UNIVERSITE MOHAMED KHIDER DE BISKRA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

DEPARTEMENT DE GENIE ELECTRIQUE
Filière : Electronique

Support de cours

Logique combinatoire et séquentielle

(2018- 2019)

Dr. Hendaoui Mounira

Courriel : m.hendaoui@univ-biskra.dz

Module : Logique combinatoire et séquentielle

Contenu de Module

Chap1 : Algèbre de BOOLE et simplification des fonctions logiques.

Chap2 : Systèmes de numération et codage de l'information.

Chap3 : Circuits combinatoires transcodeurs.

Chap4 : Circuits combinatoires aiguilleurs

Chap5 : Circuits combinatoires de comparaison.

Chap6 : Les operateurs arithmétiques additionneurs et soustracteurs.

Chap7 : Les bascules.

Chap8 : Les compteurs.

Chap9 : Les registres.

Généralités

Depuis 1980, le traitement numérique remplace le traitement analogique des signaux, ce qui a crée une révolution dans l'électronique.

Pour passer de la forme analogique vers la forme numérique, on peut établir le schéma type d'une chaîne de traitement numérique :

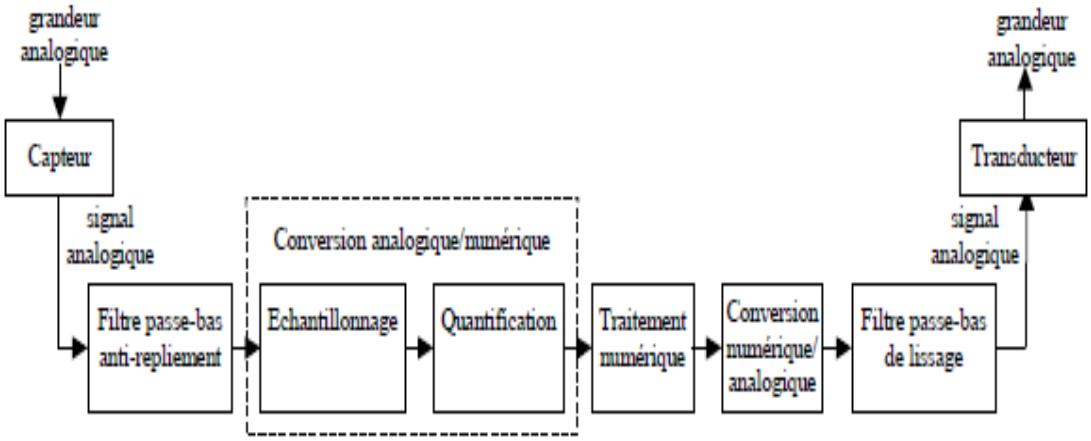


Figure 1 : Chaine de traitement numérique d'un signal.

Les trois opérations principales dans cette chaîne sont :

- La transformation du signal analogique en un signal numérique s'exécute à travers le filtrage passe-bas anti-repliement, l'échantillonnage et la quantification. La conversion analogique/numérique, s'établie dans les deux dernières étapes.
- Le traitement numérique.
- La transformation du signal numérique en un signal analogique s'exécute à travers la conversion numérique/analogique et le filtrage passe-bas de lissage.

Un système numérique complexe est le résultat d'un assemblage hiérarchique d'opérateurs logiques élémentaires réalisant des opérations simples sur des variables logiques. Ces variables logiques ne prennent que deux états : vrai ou faux. Il y'a plusieurs systèmes physiques qui ne pouvant prendre que deux états:

- interrupteur fermé ou ouvert,
- lampe allumée ou pas,
- moteur en marche ou pas,

Généralement on donne à l'état vrai d'une variable logique la valeur binaire 1 et la valeur 0 à l'état faux. C'est la logique positive. Contrairement à la logique positive, la logique négative donne la valeur 0 à l'état vrai et la valeur 1 à l'état faux, mais dans ce cours, on travaillera toujours en logique positive.

Le mathématicien britannique BOOLE a créé une algèbre Booléenne à l'aide de variables logiques ne pouvant prendre que deux états et d'opérations élémentaires portant sur une ou 2 variables, cet algèbre est faite pour traiter que de la logique combinatoire.

Un circuit combinatoire est un circuit où la sortie des circuits numériques ne dépend que de l'état présent des entrées (sans mémoire des états passés) et cela limite considérablement le domaine de leurs applications. D'ici vient l'importance des circuits séquentiels qui permettent la mise au point de systèmes dont le fonctionnement dépend non plus seulement des entrées reçues, mais aussi de l'état précédent. Le circuit séquentiel utilise une partie mémoire qui va lui permettre de retrouver l'état induit par les entrées passées.

Dans ce cours, on va traiter la logique combinatoire et la logique séquentielle.

Chapitre1 : Algèbre de BOOLE et simplification des fonctions logiques.

1.1 Introduction

Dans l'algèbre de ***Boole***, une variable ou une fonction logique de *n* variables ne peut prendre que 0 ou 1 comme valeur possible.

1.2 Les fonctions logiques

Dans l'algèbre de Boole, trois fonctions élémentaires suffisent pour définir cet algèbre : la complémentation, l'addition logique, et le produit logique.

2.2.1 La fonction NON

La fonction NON est la fonction de complémentation ou d'inversion logique où la variable $\bar{A}$ est le complément de la variable *A*.

L'opérateur correspondant, appelé inverseur, Le tableau 1.1 donne la table de vérité de la fonction NON et les symboles logique utilisés pour l'opérateur correspondant.

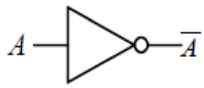
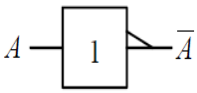
Table de vérité		Symbole traditionnel	Symbole normalisé
A	$\bar{A}$		
0	**1**		
1	**0**		

Tableau 1.1 : La fonction NON.

1.2.2 La fonction ET

La fonction ET est la fonction produit logique de deux variables logique A et B,la fonction ET se note AB,A.B ou bien encore A∧B.Le tableau 1.2 donne la table de vérité de la fonction ET, et les symboles logiques de l'opérateur ET.

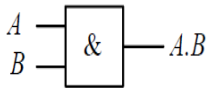
Table de vérité	Symbole traditionnel	Symbole normalisé
<table><tr><th>A</th><th>B</th><th>A.B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>		

Tableau 1.2 : La fonction ET.

1.2.3 La fonction OU

La fonction OU est l'addition logique de 2 variables A et B, cette fonction se note A+B ou A∨B. Le tableau 1.3 donne la table de vérité de cette fonction ainsi que les symboles logiques de l'opérateur OU.

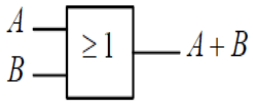
Table de vérité	Symbole traditionnel	Symbole normalisé
<table><tr><th>A</th><th>B</th><th>A+B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>		

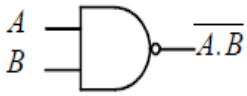
Tableau 1.3: La fonction OU.

1.2.4 La fonction NON ET (NAND)

La fonction NAND est la fonction inverse de la fonction ET. Le tableau 1.4 donne la table de vérité de cette fonction ainsi que les symboles logiques de son opérateur logique.

Table de vérité	Symbole traditionnel	Symbole normalisé

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



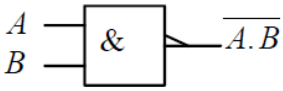


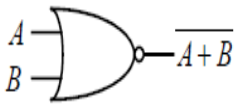
Tableau 1.4: La fonction NAND.

1.2.5 La fonction NON OU (NOR)

La fonction NOR est la fonction inverse de la fonction OU. Le tableau 1.5 donne la table de vérité de cette fonction ainsi que les symboles logiques de son opérateur logique.

Table de vérité	Symbole traditionnel	Symbole normalisé

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



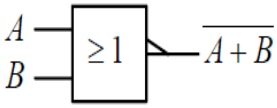


Tableau 1.5: La fonction NOR.

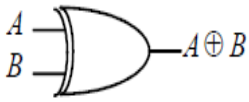
1.2.6 La fonction OU exclusif (XOR)

La fonction OU exclusif est une fonction logique où son operateur aura la valeur VRAI seulement si les deux opérandes A et B ont des valeurs distinctes. Le tableau 1.6 donne la table de vérité de cette fonction ainsi que les symboles logiques de l'opérateur XOR.

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

Table de vérité	Symbole traditionnel	Symbole normalisé

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



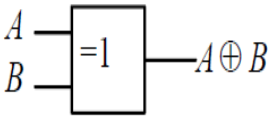


Tableau 1.6 : La fonction XOR

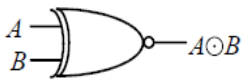
1.2.7 La fonction ET inclusif (XNOR)

La fonction XNOR est la fonction inverse de la fonction XOR. Le tableau 1.7 donne la table de vérité de cette fonction ainsi que les symboles logiques de son opérateur logique.

$A \otimes B = \overline{A \oplus B} = AB + \overline{A} \cdot \overline{B}$.

Table de vérité	Symbole traditionnel	Symbole normalisé

A	B	$A \otimes B$
0	0	1
0	1	0
1	0	0
1	1	1



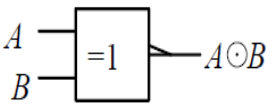


Tableau 1.7 : La fonction XNOR.

1.3 Propriétés des fonctions NON, ET, et OU

En algèbre de Boole les propriétés fondamentales se résume dans ce tableau :

Propriétés :	Fonctions ET :	Fonctions OU :
Commutativité	A . B = B . A	A + B = B + A.
Associativité	A (BC) = (AB) C = ABC	A +(B + C) =(A + B) +C =A + B + C
Eléments neutres	(A . 1) = A	(A + 0) = A.
Eléments absorbants	A.0 = 0. A = 0	A +1= 1+ A = 1
L'idempotence	(A . A) = A	(A + A) = A
L'inversion	$\overline{\overline{A}} = A,\ \overline{A}\cdot A = 0$	$\overline{A} + A = 1$
Distributivité du : ET/OU et du : OU /ET :	$A(B+C) = AB + AC$ $(A+B)C = AC + BC$	$A + BC = (A+B)(A+C)$ $AB + C = (A+C)(B+C)$

Tableau 1.8 : Propriétés des fonctions NON, ET, et OU.

Autres relations

1. $A + AB = A$;
2. $A(A+B) = A$;
3. $A + \overline{A}B = A + B$;
4. $A(\overline{A} + B) = AB$.

1.4 Théorème de Morgan

Les lois de De Morgan ont été formulées par le mathématicien britannique Augustus De Morgan (1806-1871).

Cet loi montre que la négation de la disjonction de deux propositions est équivalente à la conjonction des négations des deux propositions, ce qui signifie que non (A ou B) est (non A) et (non B), et que la négation de la conjonction de deux propositions est équivalente à la disjonction des négations des deux propositions, ce qui signifie non(A et B) est (non A) ou (non B).

Mathématiquement on peut avoir les deux expressions suivantes :

1. $\overline{A+B} = \overline{A}\cdot\overline{B}$;
2. $\overline{A\cdot B} = \overline{A} + \overline{B}$.

Et en générale on a :

1. $\overline{\sum_i X_i} = \prod_i \overline{X_i}$
2. $\overline{\prod_i X_i} = \sum_i \overline{X_i}$

1.5 Simplification des fonctions combinatoires.

1.5.1 La forme algébrique des fonctions

On peut avoir l'expression d'une fonction F en à l'aide des combinaisons des variables *A, B,* et *C* pour lesquelles F est égale à 1.

Exemple : d'après la table de vérité suivante, F vaut 1 pour les combinaisons 0, 1, et4 :

N°	A	B	C	F	$\overline{F}$
0	0	0	0	1	0
1	0	0	1	1	0
2	0	1	0	0	1
3	0	1	1	0	1
4	1	0	0	1	0

Tableau 1.9 : table de vérité de F.

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

1.5.2 Simplification des fonctions logiques

1.5.2.1 Simplification algébrique

Elle demande l'application des propriétés de l'algèbre de Boole aux expressions algébriques des fonctions logiques.

Exemple : simplification de : F1 = BC + AC + AB + B.

D'abord : AB + B = B donc F1 = BC + AC + B et BC + B = B donc :

F1 = AC + B.

1.5.2.2 Simplification par diagramme de Karnaugh

Le diagramme ou tableau de Karnaugh est un outil graphique qui permet de simplifier de façon méthodique une fonction logique, pratiquement ils ne sont utilisables que pour un nombre de variables inférieur ou égal à 6.

a) Tableau de Karnaugh d'une fonction de 2 variables :

Exemple :

B \ A	$\overline{A}\,\overline{B}$	$A\,\overline{B}$
$\overline{B}$	$\overline{A}\,B$	AB

B \ A	1	1
$\overline{B}$	1	0

Tableau 1.10 : Diagramme de Karnaugh de 2 variables.

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

b) Fonction de 3 variables : Exemple :

C \ B A	$\overline{A}\,\overline{B}$	$\overline{A}B$	AB	$A\overline{B}$
$\overline{C}$	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
C	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

Tableau 1.11 : Diagramme de Karnaugh de 3 variables

C \ B A	00	01	11	10
0	1	1	1	0
1	1	0	1	0

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

c) Fonction de 4 variables : Exemple :

D \ B A	00	01	11	10
00		1	1	
01				
11		1		1
10				1

Tableau 1.12: Diagramme de Karnaugh de 4 variables.

$$F = \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}BCD + A\overline{B}CD + A\overline{B}C\overline{D} .$$

d) Fonction de 5 et 6 variables :

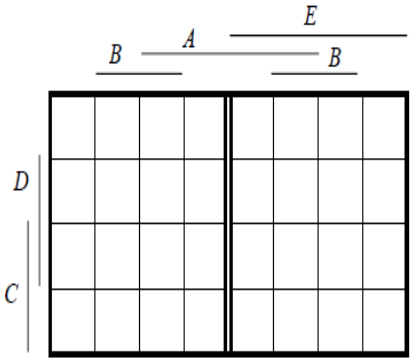


Tableau 1.13 : Diagramme de Karnaugh de 5 variables

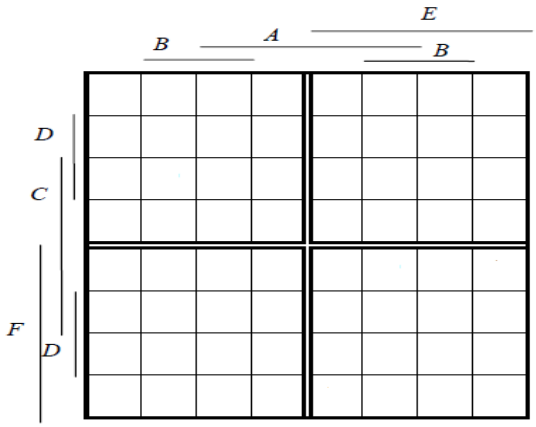


Tableau 1.14 : Diagramme de Karnaugh de 6 variables.

.

1.5.2.3 Principe de la simplification

On regroupe les cases adjacentes contenant un 1 par paquets de 2^n .

Exemple : Simplification de :

$$F = \overline{A}\overline{B}CD + AB\overline{C}\,\overline{D} + \overline{A}\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D + AB\overline{C}D + A\overline{B}\,\overline{C}D + \overline{A}\overline{B}CD + ABCD + A\overline{B}CD + \overline{A}BC\overline{D} .$$

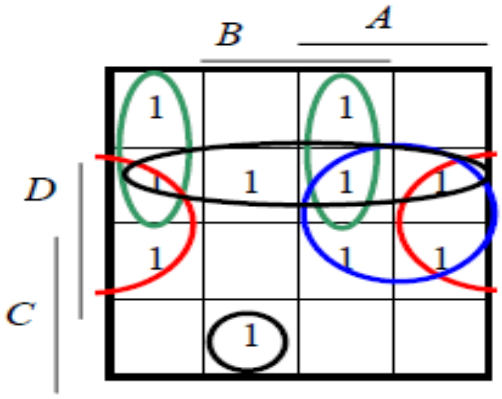
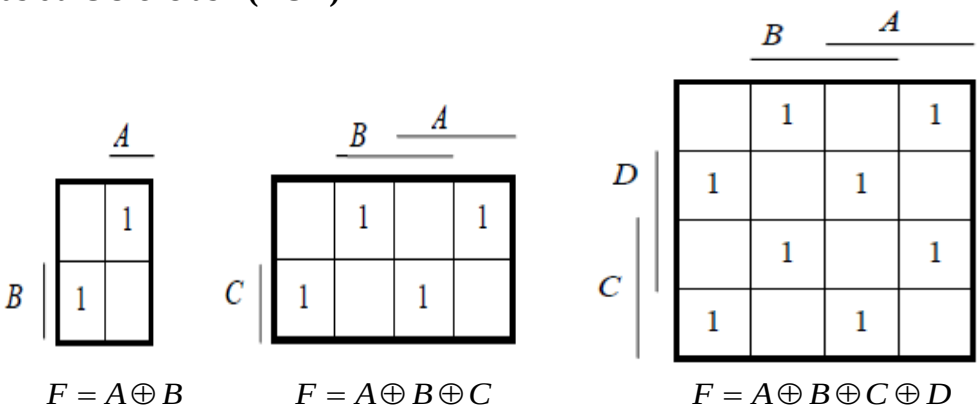


Tableau 1.15: Diagramme de Karnaugh de 4 variables.

Après simplification on aura :

$$F = AD + \overline{B}D + \overline{C}D + AB\overline{C} + \overline{A}\overline{B}\overline{C} + \overline{A}BC\overline{D}$$

Les cas du OU exclusif (XOR)



$F = A \oplus B$ $F = A \oplus B \oplus C$ $F = A \oplus B \oplus C \oplus D$

Au-delà de 6 variables, l'utilisation des diagrammes de Karnaugh devient presque impossible. Il est alors nécessaire pour simplifier les fonctions logiques d'utiliser des méthodes algorithmiques appliquées par les calculateurs.

Chapitre 2 : Système de numération et codage de l'information

2.1 Comment représenter des nombres dans des bases déférentes de numération et comment faire les conversions entre ces bases ?

Les bases de numération les plus utilisées sont : la base décimale (base 10), la base binaire (base 2), la base octale (base 8) et la base hexadécimale (base 16).

La numération binaire utilise les 2 bits 0 et 1, la numération octale utilise 8 chiffres : 0, 1, 2, 3, 4, 5, 6, 7 et la numération hexadécimale utilise 16 symboles : 0, 1, 2, ..., 9, A, B, C, D, E, F (les symboles A à F ont pour équivalents décimaux les nombres de 10 à 15).

Les conversions les plus utilisées sont les suivantes

- base *b* vers base 10 ;
- base 10 vers base *b ;*
- base 2 vers base (8 ou 16) ;
- base (8 ou 16) vers base 2 ;

Exemple 1 : conversion du nombre binaire fractionnaire $N_{(2)} = 110011{,}1001_{(2)}$ en base 10.

$$N = 1.2^5 + 1.2^4 + 1.2^1 + 1.2^0 + 1.2^{-1} + 1.2^{-4} = 51{,}5625_{(10)}$$

Exemple 2 : conversion du nombre octal entier $N_{(8)} = 4513_{(8)}$ en base 10.

$$N = 4.8^3 + 5.8^2 + 1.8^1 + 3.8^0 = 2379_{(10)}$$

2.2 Base 10 vers base b

2.2.1 Nombres entiers

La conversion d'un nombre décimal vers une autre base se réalise on appliquant la méthode des divisions successives de ce nombre par cette base, les restes successifs forment alors le nombre converti voulu.

Exemple 1 : conversion de $N_{(10)}$ =52 en base 2.

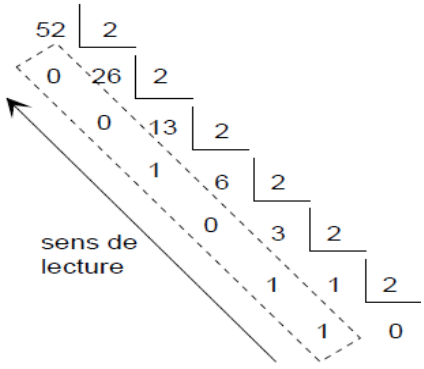


Figure 2.1 : conversion de $52_{(10)}$ en base 2.

Donc : $52_{(10)} = 110100_{(2)}$.

Exemple 2 : conversion de $N_{(10)}$ = 90 en base 8.

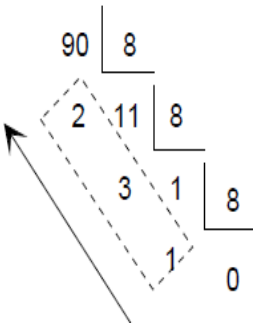


Figure 2.2 : conversion de $90_{(10)}$ en base 8.

$90_{(10)} = 132_{(8)}$.

2.2.2 Nombres fractionnaires

Pour convertir un nombre fractionnaire de la base 10 vers une autre base, il faut procéder en deux étapes :

- On convertie la partie entière du nombre comme indiqué précédemment ;
- On convertie la partie fractionnaire du nombre par multiplications successives : on multiplie successivement la partie fractionnaire par la base voulue, en retenant les parties entières qui apparaissent au fur et à mesure.

Exemple 1 : conversion de $N_{(10)}$ = 12,925 en base 2.

- partie entière : $12_{(10)} = 1100_{(2)}$.
- partie fractionnaire :

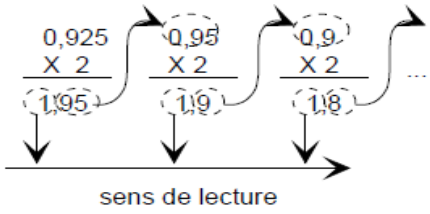


Figure 1.3 : conversion de $0{,}925_{(10)}$ en base 2.

Donc : $0{,}925_{(10)} = 0{,}111\ldots._{(2)}$.

2.3 Base 2 vers base 2^n

L'utilisation des bases 2^n (8 et 16 ….) permet de réduire le nombre de symboles à écrire tout en conservant la possibilité de conversion instantanée en binaire.

Pour convertir un nombre de la base 2 vers la base 2^n , il suffit de regrouper les bits par groupes de *n bits* (3 bits pour la base octale et 4 bits pour la base hexadécimale).

Exemple 1 : conversion de $N_{(2)}$ = 1100111010101 en base 8 puis 16.

$$Base\ 8 :\ N = 1\ 100\ 111\ 010\ 101_{(2)} = \underbrace{001}_{1_{(8)}}\ \underbrace{100}_{4_{(8)}}\ \underbrace{111}_{7_{(8)}}\ \underbrace{010}_{2_{(8)}}\ \underbrace{101}_{5_{(8)}} = 14725_{(8)}.$$

$$Base\ 16 :\ N = 1\ 1001\ 1101\ 0101_{(2)} = \underbrace{0001}\ \underbrace{1001}\ \underbrace{1101}\ \underbrace{0101} = 19D5_{(16)}.$$

Exemple 2 : conversion de $N_{(2)}$ = 110100110,101101 en base 8 puis 16.

$$Base\ 8 :\ N = 110\ 100\ 110{,}101\ 101_{(2)} = \underbrace{110}_{6_{(8)}}\ \underbrace{100}_{4_{(8)}}\ \underbrace{110}_{6_{(8)}}{,}\underbrace{101}_{5_{(8)}}\ \underbrace{101}_{5_{(8)}} = 646{,}55_{(8)}$$

$$Base\ 16 :\ N = 1\ 1010\ 0110{,}1011\ 01_{(2)} = \underbrace{0001}_{1_{(16)}}\ \underbrace{1010}_{A_{(16)}}\ \underbrace{0110}_{6_{(16)}}{,}\underbrace{1011}_{B_{(16)}}\ \underbrace{0100}_{4_{(16)}} = 1A6{,}B4_{(16)}$$

2.4 Base 2^n vers base 2

Pour la conversion inverse, il suffit de développer chaque symbole de la représentation dans la base 2^n sur *n* bits.

Exemple 1 :

$$4A1_{(16)} = \underbrace{0100}_{4\text{ en base 2}}\ \underbrace{1010}_{A\text{ en base 2}}\ \underbrace{0001}_{1\text{ en base 2}} = 010010100001_{(2)}.$$

Exemple 2 :

$$273{,}15_{(8)} = 010\ 111\ 011{,}001\ 101 = 10111011{,}001101_{(2)}$$

2.5 Représentation binaire des nombres entiers signés et non signés

Les nombres entiers peuvent être représentés en base deux par un vecteur de n bits,le poids du bit d'indice i est 2^i.

En représentation non signée, la gamme de valeurs avec N bits est :

$$0 \leq n \leq 2^N - 1.$$

En représentation signée, la gamme de valeurs avec N bits est :

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

Aux systèmes binaires les nombres entiers signés sont présentés par le complément à 2 (vrai) de ce nombre binaire, ou par la représentation module + signe.

2.5.1 Représentation des nombres entiers signés en complément à 2

Dans les ordinateurs, le complément à 2 est le mode de représentation le plus utilisé en arithmétique binaire pour coder les nombres entiers.

Les nombres positifs se représentent par leur valeur binaire naturelle. Par exemple +3 est représenté par 0000 0011 sur un format de 8 bits, mais les nombres négatifs s'obtient comme suit :

- On part de la représentation binaire naturelle de l'opposé arithmétique du nombre à coder (nombre positif),
- On calcule son complément à 1 (CA1) ou complément restreint. Celui-ci est obtenu en inversant tous ses bits,
- On en déduit son complément à 2 (CA2) ou complément vrai en ajoutant 1 au niveau

du LSB.

Exemple : représentation du nombre (-5) en CA2 sur un format de 8 bits.

- Représentation binaire naturelle de +5 : 5 = 0000 0101,
- CA1 de +5 : $\overline{5}$ = 1111 1010,
- CA2 de +5 : −5 = 1111 1011.

On identifie le CA2 d'un nombre à son opposé arithmétique car :

$A + (-A) = 2^n = 0 \bmod 2^n$, si n est le format de représentation du nombre A. En effet, soit : $A = a_{0n-1} \ldots a_1 a_0$, alors $\bar{A} = \overline{a_{n-1}} \ldots \overline{a_1}\ \overline{a_0}$, et donc :

$A + \bar{A} = 2^n - 1, et\ (-A) = \bar{A} + 1.$

La représentation en complément à 2 présente les caractéristiques suivantes :

• Le principe d'obtention de l'opposé d'un nombre négatif est le même que celui permettant d'obtenir l'opposé d'un nombre positif,

• Le nombre 0 a une représentation unique,

• Un format sur n bits permet de coder en CA2 les nombres N vérifiant

$$-2^{n-1} \leq N \leq +2^{n-1} - 1.$$

Exemple : pour n = 4

$N_{(10)}$	$N_{(2)}$	$\overline{N}_{(2)}$	$-N_{(2)}$	$-N_{(10)}$
0	0000	1111	0000	0
1	0001	1110	1111	-1
2	0010	1101	1110	-2
3	0011	1100	1101	-3
4	0100	1011	1100	-4
5	0101	1010	1011	-5
6	0110	1001	1010	-6
7	0111	1000	1001	-7
			1000	-8

Tableau 2.1 : représentation en complément à 2 sur 4 bits.

2.5.2 Représentation module + signe

Il s'agit d'une représentation parfois utilisée car plus simple que celle du CA2, mais qui est moins bien adaptée aux opérations arithmétiques.

Dans cette représentation, le bit de poids le plus fort représente le signe (MSB = 0 => nombre positif, MSB = 1 => nombre négatif), et les autres bits la valeur absolue du nombre. Ainsi, un format de n bits permet de coder les nombres compris entre :

$-(2^{n-1} - 1)\ et\ 2^{n-1} - 1.$

Exemple : pour $n = 4$

$N_{(10)}$	$N_{(2)}$	$-N_{(2)}$	$-N_{(10)}$
0	0000	1000	0
1	0001	1001	-1
2	0010	1010	-2
3	0011	1011	-3
4	0100	1100	-4
5	0101	1101	-5
6	0110	1110	-6
7	0111	1111	-7

Tableau 2.2 : représentation "module + signe" sur 4 bits.

2.6 Les types des codes binaires

Pour les codes binaires, on trouve les codes binaires numériques pondérés, les codes binaires numériques non pondérés et les codes alphanumériques.

2.6.1 Codes pondérés :

Si la position de chaque symbole dans chaque mot correspond à un poids fixé, on peut dire que ce code est pondéré: par exemple 1, 10, 100, 1000 ... pour la numération binaire, et 1, 2, 4, 8, ... pour la numération décimale.

Les codes pondérés sont : le code binaire pur et ses dérivés comme le code octal et le code hexadécimal, le code BCD…etc

2.6.1.1 Le code DCB (Décimal Codé Binaire)

Dans le code DCB ou BCD (Binary-Coded Decimal) chaque chiffre d'un nombre décimal (de 0 à 9) est codé à l'aide de 4 bits (de $0000_{(2)}$ à $1001_{(2)}$). Ainsi le code BCD n'utilise que 10 mots de codes de 4 bits.

Exemple :

$1995_{(10)} = (0001\ 1001\ 1001\ 0101)_{(BCD)}$.

2.6.2 Codes non pondérés

Dans le cas des codes non pondérés, il n'y a pas de poids affecté à chaque position des symboles. On convient simplement d'un tableau de correspondance entre les objets à coder et une représentation binaire.

2.6.2.1 Code excédent 3

Le code excédent 3 est un code non pondérés qui utilise, tout comme le code BCD, 10 mots de codes, auxquels on fait correspondre les 10 chiffres décimaux.

N°	Code excédent 3
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Tableau 2.3 : code excédent 3.

2.6.2.2 Code binaire réfléchi ou code de Gray

Ce code numérique n'étant pas pondéré, il est peu employé pour les opérations arithmétiques.

N°	Code de Gray
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0

Tableau 2.4 : c ode de gray pour 4 bits.

2.6.3 Codes alphanumériques

Plus les codes numériques il y'a des codes alphanumériques. Le code ASCII (American Standard Code for Information Interchange) est parmi les codes alphanumériques les plus connu, il contient 128 combinaisons (lettres, chiffres, signes de ponctuation, caractères de contrôle,… etc.) qui sont codées sur 7 bits, mais les transmissions de données s'effectuant souvent sur un format de 8 bits, le dernier bit est utilisé pour le contrôle de parité du message.

3. Opérations arithmétiques et soustraction

3.1 L'addition arithmétique

3.1.1 Addition de nombres binaires non signés

Pour l'addition de nombres binaires non signés, il faut d'abord choisir le nombre de bits à utiliser pour représenter les deux nombres.

Le principe de l'addition est dans toutes les bases similaires à celui de l'addition décimale : on additionne bit par bit en partant des poids faibles, et en propageant éventuellement une retenue.

La somme des nombres peut nécessiter un bit de plus que celui des opérandes.

Exemple :

9	1001
+ 2	0010
11	1011

9	00001001
+ 182	10110110
191	10111111

9	1001
+ 7	0111
16	10000

170	10101010
+ 182	10110110
352	101100000

Dans cet exemple le résultat de la somme pour les deux dernières opérations demande un bit de plus par rapport aux opérandes.

3.1.2 Addition de nombres binaires signés en complément à 2

Pour les nombres binaires signés, si le format des nombres est fixe, le résultat de l'addition peut donner lieu à un dépassement de capacité (overflow *OVF*), C'est pour cela qu'il faut d'abord choisir le nombre de bits à utiliser pour représenter les deux nombres, en faisant l'extension du signe correctement.

L'addition se fait bit par bit et on laisse tomber toute retenue finale.

Il ne peut pas y avoir de dépassement de capacité ou de débordement si les deux nombres ont un signe différent, mais un débordement peut se produire si les deux nombres ont le même signe et que la somme a un signe différent.

Pour garantir qu'un débordement n'aura pas lieu, on ajoute un bit aux opérandes avant de faire l'addition.

Exemple :

7	0111
+ (-2)	1110
5	1 0101

-6	1010
+ 7	0111
1	1 0001

Dans cet exemple Il n'ya pas de débordement car les deux nombres ont un signe différent.

Exemple :

7	0111
+ 7	0111
14	0 1110

(a) résultat incorrect

7	0 0111
+ 7	0 0111
14	00 1110

(b) : résultat incorrect

Dans cet exemple lorsque les deux nombre sont représentés sur 4 bits, le résultat est incorrect, car le résultat a un signe différent des opérandes. . (voir (a))

Mais pour avoir un résultat correct il faut ajouter un bit de plus aux deux opérandes. (voir (b))

Pour plus d'exemples sur l'addition de nombres binaires signés, voici les opérations suivantes :

```
  0000 0110 (+6)
+ 0000 0100 (+4)
-----------------
  0000 1010  (+10)
  OVF=0
  C=0
```

résultat correct

```
  0111 1111 (+127)
+ 0000 0001 (+1)
-----------------
  1000 0000  (-128)
  OVF=1
  C=0
```

résultat incorrect

```
  0000 0100 (+4)
+ 1111 1110 (-2)
-----------------
1 0000 0010  (+2)
  OVF=0
  C=1 ignoré
```

résultat correct

```
  0000 0100 (+4)
+ 1111 1100 (-4)
-----------------
1 0000 0000  (0)
  OVF=0
  C=1 ignoré
```

résultat correct

3.2 La soustraction arithmétique

3.2.1 Soustraction de nombres binaires signés en complément à 2

En arithmétique binaire, la soustraction est appliquée sur des nombres signés. Dans ce cas, cette opération se ramène dans tous les cas à une addition.

Une approche simple consiste à procéder comme pour l'addition, mais on change le signe du nombre à soustraire en inversant ses bits et en ajoutant 1.

Un débordement ou un dépassement de capacité (overflow *OVF*) peut avoir lieu. Les règles de détection sont similaires à celle de l'addition.

On garantit toujours qu'un débordement n'aura pas lieu en ajoutant un bit aux opérandes avant de faire la soustraction.

Exemples :

7	0111
- 2: inv(2)	1101
	1
5	1 0101

résultat correct

7	0111
- (-2): inv(-2)	0001
	1
9	0 1001

résultat correct

Dans cet exemple les deux opérandes ont deux signes différents, c'est pour cela que les résultats de la soustraction sont corrects.

Chapitre 3 : Circuits combinatoires de transcodeurs.

3.1 Introduction

Un circuit numérique réalisant une fonction d'un operateur combinatoire est un circuit combinatoire.

Plus les opérateurs élémentaires cités au chapitre 1, on distingue comme opérateurs combinatoires standard :

- les opérateurs de transcodage,
- les opérateurs d'aiguillage (de multiplexage/démultiplexage),
- les opérateurs de comparaison,
- les opérateurs arithmétiques.

3.2 Les opérateurs de transcodage

Comme opérateurs de transcodage on trouve les codeurs, décodeurs et les convertisseurs

3.2.1 Le codeur : c'est un opérateur qui interprète une information dans un code donné.

Exemple : Codeur binaire 8 vers 3

Pour ce codeur une seule entrée doit être active à la fois.

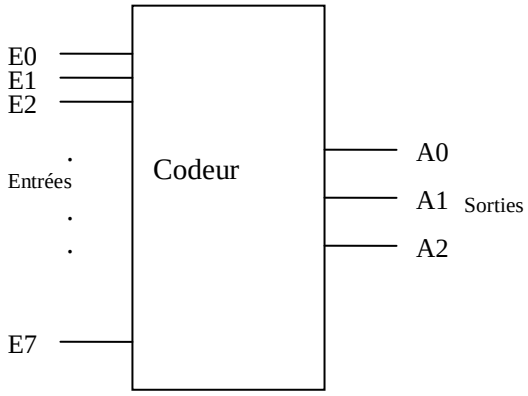


Figure 3.1 : codeur 8 vers 3

Sa table de vérité est :

Entrées	Sorties		
	A2	**A1**	**A0**
E0	**0**	**0**	**0**
E1	**0**	**0**	**1**
E2	**0**	**1**	**0**
E3	**0**	**1**	**1**
E4	**1**	**0**	**0**
E5	**1**	**0**	**1**
E6	**1**	**1**	**0**
E7	**1**	**1**	**1**

Tableau 3.1 : Table de vérité du codeur 8 vers 3.

A la sortie du codeur on aura les équations suivates :

A0 = E1+E3+E5+E7 ;

A1 = E2+E3+E6+E7 ;

A2 = E4+E5+E6+E7 ;

3.2.2 Le décodeur : c'est un opérateur qui permet d'extraire une ou des informations à partir d'un code donné. On générale le circuit du décodeur a le schéma suivant :

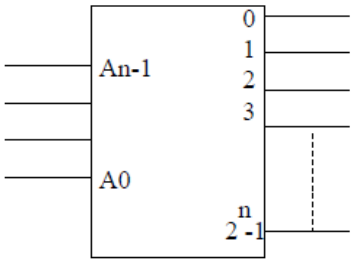


Figure 3.2 : décodeur binaire n vers 2^n (1 parmi 2^n).

Exemple : Décodeur 3 vers 8

Entrées			Sorties							
C	B	A	S0	S1	S2	S3	S4	S5	S6	S7
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
1	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Tableau 3.2 : table de vérité du décodeur 3 vers 8.

Les sorties du décodeur sont données par les relations suivantes :

$S0 = \overline{C}\,\overline{B}\,\overline{A}$;

$S1 = \overline{C}\,\overline{B}A$;

.

.

.

$S7 = CBA$.

3.2.3 Les convertisseurs (transcodeurs)

Ces opérateurs sont des transcodeurs permettent de convertir un nombre écrit dans un code C1 vers un autre code C2.

Exemple : Le convertisseur Binaire/Gray

N° =	Code Binaire				Code Gray			
	B_4	B_3	B_2	B_1	G_4	G_3	G_2	G_1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Tableau 3.3 : passage du code binaire au code Gray sur 4 bits.

Convertisseur binaire/Gray

$G_i = B_i \oplus B_{i+1}$ pour $i = 1,.....,n-1$

Et

$G_n = B_n$.

3.3 La fiche Technique du circuit intégré du décodeur 3 vers 8 (74HCT138)

3.3.1 Configuration du circuit intégré du décodeur

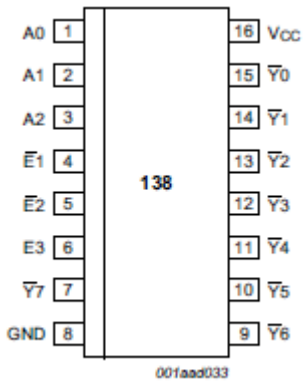


Figure 3.3 : Configuration des broches.

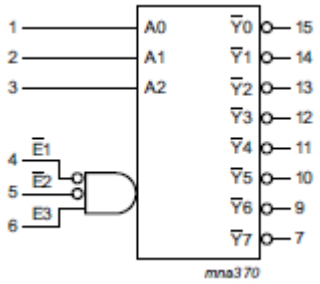


Figure 3.4: symbole logique.

3.3.2 Description des broches du circuit intégré du décodeur

Symbole	Broches	Description des broches
A0, A1, A2	1, 2, 3	Entrées A0, A1, A2
$\overline{E1}, \overline{E2}$	4, 5	Entrées de validation $\overline{E1}, \overline{E2}$ (active Bas)
E3	6	Entrées de validation E3 (active Haut)
$\overline{Y0}, \overline{Y1}$ ……, $\overline{Y7}$	15, 14, 13, 12, 11, 10, 9, 7	Sorties $\overline{Y0}, \overline{Y1}$, ……, $\overline{Y7}$ (active Bas)
GND	8	Masse (0 V)
VCC	16	Tension d'alimentation positive

3.3.3 Le schéma logique du circuit intégré du décodeur

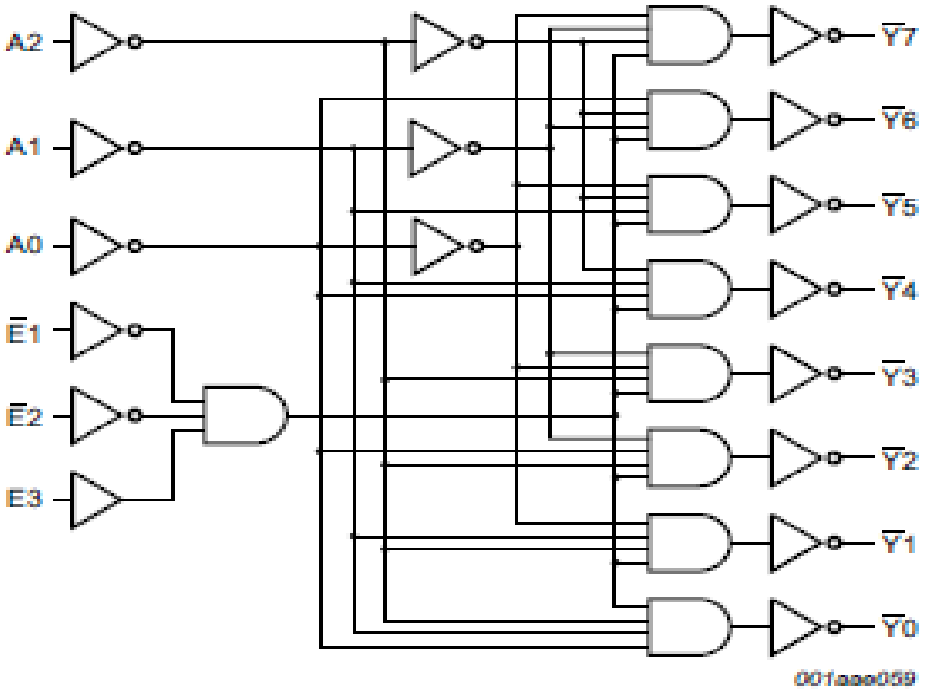


Figure 3.5 : Schéma logique du décodeur.

Chapitre 4 : Circuits combinatoires aiguilleurs.

4.1 Introduction

Les circuits de multiplexage/démultiplexages sont des circuits d'aiguillage qui sont capable de faire la conversion parallèle/série de l'information (multiplexage) ou l'inverse c'est-à-dire la conversion série/parallèle de l'information binaire (démultiplexage).

4.2 Le multiplexeur

C'est un circuit combinatoire qui peut orienter des informations provenant de *N* canaux vers un seul canal.

ce circuit combinatoire à N = 2^n entrées de données,D_0 , $D_1, D_2, \ldots . D_{N-1}$, n entrées d'adresse $A_{n-1}, A_{0n-2}, \ldots \ldots A_0$, et une sortie S sans oublié une entrée de validation. Le multiplexeur peut, à l'aide de n bits d'adresse de sélectionner une de ses 2^n entrées et la dirigé vers la sortie.

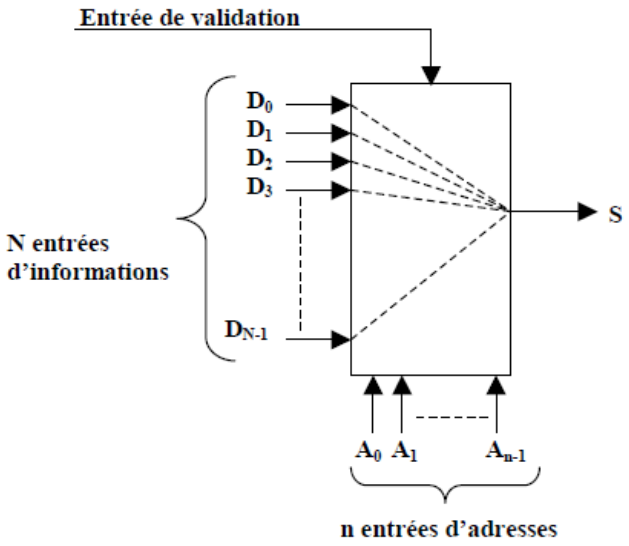


Figure 4.1 : multiplexeur 2^n vers 1

Où la sortie du multiplexeur est : $S = \sum_{i=0}^{2^n-1} m_i D_i$, et m_i est le $i^{ème}$ minterme des variables $A_{n-1}, A_{0n-2}, \ldots \ldots A_0$.

Exemple :

$m_0 = \overline{A_{n-1}} \ldots\ldots\ldots \overline{A_2}\,\overline{A_1}\,\overline{A_0}$, $m_1 = \overline{A_{n-1}} \ldots\ldots\ldots \overline{A_2}\,\overline{A_1}A_0$, ……..

Exemple : Multiplexeur 4 vers 1(voire Figure 4.2).

Ce multiplexeur dispose de :

- 4 entrées E0, E1, E2 et E3
- 2 entrées d'adresse A0 et A1
- 1 sortie S
- 1 entrée de validation V.

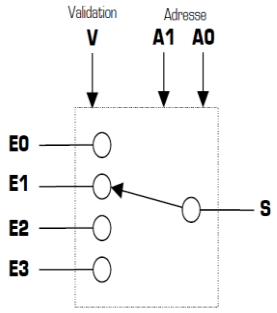


Figure 4.2 : multiplexeur 4 vers 1.

S = V $\overline{A1}$ $\overline{A0}$ E0 + V $\overline{A1}$ A0 E1 + V A1 $\overline{A0}$ E2 + V A1 A0 E3.

Et puisque : V=1, alors on aura :

S = $\overline{A1}$ $\overline{A0}$ E0 +$\overline{A1}$A0 E1+A1 $\overline{A0}$ E2 +A1A0 E3.

4.3 Le démultiplexeur

Contrairement au multiplexeur le démultiplexeur ne possède qu'une seule entrée de donnée dont la valeur est dirigée vers une sortie parmi les N=2^n sorties.

La donnée présente sur l'entrée du démultiplexeur sera dirigée vers une sortie déterminée par une adresse codée par n entrées d'adresses.

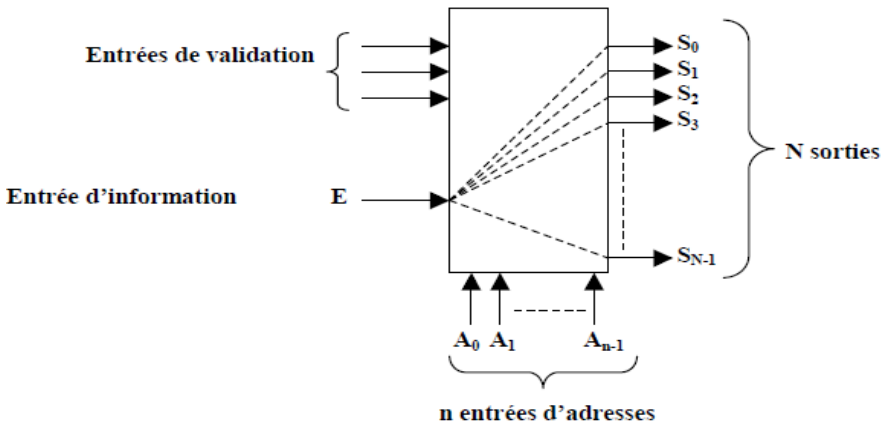


Figure 4.3 : Démultiplexeur 1 vers 2^n.

Exemple : Démultiplexeur 1 vers 4 qui dispose de :

- 1 entrée de donnée E,

- 4 sorties S0, S1, S2 et S3,
- 2 entrées d'adresse A0 et A1,
- 1 entrée de validation V.

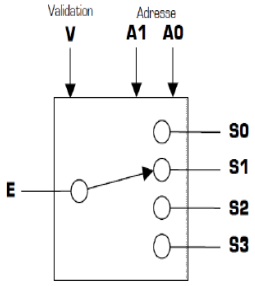


Figure 4.4: Démultiplexeur 1 vers 4.

Les équations des sorties :

S0 = V $\overline{A1}$ $\overline{A0}$ E;

S1 = V $\overline{A1}$ A0 E;

S2 = V A1 $\overline{A0}$ E Et

S3 = VA1A0 E

Avec V=1, on aura donc:

S0 = $\overline{A1}$ $\overline{A0}$ E;

S1 = $\overline{A1}$ A0 E;

S2 = A1 $\overline{A0}$ E et

S3 =A1A0 E.

4.4 La fiche Technique du circuit intégré du multiplexeur 2 vers 1 (74LVC1G157-Q100)

4.4.1 Configuration du circuit intégré du multiplexeur

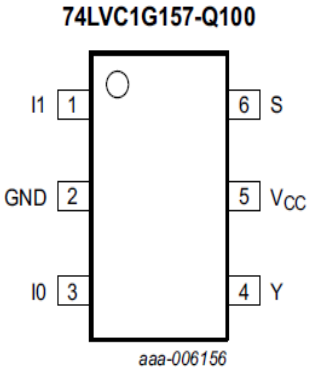


Figure 4.5 : Configuration des broches.

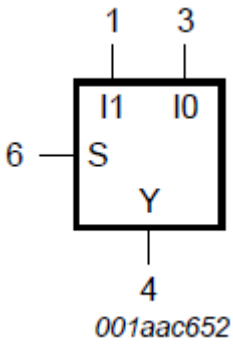


Figure 4.6: symbole logique.

4.4.2 Description des broches du circuit intégré du multiplexeur

Symbole	Broches	Description des broches
I1	1	Entrée de données 1
GND	2	Terre (0 V)
I0	3	Entrée de données 0
Y	4	Sortie du multiplexeur
VCC	5	Tension d'alimentation
S	6	Entrée de sélection de données commune

4.4 .3 Le schéma logique du circuit intégré du multiplexeur

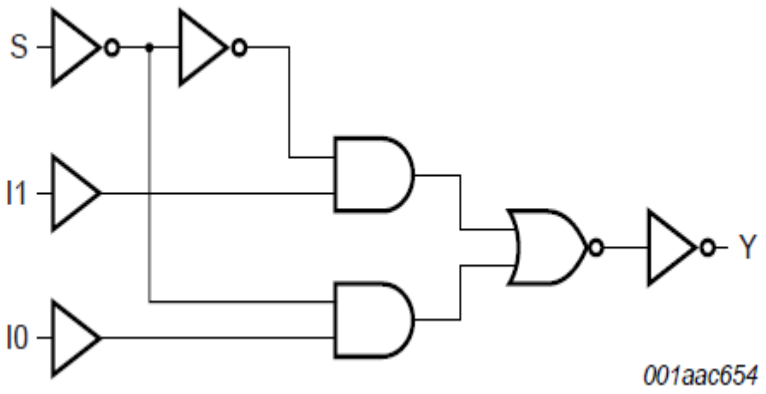


Figure 4.7 : Schéma logique du multiplexeur.

Chapitre 5 : Circuits combinatoires de comparaison.

5.1 Introduction

Dans la logique combinatoire nous avons besoin d'avoir des operateurs qui peuvent faire la comparaison entre les informations binaire, c'est opérateurs s'appellent les comparateurs.

5.2 Les comparateurs

Un comparateur logique est un circuit combinatoire qui effectue la comparaison entre 2 nombres binaires généralement notés A et B. Il possède 3 sorties notées :
(A=B), (A>B) et (A< B).

Exemple : Comparaison entre deux éléments binaire A et B de 1 bit.

A	B	$A = B$	$A > B$	$A < B$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

Tableau 5.1 : Table de vérité d'un comparateur de 1 bit de A et B.

Où les sorties du comparateur sont :

$$(A = B) = \overline{A \oplus B}$$
$$(A > B) = A\overline{B}$$
$$(A < B) = \overline{A}B$$

Exemple : Comparateur de deux mots de 4 bits A et B.

Le circuit de référence (XX 85) (figure 5.1) représente un comparateur de deux mots de 4 bits ou plus.

Plus les entrées de données des deux mots à comparer, le comparateur de référence (XX 85) possède trois entrées (A>B)in ,(A=B)in) et (A<B)in , permettant de cascader les comparateurs pour pouvoir comparer des nombres de plus de 4 bits. Si le comparateur est utilisé seul, les entrées (A>B)in , (A=B)in et (A<B)in doivent être connectées respectivement à 0, 1, et 0.

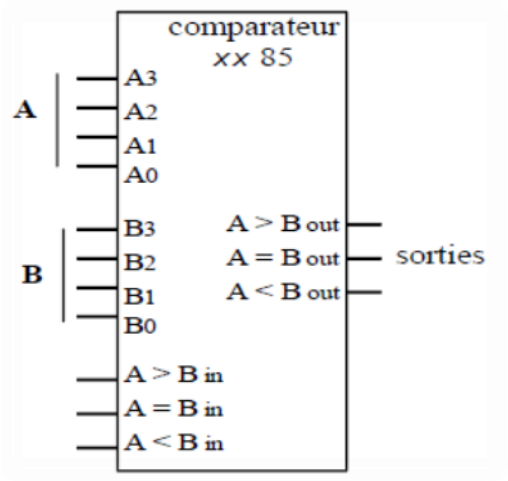


Figure5.1: comparateur complet 4 bits *XX* 85.

Entrées de données				Entrées de mise en cascade			Sorties		
A_3 , B_3	A_2 , B_2	A_1 , B_1	A_0 , B_0	$A > B_{in}$	$A = B_{in}$	$A < B_{in}$	$A > B_{out}$	$A = B_{out}$	$A < B_{out}$
$A_3 > B_3$	X	X	X	X	X	X	1	0	0
$A_3 < B_3$	X	X	X	X	X	X	0	0	1
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	1	0	0
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	1	0	0	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	0	1	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0	0	1	0

Tableau 5.2 : table de vérité du comparateur de 4 bits *XX* 85.

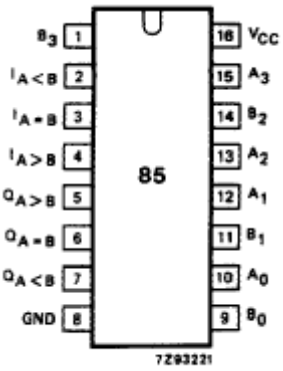
Remarque

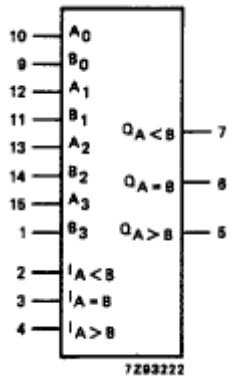
Pour comparer deux nombres de 8 bits, il suffit de relier les sorties *(A>B)out*,*(A=B)out* , et *(A<B)out* du comparateur qui possède les 4 bits de poids faibles aux entrées *(A>B)in*,*(A=B)in* et *(A<B)in* du comparateur qui possède les 4 bits de poids forts. Dans ce cas, les valeurs logiques 010 sont appliquées respectivement sur les entrées *(A>B) in*, *(A=B)in* et *(A<B)in* du premier comparateur.

5.3 La fiche Technique du circuit intégré du comparateur 74HC/HCT85

(comparateur 4 bit 74HC/HCT85)

5.3.1 Configuration du circuit intégré du comparateur





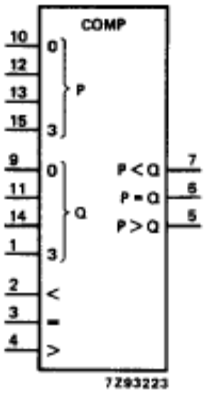


Figure 5.2 : Configuration des broches. Figure 5.3: symbole logique 1. Figure 5.4: symbole logique 2.

5.3.2 Description des broches du circuit intégré du comparateur

Broches	Symbole	Nom et fonction
2	$I_{A<B}$	entrées de mise en cascade (A < B)
3	$I_{A=B}$	entrées de mise en cascade (A = B)
4	$I_{A>B}$	entrées de mise en cascade (A > B)
5	$Q_{A>B}$	sortie (A > B)
6	$Q_{A=B}$	sortie (A = B)
7	$Q_{A<B}$	sortie (A < B)
8	GND	Masse (0 V)
9, 11, 14, 1,	B_0 à B_3	Entrées du mot B
10, 12, 13, 15	A_0 à A_3	Entrées du mot A
16	V_{CC}	Tension d'alimentation positive

5.3.3 Le schéma logique du circuit intégré du comparateur

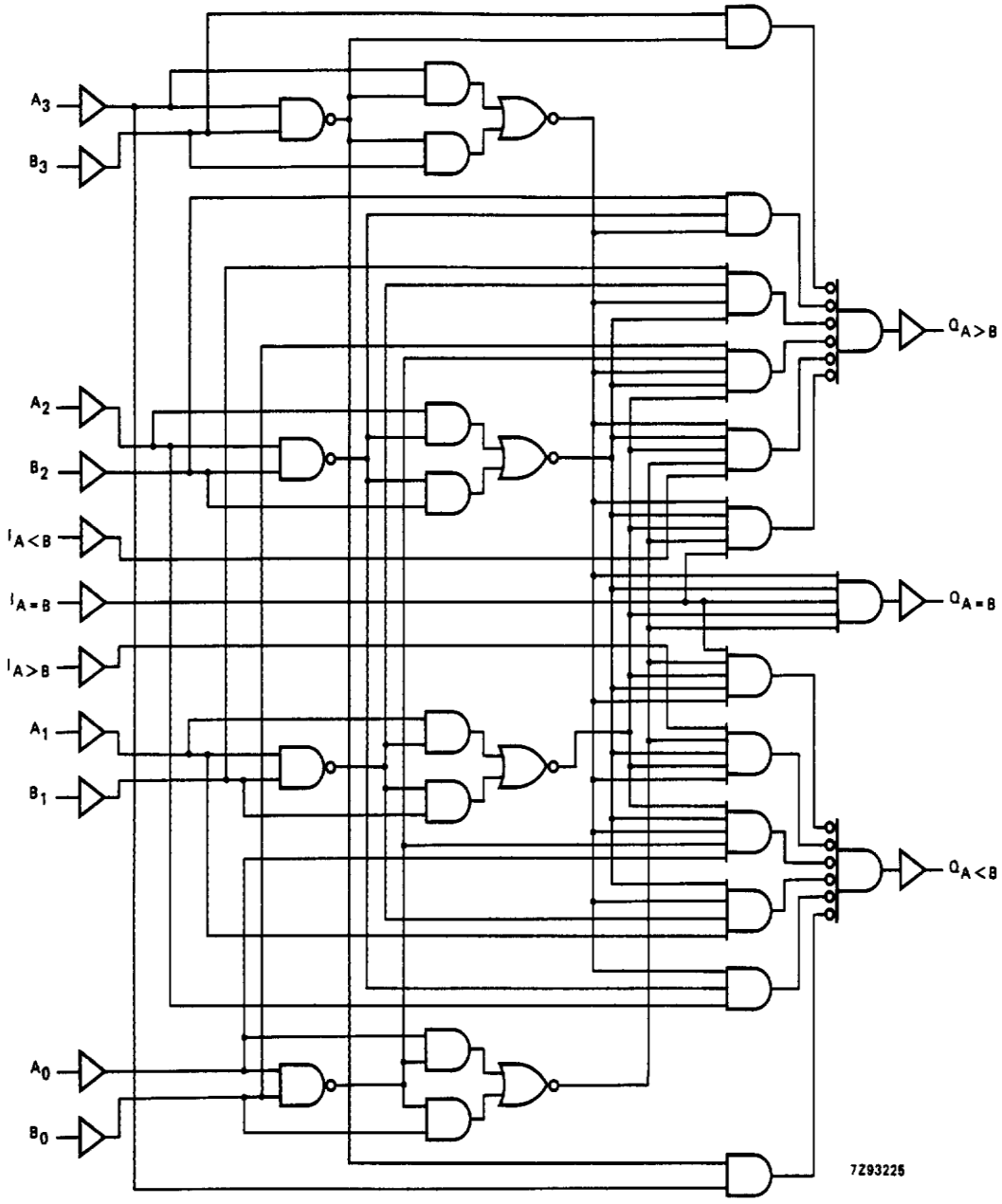


Figure 5.5 : Schéma logique du comparateur.

Chapitre 6 : Circuits arithmétiques additionneurs et soustracteurs.

6.1 Introduction

Les additionneurs et les soustracteurs sont les operateurs fondamentaux pour faire les opérations arithmétiques d'addition, soustraction, multiplication et même division.

6.2 Les additionneurs

Toutes opérations d'additions de deux mots binaires, se réalise par l'operateur additionneur. L'addition est la fonction arithmétique la plus couramment rencontrée dans les systèmes numériques.

6.2.1 Le demi-additionneur

Le demi-additionneur fait l'addition de deux bits binaire A_k et B_k sans retenue mais à la sortie on aura la somme S_k avec la retenue C_k.

A_k	B_k	S_k	C_k
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tableau 6.1 : table de vérité du demi-additionneur binaire.

Où $S_k = A_k \oplus B_k$ et $C_k = A_k . B_k$.

Le circuit logique d'un demi-additionneur est :

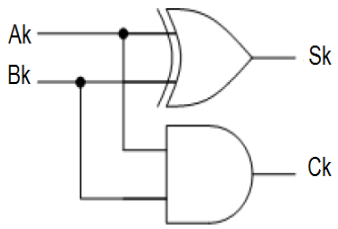


Figure 6.1 : Demi-additionneur.

6.2.2 L'additionneur complet

L'additionneur complet est l'élément de base pour additionner deux nombres binaires à plusieurs bits où les bits de même poids seront additionnés successivement avec la retenue de l'addition précédente C_{k-1} c'est-à-dire qu'il prend en compte une retenue entrante.
(voir figure 6.2),

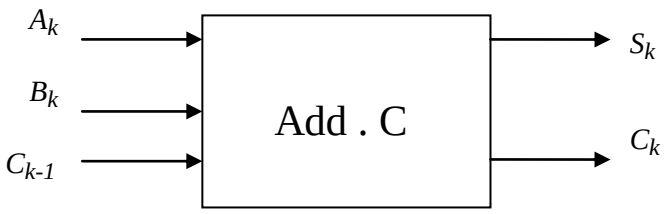


Figure 6.2 : Additionneur complet.

La table de vérité de l'additionneur complet est le suivant :

A_k	B_k	C_{k-1}	S_k	C_k
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

Tableau 6.2 : table de vérité de l'additionneur complet.

Les équations logiques des sorties sont :

$$S_k = A_k \oplus B_k \oplus C_{k-1}$$

Et

$$C_k = A_k B_k + (A_k + B_k) C_{k-1} = A_k B_k + (A_k \oplus B_k) C_{k-1}.$$

Exemple :

Additionneur de deux mots A (A_3, A_2, A_1, A_0) et B (B_3, B_2, B_1, B_0) de 4 bits qui est réalisé par l'association de 4 additionneur de 1 bit :

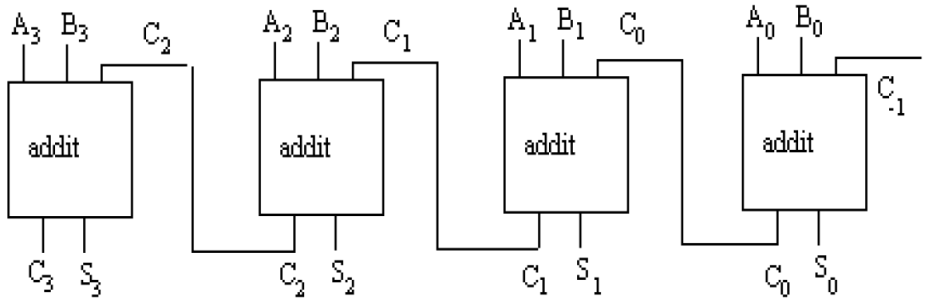


Figure 6.3 : Additionneur de deux mots A et B de 4 bits.

On peut réaliser l'additionneur complet à partir de deux demi-additionneurs et d'un opérateur OU (figure 6.4).

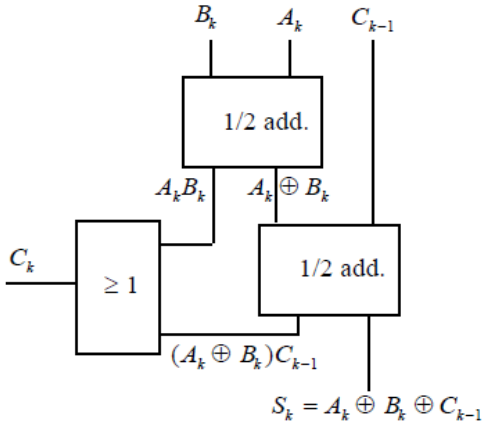


Figure 6.4 : une réalisation de l'additionneur complet.

Et en générale en peut additionner deux nombres binaires de n bits par l'additionneur suivant :

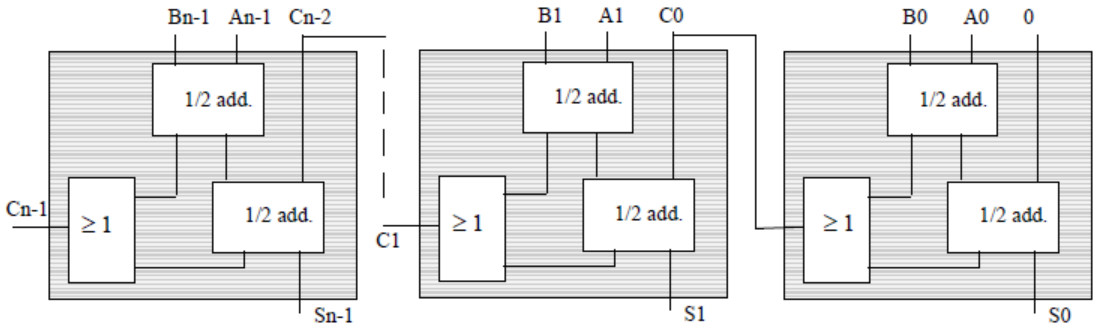


Figure 6.5 : Additionneur de n bits à retenue propagée.

6.3 Les soustracteurs

6.3.1 Demi-soustracteur

C'est un circuit qui peut faire la soustraction de deux nombre binaires de 1 bit chacun. Le circuit a deux entrées A_k, B_k et deux sorties, la différence D_k et la retenue C_k .

La table de vérité d'un demi-soustracteur est la suivante :

A_k	B_k	D_k	C_k
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tableau 6.3 : Table de vérité du demi-soustracteur.

Les équations de sorties sont :

$D_k = A_k \oplus B_k$ et $C_k = \overline{A_k} B_k$

6.3.2 Soustracteur complet

C'est un circuit capable de faire la soustraction de deux bits de rang k, (A_k - B_k) tout en tenant compte de la retenue C_{k-1} provenant de la soustraction des bits de rang directement inférieurs. On aura deux sorties D_k et C_k .

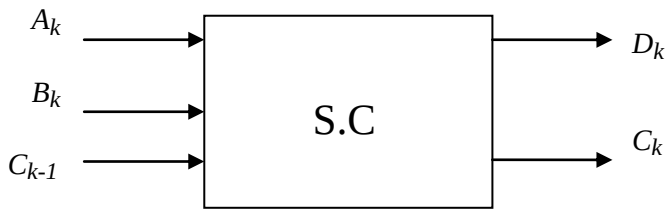


Figure 6.6 : Soustracteur complet.

A_k	B_k	C_{k-1}	D_k	C_k
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

Tableau 6.4 : Table de vérité du soustracteur complet.

Les équations de sorties sont :

$$D_k = A_k \oplus B_k \oplus C_{k-1}.$$

Et

$$C_k = \overline{A_k}\, B_k + (\overline{A_k} + B_k)\, C_{k-1} = \overline{A_k}\, B_k + \overline{(A_k \oplus B_k)}\, C_{k-1}.$$

Chapitre 7 : Les bascules

7.1 Introduction

Les bascules sont des circuits séquentielles où les sorties de ces circuits dépendent des entrées et de l'état précédent, c'est donc l'effet mémoire.

7.2 Les bascules

La Bascule est un point mémoire temporaire permettant de stocker des informations pouvant être annulées à tout moment.

Les bascules sont des circuits bistables, qui possèdent deux états stables "1"ou "0".

7.3 Bascule RS

7.3.1 Principe du Bascule RS.

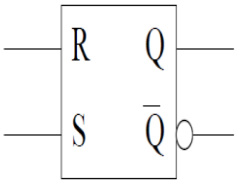


Figure 7.1 : Bascule RS.

- l 'entrée S (Set) est l 'entrée de mise à 1 ou mémorisation de l 'information reçue.
- l 'entrée R (Reset) est l 'entrée de mise à 0 ou l 'effacement de la mémoire.
- la sortie Q donne l'information mémorisée.
- la sortie $\bar{Q}$ correspond au complément de Q.

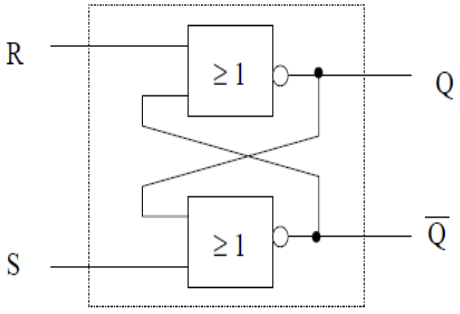


Figure 7.2 : Le logigramme du bascule RS avec portes NOR.

7.3.2 Fonctionnement de la bascule RS

D'après la table de vérité de la bascule RS :

.

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	X

L'équation de fonctionnement de cette bascule est :

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

7.3.3 Les chronogrammes

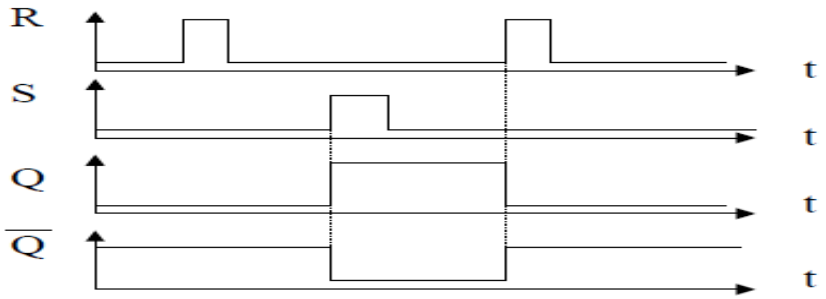


Figure 7.3 : Les chronogrammes de la bascule RS.

7.4 Les bascules synchrones

Les Bascules synchrones sont des circuits séquentiels où les entrées ne sont sensibles aux signaux appliqués que pendant l'activation par le signal d'horloge.

7.4.1 L'horloge

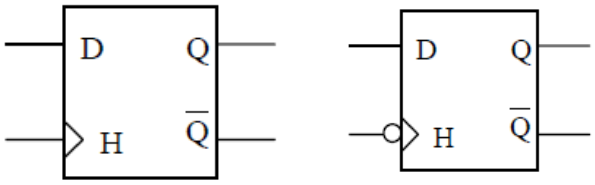
L'horloge H ou Ck (Clock) est une entrée sur laquelle est appliqué un signal carré de période définie :

 : Horloge active au niveau haut.

 : Horloge active au niveau bas.

7.4.2 Bascules synchronisées sur un front

Ces bascules font l'acquisition de la donnée et réalisent la commande sur un front d'horloge. Elles peuvent être actives sur le front montant ou sur le front descendant de l'horloge.



Front montant Front descendant

Figure 7.4: Front montant et front descendant.

La figure suivante montre le chronogramme du front montant et front descendant :

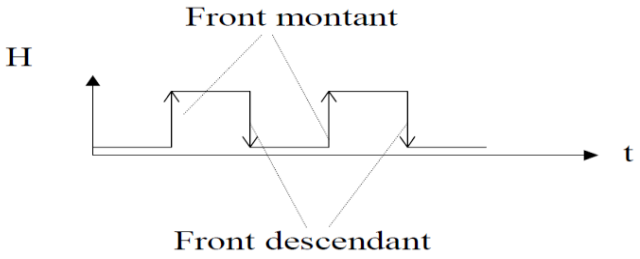


Figure 7.5 : Front montant et front descendant.

7.4.3 Les entrées de forçage Preset (P) et Clear (Cl)

Les entrées de forçage ou de pré positionnement permettant la mise à 1 ou à 0 quel que soit l'état de la bascule.

Preset (P) : mise à 1 et Clear (Cl) : mise à 0, ces entrées sont prioritaires sur les entrées synchrones et elles sont souvent actives au niveau bas.

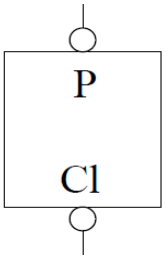


Figure 7.6 : Les entrées de forçage P et Cl.

7.5 Bascule maître-esclave

Cette bascule est constituée de deux bascules fonctionnant alternativement et commutant à des instants différents de l'horloge.

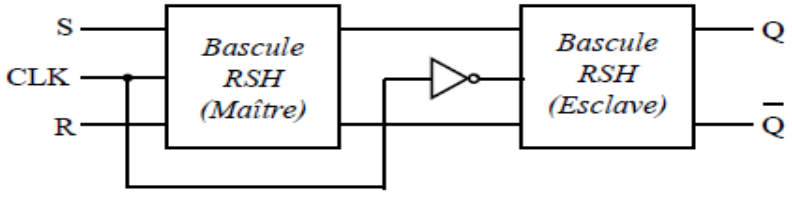


Figure 7.7 : Bascule maitre-esclave.

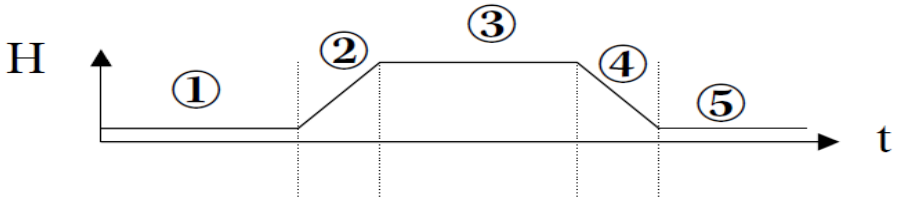


Figure 7.8:L'horloge de la bascule maitre-esclave.

(1) : Isolation de la bascule entière

(2) : Séparation du maître et de l 'esclave

(3) : Action des entrées sur le maître

(4) : Entrées maître verrouillées ; transfert maître vers esclave

(5) : Esclave recopie la sortie du maître

7.6 Bascule D

7.6.1 Bascule D (latch)

On peut dire que cette bascule est une bascule RS avec : $D = S = \bar{R}$.

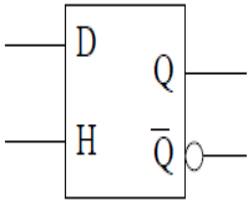


Figure 7.9 : Bascule D.

C'est une bascule de mémorisation.

H	D	Q_{t+1}	
0	X	Q_t	bloquée
1	0	0	$Q_{t+1} = D$
1	1	1	

$Q_{t+1} = DH + Q_t.\bar{H}$.

Si H = 1, on recopie l'entrée en sortie sinon on conserve en mémoire l'information.

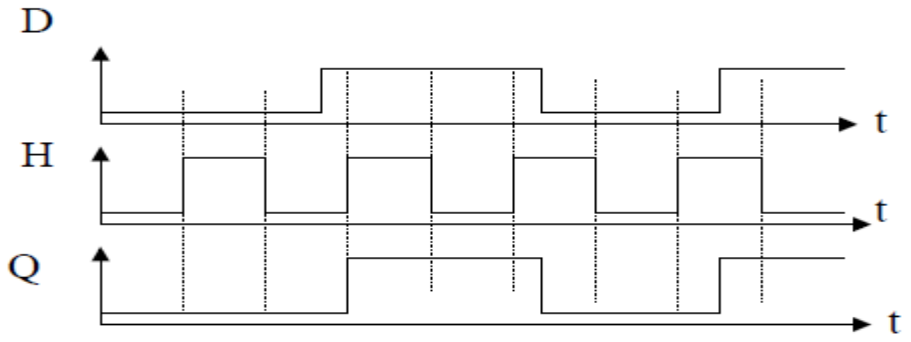


Figure 7.10:Le chronogramme de la bascule D.

Remarque : Les changements d'état ne sont autorisés que pour H = 1.

7.6.2 D à déclenchement sur front montant

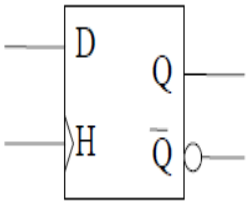


Figure 7.11 : Bascule D synchrone.

D	H	Q_{t+1}	
X	0	Q_t	
X	1	Q_t	mémoire
X	↓	Q_t	
0	↑	0	
1	↑	1	$Q_{t+1} = D$

Au front actif, la bascule enregistre l'information présente en entrée et la sortie prend le même état. En dehors du front actif, la bascule est bloquée et conserve l'état précédent.

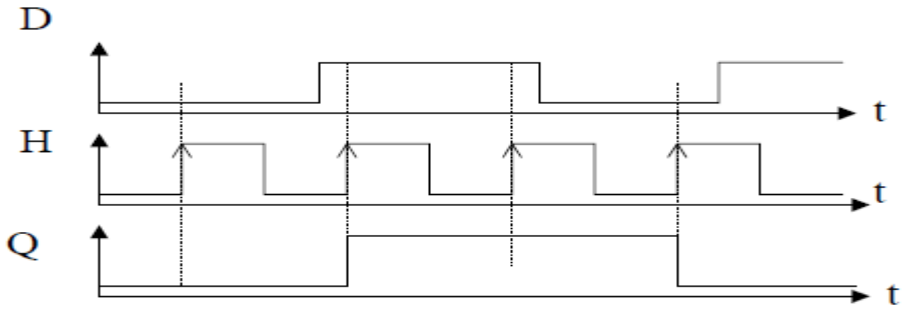


Figure 7.12 : Chronogramme du bascule D sur front montant.

7.7 Bascule T

La bascule T commute à chaque impulsion d'horloge si son entrée T est active, mais si son entrée T est inactive, elle conserve son état.

H	T	Q_{t+1}
0	X	Q_t
1	X	Q_t
↑	0	Q_t
↑	1	$\overline{Q}_t$

On peut obtenir la bascule T a partir de la bascule D :

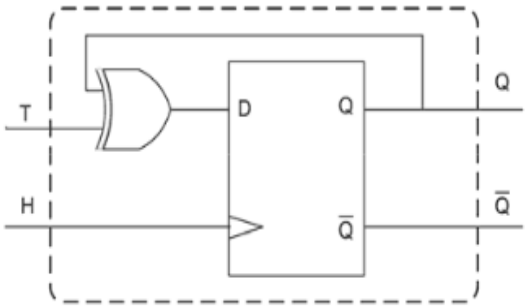


Figure 7.13 : Bascule T

7.8 Bascule JK

La bascule JK est une bascule RS plus développée et ne possédant plus d'indétermination car la combinaison (J,K) = (1,1) est maintenant applicable.

7.8.1 La Bascule JK à fonctionnement sur front montant

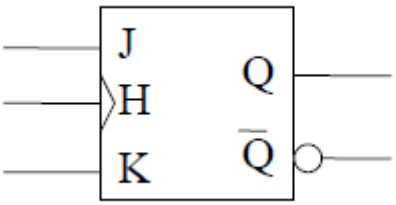


Figure 7.14 : Bascule JK

Sa table de vérité est la suivante :

H	J	K	Q_{t+1}	
0	X	X	Q_t	
1	X	X	Q_t	
↓	X	X	Q_t	mémoire
↑	0	0	Q_t	
↑	0	1	0	mise à 0
↑	1	0	1	mise à 1
↑	1	1	$\overline{Q_t}$	Inversion

Les chronogrammes de la bascule JK :

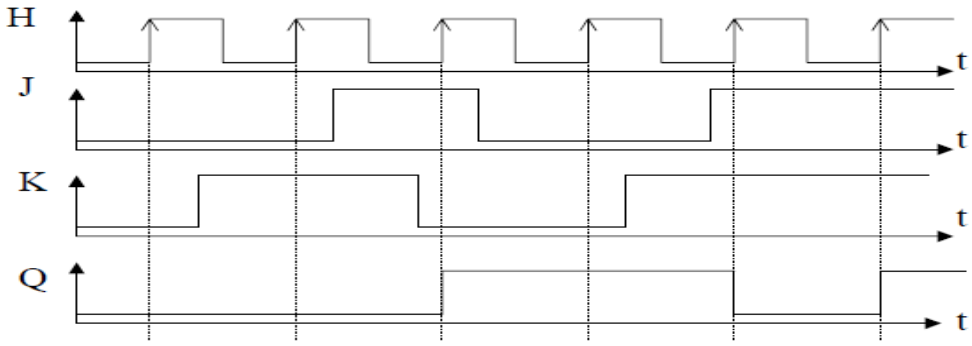


Figure 7.15 : Les Chronogrammes de la bascule JK.

7.9 Diviseur de fréquence

Le diviseur de fréquence est un élément de base du compteur asynchrone où Q oscille entre 0 et 1 à chaque front actif de l'horloge.

7.9.1 Diviseur de fréquence avec une bascule D

On relie $\bar{Q}$ à l'entrée D afin que la sortie Q change d 'état à chaque front d 'horloge actif (toggle).

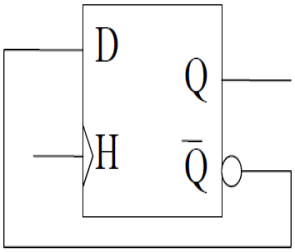


Figure 7.16 : Diviseur de fréquence.

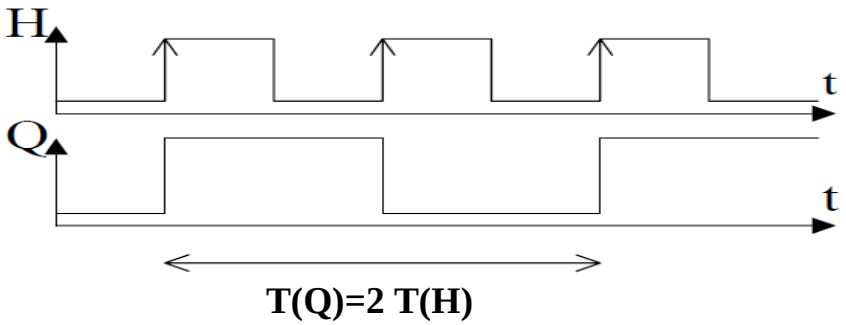


Figure 7.17:Chronogramme d'un diviseur de fréquence avec bascule D.

7.9.2 Diviseur de fréquence avec une bascule JK

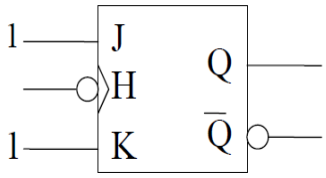


Figure 7.18:Bascule JK.

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

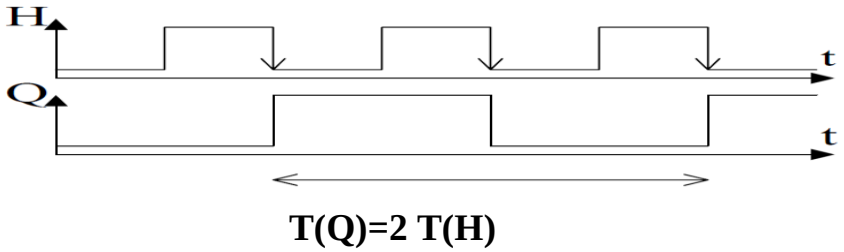


Figure 7.19:Chronogramme d'un diviseur de fréquence avec bascule JK.

Chapitre 8 : Les compteurs

8.1 Introduction

Une des applications des bascules est le comptage synchrone et asynchrone car les composants élémentaires des compteurs sont les bascules.

Un compteur est la correspondance univoque entre le nombre d'impulsions en entrée et l'état des sorties.

Le modulo d'un compteur est le nombre de combinaisons différentes obtenues en sortie de ce compteur.

Pour un compteur de n bits on aura au maximum 2^n combinaisons différentes.

8.2Caractéristiques générales des compteurs :

- commande d'horloge (synchrone ou asynchrone)
- capacité de comptage
- code de comptage
- vitesse de comptage
- comptage/décomptage
- possibilités de présélection.

8.3 Compteur asynchrone

C'est un compteur série qui propage en cascade l'ordre de changement d'états (horloge des bascules).

8.3.1 Compteur modulo 2^n à cycle complet

Principe : on place n bascules câblées en diviseur de fréquence en cascade en reliant :

– Q_i à l'horloge de la i+$1^{ème}$ bascule pour des bascules à front descendant

– $\overline{Q_i}$ à l'horloge de la i+$1^{ème}$ bascule pour des bascules à front montant

les sortiesQ_0,Q_1, …Q_{n-1} permettent alors de compter en code binaire, Q_0 le poids faible étant la sortie de la première bascule.

Exemple : Un compteur modulo 4 à bascules D à front montant (↑).

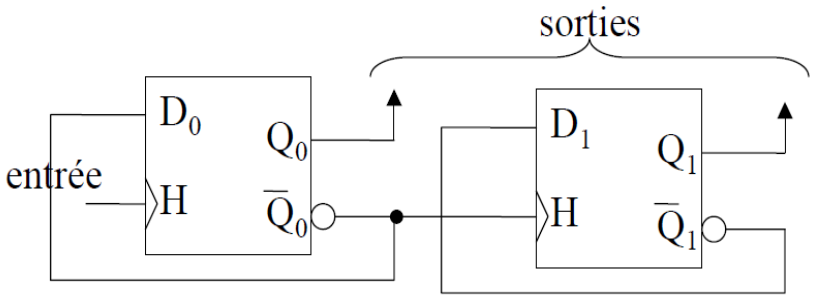


Figure 8.1 : Compteur asynchrone modulo 4.

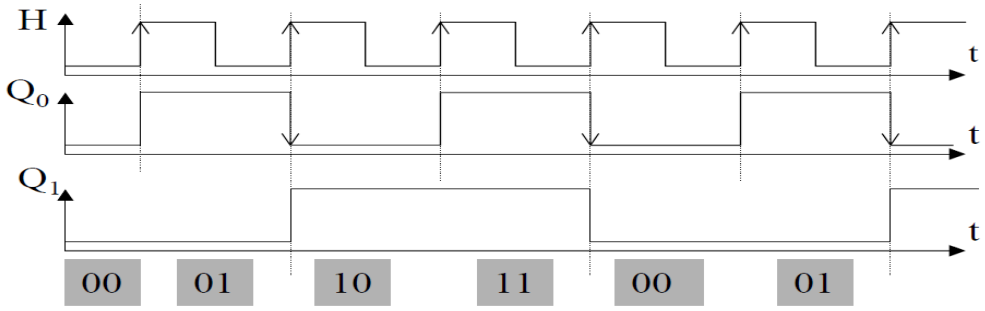


Figure 8.2 : Chronogramme d'un compteur asynchrone complet modulo 4.

8.3.2 Décompteur asynchrone modulo 2^n à cycle complet

Ce décompteur est constitué de n bascules en diviseur de fréquence en cascade avec :

- soit on réalise un compteur et on sort sur les sorties $\overline{Q_0}, \overline{Q_1}, \ldots\ldots, \overline{Q_{n-1}}$.
- soit on inverse la règle de cascade pour obtenir le décomptage sur les sorties $Q_0, Q_1, \ldots\ldots, Q_{n-1}$.

– la sortie Q_i est reliée à l'horloge de la i+1ème bascule pour des bascules à front montant.

– la sortie $\overline{Q_i}$ est reliée à l'horloge de la i+1ème bascule pour des bascules à front descendant.

Exemple : Un décompteur modulo 4 à bascules D à front descendant (↓).

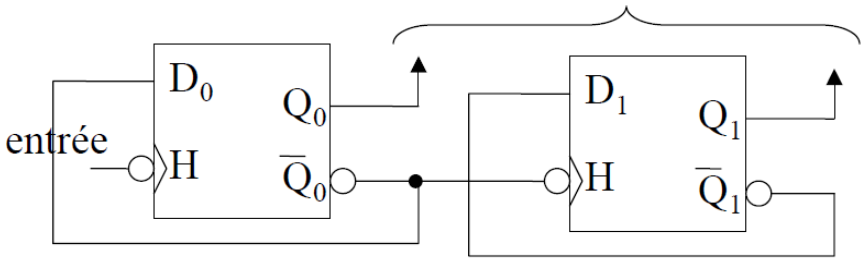


Figure 8.3 : Décompteur asynchrone modulo 4 complet.

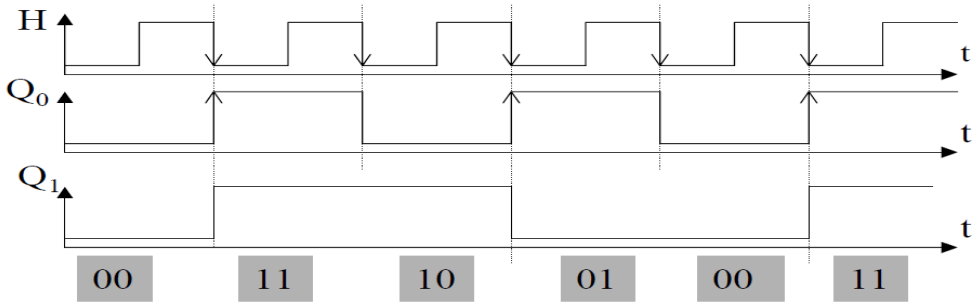


Figure 8.4 : Chronogramme d'un décompteur asynchrone modulo 4 complet.

8.3.3 Compteur modulo < 2^n à cycle incomplet

On réalise un compteur modulo $\mathbf{2^n}$ avec n bascules puis on utilise les entrées asynchrones Preset et Clear pour interrompre le cycle.

Exemple : Compteur asynchrone qui compte :0, 1, 2, 0, 1,2,0,...

Etat	Q_1	Q_0
0	0	0
1	0	1
2	1	0
3	1	1

On détectant l'état 3 le comptage doit être interrompu ce qui activera la mise à 0 des bascules d'où $Cl_0 = Cl_1 = \overline{Q1.Q0}$ si les entrées asynchrones sont actives au niveau bas.

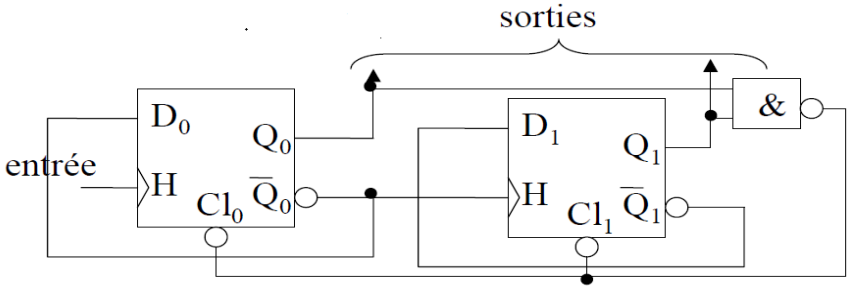


Figure 8.5 : Compteur asynchrone à cycle incomplet.

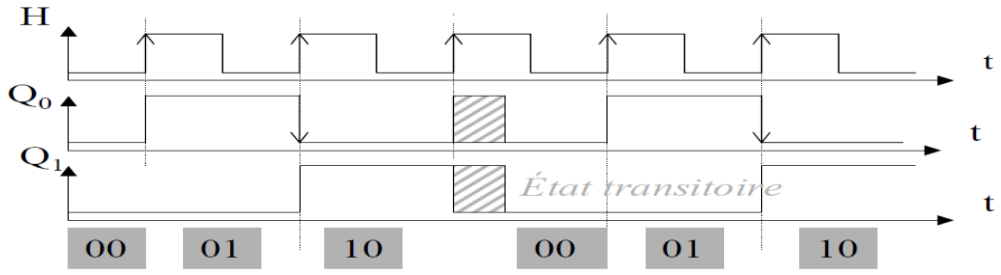


Figure 8.6 : Chronogramme d'un compteur asynchrone à cycle incomplet.

8.4 Compteur synchrone

Le compteur synchrone est un compteur parallèle où toutes les bascules auront une horloge commune afin de garantir la simultanéité des changements d'état.

8.4.1 Principe

Un compteur modulo 2^n nécessite aussi n bascules

On ajoute de la logique combinatoire reliant les sorties des bascules aux entrées afin de réaliser la séquence voulue.

8.4.2 Table de transition

La table de transition indique l'état des entrées en fonction de la transition souhaitée.

8.4.2.1 Bascule D

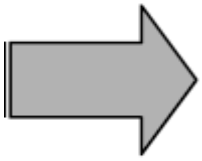
La table de transition de la bascule D :

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

8.4.2.2 Bascule JK

La table de vérité et la table de transition de la bascule JK :

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q_t}$



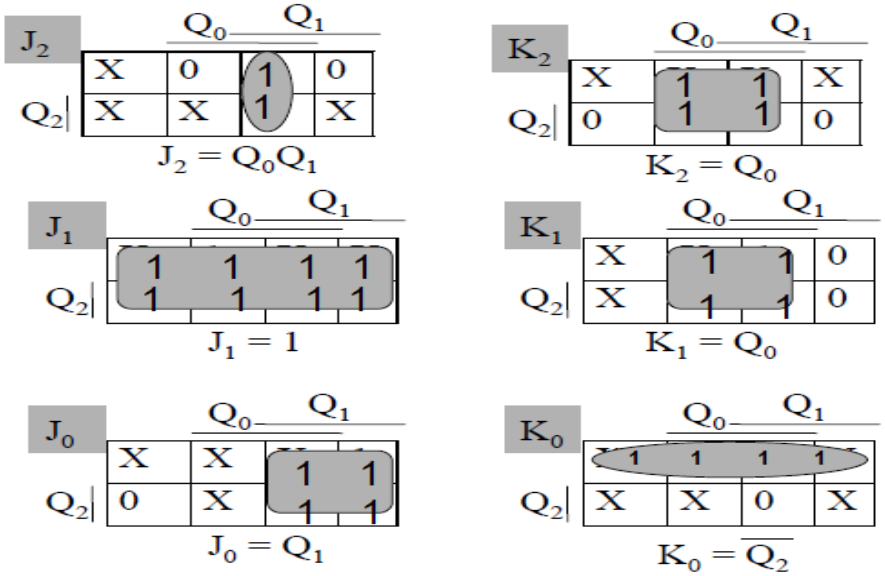
Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Exemple : Un compteur modulo 6 avec bascule JK à front descendant (↓) et qui compte : 1,2,3,4,6,7,1,2,3,4,6,7,1,…

La table de transition de ce compteur est la suivante :

Etat	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	1	0	X	1	X	X	1
2	0	1	0	0	X	X	0	1	X
3	0	1	1	1	X	X	1	X	1
4	1	0	0	X	0	1	X	0	X
6	1	1	0	X	0	X	0	1	X
7	1	1	1	X	1	X	1	X	0
1	0	0	1						

Et à l'aide du tableau de Karnaugh on trouve les différentes entrées K_i et J_i de ce compteur :



La figure suivante montre le schéma logique de ce compteur :

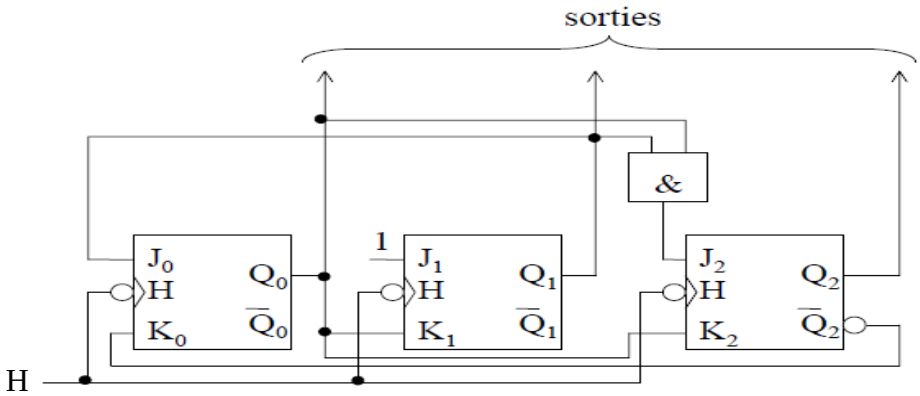


Figure 8.7 : Compteur synchrone modulo 6 avec JK (↓).

Remarque : les décompteurs sont conçus de la même façon.

Chapitre 9 : Les registres

9.1 Introduction

Le registre est un ensemble de cellules mémoire élémentaires (bascules) dans lequel un mot binaire est conservé provisoirement.

9.2 Les types de registres

9.2.1 Le Registre de mémorisation

Le registre de mémorisation est un registre tampon qui permet de temporiser le transfert d'informations entre deux sous-ensembles logiques.

Les différents étages sont indépendants les uns des autres ; certains signaux agissent sur l'ensemble des étages tels que l'horloge, le chargement et la mise à 0.

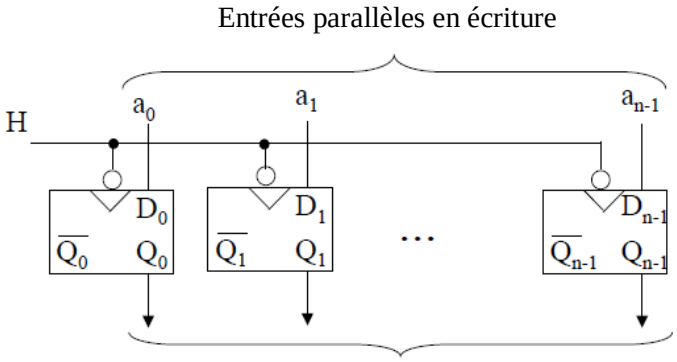


Figure 9.1 : Registre de mémorisation.

Au front descendant de H, chaque bascule recopie en sortie l'information en entrée.

Remarque : on peut utiliser les entrées asynchrones Preset et Clear pour commander la lecture et l'écriture.

9.2.2 Le Registre à décalage

Chaque décalage à droite d'une information binaire correspond à une division binaire par 2.

Il faut autant de bascules qu'il y a d'éléments binaires à mémoriser.

9.2.2.1 Les décalages

Exemple : mot de 4 bits.

1	0	1	1

Décalage à gauche

0	1	1	0

Décalage à droite

0	1	0	1

Rotation gauche

0	1	1	1

Rotation droite

1	1	0	1

9.2.2.2 Les entrées/sorties

On peut trouver des différentes entrées/sorties :

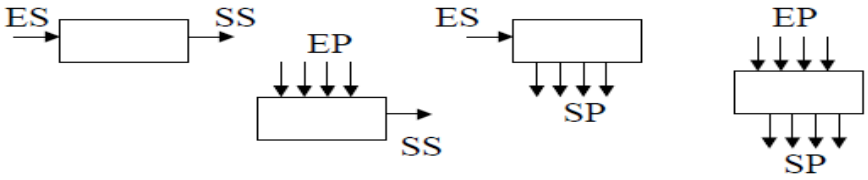


Figure 9.2 : Les différents types des entrées / sorties.

9.2.2.3 Registre universel

Ce registre contient des entrées série et parallèles et sorties série et parallèles.

Exemple : Un registre universel avec décalage à droite.

Pour ce type de registre il faut 4 impulsions d'horloge pour charger le registre et 4 impulsions d'horloge pour le décharger.

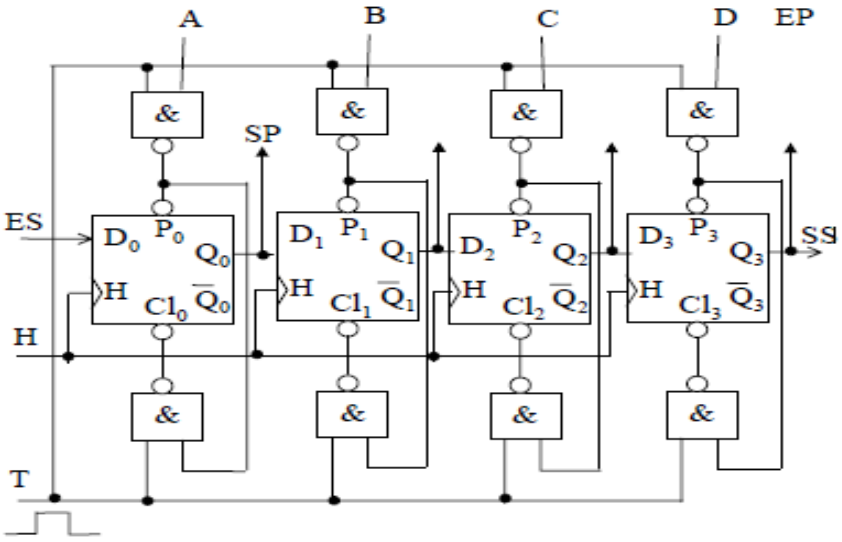


Figure 9.3 : Registre universel.

9.3 Applications des registres

Parmi les applications des registres, on trouve : la conversion parallèle/série, la conversion série/parallèle et le stockage d'information.

9.4 La fiche Technique du circuit intégré du registre à décalage universel (SN74LS194A)

9.4.1 Configuration du circuit intégré du registre universel (SN74LS194A)

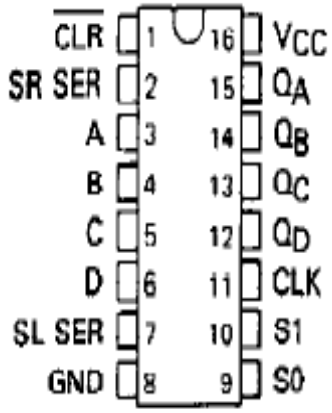
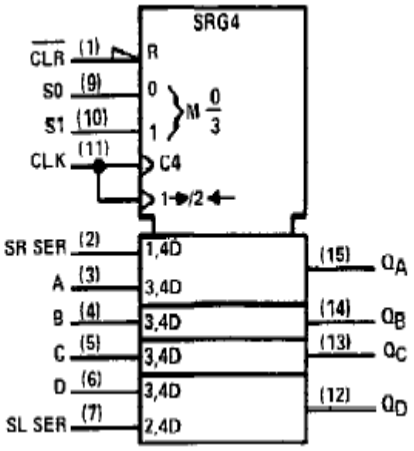


Figure 9.4 : Configuration des broches du registre.



. Figure 9.5: symbole logique du registre.

9.4.2 La table de fonction du registre universel (SN74LS194A)

Entrées										Sorties			
Niveau	Mode		Horloge	Séries		Parallèles				Q_A	Q_B	Q_C	Q_D
	S1	S0		Gauche	Droite	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
H	H	H	↑	X	X	a	b	c	d	a	b	c	d
H	L	H	↑	X	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	L
H	L	L	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

9.4.3 Le schéma logique du circuit intégré du registre universel (SN74LS194A)

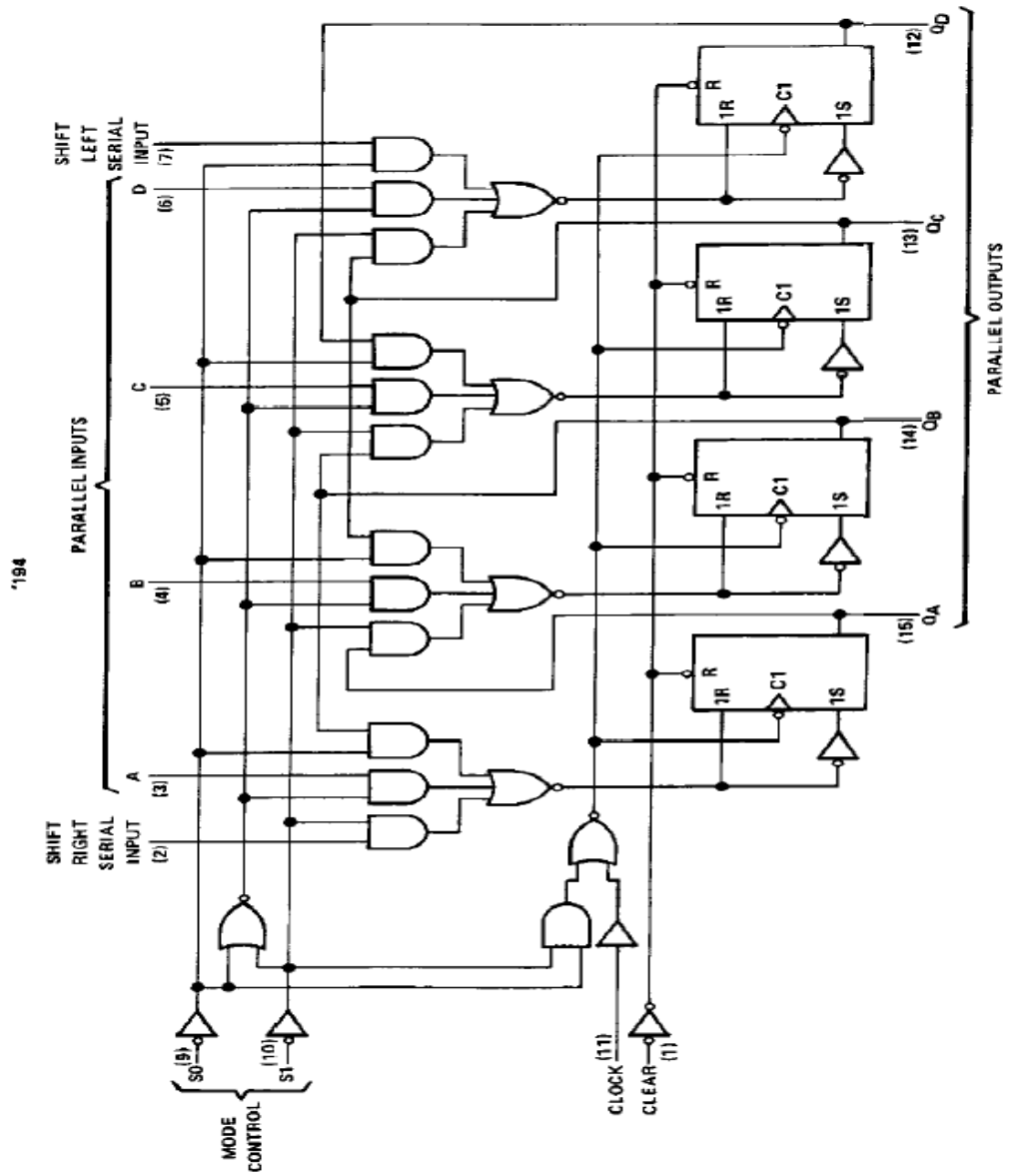


Figure 9.6 : Schéma logique du registre universel (SN74LS194A).

10. Bibliographie

[1] J.-C. Lafont et J.-P.Vabre,Cours et problèmes d'électronique numérique,Ellipses,1986.

[2] R. Strandh , I. Durand, Architecture de l'ordinateur : Portes logiques, circuits combinatoires, arithmétique binaire, circuits séquentiels et mémoires,exemple d'architecture, Dunod,2005.

[3] T. Ndjountche,Electronique numérique 1: Circuits logiques combinatoires, ISTE Editions, 2016.

[4] M. Gindre et D. Roux, Electronique numérique : logique combinatoire et technologie, cours et exercices, McGraw-Hill, Paris, 1987.

[5] J.-M. Bernard et J. Hugon, Pratique des circuits logiques, Collection technique et scientifique des télécommunications, Eyrolles, 1990.

[6] J.-M. Muller, Arithmétique des ordinateurs, opérateurs et fonctions élémentaires, Etudes et recherches en informatique, Masson, 1989.

[7] J.Letocha,Introduction aux circuits logiques,McGraw-Hill,1997.

[8] « Logique|FarnellFR ».[Enligne].Disponible sur:https://fr.farnell.com/c/semiconducteurs-circuits-integres/logique.[Consulté le: 28-mars-2019].

[9] « Décodeurs/Encodeurs|FarnellFR ».[Enligne].Disponiblesur: https://fr.farnell.com/c/semiconducteurs-circuits-integres/logique/decodeurs-encodeurs. [Consulté le: 28-mars-2019].

[10] « Multiplexeurs/Démultiplexeurs|FarnellFR ».[Enligne].Disponiblesur: https://fr.farnell.com/c/semiconducteurs-circuits-integres/logique/multiplexeurs-demultiplexeurs. [Consulté le: 28-mars-2019].

[11] « Comparateurs-numériques|FarnellFR ».[Enligne].Disponiblesur: https://fr.farnell.com/c/semiconducteurs-circuits-integres/logique/comparateurs-numeriques. [Consulté le: 28-mars-2019].

[12] M. Gindre,D.Roux, « Logique séquentielle : cours et exercices », Ediscience international, 1994.

[13] « SN74LS194AN - Registre à décalage, Famille LS, 74LS194, Universel, 1 Elément, 4 bit, DIP, 16 Broche(s) ». [En ligne]. Disponible sur: https://fr.farnell.com/texas-instruments/sn74ls194an/logic-bi-dir-univ-shift-reg-16dip/dp/1740122. [Consulté le: 01-avr-2019].