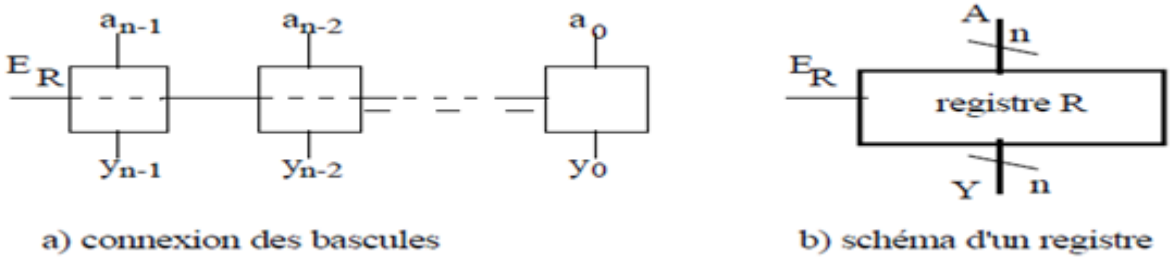


Figure 2.15 : Registre n bits.

Le type de bascules utilisé est choisi d'après l'utilisation du registre. On utilise souvent des bascules à deux entrées d'activation (**horloge** et **autorisation**) : l'horloge du système est connectée à l'entrée d'horloge des bascules, le signal E_R est connecté à l'entrée d'autorisation.

3.2. Classification des registres

Il existe quatre types de registre selon leurs modes de fonctionnement :

- Des registres à entrées parallèles et sorties parallèles : **PIPO** *(Parallel IN-Parallel OUT).*
- Des registres à entrées parallèles et sorties séries : **PISO** *(Parallel IN-Serial OUT).*
- Des registres à entrées séries et sorties parallèles : **SIPO** *(Serial IN- Parallel OUT).*
- Des registres à entrées séries et sorties séries : **SISO** *(Serial IN- Serial OUT).*

3.2.1. Registre de mémorisation (Registre parallèle)

C'est un registre qui permet de sauvegarder une information binaire sous forme d'un mot de n bits.

Il est donc constitué de n bascules, mémorisant chacune un bit. L'information sur n bits est chargée (écrite) puis elle est conservée et devient disponible en lecture.

➢Association de n bascules D pour mémoriser n bits

Exemple 2.3 : ***Registre de mémorisation 4 bits***

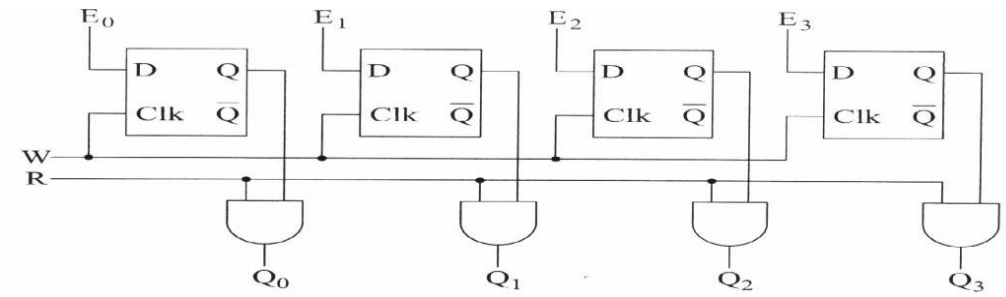


Figure 2.16 : Registre de mémorisation 4 bits réalisé à base de bascules D.

Les 4 bascules sont chargées en parallèle et lues en parallèle en synchrone avec le signal d'écriture W. Ce type de registre est appelé aussi registre **PIPO.**

➢Les entrées présentés sur E0, E1, E2, E3 sont mémorisées en synchronisation avec le signal W

➢Elles peuvent être lues sur les sorties Q0, Q1, Q2, Q3 en coïncidence avec le signal de validation R

Schéma fonctionnel d'un registre PIPO

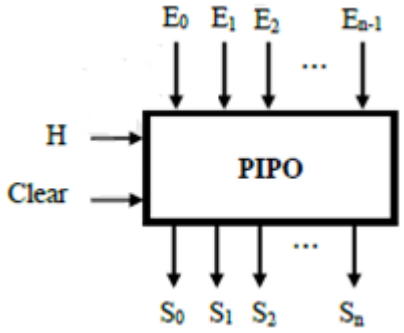


Figure 2.17: Schéma bloc d'un registre PIPO.

3.2.2. Registre à décalage

Ce type de registre est principalement utilisé comme mémoire d'information dynamique, il décale les informations vers la droite ou vers la gauche. Un registre de décalage est représenté par des bascules D interconnectées de façon a ce que l'état de la bascule de rang i soit transmis a la bascule de rang i+1 (ou i-1) quand un signal d'horloge est applique a l'ensemble des bascules

Le registre de décalage a 2 types d'entrées (2 fonctionnalités)

- Chargement parallèle, comme dans un registre de mémorisation (si decal =0)
- décalage à droit = chargement série (si decal = 1) (**SISO)**
  - l'information est présentée séquentiellement bit après bit a la 1ère bascule
  - à chaque coup d'horloge, un nouveau bit est présenté et ceux déjà charges sont décalés d'un rang

Exemple 2.4 : chargement le mot $a_1a_2a_3a_4$

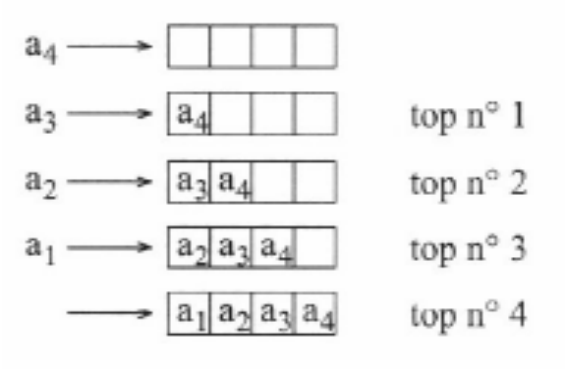
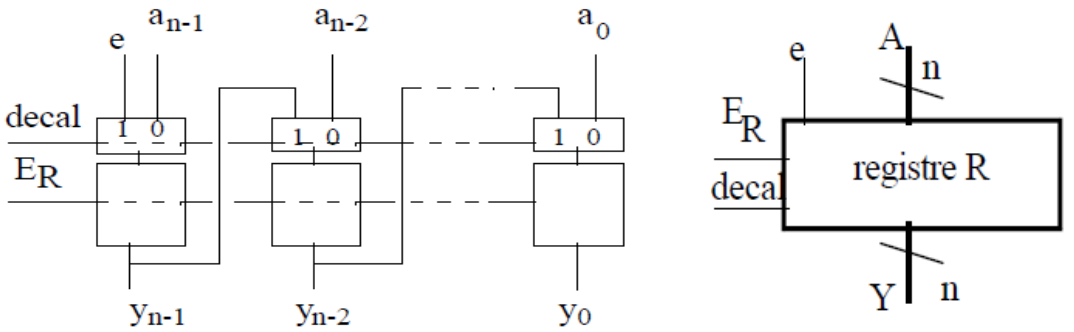


Figure 2.18 : Chargement d'un mot de 4 bits.

L'information est présentée séquentiellement bit après bit à la 1ère bascule, à chaque coup d'horloge, un nouveau bit est présenté et ceux déjà charges sont décalés d'un rang

Remarque : En entrée de chaque bascule, un multiplexeur commandé par le signal "decal" permet de sélectionner soit la valeur du bit de gauche, soit la valeur d'entrée. Dans tous les cas, il faut activer le signal ER pour charger le registre.



a) Connexion entre les bascule b) schéma bloc de registre à décalage

Figure 2.19 : Registre de décalage.

A. Décalage à droite

La bascule du rang i doit recopier la sortie de la bascule du rang (i-1) ainsi son entrée doit être connectée à la sortie (i-1).

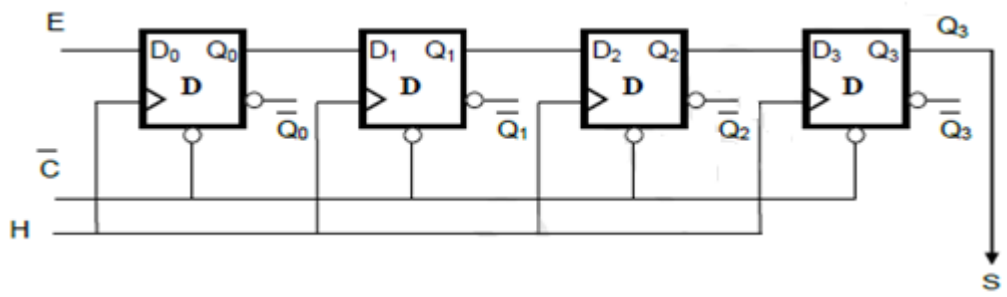


Figure 2.20 : Registre de décalage à droite de 4 bits réalisé à base de bascules D.

B. Décalage à gauche

L'entrée de la bascule du rang i doit recopier la sortie de la bascule du rang (i+1).

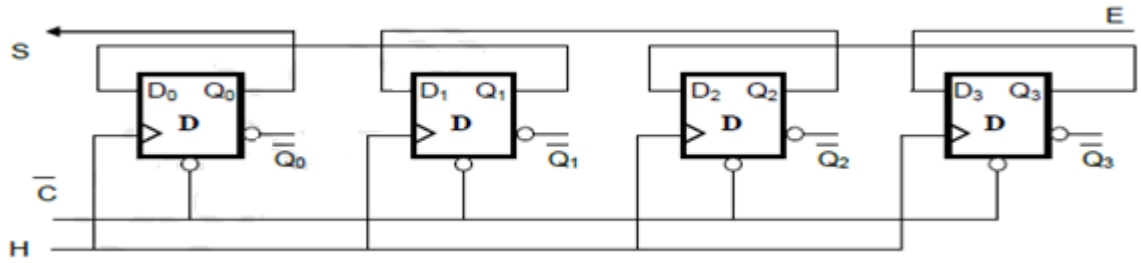


Figure 2.21 : Registre de décalage à gauche de 4 bits réalisé à base de bascules D.

C. Décalage réversible

Il existe des registres à décalage réversibles, c'est à dire des registres ou le décalage s'effectue vers la droite et vers la gauche en fonction du niveau logique applique à l'entrée S : « sens de décalage ».

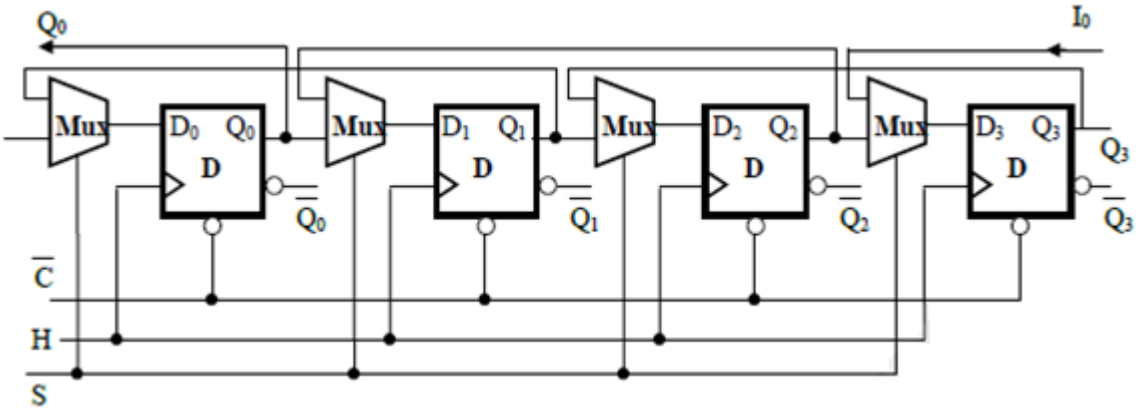


Figure 2.22 : Registre de décalage réversible de 4 bits réalisé à base de bascules D.

En fonction de la valeur de l'entrée S, on a l'opération suivante :

0 : Décalage à gauche

1 : Décalage à droite

3.2.3. Registre mixte

On peut trouver des registres mixtes, donc on peut écrire en parallèle et lire en série (**PISO**), ou vice versa (**PISO**), ou qui offrent les deux possibilités « au choix ».

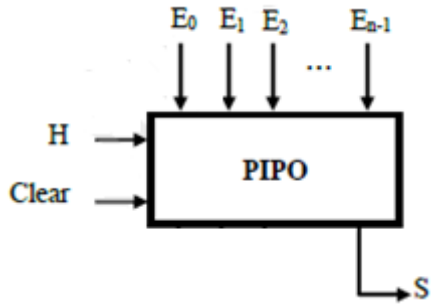
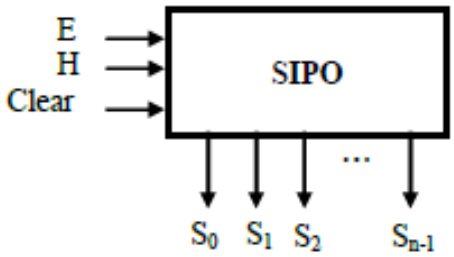
Registre PISO	*Registre SIPO*
	

Figure 2.23: Schéma bloc de deux registres mixtes PISO et SIPO.

4. Les mémoires

Une mémoire est un dispositif capable d'enregistrer, de conserver et de restituer des informations de taille importante.

4.1. Mémoires vives RAM (signifie en anglais : Random Access Memory)

Une mémoire vive est une mémoire dans laquelle on peut lire et écrire : Read/Write Memory ou RWM. Une mémoire vive est composée d'un ensemble ordonné de 2^m cellules, chaque cellule contenant un mot de n bits,

Pour pouvoir identifier individuellement chaque mot on utilise m lignes d'adresse. La taille d'un bloc mémoire est donc 2^m, le premier mot se situant à l'adresse 0 et le dernier à l'adresse $2^m - 1$.

Parmi les caractéristiques d'une mémoire nous trouvons la capacité et le format. La capacité représente le nombre total de bits et le format correspond à la longueur des mots. Si m est le nombre de bits d'adresse et n est le nombre de bits par mot, la capacité de la mémoire est donnée par : Capacité = 2^m mots = 2^m n bits)

4.2. Mémoires RAM Statiques / Dynamiques

Il existe deux grandes familles de mémoires RAM : les RAM statiques (SRAM) ou les RAM dynamiques (DRAM).

Dans le cas des RAM statique, le point mémoire élémentaire est une bascule.

Dans le cas des mémoires dynamiques (DRAM), l'élément de mémorisation est un condensateur (capacité) commandée par un transistor.

4.3. Mémoires mortes

Les mémoires mortes ou ROM (signifie en anglais : Read Only Memory) est une mémoire non volatile et non réinscriptible. Cette mémoire ne contient que le moniteur réalisant le chargement du système d'exploitation et les entrées sorties de plus bas niveau sont des mémoires à lecture et écriture qui permettent

5. Les automates

- L'étude des circuits combinatoires repose sur l'algèbre de Boole. Celle des circuits séquentiels repose sur la théorie des automates finis.

Une machine à état finis (automate) est une machine séquentielle algorithmique caractérisée par un vecteur d'entrées, un vecteur de sorties, et une séquence d'états définissant son comportement.

Tout circuit séquentiel peut être modélisé formellement par un automate. Inversement, "effectuer la synthèse d'un automate" est "réaliser un circuit séquentiel". Après avoir défini un automate.

5.1. Définition formelle

Un automate A est défini par un quintuplet (Q , i , E ,S , δ) où :
- Q est l'ensemble des états de l'automate,
-I est l'état initial
- E est son vocabulaire d'entrée (ensemble des valeurs possibles des entrées),
- S est son vocabulaire de sortie
- δ est l'ensemble des transitions $\subset Q \times E \dashrightarrow Q \times Q$

5.2. Représentation de l'automate

5.2.1. La table de transition

Les diagrammes de transitions ou les tables de transitions ou les matrices de transitions décrivent sous forme matricielle les fonctions E et S. Elles doivent contenir les états q_i à l'instant t, les entrées e_i, les états futurs q_i^+ et les sorties s_i , (Q, I, E, S, δ)

Exemple Soit un automate $M = (Q, I, E, S, \delta)$ définie par : $Q = I = E = \{0, 1\}$ et dont les fonctions de transition et de sortie sont définies par les tables suivantes :

S	0	1
A	A	B
B	A	B

Tableau 2.10: Exemple de fonctions d'E / S

δ	0	1
A	0	0
B	1	1

Tableau 2.11: Exemple de fonctions de transition

5.2.2. Le graphe d'états

Un automate peut être spécifié par un graphe d'états où :

- chaque sommet représente un état,
- chaque arc représente une transition d'un état à un autre, et est étiqueté par l'élément du vocabulaire d'entrée qui conditionne cette transition,
- les sorties sont associées aux arcs (automate de Mealy) ou aux états (automate de Moore).

Exemple 2.5: Diagramme d'état de l'exemple précédent.

$A \xrightarrow{1/0} B$

$A \xrightarrow{0/1} A$

$B \xrightarrow{1/0} A$

$B \xrightarrow{1/1} B$

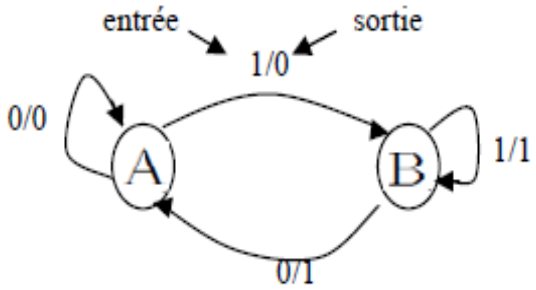


Figure 2.24: Exemple d'Automate.

$A \xrightarrow{X/Y} B$ Signifie que si on lit x dans l'état A alors on écrit y et on passe à l'état B.

On a dans ce cas δ (A, x) = (B,y).

On met une flèche pour indiquer l'état initial (ici A)

Exemple 2.6 :

Ensemble d'états : Q = {A,B,C}

L'état initial : i = A

Alphabet d'entrée : E = {0, 1}

Alphabet de sortie : S = {0, 1}

Fonction de transition δ :

δ (A, 0) = (C, 1)

δ (A, 1) = (B, 0)

δ (B, 0) = (C, 0)

δ (B, 1) = (B, 1)

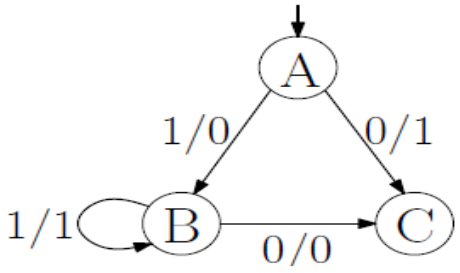


Figure 2.25: Diagramme d'état.

Exemple 2.7 : Diagrammes d'état des bascules JK, T et D

Rappelons les tables caractéristiques réduites des bascules JK, T et D

Q	Q^+	J	K	T	D
0	0	0	x	0	0
0	1	1	x	1	1
1	0	x	1	1	0
1	1	x	0	0	1

Tableau 2.12: Table de transition des bascules JK, T et D

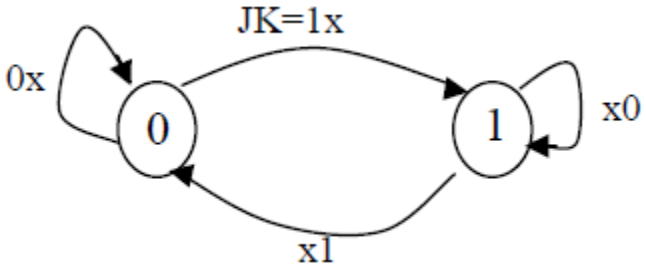


Figure 2.26: Diagrammes d'état des bascules JK.

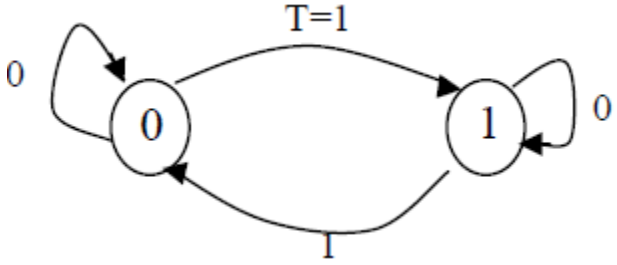


Figure 2.27: Diagrammes d'état des bascules T.

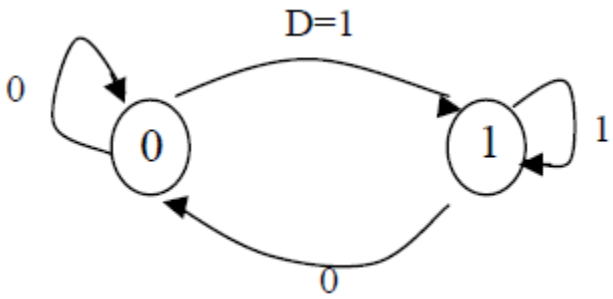


Figure 2.28: Diagrammes d'état des bascules D.

Remarque On ne peut pas avoir deux transitions sortant d'un même état qui correspondent à la même lettre.

5.3. Différents types de machines

Pour représenter les automates, il existe deux architectures différentes : la machine de MOORE et la machine de MEALY. Ils diffèrent seulement par la manière dont la sortie est générée

5.3.1. Machine de Moore (Le nom de machine de Moore fait référence au professeur Edward F. Moore de l'université de Winsconsin- Madison aux États-unis qui en est l'inventeur.)

On dit que la machine est une machine **de Moore** lorsque la sortie dépend uniquement de l'état courant (les sorties à l'intérieur de l'état), C a dire Les conditions (valeurs du vecteur d'entrée) sont notées sur les arcs et la valeur des sorties est généralement à l'intérieur du cercle (séparée du nom de l'état par un trait oblique : /).

Voici un exemple de machine à états finis où les sorties sont indiquées à l'intérieur des cercles :

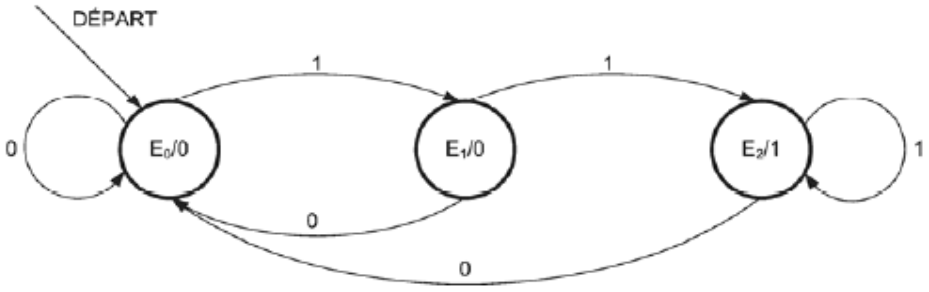


Figure 2.29: Exemple de machine à états finis de Moore.

● **Le fonctionnement :**

- au départ, la machine est à E0 et sa sortie à 0 ;
- tant que la machine ne reçoit pas 1, elle demeure à E0 et sa sortie à 0 ;
- si la machine reçoit 1, elle change d'état et part vers E1 : le début de la séquence 11 est détecté. La sortie demeure à 0 ……etc

5.3.2. Machine de Mealy (Le nom de machine de Moore fait référence à G.H. Mealy qui les fit connaître dans un article au Bell System Tech en 1955).

On dit que la machine est une machine de Mealy lorsque la sortie dépend de l'état courant et de l'entrée(les sorties sur les arcs). C a dire Les conditions (valeurs du vecteur d'entrée) sont notées sur les arcs et la valeur des sorties est généralement indiquée sur l'arc (séparée des entrées par un trait oblique : /)

Voici un exemple de machine à états finis où les sorties sont indiquées sur les arcs :

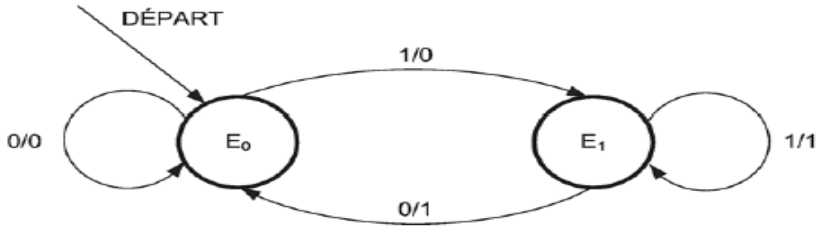


Figure 2.30: Exemple de machine à états finis de Moore.

6. Réalisation d'automates

6.1. Analyse d'un circuit séquentiel

Analyser un circuit séquentiel c'est déterminer son rôle.

Pour analyser un circuit séquentiel, on peut suivre les étapes suivantes :

1. Déterminer les fonctions des variables d'entrée.
2. Dresser la table caractéristique du circuit. De cette table, déduire les variables de sortie en se basant sur les expressions logiques des variables d'entrée.

Elle a la forme suivante :

Variables d'entrée	Q	Q^+
Connue.	connue	A déterminer

Tableau 2.13: Table de Caractéristique « Cas : analyse d'un circuit séquentiel »

Pour déterminer les états de sortie Q+, il faut utiliser la table caractéristique dont il est question dans le circuit à analyser.

3. Déduire le rôle du circuit analysé.

Exemple 2.8 : Analysons le circuit suivant :

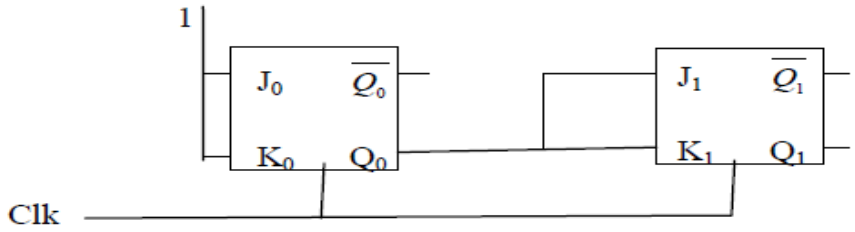


Figure 2.31: Exemple d'un circuit séquentiel.

● **Détermination des fonctions d'entrée de chaque bascule.**

J0=K0=1.

J1=K1=Q0.

● **Table caractéristique**

Q_1	Q_0	J_1	K_1	J_0	K_0	Q_1^+	Q_0^+
0	0	0	0	1	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0

Tableau 2.14: Exemple d'une table de Caractéristique d'un circuit séquentiel.

● **Conclusion :** C'est un compteur binaire comptant de 0 jusqu'à 3.

6.2. Synthèse d'un circuit séquentiel

La synthèse d'un circuit consiste à élaborer le circuit logique à partir du cahier de charge.

Pour faire la synthèse d'un circuit on peut suivre les étapes suivantes :

1. Etablir la table d'excitation correspondant au circuit à réaliser. Elle est de la forme :

Q	Q^+	Variables d'entrée
connue	connue	A déterminer

Tableau 2.15: Table de Caractéristique « Cas : synthèse d'un circuit séquentiel ».

2. Déduire l'expression de chaque variable d'entrée.

3. Réaliser le circuit en utilisant les bascules et les portes logiques requises.

6.3. Les compteurs

Un compteur est un système séquentiel auquel on imposer une séquence bien déterminée, c'est-à-dire une suite d'états binaires. Ces états restent stables et accessibles entre les impulsions de l'horloge.

Un compteur est composé d'une association de n bascules interconnectées par des portes logiques.

Ses caractéristiques principales :

· Capacité maximale du compteur (modulo)

· Nature : compteur ou décompteur ou aléatoire.

· Fonctionnement synchrone ou asynchrone.

· Le code dans lequel est exprimée la valeur sortie : binaire naturel, BCD, code Gray, …

6.3.1. Compteur modulo N

On appelle modulo du compteur le nombre N tel que le compteur compte jusqu'à (N-1). La $N^{ème}$ impulsion le remet obligatoirement à zéro. Si n est le nombre des bascules; N le modulo du compteur, on a:

$$\boldsymbol{2^{n-1} \leq N \leq 2^n}$$

6.3.2. Compteurs/ décompteur

On distingue deux types de compteurs :

A. Le compteur progressif (compteur)

Un compteur binaire est dit progressif (up counter), si ses états successifs représentent un nombre binaire croissant en fonction des impulsions d'entrées.

B. Compteur régressif (décompteur)

Un compteur binaire est dit régressif (down counter), si ses états successifs représentent un nombre binaire décroissant en fonction des impulsions d'entrées.

6.3.3. Compteurs synchrones et asynchrones

On distingue deux classes de compteurs

· Compteurs synchrones.

· Compteurs asynchrones.

A. Compteurs synchrones.

Un compteur binaire est dit synchrone si toutes les bascules reçoivent simultanément le même signal d'horloge. La synthèse des compteurs synchrones consiste à faire décrire au compteur une séquence déterminée il faut à chaque impulsion d'horloge définir les entrées J et K synchrones. Pour cela, on utilise la table de transition **(table d'excitation)** de la bascule JK

Exemple 2.9 : Compteur synchrone modulo 8

Réalisons un compteur modulo 8, formé de bascules type JK. Comme $2^2<7<2^3$ alors on a besoin de rois bascules.

● La table d'excitation

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	x	1	x	1	x	1

Tableau 2.16: Table d'excitation d'un compteur synchrone modulo 8.

● Les équations logiques

Q_3 \ Q_2Q_1	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$J_3=Q_2Q_1$

Tableau 2.17 : Table de karnaught de J_3.

Q_3 \ Q_2Q_1	00	01	11	10
0	x	x	x	x
1	0	0	1	0

$K_3= Q_2Q_1$

Tableau 2.18 : Table de karnaught de K_3.

Q_3 \ Q_2Q_1	00	01	11	10
0	0	1	x	x
1	0	1	x	x

$J_2=Q_1$

Tableau 2.19 : Table de karnaught de J_2.

Q_3 \ Q_2Q_1	00	01	11	10
0	x	x	1	0
1	x	x	1	0

$K_2=Q_1$

Tableau 2.20 : Table de karnaught de K_2.

Q_3 \ Q_2Q_1	00	01	11	10
0	1	x	x	1
1	1	x	x	1

$J_1=1$

Tableau 2.21 : Table de karnaught de J_1.

Q_3 \ Q_2Q_1	00	01	11	10
0	x	1	1	x
1	x	1	1	x

$K_1=1$

Tableau 2.22 : Table de karnaught de K_1.

● **Schéma logique**

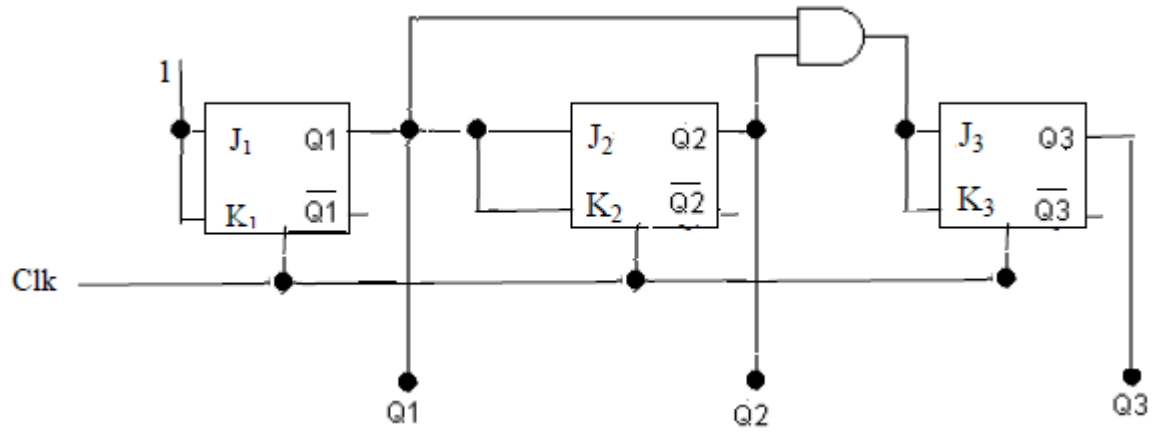


Figure 2.32: Schéma logique d'un Compteur synchrone modulo 8.

Exemple 2.10 : Décompteur synchrone modulo 8

Réalisons un décompteur modulo 8, formé de bascules type JK. Comme $2^2<7<2^3$ alors on a besoin de trois bascules.

● **La table d'excitation**

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	J_3	K_3	J_2	K_2	J_1	K_1
1	1	1	1	1	0	x	0	x	0	x	1
1	1	0	1	0	1	x	0	x	1	1	x
1	0	1	1	0	0	x	0	0	x	x	1
1	0	0	0	1	1	x	1	1	x	1	x
0	1	1	0	1	0	0	x	x	0	x	1
0	1	0	0	0	1	0	x	x	1	1	x
0	0	1	0	0	0	0	x	0	x	x	1
0	0	0	1	1	1	1	x	1	x	1	x

Tableau 2.23: Table d'excitation d'un décompteur synchrone modulo 8.

● **Les équations logiques**

Q_2Q_1 / Q_3	00	01	11	10
0	1	0	0	0
1	x	x	x	x

$J_3=\overline{Q_2}\,\overline{Q_1}$

Tableau 2.24 : Table de karnaught de J_3.

Q_2Q_1 / Q_3	00	01	11	10
0	x	x	x	x
1	1	0	0	0

$K_3=\overline{Q_2}\,\overline{Q_1}$

Tableau 2.25 : Table de karnaught de K_3.

Q_2Q_1 / Q_3	00	01	11	10
0	1	0	x	x
1	1	0	x	x

$J_2=\overline{Q_1}$

Tableau 2.26 : Table de karnaught de J_2.

Q_2Q_1 / Q_3	00	01	11	10
0	x	x	0	1
1	x	x	0	1

$K_2=\overline{Q_1}$

Tableau 2.27 : Table de karnaught de K_2.

Q_2Q_1 / Q_3	00	01	11	10
0	1	x	x	1
1	1	x	x	1

$J_1=1$

Tableau 2.28 : Table de karnaught de J_1.

Q_2Q_1 / Q_3	00	01	11	10
0	x	1	1	x
1	x	1	1	x

$K_1=1$

Tableau 2.29: Table de karnaught de K_1.

● **Schéma logique**

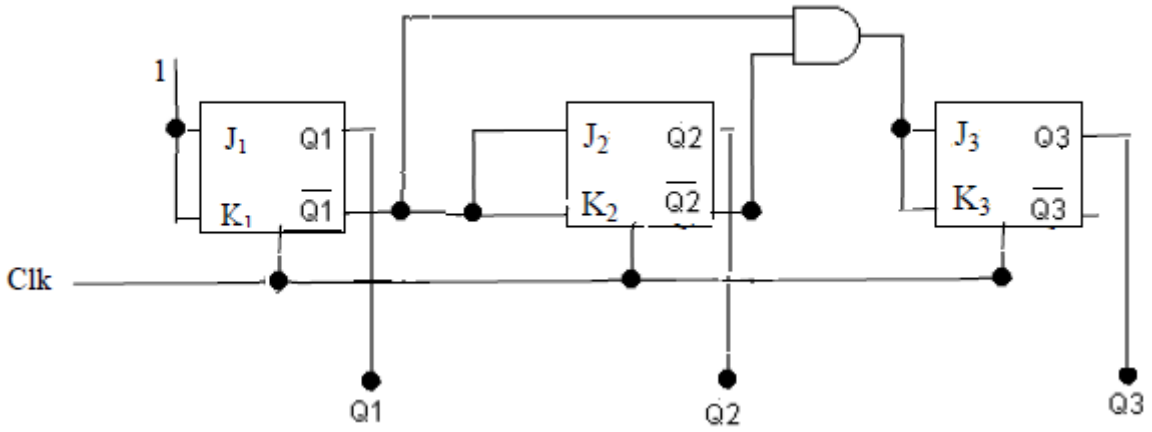


Figure 2.33: Schéma logique d'un décompteur synchrone modulo 8.

Exemple 2.11 : Compteur synchrone modulo 5

Faisons la synthèse d'un compteur synchrone progressif modulo 5 (compte de 0 à 4). Comme $2^2<5<2^3$ alors on a besoin de trois bascules.de type JK (par exemple).

● **La table d'excitation**

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	0	x	1	0	x	0	x
1	0	1	x	x	x	x	x	x	x	x	x
1	1	0	x	x	x	x	x	X	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

Tableau 2.30: Table d'excitation d'un compteur synchrone modulo 5.

● **Les équations logiques**

Q_3 \ Q_2Q_1	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$J_3=Q_2Q_1$

Tableau 2.31 : Table de karnaught de J_3.

Q_3 \ Q_2Q_1	00	01	11	10
0	x	x	x	x
1	1	x	x	x

$K_3= 1$

Tableau 2.32 : Table de karnaught de K_3.

Q_3 \ Q_2Q_1	00	01	11	10
0	0	1	x	x
1	0	x	x	x

$J_2=Q_1$

Tableau 2.33 : Table de karnaught de J_2.

Q_3 \ Q_2Q_1	00	01	11	10
0	x	x	1	0
1	x	x	1	x

$K_2=Q_1$

Tableau 2.34: Table de karnaught de K_2.

Q_3 \ Q_2Q_1	00	01	11	10
0	1	x	x	1
1	0	x	x	x

$J_1=\overline{Q_3}$

Tableau 2.35 : Table de karnaught de J_1.

Q_3 \ Q_2Q_1	00	01	11	10
0	x	1	1	x
1	x	x	x	x

$K_1=1$

Tableau 2.36 : Table de karnaught de K_1.

● **Schéma logique**

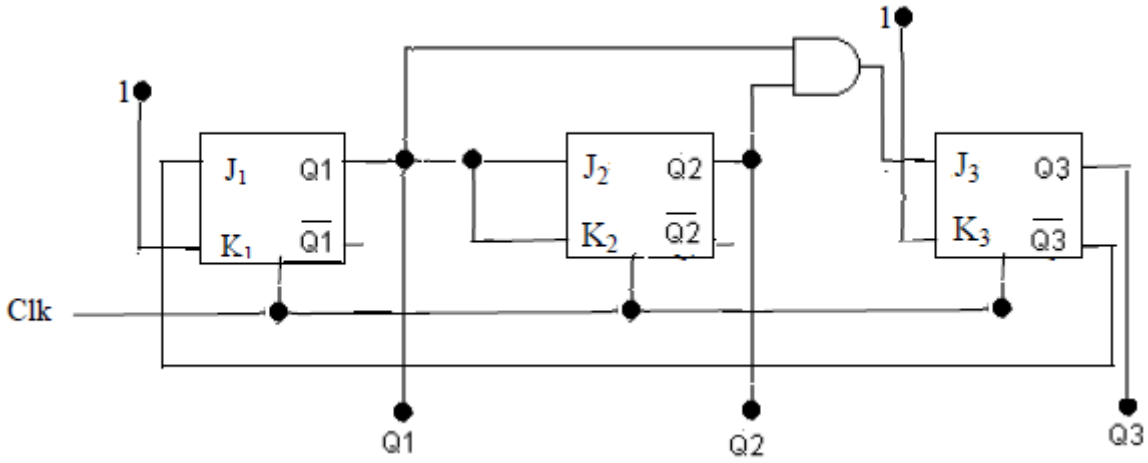


Figure 2.34: Schéma logique d'un compteur synchrone modulo 5.

B. Compteurs asynchrones.

Un compteur binaire est dit asynchrone si le signal d'horloge est appliqué seulement à l'entrée de la première bascule (premier étage), le signal d' horloge du i^{ème} étage n'est autre que la sortie du (i-1)^{ème} étage. C à dire l'état de chaque bascule est fonction des états des bascules précédentes.

On réalise ce compteur à partir des bascules T ou de bascules JK avec *J=K=1* sensibles au front négatif (descendant) du signal H $(Q_n = \overline{Q_{n+1}})$

Exemple 2.12 : Compteur asynchrone modulo 8

Faisons la synthèse d'un compteur asynchrone modulo 8 par bascules type T. Comme $2^2<7<2^3$ alors on a besoin de trois bascules type T.

● **La table de vérité des transitions**

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Tableau 2.37: Table de transitions d'un compteur asynchrone modulo 8.

● **Chronogramme de fonctionnement**

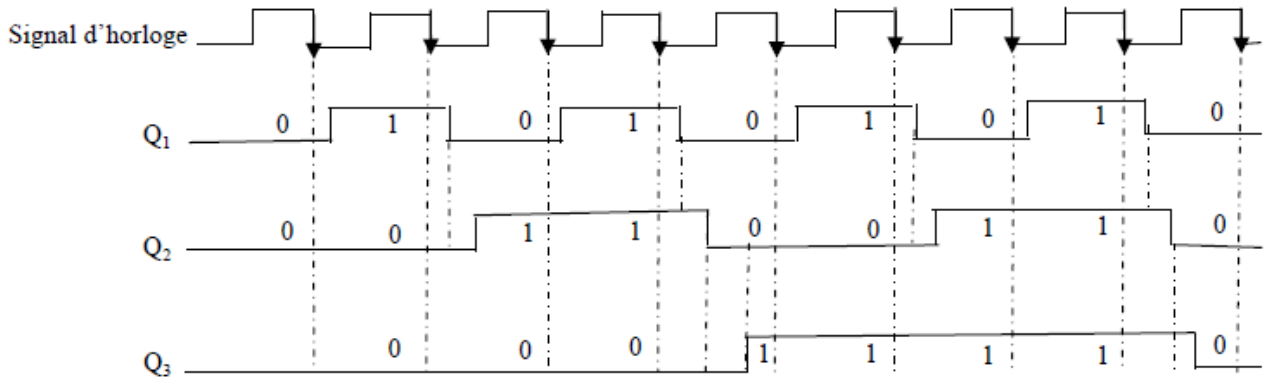


Figure 2.35: Chronogramme de fonctionnement d'un compteur asynchrone modulo 8.

En analysant ce chronogramme on remarque que, qu'un étage d'ordre i change d'état si l'étage précédent passe de l'état '1' à l'état '0' et conserve son état dans les autres cas.

● **Schéma logique.**

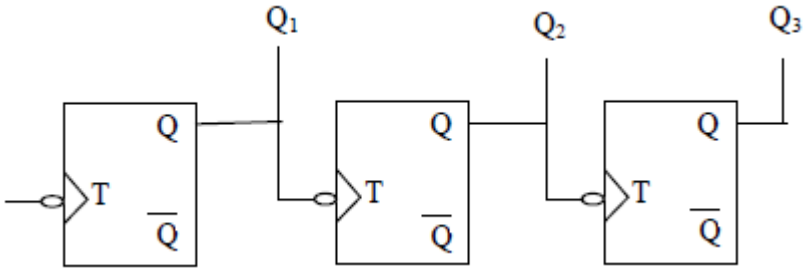


Figure 2.36: Schéma logique d'un compteur asynchrone modulo 8.

Exemple 2.13 : Décompteur asynchrone modulo 8 à front montants

Faisons la synthèse d'un compteur asynchrone modulo 8 à fronts montants par bascules type T. Comme $2^2<=8<=2^3$ alors , on a besoin de trois bascules type T.

● **La table de vérité des transitions**

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+
1	1	1	1	1	0
1	1	0	1	0	1
1	0	1	1	0	0
1	0	0	0	1	1
0	1	1	0	1	0
0	1	0	0	0	1
0	0	1	0	0	0
0	0	0	1	1	1

Tableau 2.38: Table de transitions d'un décompteur asynchrone modulo 8.

● **Chronogramme de fonctionnement**

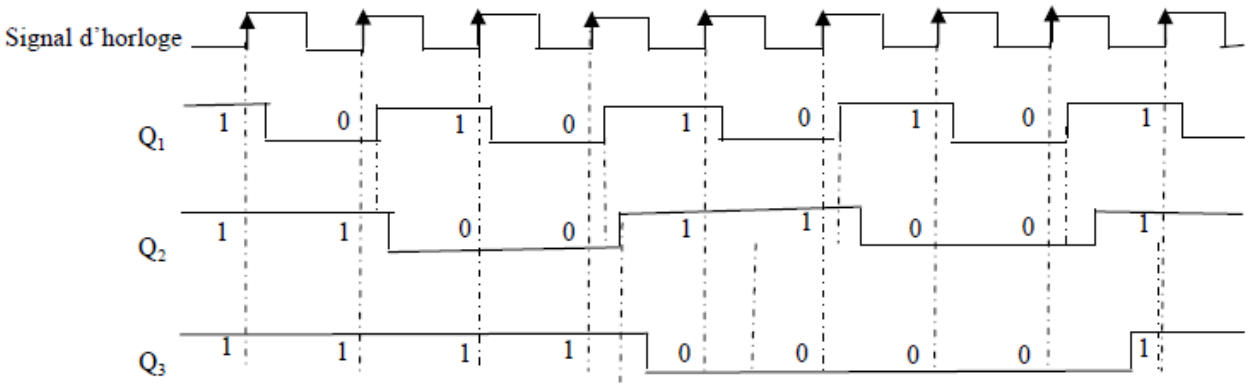


Figure 2.37: Chronogramme de fonctionnement d'un décompteur asynchrone modulo 8.

En analysant ce chronogramme on remarque que, qu'un étage d'ordre i change d'état si l'étage précédent passe de l'état '0' à l'état '1' et conserve son état dans les autres cas.

● **Schéma logique**

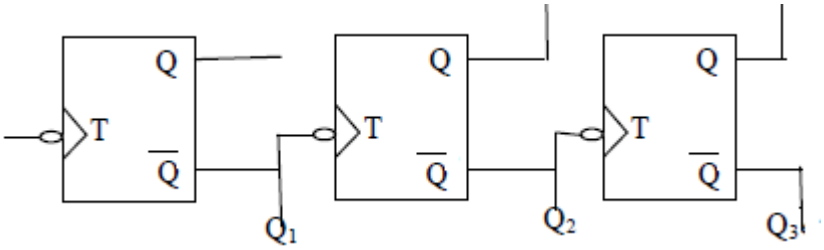


Figure 2.38: Schéma logique d'un décompteur asynchrone modulo 8.

Chapitre 3 : Les circuits intégrés

Introduction

Le besoin de réduction des coûts ont depuis longtemps poussé les fabricants de composants électroniques à intégrer au maximum les structures électroniques donnant naissance aux *Circuits Intégrés*.

1. Définitions

Un **circuit intégré** (**CI**) désigne un bloc constitué par un monocristal de silicium *(Puce)* de quelques millimètres carrés à l'intérieur, du quel se trouve inscrit en nombre variable des composants électroniques élémentaires *(Transistors, diodes, résistances, condensateurs, ...).*

2. Les types de circuits intégrés

On en distingue deux types: analogique et numérique.

2.1. Les Circuits Intégrés Linéaires (analogiques)

Les circuits intégrés analogiques les plus simples peuvent être de simples transistors encapsulés les uns à côté des autres sans liaison entre eux, jusqu'à des assemblages complexes pouvant réunir toutes les fonctions requises pour le fonctionnement d'un appareil dont il est le seul composant.

2.2. Les Circuits Intégrés logiques (Numériques)

Les circuits intégrés numériques correspondent aux différentes portes et circuits logiques (combinatoires et séquentiels qui sont tous construits à base des transistors. Les circuits intégrés numériques les plus simples sont des portes logiques (AND, OR, NAND, NOR, NON), les plus complexes sont les microprocesseurs et les plus denses sont les mémoires.

3. Caractéristiques d'un CI

Les circuits intégrés possèdent plusieurs caractéristiques à prendre en compte lors de la conception d'un dispositif électronique. On distingue les caractéristiques comportementales et les caractéristiques liées aux performances et d'autres liés aux compatibilités entre plusieurs familles de CI.

Caractéristiques comportementales :

- Délai entre changement des entrées et mise à jour de la sortie (délai de propagation)
- taux de sensibilité aux bruits (parasites) peut réduire la fiabilité
- Limites de puissance : Une sortie ne peut fournir de signal à plus de N entrées 33

Caractéristiques de performances :

- consommation électrique
- vitesse de propagation des signaux

Caractéristiques de compatibilité entre famille de CI :

- Gammes différentes de tension d'alimentation
- Codage incompatible des niveaux de tension H (height ou haut) et L (low ou bas)

4. Composition d'un CI

Un CI est généralement composé d'une surface (**puce**) de silicium de quelques millimètres carrés à l'intérieur du quel se trouve inscrit en nombre variable des composants électroniques élémentaires *(Transistors, diodes, résistances, condensateurs, etc.).*

Cette puce est reliée à la montre extérieure par des connecteurs se traduisant sous forme de **broches** ou de **pattes**. Le tous est maintenu par un **support** isolant.

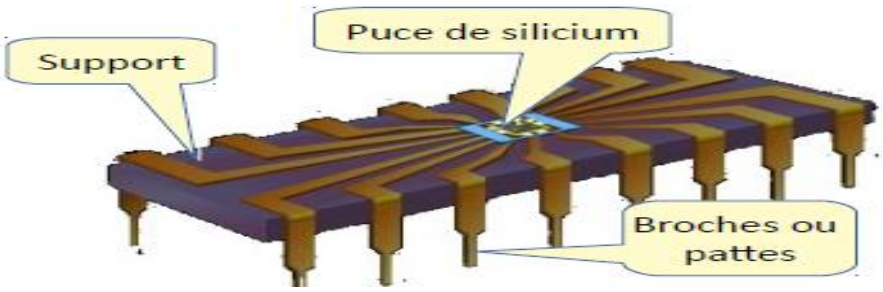


Figure 3.1 : Exemple de CI.

5. Montage d'un circuit utilisant des CI

Les montages électroniques se font par la mise en liaisons de plusieurs composants de bases (circuits intégrés ou composants discrets comme les transistors, les diodes, les résistances, les bobines et les capacités) soudés ou implantés sur des supports isolants. Pour maintenir l'ensemble on utiliser soit un circuit imprimé soit une maquette d'essai.

Voici un exemple de montage électronique sur une maquette d'essai :

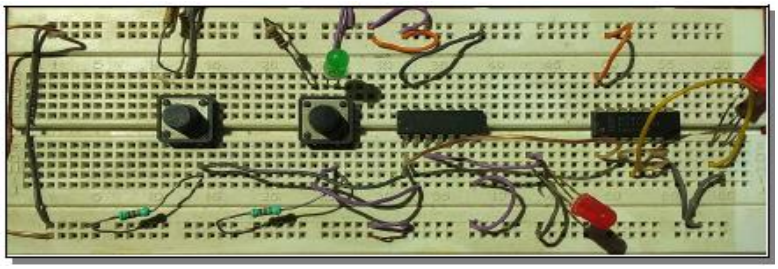


Figure 3.2 : Exemple de réalisation d'un CI sur une plaque d'essai.

Voici un exemple de montage réalisé à base d'un circuit imprimé :

Figure 3.3 : Exemple de montage d'un CI réalisé à base d'un circuit imprimé.

6. Familles des circuits intégrés logiques

Il existe plusieurs familles de circuits technologiques. Les 2 plus utilisées sont:

- TTL *(Transistors Transistors Logic)*
- CMOS *(transistors à effet de champ MOS - Complémentaire - Métal - Oxyde - Semi-conducteur)*

6.1. Circuits logiques TTL

6.1.1. Présentation

Transistor-Transistor Logic ou **TTL** est une famille de circuits logiques utilisée en électronique inventée dans les années 1960. Cette famille est réalisée avec la technologie du transistor bipolaire et tend à disparaître du fait de sa consommation énergétique élevée (comparativement aux circuits CMOS).

6.1.2. Caractéristiques

- Gamme d'alimentation : **5 V +/- 5%.**
- Gamme de température : **de 0 °C à + 70 °C.**
- Puissance dissipée : environ **2 mW** par porte (série LS).

- Fréquence de fonctionnement: jusqu'à **3 MHz.**
- Sortance: jusqu'à **20** (série LS). (Nombre d'entrées que l'on peut relier à une sortie de porte)

6.1.3. Les avantages

- Les entrées laissées en 'l'air' ont un état logique à 1 par défaut.
- Une bonne immunité au bruit.
- Un temps de propagation faible.

6.1.4. Les inconvénients

- L'alimentation doit être précise à 5V +/- 5 % sinon on risque de détruire le circuit.
- Du fait qu'elle est réalisée avec des transistors bipolaires elle consomme pas mal de courant comparer à la famille CMOS. (Car les transistors bipolaires sont commandés en courant).

6.2. Circuits logiques CMOS

6.2.1. Présentation

CMOS est l'abréviation de "**Complementary Metal Oxide Semi-conductor**". Le premier dispositif MOS est apparu en 1960. Son développement a été rendu possible par les progrès réalisé par la technologie TTL. Cette famille est réalisée avec des transistors à effet de champs.

6.2.2. Caractéristiques

- Gamme d'alimentation : **de 3 V à 15 V.**
- Gamme de température: **de – 40 °C à + 85 °C**
- Puissance dissipée : environ **10 nW** par porte.
- Fréquence de fonctionnement: jusqu'à **12 MHz.**
- Sortance: jusqu'à **50** (série 4000B).(Nombre d'entrées que l'on peut relier à une sortie de porte)
- Excellente immunité aux bruits.

6.2.3. Les avantages

- L'alimentation peut aller de 3V à 15V.

- Le courant d'entrée est nul, car elle est réalisée avec des transistors à effet de champs. (Les transistors à effet de champs sont commandés en tension).
- Une excellente immunité au bruit.

6. 2.4. Les inconvénients

- La vitesse de commutation est plus faible que pour la technologie TTL.
- Sensibilité aux décharges électrostatiques.

Références

Références

M. GOSSA : « Cours AU3: systèmes logiques séquentiels », ISEFC, OCTOBRE 2002,

J. LAGRASSE – M. COUVOISIER – J.P. RIHOD : « La logique combinatoire », 3eme édition DUNODE,

J. LAGRASSE – M. COUVOISIER – J.P. RIHOD : « La logique séquentielle », 3eme édition DUNODE,

D. MANG: « Analyse et synthèse des systèmes logiques », 2eme édition DUNODE,

M. AUMIAUS : « Logique binaire: fonctions logiques et arithmétique », 2eme édition DUNODE.

Robert Strandh, Irène Durand, « Architecture de l'ordinateur », Collection: Sciences Sup, Dunod 2005,

M.F KHALFI : « Cours Structure Machine », Université djilali liabès de SIDI BEL-ABBES, 2015-2016

H. BENSALAH – S.MFAZIL DIB- M. MAMCHAOUI : « Cours Structure Machine 2 », Univesité Abou Bekr Belkaïd Tlemcen

L. BOUZIDI: « Cours Structure Machine 2 », Univesité de Béjaya , 2018-2019