République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Abou Bekr Belkaid - Tlemcen
Faculté de technologie.
Département GEE.

Polycopié

Cours

Logique et Calculateur Numérique

Licence Electronique

Par
MOULAI KHATIR Ahmed Nassim
Année universitaire
2016-2017

Cours

Logique et Calculateur Numérique

Programme :

Chap I : Système de numération et codage.
Chap II : Algèbre de Boole et fonction booléennes.
Chap III : Simplification des fonctions logiques.
Chap IV : Analyse et synthèse des circuits logiques combinatoire.
Chap V : Analyse et synthèse des circuits logiques séquentiels.

Chapitre I : Système de numération et codage

Introduction :

Pour la mesure des grandeurs physiques dans la plus part des domaines (Température, Vitesse, Voltage…), on peut représenter précisément leurs valeurs sous deux formes différentes :

1- Représentation analogique.
2- Représentation numérique.

a - Représentation analogique : Dans cette représentation on fait correspondre à une grandeur physique une autre grandeur physique qui lui est proportionnelle.

Ex : Soit le circuit suivant :

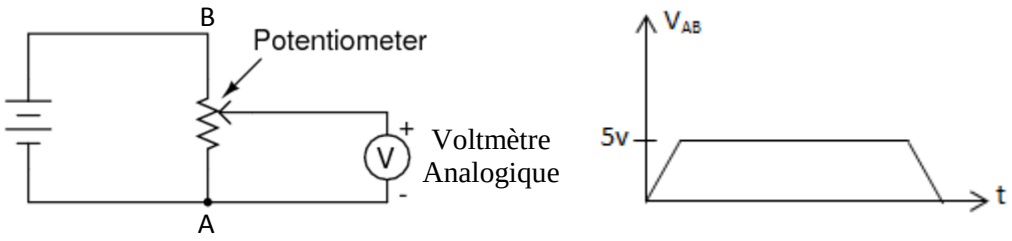


Figure I.1 : Mesure Analogique

Lorsqu'on fait déplacer le curseur du potentiomètre de bas en haut, la tension entre les points A et B croit progressivement de 0V jusqu'à 5V. Si le curseur se déplace de haut en bas, la tension décroit peu à peu de 5V jusqu'à attendre 0V.

Dans ce cas la valeur de la tension V_{AB} est donnée par la position de l'aiguille sur le cadran du voltmètre (donc on a fait correspondre à la grandeur « tension » une autre grandeur physique qui est la position de l'égaille « l'angle θ »
Donc un élément analogique est un dispositif dont la sortie varie d'une manière continue avec l'entrée

b- Représentation Numérique (Digitale): Prenons l'exemple précèdent; cette foi si en remplace le voltmètre analogique par un voltmètre numérique. Dans ce cas on fait correspondre à la tension V_{AB} un nombre.

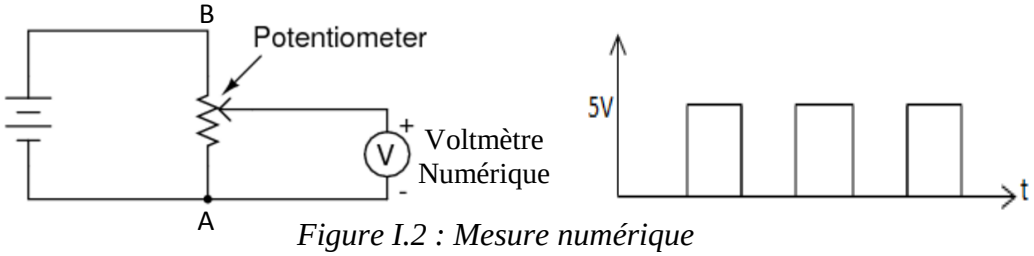


Figure I.2 : Mesure numérique

En déplaçant le curseur, on peut lire sur le cadran du voltmètre les chiffres 0, 1, 2, 3, 4 …
On remarque que la sortie varie de façon discrète (discontinue).
Les circuits numériques digitaux ne fonctionnent qu'à l'aide des signaux discrets.
L'afficheur utilisé ici est constitué de quatre segments qui ne peuvent être allumés qu'à l'aide d'un signal carré

Ces circuits n'admettent que deux valeurs significatives :

- Une tension haute (de niveau haut) désignée par la valeur 1 (logique 1 ou état 1).
- Une tension basse (de niveau bas) désignée par la valeur 0 (logique 0 ou état 0).

Pourquoi utiliser des circuits digitaux :
Les systèmes digitaux sont nécessaires quand l'information doit être utilisée dans d'autres calculs ou affichage (en nombres et/ou en lettres).

Les avantages des circuits digitaux :

1- L'information peut être stockée
2- Les données peuvent êtres utilisées pour des calcules précis
3- La possibilité de programmation (rendre un système intelligent) ↔ compatibilité avec les micro-ordinateurs.
4- Les circuits digitaux sont disponibles sur des circuits intégrés pas chers.

Les limites :

1- La majorité des phénomènes réels sont de nature analogique.
2- Le traitement analogique est généralement plus simple et plus rapide.

I- Systèmes de numération :

L'ensemble des outils informatiques sont basés sur les mêmes principes de calcul (loi de tout ou rien). Les calculs habituels sont effectués dans le système de numération décimal, par contre le calculateur électronique ne peut pas utiliser ce système car le circuit électronique ne permet pas de distinguer 10 états. Le système de numération binaire ne comportera que 2 états 0 et 1.

I.1- la valeur du digit (chiffre):

Considérons le nombre 528 :
Le digit 5 représente 500 à cause de sa 3eme position à gauche du point décimal.
Le digit 2 représente 20 à cause de sa 2eme position à gauche du point décimal.
Le digit 8 représente 8 à cause de sa 1ere position à gauche du point décimal.

Le nombre total représente donc 500+20+8
- On remarque que chaque digit a un poids en fonction de sa position.

- Ceci est valable pour tous les systèmes de numérotation qui existent (pas uniquement pour le système décimal).
Tout nombre $\mathbf{N_b}$ qui comprend **n** chiffres donné en base **b** peut être exprimé comme la somme de ses coefficients pondérés.

$N_b = x_n x_{n-1} x_{n-2} \ldots x$
$N_b = x_n b^n + x_{n-1} b^{n-1} + x_{n-2} b^{n-2} + \ldots\ldots + x_0 b^0$
$N_b = \sum_{i=0}^{n} x_i b^i$

Ex. $528 = 5.10^2 + 2.10^1 + 8.10^0$

I.2- Système décimal :

Le système décimal est celui dans lequel nous avons le plus l'habitude d'écrire. Chaque chiffre peut avoir 10 valeurs différentes : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, de ce fait, le système décimal a pour base 10.

Développement en polynôme d'un nombre dans le système décimal
• Soit le nombre 1978, ce nombre peut être écrit sous la forme suivante :
$1978 = 1000 + 900 + 70 + 8$
$1978 = 1*1000 + 9*100 + 7*10 + 8*1$
$1978 = 1*10^3 + 9*10^2 + 7*10^1 + 8*10^0$

Cette forma s'appelle la forme polynomiale
Un nombre réel peut être écrit aussi sous la forme polynomiale
$1978,265 = 1*10^3 + 9*10^2 + 7*10^1 + 8*10^0 + 2*10^{-1} + 6*10^{-2} + 5*10^{-3}$.

Cependant il est possible d'imaginer d'autres systèmes de nombres ayant comme base un nombre entier différent.

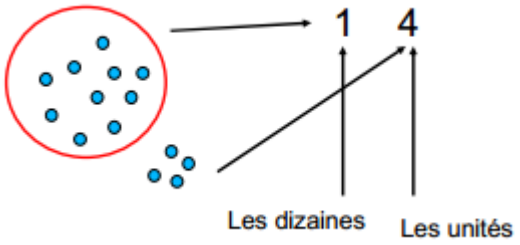
I.3- Système octal :

Le système octal utilise un système de numération ayant comme base 8 (octal => latin : octo = huit). Dans ce système nous n'aurons plus 10 symboles mais 8 seulement : 0, 1, 2, 3, 4, 5, 6, 7
Ainsi, un nombre exprimé en base 8 pourra se présenter de la manière suivante : $(745)_8$. Cette base obéira aux mêmes règles que la base 10, vue précédemment, ainsi on peut décomposer $(745)_8$ de la façon suivante :
$(745)_8 = 7 \times 8^2 + 4 \times 8^1 + 5 \times 8^0$
$(745)_8 = 7 \times 64 + 4 \times 8 + 5 \times 1$
$(745)_8 = 448 + 32 + 5$
Nous venons de voir que : $(745)_8 = (485)_{10}$.

I.4- Système binaire :

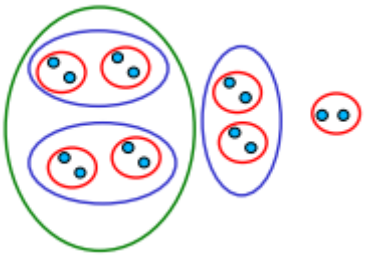
Dans le système binaire, chaque chiffre peut avoir 2 valeurs différentes : 0, 1. De ce fait, le système a pour base 2.

Exemple illustratif :
Supposons qu'on a 14 jetons, si on forme des groupes de 10 jetons. On va obtenir 1 seul groupe et il reste 4 jetons.



.

. Maintenant on va former des groupes de 2 jetons (on obtient 7 groupes)
. Par la suite on va regrouper les 7 groupes 2 à 2 (on obtient 3 groupes).
. On va regrouper ces derniers aussi 2 à 2 (on obtient 1 seul groupe)
. Le schéma illustre le principe :



Nombre de jetons qui restent en dehors des groupes : 0
Nombre de groupes qui contiennent 2 jetons : 1
Nombre de groupes qui contiennent 2 groupes de 2 jetons : 1
Nombre de groupes qui contiennent des groupes de 2 groupes de 4 jetons : 1
Si on regroupe les différents chiffres on obtient : 1110
1110 est la représentation de 14 dans la base 2

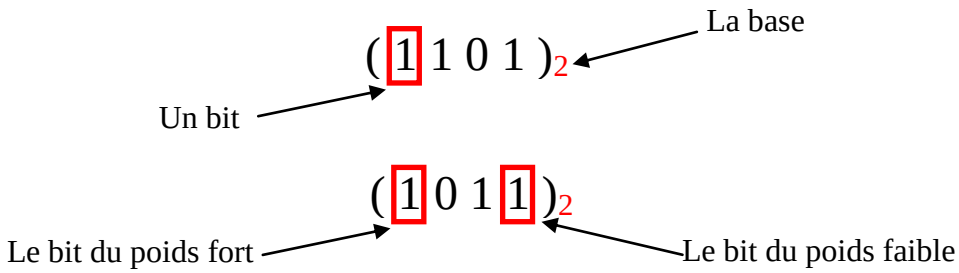


Figure I.3 : Poids d'un nombre binaire

Comptage en binaire :

Sur un seul bit : 0 , 1

Sur deux bits : 4 combinaisons = 2^2

Binaire	Décimal
00	0
01	1
10	2
11	3

Sur trois bits : 8 combinaisons = 2^3

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

Table I.1 : Comptage en binaire

I.5- Système hexadécimale :

Le système hexadécimal utilise les 16 symboles suivant :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F : De ce fait, le système a pour base 16.

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

Table I.2 : Correspondance Décimal-Hexadécimal-Binaire

Un nombre exprimé en base 16 pourra se présenter de la manière suivante : $(5AF)_{16}$
La correspondance entre base 2, base 10 et base 16 est indiquée dans le tableau I.2 :

Le nombre $(5AF)_{16}$ peut se décomposer comme suit :
$(5AF)_{16} = 5 \times 16^2 + A \times 16^1 + F \times 16^0$

En remplaçant A et F par leur équivalent en base 10, on obtient :
$(5AF)_{16} = 5 \times 16^2 + 10 \times 16^1 + 15 \times 16^0$
$(5AF)_{16} = 5 \times 256 + 10 \times 16 + 15 \times 1$
donc = $(5AF)_{16} = (1455)_{10}$

II- Transcodage :

En électronique digital presque tous les circuits digitaux (calculateurs, ordinateurs) ne comprennent que les nombres binaires, mais les gens ne comprennent que les nombres décimaux. Donc nous devons traduire ou convertir du décimal au binaire et du binaire au décimal.

II.1. Conversion binaire-décimal :

Soit le nombre binaire $(110011)_2$

	2^5	2^4	2^3	2^2	2^1	2^0
N=	1	1	0	0	1	1

• point binaire

$N=1.2^0+1.2^1+0.2^2+0.2^3+1.4^2+1.2^5$
La conversion du nombre binaire N dans son équivalent décimal est immédiate, Elle découle du calcul effectué dans la base 10
$N=1+2+16+32= (51)_{10}$

II.2. Conversion décimal - binaire :

Cas d'un nombre Entier :

Exemple : $(35)_{10} = (??)_2$

Après divisions on a : $(35)_{10} = (100011)_2$

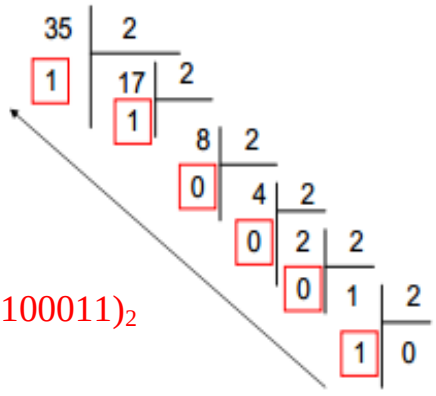


Figure I.4 : Devisions successives par 2

1er Méthode : La conversion d'un nombre décimal N dans son équivalent binaire peut être effectuée par une succession de devisions par 2 jusqu'à l'obtention d'un quotient nul (*Fig. I.4)* et prendre le reste des divisions dans l'ordre inverse.

2eme Méthode : Soustraction successive
La méthode consiste à retrancher du nombre N_{10} la plus grande puissance entière de contenue dans N_{10}. On considère ensuite le reste, et on recommence l'opération jusqu'à l'obtention de 0

Cas d'un nombre réel :
Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle.
• La partie entière est transformée en effectuant des divisions successives.
• La partie fractionnelle est transformée en effectuant des multiplications successives par 2.

Exemple

$35{,}625 = (?)_2$
Partie Entière : $35 = (100011)_2$
Partie Fractionnelle : $0{,}625 = (?)_2$

$0{,}625 * 2 = 1{,}25$
$0{,}25 * 2 = 0{,}5$
$0{,}5 * 2 = 1{,}0$

$(0{,}625) = (0{,}101)_2$
Donc : $35{,}625 = (100011{,}101)_2$

Exemple 2
$(0{,}4)10 = (??)_2$

$0{,}4 * 2 = 0{,}8$
$0{,}8 * 2 = 1{,}6$
$0{,}6 * 2 = 1{,}2$
$0{,}2 * 2 = 0{,}4$

$(0{,}4) = (0{,}0110)_2$

Remarque
Le nombre de bits après la virgule détermine la précision de la conversion

II.3. Conversion du décimal à une base X :

La conversion se fait en prenant les restes des divisions successives sur la base X dans le sens inverse.

Exemple

$35 = (??)_3$
$35 = (1022)_3$

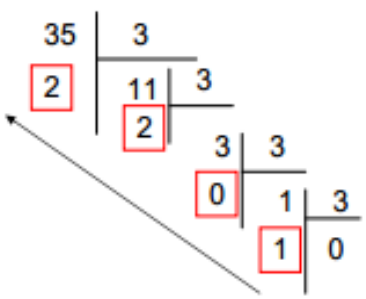
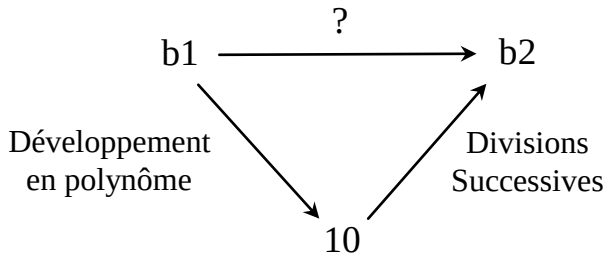


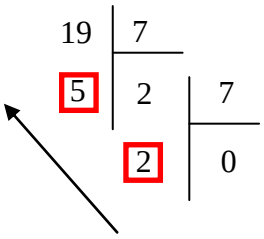
Figure I.5 : Devisions successives par 3

II.4. Conversion d'une base b1 à une base b2 :

• Il n'existe pas de méthode pour passer d'une base b1 à une autre base b2 directement.
• L'idée est de convertir le nombre de la base b1 à la base 10 , en suit convertir le résultat de la base 10 à la base b2 .



Exemple
$(34)_5 = (?)_7$
$(34)_5 = 3*5^1 + 4 * 5^0 = 15 + 4 = (19)_{10} = (19)_{10} = (?)_7$



$(19)_{10} = (25)_7$
$(34)_5 = (25)_7$

II.5. Conversion : binaire – octal

L'avantage que présentent les systèmes octal et hexadécimal est la facilité de la conversion directe au système binaire.
Pour passer d'un nombre écrit en binaire a son équivalent octal, il faut diviser ce nombre en groupement de 3 bits commençant du point décimal (point binaire), et remplacer chaque groupement par son équivalent en octal.

Exemple :
$(345)_8 = (\underline{011}\ \underline{100}\ \underline{101})_2$
$(65,76)_8 = (\underline{110}\ \underline{101}, \underline{111}\ \underline{110})_2$
$(35,34)_8 = (\underline{011}\ \underline{101}, \underline{\mathbf{0}11}\ \underline{100})_2$

Octal	Binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Remarque : Le remplacement se fait de droit à gauche pour la partie entière et de gauche à droite pour la partie fractionnelle.

II.6. Conversion : octal – binaire :

Pour passer du système octal au système binaire, il suffit d'inverser le processus i.e. écrire chaque chiffre (digit) en octal en son équivalent binaire sur 3 bits.

Exemple: soit à convertir le nombre octal $(254)_8$
Octal : 2 5 4
Binaire : 010 101 100 = $(10101100)_2$
Ou tout simplement passer par la base 10 en suite effectuer des divisions successives par 2

II.7. Conversion hexadécimal – binaire :

En Hexa chaque symbole de la base s'écrit sur 4 bits
L'idée de base est de replacer chaque symbole par sa valeur en binaire sur 4 bits (faire des éclatements sur 4 bits).

Exemple
$(345B)_{16} = (0011\ 0100\ 0101\ 1011)_2$
$(AB3,4F6)_{16} = (1010\ 1011\ 0011,\ 0100\ 1111\ 0110)_2$

II.8. Conversion binaire – hexadécimal :

L'idée de base est de faire des regroupements de 4 bits à partir du poids faible.
Par la suite remplacer chaque regroupement par la valeur Hexa correspondante.

Exemple
$(11001010100110)_2 = (0011\ 0010\ 1010\ 0110)_2 = (32A6)_{16}$
$(110010100,10101)_2 = (0001\ 1001\ 0100,\ 1010\ 1000)_2 = (194,A8)_{16}$

III - Opérations arithmétiques en binaire :

III.1. Addition :

Tableau d'addition :

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

Avec retenue de 1

Exemple

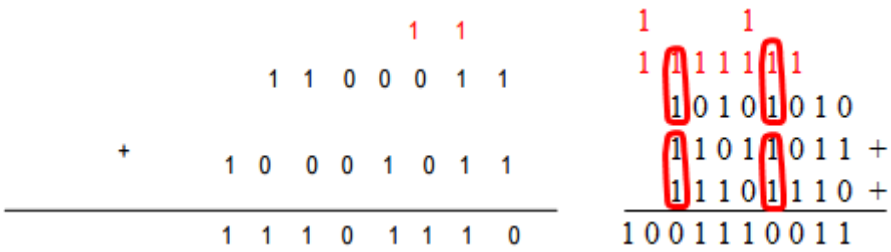


Figure I.6 : Addition Binaire

III.2. Soustraction :

Tableau de soustraction :

0 - 0 = 0	
0 - 1 = 1	Avec retenue de 1
1 - 0 = 1	
1 - 1 = 0	

Ex :

```
 1 11 10 10        0 1 10 10 10 1 1 10
10 11 10  1        0 10 11 11 1 0 10 1
 0  1  1  1        0 0  0  0  1 1 0  1
```

Figure I.7: Soustraction Binaire

III.3. Multiplication :

Tableau de multiplication :

0 X 0 = 0
1 X 0 = 0
0 X 1 = 0
1 X 1 = 1

```
        1 0 1 1              1 0 0 1, 0 1 1
        1 1 0 1            1 0 1 0 0, 1 1
      ---------          ---------------
        1 0 1 1              1 0 0 1 0 1 1
      1 0 1 1 . .          1 0 0 1 0 1 1 .
    1 0 1 1 . . .        1 0 0 1 0 1 1 . . . .
  ---------------      1 0 0 1 0 1 1 . . . . . .
  1 0 0 0 1 1 1 1      ---------------------------
                       1 1 0 0 0 0 1 0, 1 0 0 0 1
```

Figure I.8 : Multiplication Binaire

III.4. Division :

Tableau de division :

0 : 0 = indéterminé
0 : 1 = 0
1 : 0 = indeterminé
1 : 1 = 1

La division binaire s'effectue à l'aide de soustractions et de décalages, comme la division décimale, sauf que les digits du quotient ne peuvent être que 1 ou 0. Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0

Exemple de division

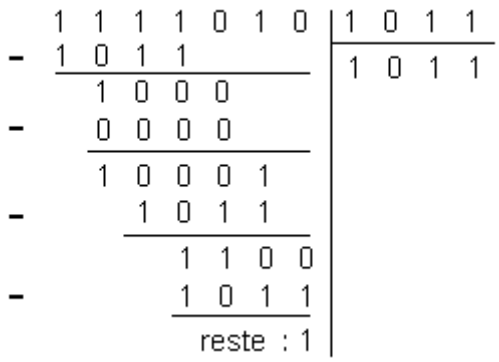


Figure I.9 : Division Binaire

IV. Les codes Binaires

Pour des besoins de transmission d'informations un nombre important de codes a été créé :

IV.1. Le code complément à deux

Le complément à deux est une représentation binaire des entiers relatifs qui permet d'effectuer les opérations arithmétiques usuelles naturellement.
Dans une telle écriture on utilise le bit de poids fort (bit le plus à gauche) du nombre pour contenir la représentation de son signe (le zéro étant considéré comme positif et le 1 comme négatif).
La première idée est de marquer le signe du nombre de façon simple : le signe puis la représentation de sa valeur absolue.
Ainsi sur 8 bits y compris le bit de signe :

Bit se signe	Données	Décimal
0	0000010	+2
1	0000010	-2

Malheureusement cette représentation possède deux inconvénients.
• Le nombre zéro (0) possède deux représentations: 0 0000000 et 1 0000000 sont respectivement égaux à 0 et -0.
• L'autre inconvénient (majeur) est que cette représentation n'est pas compatible avec l'addition usuelle.
Ainsi:

0 0000011 + 1 0000100 = 1 0000111
Soit 3 + (-4) = (-7) au lieu de (-1)

Pour la résolution de ce problème on utilise la notation en complément à deux.
Les nombres positifs sont représentés comme attendu avec un bit de signe à 0
Les nombres négatifs sont obtenus de la manière suivante :

• On complémente à 1 les bits de l'écriture binaire de sa valeur absolue (Inversion de tout les bits).
• On ajoute 1 au résultat trouvé.

Les deux inconvénients précédents disparaissent alors.
Ex :
Pour coder (-4) :
• On prend le nombre positif + 4 : 0 0000100
• On inverse les bits : 1 1111011
• On ajoute 1 : 1 1111100 correspond à – 4 en Complément à 2

Le bit de signe est automatiquement mis à 1 par l'opération d'inversion.
On peut vérifier que cette fois l'opération 3 + (-4) se fait sans erreur :

0 0000011 + 1 1111100 = 1 1111111

Le complément à deux de 1 1111111 est 0 0000001 soit 1 en décimal, donc 1 1111111 = (-1) en décimal.
Le résultat de l'addition usuelle de nombres représentés en complément à deux est le codage en complément à deux du résultat de l'addition des nombres.

IV.2. Le code binaire naturel (Base 2) ou code 1248 :

Décimal	Binaire	
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	
10	1010	
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

Table I.3 : Code Binaire Naturel

IV. 3. Le Code Binaire Réfléchi (Code Gray):

Si on utilise le code binaire naturel pour effectuer un montage en risque d'avoir un problème en comptage avec les chiffres qui changes en même temps, La solution est d'utiliser le code réfléchi (Gray).

Décimal	Binaire Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Table I.4 : Code Binaire Réfléchi

IV. 4. Le Code BCD (Binary Code Decimal):

C'est la traduction en binaire de chaque chiffre décimal.
Chaque chiffre est exprimé séparément par un demi-octet (quatre bits).

Décimal	Binaire Gray
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table I.5 : Code Binaire Décimal (BCD)

Ex.

$(155)_{10} \xrightarrow{BCD} (0001\ 0101\ 0101)_{BCD}$

$$\begin{array}{lll} 33 & & 0011\ \ 0011 \\ \underline{23} & \xrightarrow{BCD} & \underline{0010\ \ 0011} \\ 56 & & 0101\ \ 0110 \end{array}$$

IV. 5. Code avec Parité

C'est un code binaire pour le quel on rajoute un bit de parité pour savoir si la transmission s'est bien effectuée sans erreurs. On a deux possibilités de parité :

Parité paire :

Dans le cas de la parité paire on compte le nombre de 1 contenu dans le nombre y compris le bit de parité, si ce nombre est paire on ajoute un 0 si non on ajoute un 1 comme bit de parité.

$108_{10} = (01101100)_2$
$= (\mathbf{0}01101100)_{2PP}$

$100_{10} = (01100100)_2$
$= (\mathbf{1}01101100)_{2PP}$

Parité impaire :

Dans le cas de la parité paire on compte le nombre de 1 contenu dans le nombre y compris le bit de parité, si ce nombre est impaire on ajoute un 0 si non on ajoute un 1 comme bit de parité.

$108_{10} = (01101100)_2$
$= (\mathbf{1}01101100)_{2PI}$

$100_{10} = (01100100)_2$
$= (\mathbf{0}01101100)_{2PI}$

Chapitre II : Algèbre de Boole et fonction booléennes.

Introduction :

Les circuits électroniques sont classés en deux grandes catégories :
Les circuits digitaux (numériques) et les circuits analogiques
Les circuits analogiques ont une amplitude variable continuellement, par contre un circuit digital est un circuit présentant un signal qui n'admet que deux niveaux (Ex +5V et 0V) équivalent au nivaux 1 et 0.
Un interrupteur est par exemple un circuit digital

A= 0 : Interrupteur Ouvert

A=1 : Interrupteur Fermé

I. Algèbre de Boole et opérateurs logiques

En électronique numérique le jeu consiste à produire des sorties numériques (fonction logiques) a partir d'entrées numériques (variables logiques).

I.1. Définitions

L'algèbre de Boole est l'ensemble B=$\{0,1\}$ muni de trois lois élémentaires.

- Deux lois de composition : "ou" noté "x+y" (loi d'addition logique et "ET" noté "x.y" (loi de multiplication logique).
- Une loi de complémentation (négation) "Non" noté $\overline{X}$

Avec X et Y sont deux éléments de l'ensemble B.
Ces lois peuvent être réalisées à partir d'un groupe d'opérations élémentaires appelées portes logiques (opérateurs logiques).

I.2. Opérateurs logiques

Chaque opérateur est représenté par un symbole, et sa fonction par une table de vérité. Ces opérateurs (portes logiques) sont conçus actuellement à base de Transistor qui est l'élément qui présente la brique de construction des circuits électroniques comme les puces, les micro-processeurs ... La technologie actuellement utilisée pour fabriquer ces portes est la technologie MOS (Metal-Oxide-Semiconductor). Il existe deux types de transistors MOS : les transistors de type *n* et les transistors de type *p*.

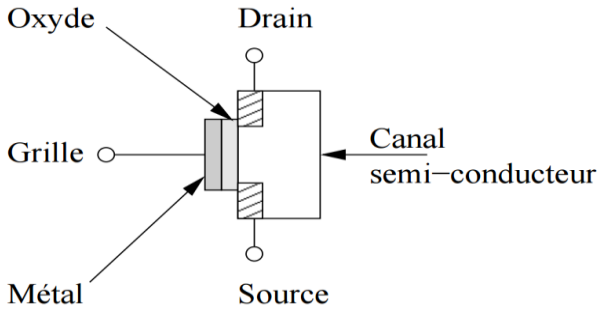


Figure II.1 : Le transistor MOS

Un tel transistor (*Fig.II.1)* est composé de trois broches appelées drain, grille et source. Les rôles du drain et la source sont presque symétriques et interchangeables.

Le fonctionnement de ce transistor est semblable à un interrupteur électrique entre la source et le drain commandé par la

- Pour un n-MOS : Si la grille est mise à une tension positive, la source et le drain sont connectés. Si au contraire, la grille est mise à une tension de 0V, le circuit entre la source et le drain est ouvert.
- Pour un p-MOS c'est l'inverse : Le drain et la source sont connectés lorsque la tension appliquée à la grille est 0V

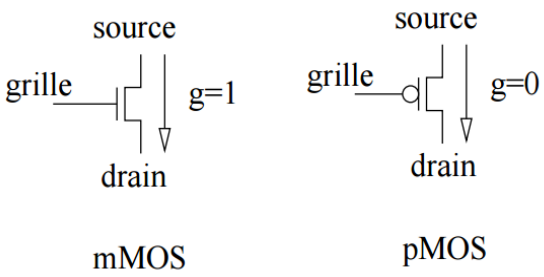
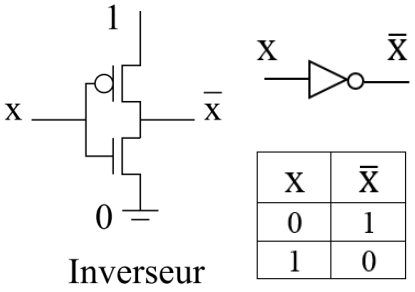


Figure. II.2 : Les types nMOS et pMOS

Les micro-processeurs actuels utilisent des transistors des deux types. On parle alors de technologie CMOS (Complementary Metal-Oxide-Semiconductor).

Cette technologie (*Fig.II.3)* utilise les deux types du transistor (n-MOS et p-MOS), Si on applique un niveau haut à l'entrée, le transistor N est passant et le P est bloqué. La sortie est donc à l'état bas. Inversement, si on applique un état bas à l'entrée, le transistor N est bloqué et le P est passant. Donc la sortie est à l'état haut. Ce mécanisme présente une fonction d'inversion. Qui forme la porte de négation



x	$\bar{x}$
0	1
1	0

Figure. II. 3 : L'inverseur CMOS

I.1. La Porte Non (porte inverse)

C'est une porte à une seule entrée, la variable de sortie est l'inverse de l'entrée.

Symbole:

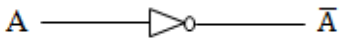


Table de vérité :

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

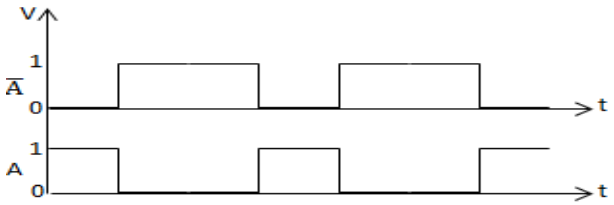
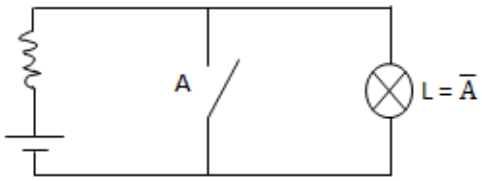


Figure. II. 4 : Diagramme temporel d'une porte "Non"

La sortie est égale à 1 si et seulement si l'entrée est égale à 0 et réciproquement.

Exemple

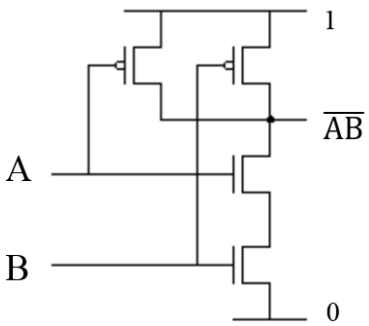


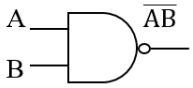
La lampe ne s'allume (L=1) que si A (l'interrupteur) est ouvert (A=0)

I.2. La porte NAND

La porte de négation de AND *(NAND)* présente en entrée deux valeurs 0 ou 1. Si les deux entrées sont à état haut, la sortie est à niveau bas. Et si au moins une des deux entrées est à un niveau bas, cette sortie vaut 1.

Le circuit de la porte *NAND* en logique CMOS peut être formé de quatre transistors dont deux n-MOS et deux p-MOS.





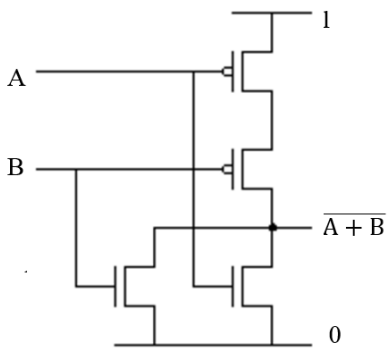
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

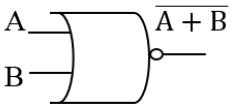
Figure. II. 5 : Schéma et symbole de la porte NAND

I.3. La porte NOR

La porte NOR (Non de OR) (Fig. II.6) prend en entrée deux valeurs 0 ou 1. La sortie est basse si au moins une des entrées est à 1 et elle est haute si les deux entrées sont à 0.

En logique CMOS, le circuit de la porte NOR peut être former de quatre transistors dont deux n-MOS et deux p-MOS.





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

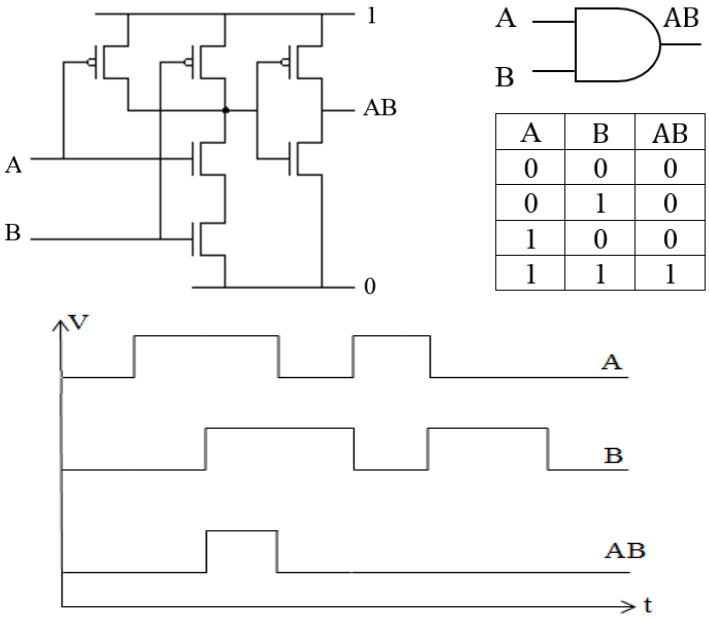
Figure. II. 6 : Schéma et symbole de la porte NOR

Les deux transistors P en parallèle dans le circuit du *NAND* sont en série dans le circuit de la porte *NOR*, Et contrairement, les deux transistors N en série dans le circuit de la porte *NAND* sont en parallèle dans le circuit de la porte *NOR*. Donc le circuit de la porte NOR (Fig. II.6) est le circuit dual du celui de la porte *NAND*

I.4. Les portes AND et OR :

En remplaçant chaque transistor n-MOS du schéma de la porte *NOR* par un transistor p-MOS et inversement, on obtient le schéma de la porte *AND*.

Les circuits des portes *AND* et *OR* sont respectivement obtenus en combinant un circuit de la porte *NAND* et *NOR* avec un inverseur.



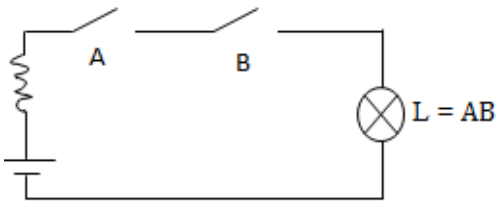
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

Figure. II. 7 : Schéma et symbole de la porte AND

Fonctionnement d'une porte AND

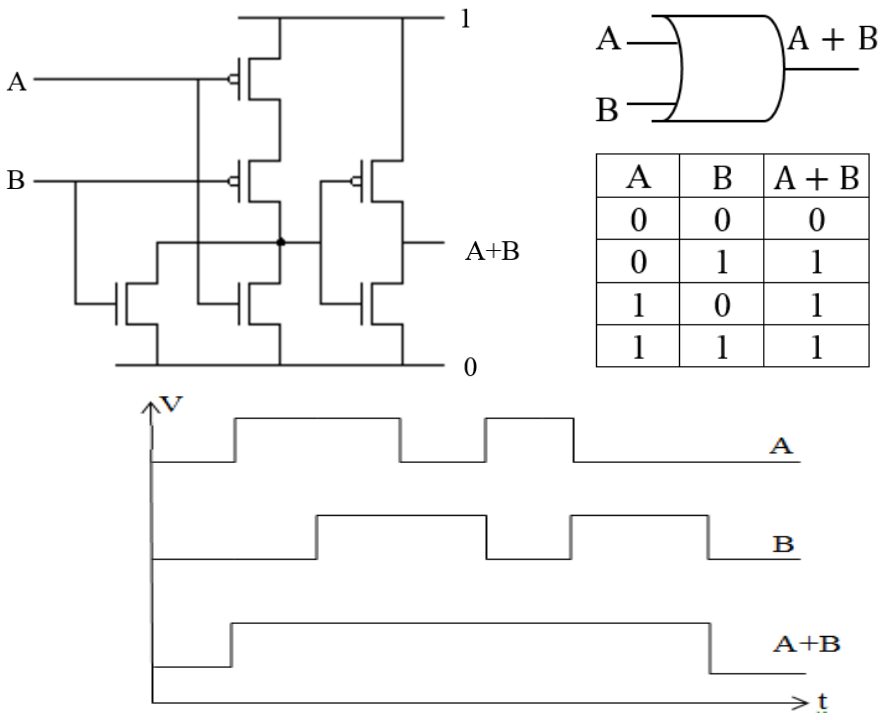
A.B = 1 si et seulement si A =1 et B=1

Exemple



La porte *AND (Fig. II.7)* prend en entrée deux valeurs 0 ou 1. La sortie est au niveau haut si les deux entrées valent 1 et elle est au niveau bas sinon. La table de vérité est donnée ci-dessous.

La porte *OR (Fig. II.8)* prend en entrée deux valeurs 0 ou 1. La sortie est au niveau bas si les deux entrées valent 0 et elle est au niveau haut sinon.



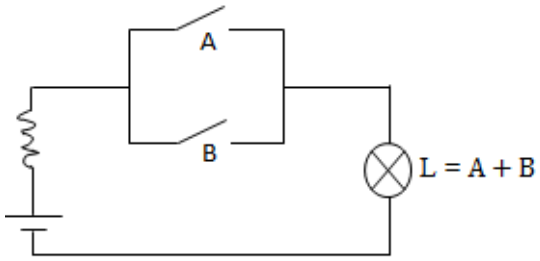
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Figure. II. 8 : Schéma et symbole de la porte OR

Fonctionnement de la porte OU (OR)

A+B =1 si et seulement si l'une au moins des deux variables vaut 1.

Exemple :



I.5. La Porte OU exclusif (XOR) :

Soient A et B les entrées d'une porte XOR ; la sortie s'écrit dans ce cas : A ⊕ B

Symbole :

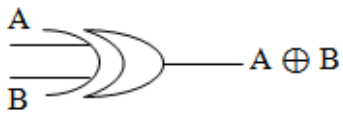


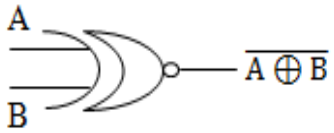
Table de vérité :

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

La sortie A ⊕ B vaut 1 si les deux entrées sont différentes

I.6. La Porte NON OU exclusif (NXOR) :

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



La sortie $A \oplus B$ vaut 1 si les deux entrées sont égales.

II. Les différentes relations logiques :

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Loi de commutativité : $\begin{cases} A + B = B + A \\ A.B = B.A \end{cases}$

Loi de distributivité : $\begin{cases} A(B + C) = AB + AC \\ A + (BC) = (A + B).(A + C) \end{cases}$

Loi d'association : $\begin{cases} A + (B + C) = (A + B) + C = A + B + C \\ A\,(B\,C) = (A\,B)\,C = A\,B\,C \end{cases}$

Elements Neutres : $\begin{cases} A + 0 = A \\ A\,.\,1 = A \end{cases}$

Elements Absorbants : $\begin{cases} A + 1 = 1 \\ A\,.0 = 0 \end{cases}$

Loi d'Indépendance : $\begin{cases} A + A = A \\ A.A = A \end{cases}$

Loi de Complémentarité : $\begin{cases} A + \bar{A} = 1 \\ A.\bar{A} = 0 \end{cases}$

Lois d'absorption :

$A + AB = A$

$A(A + B) = A$

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

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

II.1. Théorème de MORGAN

1er théorème : Le complément d'un produit des variables est égal à la somme des compléments des variables.

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

2em théorème : Le complément d'une somme des variables est égal au produit des compléments des variables.

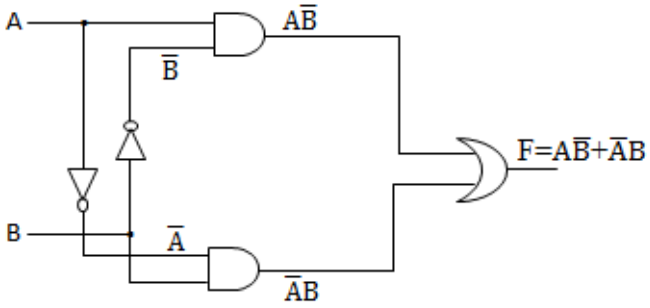
$$\overline{A + B + C} = \overline{A}.\overline{B}.\overline{C}$$

II.2. Les fonctions logiques

Toute fonction logique peut être réalisée de la manière suivante :
Soit avec des opérateurs ET, OU, NON (AND, OR, NOT)
Soit avec des opérateurs NON ET (NAND)
Soit avec des opérateurs NON OU (NOR)
On appel fonction logique une combinaison de plusieurs variables booléenne (0 ou 1) reliées par des opérateurs logiques.

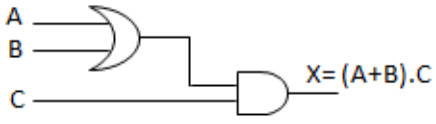
Ex.
Soit F=A$\overline{B}$ + $\overline{A}$B

Logigramme:



Mise sous forme algébrique :
Tout circuit logique; quelque soit sa complexité peut être décrit au moyen des opérations booléenne

Ex.



Sans les parenthèses, notre interprétation sera erronée
Car x=A+B.C signifie que A est réuni dans une porte OU avec le produit B.C.
Pour lever cette indétermination, c'est les opérateurs ET qui sont appliqués en premier, sauf s'il y'a des parenthèses.

II.3. Table de vérité

On peut représenter une fonction logique de n variables $f(x_1, x_2, \ldots\ldots x_n)$ par un tableau de 2^n lignes et n+1 colonnes (n colonnes pour n variables et une colonne pour la fonction).
L'ordre des combinaisons sera croissant du haut vers le bas, et elles seront écrites dans le code binaire naturel (CBN).

x_n	. . . . .	x_2	x_1	F
0	. . . .	0	0	.
.		:	:	.
.		:	:	.
.		:	:	.
1	. . . . .	1	1	.

Table II.1 : Table de vérité pour une fonction à n Variables

Exemple:
$F=\overline{A}B+B\overline{C}+ABC = F(A, B, C)$
3 variables $\rightarrow 2^3$ combinaisons 8 lignes.

Equivalent Décimal	A	B	C	$\overline{A}$	$\overline{C}$	$\overline{A}B$	$B\overline{C}$	ABC	F
0	0	0	0	1	1	0	0	0	0
1	0	0	1	1	0	0	0	0	0
2	0	1	0	1	1	1	1	0	1
3	0	1	1	1	0	1	0	0	1
4	1	0	0	0	1	0	0	0	0
5	1	0	1	0	0	0	0	0	0
6	1	1	0	0	1	0	1	0	1
7	1	1	1	0	0	0	0	1	1

II.4. Formes canoniques :

Toutes les fonctions logiques peuvent se mettre sous 2 formes standards appelées formes canoniques :
1ère forme canonique : somme canonique $\sum\prod$
Une fonction logique de n variables binaires est écrite sous la forme d'une somme canonique si chacun de ces termes (monômes) est un produit de n éléments pris parmi les n variables ou leurs compléments.
Ex.
F = AB + AB + B
2eme forme canonique : produit canonique $\prod\sum$
Une fonction logique de n variables binaires est écrite sous la forme d'un produit de somme si chacun de ses termes est une somme de n éléments pris parmi les n variables ou leurs compléments.
Ex.
$F = (\overline{A}+B).(A+B).(A+\overline{B})$

Exemple: Soit la fonction F définie par la table de vérité suivante:

Code Décimal	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Exprimer la fonction F sous forme deux formes canoniques :

1er Forme :
La fonction F vaux 1 pour :

Combinaisons	Monômes correspondants
011	$\overline{A}BC$
101	$A\overline{B}C$
110	$AB\overline{C}$
111	ABC

Et par suite : $F=\overline{A}BC+A\overline{B}C+AB\overline{C}+ABC$
Ainsi la fonction F est écrite sous la forme de somme de termes de trois variables.

2eme Forme :
La fonction F vaux 0 pour :

Combinaisons	Monômes correspondants
000	$\overline{A}\,\overline{B}\,\overline{C}$
001	$\overline{A}\,\overline{B}C$
010	$\overline{A}\,B\overline{C}$
100	$A\,\overline{B}\,\overline{C}$

Et par suite :
$\overline{F}=\overline{A}\,\overline{B}\,\overline{C}+\overline{A}\,\overline{B}C+\overline{A}\,B\overline{C}+A\,\overline{B}\,\overline{C}$
$F=(A+B+C).(A+B+\overline{C}).(A+\overline{B}+C).(\overline{A}+B+C)$
Ainsi est écrite sous forme de produit de somme de variables.

II.5. Expression numérique:

On peut représenter une fonction logique par la valeur décimale (numéro de la ligne de la table de vérité) de chaque combinaison des variables pour la quelle la fonction logique vaut 1.

Exemple:

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

Code Décimal	A	B	C	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

On peut écrire : $F=\sum(0,1,3,6,7)$ ou $\overline{F}=\sum(2,4,5)$

III. Matrice de combinaisons (Tableau de KARNAUGH)

Ce tableau est comparable à la table de vérité.

Pour une fonction F de n variables, le TK sera composé de 2n cases contenant les états de la fonction F (0 ou 1) correspondant aux 2^n combinaisons possibles de n variables. Ces variables sont représentés sur deux axes, de plus elles sont disposées de telle façon qu'un seul bit d'entrée change d'état au passage d'une case à une case voisine (adjacente). Les combinaisons seront donc représentées dans le code binaire réfléchi.

Exemple:

TK pour une fonction à 3 variables

Soit une fonction F à 3 variables A, B et C définie par la TV suivante:

Code Décimal	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Construire le tableau de Karnaugh correspondant à cette fonction.

3 variables réparties en deux groupes :

Le tableau de KARNAUGH contient 2^3 cases : { 1 variable sur la verticale ; 2 variables sur l'horizontale

A	BC	00	01	11	10
0					
1					

4 variables 2^4 cases :

{ 2 variables sur la verticale ; 2 variables sur l'horizontale

AB	BC	00	01	11	10
00					
01					
11					
10					

Chapitre III : Simplification des Fonctions Logiques

Introduction :

Puisque la réalisation d'une fonction logique n'est pas unique, il est souvent souhaitable pour des raisons d'optimisation de disposer de sa forme minimale.

La simplification à cette forme minimal peut être faite de deux façons différentes : une première qui est fondée sur l'application des lois et des théorèmes de l'algèbres booléenne et une deuxième qui est une méthode graphique qui suit une démarche systématique.

I. Simplification Algébrique

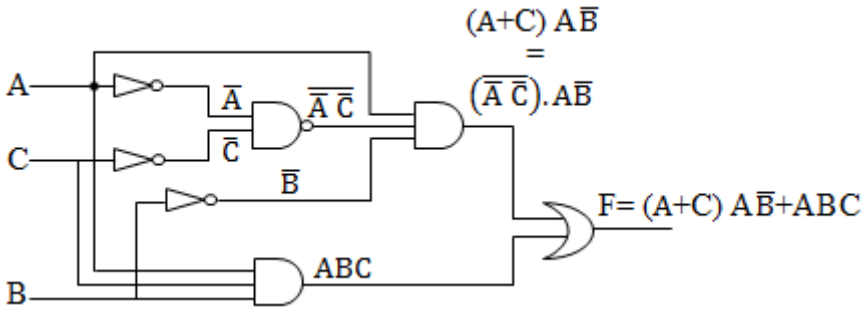
La simplification algébrique est un processus d'approximation successive basée sur deux étapes essentielles :

1- La transformation par applications successives des théorèmes de MORGAN et par multiplication des termes de l'expression pour obtenir une somme de produits.

2- La vérification de chaque produit pour trouver les variables communes afin d'éliminer un ou plusieurs termes.

Exemple

Simplifier par la méthode algébrique le circuit logique suivant :



La première étape consiste à établir l'expression de la sortie F= (A+C) A$\overline{B}$+ABC

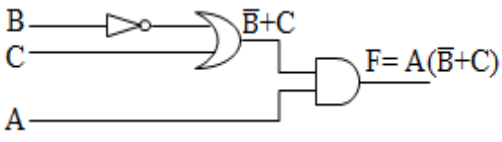
1ère étape de simplification:

Obtenir la somme de produits : F= A$\overline{B}$+ A$\overline{B}$C+ABC

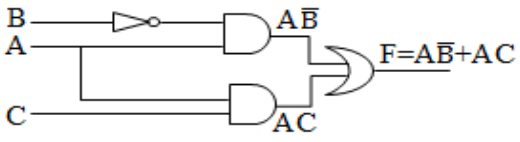
2eme étape de simplification:

F= A$\overline{B}$ + AC (B+$\overline{B}$) = A$\overline{B}$ + AC

Le circuit logique simplifier est donné par:



Ou



II. Simplification par la méthode de KARNAUGH (Graphique)

Dés qu'un T K est rempli de "1" et "0", il s'agit pour obtenir l'expression de la forme F sous forme d'une somme de produit d'additionner logiquement les cases qui contiennent un "1".

Ex.

Soit la fonction F définie par le TK suivant :

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

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

Il est possible de simplifier l'expression de F en combinant selon des règles précises les cases du tableau qui contiennent des 1.

On donne à ce processus de combinaison le nom réunion.

a- Réunion des doublets (paires):

Soient les T K représentés ci-dessous :

A \ BC	00	01	11	10
0	0	1	1	0
1	0	0	0	0

(a)

A \ BC	00	01	11	10
0	0	1	0	0
1	0	1	0	0

(b)

Dans le TK a, il y'a deux 1 qui sont voisins horizontalement, le premier a les coordonnées $\overline{A}\overline{B}C$, tandis que le second à les cordonnées$\overline{A}BC$.

Dans ces deux termes, seule la variable B change d'état, donc ces deux termes peuvent êtres réunis en éliminant la variable B. donc le résultat est $\overline{A}C$.

Le même procédé peut être appliqué au tableau (b) ou les deux 1 sont voisins verticalement.

En regroupant ces deux 1 on élimine la variable 1 car c'est la seule qui change d'état.

Donc on aura en résultat $\overline{B}C$

Exemple

A \ BC	00	01	11	10
0	0	0	0	0
1	1	0	0	1

Il existe deux 1 situés dans la colonne de droite et dans la colonne de gauche (sur la même ligne). Ces deux "1" sont également adjacentes leurs réunion élimine la variable B.

$X = A\overline{C}$

Exemple2

Simplifier la fonction F définie par le TK suivant :

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

→ $\overline{A}\overline{B}C$

→ $A\overline{B}\overline{D}$

Dans ce tableau deux doublet peuvent êtres réunis, celui formé par les "1" adjacents horizontalement et celui formé par les "1" de la colonne de gauche et droite.

Le résultat de ces réunions $F=\overline{A}\overline{B}C + A\overline{B}\overline{D}$

En résumé : la réunion d'un doublet de 1 adjacents dans un TK élimine la variable qui qui est à la fois non complémentée et complémentée.

b- Réunion en groupe de quatre:

Il peut arriver qu'un TK contienne quatre "1" qui soient adjacents

La figure suivante illustre plusieurs exemples de telles réunions.

A \ BC	00	01	11	10
0	1	1	1	1
1	0	0	0	0

→ $\overline{A}$

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

$\overline{C}D$

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

→ BD

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

→ $A\overline{D}$

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

→ $\overline{B}\overline{D}$

Quand en réunis des groupes de quatre on ne retrouve que dans le terme résultant que les variables qui restent sous la même forme dans toutes les cases

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

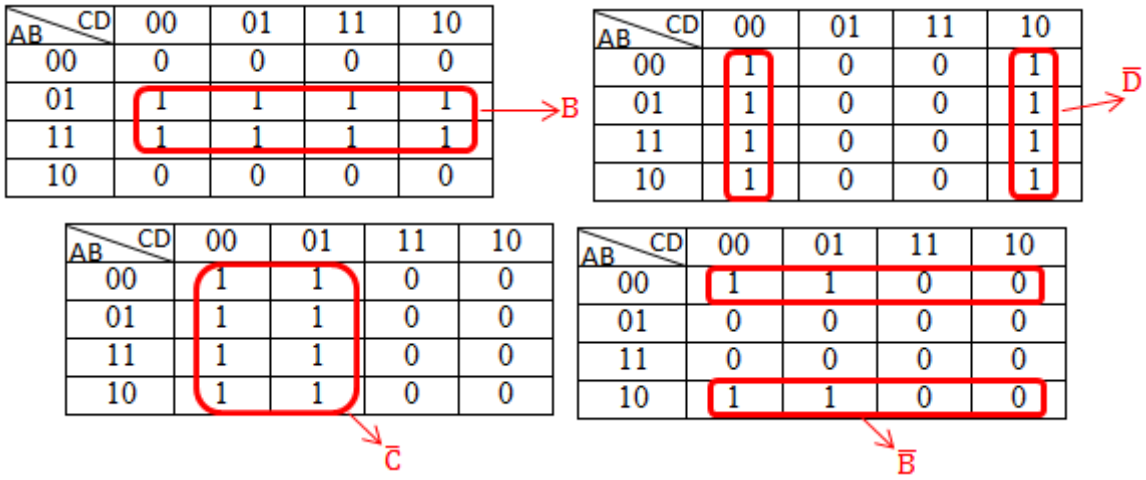
F = $\overline{A}B\overline{C} + \overline{A}CD + ABC + A\overline{C}D$

En Résumé: la réunion des groups de quatre 1 adjacents élimine les deux variables qui sont à la fois non complémentées et complémentées.

c- Réunion en groupe de huit (octets):

La réunion d'un octet dans un TK donne lieu à l'élimination de 3 variables.

Ex.



II.1. Le processus de simplification au complet

Il est claire que plus on à des 1 dans le regroupement, plus le nombre de variables éliminées est grand,

Donc pour simplifier l'expression logique d'un T.K il suffit de trouver le minimum de groupements associant le maximum de "1" adjacents.

Ex: Soit la fonction F définie par le TK suivant:

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

Il existe deux groupements de 4 et un groupement de 2

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

II.2. Les conditions indifférentes

Certains circuits logiques peuvent être conçus pour que certaines conditions d'entrée ne correspondent à aucun niveau de sortie particulier (ni haute ni basse).
Ceci est illustré par l'exemple suivant.

A \ BC	00	01	11	10
0	0	0	1	Ø
1	0	Ø	1	1

La table de vérité est :

A	B	B	F
0	0	0	0
0	0	1	0
0	1	0	Ø
0	1	1	1
1	0	0	0
1	0	1	Ø
1	1	0	1
1	1	1	1

Conditions indifférentes

Dans ces deux tables, aucune valeur de F ne figure pour les conditions :
A,B,C = 0,1,0 et *A,B,C = 1,0,1*
Un concepteur de circuit est libre de mettre des "0" et des "1" vis-à-vis des conditions indifférentes, en vue de produire l'expression de sortie la plus simple

Exemple1
Dans le TK précèdent nous devons décider quel Ø de sortie est remplacé par un 1 et quel Ø est remplacer par un 0 pour donner l'expression la plus simple.

A \ BC	00	01	11	10
0	0	0	1	Ø
1	0	Ø	1	1

$F = B$

Exemple2
Simplifier la fonction F définie par la TV suivante :

A	B	C	D	S
0	0	0	0	1
0	0	0	1	1
0	0	1	0	Ø
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	Ø
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	Ø
1	1	1	1	Ø

AB \ CD	00	01	11	10
00	1	1	1	Ø
01	1	1	0	0
11	0	0	Ø	Ø
10	1	Ø	1	0

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

NB. Cette fonction peut être définie de la manière suivante :
$S = \sum(0,1,3,4,5,8,11) + Ø(2,9,14)$

II.3. Récapitulation

La méthode de simplification avec TK plusieurs avantages comparativement à la méthode algébrique, mais elle reste peut efficace pour les fonctions à grand nombre de variables.
TK d'une fonction à 5 variables *(A,B,C,D,E)*

AB \ CDE	000	001	011	010	110	111	101	100
00								
01								
11								
10								

III. Les Circuits arithmétiques
III.1. Additionneur binaires
III.1.1. Demi-additionneur

Soit deux chiffres binaires a et b à additionner.
Quatre cas se présentent alors :

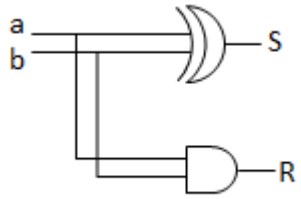
a	b	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Equation logique :

$S = \bar{a}b + a\bar{b} = a \oplus b$

$R = ab$

Circuit logique



Symbole

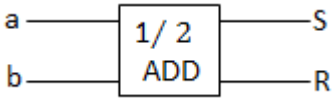


Figure III.1 : Demi-Additionneur

Le circuit qui réalise S et R s'appel demi additionneur. C'est un circuit qui ne permet pas de tenir compte de la retenue précédente.
Pour Y parvenir on utilise un additionneur complet.

III.1.2. Additionneur complet

Soient a et b deux nombre a quatre bits chacun.
La somme S de A et B est obtenue de la manière suivante :

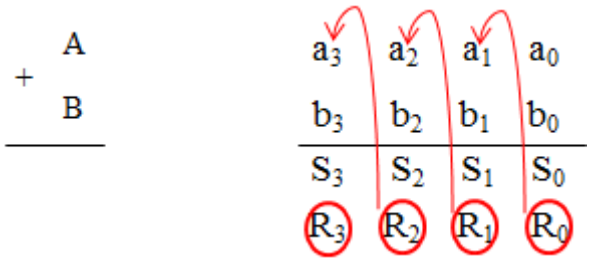


Figure III.2 : Processus d'addition sur 4bits

Nous remarquons qu'a chaque étape de l'addition, nous additionnons 3 bits : a_i , b_i et le bit de retenue provenons du rang précédent r_{i-1}d. Le résultat de l'addition de ces 3 bits est un nombre à deux bits : le bit de la somme S et le bit de retenue R. ce dernier doit être ajouté au rang a gauche, d'où la TV suivante:

a_i	b_i	r_{i-1}	S_i	R_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Symbole

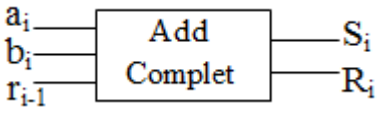


Figure III.3 : Additionneur Complet

Simplification par le T.K :

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

$$S_i = \bar{a_i}\bar{b_i}r_{i-1} + \bar{a_i}b_i\overline{r_{i-1}} + a_i\bar{b_i}\overline{r_{i-1}} + a_ib_ir_{i-1}$$

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

$$r_i = a_ir_{i-1} + a_ib_i + b_ir_{i-1}$$

L'expression pour S_i devient :

$$S_i = \bar{a_i}(\bar{b_i}r_{i-1} + b_i\overline{r_{i-1}}) + a_i(\bar{b_i}\overline{r_{i-1}} + b_ir_{i-1})$$
$$= \bar{a_i}(b_i \oplus r_{i-1}) + a_i(\overline{b_i \oplus r_{i-1}})$$
$$\boxed{S_i = a_i \oplus (b_i \oplus r_{i-1})}$$

Les expressions de Si et Ri peuvent être matérialisées par le circuit logique suivant :

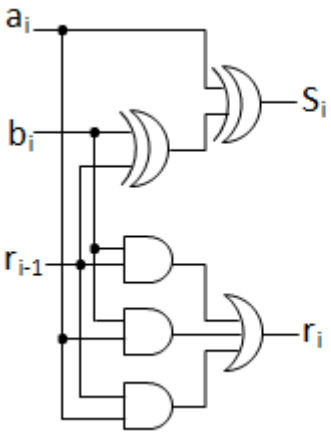


Figure III.4 : Circuit d'un Additionneur Complet

III.1.3. Additionneur complet à l'aide de deux demi-additionneurs

Reprenons les deux expressions logiques de S_i et r_i :

$S_i = a_i \oplus (b_i \oplus r_{i-1})$

$r_i = a_i r_{i-1} + a_i b_i + b_i r_{i-1}$

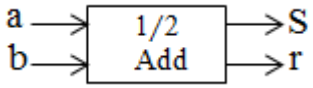
Ou

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

$$r_i = b_i r_{i-1} + a_i \overline{b_i} r_{i-1} + a_i b_i \overline{r_{i-1}}$$

$$r_i = b_i r_{i-1} + a_i (b_i \oplus r_{i-1})$$

Les équations logiques d'un demi-additionneur sont les suivantes :



$S = a \oplus b$

$R = ab$

Pour concevoir un additionneur complet a l'aide d'un demi-additionneur :

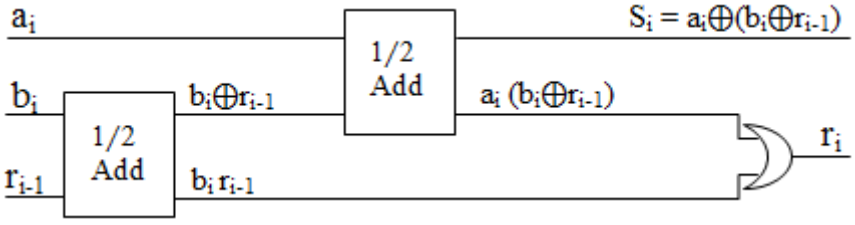


Figure III.5 : Additionneur complet à l'aide des Demi-Additionneurs

III.1.4. Conception d'un additionneur binaire parallèle

On va concevoir un circuit logique qui effectue la somme de deux nombres binaires A et B à n bits chacun. L'opération consiste à accoler des additionneurs complets identiques pour chaque rang binaire.

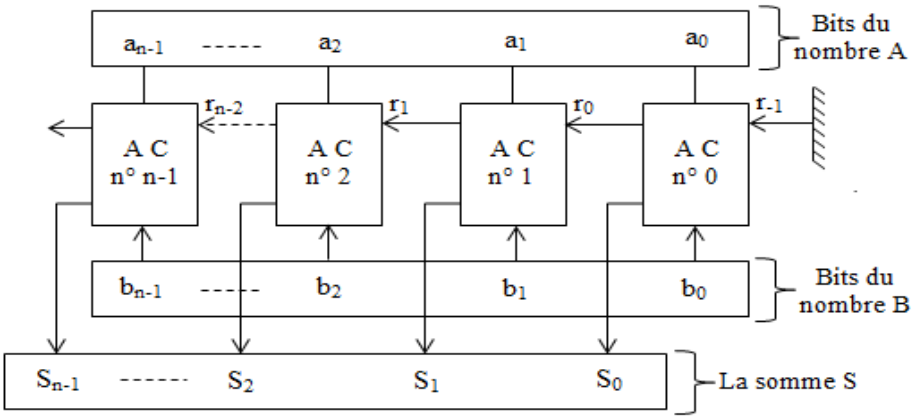


Figure III.6 : Additionneur Binaire Parallèle

Les bits correspondant des nombres a_i et b_i et le bit de retenue généré par l'addition précédente sont appliqués a des additionneurs complets.
Par exemple, les bits a_1 et b_1 sont transmis a l'AC1 en meme temps que r_0 qui est le bit de retenue produit par l'addition de a_0 et b_0.
Ce montage est appelé un additionneur parallele car tout les bits sont appliqués et et additionnés simultanément.

III.1.5. Additionneur parallèle intégré

L'un des plus courants boitiers (CI) d'additionneurs parallèles de 4 bits sont 7483A, 74LS83A, 74283 et 74LS283…
Le symbole logique du 74LS83 est le suivant :

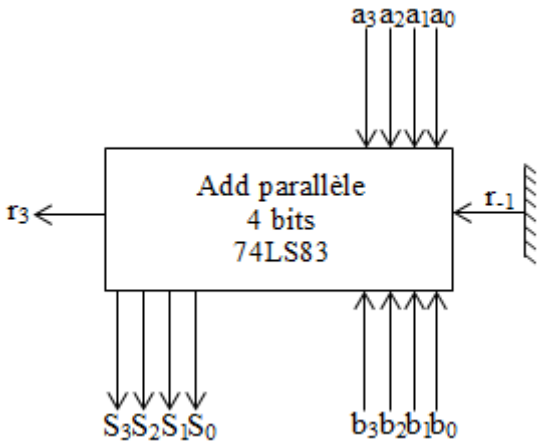


Figure III.7 : Additionneur parallèle intégré

III.1.6. Montage en cascade d'additionneurs paralleles

Il est possible de raccorder deux ou plusieurs add paralleles en cascade afin d'additionner des nombres ayant un plus grand nombre de bits

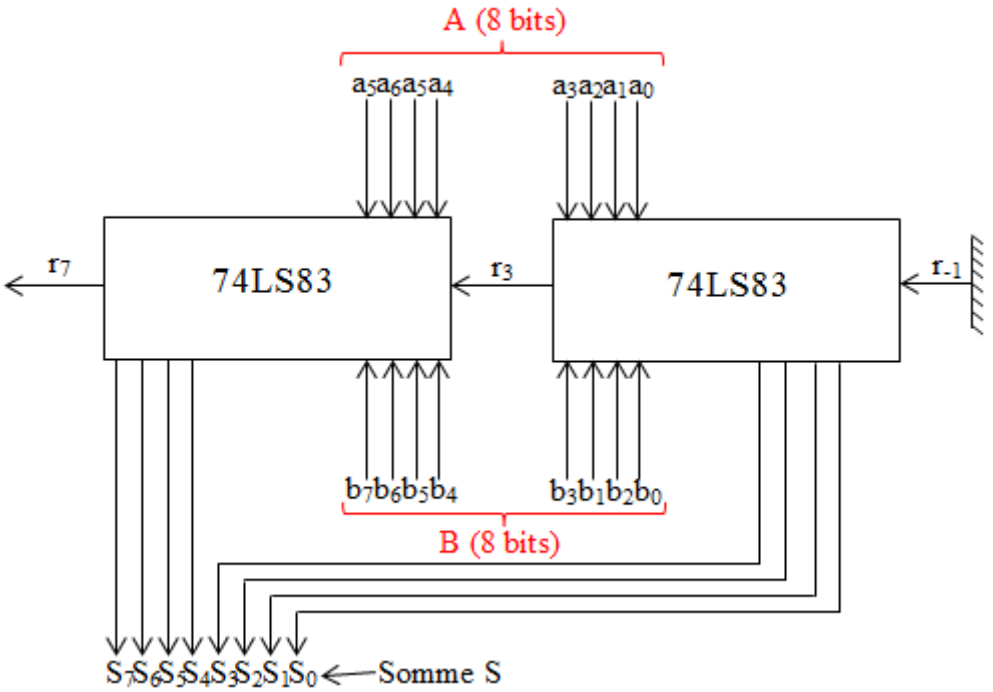


Figure III.8 : Deux additionneurs paralleles montés en cascade

Exemple

Soient les deux nombres A et B donnés par :

Il n'y a donc pas de déppasement sur le bit 7 (r_7=0)

$$A = 10001001$$
$$B = 01001000$$
$$S = 11010001$$

III.2. Soustractions binaires

III.2.1. Demi-Soustracteur

La différence D et la retenue C de deux nombres A et B sont présentées par la TV suivante :

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

Equations logiques:

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

Circuit logique :

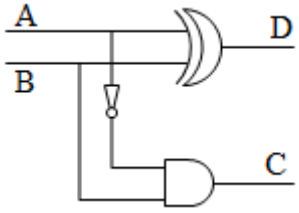


Figure III.9 : Demi-soustracteur

III.2.2. Soustracteur complet:

A l'aide de deux demi soustracteurs on peut réalisé un soustracteur complet présenté par la TV suivante :

a_i	b_i	C_{i-1}	D_i	C_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

a_i \ b_ic_{i-1}	00	01	11	10
0	1	0	1	0
1	0	1	0	1

$$D_i = (a_i \oplus b_i) \oplus c_{i-1}$$

a_i \ b_ic_{i-1}	00	01	11	10
0	0	1	1	1
1	0	1	1	0

$$C_i = \overline{a_i} b_i + (c_{i-1}(\overline{a_i \oplus b_i}))$$

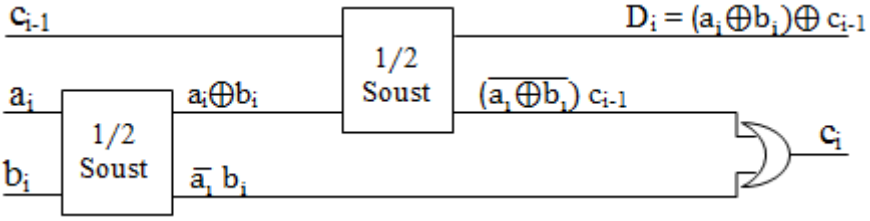


Figure III.10 : Soustracteur Complet à l'aide des Demi-Soustracteurs

III.2.3. Notation complément à deux

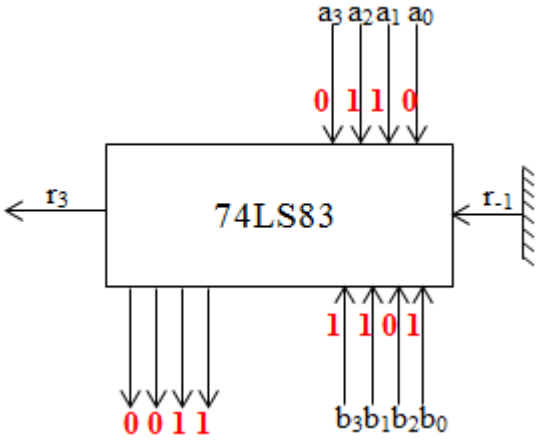
L'addition et la soustraction des nombres signés se résument à une simple addition si on exprime ces nombres selons la notation en complément à 2.

Complément a 2 de B est $\overline{B} + 1$

Donc A-B=A+ $\overline{B} + 1$

Exemple

A	+6	6	6
-	-	-	+
B	+3	3	(-3)



L'additionneur parallele produit alors la somme 0011 en sortie, et qui correspond a +3.

La retenue r_3 est à 1, mais cette retenue est rejetée dans la méthode complément à deux.

La figure suivante illustre un additionneur peut servir à la soustraction :

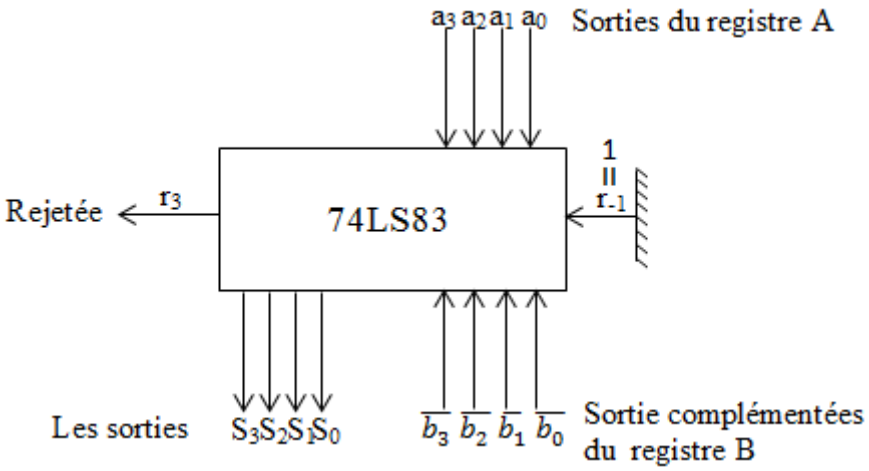


Figure III.11 : Additionneur parallele seravnt à soustraire

On utilise les sorties complémentées du registre B pour complémenté B à 1, et en plus en transforme r_{-1} à un 1 qui va etre ajouté au poid le plus faible de l'additionneur a fin d'obtenir le complément à 2 de B.
Donc S est le résultat de la soustraction et r_3 est le signe de ce résultat.

III.3. Additionneur / Soustracteeur

Le circuit suivant est à la fois add et soust complet dans la notation complément à 2 .

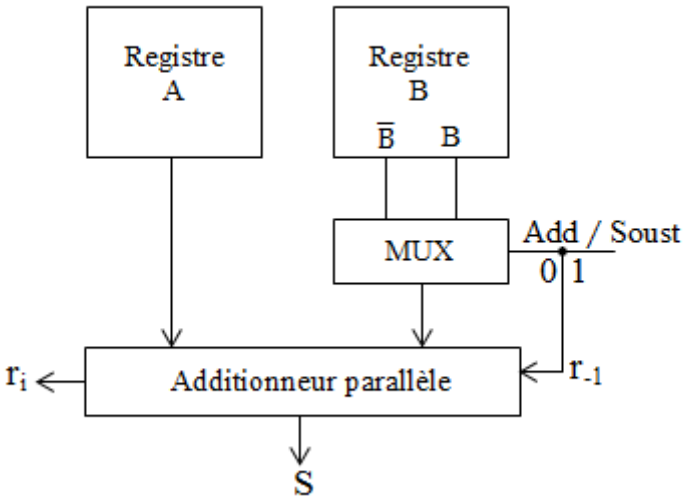


Figure III.12 : Additionneur / Soustracteur

Cet Additinneur / Soustracteur est commandé par le signal de commande Add / Soust.
Quand ce signal est au niveau BAS, le circuit fonctionne comme un additionneur
Quand ce signal est au niveau HAUT, le circuit fonctionne comme un soustracteur.

III.4. Additionneur BCD :

Rappelons que ce code fait correspondre a chaque nombre décimal un code binaire de quatre bites compris entre 0000 et 1001
L'addition avec le code BCD présente deux cas :

a- Somme infèrieure ou égale à 9 :

Exemple1
Additionnons 5 et 4 en utilisant leurs représentation BCD :

```
  5      0101 <-- BCD de 5
+      +
  4      0100 <-- BCD de 4
  9      1001 <-- BCD de 9
```

L'addition est éffectuée comme une addition binaire normale

Exemple2

```
  45     0100 0101   <--   BCD de 45
+      +
  33     0011 0011   <--   BCD de 33
  9      0111 1000   <--   BCD de 78
```

Dans ce cas l'addition est un processus direct équivalent à l'addition binaire.

b- Somme superieure à 9 :
Exemple
Additionnons 6 et 7 en BCD :

```
  6      0110 <-- BCD de 6
+      +
  7      0111 <-- BCD de 7
  13     1101 <-- Code invalide en BCD
```

1101 n'existe pas en code BCD, dans un tel cas il faut corriger la somme en ajoutant 6(0110) afin de sauter 6 valeurs qui présentent un code non valide.

```
  6      0110 <-- BCD de 6
+      +
  7      0111 <-- BCD de 7
  13     1101
       +
         0110
  (0001) (0011) <-- BCD de 13
    1      3
```

Chapitre IV: Analyse et synthèse des circuits logiques combinatoires

---

I. les circuits combinatoires

Les circuits combinatoires sont des circuits avec lesquels on peut établir plusieurs opérations comme:

a- Codage et décodage: C'est la transposition des données d'un code à un autre.

b- Le multiplexage: Choix d'un groupe de données parmi plusieurs.

c- Le démultiplexage: Aiguillage des données vus une destination parmi plusieurs.

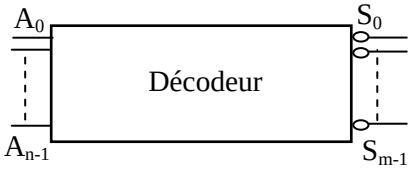
d-Acheminement par Bus: Transmission de données entre plusieurs dispositifs par d'intermédiaire d'un bus commun.

II. Le décodeur

Le décodeur est un circuit logique qui établit la correspondance entre un code d'entrée binaire de n bits et m lignes de sortie, pour chacune des combinaisons possibles des entrées, une seule ligne de sortie est validée.

Donc pour chacune des 2^n combinaisons possibles, une seule est à l'état haut, et toutes les autres deumeurent au niveau bas.

Rq: De nombreux décodeurs sont conçus pour avoir des sorties vraies au niveau bas (sortie active est au niveau bas tandis que toutes les autres sont au niveau haut). Donc le docodeur va présentés des inversions en sortie.



II.1. Décodeur 3 entrées 8 sorties (1 parmi 8)

C'est un circuit a trois vois d'entrée et 8 vois de sortie. Sa TV est la suivante :

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{A}\bar{B}\bar{C}$
$D_1 = \bar{A}\bar{B}C$
$D_2 = \bar{A}B\bar{C}$
$D_3 = \bar{A}BC$
$D_4 = A\bar{B}\bar{C}$
$D_5 = A\bar{B}C$
$D_6 = AB\bar{C}$
$D_7 = ABC$

Le circuit logique de ce décoeur est le suivant :

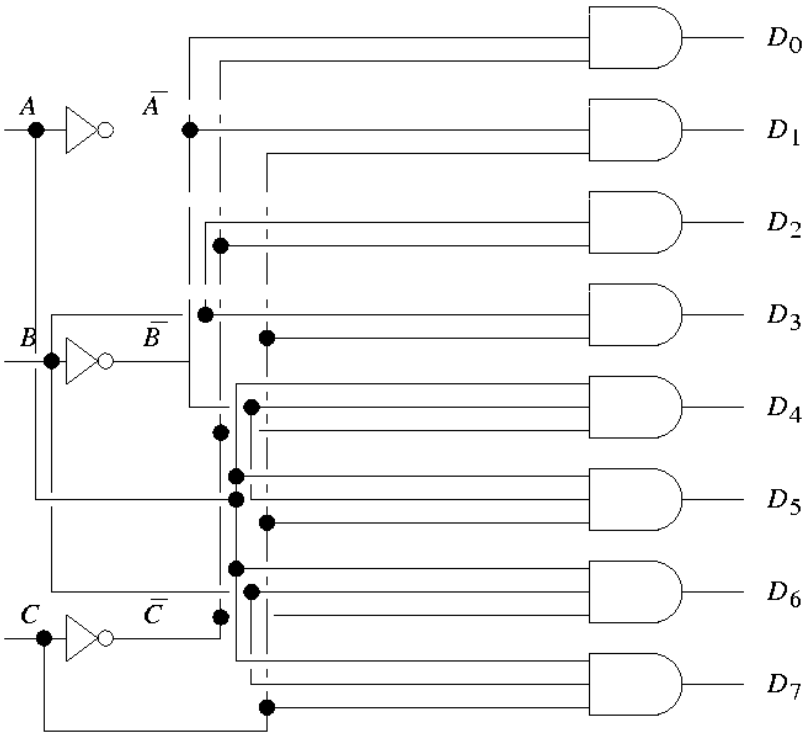


Figure IV.1 : Décodeur 3 Entrées / 8 Sorties

On peut ajouter au décodeur si-dessus une ligne commune qu'on appelle validaion racordé à la quatrième entrée de chaque porte AND. Quand cette ligne est au niveau hau; le décodeur fonctionne normalement, et si elle est gardée au niveau bas, toute les sortie sont forcé au niveau bas quelque soit les niveau appliqués au entrées A,B,C donc ce décodeur est validé seulement si le signal validation est au niveau haut.

II.2. Décodeur BCD décimal

A	B	C	D	Sortie active
0	0	0	0	$\overline{D_0}$
0	0	0	1	$\overline{D_1}$
0	0	1	0	$\overline{D_2}$
0	0	1	1	$\overline{D_3}$
0	1	0	0	$\overline{D_4}$
0	1	0	1	$\overline{D_5}$
0	1	1	0	$\overline{D_6}$
0	1	1	1	$\overline{D_7}$
1	0	0	0	$\overline{D_8}$
1	0	0	1	$\overline{D_9}$

$$D_0 = \bar{A}\bar{B}\bar{C}\bar{D}$$
$$D_1 = \bar{A}\bar{B}\bar{C}D$$
$$D_2 = \bar{A}\bar{B}C\bar{D}$$
$$D_3 = \bar{A}\bar{B}CD$$
$$D_4 = \bar{A}B\bar{C}\bar{D}$$
$$D_5 = \bar{A}B\bar{C}D$$
$$D_6 = \bar{A}BC\bar{D}$$
$$D_7 = \bar{A}BCD$$
$$D_8 = A\bar{B}\bar{C}\bar{D}$$
$$D_9 = \bar{A}BC\bar{D}$$

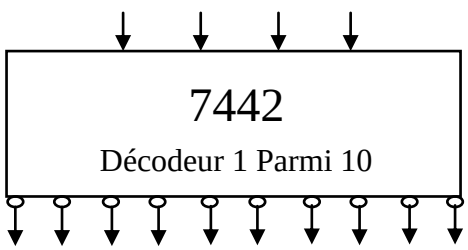


Figure IV.2. Décodeur BCD décimal valide à état bas

Le Décodeur BCD décimal présente en entrée un code binaire a qautre bits et dix lignes de sortie (de 0000 à 1001). Les codes à qautre bits inutilisés (1010 à 1111) n'active aucune des sorties lorqu'ils sont appliqués.
La figure suivante présente le shéma logique d'un décodeur BCD décimal 74XX42

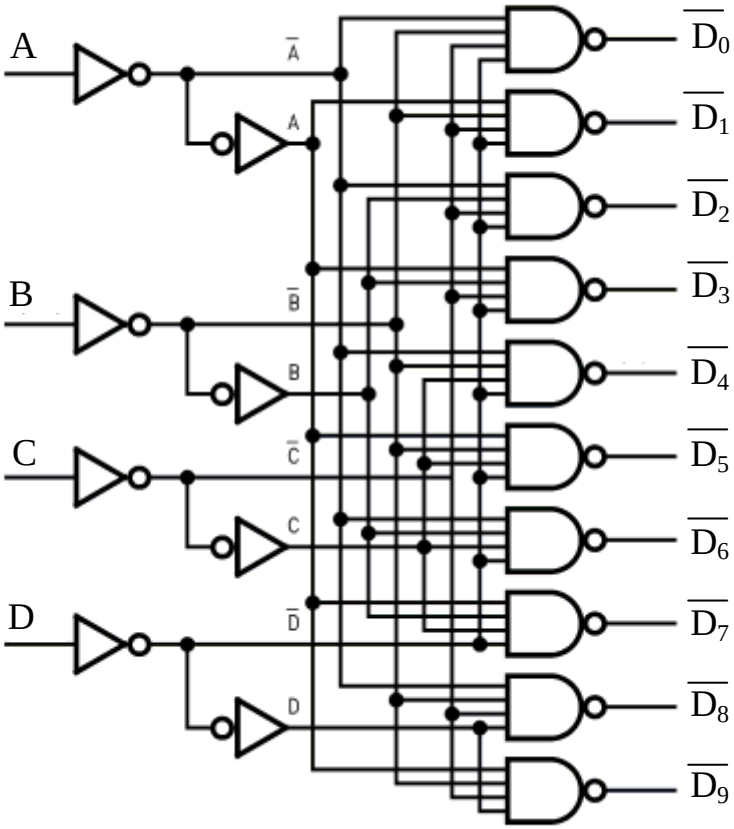


Figure IV.3 : Logigramme d'un décodeur BCD décimal 74XX42

Une sortie ne passe à zéro qu'au moment ou son entrée BCD est appliquée
Exemple : DCBA = 0101 → La Sortie $\overline{S5}$ est au niveau 0

II.3. Décodeur BCD – 7 Segments

Dans de nombreux affichages numériques les dix chiffres 0 à 9 et parfois les caractères héxadécimaux de A à F sont configurés au moyen de 7 Segments

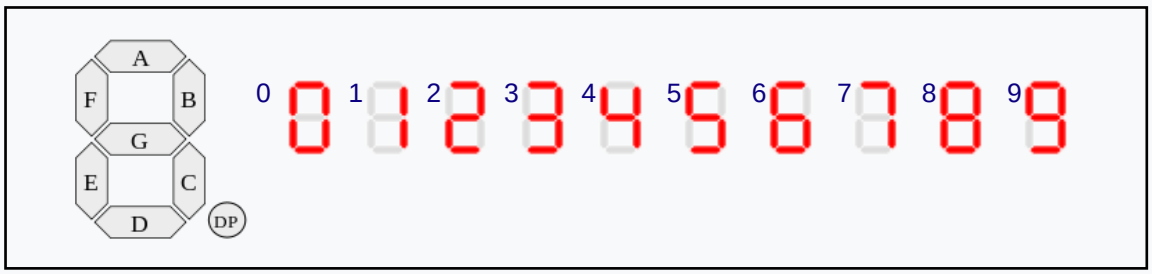


Figure IV.4 : Affichage numériques de 0 à 9 au moyen de 7 Segments

Chaque segment est constitué d'un matériau qui émet de la lumière quand il est traversé par un courant. Les matériaux les plus utilisés sont les diodes électroluminescentes (LED)

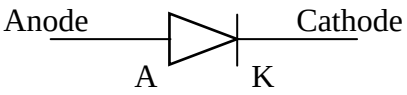


Figure IV.5 : LED : diode électroluminescente

Décimal	A	B	C	D	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0

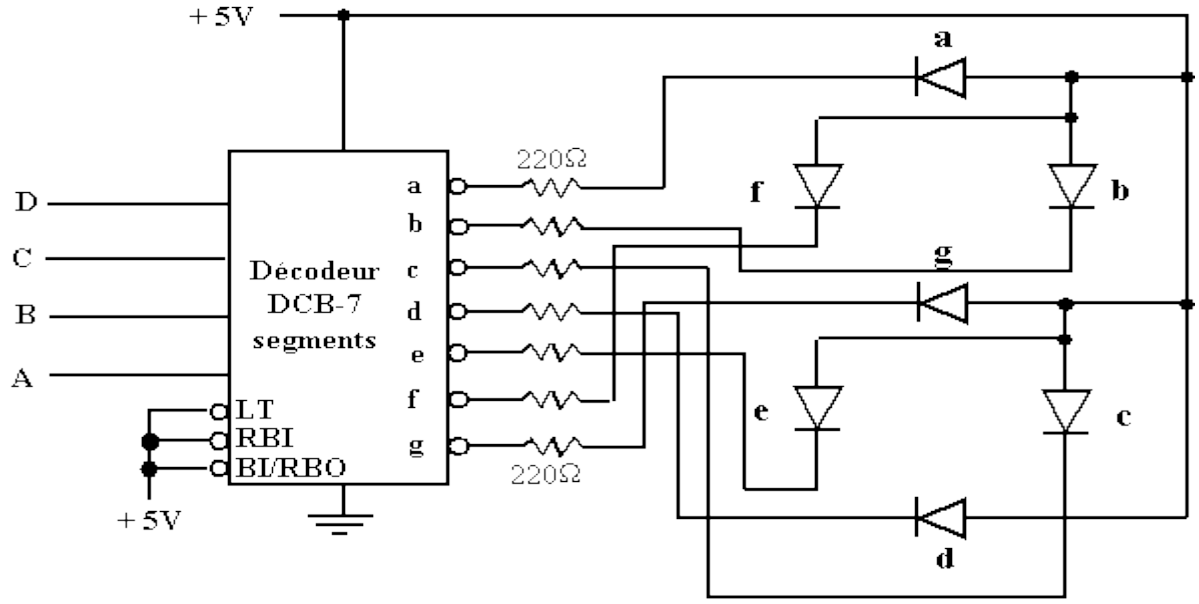


Figure IV.6 : Afficheur 7 segments à anode commune 7447

Par exemple pour afficher 1 il faut que les segments a et b sont allumés et les autres sont éteins.

Un décodeur 7 segements accepte en entrée les les 7 codes BCD et rend avtive les sorties qui vont permettre de faire passer un courant dans les segments qui forment le chiffre décimal correspondant

Le 7447 est un décodeur 7 segments dont chaque segment est constitué des diodes LED. Les anodes dans ces diodes sont toutes réunies à Vcc (+5V) et leurs cathodes sont connectées a travers des resistances limitatrices de courant aux sorties appropriées du décodeurs.

Ce dernier présente des sorties actives a niveau bas, sa table de vérité et sont circuit sont présentés en *Fig.V.6.*
Exemple : Soit l'entrée DCBA = 0101, qui correspond à 5V, En réponse a cette entrée les sorties
$\bar{a}\ \bar{f}\ \bar{g}\ \bar{c}\ \bar{d}$ sont amenées au niveau bas (racordés à la masse)

Il existe un autre type d'afficheur à 7 segments dit a cathode commune dans lequel toute les cathodes sont réunies à la masse.
Dans cet afficheur les sorties sont vraies au niveau haut (Ex 7448)

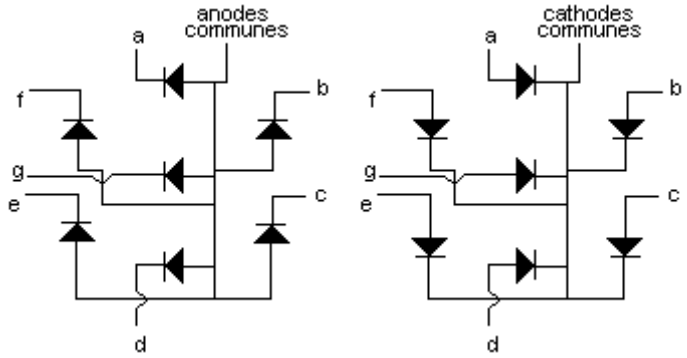
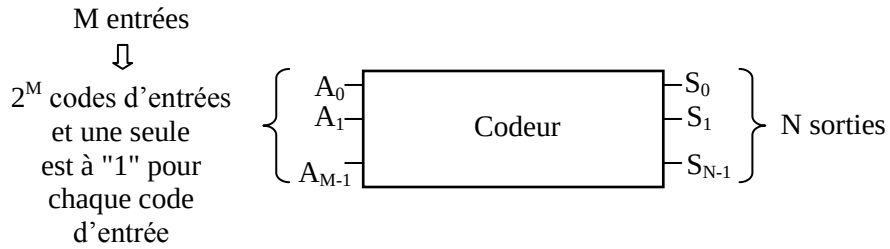


Figure IV.7 : Comparaison entre l'Afficheur7 segments à anode commune 7447 et à cathode commune 7448

III. Les Codeurs

Le processus inverse du décodage est le codage présente un circuit logique avec un certain nombre d'entrée (M) dont une seule est active et qui correspond à une représentation de sortie de N bits



III.1. Codeur Octal-Binaire (8/3)

Un codeur Octal binaire a 8 vois d'entrée et produit une représentation binaire de 3 bits
Exemple

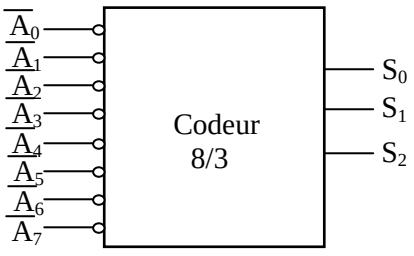


Figure IV.8 : Codeur 8entrées/3Sorties active à état bas

$\overline{A_0}$	$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	S_2	S_1	S_0
Ø	1	1	1	1	1	1	1	0	0	0
Ø	0	1	1	1	1	1	1	0	0	1
Ø	1	0	1	1	1	1	1	0	1	0
Ø	1	1	0	1	1	1	1	0	1	1
Ø	1	1	1	0	1	1	1	1	0	0
Ø	1	1	1	1	0	1	1	1	0	1
Ø	1	1	1	1	1	0	1	1	1	0
Ø	1	1	1	1	1	1	0	1	1	1

Par exemple au niveau bas sur $\overline{A_3}$ toutes les autres entrées sont à 1 et on a la sortie (011) qui représente le code binaire de 3

$$\begin{cases} S_0 = \overline{\overline{A_1}} + \overline{\overline{A_3}} + \overline{\overline{A_5}} + \overline{\overline{A_7}} \\ S_1 = \overline{\overline{A_2}} + \overline{\overline{A_3}} + \overline{\overline{A_6}} + \overline{\overline{A_7}} \\ S_2 = \overline{\overline{A_4}} + \overline{\overline{A_5}} + \overline{\overline{A_6}} + \overline{\overline{A_7}} \end{cases}$$

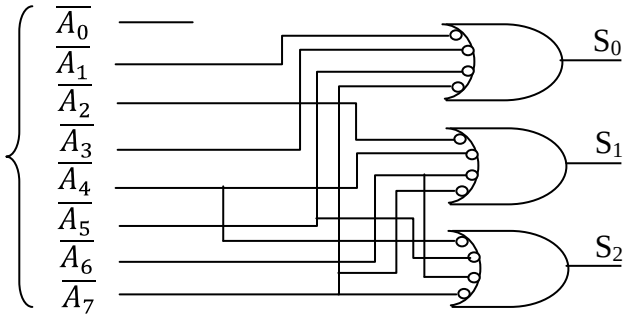


Figure IV.9 : Codeur Octal/Binaire 8/3

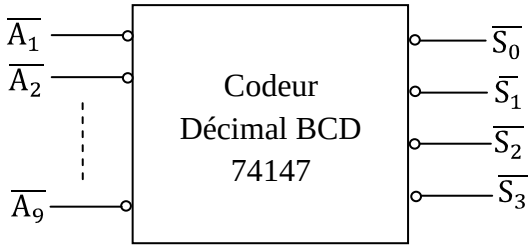
NB : $\overline{A_0}$ n'est pas connéctée à une porte logique puisque les sorties du codeur sont normalement à (000) quand aucune des entrées de $\overline{A_1}$ a $\overline{A_7}$ n'est au niveau bas

III.2. Codeur de priorité

Un codeur de priorité présente une sortie active lorsque deux entrées sont validées, dans un tel codeur quand par exemple les deux entrées $\overline{A_5}$ et $\overline{A_3}$ sont activées au même temps ; la répense donnée en sortie est 101 donc la sortie correespond au nombre le plus élevés.

III.2.1 Codeur de priorité Decimal – BCD

Le symbole logique du codeur de priorité Décimal – BCD (74147) est présenté comme suivant



Si $\overline{A_1} = 0$ et On a en sortie

$\overline{S_0} = 0$
$\overline{S_1} = 1$
$\overline{S_2} = 1$
$\overline{S_3} = 1$

Figure IV.10 : Codeur de priorité Décimal - BCD

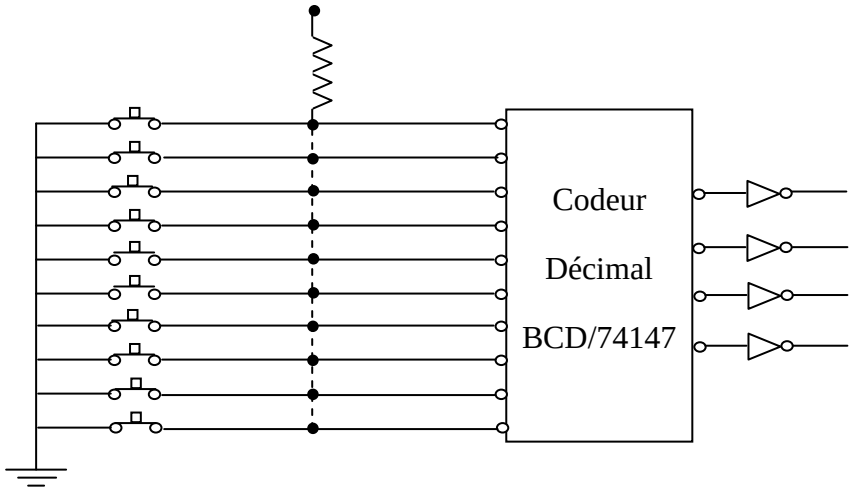
Ce circuit possède 9 entrées vraies au niveau bas représentant les 9 chiffres décimaux (1 à 9) et produit la représentation BCD complémentée correspondante à l'entrée la plus haute mise au niveau vrai

$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$\overline{A_8}$	$\overline{A_9}$	$\overline{S_3}$	$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$
1	1	1	1	1	1	1	1	1	1	1	1	1
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	0	1	1	1	0
Ø	Ø	Ø	Ø	Ø	Ø	Ø	0	1	1	1	0	1
Ø	Ø	Ø	Ø	Ø	Ø	0	1	1	1	1	0	0
Ø	Ø	Ø	Ø	Ø	0	1	1	1	1	0	1	1
Ø	Ø	Ø	Ø	0	1	1	1	1	1	0	1	0
Ø	Ø	Ø	0	1	1	1	1	1	1	0	0	1
Ø	Ø	0	1	1	1	1	1	1	1	0	0	0
Ø	0	1	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	0	1	1	0

Les sorties du 74147 sont normalement à 1 quand aucune des entrées n'est a son niveau vrai cesi correspond à la condition d'entrée du chiffre décimal 0, pour obtenir le code BCD naturel à partir des sorties du 74147 il faut ajouter un inverseur à chacune des sorties

III.2.2 Codeur d'interrupteur

Dans la figure suivante on montre comment on peut réaliser avec le circuit intégré 74147 un codeur d'interrupteur



Les interrupteurs sont du type ouvert au repos de sorte que les entrées du codeur sont généralement toute à 1 quand c'est le cas la sortie BCD est (0000). Quand une touche est enfoncée (interrupteur) le circuit donne en sortie le code BCD correspondant a ce chiffre.
Le codeur interupteur peut servir toutes les fois que l'on doit introduire manuellement des données BCD dans un système numérique. Un exemple parfait de cela est la calculatrice éléctronique dans laquelle un opérateur introduit un nombre décimal en appuyant succéssivement sur les interrupteurs du clavier

IV. Les transcodeurs

Un transcodeur (convertisseur de codes) est un dispositif permettant de passer du nombre N écrit dans le code C1 au même nombre N écrit dans le code C2. L'utilisation des transcodeurs est relativement limitée, ce qui explique qu'on ne les trouve pas tous sous forme de circuits intégrés : il faut alors les réaliser à l'aide de portes logiques ET-NON, OU NON ... etc.

La réalisation pratique d'un transcodeur passe par sa table de vérité, puis par l'écriture et la simplification des équations de sorties avec les tableaux de Karnaugh.

Parmi les transcodeurs que l'on trouve en circuits intégrés, on peut citer :

- les transcodeurs décimal / BCD (circuit 74147).
- les transcodeurs BCD / décimal (circuits 7442, 7445, et 4028).
- les transcodeurs XS 3 / décimal (circuit 7443).
- les transcodeurs Gray excédant 3 (code Gray+3) / décimal (circuit 7444).
- les transcodeurs DCB / afficheur 7 segments (circuits 7448, 7511, 4543, 4511).
- les transcodeurs binaire 5 bits / DCB (circuit 74185) et DCB / binaire 5 bits (circuit 74184).

IV. 1. Transcodeur 10 vers 4(10 entrées - 4 sorties)

C'est un codeur qui reçoit en entrée un chiffre décimal sur une des dix entrées et génère en sortie l'équivalent binaire sur les Sorties S0 à S3. Une seule entrée doit être active à la fois.

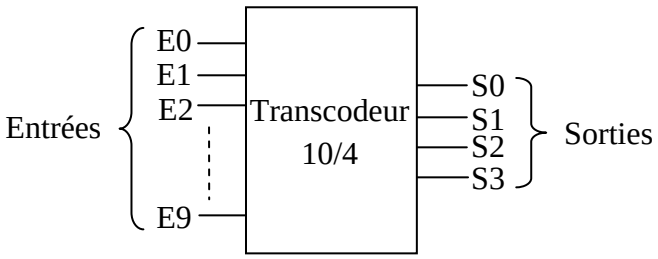


Table de Vérité :

Entrée Active (=1)	Sorties			
	S0	S1	S2	S3
E0	0	0	0	0
E1	0	0	0	1
E2	0	0	1	0
E3	0	0	1	1
E4	0	1	0	0
E5	0	1	0	1
E6	0	1	1	0
E7	0	1	1	1
E8	1	0	0	0
E9	1	0	0	1

Equations logiques :

S0 = E1 + E3 + E5 + E7 + E9

S1 = E2 + E3 + E6 + E7

S2 = E4 + E5 + E6 + E7

S3 = E8 + E9

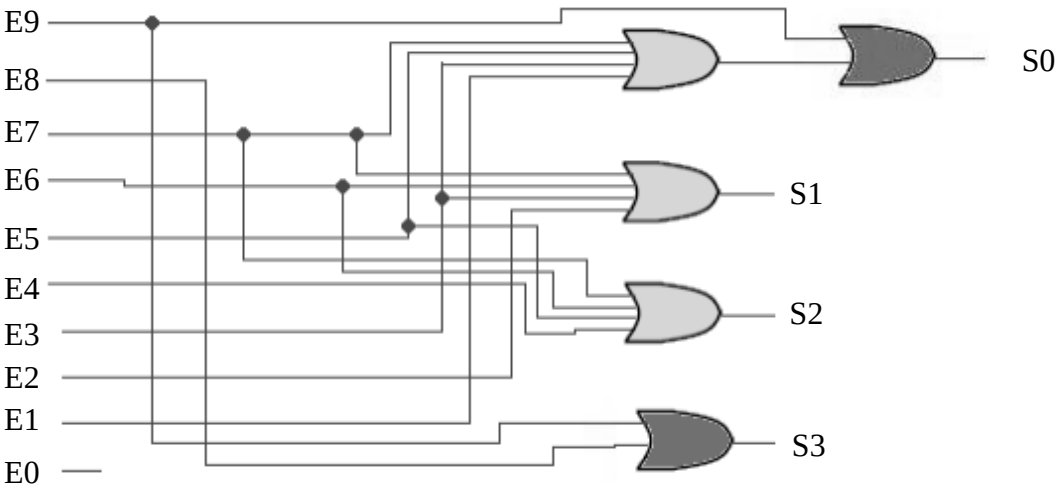
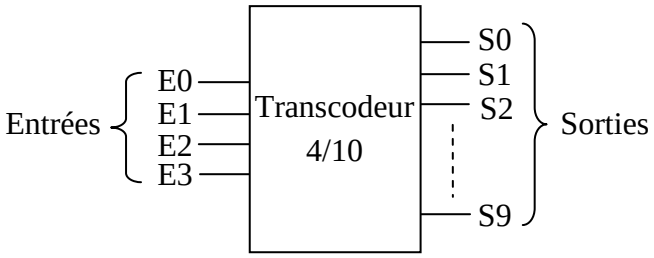


Figure IV.11 : Transcodeur Comparaison 10 vers 4

Transcodeur 4 vers 10 (4 entrées vers 10 sorties)

C'est un décodeur qui reçoit en entrée un chiffre binaire sur autre bits (Quatre entrées) et génère en sortie une seul sortie activée indiquant son équivalence décimale



E0	E1	E3	E4	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Les équations :

$S0 = \overline{E0}\ \overline{E1}\ \overline{E2}\ \overline{E3}$ $S3 = \overline{E0}\ \overline{E1}\ E2\ E3$ $S6 = \overline{E0}\ E1\ E2\ \overline{E3}$ $S9 = E0\ \overline{E1}\ \overline{E2}\ E3$

$S1 = \overline{E0}\ \overline{E1}\ \overline{E2}\ E3$ $S4 = \overline{E0}\ E1\ \overline{E2}\ \overline{E3}$ $S7 = \overline{E0}\ E1\ E2\ E3$

$S2 = \overline{E0}\ \overline{E1}\ E2\ \overline{E3}$ $S5 = \overline{E0}\ E1\ \overline{E2}\ E3$ $S8 = E0\ \overline{E1}\ \overline{E2}\ \overline{E3}$

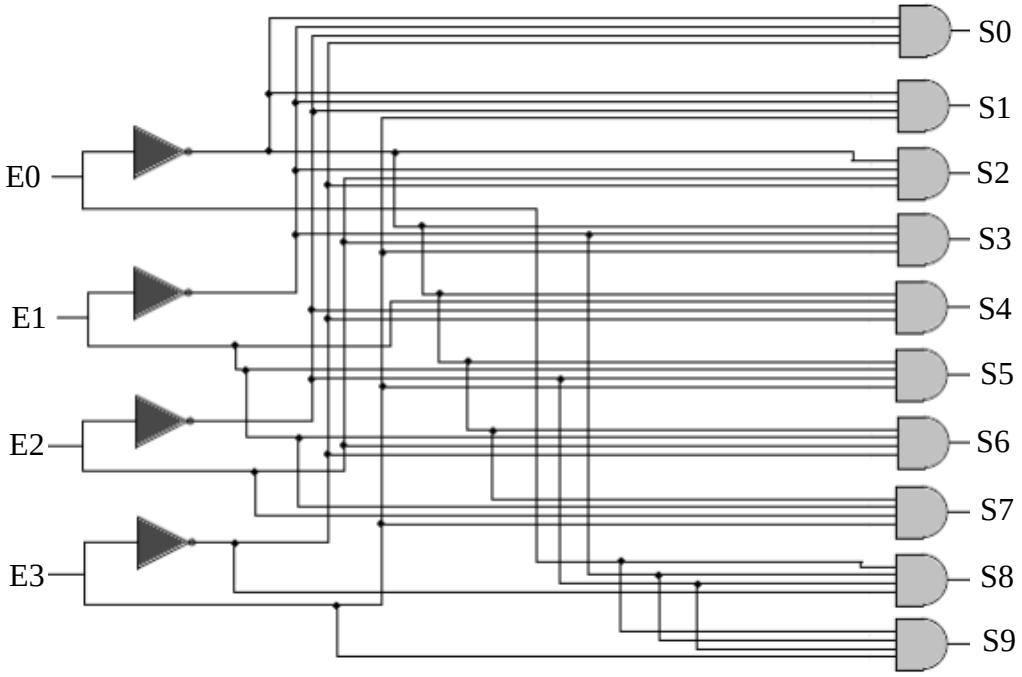
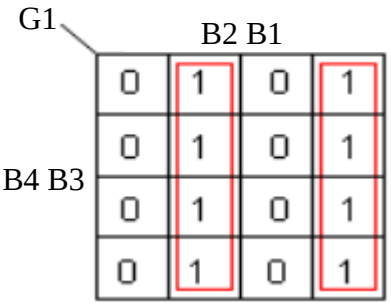


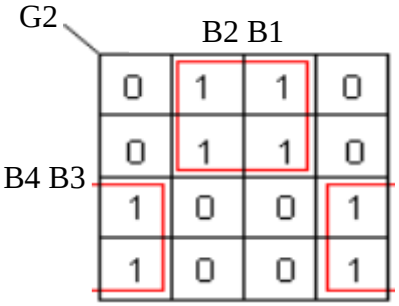
Figure IV.12: Transcodeur Comparaison 4 vers 10

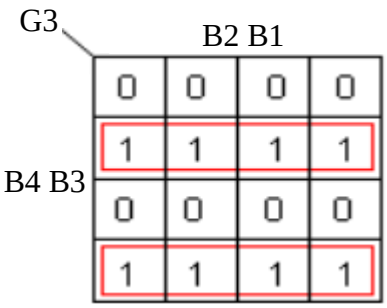
Transcodage Binaire → Gray (4 entrées vers 4 sorties)

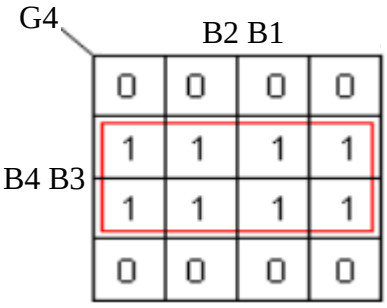
L'équivalence en code Gray pour les nombres décimaux de 0 à 15 est représentée par la table suivante :

Nombre Décimal	Code binaire Naturel				Code Gray			
	B4	B3	B2	B1	G4	G3	G2	G1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0









Donc les équations logiques sont :

$$G1 = B1\,\overline{B2} + B2\,\overline{B1} = B1 \oplus B2$$
$$G2 = B2\,\overline{B3} + B3\,\overline{B2} = B2 \oplus B3$$
$$G3 = B3\,\overline{B4} + B3\,\overline{B4} = B3 \oplus B4$$
$$G4 = B4$$

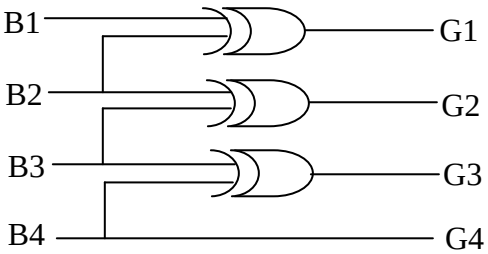


Figure IV.13 : Transcodeur Binaire Gray

Transcodeur Gray → Binaire (4 entrées vers 4 sorties)

Pour obtenir le circuit d'un transcodeur du code gray vers le code binaire, il suffit d'inversé la table de vérité du transcodeur binaire vers gray, donc les entrées vont êtres G1, G2, G3 et G4 et les sorties sont B1, B2, B3 et B4.

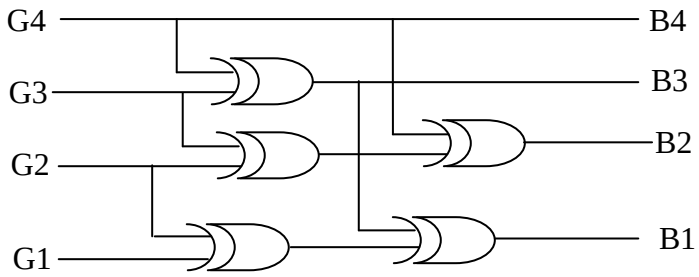
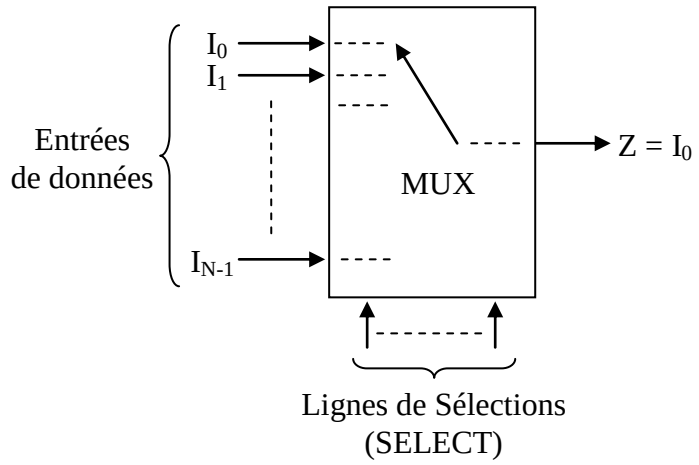


Figure IV.14 : Transcodeur Gray Binaire

V. Les Multiplexeurs (Sélecteurs de données)

- Un multiplexeur ou un sélecteur de données est un circuit logique ayant plusieurs entrées de données mais seulement une sortie qui communique ces données, sont symboles est MUX
- L'aiguillage (l'orientation de l'entrée) de données qui nous intéresse sur la sortie est commandé par les entrées (SELECT) appelées parfois (entrées d'adresse)
- La figure suivante illustre le symbole des multiplexeurs.



Le code d'entrée SELECT détermine quelle entrée est transmise jusqu'à la sortie

Figure IV.15 : Multiplexeur à N entrées – 1Srtie

Un multiplexeur se comporte comme un commutateur dans lequel un code numérique appliqué aux entrées SELECT commande les entrées des données qui sont raccordées à la sortie, il peut y'avoir sur la sortie Z les données introduites sur I0 quand on applique un certain code d'entrées SELECT ou bien Z peut avoir les données de I1 en réponse à un autre code d'entrées SELECT

Autrement dit un multiplexeur choisi une source de données parmi N et transmet celle-ci à la seule voix de sortie Z (c'est ce qu'on appelle le Multiplexage)

V.1. Multiplexeurs élémentaires à deux entrées

E	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A (lignes 0 0 1 1) ; B (lignes 0 1 0 1)

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

$$Z = A\overline{E} + EB$$

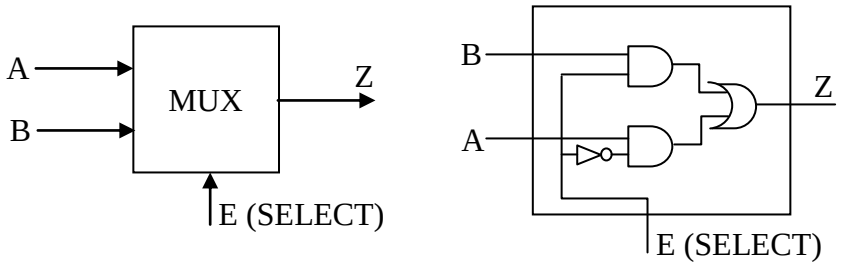
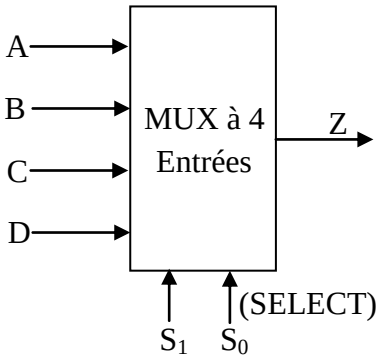


Figure IV.16 : Multiplexeur à deux entrées

V.2. Multiplexeurs à quatre entrées



S_1	S_0	Z
0	0	A
0	1	B
1	0	C
1	1	D

$S_0S_1 = 00 \rightarrow Z = A$
$S_0S_1 = 01 \rightarrow Z = B$
$S_0S_1 = 10 \rightarrow Z = C$
$S_0S_1 = 11 \rightarrow Z = D$

$$Z = A\overline{S_0}\,\overline{S_1} + BS_0\overline{S_1} + C\overline{S_0}\,S_1 + DS_0S_1$$

Figure IV.13 : Multiplexeur à quatre entrées

Le même principe de base va servir à construire un multiplexeur à quatre entrées, dans ce cas il y a quatre entrés qui son transmises à la sortie selon un choix entre quatre combinaisons possibles, ces combinaisons sont fournies par deux entrées de sélection S_1 et S_2

Pour le cas d'un multiplexeur à 8 entrés (I_0 I_1 ……I_7) avec trois entrées de sélection S_0 S_1 et S_2 on a l'équation suivante :

$$Z = I_0\overline{S_2}\,\overline{S_1}\,\overline{S_0} + I_1\overline{S_2}\,\overline{S_1}\,S_0 + I_2\overline{S_2}\,S_1\,\overline{S_0} + I_3\overline{S_2}\,S_1S_0 + I_4S_2\,\overline{S_1}\,\overline{S_0} + I_5S_2\,\overline{S_1}\,S_0 + I_6S_2S_1\,\overline{S_0}$$

$$+ I_7S_2S_1S_0$$

V.3. Multiplexeurs 74xx151

C'est un multiplexeur à 8 entrées qui dispose d'une entrée de validation $\overline{E}$ et qui fourni une sortie normale Z et une sortie complémentée $\overline{Z}$

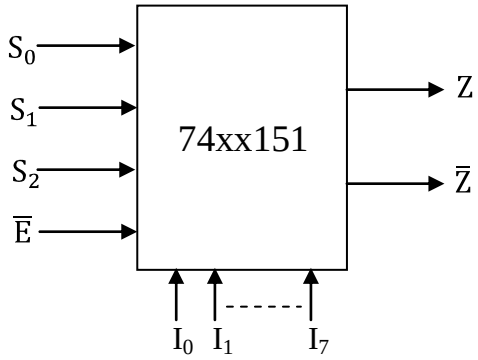


Figure IV.17 : Multiplexeur 74xx151 à 8 entrées

Quand $\overline{E} = 0$: les entrées SELECT S_0 S_1 et S_2 choisissent une entrée de données (I_0 I_1 ……I_7) dont les valeurs se trouvent sur la sortie Z, et auand $\overline{E} = 1$ le multiplexeur est invalide de sorte que $Z = 0$ quelque soit le code d'entrée de sélection

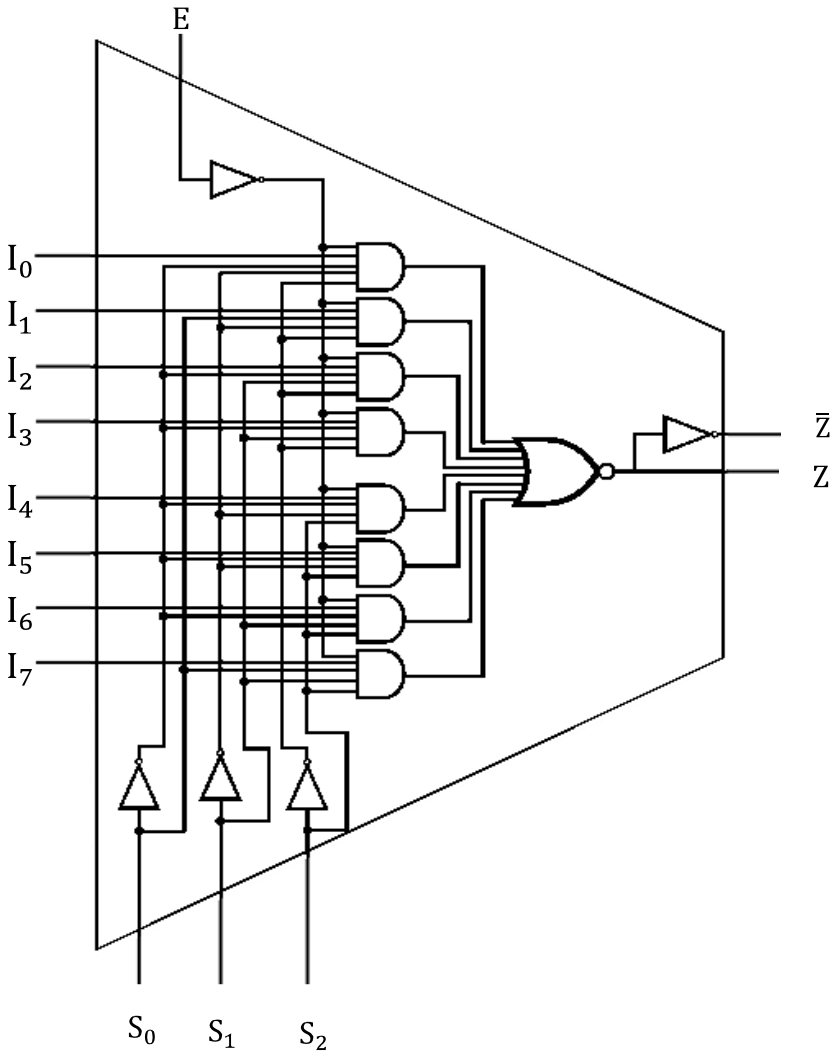


Figure IV.18 : Logigramme d'un Multiplexeur 74xx151

$\overline{E}$	S_2	S_1	S_0	$\overline{Z}$	Z
1	Ø	Ø	Ø	1	0
0	0	0	0	$\overline{I_0}$	I_0
0	0	0	1	$\overline{I_1}$	I_1
0	0	1	0	$\overline{I_2}$	I_2
0	0	1	1	$\overline{I_3}$	I_3
0	1	0	0	$\overline{I_4}$	I_4
0	1	0	1	$\overline{I_5}$	I_5
0	1	1	0	$\overline{I_6}$	I_6
0	1	1	1	$\overline{I_7}$	I_7

$$Z = E\, I_0 \overline{S_2}\ \overline{S_1}\ \overline{S_0} + E\, I_1 \overline{S_2}\ \overline{S_1}\ S_0 + E\, I_2 \overline{S_2}\ S_1\ \overline{S_0} + E\, I_3 \overline{S_2}\ S_1 S_0 + E\, I_4 S_2\ \overline{S_1}\ \overline{S_0} + E\, I_5 S_2\ \overline{S_1}\ S_0 + E\, I_6 S_2 S_1\ \overline{S_0} + E\, I_7 S_2 S_1 S_0$$

V.3. Réalisation d'un multiplexeur à 16 entrées à l'aide de deux circuits intégrés 74xx151

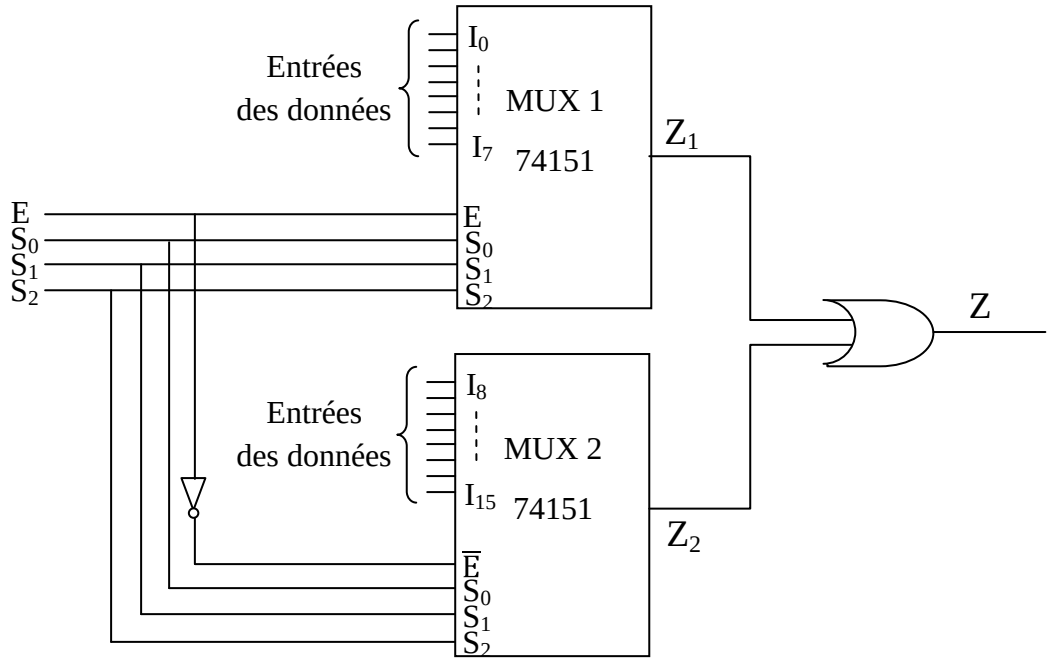


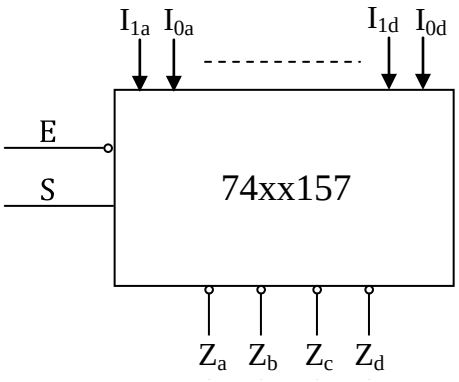
Figure IV.19 : Multiplexeur à 16 entrées à l'aide de deux MUX 74xx151

Ce circuit est constitué de deux multiplexeurs à 8 entrées 74151, les deux sorties de ces deux multiplexeurs sont réunis par une porte OU pour produire une seule sortie Z, il fonctionne comme un multiplexeur à 16 entrées.

Les quatre entrées de sélection E, S_2, S_1, S_0 choisissent l'une des 16 entrées pour la faire passer jusqu'à Z, l'entrée E choisie le multiplexeur qui est validé, quand E = 0 Le MUX 1 est validé et si E = 1 Le MUX 2 est validé

V.4. Quatre multiplexeur à 2 entrées (74xx157)

Il s'agit d'un boitier intégré qui enferme 4 MUX à deux entrées, sont schéma logique est le suivant :



$\overline{E}$	S	Z_a	Z_b	Z_c	Z_d
1	Ø	0	0	0	0
0	0	I_{0a}	I_{0b}	I_{0c}	I_{0d}
0	1	I_{1a}	I_{1b}	I_{1c}	I_{1d}

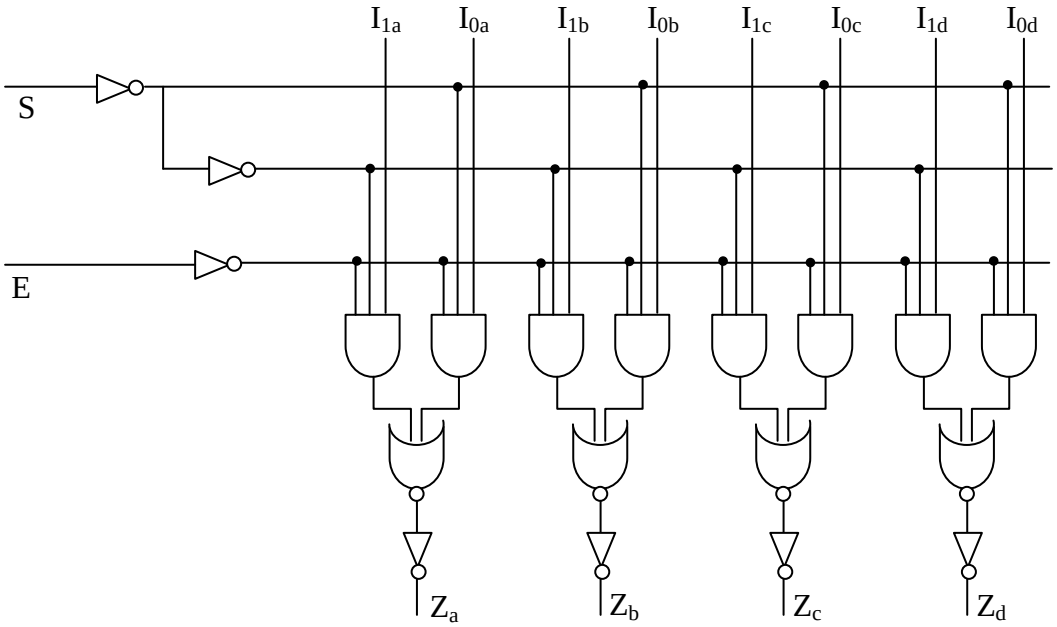
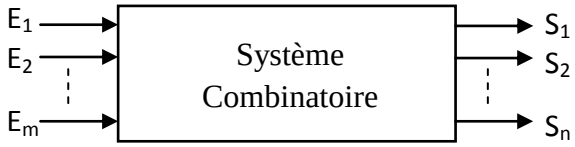


Figure IV.20 : Multiplexeur 74157

Chapitre V : Analyse et synthèse des circuits logiques séquentiels (Méthode d'Hoffman)

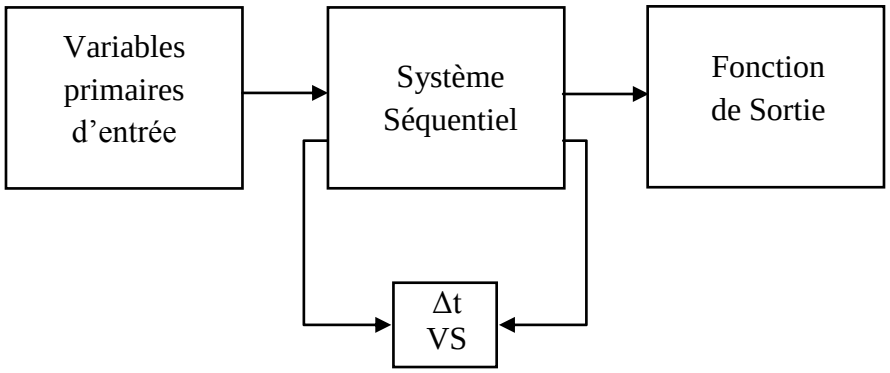
I. Généralités

En logique combinatoire les grandeurs de sorties sont entièrement définies en fonctions des grandeurs d'entrées



$Si = f(E_1, E_2 \ldots E_n)$ Avec i = 1, 2 , ….. , m

En logique séquentiel on peut avoir deux états différents pour une même combinaison des variables n'entrée. Donc la sortie d'un système séquentiel dépond non seulement des variables d'entrées mais aussi de son état antérieur.
Ces états antérieurs sont représentés par des nouvelles variables appelées variables secondaires



VS : Variables Secondaires
Δt : Retard introduit par la réponse du système

II. Plan Général d'étude d'un automatisme par la méthode d'HUFFMAN

La méthode d'HUFFMAN comporte 5 étapes :

1) Dénombrement des états stables (Graphe des états)
2) Etablissement de la matrice primitive
3) La matrice réduite – Polygone de liaison
4) Matrice des variables secondaires
5) Matrice de sortie.

1) Dénombrement des état stables (Graphe des états)

Le graphe des états est constitué de la manière suivante :
L'état stable initial est numéroté ①
Les états stables suivants sont numérotés à leurs tours et chaque numéro est entouré d'un cercle
Les liaisons joignant ces différents états indiquent les transitions possibles d'un état stable vers un autre, représentés par des flèches

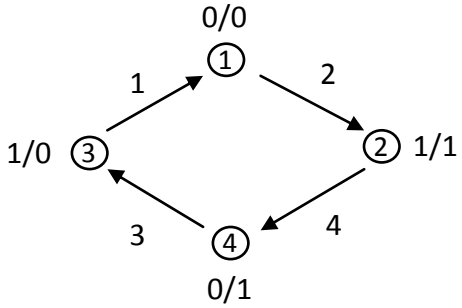
Exemple

Il s'agit d'allumer et d'éteindre une lampe L avec un même bouton poussoir a
- Donner les différents états stables ainsi que le graphe des états

Table de vérité :

Etats Stable	a	L
①	0	0
②	1	1
③	0	1
④	1	0

Graphe des états :



Le passage entre ① et ② est unidirectionnel
En partant de l'état de repos ①, si on appuie sur le bouton a la lampe s'allume : c'est le deuxième état stable ②
Si en relâche le bouton on ne revient pas à l'état initial mais la lampe reste allumé ; c'est le nouveau état (0/1)

2) Matrice primitive :

Elle est construite de la manière suivante :
Le nombre de colonnes est égale au nombre de combinaisons que l'on peut former avec les variables d'entrée (primaires) et le nombre de lignes est égale au nombre des états stables
Chaque état stable est écrit dans l'ordre chronologique d'apparition dans la colonne correspondante à la combinaison des variables primaires qui lui a donnée naissance.
Chaque état instable est inscrit dans la même colonne de l'état stable qui le suit et dans la même ligne de l'état stable qui le précède
- Les sorties seront ensuite sous forme de colonnes, pour chaque état stable en indique l'état des sorties.
Rq : les cases vides (hachurées) restantes correspondent à des impossibilités de fonctionnement

Etat \ a	0	1	L
1	①	2	0
2	3	②	1
3	③	4	1
4	1	④	0

Les chiffres non entourés de cercles représentent les états transitoires (instables)

Le passage de l'état stable ① à l'état stable ② s'effectue en passant par l'état transitoire 2 qu'on inscrit dans la même colonne de l'état stable ② et dans la même ligne que l'état stable ① et de même pour les autres états transistors

3) Matrice réduite :

On va introduire des variables supplémentaires appelées variables secondaires dont le nombre croit avec le nombre de lignes de la matrice primitive

Exemple Pour 5 états stables, il faut trois variables secondaires $2^2 < 5 < 2^3$

Règle de réduction de la matrice primitive :

La matrice réduite est obtenue par fusionnement des lignes de la matrice primitive
On dit que deux lignes fusionnent si dans une colonne donnée les deux lignes considérées ont soit un état stable et un état transistor de même numéro, soit deux états instables de même numéro, soit un état (stable ou instable) et une case hachurée.

Exemple

①	3	2	
1	3		4

→

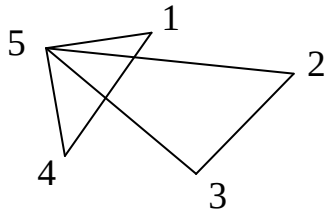
①	3	2	4

Pour effectuer une réduction d'une matrice primitive on fait appel au polygone de liaison construit de la manière suivante
- Les sommets du polygone représentent les lignes de la matrice primitive.
- On joint deux sommets ensemble si les lignes correspondantes fusionnent.

Exemple

ab	00	01	11	10	S
1	①	4		2	0
2	3		5	②	1
3	③	4		2	1
4	1	④	5		0
5		4	⑤	2	1

→



Les solutions possibles : 1- 4, 2-3-5
Ou bien 2-3, 1-4-5

Le polygone de liaison possède 5 sommets car la matrice primitive possède 5 états stables et présente deux possibilités de fusionnement (deux solutions)
Remarque : il est préférable de fusionner les lignes de la matrice primitive pour lesquelles la sortie présente le même état donc on aura le choix suivant :

:

ab	00	01	11	10	S
1- 4	①	④	5	2	0
2-3-5	③	4	⑤	②	1

4) Matrice de variables secondaires :

Ce sont des tableaux de KARNAUGH qui comportent le même nombre de cases que la matrice réduite et consiste à attribuer des combinaisons binaires dans le code Gray aux lignes de la matrice réduite donc leur attribuer des variables qui sont des variables secondaires.

Si "***l***" est le nombre de ligne de la matrice réduite et "***v***" et le nombre de variables secondaires

Alors : $2^{v-1} < l \leq 2^{v}$

Exemple : 4 lignes : Deux variables secondaires ***x, y***

état	0	1
1	①	2
2	3	②
3	③	4
4	1	④

Matrice des variables secondaires →

xy \ *a*	0	1
00	00	01
01	11	01
11	11	10
10	00	10

On donne aux états stables d'une ligne la même valeur binaire que celle de la variable secondaire pour la ligne considérée. La valeur binaire des états instables sera celle des états stables ayant le même numéro.

Equation des variables secondaires :

Dans la matrice de variables secondaires tous les éléments de gauche de chaque combinaison xy servirent a construire le tableau de KARNAUGH relatif à la variable x et tous les éléments de droite seront ceux de la variable y

xy \ *a*	0	1
00	0	0
01	1	0
11	1	1
10	0	1

$X = \overline{a}y + ax$

xy \ *a*	0	1
00	0	1
01	1	1
11	1	0
10	0	0

$Y = \overline{a}y + a\overline{x}$

5) Matrice de Sortie :

-Les équations des fonctions de sortie sont obtenues à partir de tableaux de KARNAUGH construit de la manière suivante :

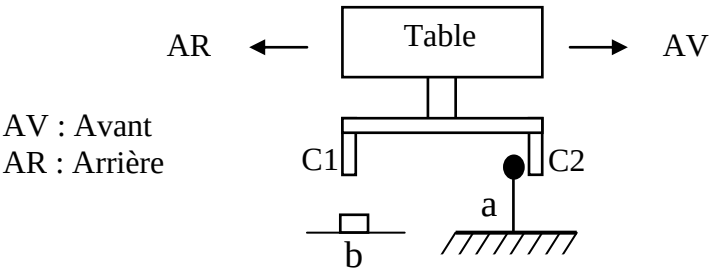
-Les états stables de la matrice réduite seront remplacés par la valeur binaire correspondante de la fonction de sortie. Pour les états instables la variable de sortie prend la même valeur logique de l'état stable

xy \ *a*	0	1
00	0	1
01	1	1
10	1	0
11	0	0

$$L = \overline{a}y + a\overline{x} = Y$$

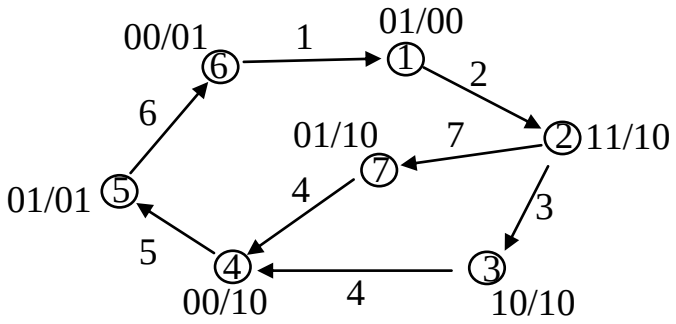
Exemple 2

On construit le dispositif illustré sur la figure suivante :

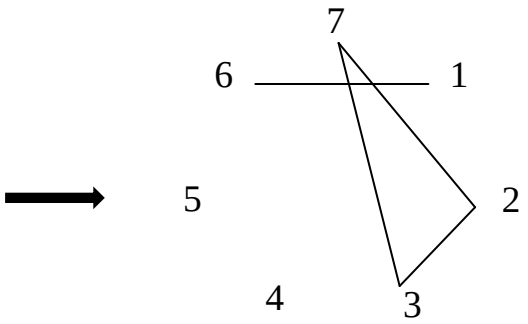


AV : Avant
AR : Arrière

A l'état initial la table est à l'arrêt. Le bouton a est actionné par C2, l'action sur le bouton b provoque le déplacement de la table vers l'avant jusqu'a ce que C1 atteint le bouton a, la table change de direction et se déplace vers l'arrière. Une fois que le bouton a est actionné par C2 la table s'arrête.



Etat \ *ab*	00	01	11	10	AV	AR
1		①	2		0	0
2		7	②	3	1	0
3	4			③	1	0
4	④	5			1	0
5	6	⑤			0	1
6	⑥	1			0	1
7	4	⑦			1	0



Les solutions possibles sont : 2, 3, 7 - 1, 6 - 4 - 5

ab *Etat*	00	01	11	10
1-6	⑥	①	2	
2-3-7	4	⑦	②	③
4	④	5		
5	6	⑤		

Matrice de variables secondaires :

ba *xy*	00	01	11	10
00	00	00	01	Ø
01	11	01	01	01
11	11	10	Ø	Ø
10	00	10	Ø	Ø

$$2^{v-1} < I \leq 2^{v}$$

On a 7 états stables donc le nombre de variables secondaires est 2

xy

ba *xy*	00	01	11	10
00	0	0	0	Ø
01	1	0	0	0
11	1	1	Ø	Ø
10	0	1	Ø	Ø

$$x = \bar{a}\bar{b}y + a\,\bar{b}x$$

ba *xy*	00	01	11	10
00	0	0	1	Ø
01	1	1	1	1
11	1	0	Ø	Ø
10	0	0	Ø	Ø

$$y = b + y(\bar{x} + \bar{a})$$

Matrice de sortie:

ba *xy*	00	01	11	10
00	01	00	Ø	Ø
01	Ø	10	10	10
11	10	Ø	Ø	Ø
10	Ø	01	Ø	Ø

ba *xy*	00	01	11	10
00	01	00	Ø	Ø
01	Ø	10	10	10
11	10	Ø	Ø	Ø
10	Ø	01	Ø	Ø

$$AV = y$$

ba *xy*	00	01	11	10
00	01	00	Ø	Ø
01	Ø	10	10	10
11	10	Ø	Ø	Ø
10	Ø	01	Ø	Ø

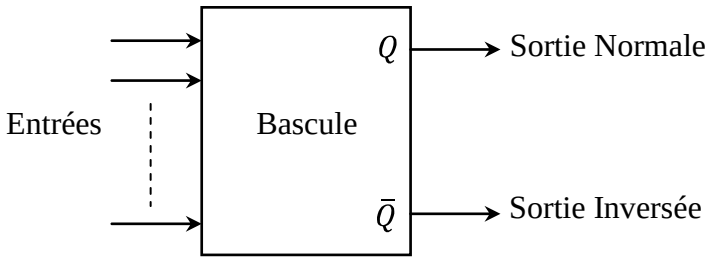
$$AR = \bar{a}\bar{b}\bar{x}\bar{y} + x\bar{y} = \bar{y}(x + \bar{a}\bar{b}\bar{x})$$

III. Les circuits séquentiels

Les sorties externes d'un système séquentiel dépendent autant des entrées externes que des informations mémorisées. L'élément de mémorisation le plus important est la Bascule qui est constituée d'un ensemble de portes logiques tel qu'il existe de différentes façons pour monter ces portes logiques afin d'obtenir des bascules

III.1. Les Bascules

Ce sont des cellules de mémoire permanentes a deux états stables, le passage d'un état stable à un autre se fait par commande extérieure, sont symbole générale est présenté par la figure suivante : $\bar{y}$



Etats de sorties :

$Q = 1$, $\bar{Q} = 0$: Etat haut
$Q = 0$, $\bar{Q} = 1$: Etat bas

On remarque deux sorties désignées par Q et $\bar{Q}$ qui sont l'inverse l'une de l'autre. La sortie $\bar{Q}$ est la sortie inverse et elle présente le nom de la sortie normal Q. Chaque fois que l'on fait référence à l'état d'une bascule on désigne la situation de sa sortie normale Q. Une bascule à donc deux états de fonctionnement.

III.1.1. Eléments de mémoire en porte NAND (Bascule RS)

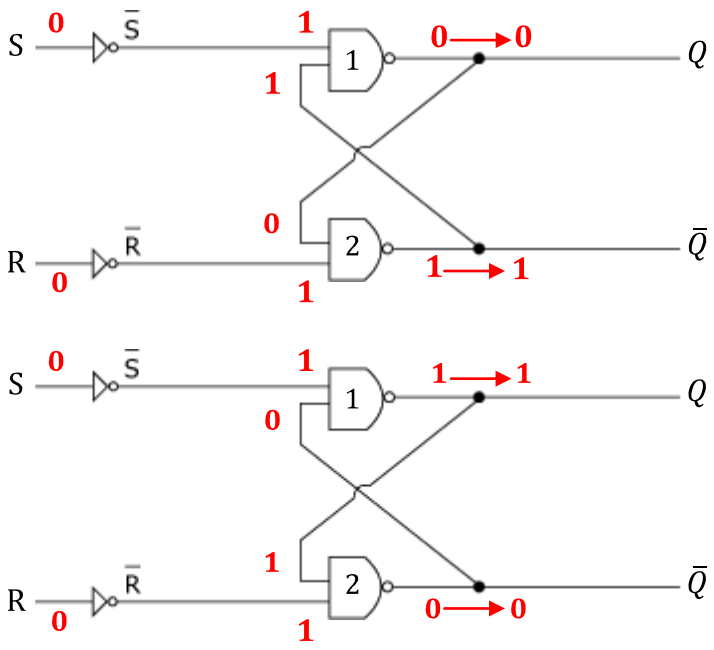
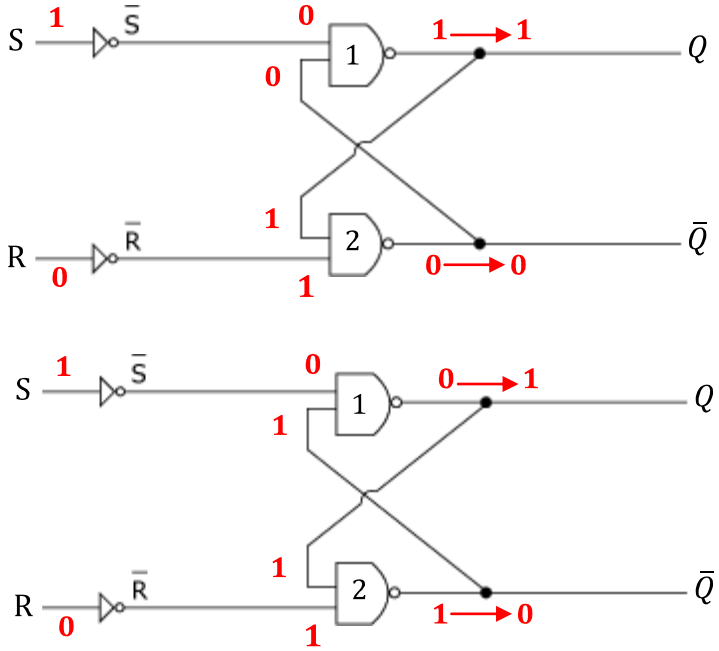


Figure V.1 Bascule RS à l'aide de porte NAND

La bascule la plus élémentaire est construite à base de deux portes NAND ou de deux portes NOR, ces deux portes (NAND ou NOR) sont raccordées de sorte que la sortie de chacune est raccordée à l'entrée de l'autre. Les sorties de ces portes appelées Q et $\bar{Q}$ respectivement sont les sorties de l'élément de mémoire.
Les entrées de cette bascule sont désignées par **S** (Set : Mise à 1) et **R** (Reset : Mise à zéro), elles se trouvent normalement toutes les deux au niveau bas, ce qui donne les deux possibilités montrées en Fig.
Donc dans les deux cas ($Q = 1$ et $Q = 0$) l'application d'un niveau bas aux deux entrées R et S maintient les sorties de la bascule dans leurs état initiale

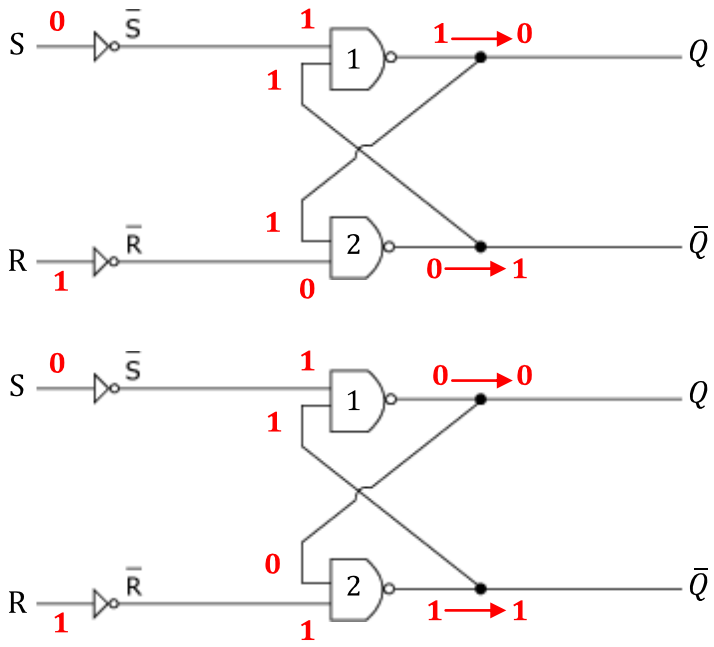
III.1.2. Mise à 1 de la Bascule



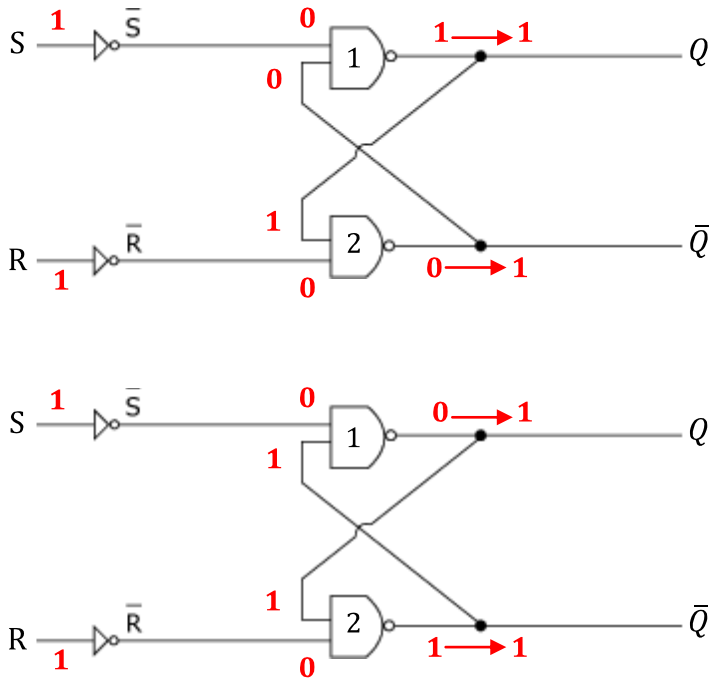
L'application d'une impulsion du niveau haut à l'entrée S met toujour la bascule à l'état haut ($Q = 1$ et $\bar{Q} = 0$) quel que soit son état initial : c'est effectivement sa mise à 1 .

III.1.3. Mise à 0 de la Bascule

L'application d'une impulsion du niveau haut à l'entrée R met toujour la bascule à l'état bas ($Q = 0$ et $\bar{Q} = 1$) quel que soit son état initial : c'est effectivement sa mise à 0.



III.1.4. Mise à 0 et à 1 simultanément



Cette situation produit des niveaux hauts en deux sorties de la bascule de sorte que $(Q = \bar{Q} = 1)$ donc il s'agit d'une condition indésirable puisque les deux sorties sont supposées être l'inverse l'une de l'autre

Récapitulation (Résumé)

La déscription précédente peut se mètre sous la forme d'une table de vérité

S	R	Q_{n-1}	Q_n	
0	0	0	0	Mémorisation
0	0	1	1	
0	1	0	0	Mise à 0
0	1	1	0	
1	0	0	1	Mise à 1
1	0	1	1	
1	1	0	Ø	Etat Indésirable
1	1	1	Ø	

Q_{n-1} \ SR	00	01	11	10
0	0	0	Ø	1
0	1	0	Ø	1

$$Q_n = S + \bar{R}Q_{n-1} \rightarrow Q = S + \bar{R}Q$$

$$Q = \overline{\overline{S + \bar{R}Q}} = \overline{S.\overline{\bar{R}Q}}$$

III.1.5. Bascule RS à l'aide des portes NOR

$$Q = S + \bar{R}Q$$

$$\bar{Q} = \overline{S + \bar{R}Q} = \overline{S + \overline{\overline{\bar{R}Q}}} = \overline{S + \overline{R + \bar{Q}}} = \bar{Q}$$

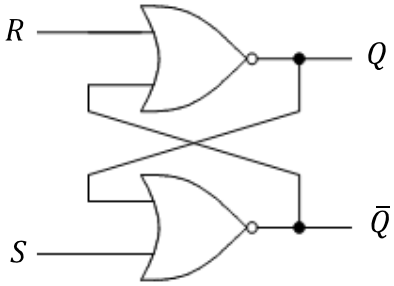


Figure V.2. Bascule RS à l'aide de porte NOR

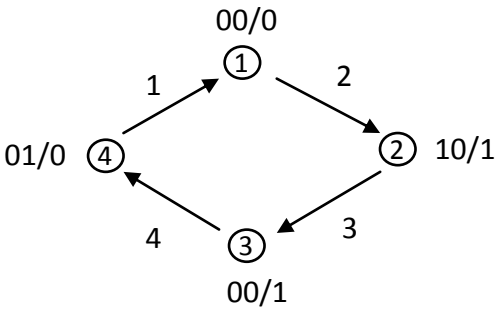
III.1.6. Analyse de la bascule RS par la méthode d'HUFFMAN :

1) <u>Dénombrement des état stables (Graphe des états)</u>

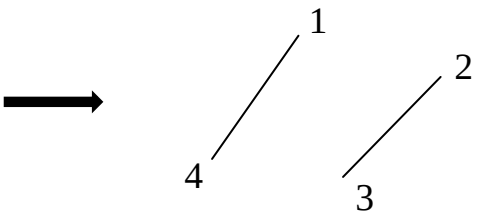
Table de vérité :

Etats Stable	S	R	Q
①	0	0	0
②	1	0	1
③	0	0	1
④	0	1	0
⑤	0	0	0

Graphe des états :



SR	00	01	11	10	Q
1	①	4		2	0
2	3			②	1
3	③	4		2	1
4	1	④			0



Les solutions possibles sont : 1 - 4, 2 - 3

ab / *Etat*	00	01	11	10
1-4	①	④		2
2-3	③	4		②

$2^{v-1} < l \leq 2^{v}$

On a 4 états stables donc le nombre de variables secondaires est 1

Matrice de variables secondaires :

SR / x	00	01	11	10
0	0	0	Ø	1
0	1	0	Ø	1

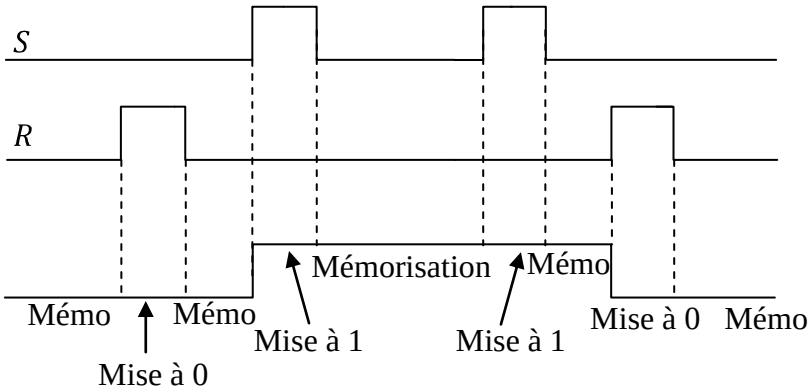
$X = S + \bar{R}x$

Matrice de sortie :

SR / X	00	01	11	10
0	0	0	Ø	1
0	1	0	Ø	1

$Q = S + \bar{R}x = X$

Exemple :
Les formes d'onde de la figure suivante sont appliquées aux entrées de la bascule RS
Trouver la forme d'onde Q (Supposant qu'au départ Q=0)

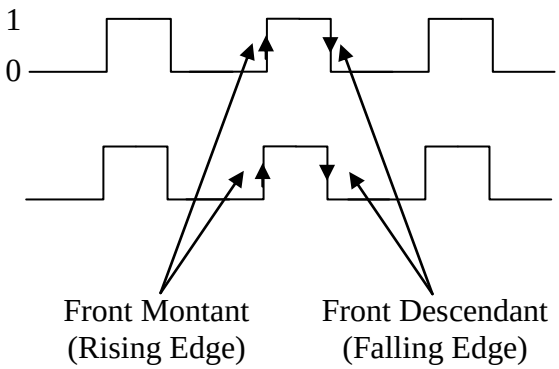


III.2. Signal d'horloge et Bascules Synchrones

Un signal d'horloge présente un signal électrique oscillant qui a comme rôle de rythmer les actions d'un circuit. Sa période est appelée cycle d'horloge. À chaque cycle d'horloge, des

calculs peuvent être effectués en utilisant les sorties de bascules. Ce signal est d'onde rectangulaire ou carrée

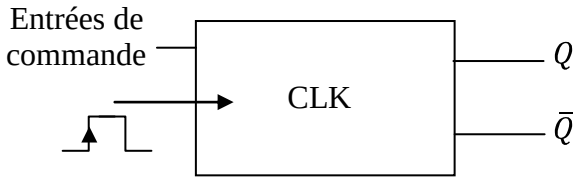
Le Rapport Cyclique :
Est équivalent à $\frac{TI}{T}$
T_I : Durée d'impulsion
T : Période du signal



Quand un signal d'horloge passe de 0 à 1 on parle dans ce cas d'une transistion positive ou d'un front montant et s'il passe de 1 à 0 on parle d'un front déscendant, ces fronts sont identifiés au moyen de flèches sur le signal d'horloge.

III.2.1. Bascules Synchrones

Toute bascule synchrone dispose d'une entr »e d'horloge désigné par (CLK)H. Les entrées de commandes déterminent l'état de sortie de la bascule (comme les entrées S et R) mais cet état n'appaeais qu'au moment ou se produit la transistion dans le signal d'horloge, autrement dit leur effet est synchronisé avec l'application du signal d'horloge CLK, c'est pour cte raison qu'elle sont appelés entrés synchronisées.



III.2.2. Temps de stabilisation et temps de maintient

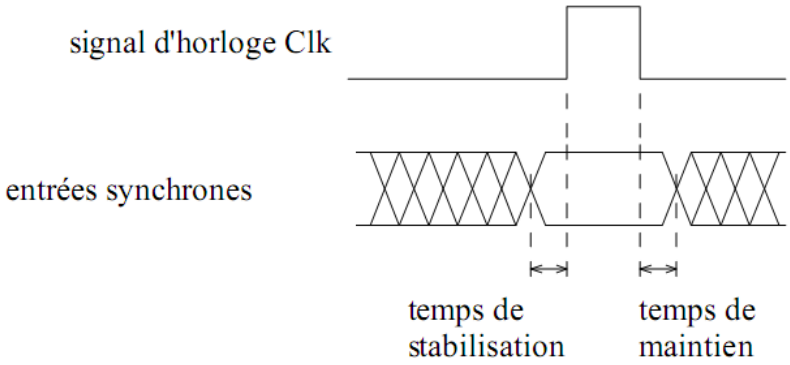


Figure V.3. Temps de stabilisation et temps de maintient

Deux conditions de synchronisation doivent etre vérifiées pour qu'une bascule synchrone répond correctement a ses entrés de commande présentent en fig.V.3

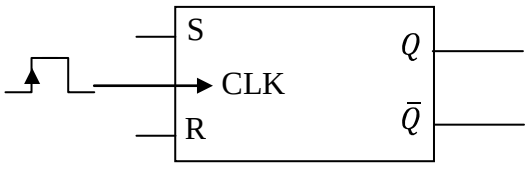
Temps de stabilité T_S

C'est l'interval qui précéde immédiatement le front déclancheur du signal d'horloge, pendant lequel l'entrée synchrne doit etre gardé au niveau approporié. Si on ne respecte pas ce temps, il n'est pas garantie que la bascule répondera correctement à l'arrivé du front.

Temps de Maintient T_M

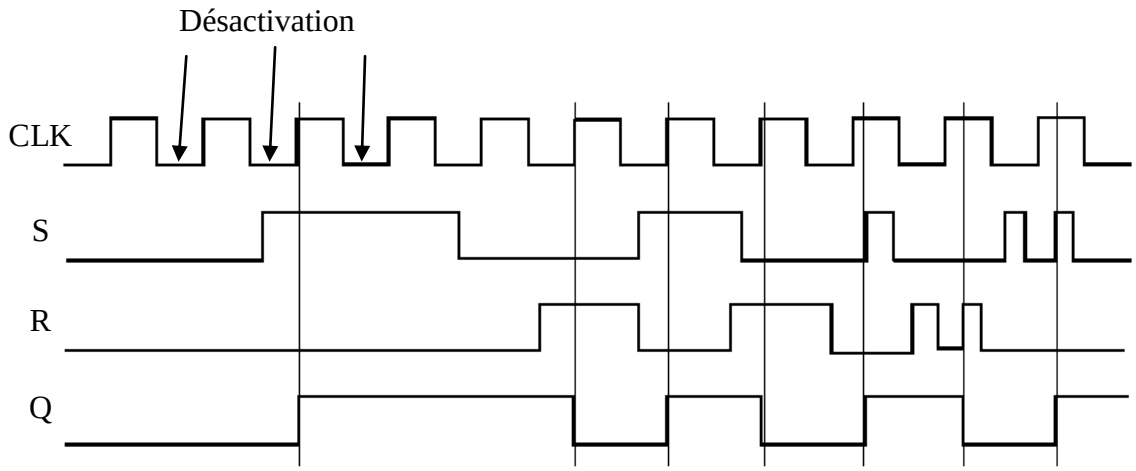
C'est l'interval qui précéde immédiatement le front déclancheur du signal, pendant lequel l'entrée synchrne doit etre au niveau approprié dans ce cas l'entré de commande doit rester stable (inchangée) pendant une durée de $T=T_S+T_M$
Les bascule en généralement des T_S et T_M de l'ordre nanoseconde

III.2.3. Bascules RS Synchrone



S	R	CLK	Q
Ø	Ø	0	Inchangée
0	0	1	Mémorisation
0	1	1	0
1	0	1	1

Bascule désactivée



C'est une bascule qui passe d'un état à un autre seulement quend le signal d'horloge passe au niveau approprié les entrée de commade R et S réagissent au niveaua logiques de CLK 1 ou 0 et non au fronts montants ou déscendants
L'entrée CLK est celle qui déclanche le changement dans la bascule dicté par les valeurs appliquées sur S et R

III.3. Bascules JK Synchrone

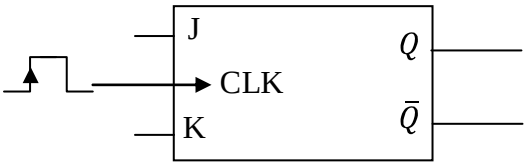
C'est une bascule qui comporte deux entrées de contrôle : J (Jack) et K (King). Le fonctionnement est synchrone à une entrée d'horloge CLK, donc sortie ne change d'état qu'au moment d'un front d'horloge, montant ou descendant :

Pour J = K = 0, il y a conservation du dernier état logique Q_{n-1} indépendamment de l'horloge : état mémoire.

Pour J = K = 1, le système bascule à chaque front d'horloge, la bascule passe toujours à l'état opposé, on dit que c'est le mode de basculement

Pour J différent de K, la sortie Q est la même que l'entrée J et la sortie $\overline{Q}$ est la même que l'entrée K à chaque front d'horloge.

Le symbole et la table de vérité de cette bascule est présenté comme suivant :



	J	K	CLK	Q_n
Mémorisation	0	0	↑	Q_{n-1}
Mise à 0	0	1	↑	0
Mise à 1	1	0	↑	1
Basculement	1	1	↑	$\overline{Q}_{n-1}$

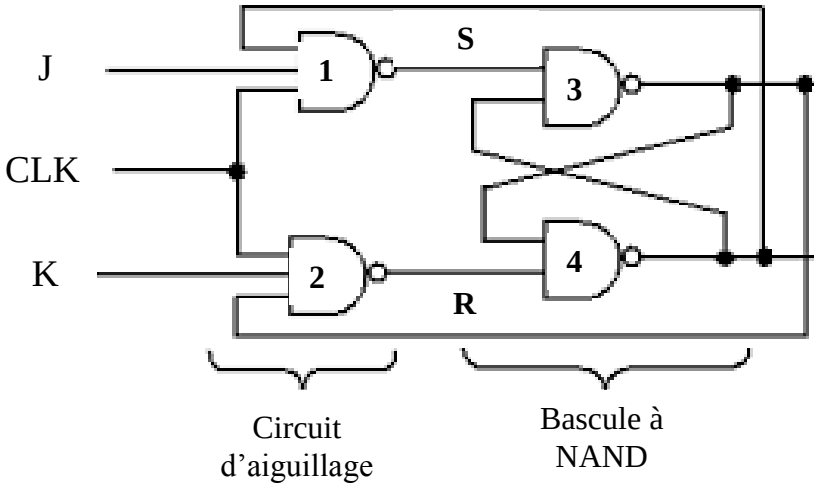


Figure V.4. Bascule JK

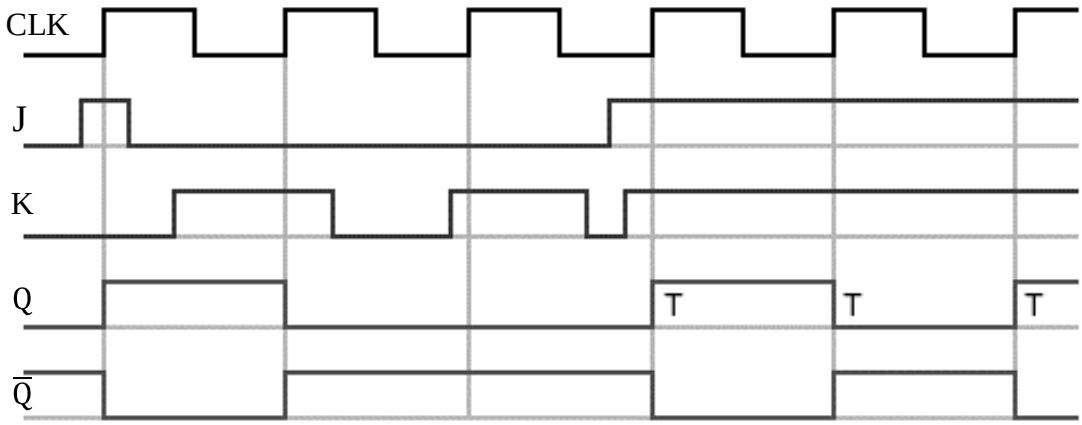


Figure V.5. Logigramme d'une Bascule JK

III.4. Bascule D Synchrone

Contrairement aux bascules RS et JK, cette bascule ne possède qu'une entrée de commande synchrone, appelée Δ_D Avec Delta Δ : Données ;
Delay D : Retard.

Le fonctionnement de cette bascule est très simple : La sortie Q prend l'état de l'entrée D quand la bascule est déclenchée par le front approprié (Montant ou descendant)
Autrement, le niveau actuellement en D se retrouvera mémorisé dans la bascule à l'instant du front déclencheur

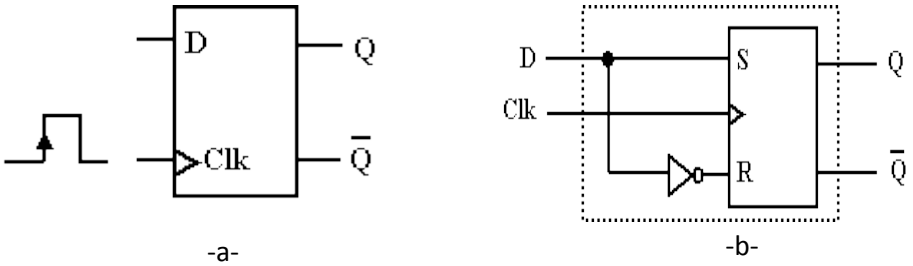


Figure V.6 : a-Bascule D b- Bascule D à l'aide d'une Bascule SR

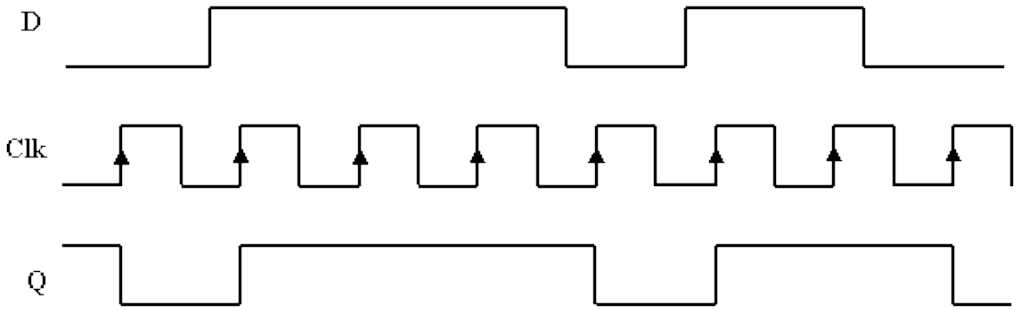


Figure V.7 : Logigramme d'une Bascule D

III.5. Entrées Asynchrones

La majeure partie des bascules synchrones possèdesnt en plus, des entrées asynchrones qui agissent indépendament des entrées syncchrones et du signal d'horloge, on a recour à de tel entrées pour forcer à toutes instant la remise à 1 et la remise à 0 quelque soit les conditions des entrées synchrones
Une autre facon de les présenter est de dire que se sont des entrées prioritaires qui imposent un état à la bascules. Malgré les commandes lancées par les autres entrées, leurs action est immédiate
Ces entrées sont généralement actives à état bas

Symbole

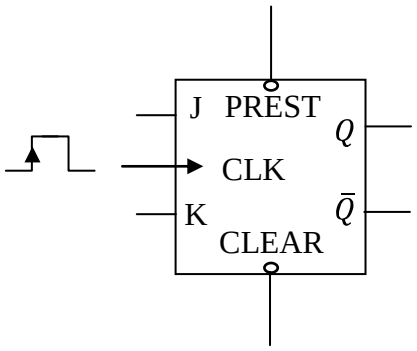


Table de Vérité

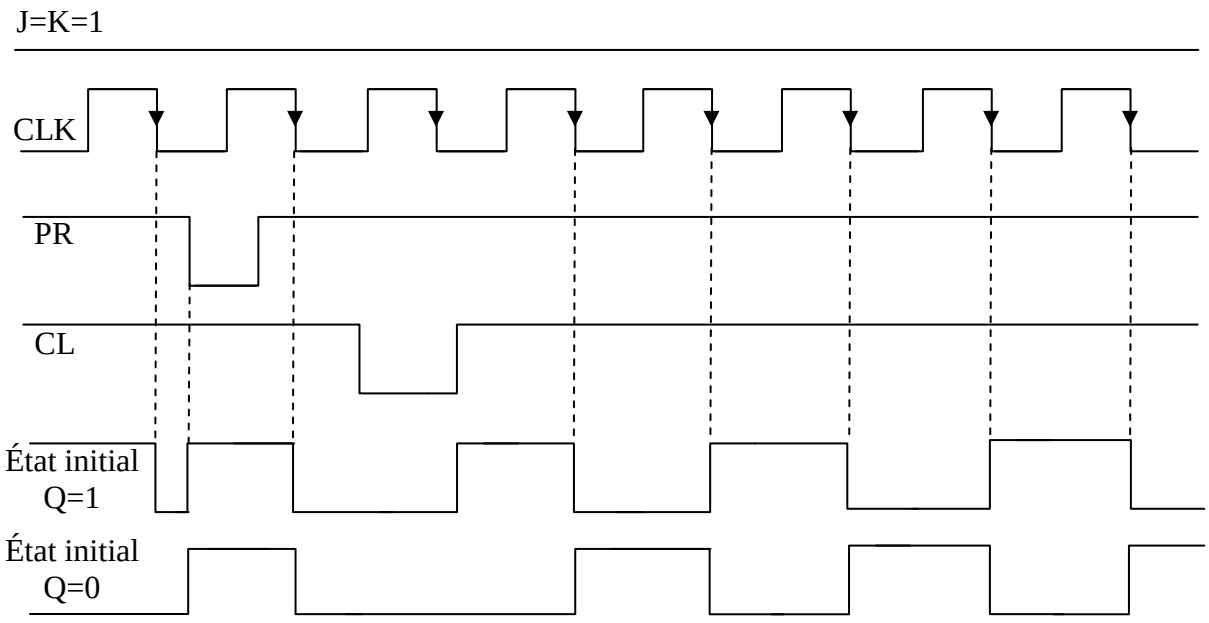
PR	CL	Réponse de la Bascule
0	0	Front=Synchrone
0	1	Q = 1
1	0	Q = 0
1	1	Unitulisée

PREST : entrées asynchorone pour la mise à 1
CLEAR : entrées asynchorone pour la mise à 0

Priorités des entrées d'une bascule asynchrone (Possédant des entrées asynchrones)

L'ordre des priorité des entrées d'une commande asynchrone est le suivant :
1- PR, CL
2- CLK
3- Entrées Synchrones (J, K)

Exemple:



Exemple des Bascules en CI

7474 : Boitier integrant 2 bascules D Synchrones (TTL)
74LS112 : Deux bascules JK Synchrones (TTL)
401313 : Deux bascules D Synchrones (MOS)

74HC112 : Deux bascules JK Synchrones (CMOS)
Toutes ces bascules ont un temps de maintient très petit ce qui une caractéristique de la plus part des bascules modernes déclanchées par un signal d'horloge. La séries 74HC des dispositifs CMOS à des paramètres de synchronisation qui se comparent à ceux des dispositifs TTL par contre la série 4000 est beaucoup plus longte

IV. Les Compteurs binaires

Les bascules sont des outills très polyvalents grace aux quels on peut envisager un grand nombre d'applications, parmis les courantes on trouve les compteurs

Définition

Un compteur binaire est un dispoitif possédant une entrées de comptage appelée Horloge, une entrée de remise à 0 et un nombre variable de sorties codées en binaire et dépendant du nombre d'impulsions d'horloge reçues par le compteur.

IV.1. Les compteurs Asynchrones

C'est un ensemble de bascules connéctées de la manière suivante
-Les impulsion d'horloges sont appliqué à la borne CLK de la première Bascule (A), celle-ci commute chaque fois qu'arrive un front déclancheur du signal d'horloge.
-La sortie normale de la bascule A représente le signal d'horloge pour la bascule suivante (B), de sorte que cette dernière commute chaque fois que le signal provenant de A passe par une transition
-De la meme manière la 3eme Bascule (C) commute quand la sortie de la bascule (B) fournie le front déclancheur de la bascule (C) et ainsi de suite
Exemple Soit le compteur Asynchrone à trois bascule suivant :

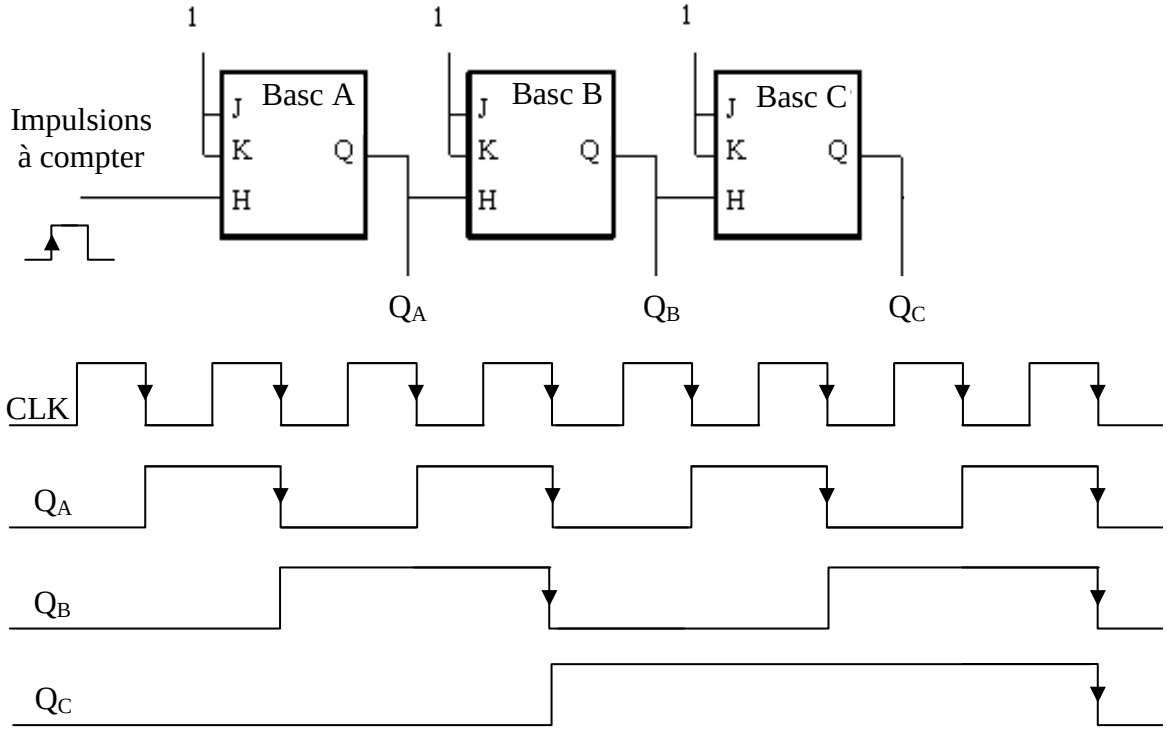


Figure V.8 : Compteur Asynchrone à trois bascule

A l'arrivé du 8eme impulsionles bascules commutent et reviennent à leurs état initial 000, on dit dans ce cas que le compteur est recyclé et qui'il recommence le cycle de dénombrement des impulsions qui arrivent

IV.2. Le Modulo

Le compteur représenté ci-dessus possède 8 états distingues (000 …. 111) on dit alors que c'est un compteur asynchrone (à propagation) modulo 8. Le modulo est donc le nombre d'état occupé par le compteur pendant un cycle complet avant son retour à l'état initial ;
Le modulo est porté à une valeur plus élévée simplement en ajoutant d'autres bascules :
N bascules ---> Module 2^n

Exemple: 4 Bascules ----> Modulo $2^4 = 16$

IV.2.1. Division de la fréquence

Nous remarquons dans le montage du compteur asynchrone que la sortie de chaque bascule est une forme d'onde dont la fréquence est la moitié de celle du signal d'horloge qui lui est appliqué

Exemple:

Supposant que la fréquence du signal d'horloge de la figure précédente est de 8KHz donc

$T = \frac{1}{8.10^3} = 0.125.10^{-3}\ s = 0{,}125\ ms$

fQ_A = 4KHz donc : $T = \frac{1}{4.10^3} = 0.25.10^{-3}\ s = 0{,}25\ ms$

fQ_B = 2KHz donc : $T = \frac{1}{2.10^3} = 0.5.10^{-3}\ s = 0{,}5\ ms$

fQ_c = 1KHz donc : $T = \frac{1}{1.10^3} = 1.10^{-3}\ s = 1\ ms$

En générale la sortie de la dernière bascule d'un compteur est une onde dont la fréquence est celle du signal d'horloge divisée par le modulo du compteur. Donc le compteur modulo 8 peut être appelé par compteur diviseur par 8

IV.2.2. Compteur modulo < 2^N :

Les compteurs asynchrones comme celui de la figure précédente ne peut avoir des modulos différents de 2N (N : Nombre de Bascule) cependant il est possible de modifier ce compteur élémentaire pour obtenir de modulo < 2^N en permettant au compteur de sauter certaines combinaisons de la suite des nombres binaires.

Exemple

Pour réaliser un compteur modulo 5 on a besoin de 3 bascules car $2^2=4<5$ comme le montre la figure suivante :

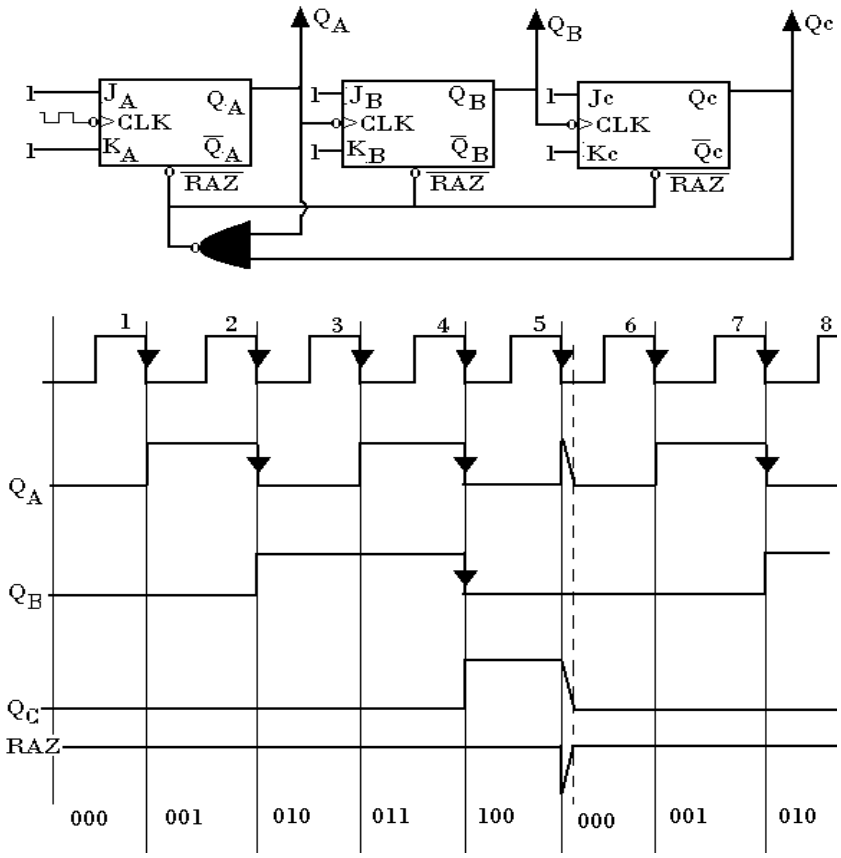


Figure V.9 : Compteur Asynchrone Modulo 5

La sortie de la porte est connectée à l'entrée de remise à Zéro (RAZ) ou (CLEAR) de chaque bascule, tant que cette sortie est à 1 le comptage n'est pas affecté, lorsqu'elle est à 0 toutes les bascules sont ramenées à 0 recompte immédiatement à partir de l'état 0 0 0

Les entrées de la porte NAND sont les sorties des bascules A et C de tel sorte que cette sortie passe à 0 lorsque A=C=1, cette condition est vérifiée quand le compteur passe de l'état 100 à 101 (i.e. à la 5^{eme} impulsion). Le niveau bas de la porte NAND place immédiatement le compteur à l'état 000. Dès que les bascules sont mises à 0 la sortie de la porte NAND revient à 1 puisque la condition A=C=1 n'existe plus

Remarque

Bien que le compteur passe très brièvement par l'état par l'état 101 (Quelques nanosecondes) avant sont recyclage à 000 on dit que c'est un compteur modulo 5 car son cycle de comptage est répété toutes les 5 combinaisons

Pour construire un compteur qui débute de l'état 0 et qui possède un modulo x on doit suivre les trois étapes suivantes :

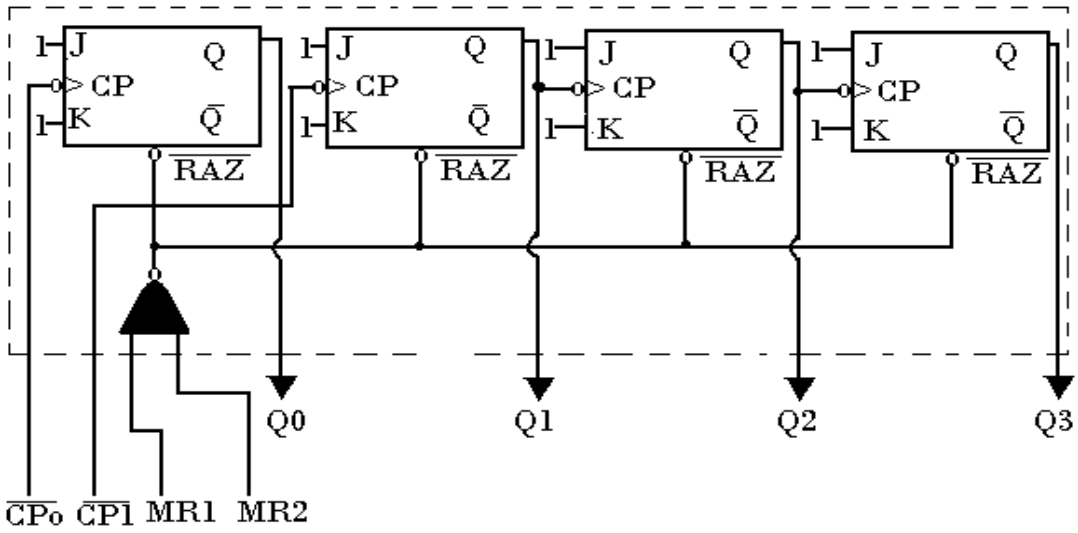
1- Trouver le plus petit nombre de bascule tel que : $2^N \geq X$ et raccorder ces bascules de manière à construire un compteur

Si 2^N = X les étapes 2 et 3 sont ignorés

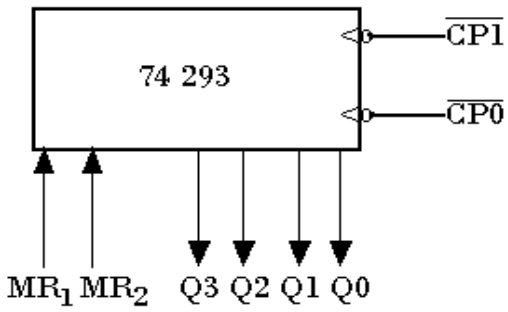
2- Connecter la sortie de la porte NAND en entrée asynchrone de remise à 0 de toutes les bascules
3- Déterminer quelles bascules sont à l'état 1quand le nombre binaire =X et raccorder alors les sorties normales des bascules ainsi déterminées au entrés de la portes NAND

IV.3. Les compteurs asynchrones en Circuit Intégré (CI)

Il existe plusieurs circuits qui intègrent les compteurs l'un d'eux est le boîtier 74293 (ou le 7493). Ce C.I intègre présente 4 bascules JK et une porte NAND connectée de la manière suivante.



Symbole simplifié:



CP : Entrée de l'horloge active au front descendant
Q : Sortie de Bascule
MR : Remise à 0

Figure V.10 : Compteur Asynchrone en CI : 74293

IV.4. Décompteurs asynchrone :

Tout les compteurs étudiés jusqu'à présent comptent progressivement à partir de 0, c'est donc des compteurs progressives, il est relativement facile de réaliser des décompteurs asynchrones qui partent d'un nombre maximal pour arriver jusqu'à 0, et qui présente un câblage réalisé de la manière suivante:

- Les bascules doivent réagir au front descendant et monté en trigger.
- Le signal d'horloge est appliqué à la première bascule.
- La sortie complémentée de chaque bascule est appliquée à l'entrée d'horloge de la bascule suivante.

Les sorties des bascules constituent directement les sorties du décompteur.

Décompteurs modulo 8 à Bascules JK

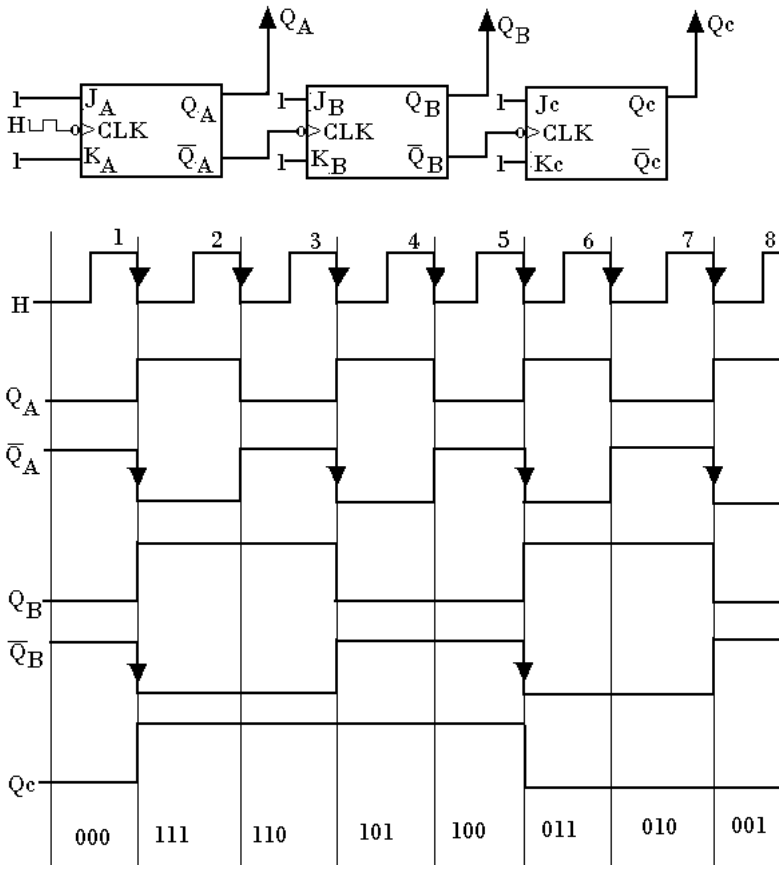


Figure V.11 : Décompteur Asynchrone Modulo 8

A, B et C représente les états de sorties des bascules du décompteur. On remarque que la bascule A change d'état a chaque décrémentation comme dans le cas d'un compteur progressif. La bascule B commute chaque foi que A passe de 0 à 1, et la bascule C commute chaque fois que B passe de 0 à 1. Ainsi dans un décompteur chaque bascule sauf la première passe à l'état opposé quand la bascule qui la précède effectue la transition de 0 à 1. Si les

bascules sont déclenchées par un front descendant du signal d'horloge il faut placer un inverseur avant chaque entrée CLK

IV.5. Les compteurs Synchrones

On utilise les compteurs synchrones pour résoudre le problème de retard de propagation des compteurs asynchrones. Dans ce type de compteurs toutes les bascules sont déclenchées par l'horloge au même moment.

Etant données que les impulsions d'horloges sont appliquées à toutes les bascules ; il doit y avoir un certain mécanisme qui indique quand une impulsion d'horloge doit faire commuter une bascule ou la laisser dans le même état

- On réalise un tel mécanisme en utilisant les entrées J et K (ou D)

IV.5.1. Connexion d'un compteur synchrone

Dans les compteurs synchrones toutes les bascules sont déclenchées par l'horloge au même instant, avant chaque impulsion d'horloge, chacune des entrées J et K des bascules des compteurs doit se trouver dans le niveau approprié qui assure le passage de chaque bascule dans le bon état.

Exemple

Etat Présent			*Etat Suivant*		
C	*B*	*A*	*C*	*B*	*A*
1	*0*	*1*	*1*	*1*	*0*

A l'arriver du front déclencheurs les entrées J et K doivent se trouver au bon niveau pour que la bascule A passe de 1 à 0 et la bascule B passe de 0 à 1et pour que la bascule C ne change pas d'état. Ceci est réalisé à l'aide de la table des états de la bascule J, K

Transition à la sortie	Etat précédent (Q_n)	Etat Suivant (Q_{n+1})	J	K
$0 \rightarrow 0$	0	0	0	Ø
$0 \rightarrow 1$	0	1	1	Ø
$1 \rightarrow 1$	1	1	Ø	Ø
$1 \rightarrow 0$	1	0	Ø	1

JK / Q	00	01	11	10
0	0	0	1	1
0	1	0	0	1

Table de KARNAUGH de la bascule JK Q_n : Etat présent, Q_{n-1} : suivant.

IV.5.2. Procédure de conception :

Etape 1: Déterminer le nombre de bits (nombre de bascules)

Déterminer le nombre de bits voulu et la séquence de comptage

Exemple : Compteur à 3 bits qui se compte selon la séquence suivante

C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

→ 3 Bits
→ Compteur modulo 5

Etape 2:

A partir de cette table de séquence, construire une table pour énumérer tout les états présents et leurs états suivant et indiquer dans des colonnes supplémentaires les niveaux qui doivent se trouver sur les entrées J et K pour que la transition se passe vers l'état suivant. Dans notre exemple on utilise trois bascules C, B, A ayant chacune des entrées J et K par conséquent il faut ajouter 6 colonnes dans la table correspondants aux états J_A, K_A, J_B, K_B, J_C , K_C ,

Etat présent			Etat suivant			J_C	K_C	J_B	K_B	J_A	K_A
C	B	A	C	B	A						
0	0	0	0	0	1	0	Ø	0	Ø	1	Ø
0	0	1	0	1	0	0	Ø	1	Ø	Ø	1
0	1	0	0	1	1	0	Ø	Ø	0	1	Ø
1	1	1	1	0	0	1	Ø	Ø	1	Ø	1
1	0	0	1	0	0	Ø	1	0	Ø	0	Ø
1	0	1	1	0	1	Ø	Ø	0	Ø	1	Ø

Etape 3:

Etablir à partir de cette table les équations de J et K des bascules en fonction de Q_A, Q_B et Q_C

C \ BA	00	01	11	10
0	0	0	1	0
0	Ø	Ø	Ø	Ø

$J_C = AB$

C \ BA	00	01	11	10
0	Ø	Ø	Ø	Ø
0	1	Ø	Ø	Ø

$K_C = 1$

C \ BA	00	01	11	10
0	0	1	Ø	Ø
0	0	Ø	Ø	Ø

$J_B = A$

C \ BA	00	01	11	10
0	Ø	Ø	1	0
0	Ø	Ø	Ø	Ø

$K_B = A$

C \ BA	00	01	11	10
0	1	Ø	Ø	1
0	0	Ø	Ø	Ø

$J_A = \overline{C}$

C \ BA	00	01	11	10
0	Ø	1	1	Ø
0	Ø	Ø	Ø	Ø

$K_A = 1$

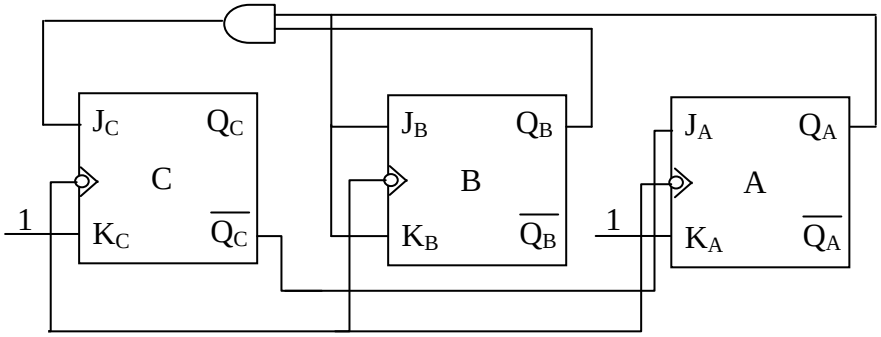


Figure V.12 Compteur Synchrone modulo 5

Réalisation d'un compteur synchrone à l'aide des Bascules D

On sait que l'information présentée en entrée d'une bascule D est transférée en sortie après l'application d'une impulsion d'horloge

Exemple : Compteur Synchrone modulo 10 :

Etat Présent				*Etat Suivant*				D_D	D_C	D_B	D_A
D	C	B	A	D	C	B	A				
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0

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

$D_D = ABC+\overline{A}D$

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

$D_C = \overline{B}C+\overline{A}C+AB\overline{C}$

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

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

$D_B = \overline{A}B+A\overline{B}\overline{D}$

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

$D_A = \overline{A}$

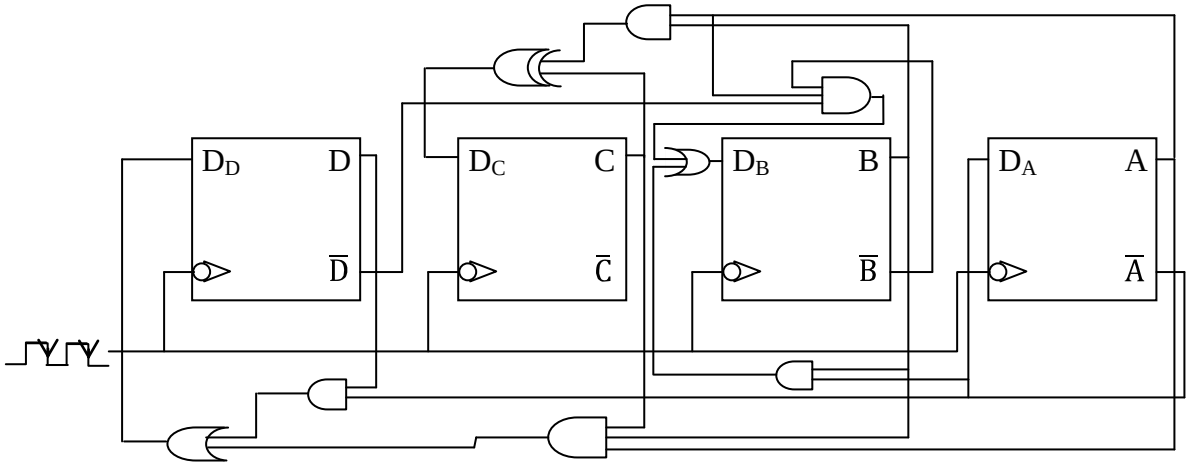


Figure V.13 Compteur Synchrone à l'aide des Bascules D

V.6. Compteur Circulaires

Il est possible d'utiliser des registres à décalage pour réaliser différents types de compteurs. Tous les compteurs circulaires ont recours à la rétraction i.e. au retro couplage de la sortie de la dernière bascule du registre sur l'entrée de la 1^{ere} Bascule, les compteurs à décalage les plus courant sont les compteurs en anneau et les compteurs Johnson.

V.6.1. Compteur en anneau

C'est un registre à décalage dans lequel la valeur de la dernière bascule est ramenée à l'entrée de la première bascule comme le montre la figure suivante :

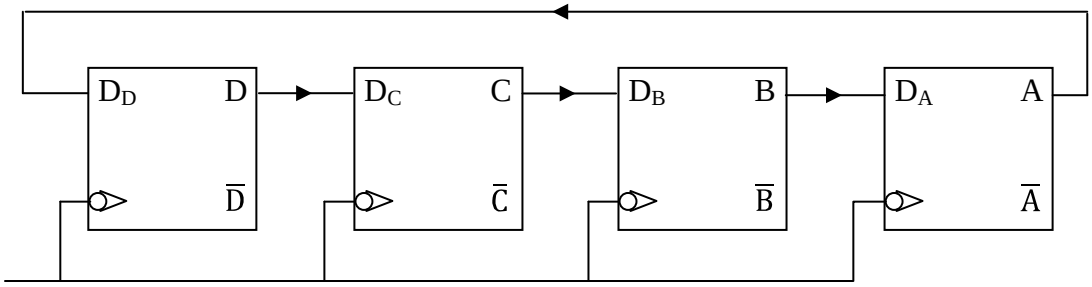


Figure V.14 Compteur en anneau

Q_3	Q_2	Q_1	Q_0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

Remarque :

- Un compteur en anneau ne fonctionne correctement que si au départ une de ses bascules est à l'état 1 et les autres bascules sont à l'état 0, il est donc nécessaire d'appliquer momentanément une impulsion à l'entrée RESET d'une bascule et CLEAR des autres bascules

V.6.2. Compteur Johnson

En modifiant le compteur en anneau il est possible de réaliser un autre type de compteur circulaire présentant certaines propriétés différentes

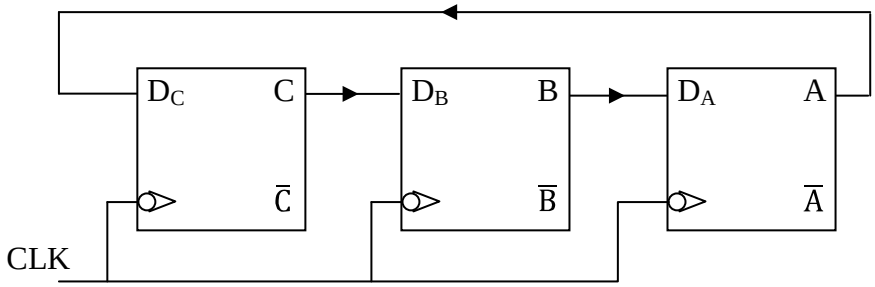


Figure V.15 : Compteur Johnson modulo 6

Le modulo d'un compteur Johnson (Notre cas modulo 6) est toujours égal à deux fois le nombre de bascules, les compteurs Johnson sont des solutions intermédiaires entre les compteurs en anneau et les compteurs binaires.

Contrairement aux compteurs en anneau on doit leurs associer des circuits de décodage qui sont toutes fois moins complexes que ceux des compteurs binaires

Impulsion	Q_2	Q_1	Q_0
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1
6	0	0	0

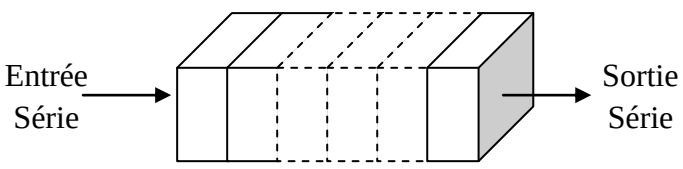
V.7. Les Registres à Décalage (Shift Registres)

V.7.1. Définition

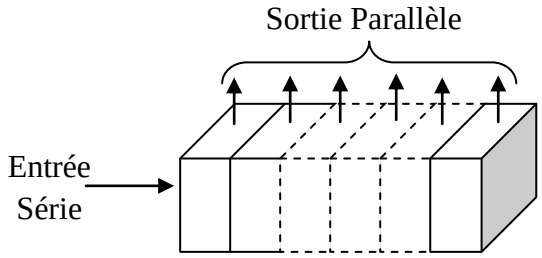
Lorsqu'on relit un groupe de bascules pour enregistrer une information on crée un registre, ce dernier accepte des données pour les mémoriser et les restituer quand il est nécessaire. Ces données peuvent être décalées de droite à gauche et inversement.

Les nombreux types de registre qui existent peuvent être classés selon la méthode d'écriture des données dans le registre et de lecture. Ces différents types sont les suivants :

Entrée Série / Sortie Série (Ecriture Série / Lecture Série)

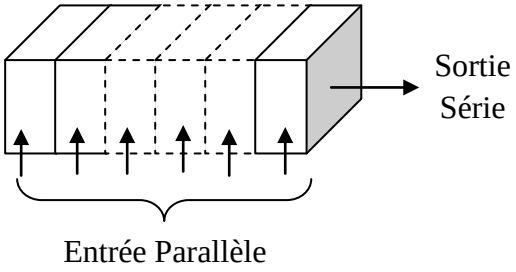


Entrée Série / Sortie Parallèle (Ecriture Série / Lecture Parallèle)

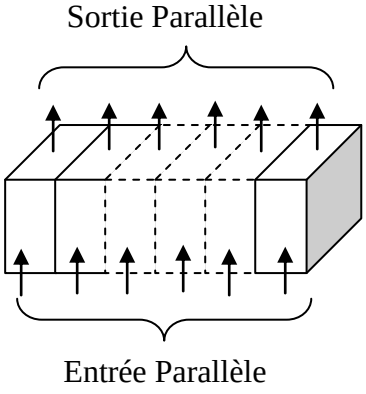


N Bascules : mots à n bits

Entrée Parallèle / Sortie Série (Ecriture Parallèle / Lecture Série)



Entrée Parallèle / Sortie Série (Ecriture Parallèle / Lecture Série)



V.7.2. Réalisation d'un registre à décalage à l'aide des bascules JK

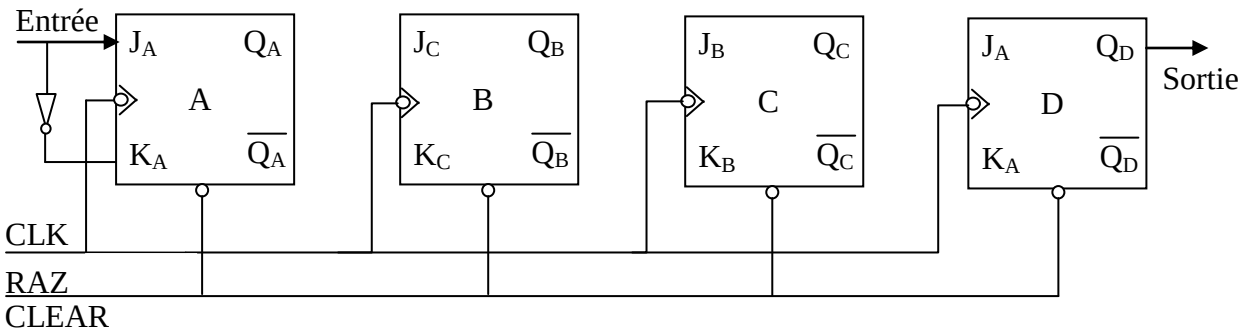


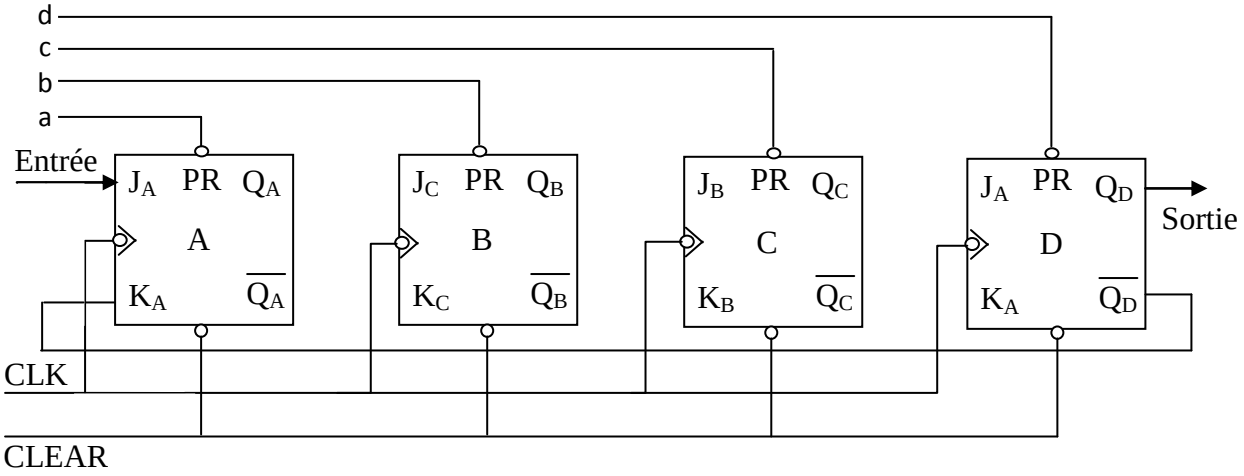
Figure V.16 :Registre à décalage à l'aide des bascules JK

CLEAR	CLK	A	B	C	D
Low(0)	X	0	0	0	0
1	$1^{er}\uparrow$	1	0	0	0
	$2^{em}\uparrow$	1	1	0	0
	$3^{em}\uparrow$	1	0	1	0
	$4^{em}\uparrow$	1	1	0	1
	$5^{em}\uparrow$	1	0	1	1
	$6^{em}\uparrow$	0	0	1	0
	$7^{em}\uparrow$	0	0	0	1
	$8^{em}\uparrow$	0	0	0	0

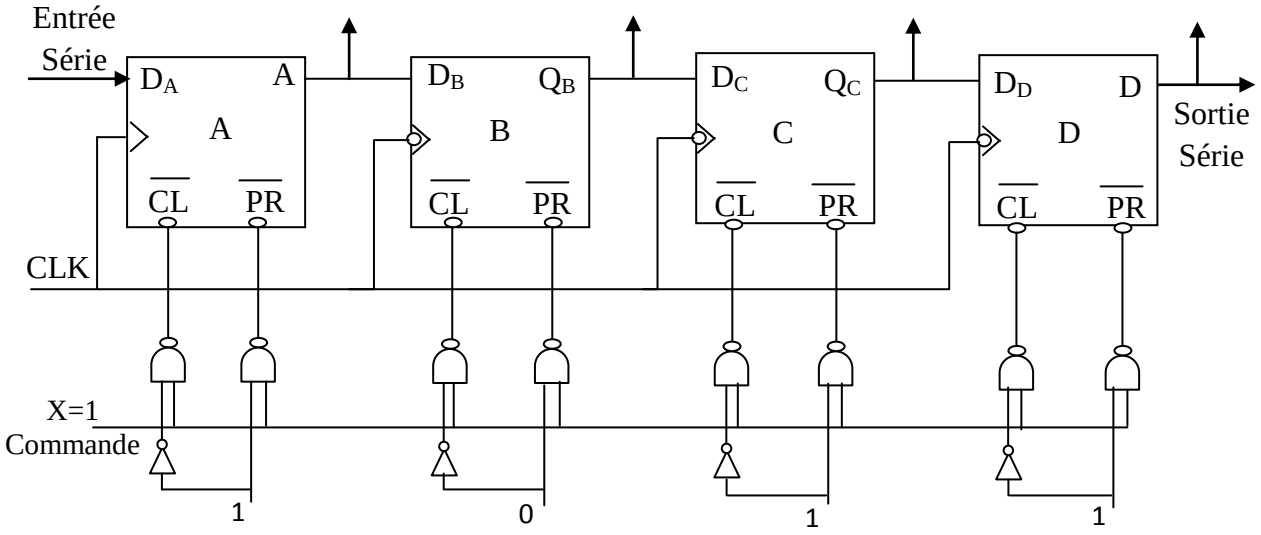
Un registre est une mémoire permettant de stocker des mots de plusieurs bits, pour cela on utilise autant de bascule que de bits à mémoriser,

Par exemple pour mémoriser 1 mot de 4 bits on utilise le registre suivant :

NB: Pour ne pas perdre les informations stockées sur le registre on le boucle sur lui-même



CLEAR	CLK	A	B	C	D
Low(0)	X	a	b	c	d
1	↑	d	a	b	c
	↑	c	d	a	b
	↑	b	c	d	a
	↑	a	b	a	d



Si X=1 les sorties de portes dépondent des entrées parallèles, cet état est inscrit instantanément sur les entrées des bascules grâce aux entrées asynchrones CL et PR. Dans le cas ou le signal X=0 toutes les sorties des portes NAND sont à 1 ce qui rend les entrées asynchrones CL et PR inactive et commence alors l'opération de décalage en fonction de CLK

V.7.3. Registre à décalage de droite à gauche

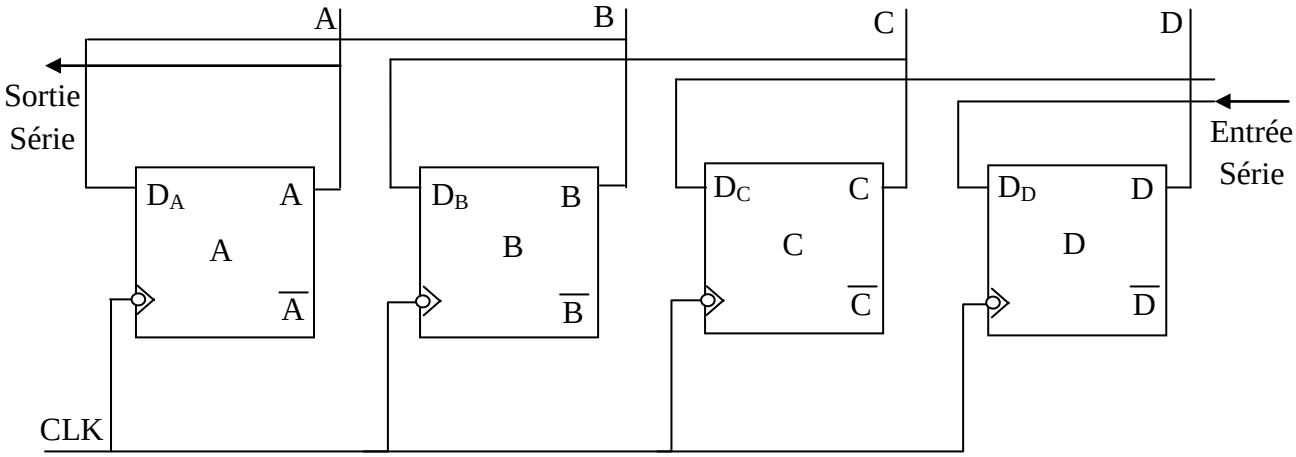


Figure V.17 : Registre à décalage de droite à gauche

CLK	A	B	C	D
$1^{er}\uparrow$	0	0	0	1
$2^{em}\uparrow$	0	0	1	0
$3^{em}\uparrow$	0	1	0	1
$4^{em}\uparrow$	1	0	1	1
$5^{em}\uparrow$	0	1	1	0
$6^{em}\uparrow$	1	1	0	0
$7^{em}\uparrow$	1	0	0	0

V.7.4. Registre à deux sens de lecture

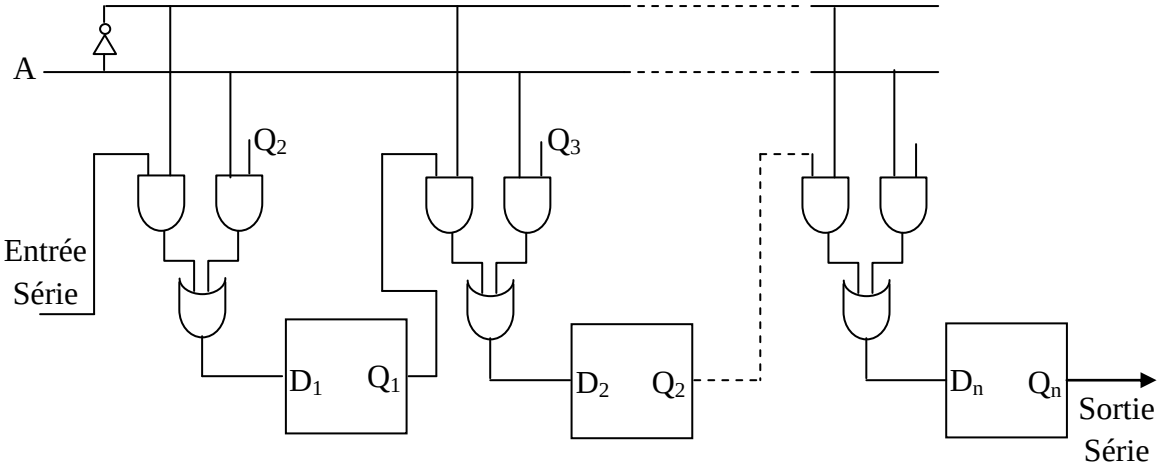


Figure V.18 :Registre à deux sens de lectures

$D_i = \overline{A}\, Q_{i-1} + A\, Q_{i+1}$

$A = 0 \rightarrow D_i = Q_{i-1}$ Décalage à droite

$A = 1 \rightarrow D_i = Q_{i+1}$ Décalage à gauche

V.7.5. Registre à décalage universel :

Un registre à décalage universel comporte toutes les entrées et les sorties possibles (Parallèles ou séries) avec tout les modes de fonctionnement

Exemple :

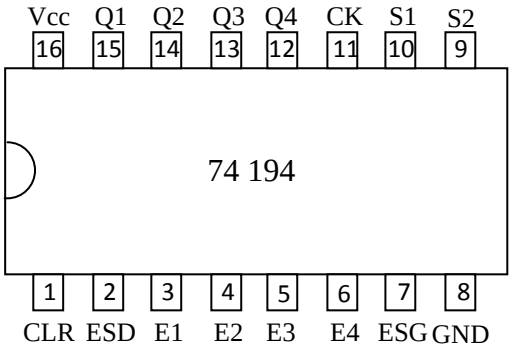


Figure V.19 : Brochage du circuit 74194

Entrées										Sorties			
CLEAR	Mode		CLK	Série		Parallèle				Q_A	Q_B	Q_C	Q_D
	S_1	S_0		Gauche	Droite	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	H	H	↑	X	X	A	b	c	d	a	b	c	d
H	L	H	↑	X	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	L