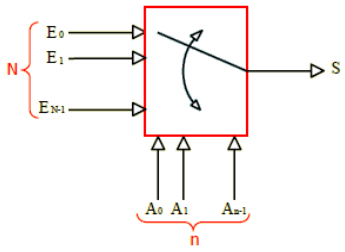
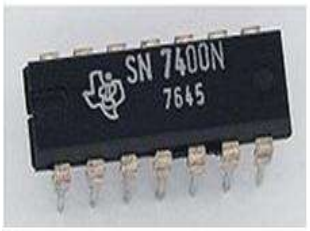
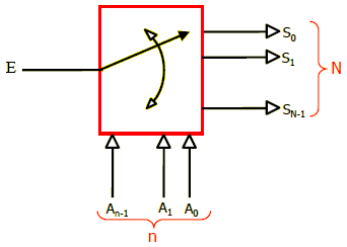
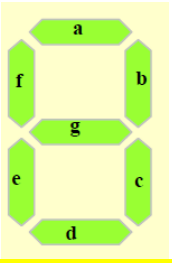
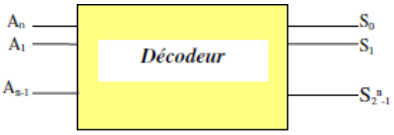
جامعة سيدي محمد بن عبدالله
كلية العلوم ظهر المهراز
Université Sidi Mohamed Ben Abdellah
Faculté des Sciences Dhar Mahraz

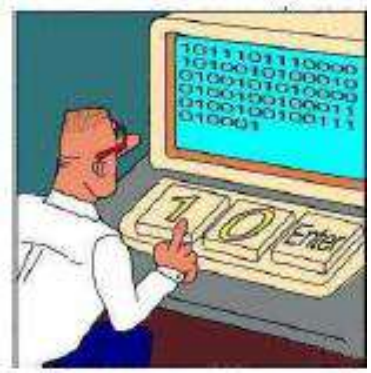












DEPARTEMENT DE PHYSIQUE

L.P.S.

ELECTRONIQUE NUMERIQUE

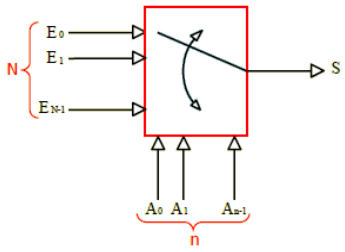
Semestre 3 – Module électronique M 20 - Filière SMI

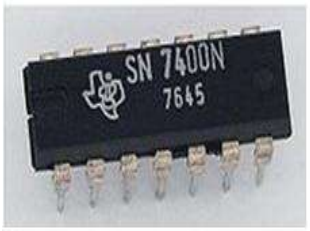
Année universitaire 2014-2015

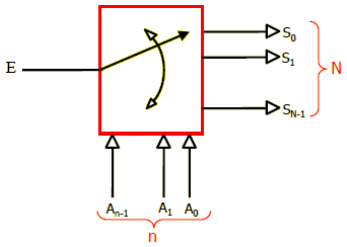
Réalisé par :

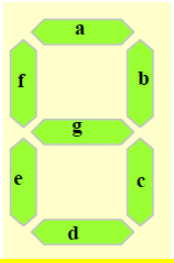
Pr. REZZOUK Abdellah

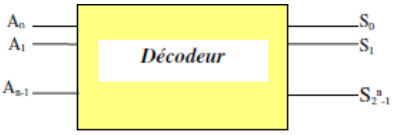
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⴰⵎⵎⴻⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
ⵜⴰⵖⴻⵍⵜ ⵜⴰⵎⴰⵔⵔⴰⵜ ⴻⵔⵔⵓⵎ ⵏ ⵣⵀⵔ ⵍⵎⵀⵔⴰⵣ
جامعة سيدي محمد بن عبدالله
كلية العلوم ظهر المهراز
Université Sidi Mohamed Ben Abdellah
Faculté des Sciences Dhar Mahraz

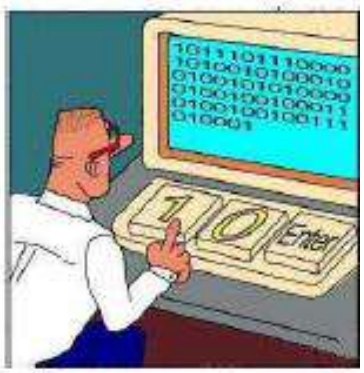












DEPARTEMENT DE PHYSIQUE

L.P.S.

Logique combinatoire

Semestre 3 – Module électronique M 20 - Filière SMI

Année universitaire 2014-2015

Réalisé par :

Pr. REZZOUK Abdellah

Partie 1 : logique combinatoire

- Chapitre 1 : Les systèmes de numération et codes
- Chapitre 2 : Variables et fonctions logiques
- Chapitre 3 : Simplifications des fonctions logiques
- Chapitre 4 : Les circuits combinatoires usuels
    - Les opérations arithmétiques (addition, soustraction, comparaison…)
    - des comparaisons (comparateur, contrôleur, générateur…)
    - d'aiguillage (multiplexeurs, démultiplexeurs)
    - de transcodage (codeur, décodeur)
    - afficheur 7 segments

Bibliographie

- Logique binaire, M. AUMIAUX, édition Masson
- Cours et PB d'électronique numérique, Alain LEBEQUE, Alain Pelet et J. Paul Vabre, Ellipse
- Notes de cours de Sylvain Tisseront, Février 2003
- Eric Leclercq : http://ludique.u-bourgogne.fr/MEMBRES/leclercq
- David Simplot : http://lifl.fr/~simplot/ens/archi

Notes de cours en ligne :

Site : http://www.fsdmfes.ac.ma/ **(voir ressources pédagogiques/filière SMI/S3/rezzouk)**

Introduction

a. Les circuits logiques peuvent être classifiés suivant deux types:

- les ***circuits logiques combinatoires*** pour lesquels la notion de temps n'intervient pas (les sorties dépendent uniquement de l'état actuel des entrées).
- les ***circuits logiques séquentiels*** pour lesquels la notion de temps intervient (les sorties dépendent des entrées et de l'historique, c'est à dire de ce qui s'est passé auparavant)..

Dans le premier chapitre, nous envisagerons deux méthodes pour représenter les nombres et nous aborderons les opérations qui s'y rattachent. Chacune de ces méthodes fait appel à un système de numération dont la base est différente.

La plus répandue, que vous connaissez bien, est celle qui utilise la base 10. L'autre, employée dans les circuits numériques, est à base 2.

Nous n'allons pas reprendre toutes les notions d'arithmétique apprises à l'école mais nous chercherons surtout à revoir certains points. Ces derniers vous seront utiles pour une meilleure compréhension de l'arithmétique employée dans les circuits d'électroniques.

Chapitre 1 : Système de numérations et codages

INTRODUCTION

Habituellement, on utilise le système décimal pour représenter les nombres, mais il est possible d'utiliser d'autres **systèmes de numération**.

Nous nous intéressons dans ce chapitre aux systèmes de numération fréquemment rencontrés en technologie numérique. Il s'agit des systèmes **binaire**, **octal**, **décimal** et **hexadécimal**. Avant de décrire ces systèmes, nous allons donner quelques propriétés générales. Ensuite, nous définissons la notion de **base** d'un système de numération, le principe d'écriture d'un nombre dans un système de base b donnée, ainsi que les règles opératoires du système binaire.

I. LES SYSTEMES DE NUMERATIONS

I.1. Propriétés générales

I.1.1. Definitions

- Système de numérotation: *est un ensemble de symboles et de règles* permettant la représentation, de n'importe quel élément d'un ensemble, d'un nombre.

- Base d'un système: *est la référence* qui permet l'écriture d'un nombre.

- La base est le nombre qui sert à définir un système de numération.

- La base du système décimal est **10** alors que celle du système octal est **8**.

I.1.2. Forme polynomiale

L'équivalent décimal de $\mathbf{(N)_b}$, d'un nombre N entier écrit dans une **base b**, s'obtient par application directe de la *forme polynomiale* suivante :

$$\boxed{(N)_b = \sum_{i=0}^{i=n-1} a_i b^i \quad avec \quad 0 \le a_i \le b-1}$$

- ***b : base*** (binaire, b=2, octal, b=8, hexadécimale, b=16 ou H, décimal, b=10).
- ***a_i : symboles ou coefficients***
- $(N)_b = a_{n-1}b^{n-1} + \ldots\ldots + a_0 b^0 = (a_{n-1}\, a_{n-2} \ldots\ldots a_1\, a_0)_b \; (N)b = a_{n-1}b^{n-1} \ldots . + a_0 b^0 = (a_{n-1} \ldots \ldots a_0)_b$
- Chaque symbole $\mathbf{a_i}$ peut prendre une valeur entre ***0 et (b-1).*** Le chiffre $\mathbf{a_i}$ présente un poids égal à $\mathbf{b^i}$.
- La notation $\mathbf{()_b}$ indique que le nombre est écrit en base b. En décimal, on ne note pas d'indice.

- ***Exemples :***

♣ Dans la base décimale b =10 : ……

♣ Dans la base binaire b = 2 : ……

♣ Dans la base octal b = 8 : ……

♣ Dans la base hexadécimal b = 16 : ……

- ***Remarque :***

On commence tjrs par les cœfficients dans le sens de la droite → gauche :

$$(N)_b = \sum_{i=0}^{i=n-1} a_i b^i = (a_{n-1}.....a_0)_b \text{ et } 0 \le a_i \le b-1$$

I.1.3. Capacité décimale

♣ Si l'on dispose de *n rangs*, symboles, on peut former $\mathbf{b^n}$ combinaisons différentes, autrement dit on peut compter de ***0 et (b^n-1).*** *C'est la capacité décimale* d'un nombre N entier dans une base b, $(N)_b$.

■ ***Exemples***

☐ ***La capacité décimale d'1 nb binaire de 12 bits est :***

……

☐ ***La capacité décimale d'1 nb octal de 4 chiffres est :***

……

☐ ***La capacité décimale d'1 nb hexadécimal de 3 caractères est :***

……

I.1.4. Vocabulaire

♣ Un « *Bit* » (contraction américaine de Binary digiT) est un digit du système binaire.

♣ Un « *Mot* » (Word en américain) est un ensemble de bits dont il faut préciser le nombre. Par exemple : un mot de 16 bits, un mot de 12 bits,…..Un mot de 10 bits sera nécessaire pour exprimer un nombre compris entre 0 et 1023 (définition de la capacité décimale).

- Un « *Quartet* » (Nibble en américain) est un mot de 4 bits.
- Un « *Octet* » (Byte en américain) est un mot de 8 bits.
- Le « Bit de poids faible » (*L.S.B.* en américain, Less Significant Bit) est le bit situé le plus à droite, donc de plus faible poids.
- Le « Bit de poids fort » (*M.S.B.* en américain, Most Significant Bit) est le bit situé le +plus à gauche,
- En unité de capacité de traitement numérique : un « *kilo* » = 2^{10} = 1024.
- Un « *kO* » = 1 kiloOctet = 1024 Octets
- Un « *MO* » = 1 MégaOctet = 1024 kO
- Un « *GO* » = 1 GégaOctet = 1024 MO

I.2. Système de numération

I.2.1. Numération décimale

Ce système de numération, usuel dans la vie quotidienne, dispose de dix symboles (les chiffres) : ***0, 1, 2, 3, 4, 5, 6, 7, 8, 9***. On travaille alors en **base 10**.

Exemples:

- ……..…….……

- ……..…….……..……

- ……..…….……

I.2.2. Numération binaire naturel (BN)

La numération en base deux (ou **numération binaire**) utilise deux symboles 0 et 1. Cette base est très commode pour distinguer les deux états logiques fondamentaux. On écrit:

$$\left(a_{n-1}\, a_{n-2}.....a_1\, a_0\right)_2 = a_{n-1}2^{n-1} + ... + a_0 2^0$$

(Expression de droite écrite dans la base 10 et $a_i \in \{0, 1\}$).

Exemples :

- ……

- ……

Un nombre à *n* symboles en base deux distingue 2^n états.

I.2.3. Numération octal

Le système octal utilise 8 symboles (***chiffres***) seulement : ***0, 1, 2, 3, 4, 5, 6, 7***

On écrit: $(N)_8 = (a_{n-1}\, a_{n-2}.....a_1\, a_0)_8 = a_{n-1}8^{n-1} + ... + a_0 8^0 = (N)_{10}$ (Expression de droite écrite dans la base 10 et $a_i \in$ {0, 1, 2, 3, 4, 5, 6, 7}).

Exemple:

- $(745)_8$ = ……

Nous venons de voir que : : **$(745)_8 = (485)_{10}$**

I.2.3. Numération hexadécimal

Le développement des systèmes micro programmés (mini- et micro-ordinateurs) a favorisé l'utilisation de ce code. Il comporte 16 symboles***, caractères*** : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.**

On écrit: $(a_{n-1}\, a_{n-2}....a_1\, a_0)_H = a_{n-1}16^{n-1} + ... + a_0 16^0$ (Expression de droite écrite dans la base 10 et **$a_i \in$ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**).

La correspondance entre base 2, base 10 et base 16 est indiquée dans le tableau ci-après :

Base 10	Base 16	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Exemples :

-……

- ……

- ……

I.3. Changements de base – Conversions

I.3.1. Conversion décimal vers base quelconque

Il existe plusieurs moyens d'effectuer une telle conversion :

Nous avons vu, d'après la forme polynomiale, qu'un nombre binaire s'écrit : $N = \sum_{i=0}^{i=n-1} a_i b^i$. Le problème revient donc à déterminer les valeurs des bits a_i.

Soit $A = \sum_{i=0}^{i=3} a_i b^i = (a_3 a_2 a_1 a_0)_b = (a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0) \quad avec \quad 0 \le a_i \le b-1$

Divisons le nombre ***A*** par la base ***b***, puis le quotient obtenu par ***b***, jusqu'à ce que le ***quotient devienne nul***. Il vient que :

$$\begin{array}{|ll|} \hline A = b(a_3 b^2 + a_2 b^1 + a_1 b^0) & + a_0 = bQ_1 + a_0 \\ Q_1 = b(a_3 b^1 + a_2 b^0) & + a_1 = bQ_2 + a_1 \\ Q_2 = b(a_3 b^0) & + a_2 = bQ_3 + a_2 \\ Q_3 = b(0) & + a_3 = 0 \quad + a_3 \\ \hline \end{array}$$

♠ Les restes successifs lus de bas en haut représentent le nombre ***A*** dans la base ***b***.

♠ Le calcul ci-dessus est valable quelque soit la base, et notamment la base 10.

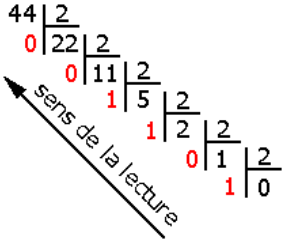
♠ Il permet d'obtenir l'expression d'un nb *décimal dans un système de base b qlq*.

♠ En résumé : ***l'expression d'un nombre décimal, A, dans un système de base b quelconque s'obtient par une répétition de division par b jusqu'à obtention d'un quotient nul*** et *A s'écrit donc (le premier reste à la position du LSB (à droite) et le dernier reste à la* position du MSB (à gauche) : A = (MSB……..LSB)$_b$ = (a_n……..a_0)$_b$

Exemple b = 2: Convertir le nombre décimal 44 en binaire puis en octal

Division	***Quotient***	***Reste***
44/8	5	$4 \rightarrow a_0$
5/8	0	$5 \rightarrow a_1$

44 = (54)$_8$
44 = (101100**)$_2$**



Les restes de ces divisions lus de bas en haut représentent le nombre binaire :

44 =(101100)$_2$

• Règle pratique pour la ***base 2*** :

44	22	11	5	2	1
0	0	1	1	0	1

0 si pair, 1 si impair $\Rightarrow$ ***(44)$_{10}$ = (101100)$_2$***

Exemple b = 10 : Convertir le nombre décimal 543 en décimal.

……
……
……

Exemple b = 8 : Convertir le nombre décimal 1223 en octal.

……
……
……
……

Exemple b = 16 : Convertir le nombre décimal 5000 en héxadécimal.

……
……
……
……

I.3.2. Toutes les conversions vers le décimal

L'équivalent décimal de **(N)$_b$**, d'un nombre N entier écrit dans une **base b**, s'obtient par application directe de la *forme polynomiale* suivante :

$$\boxed{(N)_b = \sum_{i=0}^{i=n-1} a_i b^i \quad avec \quad 0 \le a_i \le b-1} \quad (N)b = \sum_{i=0}^{i=n-1} a_i\, b^i$$

- b : base (binaire, b=2, octal, b=8, hexadécimale, b=16 ou H, décimal, b=10).
- a_i : symboles ou coefficients (on dispose de b symboles tel que **$0 \le a_i \le b-1$**).
- $(N)_b = a_{n-1}b^{n-1} + ... + a_0 b^0 = (a_{n-1}\, a_{n-2} ..... a_1\, a_0)_b$
- $(N)b = a_{n-1}b^{n-1} \ldots . + a_0 b^0 = (a_{n-1} \ldots \ldots a_0)_b$

Exemples :

♣ Dans la base décimale b =10 : ……

♣ Dans la base binaire b = 2 : ……

♣ Dans la base Hexadécimal b = 16 : ……

♣ Dans la base Octal b = 8 : ……

I.3.3. Conversions directes (sans passer par le décimal)

Dans les bases usuelles (2, 8 et 16) utilisées dans les systèmes numériques, les conversions peuvent être réalisées par exploitation de propriétés particulières aux nombres de ces bases.

I.3.3.1. Binaire ↔ octal

***En résumé**, on passe facilement du binaire à l'octal en groupant les bits par blocs de trois en allant vers la gauche puis on fait correspondre à chaque bloc son équivalent décimal.*

L'équivalent binaire d'un nombre octal s'obtient en faisant correspondre à chaque chiffre octal son équivalent binaire sur un bloc de 3 bits

Exemples : de conversion binaire ↔ octal :

a) $(100111010001)_2$ = …….

b) $(5635)_8$ = ……

c) $(10111)_2$ = ……

I.3.3.2. Binaire ↔ hexadécimal

Donc, pour convertir du binaire en hexadécimal, on divise le nombre binaire en « tranches de quatre » en partant de la droite. Chacun des « paquets » est ensuite converti en hexadécimal.

Cette méthode revient à fractionner en décompositions successives.

Le tableau suivant donne la correspondance entre les chiffres décimaux allant de 0 à F et leur équivalent binaire sur 4 bits :

Décimal	*Binaire*
0	$(0000)_2$
1	$(0001)_2$
2	$(0010)_2$
3	$(0011)_2$
4	$(0100)_2$
5	$(0101)_2$
6	$(0110)_2$
7	$(0111)_2$
8	$(1000)_2$
9	$(1001)_2$
A	$(1010)_2$
B	$(1011)_2$
C	$(1100)_2$
D	$(1101)_2$
E	$(1110)_2$
F	$(1111)_2$

Exercices :
Convertir les nombres binaires suivants en hexadécimal :
a) $(110011010001)_2$;
b) $(101011001101)_2$;
c) $(110101110001)_2$;
réponse

a) $(110011010001)_2$ = ……

b) $(101011001101)_2$ = ……

c) $(110101110001)_2$ = ……

d) $(BC34)_H$ = ……

I.3.4. Conversions indirectes (Octal ↔ hexadécimal)

L'équivalent en octal d'un nombre hexadécimal s'obtient directement en passant par l'intermédiaire de binaire :

- On convertit en binaires, les nombres hexadécimaux
- Puis on convertit en octal les nombres binaires.

Exercices : Convertir les nombres suivants en hexadécimal puis en octal:

- ……
- …… ……

I.4. Conversions en complément à 1 : C_1

Le **complément à 1,** ou **complément restreint** (CR) ou **complément logique,** d'un nombre est obtenu en inversant chaque bit. (les '0' deviennent des '1' et les '1' des '0').

Il faut noter qu'il suffit de remplacer les 0 par des 1 et vice-versa pour trouver le complément à 1 d'un nombre binaire, la procédure est donc très simple.

Nombre binaire	*Complément à 1*
$(101)_2$	$(010)_2$
$(0111)_2$	$(1000)_2$

Le nombre binaire $(\mathbf{1000})_2$ est le complément à **1** de $(\mathbf{0111})_2$. Si on additionne ces deux nombres, on obtient **:** $(\mathbf{1111})_2$.

- C'est-à-dire $\mathbf{x+CR(x) = 2^n - 1 \Rightarrow CR(x) = 2^n - 1-x}$ (*n* est le nombre de bits de *x*).
- $\mathbf{C_1(C_1(x)) = x}$ **et** $\mathbf{x+ C_1(x) = 2^n - 1}$

Il vient donc : $\boldsymbol{x + CR(x) + 1 = 2^n}$. Or 2^n correspond à un cycle qui n'est pas interprété donc équivaut à zéro. Par conséquent, ***la valeur négative (–x) est représentée par CR(x) + 1.***

La valeur $\mathbf{CR(}x\mathbf{) + 1}$ est appelée *complément vrai* (CV) ou ***complément à 2*** de *x*, $\mathbf{C_2(x).}$

Exemple : ……

I.5. Conversions en complément à 2 : C_2

*Le **complément à 2**, ou **complément vrai** (CV) ou **complément arithmétique,** d'un nombre binaire est tout simplement son complément à 1 auquel on additionne 1 : $C_2(x)=C_1(x)+1$. L'addition en binaire se fait de la même manière que l'addition décimale avec la table suivante :*

0 + 0	= 0 + retenue de 0
0 + 1	= 1 + retenue de 0
1 + 1	= 0 + retenue de 1
1 + 1 + 1	= 1 + retenue de 1

Nombre binaire	*Complément à 2*
$(101)_2$	$(011)_2$
$(0111)_2$	$(1001)_2$

Exercice : Soit à trouver le complément à **2** de **1010**. S***olution*** : $C_2(1010) =(0110)_2$

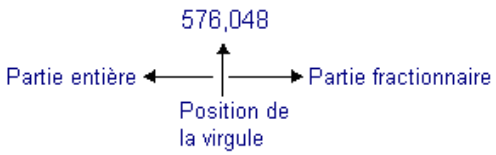
- Ce complément va nous servir à réaliser les opérations dans les systèmes numériques.

Généralement

- $C_2(C_2(x)) = x$ et $C_2(x)= - x \Rightarrow C_2(x) + x = 0$

On a vu le codage des nombres entiers positifs dans différentes bases, mais on doit pouvoir manipuler des nombres réels et des nombres négatifs

I.6. Codages des nombres réels ou fractionnaires



Exemples :
- Dans la base 10 : ……
- Dans la base 2 : ……
- Dans la base 8 : ……
- Dans la base 16 : ……

Un nombre réel est constitué de 2 parties :
- *la partie entière N_{entier} (avant la virgule) : son équivalent binaire, octal ou hexadécimal est obtenu par une répétition de division par b=2, 8 ou 16 jusqu'à obtention d'un quotient égal a zéro.*
- *La partie fractionnaire N_{frac} (après la virgule) : son équivalent binaire, octal ou hexadécimal est obtenu par une répétition de multiplication par b=2, 8 ou 16 jusqu'à obtention d'un produit entier ou jusqu'à ce que la précision soit suffisante. Car on ne peut pas toujours obtenir une conversion en un nombre fini de chiffre pour la partie fractionnaire.*
- *La partie fractionnaire dans la base b est la concaténation des parties entières obtenues dans l'ordre de leur calcul*

En générale on représente un nombre réel positif dans une base ***b*** par:

$$(N)_b = a_{n-1}b^{n-1} + ... + a_0 b^0 + a_{-1}b^{-1} + ... + a_{-m}b^{-m} = (a_{n-1}\,a_{n-2}.....a_1\,a_0\,,a_{-1}....., a_{-m})_b$$

$$(N)_b = \sum_{i=-m}^{i=n-1} a_i b^i$$

$$(a_{n-1}\,a_{n-2}.....a_1\,a_0\,,a_{-1}....., a_{-m})_b = (N_{entier}, N_{fractionnaire})_b$$

$$N_{entier} = (a_{n-1}\,a_{n-2}.....a_1\,a_0)_b \text{ et } N_{frac} = (0, a_{-1}....., a_{-m})_b$$

Exemple b = 2 : Convertir le nombre décimal réel suivant en binaire.

……
……
……
……

Exemple b = 16 : Convertir le nombre décimal réel suivant en héxadécimal.

……
……
……
……

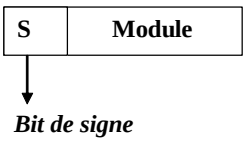
I.7. Codages des nombres signés

Les nombres signés sont représentés selon trois notations :
- Notation module plus Signe,
- Notation complément à 1
- Notation complément à 2.

Les nombres positifs ont la même représentation dans les trois notations.

1. Notation module plus signe (noté M&S)

Un nombre binaire signé comprend un bit de signe (0 pour les nombres positifs et 1 pour les nombres négatifs) et n bits indiquant le module. Ainsi, si un nombre N est codé sur n+1 bits, alors ce nombre est compris entre ***$-(2^n-1)$ et $+(2^n-1)$.***
Avec 5 bits, N est compris entre -15 et 15.



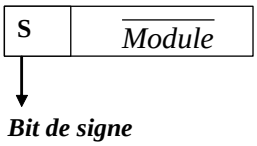
Exemple :

……
……
……
……
……
……

Dans la notation 'Module plus Signe', le zéro a deux représentations différentes (le bit de signe S prend 0 ou 1).

2. Notation complément à 1(noté C à 1)

Pour représenter un nombre N négatif en notation complément à 1, il suffit d'attribuer au bit de signe la valeur de 1 et transformer le module en son complément à 1.



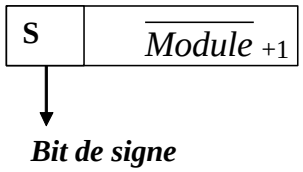
Exemple :

……
……
……
……
……
……
……
……

Dans la notation 'Notation complément à 1, le zéro a deux représentations différentes (le bit de signe S prend 0 ou 1).

3.Notation complément à 2 (noté C à 2)

Pour représenter un nombre négatif en notation complément à 2, il suffit d'attribuer au bit de signe la valeur de 1 et de transformer le module en son complément à 2.



Exemple :

……
……
……
……
……
……
……
……

Cas particulier du complément à 2 :

L'équivalent décimal du nombre binaire N=1 00...00 est -2^n.

Par conséquent, avec n+1 bits, on peut représenter des nombres signés allant de **-2^n à $+(2^n-1)$.**

II. NOTIONS SUR LES CODES

On distingue deux catégories de codes : les codes numériques et les codes alphanumériques. Jusqu'ici, nous n'avons utilisé que le code binaire naturel. Plusieurs codes sont utilisés en techniques numériques, nous citons entre autres :

- le code binaire naturel (BN)
- le code binaire réfléchi (BR) ou code Gray
- le code DCB (Décimal Codé Binaire)
- le code Excédent 3
- le code ASCII (American Standard Code for Information Interchange)
- le code de parité

II.1. Codage numérique

II.1.1. Code Binaire Naturel (BN)

Le code BN est le code dans lequel on exprime un nombre selon le système de numération binaire. C'est le code le plus simple.
Le code BN et ses dérivés (octal et hexadécimal) répondent aux règles classiques de l'arithmétique des nombres positifs (on peut calculer, il est donc pondéré).

II.1.2. Code Binaire Réfléchi (BR)

Dans le code binaire réfléchi (code Gray), deux représentations codées successives ne différent que d'un seul bit. Les tableaux suivants donnent quelques correspondances entre le codage BN et le codage BR.

En observant le tableau n°1 qui expose la correspondance entre le décimal, le binaire et le binaire réfléchi des chiffres allant de 0 à 3 sur deux bits, on remarque que lorsqu'on passe de la représentation binaire du chiffre '1' à celle de '2' ; deux bits changent. Ceci n'est pas possible en code binaire réfléchi où un seul bit change seulement.

Décimal	Binaire naturel	Binaire réfléchi
0	$(00)_2$	$(00)_2$
1	$(01)_2$	$(01)_2$
2	$(10)_2$	$(11)_2$
3	$(11)_2$	$(10)_2$

tableau 1

Le tableau 2 ci dessous expose la correspondance entre le décimal, le binaire et le binaire réfléchi sur 3 bits :

Décimal	binaire	binaire réfléchi
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tableau 2

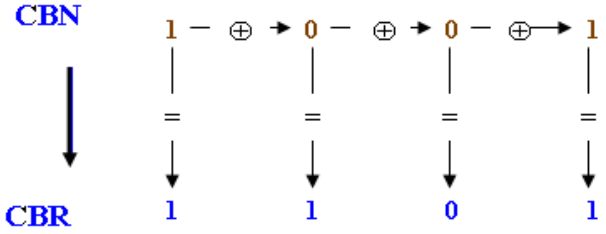
II.1.2.1. Conversion de CBN → CBR

$(X\ Y\ Z)_{BR} = (a\ b\ c)_2$ avec le 1er chiffre est recopié : *X = a, Y = b ⊕ a donc Y est la somme sans retenue de b et a, Z est la somme sans retenue de c et b,…..*

Pourquoi ? *Voir la solution de l'exercice Transcodeur BN en BR*

Exemple :

- ***$(1001)_2 = (1101)_{BR}$ ⇔ 9 en BN vaut 13 en BR***

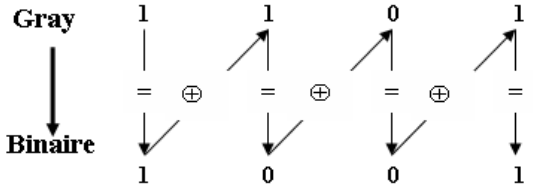


II.1.2.2. conversion de CBR → CBN

On part de la gauche vers la droite : le 1er chiffre est recopié, on trouverait de la même manière X =a, Y = X ⊕ b, Z=Y ⊕ c, *…ce qui permet de trouver la règle de conversion, sachant que* $0 \oplus n = 0$ *et* $1 \oplus n = \bar{n}$.

Pourquoi ? Voir la solution de l'exercice Transcodeur BR en BN :

Exemple : *$(1101)_{BR} = (1001)_2$ ⇔ 13 en BR vaut 9 en BN*



II.1.3. Code décimal codé binaire (DCB)

Le code DCB signifie **D**écimal **C**odé **B**inaire. *Chaque chiffre du nombre décimal est codé individuellement en son équivalent binaire sur quatre bits (quartet),* ce qui n'est pas le cas pour le code binaire naturel où on convertit le nombre décimal dans son intégralité.

Exemples: convertir en DCB puis en binaire les nombres suivants : 127, 255 et 64.

……
……
……
……
……
……

Remarques :

- Le code DCB est un code non pondéré. Il n'obéit pas à la Forme Polynomiale.
- Dans le code DCB, il faut plus de bits pour exprimer le même nombre, qu'en code binaire.
- Le code DCB n'utilise que dix quartets parmi 16. Si l'un des quartets interdits (1010,1011,1100,1101,1110,1111) se manifeste dans un calculateur utilisant le code DCB, c'est alors un signe d'erreur.

II.1.4. Code Excédent 3

Comme dans le code DCB, on code chaque chiffre selon son équivalent binaire, mais augmenté de 3 : $0 \rightarrow (0011)_2$, $1 \rightarrow (0100)_2$, …. , $9 \rightarrow (1100)_2$,

Les 6 combinaisons de 10 à 15 ne sont pas utilisées. On obtient ainsi, par exemple :

Exemple : $9708 = (1100\ 1010\ 0011\ 1011)_{exc\ 3} = (1100101000111011)_{exc\ 3}$

Remarques :

- Ce codage n'est pas pondéré, mais il présente l'avantage d'être auto complémentaire
- C'est-à-dire que la complémentation des 4 bits d'un chiffre donne son complément à 9
- En effet : $7 = (1010)_{exc\ 3}$, complémenté donne $(0101)_{exc\ 3} = 2$, et 7+2=9

II.2. Codage alphanumérique

II.2.1. Code ASCII, American Standard Code for Information Interchange

Pour manipuler d'autres éléments que des nombres, il est nécessaire de les coder. Le plus connu de ces codes, et le plus utilisé en particulier dans le monde informatique, est le code ASCII (*American Standard Code for Information Interchange*) présenté dans le ***Tableau 2***.

Utilisé pour les échanges en informatique, le code ASCII permet de coder les 26 lettres de l'alphabet (majuscules et minuscules), les 10 chiffres et les signes de ponctuation ainsi que des caractères spéciaux (? ! + - : / # @ & ….).: il utilise un octet (8bits). Cet octet donne une centaine souplesse d'utilisation, puisqu'il permet de coder des commandes de contrôle en plus des caractères alphanumériques (bits 1 à 7), d'utiliser le bit 8 comme bit de parité ou pour définir un deuxième tableau de caractères (caractères étendus).

Le tableau ci-dessous nous donne une partie du code ASCII où les 7 bits sont désignés par b_6, b_5, b_4, b_3, b_2, b_1, b_0 :

				b_6	0	0	1	1	1	1
				b_5	1	1	0	0	1	1
				b_4	0	1	0	1	0	1
b_3	b_2	b_1	b_0	HEX	2	3	4	5	6	7
0	0	0	0	0	*SP*	*0*	*@*	*P*		*p*
0	0	0	1	1	*!*	*1*	*A*	*Q*	*a*	*q*
0	0	1	0	2	*"*	*2*	*B*	*R*	*b*	*r*
0	0	1	1	3	*#*	*3*	*C*	*S*	*c*	*s*
0	1	0	0	4	*$*	*4*	*D*	*T*	*d*	*t*
0	1	0	1	5	*%*	*5*	*E*	*U*	*e*	*u*
0	1	1	0	6	*&*	*6*	*F*	*V*	*f*	*v*
0	1	1	1	7	*'*	*7*	*G*	*W*	*g*	*w*
1	0	0	0	8	*(*	*8*	*H*	*X*	*h*	*x*
1	0	0	1	9	*)*	*9*	*I*	*Y*	*i*	*y*
1	0	1	0	A	*	*:*	*J*	*Z*	*j*	*z*
1	0	1	1	B	*+*	*;*	*K*	*[*	*k*	*{*
1	1	0	0	C	*,*	*<*	*L*	**	*l*	*|*
1	1	0	1	D	*-*	*=*	*M*	*]*	*m*	*}*
1	1	1	0	E	*.*	*>*	*N*	*^*	*n*	*~*
1	1	1	1	F			*O*	*_*	*o*	*DEL*

♣ SP : Espace

♣ Les codes de valeurs inférieurs à 32 (en décimal) sont réservés pour coder des caractères de contrôle qui ne sont pas imprimable.

Exemple :

le code ASCII de ‘ K’ est : 1001011 = $(4B)_H$ = 75
le code ASCII de ‘ A’ est : 1000001 = $(41)_H$
le code ASCII de ‘ z’ est : 1111010 = $(7A)_H$ = 122

Le code ASCII 7 bits ne reconnaît pas les lettres accentuées comme dans le latin. Pour contourner cette limitation, ce code ASCII 7 bits a été étendu à 8 bits pour coder 256 caractères différents.

II.2.2. Codage avec bit de parité

Le code de parité est le code détecteur d’erreurs le plus simple. Il consiste à introduire un seul bit supplémentaire (p) appelé bit de parité. Dans le code de parité paire, p prend ‘0’ ou ‘1’ de telle manière que le nombre total de ‘1’ soit pair. C’est ce que devra vérifier le détecteur d’erreur.

Exemple : associer un bit de parité paire p aux codes ASCII suivants :

	p	**Code ASCII**
A	**0**	1000001
D	**0**	1000100
F	**1**	1000110
4	**1**	0110100

Exemple de la transmission sur un octet (8bits) : on peut réserver 7 bits pour les données et 1 bit de parité. Donc ce code ne peut être utilisé que dans des cas de transmission où le taux d’erreur est très faible ; à titre d’exemple entre un ordinateur et une imprimante.

Chapitre 2 : Variables et fonctions logiques

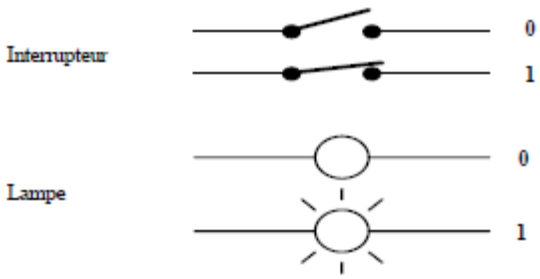
INTRODUCTION

Pour étudier les fonctions logiques on utilise une algèbre qui a été inventée au XIX^{ème} siècle par un philosophe mathématicien anglais : ***Georges Boole***. Dans ce chapitre nous nous proposons de présenter les fonctions logiques de base ainsi que leurs représentations graphiques. Nous rappellerons également les lois fondamentales de l'algèbre de Boole nécessaires à l'étude des circuits électroniques.

I. DEFINITIONS

I.1. Algèbre de Boole

C'est l'algèbre des variables qui ne peuvent prendre que deux valeurs, 0 ou 1. Ces variables sont appelées variables logiques. En pratique, on associe aux deux états d'une variable logique deux niveaux de tension :
V(0) = 0 et V(1) = 5 V pour les états 0 et 1 respectivement.



I.2. Variable booléenne ou logique

C'est une grandeur qui ne peut prendre que deux valeurs 0 ou 1.

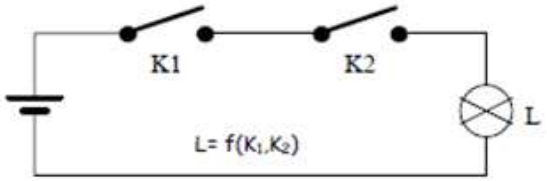
I.3. Fonction booléenne ou logique

On appelle fonction logique une manipulation de variables logiques à l'aide des opérateurs de base. Cette fonction se note : f(a_0,a_1,……, a_n). Pour la définir, il faut préciser sa valeur pour toutes les combinaisons possible. Cela peut se faire de plusieurs façons :

• Par la donnée d'une figure illustrant le fonctionnement d'un système.

• Par une définition écrite ou cahier des charges : la lampe L s'allume lorsque les interrupteurs K_1 et K_2 sont fermés tous les deux. Elle est éteinte dans tous les autres cas.

• Par la donnée de la table de vérité (*TV)*



On peut représenter toute fonction logique f à n variables sous forme d'une table de vérité à 2^n lignes et (n+1) colonnes. Les lignes représentent les combinaisons possibles des variables qui apparaissent dans l'ordre binaire naturel.

Exemple : La table de vérité relative à une fonction logique à deux variables $L(K_1,K_2)$ se compose de 3 colonnes et de 4 lignes.

Le doublet des deux variables (K_1,K_2) peut prendre 2^2 valeurs distinctes. Dans le cas général de n variables, il y aura 2^n configurations possibles.

Les variables K_1 et K_2 sont dites les entrées du système, L est dite la sortie du système.

Déc	***K_1***	***K_2***	***L***
….	….	….	L(0,0)
….	….	….	L(0,1)
….	….	….	L(1,0)
….	….	….	L(1,1)

II. OPÉRATEURS LOGIQUES

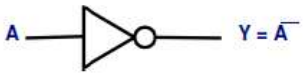
On peut montrer que toute fonction booléenne peut se synthétiser à partir d'un nombre très réduit de fonctions de deux variables ou l'extension à plusieurs variables de ces mêmes fonctions. Ces fonctions sont souvent appelées …….

II.1. Opérateur inversion NON (NOT)

La fonction ***NOT*** appelée couramment ***inverseur*** a une seule entrée et une seule sortie. La sortie d'un inverseur prend l'état 1 si et seulement si son entrée est dans l'état 0.

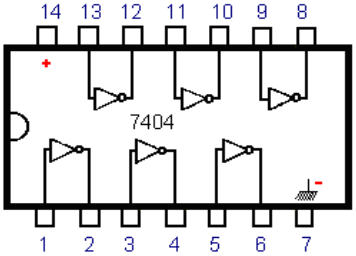
Son fonctionnement est défini par la table de vérité suivante :

A	$Y = \overline{A}$
….	….
….	….



Son logigramme est défini par le symbole graphique ci-dessus.

En pratique, l'inverseur est implanté dans le circuit intégré TTL 7404 qui comprend six inverseurs (figure n°1).



Propriété de la fonction Not

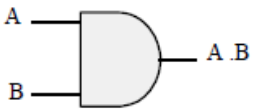
……
……
……
……
……

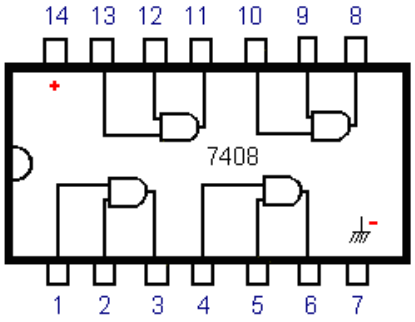
II.2. Opérateur ET (AND)

- La fonction ***AND***, encore appelée ***multiplication logique***, a deux entrées.
- *La sortie d'une fonction AND est dans l'état 1 si et seulement si toutes ses entrées sont dans l'état 1* :
- La fonction AND, notée '•' est définie par la table de vérité suivante :

A	***B***	***Y =A.B***
….	….	….
….	….	….
….	….	….
….	….	….

La fonction AND est symbolisée de la manière suivante ci dessous:





Propriété de la fonction AND

……
……
……
……
……

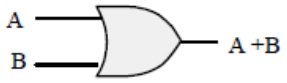
La fonction AND est implantée dans le ***CI 7408*** qui comprend 4 portes logiques AND à deux entrées.

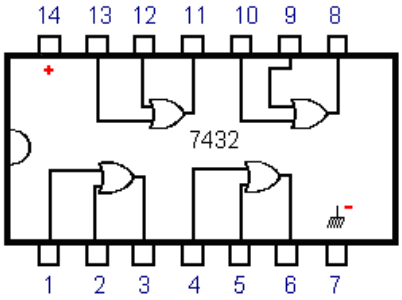
II.3. Opérateur OU (OR)

- La fonction ***OR***, encore appelée ***addition logique***, a deux entrées.
- *La sortie d'une fonction OR est dans l'état 1 si au moins une de ses entrées est dans l'état 1.*
- La fonction ***OR***, notée **+**, est définie par la table de vérité suivante :

A	***B***	***Y =A+B***
….	….	….
….	….	….
….	….	….
….	….	….

- Une porte logique OR à deux entrées est symbolisée de la manière suivante :





Propriété de la fonction OR

……
……
……
……
……
……

- La fonction OR est implantée dans le ***CI 7432*** qui comprend 4 portes logiques OR à deux entrées.

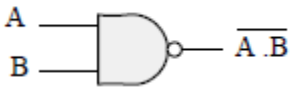
Les fonctions AND, OR et Not sont appelées : opérateurs logiques de base.

II.4. Opérateur NON-ET (NAND)

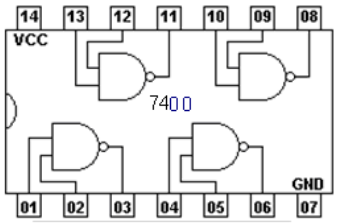
Une fonction ***NAND*** fonctionne selon la table de vérité suivante :

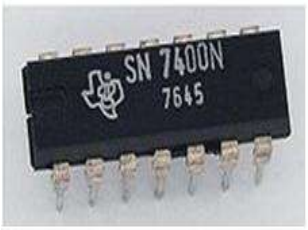
A	***B***	$Y = \overline{A.B}$
….	….	….
….	….	….
….	….	….
….	….	….

Une porte logique NAND à deux entrées est symbolisée de la manière suivante :



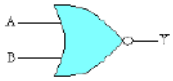
La fonction NAND est implantée dans le ***CI 7400*** qui comprend 4 portes logiques NAND à 2 entrées.





II.5. Opérateur NON-OU (NOR)

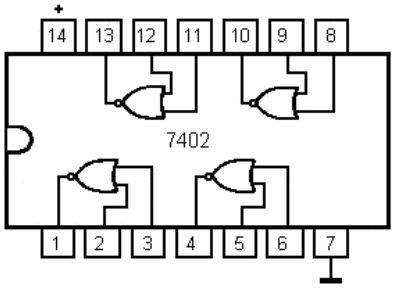
- Son logigramme est défini par le symbole graphique suivant :



- La fonction ***NOR*** a pour TV la table suivante :

A	***B***	$Y = \overline{A+B}$
….	….	….
….	….	….
….	….	….
….	….	….

- La fonction ***NOR*** est implantée dans le ***CI 7402*** qui comprend 4 portes logiques NOR à deux entrées.

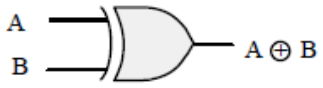


II.6. Opérateur XOR (OU exclusif)

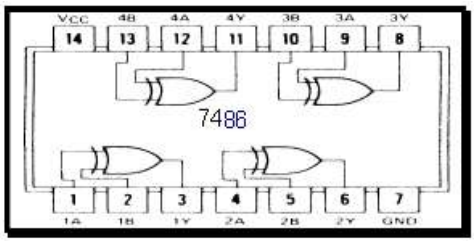
La fonction ***logique XOR*** est égal à 1 si et seulement si A = 1 ou B = 1 mais pas simultanément. Sa table de vérité est la suivante :

A	***B***	$Y = A \oplus B$
….	….	….
….	….	….
….	….	….
….	….	….

Une porte logique ***XOR*** à deux entrées est symbolisée de la manière suivante :



La fonction XOR est implantée dans le ***CI 7486*** qui comprend 4 portes logiques XOR à deux entrées.

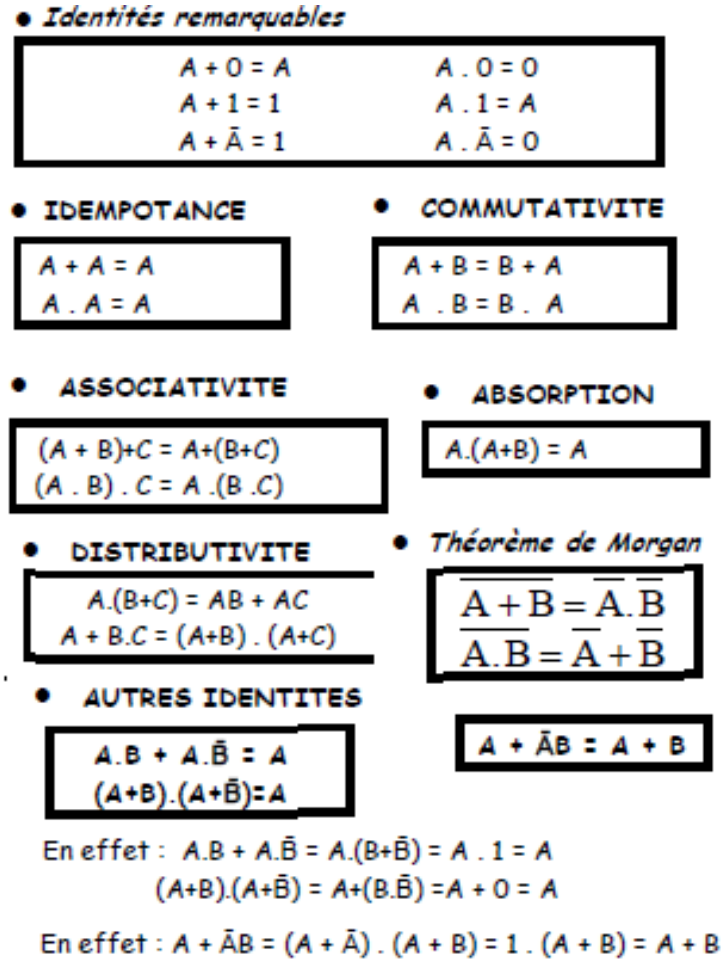


Une fonction XOR fournit un comparateur d'inégalité : XOR ne vaut 1 que si A et B sont différents. Le complément du XOR appelé XNOR correspond à un détecteur d'égalité.

La fonction XNOR est implantée dans le CI 7426 qui comprend 4 portes logiques XNOR à deux entrées.

III. LOIS FONDAMENTALES DE L'ALGEBRE DE BOOLE

L'algèbre de Boole est basée sur les lois fondamentales suivantes :



En effet : $A.B + A.\bar{B} = A.(B+\bar{B}) = A \cdot 1 = A$

$(A+B).(A+\bar{B}) = A+(B.\bar{B}) = A + 0 = A$

En effet : $A + \bar{A}B = (A + \bar{A}) \cdot (A + B) = 1 \cdot (A + B) = A + B$

IV. OPERATEURS LOGIQUES UNIVERSELLES

Nous pouvons exprimer toute fonction à l'aide de 3 opérateurs, AND, OR et NOT (complément). ***On peut démontrer que les portes logiques NAND et NOR sont universelles ou complètes ; c'est-à-dire qu'on peut réaliser à base desquelles les portes logiques de base AND, OR et NOT.*** Voir TD.

Chapitre 3 : Simplifications des fonctions logiques

INTRODUCTION

Il s'agit de chercher la fonction la plus simple d'une fonction booléenne. Ceci bien sûre dans le but d'avoir une réalisation avec un nombre minimum d'opérateurs (moins d'opérateurs, implémentation plus compacte). Il existe deux types de simplification :

- Simplification …….
- Simplification par la méthode de …….(1953)

I. FORMES CANONIQUES D'UNE FONCTION LOGIQUE

On distingue deux formes canoniques :

♣ La première forme canonique : somme de produits : $\sum(\prod)$

♣ La deuxième forme canonique : produit de sommes : $\prod(\sum)$

I.1. Première forme canonique (PFC)

La première forme canonique se réfère au *1er théorème de Shannon* qui consiste à développer toute fonction logique par rapport à l'une de ses variables sous la forme suivante :

……...

Développement par rapport à a_0.

Appliquons ce théorème à une fonction logique à deux variables :

……...
……...
……...
……...
……...

Cette façon d'écrire une fonction logique est appelée première forme canonique (PFC) qui est une somme de produits (***mintermes***).

Méthode pratique : *On peut déduire directement la PFC d'une fonction à partir de sa table de vérité en ne tenant compte que des combinaisons pour lesquelles la fonction prend **1**. Autant de **'1'** que de mintermes dans lesquels la variable apparaît sous la forme normale si elle est à **1** et sous la forme complémentée si elle est à 0.*

CBA	S(A,B,C)	
000	0	
001	0	
010	0	
011	1	$AB\bar{C}$
100	1	$\bar{A}\bar{B}C$
101	1	$A\bar{B}C$
110	1	$\bar{A}BC$
111	1	ABC

Exemple : Déduire la PFC de la fonction *S* représentée par sa table de vérité d'enface :

……

La fonction S se présente comme une somme de cinq mintermes.

Exemples :

……...
……...
……...
……...
……...
……...
……...
……...
……...

I.2. Deuxième forme canonique (DFC)

La deuxième forme canonique se réfère au *2ème théorème de Shannon* qui consiste à développer toute fonction logique par rapport à l'une de ses variables sous la forme suivante :

……...

Développement par rapport à a_0.

Appliquons ce théorème à une fonction logique à deux variables :

……...
……...
……...
……...
……...

Cette façon d'écrire une fonction logique est appelée deuxième forme canonique (DFC) qui se présente comme un produit de sommes (***maxtermes***).

Méthode pratique : *On peut déduire directement la DFC d'une fonction à partir de sa table de vérité en ne tenant compte que des combinaisons pour lesquelles la fonction prend 0. Autant de '0' que de maxtermes dans lesquels la variable apparaît sous la forme normale si elle est à 0 et sous la forme complémentée si elle est à 1.*

Exemple : Déduire la DFC de la fonction ***S*** représentée par sa table de vérité suivante :

CBA	S(A,B,C)	
000	0	(A+B+C)
001	0	($\bar{A}$+B+C)
010	0	(A+$\bar{B}$+C)
011	1	
100	1	
101	1	
110	1	
111	1	

.....

La fonction S se présente comme un produit de trois maxtermes.

Exemples :

……...
……...
……...
……...
……...
……...
……...
……...
……...

II. SIMPLIFICATION DES FONCTIONS LOGIQUES

Avant de réaliser une fonction logique sous forme d'un circuit logique, on cherche à la simplifier afin d'utiliser le moins de portes logiques possibles. Il existe deux types de simplification :

II.1 SIMPLIFICATION ALGEBRIQUE

La simplification algébrique se base sur les lois fondamentales de l'algèbre de Boole citées plus haut ainsi que les théorèmes de De Morgan.

Exemples :

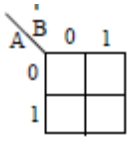
……...
……...
……...
……...
……...
……...
……...
……...
……...

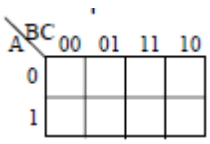
II.2. SIMPLIFICATION PAR LA METHODE DE KARNAUGH (MK)

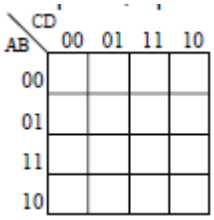
La simplification par la méthode de Karnaugh se base sur le **tableau de Karnaugh (TK)** en se servant de la propriété **d'adjacence**.

II.2.1. Tableau de Karnaugh (TK)

Un tableau de Karnaugh relatif à une fonction logique à n variables se compose ***2^n cases*** *codées selon le code binaire réfléchi*. Le TK n'est autre qu'une table de vérité disposée d'une manière astucieuse où chaque ligne de la table de vérité lui correspond une case. Les "coordonnés de la case représente l'adresse de la case. Elles sont représentées en code de Gray. Le voici pour 2, 3 puis 4 variables.





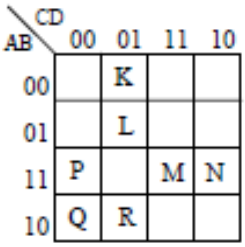


II.2.2. Propriété d'adjacence

La méthode de simplification de Karnaugh repose sur la propriété d'adjacence. *Deux cases d'un TK sont adjacentes si elles ne diffèrent que par l'état d'une variable et une seule c'est-à-dire si elles ont un coté commun.*

Sur le diagramme si contre, les cases K et L, M et N, P et Q, Q et R sont respectivement adjacentes. La propriété caractéristique du diagramme de Karnaugh est que les adresses de deux cases adjacentes sont des nombres adjacents, quand on passe de l'un à l'autre il n'y a qu'un bit qui change :

K↔0**0**01 P↔11**0**0 M↔111**1** Q↔100**0**
L↔0**1**01 Q↔1**0**00 N↔111**0** R↔100**1**

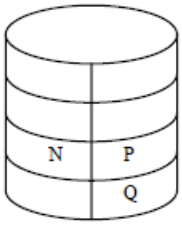


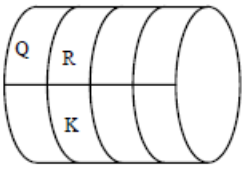
Remarquons que les cases P et N ainsi que K et R ont des adresses adjacentes :

P↔11**0**0 K↔**0**001

N↔11**1**0 R↔**1**001

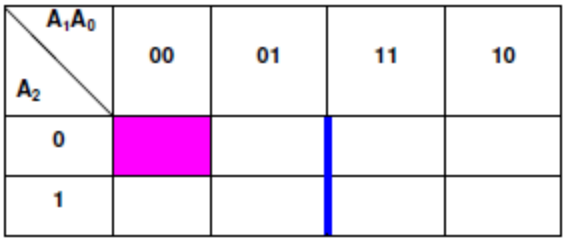
On généralise la notion d'adjacence au niveau du diagramme en disant que deux cases sont adjacentes quand leurs adresses le sont. Ainsi les cases de l'extrémité droite sont adjacentes à celles de l'extrémité gauche et les cases de l'extrémité supérieure sont adjacentes à celles de l'extrémité inférieure.



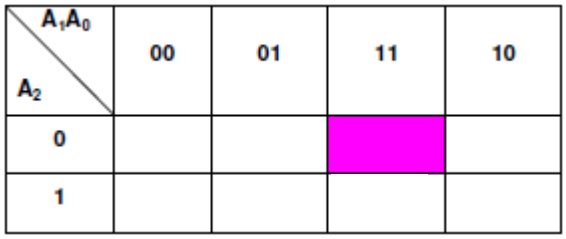


Cela se passe comme si on enroulait la feuille de papier sur laquelle est dessiné le diagramme de Karnaugh d'abord horizontalement puis verticalement.

Dans un tableau de Karnaugh *à trois variables*, chaque case a toujours trois cases qui lui sont adjacentes. Les tableaux suivants illustrent ce concept, les croix en rouge y matérialisent les cases adjacentes de la case coloriée en violet. Donc, deux cases sont adjacentes si elles sont voisines l'une de l'autre ou symétriques par rapport à un axe de symétrie représenté en bleu.

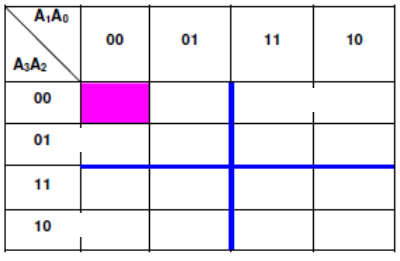


Deux par voisinage et une par symétrie.

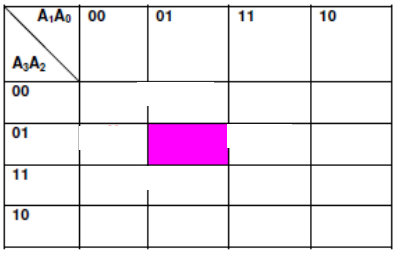


Trois par voisinage

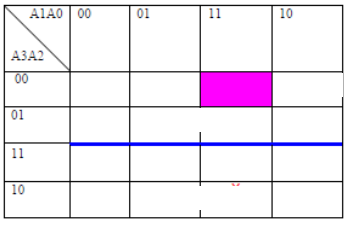
Dans un tableau de Karnaugh à *quatre variables*, chaque case a toujours quatre cases qui lui sont adjacentes. Les tableaux suivants illustrent ce concept, les croix en rouge y matérialisent les cases adjacentes de la case coloriée en violet.



2 par voisinage et 2 par symétrie.



Quatre par voisinage



3 par voisinage et 1 par symétrie

II.2.3. Procédé de simplification par la méthode de Karnaugh

La méthode de simplification de Karnaugh consiste à regrouper les cases adjacentes contenant des 1 par groupes de deux '1' (doublets), quatre '1' (quartets) ou huit '1' (octets).

Le procédé de simplification de Karnaugh se base sur les règles suivantes :

Règle 1 :
- Pour un '1' isolé, on ne peut éliminer aucune variable.
- Dans un doublet, on élimine une seule variable ; celle qui change d'état.
- Dans un quartet, on élimine deux variables ; celles qui changent d'état.
- Dans un octet, on élimine trois variables ; celles qui changent d'état.

D'où l'importance de former des groupes de '1' adjacents aussi importants que possible.

Règle 2 :
Un même '1' peut intervenir dans plusieurs groupements car x + x = x.
Le procédé de simplification d'une fonction logique par la méthode de Karnaugh dans l'ordre est le suivant :
♣ Identifier les '1' dits isolés (ne sont adjacents à aucun autre '1'),
♣ Regrouper les doublets uniques éventuels,
♣ Regrouper les octets éventuels même si des 1 se trouvent déjà dans des doublets uniques,
♣ Regrouper les quartets éventuels qui ont au moins un 1 qui n'a pas déjà été regroupé,
♣ Regrouper des '1' restants dans des doublets simples
♣ Ecrire l'équation associée à f sous sa première forme canonique simplifiée en associant à chaque groupement un terme.

Remarques
♣ Pour les fonctions avec un grand nombre de 1: On fait de regroupement de 0 et on leur associe un « maxterme ». On fait une conjonction de ces maxtermes.
♣ Si une combinaison d'entrée ne peut pas se présenter ou si pour cette combinaison la valeur de la fonction n'est pas importante, on dit que la fonction n'est pas définie en ce point :
F(a,b,c) = ϕ ou 'x' ou '-' qui se lit « don't care »
Ce point peut être remplacé par 1 ou 0 en fonction des besoins de simplification.

Exemples :
……...
……...
……...
……...
……...
……...
……...
……...
……...
……...
……...

II.2.4. Code Interdit ou quand la valeur compte peu

Etudions le cas suivant : On désire maintenir la température d'une salle de travail entre 17°C et 20°C. Nous nous procurons deux capteurs de température ayant chacun une sortie logique que nous appellerons C_{17} et C_{20} . Chaque capteur fonctionne de la manière suivante :

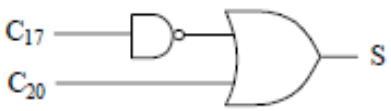
C_{17} = 0 si la température est < à T_{17} , C_{17} = 1 si la température est > à T_{17}
C_{20} = 0 si la température est < à T_{20} , C_{20} = 1 si la température est > à T_{20}

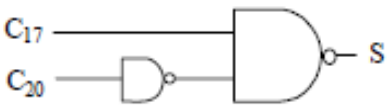
Exercice 1

On va donc essayer de construire un système logique qui délivre une alarme S chaque fois que la température sort de l'intervalle [17°,20°]. Ce qui revient à chercher la fonction S des deux variables C_{17} et C_{20}, $S = f(C_{17},C_{20})$.

Réponse :

On obtient : $\boxed{S = \overline{C}_{17} + C_{20}}$ pour le détail voir cours.





II.3. EXEMPLE

Prenons l'exemple d'une fonction f définie par le tableau de Karnaugh suivant :

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

Nous y observons trois groupements : un quartet et deux doublets, nous pouvons ainsi écrire l'équation de la fonction f sous la forme d'une somme de trois termes :

Exercice 2 : *Convertisseur Binaire naturel vers code de Gray*

D	C	B	A	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

On voit sur la table de vérité que G_3=D.

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

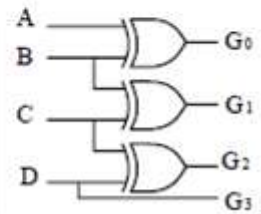
$G_2=C\bar{D}+\bar{C}D=C\oplus D$

DC \ BA	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$G_1=B\bar{C}+\bar{B}C=B\oplus C$

DC \ BA	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$G_0=A\bar{B}+\bar{A}B=A\oplus B$



Chapitre 4 : Les circuits combinatoires usuels

L'objectif de ce chapitre est de connaître les circuits élémentaires :

- Fonctions arithmétiques : additionneurs, comparateurs.
- Fonctions combinatoires : multiplexeurs, démultiplexeurs, décodeurs, codeurs.

I. LES CIRCUITS ARITHMÉTIQUES

I.1. ADDITION ARITHMÉTIQUE

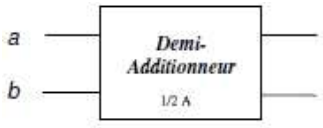
L'addition arithmétique en binaire est effectuée de la même façon que l'addition décimale, avec une ***table d'addition*** plus simple puisqu'il n'y a que quatre cas qui peuvent se manifester lorsqu'on additionne des bits qui ne prennent que deux valeurs 0 ou 1 :

0 + 0 = 0 0 + 1 = 1 1 + 0 = 1

1 + 1 = 0 + retenue de 1 sur le bit de rang supérieur.

I.1.1. Demi Additionneur (1/2 A).

Un demi additionneur *est un circuit logique combinatoire capable d'effectuer l'addition arithmétique de deux bits a et b*. La figure n°1 illustre son symbole, son circuit logique et sa table d vérité. Les sorties sont la somme (S) et la retenue (r).



b	*a*	*S*	*r*
0	0	….	….
0	1	….	….
1	0	….	….
1	1	….	….

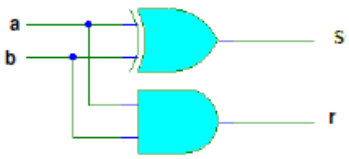


Figure n°1 : Schéma symbolique d'un demi-additionneur, TV et C.L

On déduit facilement de cette table de vérité les équations logiques suivantes :

……
……

On obtient ainsi le circuit ci-dessus pour un demi-additionneur

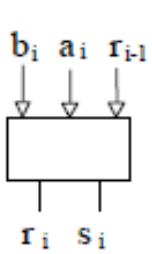
I.1.2. Additionneur complet ou élémentaire (AE).

Un additionneur complet ou élémentaire est un circuit logique combinatoire capable d'effectuer l'addition arithmétique de trois bits a_i et b_i et r_{i-1} :

- a_i et b_i sont les entrées de l'étage i et r_{i-1} est la retenue de l'étage i-1.
- La figure n°2 illustre son symbole, sa TV et son C.L.
- On déduit facilement de cette TV les équations logiques suivantes :

$$S_i = a_i \oplus b_i \oplus r_{i-1} \quad et \quad r_i = a_i.b_i + r_{i-1}.(a_i \oplus b_i) = a_i.b_i + r_{i-1}.(a_i + b_i) \quad et \quad r_{i-1} = a_{i-1}.b_{i-1}$$

Pour l'expression de $\mathbf{r_i}$, on a fait exprès de ne pas choisir la fonction la plus simple sur la table de Karnaugh afin d'avoir le terme ai⊕bi en commun avec l'expression de $\mathbf{S_i}$ ce qui permettra une réalisation plus économique (Fig. 2).



si

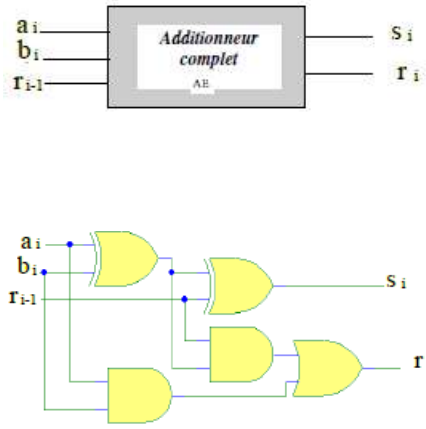
r_{i-1} \ $b_i a_i$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

ri

r_{i-1} \ $b_i a_i$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

…… ……
…… ……
…… ……
…… ……
…… ……
…… ……

On obtient ainsi le circuit suivant pour un Additionneur, élémentaire, complet



r_{i-1}	b_i	a_i	S_i	r_i
0	0	0	….	….
0	0	1	….	….
0	1	0	….	….
0	1	1	….	….
1	0	0	….	….
1	0	1	….	….
1	1	0	….	….
1	1	1	….	….

Figure n°2 : Schéma symbolique d'un additionneur complet, TV et C.L

I.1.3. Additionneur de deux nombres.

I.1.3.1. Additionneur à, retenue série, propagation de la retenue

Quand on additionne "manuellement" deux mots à n bits A et B, on refait n fois l'addition des bit du même poids en faisant attention de ne pas oublier d'inclure dans cette addition le reste de l'addition précédente. Donc la réalisation d'un additionneur de deux mots revient à cascader des additionneurs 3 bits.

	r_{n-2}	r_1	r_o		
	a_{n-1}	a_2	a_1	a_o	
	b_{n-1}	b_2	b_1	b_o	+
r_{n-1}	s_{n-1}	s_2	s_1	s_o	

L'addition de deux nombres binaires à n bits nécessite la mise en cascade de n étages additionneurs complets. Ces étages sont raccordés en connectant la retenue sortante de l'étage i à la retenue entrante de l'étage i+1. A titre d'illustration, voyons la figure n°3 où quatre additionneurs complets sont raccordés en cascade afin de pouvoir additionner deux nombres A et B à 4 bits chacun : $A = (a_3a_2a_1a_0)_2$ $B = (b_3b_2b_1b_0)_2$

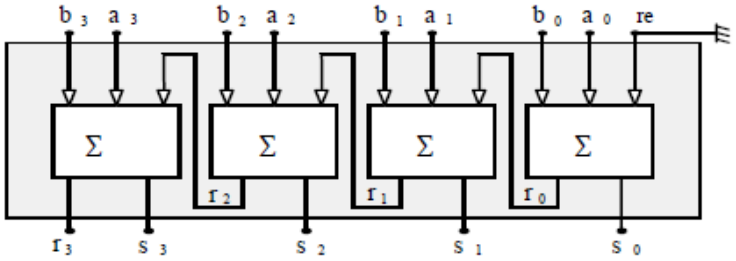


Figure n°3 : Additionneur, de deux mots de 4 bits, à retenue série

Ce genre d'additionneur est dit à propagation de la retenue ou a retenue série, car chaque étage doit "attendre" que l'étage précédent "termine" son calcul pour lui fournir le reste. Plus le nombre de bits est grand plus le délai de calcul est important, pour cette raison ce genre de circuit n'est guerre utilisé dans des applications professionnelles.

Autrement dit, les retards de propagation introduits par les étages s'additionnent de sorte que la retenue $\mathbf{r_n}$ n'est obtenue qu'après un temps de n□□. Evidemment, à mesure qu'augmente le nombre de bits, la durée nécessaire pour obtenir le résultat d'une addition augmente. Cette limitation est contournée en utilisant des additionneurs à retenue anticipée qui sont plus rapides au prix d'une complexité plus grande.

I.1.4. Additionneur intégré.

Le 7482 (***Fig. 6***) est un additionneur à retenue série de deux mots de 2 bits. Les sommes est les retenues sont calculées d'une façon assez originale pour en améliorer les performances. r_e est la retenue entrante, r_0 n'est pas accessible, r_1 = retenue de la somme de a_1 et b_1 est la retenue sortante, elle sert éventuellement à propager la retenue vers un autre additionneur.

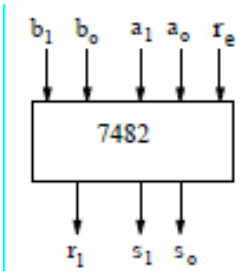


Fig. 6 : additionneur 2 mots de 2 bits

I.2. COMPARATEURS = CIRCUITS D'IDENTIFICATION

Un comparateur est un circuit, d'identification, combinatoire qui effectue la comparaison de deux nombres binaires A et B. Les trois résultats de comparaison des deux nombres A et B S(A>B), E(A=B) et I(A<B) sont directement disponibles en sortie. La figure 12 représente le symbole d'un comparateur :

- S : fonction supériorité.
- I : fonction infériorité.
- E : fonction égalité.

I.2.1. Comparateur de 2 bits .

Table de Vérité

b_i	a_i	$S_i(a_i>b_i)$	$I_i(a_i<b_i)$	$E_i(a_i=b_i)$
0	0	0	0	1
0	1	1	1	0
1	0	0	0	0
1	1	0	0	1

Symbole d'un comparateur à 2 bits

……

……

Au chapitre II, nous avons vu qu'on peut facilement détecter l'égalité de deux bits x et y en utilisant la fonction inverse de XOR, notée : $f = \overline{x \oplus y}$.

I.2.2. Comparateur de deux nombres de 3 bits chacun.

Soient A et B deux nombres binaires que nous supposerons à 3 bits chacun.

I.2.2. 1. Fonction égalité : E(A=B)

A et B sont égaux si tous les bits ayant le même poids sont égaux deux à deux.

$A = (a_2\ a_1\ a_0)$ *et* $B = (b_2\ b_1\ b_0) \Rightarrow A = B$ *si* $a_2 = b_2$ *et* $a_1 = b_1$ *et* $a_0 = b_0$

Au chapitre II, nous avons vu la fonction inverse de XOR, notée : $E_i = \overline{a_i \oplus b_i}$ *est une fonction d'égalité*. Donc :

……

……

I.2.2. 2. Fonction infériorité : I(A<B)

$A = (a_2\ a_1\ a_0)$ *et* $B = (b_2\ b_1\ b_0)$

$A<B$ *si* $a_2<b_2$ *ou* $(a_2 = b_2$ *et* $a_1 < b_1)$ *ou* $(a_2 = b_2$ *et* $a_1 = b_1$ *et* $a_0 < b_0)$. Donc

……

……

I.2.2. 2. Fonction supériorité : S(A>B)

$A = (a_2\ a_1\ a_0)$ et $B = (b_2\ b_1\ b_0)$

$A>B$ si $a_2>b_2$ ou ($a_2= b_2$ et $a_1 >b_1$) ou ($a_2= b_2$ et $a_1 =b_1$ et $a_0 >b_0$). Donc

……

……

On obtient ainsi le circuit du comparateur des deux nombres A et B à 3 bits chacun (Fig. 13).

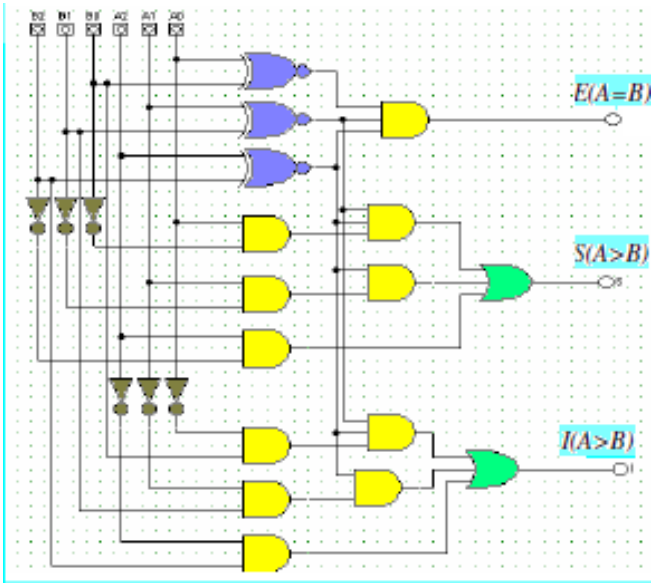


Figure n°13 : Schéma logique du comparateur des deux nombres A et B à 3 bits chacun

***I.2.3. Comparateur intégré 74LS85* :**

Le C.I. 7485 effectue la comparaison de deux nombres à 4 bits chacun. De plus, il dispose de 3 entrées notées **A = B, A > B et A < B** qui autorisent la mise en cascade de plusieurs circuits comparateurs du même type. Ainsi, on peut comparer des nombres de 8, 12, 16 bits.... Le brochage de ce circuit est donné à la figure 14, tandis que la figure 15 représente son schéma logique.

Avec ce circuit, on compare le nombre **A** composé des bits **A3, A2, A1** et **A0** (**A3 = MSB** et **A0 = LSB**) avec le nombre **B** composé des bits **B3, B2, B1** et **B0** (**B3 = MSB** et **B0 = LSB**).

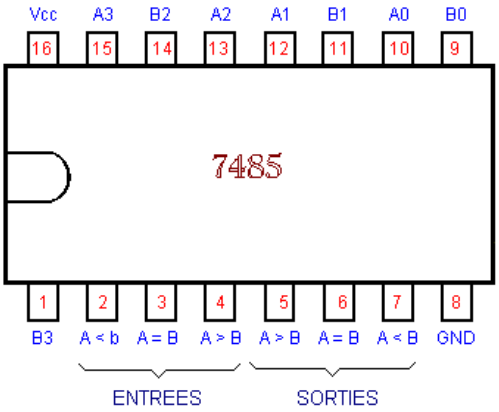


Figure n°14 : Brochage du C.I. 7485

II. LES CIRCUITS COMBINATOIRES (multiplexage et décodage)

Mis à part les circuits arithmétiques, d'autres circuits combinatoires jouent un rôle très important dans diverses applications, en particulier les décodeurs et les multiplexeurs.

II.1. LES MULTIPLEXEURS (CIRCUITS D'AIGUILLAGE).

II.1.1. Choix d'une voie (entrée) parmi N.

Un multiplexeur est un circuit combinatoire qui effectue l'aiguillage des entrées vers la sortie en fonction des entrées de sélection (adresse). ***Un Multiplexeur 1 parmi N ou 2^n vers 1*** (avec la relation $\mathbf{2^n \geq N}$).dispose de 2^n entrées, n entrées de sélection et une seule sortie.

La figure 18 illustre le schéma symbolique d'un multiplexeur $2^n \rightarrow 1$ ayant 2^n entrées, n entrées de sélection et une seule sortie S. Si l'adresse est 0, alors E_0 apparaît à la sortie, si l'adresse est 1, alors E_1 apparaît en sortie, etc …..

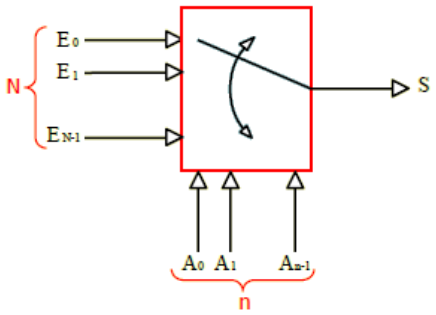


Fig. 18 : Multiplexeur 1 parmi N (1/N)

Pour choisir une voie parmi N, il faut n entrées d'adressage avec la relation $\mathbf{2^n \geq N}$. A chaque instant la sortie S est égale (connectée) à l'entrée E "pointée" par le mot adresse A_{n-1} ... A_1A_0.

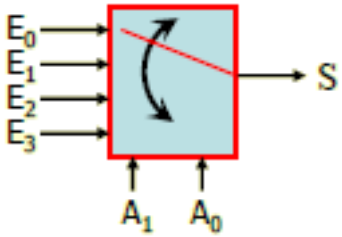
- 1 MUX 1/4 a 4 entrées + 2 entrées d'adresse.
- 1 MUX 1/8 a 8 entrées + 3 entrées d'adresse
- 1 MUX 1/10 a 10 entrées + 4 entrées d'adresse
- 1 MUX 1/16 a 16 entrées + 4 entrées d'adresse

Exemple : multiplexeur 4 vers 1 (MUX 1 parmi 4, 1/4)

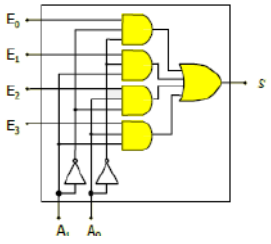
Un multiplexeur 4 vers 1 est un multiplexeur à 4 entrées E_0, E_1, E_2 et E_3 et 2 entrées adresse A_0 et A_1 et S la sortie.

L'expression logique de la sortie est : ……

La figure ci-dessous illustre le symbole de ce multiplexeur. L'aiguillage des entrées E_0, E_1, E_2 et E_3 vers la sortie S est commandé par les entrées de sélection A_0 et A_1 selon la table de vérité suivante. On obtient ainsi le circuit suivant pour un multiplexeur 4 vers 1 :



A_1	A_0	S
0	0	…
0	1	…
1	0	…
1	1	…



Pour réaliser des multiplexeurs qui ont un grand nombre d'entrées, on peut utiliser de "petits" multiplexeurs montés en pyramide. (Fig. 19)

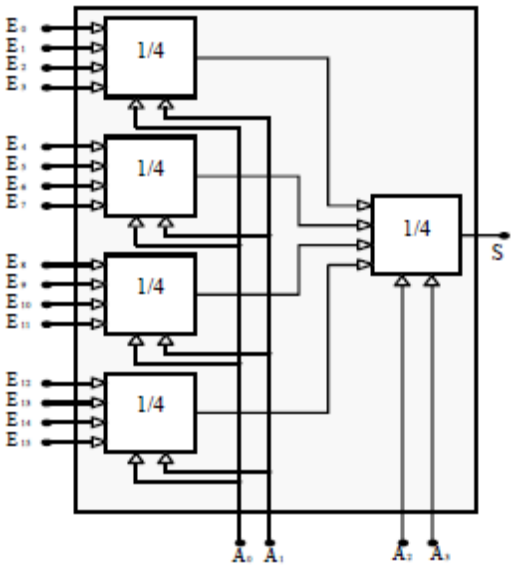


Fig. 19 : Multiplexeur 1 parmi 16 (voir TD)

II.1.3. Application 1 : Génération de fonctions

Mise à part la fonction d'aiguillage, les multiplexeurs permettent de générer n'importe quelle fonction logique. ***A titre d'exemple, prenons le cas d'un multiplexeur 4 vers 1.***

Un multiplexeur 4 vers 1 dispose de quatre entrées E_3-E_0 et deux entrées de sélection A_1-A_0. La sortie S a pour équation : $S = \overline{A}_0\overline{A}_1E_0 + A_0\overline{A}_1E_1 + \overline{A}_0A_1E_2 + A_0A_1E_3$

S est une forme complète d'une fonction à deux variables. Donc ce multiplexeur permet de générer n'importe quelle fonction à deux variables. Supposons qu'on veut générer à l'aide d'un multiplexeur 4 vers 1 la fonction : $F = \overline{A}\overline{B} + \overline{A}B$ *Fig. 22*

$S = F \Leftrightarrow A_0 = A$ *et* $A_1 = B$ *et* $E_0 = 1$ *et* $E_1 = 1$ *et* $E_2 = 0$ *et* $E_3 = 0$.

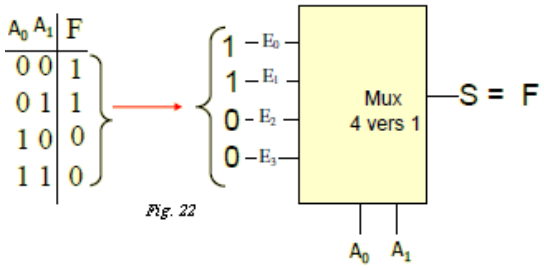


Fig. 22

Il suffit donc, de connecter les variables A et B aux entrées de sélection A_0 et A_1 d'imposer '1' aux entrées E_0 et E_1 et '0' aux entrées E_2 et E_3 comme le montre la Fig. 22.

II.2. LES DEMULTIPLEXEURS (DMUX).

Un démultiplexeur est un circuit combinatoire qui effectue l'aiguillage de l'entrée vers une des N sorties en fonction des entrées de sélection (adresse). Un démultiplexeur 1 vers 2^n (avec la relation $\mathbf{2^n \geq N}$).dispose d'une seule entrée, n entrées de sélection et 2^n sorties.

La figure 29 illustre le schéma symbolique d'un démultiplexeur 1→ N ayant N sorties S_{N-1} ... S_1S_0, n entrées de sélection et une seule entrée E.

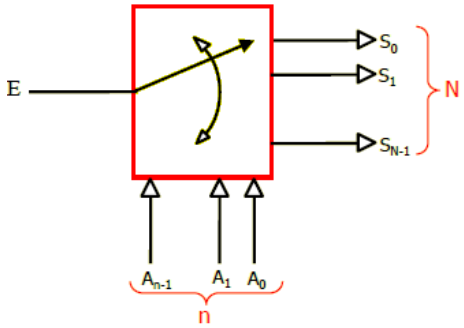


Fig. 29 : Démultiplexeur 1 vers N

- 1 DMUX 1→4 a une entrée, 4 sorties + 2 entrées d'adresse.

- 1 DMUX 1→8 a une entrée, 8 sorties + 3 entrées d'adresse.

- 1 DMUX 1→10 a une entrée, 10 sorties + 4 entrées d'adresse.

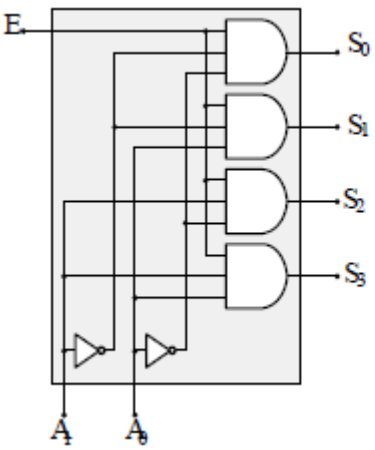
- 1 DMUX 1→16 a une entrée, 16 sorties + 4 entrées d'adresse

II.2.2. Le Démultiplexeur à 4 voies (Dmux 1→ 4)

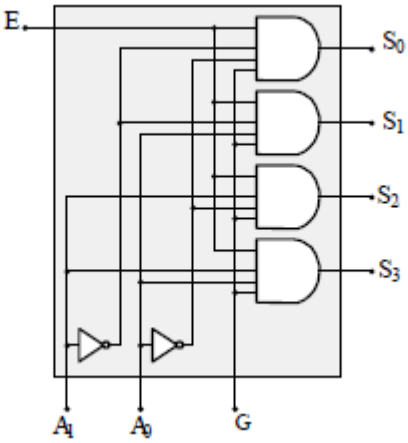
A_1	A_0	S_3	S_2	S_1	S_0
0	0	0	….	….	….
0	1	0	….	….	….
1	0	0	….	….	….
1	1	E	….	….	….

On se propose de réaliser un démultiplexeur à 4 sortie S3, S2, S1, S0, une entrée E et deux bits d'adresse A0, A1. ***Les sorties non sélectionnées sont à l'état bas '0'***.

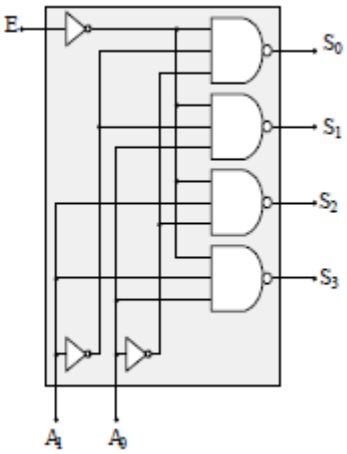
$$S_0 = \overline{A}_0\overline{A}_1E$$
$$S_1 = A_0\overline{A}_1E$$
$$S_2 = \overline{A}_0A_1E$$
$$S_3 = A_0A_1E$$



(a) : démultiplexeur 1/4



(b) démux 1→ 4 avec entrée de validation G



(c) : démux 1→4 avec entrées non sélectionnées = "H"

Fig. 33: variantes de démultiplexeur

Le schéma de Fig. 33b montre un démultiplexeur avec entrée de validation G :

- G=0 ⇒ toutes les sortie sont à 0 ie "L' ∀ l'état de E et des adresses.
- G=1 ⇒ Le circuit fonctionne en démultiplexeur normal. :
- $S_0 = G\overline{A}_0\overline{A}_1E \quad S_1 = GA_0\overline{A}_1E \quad S_2 = G\overline{A}_0A_1E \quad S_3 = GA_0A_1E$

Etudions maintenant un DMUX 1→4 dont les sorties non sélectionnées sont à l'état haut '1'. Si on rajoute des inverseurs à la sortie du DMUX de Fig. 33a (ce qui revient à remplacer les AND par des NAND), les sorties sélectionnées sont "L", '=0' mais la sortie sélectionnée est égale au complément de E, il faut donc inverser l'entrée aussi. On obtient le DMXR de la figure Fig. 33c

II.3. LES DECODEURS.

Un décodeur est un circuit logique combinatoire qui fait correspondre à un code en entrée une seule sortie active parmi plusieurs (max 2^n). Ce décodeur est appelé 1 parmi 2^n.

La figure 36 nous montre le schéma symbolique d'un décodeur binaire ayant n entrées et 2^n sorties. Quant le code d'entrée prend la valeur i, la sortie Si est active.

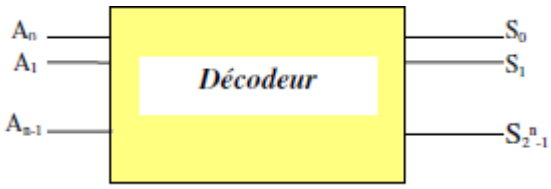


Fig. 36 : décodeur binaire à n entrées

II.3.1. Décodeur à 2 entrées et 4 sorties (1 parmi 4)

Les décodeurs sont des démultiplexeurs particulier :

- Si la sortie sélectionnée est à l'état bas, les autres sont à l'état haut. On peut utiliser le circuit de Fig. 33c et on relie E à la masse ce qui revient à supprimer cette entrée et on obtient le schéma de Fig. 37b.

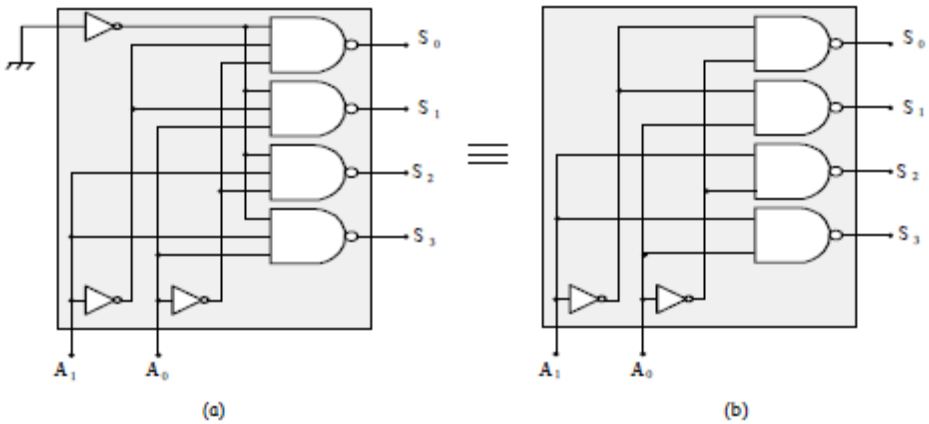


Fig. 37 : Décodeur 1 parmi 4 déduit de DMUX 1 vers 4 avec sorties non sélectionnées sont à l'état haut

- Si la sortie sélectionnée est à l'état haut, les autres sont à l'état bas. On peut utiliser le circuit de Fig. 33a et on relie E à la masse ce qui revient à supprimer cette entrée et on obtient le schéma de Fig. 38b.

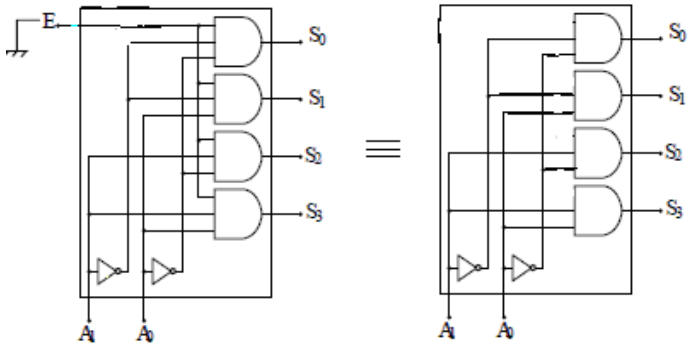
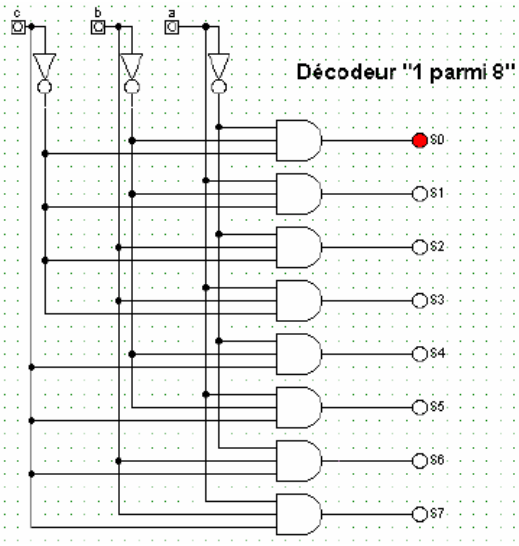


Fig. 38 : Décodeur 1 parmi 4 déduit de DMUX 1 vers 4 avec sorties non sélectionnées sont à l'état bas

II.3.2. Décodeur à 3 entrées et huit sorties (1 parmi8)

On cherche à réaliser un décodeur 1 parmi 8 qui a un code d'entrée sur 3 bits (c b a)$_2$ et huit sorties S_0 → S_7. Son fonctionnement obéit à la table de vérité suivante :

c	b	a	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



On en déduit les équations et le circuit suivants :

……
……
……
……
……
……

II.3.3. Application : décodage mémoire

Une mémoire peut être considérée comme un assemblage de registres (zones de rangement), chaque registre contient un mot mémoire (donnée, instruction) et possède un numéro d'ordre, qui est son adresse. Supposons qu'on veut lire ou écrire un mot en mémoire; on envoie l'adresse i du registre à sélectionner et le décodeur active la sortie nécessaire à la sélection du registre correspondant (figure 39).

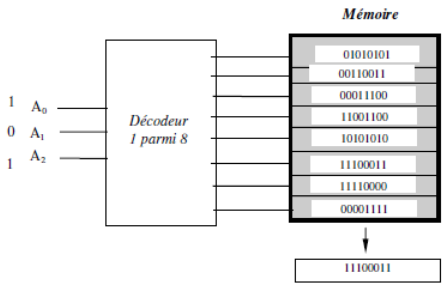


Fig. 39 : Décodage mémoire

II.3.4. Application : Décodeur DCB - 7 segments (Voir TD)

Les afficheurs les plus couramment utilisés pour l'affichage numérique sont les afficheurs sept segments qui ne sont rien d'autre qu'une association de 7 diodes (LEDs) disposées comme le montre la figure Fig. 40. On distingue deux types d'afficheurs, les afficheurs à Anodes communes (AC) et les afficheurs à cathodes communes (CC).

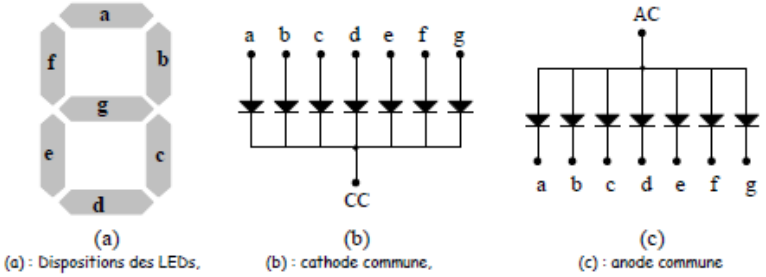
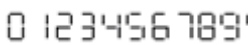


Fig. 40 : Afficheur sept segments

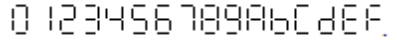
Les afficheurs cathode commune se commandent par niveau haut '1' et ceux à anode commune se commandent par niveau bas '0'.

Les nombres à afficher sont codés en BCD, chaque digit est codé en binaire sur 4 bits. Le rôle du décodeur BCD-7segment et de générer à partir du code binaire DCB d'un chiffre, la configuration adéquate des entrée a, b, c, d, e, f et g de l'afficheur afin d'allumer les LEDs qui forment le chiffre considéré.

Faisons l'étude d'un décodeur pour afficheurs cathode commune. Les chiffres générés par ce décodeur sont :



Il parait évident que ce décodeur ne doit être utilisé que pour des nombres d'entrées > 9. On peut étendre l'utilisation de ce genre de décodeur en affectant des symboles (caractères) aux combinaisons d'entrée 10, 11, 12, 13, 14 et 15. On peut par exemple étudier un décodeur BCH-7segment (Hexadécimal codé en binaires), ce décodeur générera les fontes suivantes :



On obtient les expressions suivantes pour les différents segments ce qui donne le décodeur représenter sur la figure Fig. 42.

$$a = AB + D + AC + \overline{A}\,\overline{C}$$

$$b = \overline{C} + \overline{A}\,\overline{B} + AB$$

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

$$d = D + \overline{A}B + B\overline{C} + \overline{A}\,\overline{C} + A\overline{B}C$$

$$e = \overline{A}B + \overline{A}\,\overline{C}$$

$$f = D + C\overline{A} + \overline{A}\,\overline{B} + C\overline{B}$$

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

D	C	B	A	Dec	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1
1	0	1	0	10	x	x	x	x	x	x	x
1	0	1	1	11	x	x	x	x	x	x	x
1	1	0	0	12	x	x	x	x	x	x	x
1	1	0	1	13	x	x	x	x	x	x	x
1	1	1	0	14	x	x	x	x	x	x	x
1	1	1	1	15	x	x	x	x	x	x	x

Fig. 41 : table de vérité d'un décodeur BCD 7 segment CC

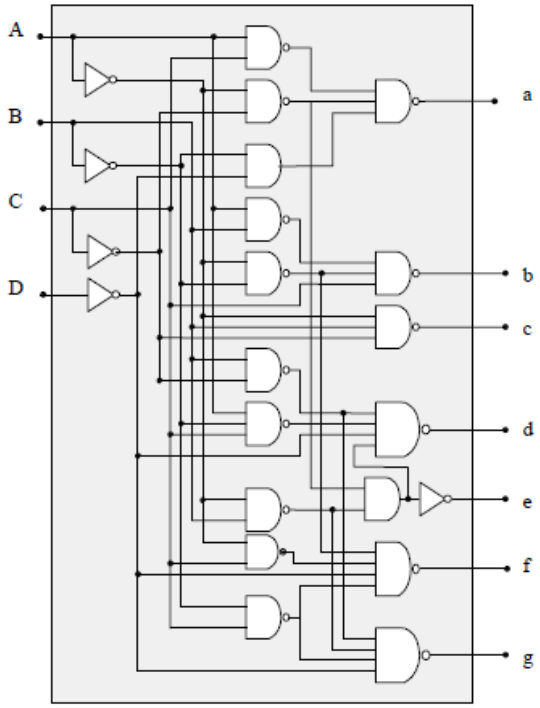


Fig. 42 : Décodeur BCD-7segments pour afficheurs CC

Le tableau ci-dessous fournit l'état des segments d'un afficheur AC pour les différentes combinaisons d'entrée.

D	C	D	A	Dec	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	1	1	1	1
0	0	1	0	2	0	0	1	0	0	1	0
0	0	1	1	3	0	0	0	0	1	1	0
0	1	0	0	4	1	0	0	1	1	0	0
0	1	0	1	5	0	1	0	0	1	0	0
0	1	1	0	6	0	1	0	0	0	0	0
0	1	1	1	7	0	0	0	1	1	1	1
1	0	0	0	8	0	0	0	0	0	0	0
1	0	0	1	9	0	0	0	0	1	0	0
1	0	1	0	A	0	0	0	1	0	0	0
1	0	1	1	B	1	1	0	0	0	0	0
1	1	0	0	C	0	1	1	0	0	0	1
1	1	0	1	D	1	0	0	0	0	1	0
1	1	1	0	E	0	1	1	0	0	0	0
1	1	1	1	F	0	1	1	1	0	0	0

Fig. 43 : table de vérité d'une décodeur BCH 7 segment AC

$a = A\overline{B}\overline{C}\,\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}CD + AB\overline{C}\,D$

$b = \overline{A}BC + \overline{A}CD + A\overline{B}C\overline{D} + ABD$

$c = \overline{A}B\overline{C}\,\overline{D} + \overline{A}CD + BCD$

$d = \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\,D + A\overline{B}\overline{C}\,\overline{D} + ABC$

$e = \overline{B}C\overline{D} + A\overline{B}\overline{C} + A\overline{D}$

$f = A\overline{C}\,\overline{D} + A\overline{B}CD + B\overline{C}\,\overline{D} + AB\overline{D}$

$g = \overline{A}\overline{B}CD + \overline{B}\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D}$

II.5. LES CODEURS.

Un codeur est un circuit logique combinatoire ayant un certain nombre d'entrées dont une seule est active à la fois. Chaque entrée active lui correspond un code en sortie. Autrement dit, il réalise l'opération inverse de celle du décodeur. La figure 46 illustre le schéma symbolique d'un codeur binaire ayant 2^n entrées et un code en sortie de n bits.

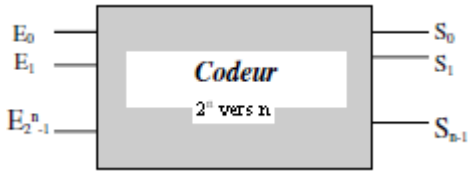


Fig. 46 : Codeur binaire à 2^n entrées et n sorties

Exemple : codeur décimal - DCB

On cherche à réaliser un circuit qui fait correspondre à chaque chiffre décimal son équivalent binaire codé sur 4 bits. C'est un circuit à 10 entrées représentant les chiffres décimaux de 0 à 9 et 4 sorties représentant les 4 bits DCB.

Il fonctionne selon la table de vérité suivante :

E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	A3	A2	A1	A0
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

On en déduit les équations suivantes :

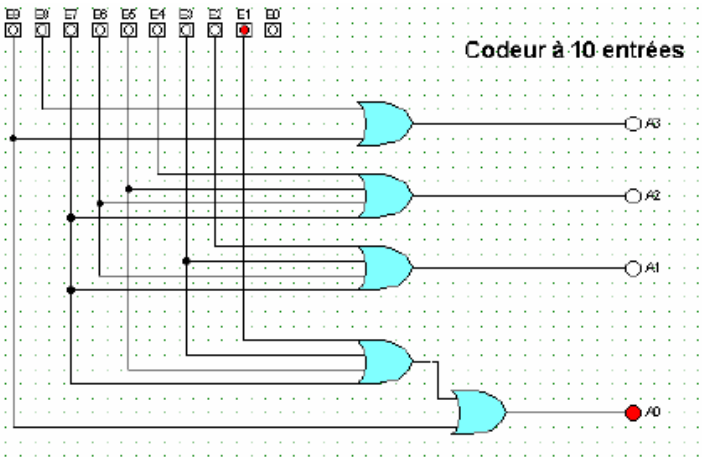
$A_3 = E_8 + E_9$

$A_2 = E_4 + E_5 + E_6 + E_7$

$A_1 = E_2 + E_3 + E_6 + E_7$

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

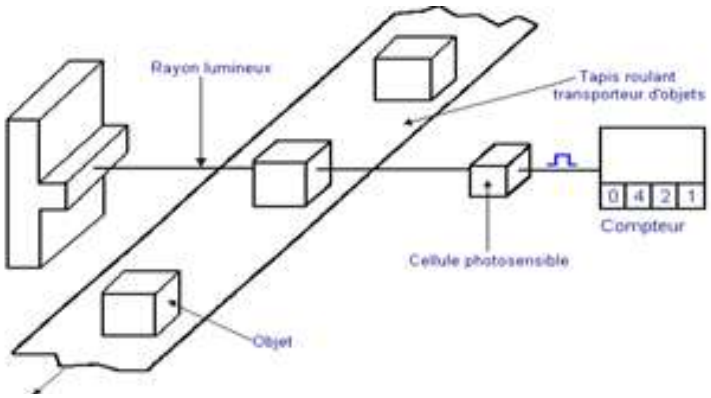
Ce qui donne le schéma suivant:

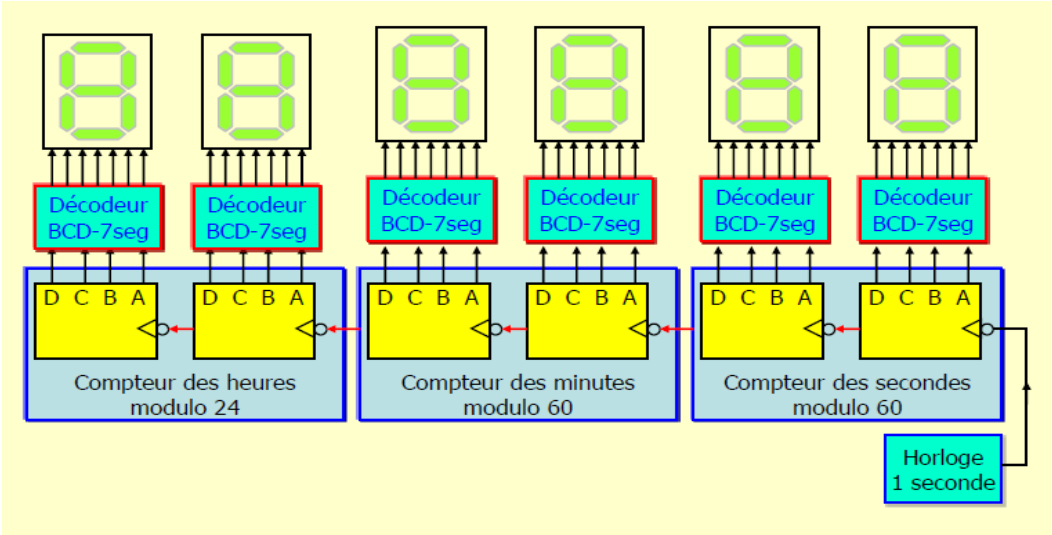


Exemple d'application d'un codeur décimal - DCB :

L'opérateur choisit le nombre d'exemplaires désiré en appuyant sur une touche d'un clavier décimal d'une photocopieuse. Le codeur décimal - DCB ci-dessus génère en sortie le code DCB.

جامعة سيدي محمد بن عبدالله
كلية العلوم ظهر المهراز
Université Sidi Mohamed Ben Abdellah
Faculté des Sciences Dhar Mahraz





DEPARTEMENT DE PHYSIQUE

L.P.S.

Logique séquentielle

Semestre 3 – Module électronique M 20 - Filière SMI

Année universitaire 2014-2015

Réalisé par :

Pr. REZZOUK Abdellah

Partie 2 : Logiques séquentielles

Chapitre 1 : Les bascules.

Chapitre 2 : compteur asynchrone.

Chapitre 3 : compteur synchrone.

Chapitre 4 : Registre à décalage.

Bibliographie

- Circuits numérique : Théorie et application, R. TOCCI, DUNOD-Gould Paris, 2ème version.
- Eléctronique numérique : Tome 1(Logique combinatoire) et TOME 2 (Logique séquentiel), R. TOCCI, DUNOD-Gould Paris.
- Pratique de l'électronique Numérique, Pierre PELLOSO, DUNOD.
- Circuits numérique, Remy Letocha, McGraw-Hill, éditeurs.

Introduction

Nous avons fait le tour des principaux circuits logiques combinatoire, à partir de ça, on peut commencer à faire des microprocesseurs mais il va falloir pouvoir mémoriser des chose : c 'est la Logique séquentielle.

Circuits de logique séquentielle : circuits dans lesquels le temps intervient dans la définition des sorties.

L'élément le plus simple est la bascule : qui est une combinaison des portes logiques et a la possibilité de mémoriser une donnée digitale (0 ou 1).

Il existe de nombreux types de bascules, mais elles sont classées en 2 catégories :

● ***Bascule asynchrone :*** où le passage d'un état à l'autre (basculement) est commandé uniquement par les entrées.

● ***Bascule synchrone :*** où l'ordre de changement d'état est donné par un signal qu'on appelle signal d'horloge.

Proverbe: ***Les hommes ont la mémoire courte, les circuits c'est pire***

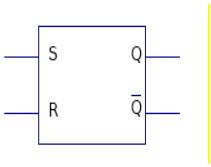
Chapitre 1 : Les bascules

Dans ce chapitre, nous examinerons :

- ☐ les bascules asynchrones, ie la bascule R-S et ses dérivées: RSC,
- ☐ les bascules J-K qui sont des circuits synchrones
- ☐ les bascules D commandées par un niveau logique (Latch).
- ☐ Les bascules D commandées par un front actif d'horloge (edge Triggered)
- ☐ Et en fin les bascules T

I. Bascule RS asynchrone

- La bascule RS asynchrone est une bascule à 2 entrées notées : ***S*** : Set (mise à 1 de Q) et ***R*** : Reset (mise à 0) = Clear.et à 2 sorties Q et $\bar{Q}$. Le symbole de la bascule RS est :



R	S	Q_n	$\bar{Q}_n$	
0	0	Q_{n-1}	$\bar{Q}_{n-1}$	sorties inchangées
0	1	1	0	**Set** : Remise à 1
1	0	0	1	**Reset** : Remise à 0
1	1	x	x	A proscrire

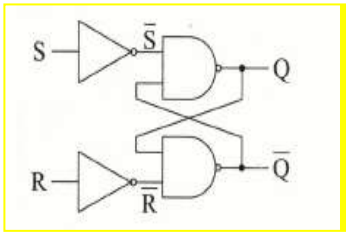
TV d'une bascule RS

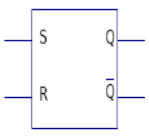
- La table de fonctionnement de la bascule RS (voir ci-dessus).

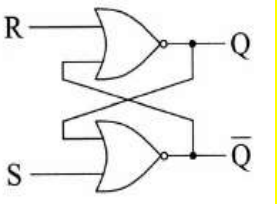
- Son équation de fonctionnement (déduite de sa T.V et T.K) est : $Q_n = S + \bar{R}Q_{n-1}$

- RS réalisées avec des portes NOR ou NAND :

$$Q_n = S + \bar{R}Q_{n-1} = \overline{\overline{S + \bar{R}Q_{n-1}}} = \overline{\bar{S}.\overline{\bar{R}Q_{n-1}}} \qquad Q_n = \bar{R}(S + Q_{n-1}) = \overline{\overline{\bar{R}(S + Q_{n-1})}} = \overline{R + \overline{(S + Q_{n-1})}}$$

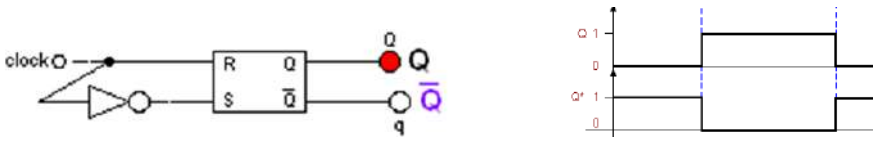






- Application : Horloge à 2 phases

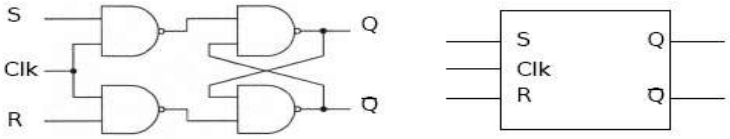
Le signal d'horloge lié à S et inversé sur l'entrée R permet d'engendrer sur les 2 sorties Q et $\bar{Q}$ de la bascule 2 signaux sans recouvrement en opposition de phase :



II. Bascule RS synchrone : RSH = RSC

La Bascule RSC est une bascule RS dans laquelle les entrées R et S ne sont prises en comptes que si elles sont en coïncidence avec un signal de commande C=1. La bascule sera bloquée quand le signal de C= 0.

Si le signal de commande est fourni par une horloge ie C=T=CLK : on parle donc de bascule synchrone. Il s'agit d'une bascule à portes NAND dont les entrées sont commandées par deux autres portes NAND comme le montre la figure ci-dessus :



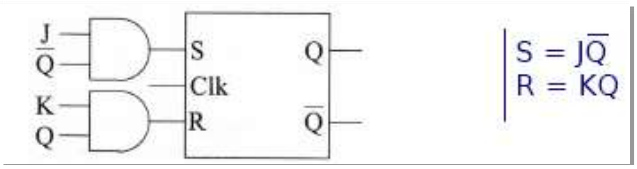
R	S	C	Q_n	Mode de fct
x	x	0	Qn-1	Mémoire
0	0	1	Qn-1	Mémoire
0	1	1	1	Mise à 1
1	0	1	0	Remise à 0
1	1	1	x	indéterminée

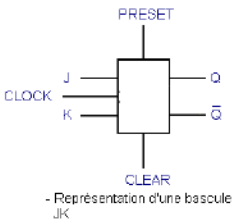
TV d'une bascule RSC=RSH

On constate qu'à chaque fois que C = 0, la bascule est en position mémoire et quand C = 1, la bascule R.S.C. se comporte exactement comme une bascule R-S classique.

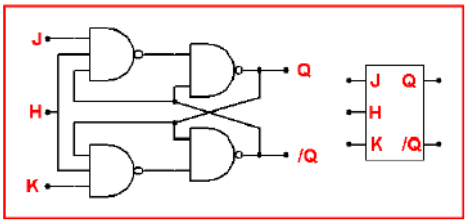
III. Bascule JK

La bascule JK permet de lever l'ambiguïté des bascules RSC = RST. Elle dispose de 2 entrées, respectivement appelées J et K. J sert à la mise à 1 et K à la mise à 0. Contrairement à la bascule RS, la combinaison J=K=1 est non interdite (la sortie passe à l'état opposée).





- Représentation d'une bascule JK

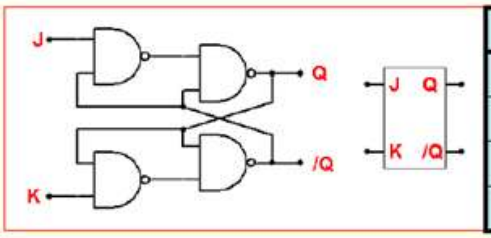


J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

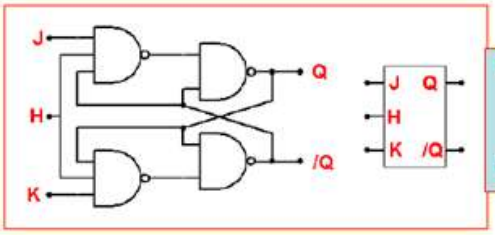
Q_n	Q_{n+1}	J_n	K_n
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

TV d'une bascule JK

Table de transition d'une bascule JK



J	K	Q
0	0	mémoire
0	1	0
1	0	1
1	1	basculement

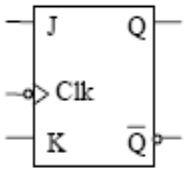


- H=1 → Table de vérité
- H=0 → sorties inchangées

- ***Bascule JK réagissant au front d'horloge***

- La Bascule réagissant sur front d'horloge sont fortement utilisées en électronique, essentiellement pour réaliser des compteurs, des registres à décalage et autres.
- Pour les réaliser, deux technique :
  - ☐ • Utilisation de détecteur de front sur l'entrée Horloge
  - ☐ • Utilisation de la structure maître esclave

H	J	K	Q	Observation
↓	0	0	Qp	Mémoire
↓	0	1	0	Sortie suit J
↓	1	0	1	
↓	1	1	$\overline{Qp}$	Alternance

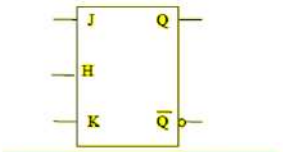


TV d'une bascule JK

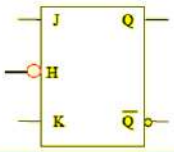
Symbole d'une bascule JK à front déscendant

- Symboles de bascule JK

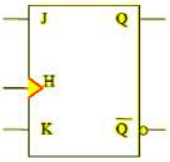
Rmq: Notez la convention de dessin pour l'entrée Horloge



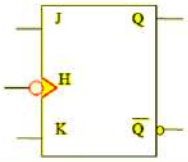
Bascule réagissant sur niveau haut de H (latch)



Bascule réagissant sur niveau bas de H (latch)



Bascule réagissant sur front montant de H

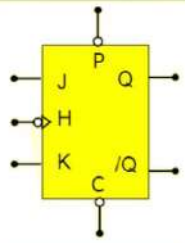


Bascule réagissant sur front descendant de H

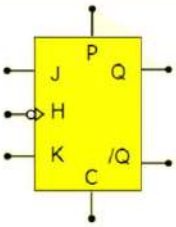
- Rôle des entrées de forcage Pr et Cr

- JK dispose aussi 2 entrées *Preset* (Pr, P, S) et *Clear* (Cr, C, R).
- Pr et Cr : entrées asynchrones (directes) prioritaires.
- J, K, et CLK entrées synchrones.

- L'entrée C force la sortie à 0 ∀ J, K, H
- L'entrée P force la sortie à 1 ∀ J, K, H
- Il ne faut pas activer C et P simultanément



C et P actifs au niveau bas



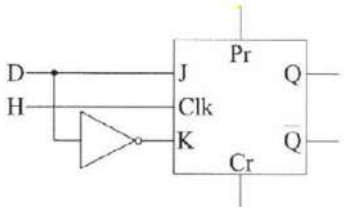
C et P actifs au niveau haut

- T.V des entrées Pr et Cr

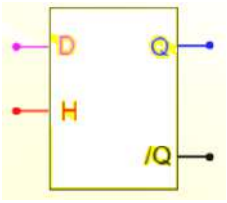
Pr	C	Q	
0	0	Interdit	Forçage simultané à 0 et à 1
0	1	1	Sortie forcée à 1
1	0	0	Sortie forcée à 0
1	1	libre	Bascule fonctionne normalement

IV. Bascule D

La bascule *D ou mémoire* D Latch est dérivée de la bascule JK. Elle dispose d'une seule entrée « D », pour positionner les sorties. Elle aussi, permet de copier l'entrée a la sortie en envoyant une donnée D sur J et son inverse sur l'entrée K.

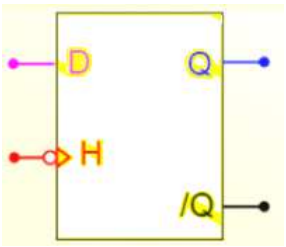


- H = 1 : Q suit D
- H = 0 : Q reste bloqué : état mémoire



D	H	Q_n	Mode de fct
x	0	Q_{n-1}	Mémoire
x	1	D	Recopie

Symbole et TV d'une bascule D Latch



Pendant le front descendant de H, Q prend la valeur de D et la garde jusqu'au prochain front

D	H	Q_n	Mode de fct
x	x	Q_{n-1}	Mémoire
x	↓	D	Recopie

Symbole et TV de bascule D edge Triggered déclenché par FD

D	C	Q_n	Mode de fct
x	x	Q_{n-1}	Mémoire
x	↑	D	Recopie

Symbole et TV de bascule D edge Triggered déclenché par FM

V. Bascule T

La bascule T (trigger) obtenue à partir d'1 bascule JK en injectant le même état dans les entrées J et K. Elle dispose d'une seule entrée ''T'' venant de nom Toggle ie bascule», pour positionner les sorties.

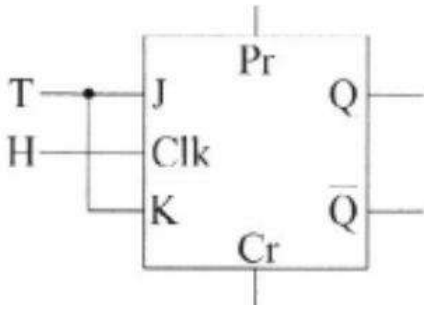
- ***Table de vérité***

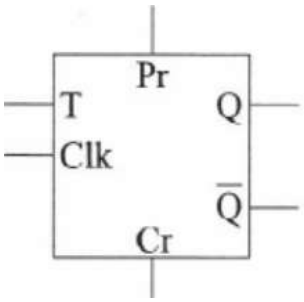
- à partir de la table de vérité de la bascule JK on a :

J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

↔ si J=K=0, $Q_{n+1}=Q_n$; si J=K=1, $Q_{n+1}=\overline{Q_n}$ ↔

T_n	Q_{n+1}
1	$\overline{Q_n}$
0	Q_n





T	CLK	Q_{n+1}	Mode de fct
x	0	Q_n	Mémoire
1	1		Commute
0	1	Q_n	Mémoire

Symbole et TV de la bascule T LATCH déclenchée par niveau (1 ou 0)

Chapitre 2 **: LES COMPTEURS ASYNCHRONES**

Application

Un compteur (ou décompteur) est un circuit électronique constitué essentiellement par un ensemble de bascules et le plus souvent d'un réseau combinatoire. Ce compteur (ou décompteur) permet de comptabiliser le nombre d'événements qui se produisent pendant un temps donné. Chaque événement est traduit en impulsion électrique. Il existe de nombreuses applications des compteurs. Nous pouvons citer:

- ☐ le comptage d'objets (figure 1),

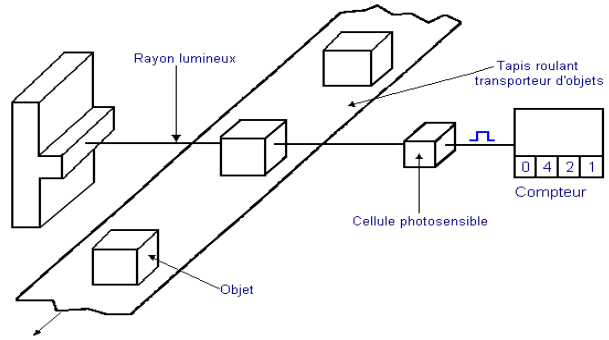


Fig. 1. - A chaque interruption du rayon lumineux par un objet, la cellule photosensible délivre une impulsion qui augmente d'une unité le contenu du compteur.

- ☐ la mesure du temps (figure 2),

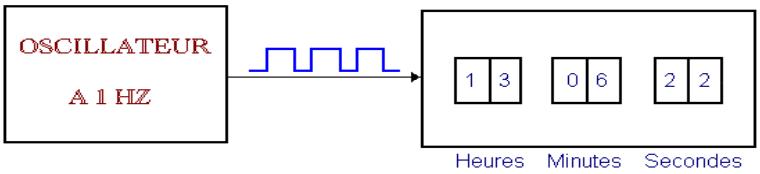


Fig. 2. - L'horloge est un compteur qui s'incrémente d'une unité toutes les secondes.

Le signal de l'horloge de fréquence 1 Hz est divisé par 60 et permet d'obtenir un signal de période 1 minute. Ce deuxième signal est à son tour divisé par 60 afin d'obtenir le signal de période 1 heure. Ensuite, il suffit de compter les heures jusqu'à 24 pour qu'une journée se soit écoulée.

- ☐ la division du temps pour l'obtention de signaux d'horloge permettant la commande des systèmes synchronisés (figure 3).

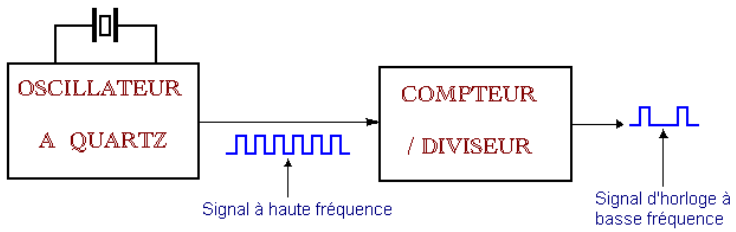


Fig. 3. - Obtention d'un signal d'horloge très stable après division du signal issu d'un oscillateur à quartz.

I. Définitions

Un compteur est un ensemble de n bascules connectées par des portes logiques et décrivant une séquence déterminée au rythme d'une horloge et présentant 2^n combinaisons possibles. Dont les états sont stables et accessibles entre deux impulsions de l'horloge où N : nombre total de combinaisons successives utilisées N ≤ 2n : modulo du compteur.

II. Compteur Asynchrone complet (N=2^n)

Le compteur asynchrone complet est un compteur binaire mod N = 2^n. Où n: nombre de bascules JK (fonctionnant en mode T) ou D. Le signal d'horloge n'est reçu que par le 1er étage (LSB)→Q_0. Le signal d'horloge des autres bascules est fourni par une sortie de l'étage précédent.

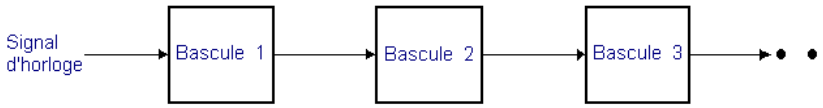


Fig. 4. - Principe de fonctionnement d'un compteur asynchrone.

- Ce compteur décrit le cycle de 0 → N-1
- Les équations d'un tel compteur à n bascules sont :
- On dit que la sortie de la bascule de rang (i-1) sert comme entrée d'horloge de la bascule de rang i

$$J_i = K_i = 1 \quad \forall i \quad 0 \le i \le n-1$$
$$CLK_0 = H,$$
$$CLK_i = Q_{i-1} \quad 1 \le i \le n$$

II.1. Compteur Asynchrone complet (N=2)

Est un diviseur par 2 de fréquence qui peut être obtenu avec bascule D comme représenté figure 5 dont la sortie $\overline{Q}$ est rebouclée sur l'entrée D. Ou avec une bascule JK comme représenté figure 8. Cette bascule fonctionne en mode TOGGLE

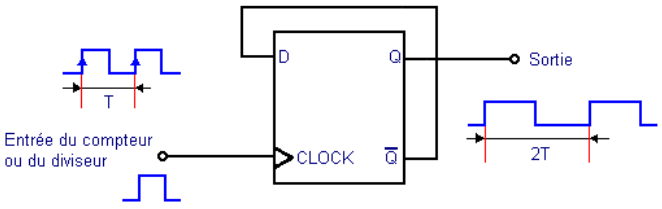


Fig. 5. - Bascule de type D raccordée en diviseur par deux.

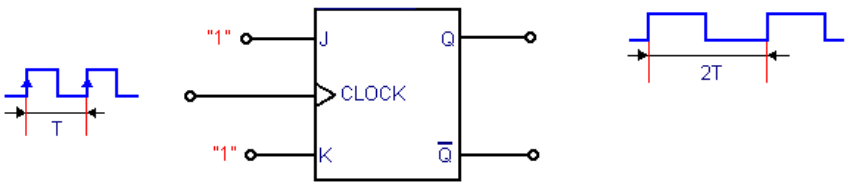


Fig. 8. - Montage d'une bascule JK en diviseur par deux.

II.2. Compteur Asynchrone complet (N=4)

Est un diviseur par 4 de fréquence qui peut être obtenu avec bascule D comme représenté figure 9 où chaque bascule est câblée en diviseur par 2. dont la sortie $\overline{Q}$ est rebouclée sur l'entrée D. Ou avec une bascule JK comme représenté figure 14. Cette bascule fonctionne en mode TOGGLE

Ce compteur décrit le cycle 0,1,2,3 avec n=2 nb bascules. Les éqs d'1 tel compteur à 2 bascules JK : $J_1 = K_1 = 1$ $CLK_1 = H$ $J_2 = K_2 = 1$ $CLK_2 = Q_1$

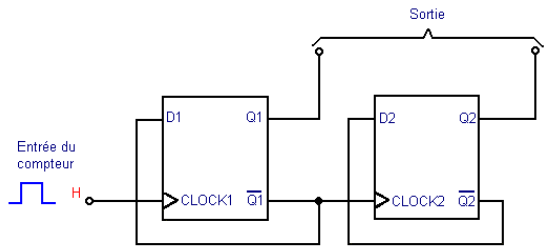


Fig. 9. - Compteur modulo 4.

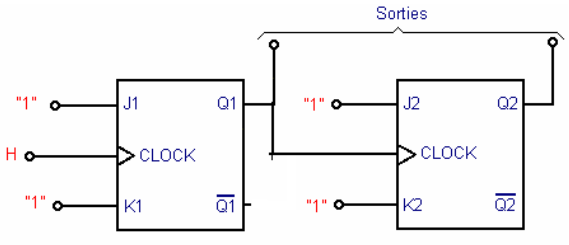


Fig. 14. - Compteur modulo 4 réalisé avec deux bascules JK.

Chronogramme de compteur asynchrone Mod4

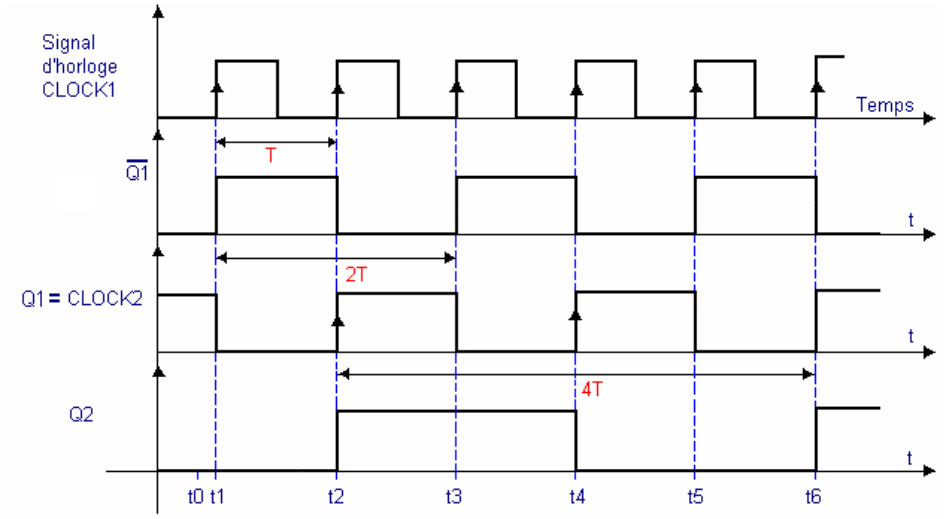
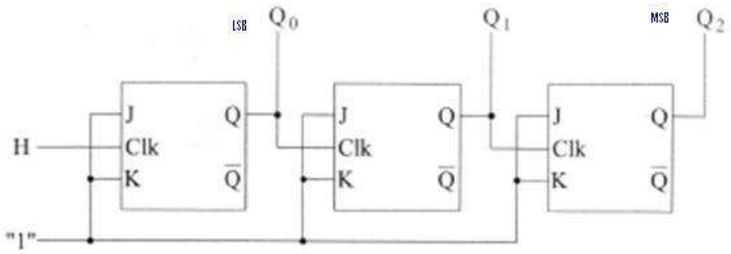


Fig. 10. - Chronogramme relatif au fonctionnement du diviseur par 4.

II.3. Compteur Asynchrone complet (N=8)

Ce compteur décrit le cycle 0,1,2,3,4,5,6,7 avec n=3 nb bascules. Les éqs d'1 tel compteur à 3 bascules JK d sorties resp. $Q_2Q_1Q_0$ que Nous supposant actives à front montant: $J_i = K_i = 1$ pour $0<i<2$; $CLK_0 = H$ $CLK_1 = Q_0$ et $CLK_2 = Q_1$

CLK_i : entrée d'horloge de la bascule de rang i. On suppose qu'initialement toutes les bascules sont à 0



- les sorties Q_0, Q_1, Q_2 fournissent des horloges de fréquence f/2, f/4 et f/8 (diviseurs de fréquence).

- Q_0, change d'état à chaque front montant du H
- Q_1, change d'état à chaque front montant du Q_0,
- Q_2 change d'état à chaque front montant t du Q_1

Chronogramme de compteur asynchrone Mod8

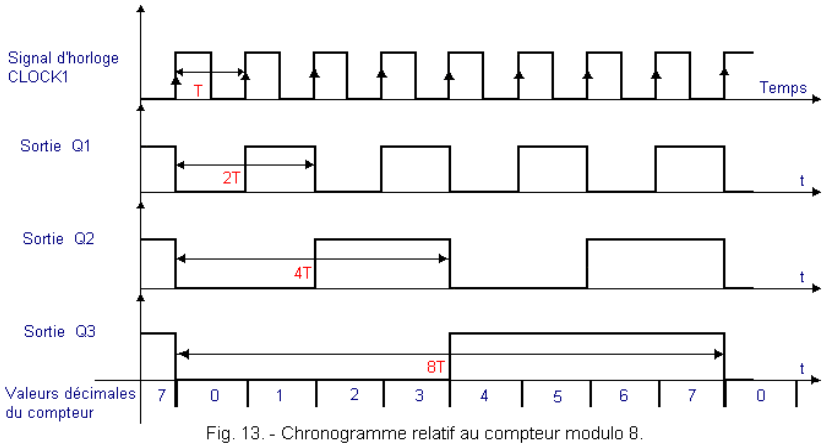


Fig. 13. - Chronogramme relatif au compteur modulo 8.

II.4. Compteur Asynchrone complet (N=16 = 2^4) : voir TD

III. Décompteur Asynchrone complet (N=2^n)

Ce décompteur décrit le cycle de N-1→0 où N=2^n. Les équations d'un tel décompteur à n bascules JK:

$$J_i = K_i = 1 \quad \forall i \quad 0 \le i \le n-1$$
$$CLK_0 = H,$$
$$CLK_i = \overline{Q}_{i-1} \quad 1 \le i \le n$$

Il existe deux façons d'obtenir un décompteur asynchrone. la première consiste à connecter l'horloge de chaque bascule à la sortie inversée de la bascule précédente. (Fig. 3.10). la figure Fig. 3.11 montre l'évolution des états du système.

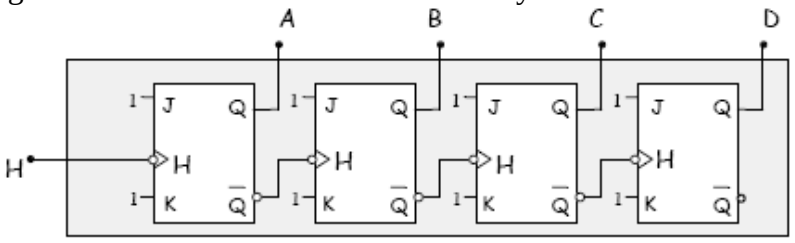


Fig. 3.10 : Décompteur Asynchrone 4 bits

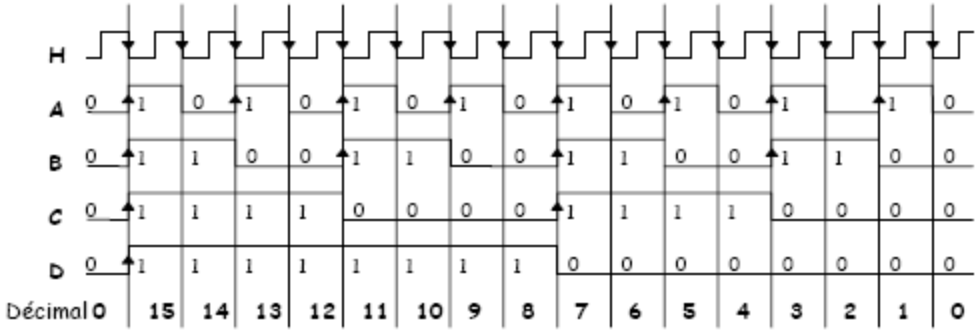


Fig. 3.11 : Chronogramme d'un décompteur asynchrone 4 bits (modulo 16)

Une deuxième méthode consiste à prendre les sorties du compteur sur les sorties inversées des bascules (Fig. 3.12 et Fig. 3.13).

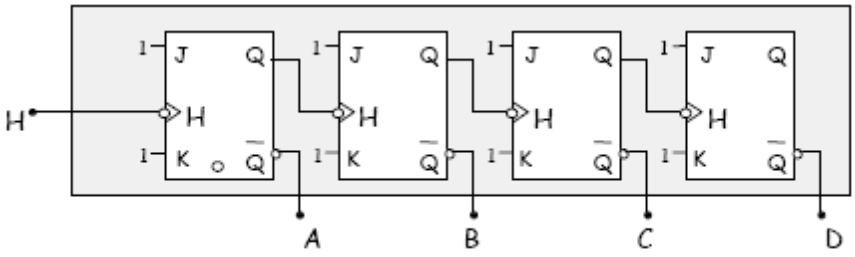


Fig. 3.12 : Décompteur asynchrone 4 bits

	D	C	B	A	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	
0	0	0	0	0	1	1	1	1	15
1	0	0	0	1	1	1	1	0	14
2	0	0	1	0	1	1	0	1	13
3	0	0	1	1	1	1	0	0	12
4	0	1	0	0	1	0	1	1	11
5	0	1	0	1	1	0	1	0	10
6	0	1	1	0	1	0	0	1	9
7	0	1	1	1	1	0	0	0	8
8	1	0	0	0	0	1	1	1	7
9	1	0	0	1	0	1	1	0	6
10	1	0	1	0	0	1	0	1	5
11	1	0	1	1	0	1	0	0	4
12	1	1	0	0	0	0	1	1	3
13	1	1	0	1	0	0	1	0	2
14	1	1	1	0	0	0	0	1	1
15	1	1	1	1	0	0	0	0	0

Fig. 3.13 : Séquence de comptage et de décomptage.

IV. Compteur/Décompteur Asynchrone

Un exemple est illustré sur la figure Fig. 3.14. L'horloge de chaque bascule est prélevée soit sur la sortie Q soit sur la sortie $\overline{Q}$ de la bascule précédente selon si l'on désire fonctionner en compteur ou en décompteur. Ceci est réalisé grâce à 3 "petits'" multiplexeurs 1 parmi 2. L'entrée de contrôle U/d permet de choisir le sens de comptage. U/D=1 $\Rightarrow$ comptage ascendant (compteur). U/D =0 $\Rightarrow$ comptage descendant (décompteur)

$$\begin{cases} J_i = K_i = 1 \quad \forall i \\ CLK_i = H, \\ CLK_{i+1} = \overline{X}.\overline{Q_i} + X.Q_i \end{cases}$$

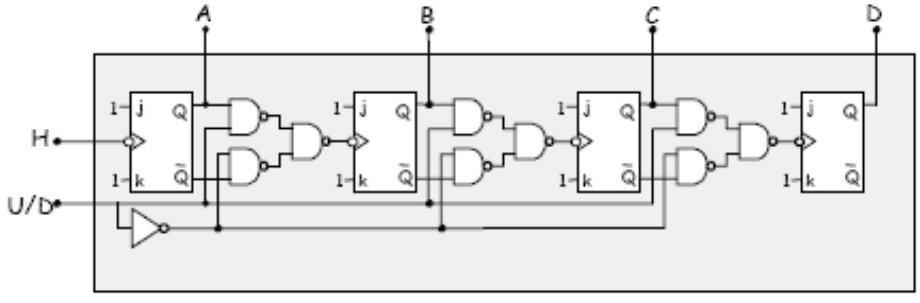


Fig. 3.14 : Compteur / décompteur 4 bits

V. Comptage Asynchrone incomplet ($N<2^n$)

Jusqu'ici nous avons étudié des compteurs qui parcourent toutes les valeurs possibles de 0 à 2^{n-1} (nombre de bascules). Il arrive qu'on ait besoin de compteurs à cycle incomplet, c.a.d. des compteurs modulo N avec $N\neq2^n$, qui comptent de 0 jusqu'à N-1 et recommence à 0.
Pour le cas des compteurs asynchrones, pour construire un compteur [N], (modulo N) avec N≠2n, on détecte l'état N, et on s'en sert pour remettre le compteur à 0 d'une façon asynchrone en agissant sur les entrées Clear des bascules JK.
Pour cela on rajoute une porte NAND supplémentaire dont:

- les entrées sont les sorties qui sont à 1 pour N
- et dont la sortie va à leurs entrées Clear.

Ce compteur parcourt une partie de son cycle complet (soient N états parmi 2^n). IL est conçu de la même manière qu'un compteur asynchrone complet :

- Le compteur est remis à 0 au moment où l'état N essaye d'apparaître, donc celui ci est remplacé par 0.
- la figure Fig.3.15 montre un compteur modulo 5 et un compteur modulo 6.

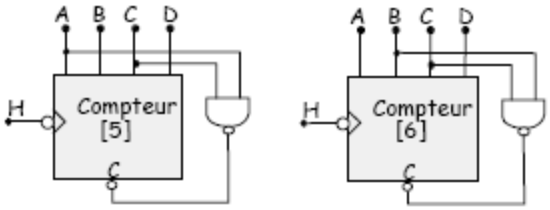


Fig. 3.15 : Compteurs modulo 5 et 6

Pour ce qui concerne les compteurs synchrones, le problème ne se pose pas, car la séquence de comptage est prise en considération lors de la synthèse des compteurs.

VI. Mise en cascade des compteurs Asynchrone

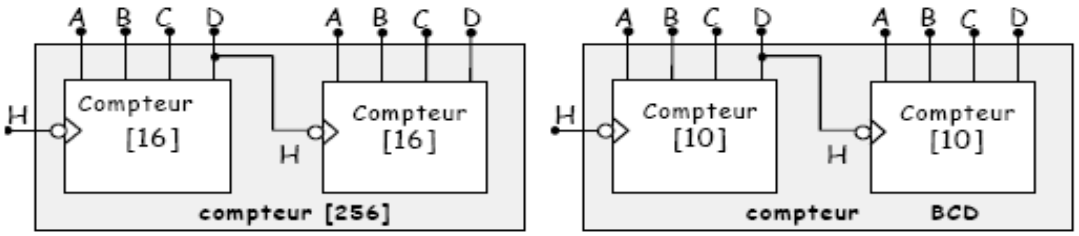


Fig. 3.16 : Compteur modulo 256 et compteur BCD modulo 100

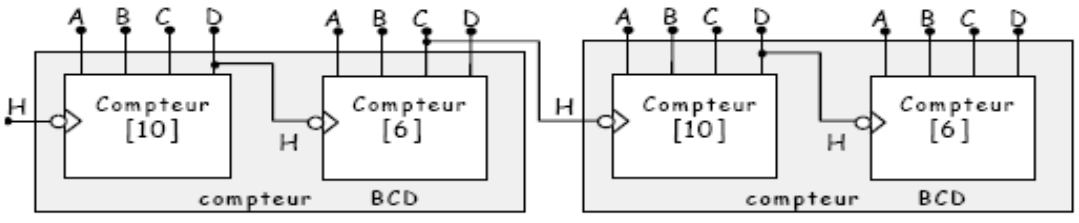


Fig. 3.17 : Compteur des secondes (à gauche) suivi du compteur des minutes

Un boîtier compteur contient généralement 4 étages. Pour constituer un compteur de plus grande taille, il faut associer plusieurs boîtiers en cascade.

Pour le cascadage des compteurs asynchrones, il suffit de relier la sortie MSB (significative) de chaque boîtier à l'entrée horloge du compteur de rang supérieur.

La figure Fig. 3.16 montre deux exemples de compteurs. Un compteur 8 bits construit avec deux compteurs 4 bits et un compteur BCD modulo 100 construit avec deux décades.

Les règles d'assemblage des bascules s'appliquent aux compteurs ; en connectant par le mode asynchrone deux compteurs modulo 10, on obtient un compteur modulo 100 : chaque fois que le premier compteur passe de 9 à 0, sa dernière bascule Q_D passe de 1 à 0 et envoie un front actif sur l'entrée d'horloge du deuxième compteur qui s'incrémente d'une dizaine.

Il faut respecter les ***règles d'assemblage*** : le 2ème compteur ne doit s'incrémenter qu'au front descendant de Q_D ; donc si ce 2ème compteur réagit aux front descendant, il devra être commandé directement par Q_D ; mais s'il fonctionne sur front montants, il devra être commandé par $\overline{Q_D}$ qui passe de 0 à 1 quand Q_D passe de 1 à 0 lors de retour du compteur de l'état 9 à l'état 0.

VII. Inconvénient et avantages des compteurs Asynchrone

- **1er inconvénient : la mauvaise fiabilité des RAZ (R : Clear) et Preset (S)**

Quand on utilise des compteurs intégrés, ce problème n'excste plus pour l'utilisateur car le fabricant l'a résolu avec une dispersion négligeable des paramètres.

- **2ème inconvénient la limitation de la vitesse de fonctionnement :**

Les compteurs asynchrones sont simples à concevoir mais ils ne peuvent pas être utilisés à n'importe quelle fréquence. En effet, à cause du retard de propagation τ introduit par chaque bascule, la sortie Q_0 n'est délivrée qu'après un temps τ, la sortie Q_1 n'est délivrée qu'après 2τ et ainsi de suite. Autrement dit, les retards de propagation introduits par les bascules s'additionnent de sorte que la sortie Q_{n-1} n'est obtenue qu'après un retard de $n\tau$ (n : nombre de bascules à utiliser).

Un bon fonctionnement impose que :

$$n.T_p \langle T_e \Leftrightarrow F_e \langle \frac{1}{n.T_p} = F_{\max}$$

Te : période du signal d'horloge H.
Fe : fréquence du signal d'horloge H.

Evidemment, à mesure qu'augmente n, la fréquence limite de fonctionnement (F_{max}) diminue. A titre d'exemple, pour un compteur modulo 100 utilisant 10 bascules ayant τ = 50 ns, sa fréquence maximale de fonctionnement sera :

$$F_{max} = \frac{1}{n\tau} = \frac{1}{10 \times 50\text{ns}} = 2\text{MHz}$$

Nous venons de voir que les compteurs asynchrones présentent un inconvénient majeur (limitation de fréquence de fonctionnement) dû au principe de leur fonctionnement asynchrone. On contourne cette limitation en utilisant des compteurs synchrones dont le signal d'horloge est le même pour toutes les bascules.

- **3ème inconvénient les états transitoires**

Le fait que toutes les bascules ne changent pas d'état simultanément, il apparaît des états transitoires fugitifs chaque fois qu'on passe d'un état à un autre. Si on note Tp le temps de propagation de chaque bascule, examinons en détail ce qui se produit quand on passe de l'état 7 à l'état 8. (Fig. 3.9).

Quand A → 0, B → 0 après un retard Tp, il en résulte un état 0110=6 qui va exister pendant Tp. même façon, quand B → 0, C → 0 après un retard Tp, il en résulte l'état transitoire 0100=4.

Quand C → 0, D → 1 mais après Tp, il en résulte l'état transitoire 0000=0. On remarque : pendant le changement d'état 7 → 8, le système en réalité passe par la séquence suivante :

$$7 \rightarrow 6 \rightarrow 4 \rightarrow 0 \rightarrow 8 .$$

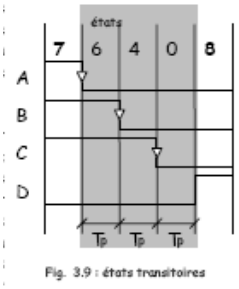


Fig. 3.9 : états transitoires

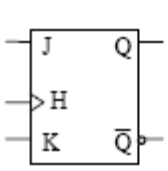
Avantages : conception très simple et rapide

Chapitre 3 : LES COMPTEURS SYNCHRONES

Intoduction

Les compteurs synchrones sont aussi réalisés à l'aide de bascule J-K. Mais à la différence des compteurs asynchrones, toutes les bascules reçoivent en parallèle le même signal d'horloge. Il en résulte qu'a chaque coup d'horloge toutes les sorties changent en même temps, il n y a donc pas d'états transitoires.

Pour faire décrire au compteur une séquence déterminée, il faut définir les entrées J et K de chaque bascule à chaque top d'horloge, en utilisant la table de transition des bascules JK (Fig. 3.19). Pour différentier "un peut" des compteurs asynchrones, on va prendre des bascules réagissant sur front montant.



H	Q_n	Q_{n-1}	J	K	J	K
↓	0	0	0	0	0	x
↓	0	1	1	1	1	x
↓	1	0	1	0	x	1
↓	1	1	0	1	x	0

Fig. 3.19 : Table des transitions d'une bascule J-K

I. Compteur synchrone complet

Nous allons raisonner de la même manière sur un compteur synchrone complet modulo 8 et les équations obtenues pour les entrées J_i et K_i seront généralisées par la suite à un compteur modulo $N = 2^n$ avec n quelconque.

Le compteur modulo 8 décrit le cycle 0,1,2,3,4,5,6,7. Il nécessite trois bascules JK ($8 = 2^3$) dont les entrées d'horloge sont connectées au même signal d'horloge H. Nous supposons que les bascules sont à front montant.

D'après les chronogrammes d'un compteur modulo 8 (figure 3.20), on peut faire les observations suivantes :

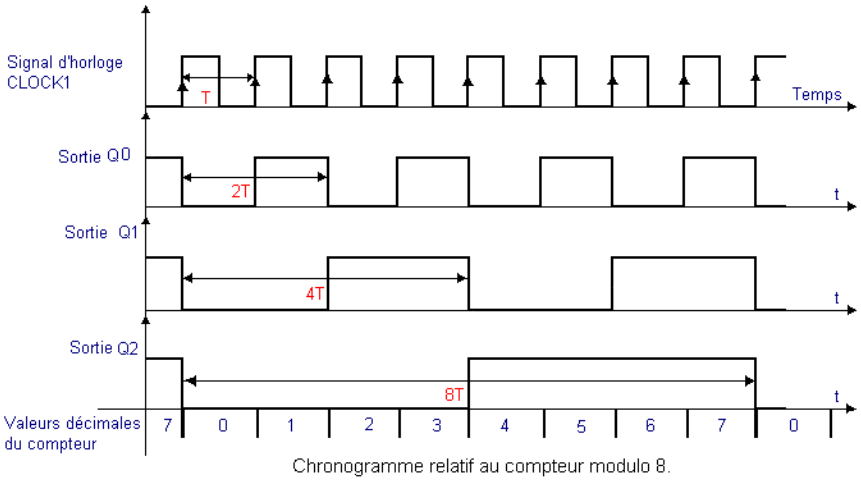


Fig. 3.20 : *Chronogramme relatif au compteur synchrone modulo 8*

- Q_0 change d'état à chaque front montant ***(FM)*** du signal d'horloge H.
- Q_1 change d'état au prochain FM du signal d'horloge H si Q_0 est à 1. Sinon, elle maintient son état.
- Q_2 change d'état au prochain FM du signal d'horloge H si Q_0 et Q_1 sont à 1. Sinon, elle maintient son état.

D'après le fonctionnement d'une bascule JK, ceci conduit aux équations suivantes pour un compteur synchrone modulo 8 :

- $CLK_i = H \quad \forall i$
- $J_0 = K_0 = 1$
- $J_1 = K_1 = Q_0$
- $J_2 = K_2 = Q_0 Q_1$

Ces équations peuvent être généralisées pour un compteur synchrone modulo N=2^n sous une forme récurrente :

- $CLK_i = H \quad \forall i$
- $J_0 = K_0 = 1$
- $J_i = K_i = Q_0 Q_1 ..... Q_{i-1} = J_{i-1}.Q_{i-1}$

pour $1 \prec i \prec n$

La figure 3. 21 montre le schéma d'un compteur synchrone modulo 8.

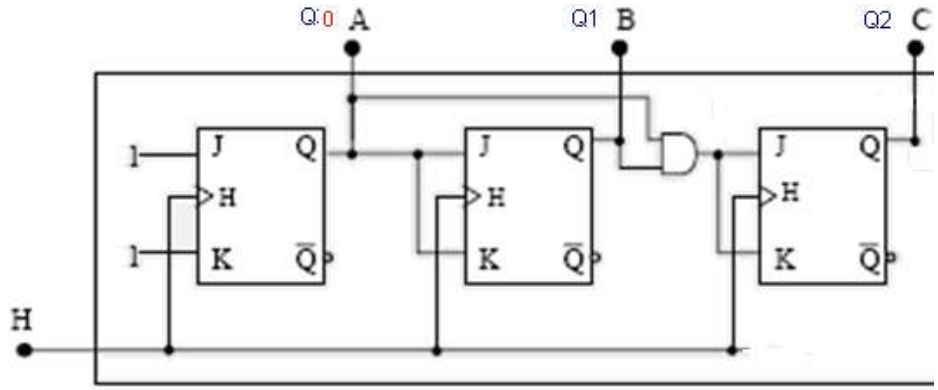


Fig. 3.21 : Compteur synchrone 3 bits

II. Décompteur synchrone complet

Un décompteur synchrone fonctionne de la même manière qu'un compteur synchrone sauf qu'à chaque front actif du signal d'horloge, il se décrémente de 1.
De la même manière, nous allons raisonner sur un décompteur synchrone modulo 8 et les résultats obtenus seront étendus à un décompteur modulo N =2^n avec n quelconque.
Le tableau ci dessous illustre le cycle complet allant de 7 à 0. Soient Q_2,Q_1,Q_0 les sorties des 3 bascules JK que nous supposons à front montant.

#top	Q_2	Q_1	Q_0	$J_2=K_2$	$J_1=K_1$	$J_0=K_0$
0	1	1	1	0	0	1
1	1	1	0	0	1	1
2	1	0	1	0	0	1
3	1	0	0	1	1	1
4	0	1	1	0	0	1
5	0	1	0	0	1	1
6	0	0	1	0	0	1
7	0	0	0	1	1	1
8	1	1	1			

Ceci se traduit par les chronogrammes et diagramme des états suivants :

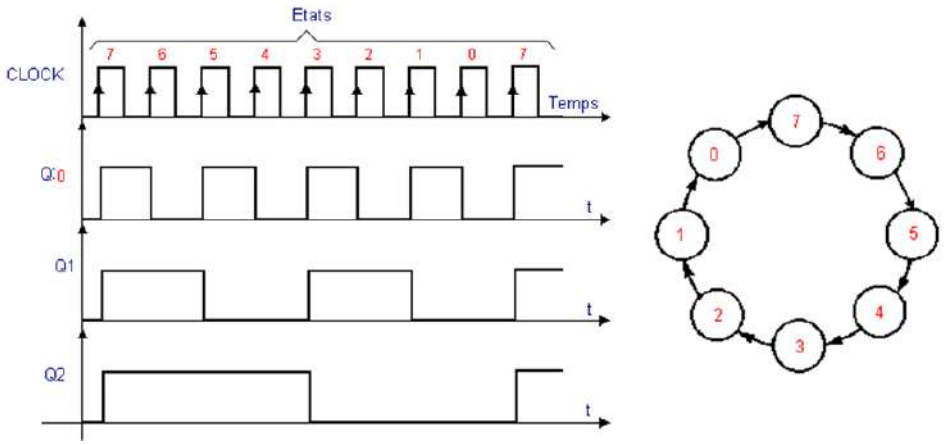


Fig.3.22. Chronogramme et le diagramme des états d'un compteur modulo 8.

On constate que :
- Q_0 change d'état à chaque front montant du signal d'horloge H.
- Q_1 change d'état au prochain front montant du signal d'horloge H si Q_0 est à 0. Sinon, elle maintient son état.
- Q_2 change d'état au prochain front montant du signal d'horloge H si Q_0 et Q_1 sont à 0. Sinon, elle maintient son état.

D'après le fonctionnement d'une bascule JK, ceci conduit à :

$-\ CLK_i = H \quad \forall i$
$-\ J_0 = K_0 = 1$
$-\ J_1 = K_1 = \overline{Q}_0$
$-\ J_2 = K_2 = \overline{Q}_0\overline{Q}_1$

$-\ CLK_i = H \quad \forall i$
$-\ J_0 = K_0 = 1$
$-\ J_i = K_i = \overline{Q}_0\overline{Q}_1.....\overline{Q}_{i-1}$
$-\ J_i = K_i = J_{i-1}.\overline{Q}_{i-1}$

Ces équations peuvent être généralisées sous une forme récurrente :

La figure 3. 23 montre le schéma d'un compteur synchrone modulo 8.

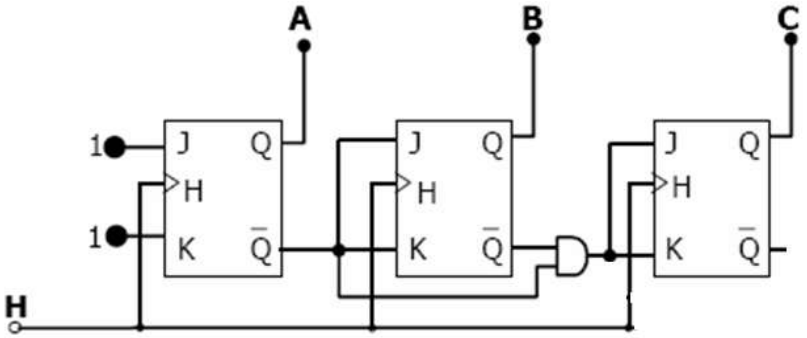


Figure3.23 Décompteur Synchrone 3 bits

III. Compteur bidirectionnel synchrone complet

- Il dispose d'une seule entrée d'horloge et d'une ligne de commande X tel que:
  - X=1 → le comptage
  - X=0 → le décomptage
- Exemple : compteur synchrone bidirectionnel modulo 8
- N. avons vu dans les paragraphe I et II que :

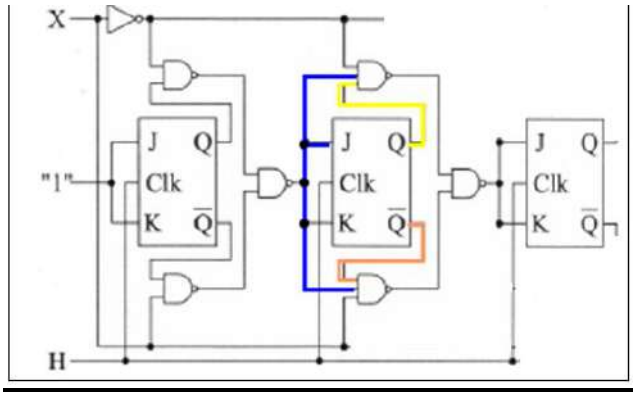
$-CLK_i = H \quad \forall i$
$-J_0 = K_0 = 1$
$-J_1 = K_1 = Q_0$
$-J_2 = K_2 = Q_0 Q_1$

$-CLK_i = H \quad \forall i$
$-J_0 = K_0 = 1$
$-J_1 = K_1 = \overline{Q_0}$
$-J_2 = K_2 = \overline{Q_0}\,\overline{Q_1}$

- Ce qui se généralise facilement au cas du compteur bidirectionnel mod 2^n par:

$-CLK_i = H \quad \forall i$
$-J_0 = K_0 = 1$
$-J_1 = K_1 = Q_0.X + \overline{Q_0}.\overline{X}$
$-J_2 = K_2 = Q_0 Q_1.X + \overline{Q_0}\,\overline{Q_1}.\overline{X}$

- $-CLK_i = H \quad \forall i$
  $-J_i = K_i = Q_0 Q_1 ... Q_{i-1}.X + \overline{Q_0}\,\overline{Q_1} ... \overline{Q_{i-1}}.\overline{X}$



IV. Compteur synchrone incomplet

La méthode de synthèse d'un compteur synchrone incomplet consiste à :

1. Dresser un tableau de transition des états présents et futurs des n bascules JK constituant le compteur (n est l'exposant de la puissance de 2 immédiatement supérieure à N). On remplie ce tableau par les valeurs à appliquer sur les entrées J_i et K_i pour obtenir une transition déterminée sur la sortie de chaque bascule en se servant du tableau de transition d'une bascule JK (figure n°).

Q_N	Q_{N+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q_N : état présent de la bascule
Q_{N+1} : état futur de la bascule

Figure: tableau de transition d'une bascule JK

2. En déduire les équations simplifiées des entrées J_i et K_i grâce aux tableaux de Karnaugh en considérant les états interdits comme indifférents.
3. Examiner l'évolution des états interdits connaissant les équations des entrées J_i et K_i. Si il existe un état interdit qui n'évolue pas vers un état du cycle normal après un certain nombre d'impulsions d'horloge, il doit être détecté en provoquant une remise à 0 asynchrone (Clear) de toutes les bascules constituant le compteur. Si tous les états interdits évoluent vers le cycle normal, on dit que le compteur est **auto-correcteur.**

IV.1. Exemple : compteur synchrone modulo 6

A titre d'illustration, nous allons concevoir un compteur synchrone modulo 6. Ce compteur nécessite 3 bascules JK Q_2, Q_1, Q_0 dont les entrées d'horloge sont connectées au même signal H. Il dispose d'un cycle normal de 6 états allant de 0 à 5.

1. Nous allons dresser le tableau de transition des états présents et futurs des 3 bascules Q_2, Q_1, Q_0 cons tituant le compteur modulo 6.

Etat présent **Etat futur**

Dec.	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	Q_2	Q_1	Q_0	**Dec.**
0	0	0	0	…	…	…	…	…	…	0	0	1	1
1	0	0	1	…	…	…	…	…	…	0	1	0	2
2	0	1	0	…	…	…	…	…	…	0	1	1	3
3	0	1	1	…	…	…	…	…	…	1	0	0	4
4	1	0	0	…	…	…	…	…	…	1	0	1	5
5	1	0	1	…	…	…	…	…	…	0	0	0	0

2) Mise en équation des entrées J_i et K_i

J_0

Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
00	…	…	…	…
01	…	…	…	…

$J_0 = ...$

K_0

Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
00	…	…	…	…
01	…	…	…	…

$K_0 = ...$

J_1

Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
0	…	…	…	…
1	…	…	…	…

$J_1 = ...$

K_1

Q_2 \ Q_1Q_0	**00**	**01**	**11**	**10**
0	…	…	…	…
1	…	…	…	…

$K_1 = ...$

J_2=

Q_1Q_0 / Q_2	**00**	**01**	**11**	**10**
0	…	…	…	…
1	…	…	…	…

$J_2 = ...$

K_2

Q_1Q_0 / Q_2	**00**	**01**	**11**	**10**
0	…	…	…	…
1	…	…	…	…

$K_2 = ...$

On en déduit les équations suivantes pour les entrées ***J_i et K_i***

$J_0 = ...$ $K_0 = ...$

$J_1 = ...$ $K_1 = ...$

$J_2 = ...$ $K_2 = ...$

3) Vérification de l'évolution des 2 états interdits

Connaissant les équations des entrées J_i, et K_i, on va déterminer l'évolution des 2 états interdits 6 et 7.

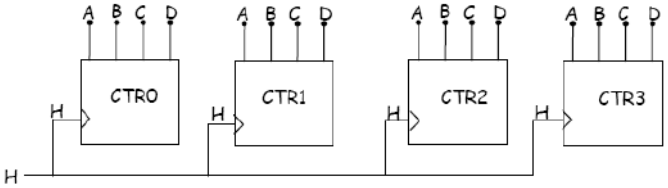
Etat présent **Etat futur**

	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	Q_2	Q_1	Q_0	
6	1	1	0	…	…	…	…	…	…	1	1	1	**7**
7	1	1	1	…	…	…	…	…	…	0	0	0	**0**

Les deux états interdits ne sont pas des états de blocage. Donc le compteur est auto - correcteur.

III.5. Mise en cascade de compteur synchrones

La mise en cascade doit être SYNCHRONES, tous les compteurs doivent recevoir la même horloge. Le problème est que de cette façon ils vont compter en parallèle et on n'aura pas le comptage désiré.



Il faut qu'un compteur ne s'incrémente que lors du débordement du compteur précédent. On va rajouter à chaque compteur une entrée de validation V et une sortie de retenue R

L'entrée de validation V permettra de le contrôler : V=1 → Comptage, V=0 → arrêt

La sortie de retenue R passe à 1 pour indiquer que le compteur est arrivé en fin de cycle.

- Compteur 4 bits, N=15→R=1, N≠15 → R=0
- Compteur par 10, N=9 → R=1, N≠9 → R=0

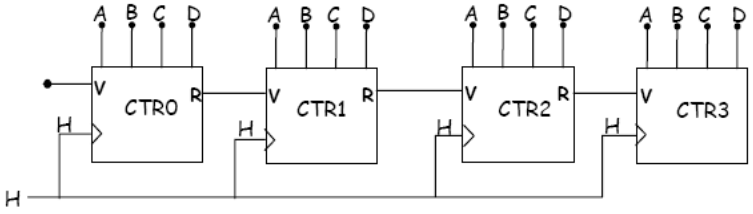


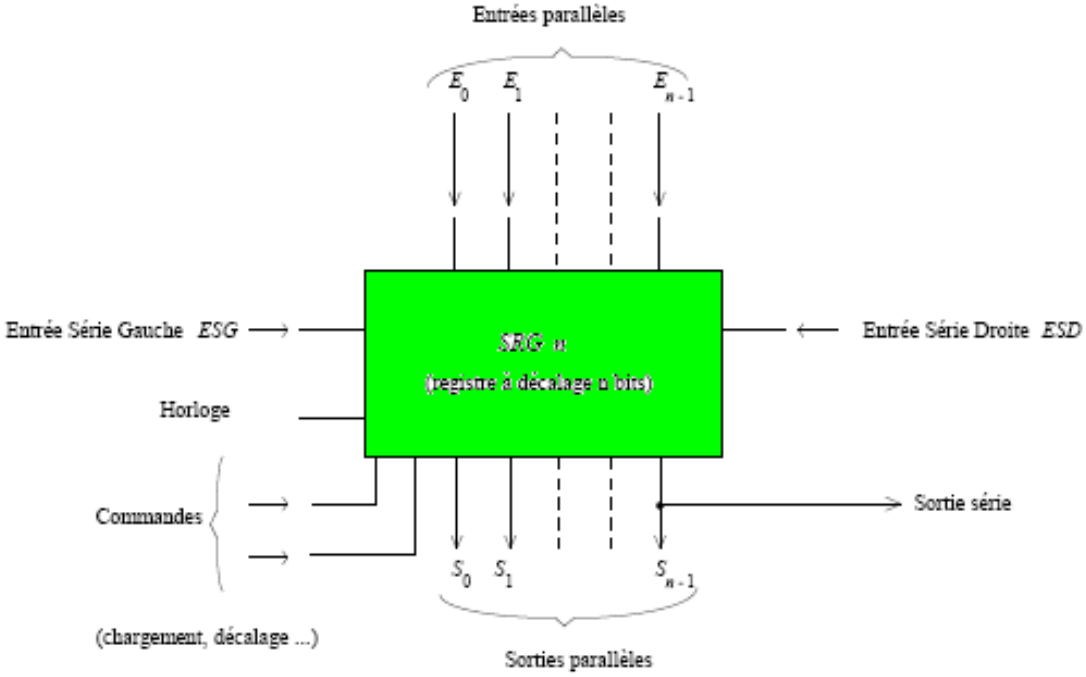
Figure 3.5 : Cascadage de compteurs syncrones

Chapitre 4 : LES REGISTRES

I. Présentation

Un registre est un ensemble ordonné de bascules synchrones (en générale D). Ils sont très utilisés comme :

- ☐ mémoire temporaire (registre de mémorisation)
- ☐ circuit d'opération de multiplication ou de division par 2
- ☐ circuit de mémoire et de traitement de l'information
- ☐ Généralement un registre peut se présenter par le schéma suivant :

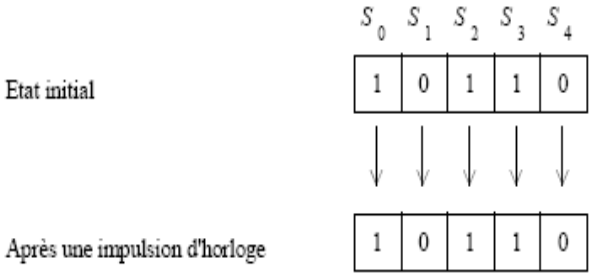


II. Fonctionnement

Les registres disposant de toutes ces entrées, sorties et commandes sont appelés ***registres universels à n cellules***. Tous les registres actuellement commercialisé n'ont pas toutes les possibilités de commande du registre universel essentiellement à cause de la limitation du nombre de broches disponibles par boîtier. Les sorties S0…Sn-1 sont les sorties des bascules constituant les cellules du registre.

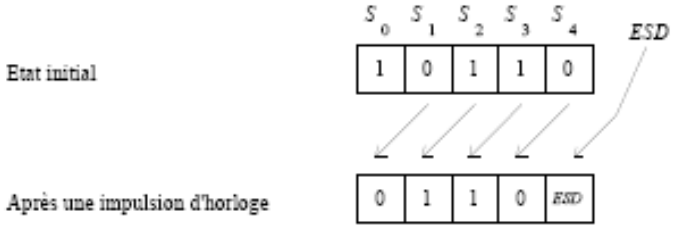
Les signaux de commande du registre permettent de :

- garder une information en mémoire. Chaque bascule conserve sa valeur malgré les impulsions d'horloge.



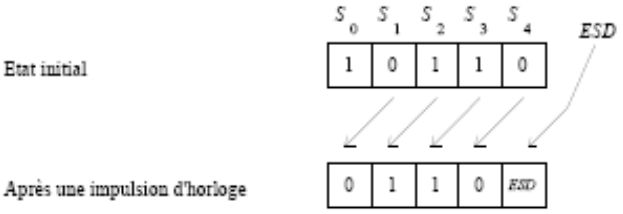
- de décaler une information de la gauche vers la droite. Le contenu de la bascule de rang i est transmis à la bascule de rang i+1 à chaque impulsion de CLK.

- ***exemple:***



S_0 prend donc la valeur de l'entrée du registre ESG (Entrée Série Gauche) tandis que la donnée présente en S4 est perdue.

- de décaler 1 information de la droite vers la gauche. Le contenu de la bascule de rang i+1 est transmis à la bascule de rang i à chaque impulsion de CLK.
- ***exemple:***



S_4 prend donc la valeur de l'entrée du registre ESD (Entrée Série Droite) tandis que la donnée présente en S_0 est perdue.

III. Constitution d'un registre

Les registres peuvent se présenter sous différentes formes selon l'accès aux entrées et aux sorties. On distingue plusieurs types :

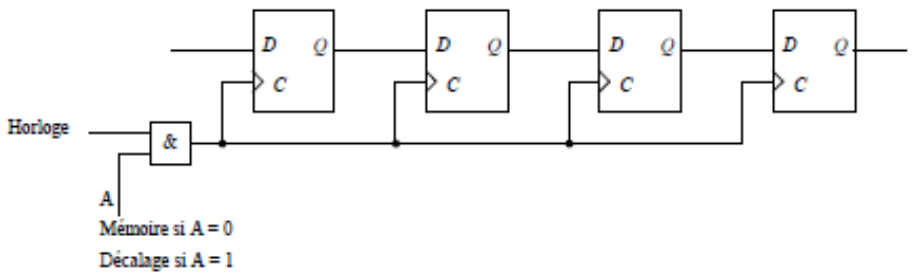
• Registre de mémorisation
• Registre à Entrée série sortie série ou registre à décalage
• Registre à Entrée série sortie parallèle
• Registre à Entrée parallèle sortie série
• Registre à entrée parallèle sortie parallèle ...

III.1. Registre de mémorisation

Suivant la nature de la bascule, la fonction mémoire peut se réaliser de différentes manières. La table suivante donne la combinaison des entrées de bascules qui réalisent la fonction mémoire.

Type	Entrées
RS	$R = S = 0$
JK	$J = K = 0$ ou $J = \overline{K} = Q$
D	$D = Q$

Une autre solution consiste à interdire l'action de CLK en intercalant une porte AND en série



- On distingue :
  - Les Registres Latché
  - Les registres à réaction sur fronts

III.1.1. Registre de mémorisation à réaction sur fronts

Un exemple de ces registres est représenté sur la figure ci-dessous. Au coup d'horloge l'information présente en $A_3A_2A_1A_0$ passe en $Q_3Q_2Q_1Q_0$ et y restera jusqu'au coup d'horloge suivant. Les changements du mot d'entrée ne sont répercutés sur la sortie qu'aux coups d'horloge.

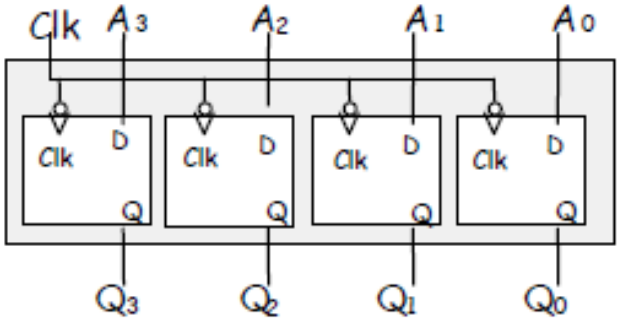


Fig. : Registre à réaction sur front

III.1.2. Registre Latché

Un exemple de ces registres est représenté sur la figure en face. Tant que l'entrée de validation G="1", la sortie $Q_3Q_2Q_1Q_0$ recopie l'entrée $A_3A_2A_1A_0$. Quand G passe à "0", l'état de la sortie restera inchangé (mémorisé, latché) jusqu'au moment où G repasse à "1".

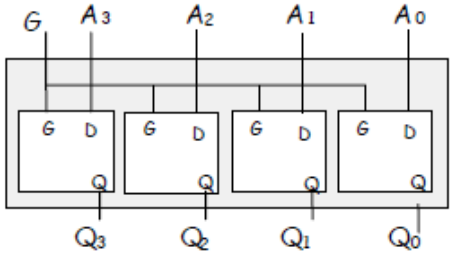
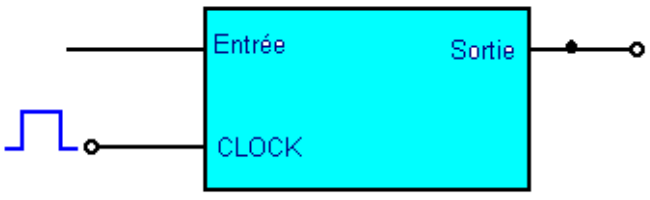


Fig. : registre Latche à 4 bits

III.2. Registres à décalage (registres entrée série- sortie série)

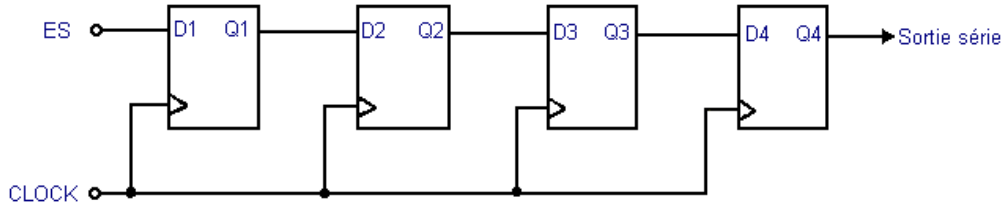
Se sont des bascules interconnectées de façon à ce que l'état de la bascule de rang **i** soit transmis à la bascule de rang **i+1 (ou i-1)** quand un signal d'horloge est appliqué à l'ensemble des bascules. Les états des bascules sont décalés d'une position à chaque front actif de l'horloge d'où le nom de «registres à décalage».
Une application importante des registres à décalage est la transmission série de données logiques. Certains registres sont capables de décaler à droite ou à gauche (registres à décalage universels) généralement réalisés avec des bascules du type maître esclave D ou RS.



REGISTRE SERIE - SERIE

III.2.1. Registres à décalage à droite

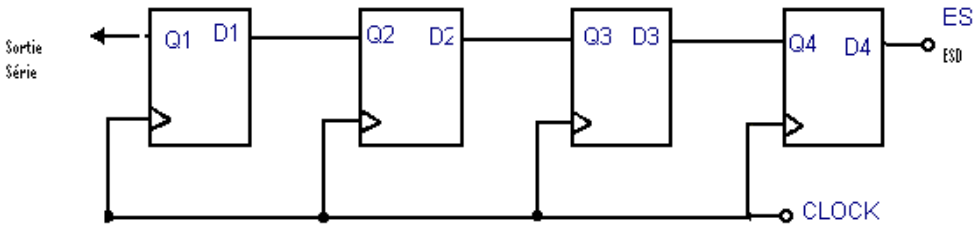
Pour comprendre le fonctionnement des registres à décalage à droite, nous allons prendre à titre d'illustration le circuit très simple ci-dessous :



- A chaque impulsion d'horloge, le contenu du registre est décalé d'un pas vers la droite. La bascule de rang i+1 prend la valeur de la sortie de la bascule du rang i à chaque FM de CLK. Son entrée D doit être connecté avec la sortie Q_i.
- Au 1er Front Montant : la donnée présente en ESG est mémorisée en Q_1, tandis que la donnée présente en Q_4 est perdue.
- nous pouvons, dans un premier temps, charger le registre avec une information, puis dans un second temps, décaler celle-ci d'un ou plusieurs pas vers la droite.
- Au bout de 4 Front Montant de CLK, l'état de la 1ère bascule sera disponible sur la sortie de la dernière bascule (D_4)

III.2.2. Registres à décalage à gauche

Le fonctionnement des registres à décalage à gauche est semblable à celui des registres à décalage à droite sauf que le sens d'évolution des états des bascules est inversé :



- L'entrée D de la bascule de rang i est reliée à la sortie de la bascule du rang i+1 à chaque FM de CLK.
- Au 1er FM : La donnée présente en ESD est mémorisée en Q_4, tandis que la donnée présente en Q_1 est perdue.
- Au bout de 4 FM de CLK l'état de la 1ère bascule à droite sera disponible sur la sortie de la dernière bascule (D_1)

III.3. Registres à décalage (registres entrée série- sortie parallèle/série)

Un registre à décalage est obtenu comme le montre la figure ***Fig. a*** par la connexion de plusieurs bascules J-K ou R-S, ou comme le montre la figure ***Fig. b*** par l'association de plusieurs bascule D.

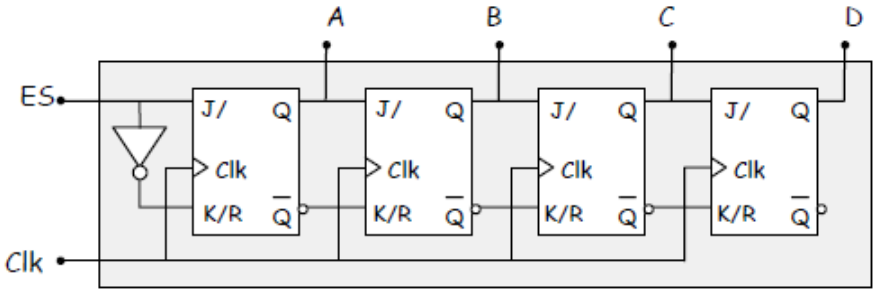


Fig. a : Registre à décalage à bascules JK ou RS, 4 bits entrée série sortie parallèle / série

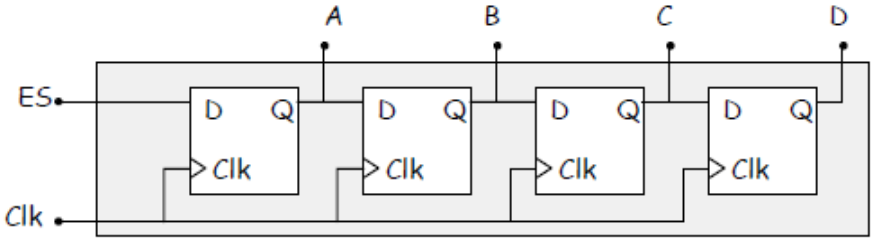


Fig. b : Registre à décalage à bascules D ,4 bits entrée série sortie parallèle / série

A chaque coup d'horloge (en général front montant), la sortie de chaque bascule prend la valeur de la sortie de la bascule qui la précède. ES est l'entrée série. Le mot ABCD constitue la sortie parallèle et SS est la sortie série.

III.4. Registres à décalage (registres entrée parallèle- sortie parallèle)

III.4.1. Registres à décalage avec chargement parallèle synchrone

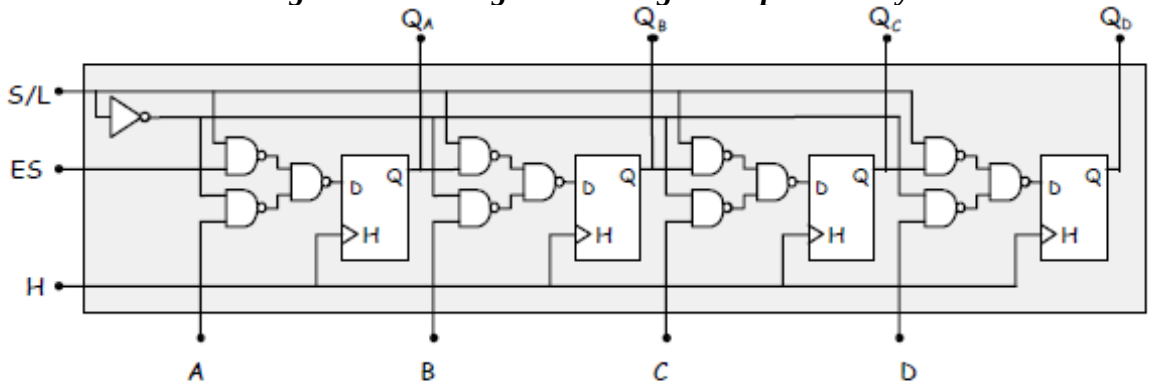
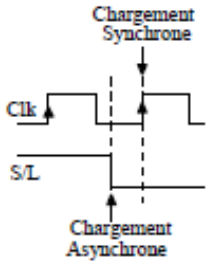


Fig. : Registre à décalage 4 bits entrée parallèle sortie parallèle

L'entrée D de chaque bascule est précédée d'un multiplexeur 1 parmi 2. Si l'entrée S/L (Shift /Load) est "1", on a ***Di = Qi-1***, au coup d'horloge, il y a décalage à droite. Si S/L est "0", ***Di = bit de poids i*** du nombre d'entrée ABCD, au coup d'horloge, Le nombre ABCD est chargé dans $Q_AQ_BQ_CQ_D$.

Parmi les applications de ce genre de registre on trouve la conversion série parallèle ou parallèle série. Dans le premier cas, le registre est placé en mode décalage (S/L=1), on charger le registre en série (4 coups d'horloge sont nécessaires), et on vient lire le nombre de sortie $Q_AQ_BQ_CQ_D$. Dans le deuxième cas, on commence par charger le nombre d'entré ABCD dans le registre (S/L=0 suivi d'un coup d'horloge), puis on repasse en mode décalage (S/L=1) et on envoie une suite de 4 coups d'horloges, à chaque coup d'horloge, un bit est disponible sur la sortie série = Q_D.

Le mode de chargement parallèle décrit ci-dessus est dit chargement **synchrone**, car le chargement se fait au front d'horloge qui suit le passage de S/L à "0".



Le chargement est synchrone avec l'horloge. Il arrive que certaines applications nécessitent que le chargement parallèle se fait au moment ou S/L passe à "0" sans attendre le front d'horloge, on parle alors d'un chargement **asynchrone**.

III.4.2. Registres à décalage avec chargement parallèle asynchrone

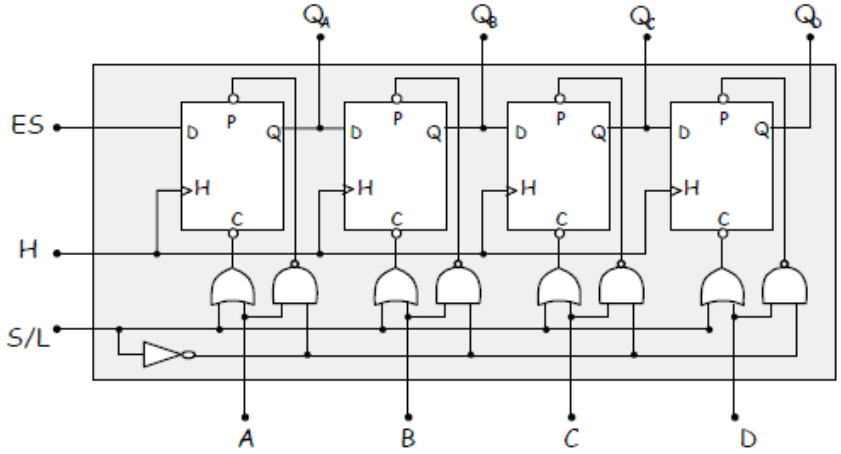


Fig. : Registre à décalage à chargement // asynchrone

III.5. Registre à décalage bidirectionnel =registre parallèle - série

Le registre parallèle série à chargement synchrone permettait de pré positionner son contenu, puis d'effectuer un décalage à droite de celui-ci grâce à l'entrée de commande SHIFT/LOAD.

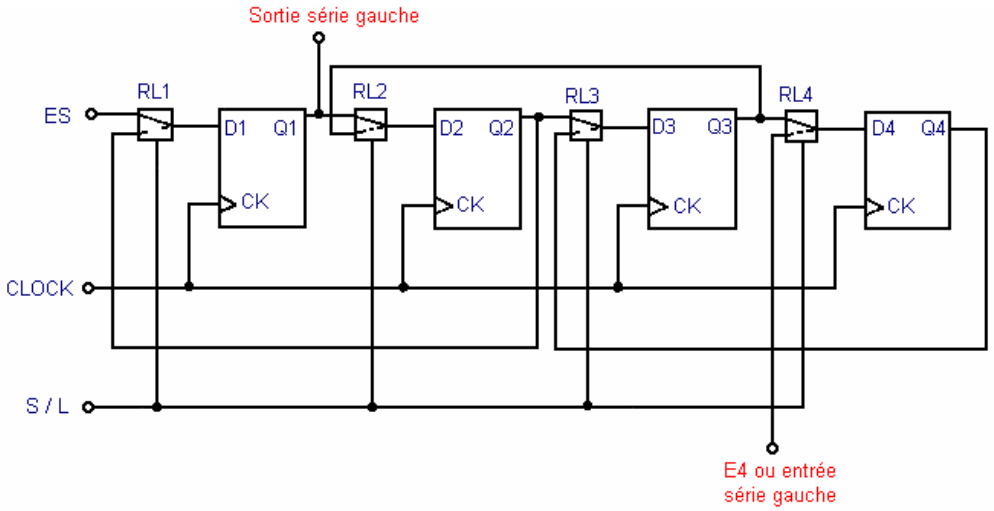


Fig. - Registre parallèle-série câblé pour effectuer un décalage à gauche.

Si S/L = 0 ⇒ l'information se décalait, de $Q_1 \rightarrow Q_4$, vers la droite. L'entrée ES constitue l'entrée série droite du registre et la sortie Q_4 la sortie série droite.

Si S/L = 1 ⇒ l'information se décalait, $Q_4 \rightarrow Q_1$, vers la gauche. L'entrée E4 devient l'entrée série gauche et la sortie Q1, la sortie série gauche.
En résumé, le registre examiné fonctionne soit en mode décalage à droite, soit en mode décalage à gauche.

Analysons par exemple, le premier réseau combinatoire RL1 du montage de la figure ci-dessus, les trois autres étant strictement identiques au premier. Supposons que lorsque l'entrée S/L=0, D1 = ES et que D1 = Q2 lorsque l'entrée S/L=1 (l'inverse pouvant exister). La TV illustrant le fonctionnement du réseau RL1 est la suivante :

Entrées			Sorties
SHIFT / LOAD	ES	Q2	D1
0	ES	X	ES
1	X	Q2	Q2

Fig. . - Table de vérité du réseau RL1.

- Le réseau RL1 réalisé à l'aide des portes NAND est donné par le schéma suivant :

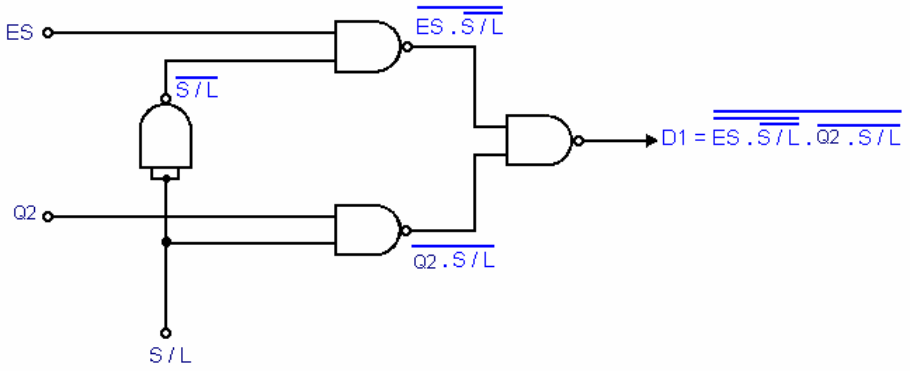


Fig. - Réseau RL1 réalisé à l'aide de portes NAND.

III.6. Registre à décalage universel

Pour différencier les trois modes suivants, chargement parallèle, décalage à droite, décalage à gauche, deux entrées de commande sont nécessaires. Celles-ci, appelées S_0 et S_1, permettent de différencier quatre modes de fonctionnement. Or, trois modes sont prévus. Le quatrième permettra d'inhiber l'action de l'horloge. Le tableau de la figure ci-dessous indique la correspondance entre chacun des modes de fonctionnement et chacune des combinaisons des entrées S_0 et S_1.

S0	S1	Modes de fonctionnement
0	0	Inhibition de l'horloge
0	1	Décalage à gauche
1	0	Décalage à droite
1	1	Chargement parallèle

Fig. - Correspondance entre les modes de fonctionnement et les combinaisons des entrées S0 et S1.

Pour obtenir ces 4 modes, il faut remplacer chacun des réseaux d'aiguillage du registre précédent, bidirectionnel, par un autre plus complexe. Si nous ajoutons le réseau d'inhibition

d'horloge, nous obtenons le schéma du registre universel quatre bits représenté à la figure ci-dessous. Chacun des réseaux logiques RL1, RL2, RL3 et RL4 a pour rôle d'aiguiller une entrée parmi les trois qui lui sont appliquées vers l'entrée D de la bascule à laquelle il est associé. Cette «commutation» est effectuée par les deux entrées S0 et S1.

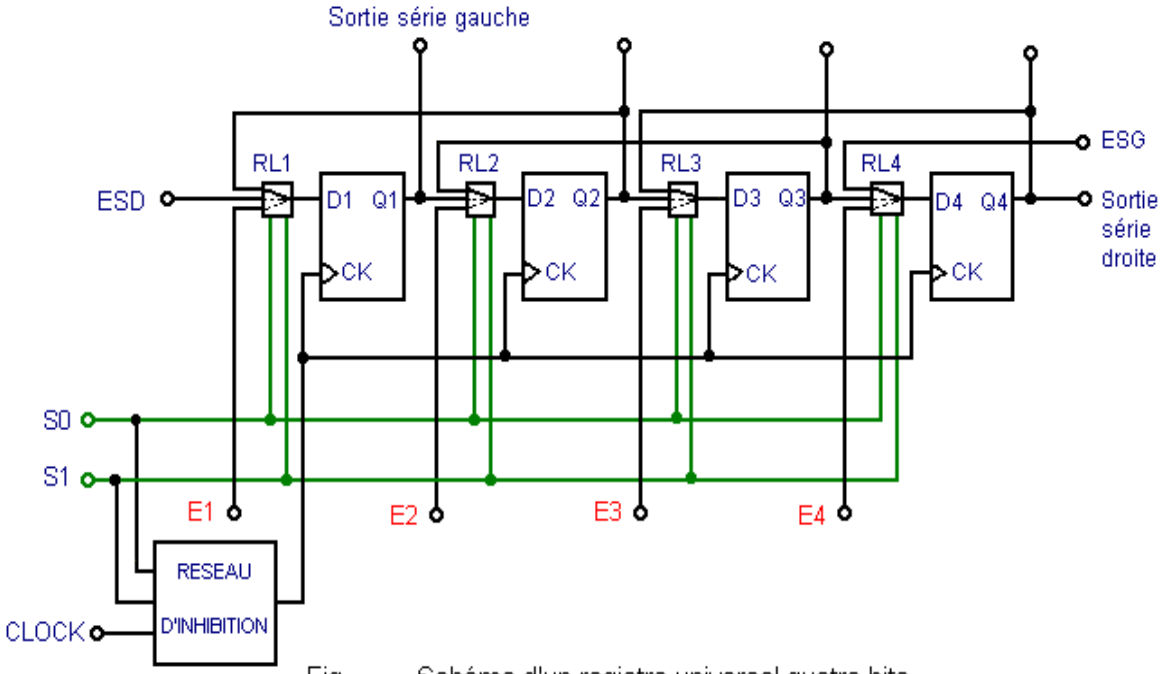


Fig. - Schéma d'un registre universel quatre bits.

Examinons chacun des modes de fonctionnement du registre :

- Si S0 = S1 = 0, le signal d'horloge appliqué sur l'entrée CLOCK n'a pas d'action. Les sorties du registre restent sur leur état.

- Si S0 = 0 et S1 = 1, alors D4 = ESG ; D3 = Q4 ; D2 = Q3 ; D1 = Q2. Le registre est ainsi «câblé» pour effectuer le décalage à gauche.

Les informations à décaler sont appliquées sur l'entrée série gauche (ESG). La sortie série s'effectue sur la sortie Q1 qui est donc la sortie série gauche.

- Si S0 = 1 et S1 = 0, alors D1 = ESD ; D2 = Q1 ; D3 = Q2 et D4 = Q3. Le registre est ainsi «câblé» pour effectuer le décalage à droite.

Les informations à décaler sont appliquées sur l'entrée série droite (ESD). La sortie série s'effectue sur la sortie Q4 qui est donc la sortie série droite.

- Si S0 = 1 et S1 = 1, alors D1 = E1 ; D2 = E2 ; D3 = E3 ; D4 = E4. Le registre est ainsi «câblé» pour effectuer le chargement parallèle.

Les informations à charger sont présentées sur les entrées parallèles E1, E2, E3 et E4. Elles sont mémorisées, à chaque front actif d'horloge, sur les sorties Q1, Q2, Q3 et Q4 du registre.

Nous voyons que tout le fonctionnement d'un registre universel repose sur le fonctionnement des réseaux logiques RL1, RL2, RL3 et RL4. Il est donc nécessaire de donner un complément d'informations sur ceux-ci. (voir TD)