UNIVERSITE CHEIKHANTA DIOP DE DAKAR

FACULTE DES SCIENCES ET TECHNIQUES

Polycopie de Cours d'électronique numérique

DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

Dr Alassane TRAORE

Chapitre 1

SYSTEMES DE NUMERATION ET CODAGE

INTRODUCTION

Les machines qui traitent l'information, dont les ordinateurs en particulier, sont constituées de circuits numériques qui ne possèdent que deux états électriques stables ; représentées par les valeurs binaires 0 et 1.

En informatique, les informations traitées ne sont toujours pas numériques, il s'agit souvent d'informations exprimées sous forme de texte : caractères alphanumériques 0,1,…..9,a ,b ,…….z et caractères spéciaux. Pour pouvoir traiter et manipuler ces informations dans la machine, une codification sous forme binaire est nécessaire.

Nous présentons dans ce chapitre les différents codes ainsi que les opérations arithmétiques de base nécessaires. Nous introduisons tout d'abord les systèmes de numération puis le codage des nombres ainsi que les opérations de base dans les ordinateurs.

1.1 Les systèmes de numération

1.1.1 Définition

La définition d'un **système pondéré** repose sur les trois notions de **: base** du système**, de digit** du système et du **poids** du digit selon son rang.

La base d'un système est un nombre entier quelconque soit B.

Les **digits** d'un système sont des caractères tous différents représentant chacun un élément de la base : α ; β ;γ ;∂ .

L'écriture d'un nombre consiste à associer plusieurs digits dans un ordre déterminé.

Exemple : N= $\beta\ \partial\ \gamma\ \alpha$

Chaque digit intervient avec un poids différent selon son rang .

Ce poids est de B^0 pour le 1er digit ; B^1 pour le 2éme digit ; B^2 pour le 3éme digit; ……B^{n-1} pour le digit de rang n.

Le nombre N exprimé dans le système de base B vaut :

N= $\beta * B^3 + \partial * B^2 + \gamma * B^1 + \alpha * B^0$

Dans un système de numération de base b ; tout nombre est représenté par une suite de chiffres allant de 0 à b-1. les chiffres sont appelés des digits , la position de chacun représente une puissance entière (positive ou négative) de la base b ; la place occupée par le digit dans un nombre représente son rang.

Exemple :

$$N_b = a_n b^n + a_{n-1} b^{n-1} + ...a_0 b^0 + a_{-1} b^{-1} + ... + a_{-n} b^{-n}$$

on a : b^i (i=n)…….0 (i=-n) : sont les puissances successives de la base b .

b=base. A_i :digit

$a_{n-1} b^{n-1}$: occupe le rang n-1 dans le nombre N_b

Remarque :

La valeur du chiffre, appelée pondération est multipliée par b chaque fois que le rang augmente d'une unité.

Pondération = digit *b^i

Définitions :

-Base : c'est le nombre de symboles distincts à partie desquels on peut réaliser n'importe quelle quantité, ces symboles sont représentés par des chiffres ou des lettres.

-Nombre : représentation d'une information dans un système de numération par l'association de chiffres. Exemple : 2004 : association de chiffres 2.0.4.

-Digit : mot anglais désignant un chiffre ou une lettre quelconque soit la base.

-Bit : mot anglais désignant un chiffre binaire : représente 1 ou 0 en numération binaire.

1.1.2 Le système décimal

La base du système décimal est 10 ; il y'a 10 digits 0, 1, 2, 3, 4, 5, 6, 7, 8,9 qui sont dans ce cas Des chiffres décimaux habituels.

Symboles utilisés : Les dix symboles utilisés sont les chiffres allant de 0 à 9 .Ils sont appelés **DIGITS**.
Exemple : soit le nombre 213 nous obtenons le tableau suivant :

Digits	2	1	3
Rangs	2	1	0
Puissance	10^2	10^1	10^0
Pondération	$2*10^2$	$1*10^1$	$3*10^0$

La somme des pondérations donne :

Σ Pondérations = $\mathbf{2*10^2 + 1*10^1 + 3*10^0} = 200 + 10 + 3 = 213_{(10)}$

Généralisons l écriture des nombres dans le système décimal.

$N_{(10)} = a_{n-1}.a_{n-2}.a_{n-3}.....a_2.a_1.a_0 \qquad a_i \in \{0\,à\,9\}$

La somme des pondérations redonne le nombre initial.

$N_{(10)} = a_{n-1}.10^{n-1} + a_{n-2}.10^{n-2} + \ .....a_2.10^2 + a_1 10^1 + a_0.10^0$

Exemple1 : le nombre 2003 exprimé en décimal signifie :

$2003=2*10^3 +0*10^2 +0*10^1 +1*10^0 = 2000+0+0+3=2003.$

Digits (U_n)	2	0	0	3
Rang (n)	3	2	1	0
Puissances	10^3	10^2	10^1	10
Pondérations	2000	0	0	3

L'addition des pondérations redonne le nombre initial : 2000+3=2003.

Exemple2 : le nombre 493,75 exprimé en décimal signifie :

493.75=400+90+3+0.7+0.05.

Digits (U_n)	4	9	3	7	5
Rang (n)	2	1	0	-1	-2
Puissances	10^2	10^1	10^0	10^{-1}	10^{-2}
Pondérations	400	90	3	0.7	0.05

La somme des pondérations donne : 400+90+3+0.7+0.05=493.75

1.1.3 Le système binaire

Dans ce système ; la base B vaut 2 et il y'a 02 digits 0 et 1 qu'on appelle dans ce cas des Bits (binary digit). Ce système de numération est le plus utilisé dans les calculateurs numériques.

Exemple1 : soit le nombre binaires 1011(2)

Bits	1	0	1	1
Rangs	3	2	1	0
Puissance	2^3	2^2	2^1	2^0
Pondération	$1*2^3$	$0*2^2$	$1*2^1$	$0*2^0$

La somme des pondérations donne :

Σ Pondérations = $\mathbf{1*2^3 + 0*2^2 + 1*2^1 + 1*2^0} = 8 + 0 + 2 + 1 = 11_{(10)}$.

Exemple2 :

le nombre N1= 1011 exprimé en binaire signifie :

N1=1011=$1*2^3+0*2^2+1*2^1+1*2^0$

=8+0+2+1=11

D'où $(1011)_2=(11)_{10}$.

Généralisation:

Par notation on écrit un nombre binaire A.

$A_{(2)} = a_{n-1}.a_{n-2}.a_{n-3}.....a_2.a_1.a_0 \qquad a_i \in \{0 \grave{a} 1\}$

La somme des pondérations donne l'équivalent décimal du nombre binaire considéré :

$A_{(10)} = a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + \ .....a_2.2^2 + a_1 2^1 + a_0.2^0 \qquad a_i \in \{0 \grave{a} 1\}$

Exemple 3:

le nombre N2= (111.11) exprimé en binaire signifie :

Bits	1	1	1	1	1
Rang (n)	2	1	0	-1	-2
Puissances	2^2	2^1	2^0	2^{-1}	2^{-2}
Pondérations	4	2	1	0.5	0.25

L'addition des pondérations donne l'équivalent décimal du nombre binaire considéré.

Le nombre N2= 4+2+1+0.5+0.25=7.75

D'où $(N2)_2=(111.11)_2=(7.75)_{10}$.

1.1.4 Le système octal

Certains calculateurs utilisent ce système de numération dont la base est 8 et il y'a 8 digits : 0,1,2,3,4,5,6,7 .

Exemple : le nombre 547 exprimé en octal signifie :

$547 = 5*8^2+4*8^1+7*8^0$.

$= 5*64+4*8+7*1=320+32+7=359$

d'où $(547)_8=(359)_{10}$

1.1.5 Le système hexadécimal

Dans ce système ; la base B vaut 16 ; et il y'a 16 digits : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Les digits de 0 à 9 sont les chiffres du système décimal et les digits de 10 à 15 sont les 06 premières lettres de l'alphabet.

Exemple

Le nombre 3BA2 exprimé en hexadécimal signifie :

Digits (chiffres)	3	B	A	2
Rang (n)	3	2	1	0
Puissances	16^3	16^2	16^1	16^0
Pondérations	12288	2816	160	2

L'addition des pondérations donne l'équivalent décimal du nombre hexadécimal considéré :

12288+2816+160+2=15266.

D'où : $(3BA2)_{16} = 15266$.

1.2 Changement de base

Il y'a trois types de conversion :

Conversion du système décimal en un autre système: cette opération s'appelle **le codage.**

Conversion d'un système autre que le décimal en un système décimal: cette opération s'appelle **le décodage.**

Conversion entre deux systèmes non décimaux: cette opération s'appelle **le transcodage.**

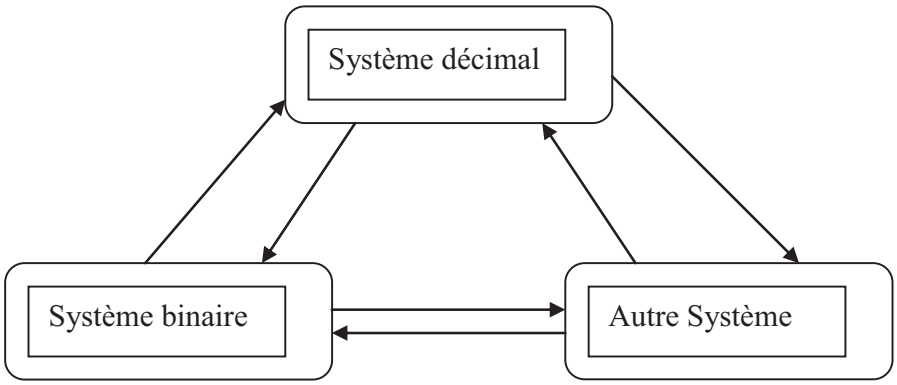


Figure 1. Schéma général de passage d'un système à un autre

Le codage d'un nombre décimal est la conversion de celui-ci du système décimal vers un système de base B, tels que $B \neq 10$.

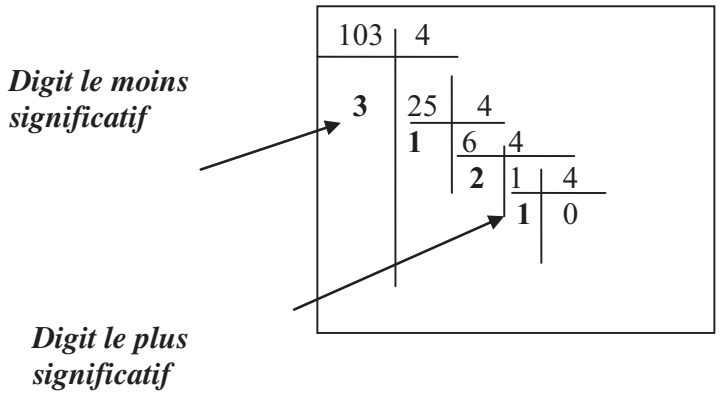
Méthode :

L'opération de codage est obtenue par la division successive d'un nombre décimal par la base du système B jusqu'au moment où le quotient devient nul.

Le nombre cherché sera obtenu en regroupant tous les restes successifs de droite à gauche.

Exemple : Ecrire en base 4 le nombre décimal 103.
Après une division successive du nombre 103 par 4, nous regroupons les restes de la division comme présenté sur la figure. Le dernier reste représente le digit le plus significatif.

$103_{(10)} = 1213_{(4)}$



1.2.1 Décimal - binaire

L'équivalent binaire d'un nombre décimal s'obtient en divisant successivement le nombre décimal par 2 jusqu'au moment ou le quotient vaut 1.

Ce dernier quotient et les restes des divisions successives regroupés de bas en haut forment le nombre binaire recherché.

Exemples

L'équivalent de $(33)_{10}$ en binaire est :

Démarche : chercher les multiples des puissances successives de la base 2 que contient ce nombre :

33=16*2+1.
16=8*2+0.
8=4*2+0.
4=2*2+0.
2=1*2+0.
1=0*2+1.

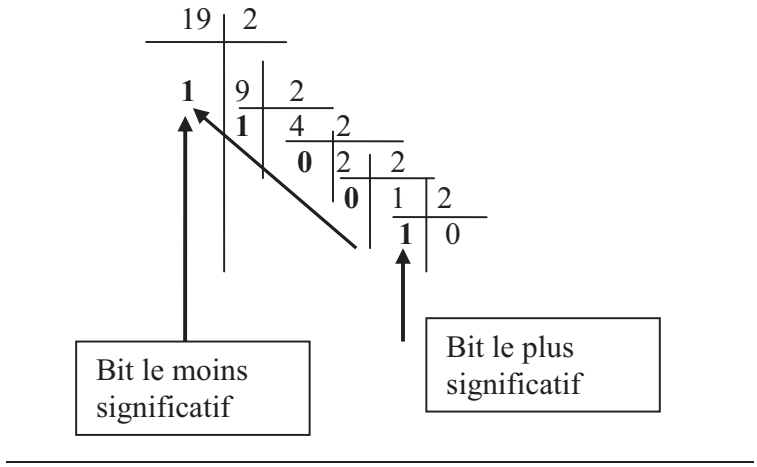
D'où $(33)_{10} = (100001)_2$.

*L'équivalent de 13 en binaire est :

13=6*2+1.
6=3*2+0
3=1*2+1.
1=0*2+1.

D'où $(13)_{10} = (1101)_2$.

L'équivalent de $(19)_{10}$ en binaire est :



Le dernier reste ainsi que les restes successifs représentent le chiffre binaire correspondant.

19=($10011)_2$

Remarque :

Dans le cas de nombres fractionnaires, on multiplie la partie fractionnaire par 2 :

*si après la multiplication, un 1 apparaît à gauche de la virgule, on ajoute 1 à la fraction binaire en formation.

*si après multiplication, c'est un 0 qui apparaît ; on ajoute un 0.

Exemple :

Cherchons l'équivalent binaire du nombre décimal 0.4375.

0.4375*2=0.875 0.0
0.875*2=1.750 0.01
0.750*2=1.5 0.011
0.5*2=1.0 0.0111
0.00*2=0.000 0.01110

d'où $(0.4375)_{10} = (0.01110)_2$.

1.2.2 Décimal – octal

La conversion se fait en divisant le nombre par 8 ; le premier reste sera le chiffre le moins significatif (CLMS) ou LSB (least significant bit).

Exemple :

Trouver l'équivalent octal du nombre $(456)_{10}$

Pour passer de la base 10 à une base B quelconque ; on divise le nombre par la base jusqu'à avoir un quotient nul.

Démarche à suivre :

456 = 57*8+0.
57=7*8+1.
7=0*8+7.

D'où $(456)_{10} = (710)_8$.

1.2.3 Décimal – hexadécimal

La conversion se fait suivant le même principe que la conversion décimale – octale ; la division sera par 16.

Exemple1 :

$72 = 4*16+8.$

$4 = 0*16+4.$

D'où $(72)_{10} = (48)_{16}$.

Exemple2 :

$45 = 2*16+13.$

$2 = 0*16+2.$

$(45)_{10} = (2D)_{16}$

Exemple 3 :

$16 = 1*16+0$

$1 = 0*16+1.$

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

1.2.4 Conversion Binaire – décimal / octal – décimal / hexadécimal – décimal

Il suffit de multiplier chaque bit ou chiffre ou digit par son poids correspondant puis de faire la somme des résultats obtenus.

Exemple 1 :

$(1011)_2 = (?)_{10}$

$(N)_{10} = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0.$

$=8+2+1=11$

$(1011)_2 = (11)_{10}.$

Exemple2 :

$(234)_8 = (?)_{10}.$

$(234)_8 = 2*8^2 + 3*8^1 + 4*8^0.$

$=128+24+4=156.$

$(234)_8 = (156)_{10}.$

Exemple 3 :

$(3C7)_{16} = (N)_{10}$

$(N)_{10} = 3*16^2 + 12*16^1 + 7*16^0.$

$=769+192+7=967.$

$(3C7)_{16} = (967)_{10}$.

1.3 Autres méthodes de conversion

1.3.1 Passage octal à binaire et binaire - octal

On peut remarquer que la base du système octal 8 est égale à la puissance 3^{eme} de la base du système binaire 2

$$8 = 2^3$$

On peut faire correspondre à chaque «digit » d'un chiffre exprimé en octal un ensemble de 3 bits pondérés du mémé nombre exprimé en binaire par exemple :

$(427)_8 = (100)\ (010)\ (111) = (100\ 010\ 111)_2$

La conversion inverse ; binaire →octal, se fait de la mémé façon, en décomposant le nombre binaire par ensemble de 3 bits

$(11010\ 110)_2 = (011)\ (010)\ (110) = (326)_8$

1.3.2 Passage hexadécimal – binaire et binaire – hexadécimal

La base du système hexadécimal, 16, est égale à la puissance 4^{eme} de la base du système binaire , on fera correspondre à chaque digit d'un nombre hexadécimal 4 bits du nombre binaire correspondant

$16 = 2^4$

Exemple :

$(683)_{16} = (0110)\ (1011)\ (0011) = (011010110011)_2$

La conversion inverse ; binaire – hexadécimal, se fait se décomposant le nombre binaire par ensemble de 4 bits

$(11010\ 1110\ 1101)_2 = (0001)\ (1010)\ (1110)\ (1101) = (1AED\ )_{16}$

L'expression hexadécimale d'un nombre binaire est très utilisée pour interprètes des résultats fournis par un « microprocesseur »

Application :

Pour convertir un nombre décimal en binaire, il est plus avantageux de convertir ce nombre en octal puis convertir ce nombre en binaire

Exemple :

Convertir $(3947)_{10}$ en base 2 :

$3947 \leftarrow (3947)_{10} = (7553)_8 = (111\ 101\ 101\ 011)_2$

Exercices :

Convertir $(894)_{10}$ en base 8

$(894)_{10} = (1576)_8$

Convertir $(101\ 101\ 100)_2$ en base 10 et en base 16

$101\ 101\ 100 = (16C\ )_{16} = 254+96+12 = 364$

Convertir $(1F4\ )_{16}$ en base 2 :

0001 1111 0100

donc $(1F4)_{16} = (1\ 1111\ 0100)_2$

Convertir $(3947)_{10}$ en base 2 :

$3947 \leftarrow (3947)_{10} = (7553)_8 = (111\ 101\ 101\ 011)_2$

1.4 Codage

Toute information en clair est constituée de 3 types de caractères :

-Des textes ensemble de mots eux mémé constitués de lettres.

-Des nombres représentés par des chiffres.

-Des symboles qui peuvent être de nature très diverse (signes de ponctuation ; opérateurs arithmétiques opérateurs logiques ; commandes auxiliaires ….)

On distingue deus catégories de codes :

-Les codes numériques qui permettent seulement le codage des nombres.

-Les codes alphanumériques qui permettent le codage d'une information quelconque (ensemble de lettre ; de chiffres et de symboles).

1.4.1Codes numériques

1.4.1 Code binaire naturel

Ce code est utilisé uniquement pour représenter les chiffres décimaux.

L'équivalent binaire d'un nombre décimal s'obtient en divisant successivement le nombre décimal par 2 .

Ce code est encore appelé code 8421.

N	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

1.4.2 Code binaire réfléchi ou Code GRAY

Ce code ne permet que la représentation des chiffres décimaux à chaque augmentation d'une unité du chiffre décimal. Un seul bit du nombre binaire équivalent change de valeur par rapport au nombre binaire précédent.

N	Binaire	Gray		
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1

On remarque que dans ce code ; chaque bit a une pondération de : $\pm (2^{n+1} - 1)$

n : étant le range du bit dans le nombre.

Le premier chiffre 1, rencontré en partant de la gauche est affecté du signe+ ; le deuxième chiffre 1 est affecté du signe- ; le 3^{eme} du signe + etc. …..

Exemple :

Soit le nombre 1011 en binaire réfléchi :

Chiffres	1	0	1	1	
Rang (n)	3	2	1	0	
Puissances	2^3	2^2	2^1	2^0	
Pondérations	$+(2^4-1)$		$-(2^2-1)$	$+(2^1-1)$	
Soit	+15		-3	+1	= 13

**Méthode de passage binaire gray :*

-Le bit de gauche du mot binaire reste inchangé.

-Additionner le bit de poids le plus significatif au bit suivant et reporter le résultat sans tenir compte de la retenue.

-Continuer l'addition de chaque bit avec le bit suivant jusqu'au bit du pois le plus faible.

Exemple :
Binaire : 1 0 1 1 0 1 0
Gray : 1 1 1 0 1 1 1
Le tableau suivant donne l'équivalent gray d'un nombre binaire tel que N≤15.

Décimal	Binaire naturel	Binaire réfléchi
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

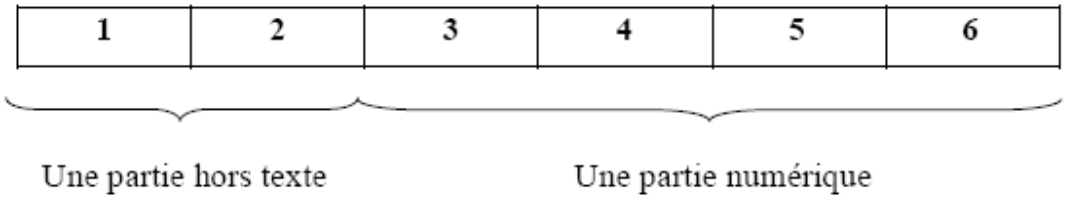
1.4.2 Codes alphanumériques

Les codes « alphanumériques » sont des codes destinés à la transmission d'information quelconques, ils ont donc à représenter au moins 36 caractères (10chiffres + 26 lettres).

Comme 36 est compris entre 2^5 et 2^6 , ils devront comporter au moins 6 bits (pour permettre la détection des erreurs, avec un bit de parité)

1.4.2.1 Code ISO

C'est une extension à 6 bits du code BCD, ce qui offre 64 combinaisons permettant de représenter les chiffres, les lettres et les signes particuliers.
La structure d'un nombre binaire dans ce code est constituée de deux parties :
Une partie hors texte constituée de deux bits de poids le plus fort.
Une partie numérique constituée des 4 bits restants.



				1	2	1	2	1	2	1	2
3	**4**	**5**	**6**	0	0	0	1	1	0	1	1
0	0	0	0			0				P	
0	0	0	1			1		A		Q	
0	0	1	0			2		B		R	
0	0	1	1			3		C		S	
0	1	0	0			4		D		T	
0	1	0	1			5		E		U	
0	1	1	0			6		F		V	
0	1	1	1			7		G		W	
1	0	0	0	(		8		H		X	
1	0	0	1	)		9		I		Y	
1	0	1	0	*		:		J		Z	
1	0	1	1	+		;		K			
1	1	0	0	,		<		L			
1	1	0	1	-		=		M			
1	1	1	0	.		>		N			
1	1	1	1	/				O			

1.4.2.2 Code ASCII : **(American standard code for information interchange)**

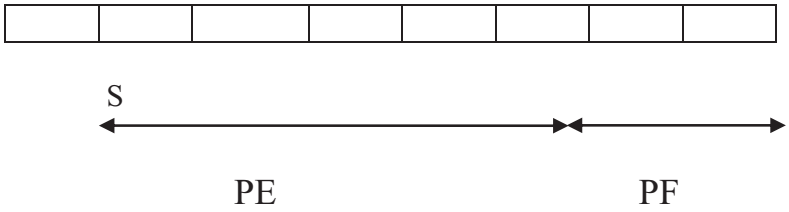
Dans ce genre de code on représente les lettres, les chiffres, les signes par des combinaisons binaires. Ce code est constitué de 8 bits (octet) dont un, celui de gauche, est en général un bit de parité. Ce code permet donc de représenter 256 caractères.

b7b6b5 / b4b3b2b1	000	001	010	011	100	101	110	111
0000	NUL	DLE	Espace	0	@	P	'	P
0001	SOH	DC_1	!	1	A	Q	a	
0010	STX	DC_2	"	2	B	R	b	
0011	ETX	DC_3	≠	3	C	S	c	
0100	EOT	DC_4	$	4	D	T	d	
0101	ENQ	NAK	%	5	E	U	e	
0110	ACK	SYN	&	6	F	V	f	
0111	BEL	ETB	'	7	G	W	g	
1000	BS	CAN	(	8	H	X	h	
1001	HT	EM	)	9	I	Y	i	
1010	LF	SUB	*	:	J	Z	j	
1011	VT	ESC	+	;	K	[	k	
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	≈
1111	SI	US	/	?	O	_	o	DEL

1.5 Représentation des nombres

1.5.1 Représentation des nombres en virgule fixe

Un nombre de m éléments binaire est partagé en deux parties dont l'une correspond aux valeurs entières et l'autre aux valeurs fractionnaires la position de la virgule fixe est fictive ; fixée par l'utilisateur.



Par convention :
Nombre > 0 : S = 0
Nombre < 0 : S = 1 S : Bit de signe

PE : partie entière : 5 bits
PF : partie fractionnaire : 2 bits Format du nombre .

Exemple1 :

Représenter sur PE = 4 bits : + 7.5 selon le format donné

S = 0 0 0111. 100

Exemple 2 :

-12.5 1 1100. 11

Exemple 3 :

+17.5 impossible selon le format.

1.5.2 Les différentes représentations des nombres dans les ordinateurs

Il y'a 3 représentations possibles pour réaliser les opérations de base (ADD ; Soust ;DIV ;Mult)

-Représentation en module et signe (MS)
- Représentation en complément à « 1 » → (C1)
- Représentation en complément à « 2 » → (C2)

a- Représentation en module et signe

N : nombre → Représente le nombre en signe et valeur absolu $\pm|N|$

Exemple :

Représenter le nombre binaire en module et signe.

PE = 5 Bits ;

PF = 2 Bits 1 01111. 10

N = - 15.50

b- Représentation en complément à"1" ou complément restreint C1

Cette représentation concerne les nombres négatifs

A = a_{n-1} a_{n-2} …….. a_1 a_0

$(A)_{c1} = \overline{a_{m-1}}\ \overline{a_{m-2}}.....\overline{a_1}\ \overline{a_2}$

Exemple :

A = -17

On écrit d'abord le nombre en module et signe

A = -17 → donner son complément à « 1 »

A = $(1\ \ 10001)_{MS}$ → $(1\ \ 01110)_{C1}$

Exemple1

Représenter : +12.5 en C1 tel que PE = 4 bits.

La représentation en C1 des nombres positifs c'est leur représentation en module et signe

A = +12.5 = $(0\ 1100\ ;\ 10)_{MS}$ = $(0\ 1100,10)_{C1}$

Exemple2

-31.75 en C1

Suivant le format -S = 1 bit.
-PE = 5 bits.
-PF = 2 bits.

A = -31.75 $(1\ \ 11111\ .\ 11)_{MS}$

$= (1 \quad 00000.\ 00)_{C1}$

Remarque: $A + A(C1) = 11\ldots\ldots1) = 2^n - 1$

b- Représentation des nombres en complément à « 2 »

Le complément vrai d'un nombre négatif est le complément à « 1 » de ce nombre auquel on ajoute un « 1 » au bit de poids faible.

$$(A)_{c2} = (A)_{c1} + 1.$$

Exemple

Représenter en (C2) le nombre A =-11

$(A)_{MS} = (1 \quad 1011)_{MS} = (1 \quad 0100)_{c1}$

$= (1 \quad 0101)_{c2}$

*Représenter A = -20.25

$(1 \quad 10100.01)_{MS} = (1 \quad 01011.10)_{c1}$

$(1 \quad 01011.11)_{c2}$

-2^{éme}methode

On peut représenter le complément à2 d'un nombre négatif en inversant tous les bits à partir du premier « 1 » représenté le + à droite et qu'il faut conserver :

Exemple :

A = -20.25

$(1 \quad 10100.01)_{MS}$

$(1 \quad 01011.11)_{c2}$

1.6 Arithmétique binaire : (Opérations dans le système binaire)

1.6.1 L'addition

L'addition des nombres binaires s'effectue de la même façon que l'addition des nombres décimaux en appliquant la table d'addition suivante :

A	B	$\sum$ Somme	C(retenue)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Règle N°5 : 1+1+1=1 et retenue C=1.

Exemple 1 :

Soit à additionner les deux nombres suivants : A=1100 et B=1010.

1 1 0 0 (12+10=22)

1 0 1 0

1 0 1 1 0

Exemple 2 : soit à additionner les 02 nombres binaires suivants : A=110 et B=111.

1 1 0
1 1 1 L'opération en décimal 7+6=13
1 1 0 1

Le symbole d'un demi additionneur est le suivant :

Figure 2. Schéma synoptique d'un demi additionneur (Semi adder en anglais)

Le demi additionneur est incapable de réaliser la 5^{éme} règle ; pour cela l'additionneur complet a été conçu pour prendre en considération la retenue intermédiaire

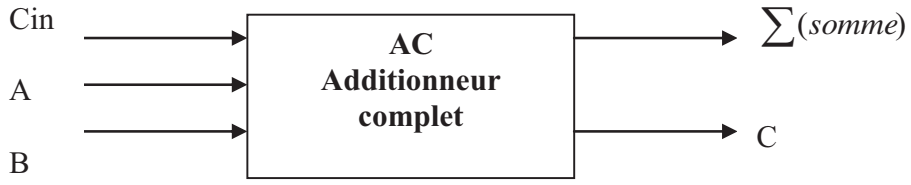


Figure 3. Schéma synoptique d'un additionneur complet (Full adder en Anglais)

Cin : retenue intermédiaire.
A , B : nombres à additionner .

Cette opération est possible en combinant deux demi-additionneurs comme présenté par la figure suivante . En pratique pour minimiser le nombre de composants, ou de portes dans un circuit intégré, un tel additionneur est réalisé directement.

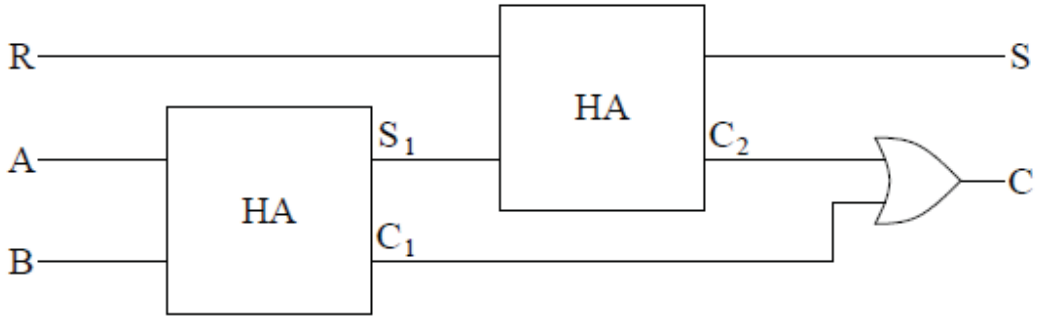


Figure 4. Réalisation d'un additionneur complet

L'étude complète d'un additionneur complet sera au présenté au chapitre suivant.

1.6.2 Soustraction

La soustraction s'effectue de la même façon en appliquant la table de soustraction :

Nombre binaire A	Nombre binaire B	D(différence)	C(retenue)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Exemple1 :

Soit à calculer la différence A-B tel que :
A=10110 ; B=1100

```
  1  0  1  1  0
 1
  0  1  1  0  0
  0  1  0  1  0
```

En décimal 22-12=10

Le circuit qui peut réaliser de simples soustractions est appelé demi_ soustracteur (DS).

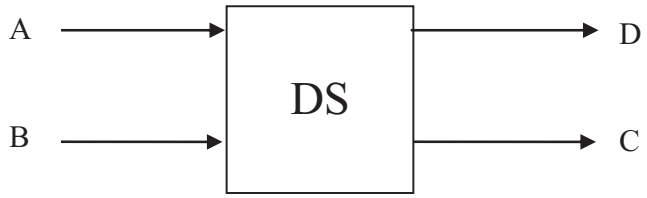


Figure 5. Schéma synoptique d'un demi soustracteur

Exemple 2 :

Calculer la différence A-B tel que :
A=100101 ; B=1010

```
0 0  0  1  0  1          en décimal  37-10=27
-
0   0  1  0  1  0
=
0 1  1  0  1  1
```

Cette soustraction a entraîné plusieurs retenues ; on fera appel donc à un soustracteur complet (SC) qui prend en considération les retenues intermédiaires tel que schématisé sur la fig 5.

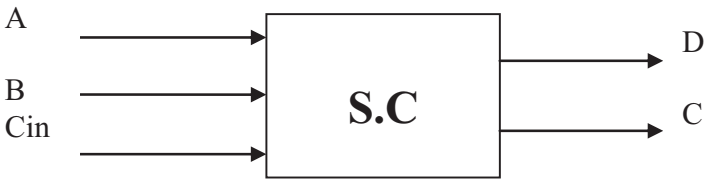


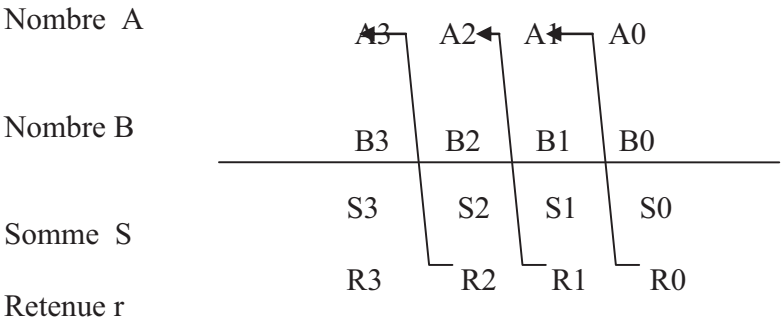
Figure 6. Symbole d'un soustracteur complet

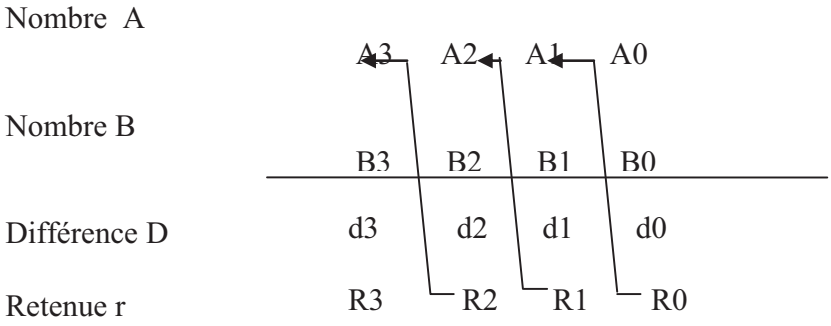
1.6.3 Quelques remarques

A- Addition et soustraction de nombres binaires non signés
soit à additionner le nombre A et le nombre B tel que :

S est la somme (S3 S2 S1 S0)

R est la retenue (R3 R2 R1 R0).





Étudions les conditions aux limites ou les cas de dépassement :
Exemple1 :

5	0 1 0 1	
+ 3	0 0 1 1	
=+8	1 0 0 0	et le résultat est en vraie grandeur (r=0)

Exemple 2 :

12		1	1	0	0	
- 6		0	1	1	0	
6		0	1	1	0	le résultat en vraie grandeur(r=0).

Exemple 3 :

12		1	1	0	0	
+ 9		1	0	0	1	
=21	1	0	1	0	1	dépassement supérieur : le format est Insuffisant.

Exemple 4 :

7		0	1	1	1	
- 10		1	0	1	0	
=	1	1	1	0	1	dépassement inférieur : anomalie.

Remarque

Si après une addition ou une soustraction :

a- La retenue terminale R=0 : le résultat est en vraie grandeur et dans la limite des n bits.

b- La retenue terminale R=1 :

- un dépassement supérieur pour l'addition : format insuffisant.
- Un dépassement inférieur pour la soustraction : nombre négatif ou anomalie.

B- Addition et soustraction de nombres signés en signe magnitude

L'addition ou la soustraction de 02 nombres binaires entiers signés en signe magnitude doit tenir compte des 03 paramètres suivants :

- du signe de chacun des deux nombres.
- Du type d'opération (addition ou soustraction).
- Du classement relatif de leurs valeurs absolues.

Exemple1 :

En décimal, l'opération suivante doit s'effectuer dans l'ordre suivant :

+5 + (-9) = 5 – 9 =- (9-5).

En binaire l'opération se traduit :

(00101) + (11001) = (+0101) + (- 1001)= - (1001- 0101) = -4

Dans ce cas l'addition se ramène à une soustraction en inversant les opérandes. le signal final dépend de la comparaison des valeurs absolues.

Exemple2 : +6 +(-10) = 6-10 = - (10-6)=-(4).

C- Addition et soustraction en complément à 2

L'avantage de la représentation en complément à 2 est que l'on ne s'occupe pas des signes lors des opérations arithmétiques. on additionne ou l'on soustrait les nombres, bit de signe compris . le résultat est obtenu en complément à 2 avec le signe correct.

+2 0 0 1 0 +3 0 0 1 1 =+5 0 1 0 1 ***bit signe=0> résultat correct***	-3 1 1 0 1 + -4 1 1 0 1 =-7 1 0 0 1 ***le résultat en c2= (1 111)en MS***
-7 1 0 0 1 - -3 1 1 0 1 =-4 1 1 0 0 ***le résultat en c2= (1 1 0 0)en MS***	-3 1 1 0 1 + +2 0 0 1 0 =-1 1 1 1 1 ***le résultat est en c2=(1 001)en MS***

Les cas d'overflow pouvant se présenter sont :

Cas d'overflow 1 +5 0 1 0 1 + +7 0 1 1 1 =+12 1 1 0 0	Cas d'overflow 2 +6 0 1 1 0 - -4 1 1 0 0 =+10 1 0 1 0
Cas d'overflow 3 -3 1 1 0 1 - +6 - 0 1 1 0 =-9 0 1 1 1	Cas d'overflow 4 -5 1 0 1 1 + -6 1 0 1 0 -11 0 1 0 1

On résume les cas de débordement comme suit : tel que :
SA ; SB ; signe de A et de B.
Op : type d'opération (0 pour l'addition et 1 pour la soustraction)
SR : signe du résultat (0 pour un nombre positif et 1 pour un nombre négatif)

1- SA=0 ; SB=0 ; OP=0(addition).
SR=1.

2- SA=0 ; SB=1 ; OP=1 (soustraction)
SR=1.

3- SA=1 ; SB=0 ; OP=1
SR=0.

4- SA=1 ; SB=1 ; OP=0
SR=0.

D- Addition et soustraction en virgule fixe

L'addition et la soustraction des nombres binaires en virgule fixe (VF) ainsi que les problèmes d'overflow sont exactement les mêmes comme pour des nombres signés en complément à 2.

+ 5.25	0	1 0 1. 0 1
+ 3.75	0	0 1 1. 1 1
= 9.00	1	0 0 1. 0 0

E- Addition et soustraction en BCD

Lors d'une addition de 02 nombres binaires codés en BCD ; il faut tenir compte :

1- dépassement de capacité de chaque quartet.
2- Retenue terminale de chaque quartet.

La correction des erreurs d'addition en BCD est réalisée en ajoutant le nombre 06 (011) en binaire au quartet pour lequel l'un de ces deux cas est détecté ([1] Et [2]).

Aucune correction 53 0 1 0 1 0 0 1 1 +42 0 1 0 0 0 0 1 0 =95 1 0 0 1 0 1 0 1	**Cas de correction [1]** 39 0 0 1 1 1 0 0 1 +24 0 0 1 0 0 1 0 0 63 0 1 0 1 1 1 0 1 + 0 1 1 0 0 1 1 0 0 0 1 1
Cas de correction [1] 84 1 0 0 0 0 1 0 0 +71 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 + 0 1 1 0 155 = 1 0 1 0 1 0 1 0 1	**Cas de correction [2]** 49 0 1 0 0 1 0 0 1 +18 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 + 0 1 1 0 67 = 0 1 1 0 0 1 1 1
Cas de correction [2] 95 1 0 0 1 0 1 0 1 +73 0 1 1 1 0 0 1 1 = 0 0 0 0 1 0 0 0 + 0 1 1 0 168 = 1 0 1 1 0 1 0 0 0	**Cas de correction [1]et [2]** 58 0 1 0 1 1 0 0 0 +79 0 1 1 1 1 0 0 1 = 1 1 0 1 0 0 0 1 + 0 1 1 0 0 1 1 0 =137= 1 0 0 1 1 0 1 1 1

Principe de la soustraction en BCD

Lors d'une soustraction en BCD ; il faut tenir compte du classement relatif de leurs valeurs absolues ; le signe final est le signe du plus grand nombre ; une correction est nécessaire lorsqu'on détecte une retenue terminale.

La correction consiste à soustraire par le nombre 6 (0110 en binaire) ; si une retenue terminale du dernier quartet apparaît, le résultat est faux parce que le classement des valeurs absolues n'a pas été respecté .

Exemple d'illustration :

Aucune correction	**Correction due à la retenue**
35 0 0 1 1 0 1 0 1 -14 0 0 0 1 0 1 0 0 =21 0 0 1 0 0 0 0 1	34 0 0 1 1 0 1 0 0 -15 0 0 0 1 0 1 0 1 - 0 1 1 0 19 = 0 0 0 1 1 0 0 1
Correction due à la retenue	**Correction due au dépassement**
32 0 0 1 1 0 0 1 0 -29 0 0 1 0 1 0 0 1 = 0 0 0 0 1 0 0 1 - 0 1 1 0 =03 = 0 0 0 0 0 0 1 1	39 0 0 1 1 1 0 0 1 - 42 0 1 0 0 0 0 1 0 = 1 1 1 1 0 1 1 1 - 0 1 1 0 -03 = 1 0 0 1 0 1 1 1 **le résultat est incorrect car le classement des valeurs n'a pas été respecté.**

1.6.4 Multiplication

La multiplication repose sur les règles suivantes :

0*0=0
0*1=0
1*0=0
1*1=1

Exemple :

(1101) * (11))= $(13)_{10}$ *$(3)_{10}$= ?

$$
\begin{array}{rcccccc}
 & & 1 & 1 & 0 & 1 & \\
 & & & & 1 & 1 & \_ \\
\hline
= & & 1 & 1 & 0 & 1 & \\
+ & & & & & & \\
 & {\scriptstyle 1}1 & 1 & 0 & 1 & & \\
\hline
= & 1\,0 & 0 & 1 & 1 & 1 &
\end{array}
$$

Le résultat est bien 39 en décimal

1.6.5 Division binaire

Elle repose sur les règles suivantes :

$$\left.\begin{array}{l} 0/0 \\ 1/0 \end{array}\right\} \textit{cas indéter} \min \textit{és}$$

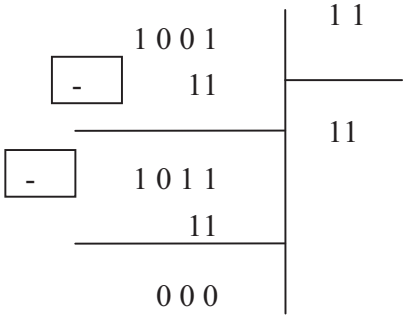
$$0/1 = 0$$

$$1/1 = 1$$

Exemple :

Effectuer la division suivante

1 0 0 0 | 1 1



Exercices d'application : Les systèmes de numération

Partie I : Codage et conversion

EXERCICE N°1 *Conversion*

1°). Conversion Binaire-Décimale
10101010_2 1010110101110_2 $10110,10101_2$ $1011011,11011_2$

2 °). Conversion Décimale-Binaire
152_{10} 1240_{10} $1000,125_{10}$

3 °). Conversions d'une base à une autre
1234(10) = ? (16)
138,145(10) = ? (2)
754(8) = ? (10)
A9F(16) = ? (10)
234(5) = ? (7)
3FE(16) = ? (4)
312,3(4) = ? (8)
517(8) = ? (16)
11001.01 (2) = ? (16)

4 °). Remplir le tableau suivant :

décimal	Binaire	hexadécimal	BCD
49			
	1101001		
		5F	

EXERCICE N°2

1. Donner la représentation en virgule fixe des nombres suivants : (PE=7bits, PF=6bits)
   $(72,5)_{10}$; $(16,35)_{10}$; $(45,125)_{10}$

2. Représenter en hexadécimal les nombres suivants :
   $(72,5)_8$; $(74,05)_8$; $(22,07)_8$

Partie II : Arithmétique binaire

EXERCICE N°3

1 °). Opérations en binaire pur :
10110111, 1001 10010011,1110
+ 10101011, 0110 + 01111101,1001

11000101,0110 11010101

− 10111001,1111 − 01010110

10100101,1101 10101010
* 11011,101 * 1000

10001100 / 101
11101011 / 100

6 °). Opérations en Hexadécimal :

2E + 3F ; 3FD+978 ; 1F2D+ FABC

BA-AB ; F7A-DC5 .

5 °). Opérations en B.C.D. :
Faire les calculs suivants en BCD
(456 +234)= ?
(145-899)= ?
(439-123)= ?
$135_{10} + 255_{10} =$?
$9578_{10} + 9215_{10} =$?

4 °). Opérations en nombres signés sur en complément à 2 :

Faites l'addition et la soustraction des nombres signés en complément à 2, vérifier le résultat en base 10.

+2+ (+3)
-7- (-3)
-3 + (-4)
-3+ (+2)

EXERCICE N°4

Transformation de codes
Représentez en code Gray les nombres décimaux suivants : 35, 36,37. Le code gray est plus stable pourquoi ?
Donnez le complément à 1 et le complément à 2 des nombres décimaux suivnats (+31), (-31),(+45), (-45).

EXERCICE N°5

1) Représentez le nombre 24810 dans les bases 2, 3, 8, 9 et 16.
(Utilisez la technique des divisions successives pour les bases 2, 3 et 16.)

2 °). Conversions
Convertir les nombres hexadécimaux suivants :

BEDF; 4321; AA55; A987 En base 2, 4, 8.

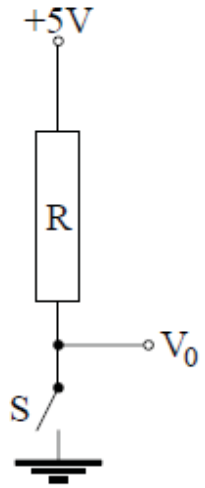
EXERCICE N° 6

1) $1110001_2 = ?_{10}$
2) $100110110_2 = ?_8$
3) $1101100110110_2 = ?_{16}$
4) $100.101_2 = ?_{10}$
5) $1100110.010_2 = ?_8$
6) $1100110110.00101_2 = ?_{16}$
7) $47.75_{10} = ?_2$
8) $145_{10} = ?_8$
9) $287.99_{10} = ?_{16}$
10) $138.32_{10} = ?_2$
11) $783.13_{10} = ?_8$
12) $1443.44_{10} = ?_{16}$
13) $542.756_8 = ?_2$
14) $621_8 = ?_{10}$
15) $743.2_8 = ?_{16}$
16) $671.22_8 = ?_2$
17) $1342.53_8 = ?_{10}$
18) $2045.42_8 = ?_{16}$
19) $AB02.C01_{16} = ?_2$
20) $FF0_{16} = ?_{10}$
21) $D12.1_{16} = ?_8$
22) $A2.B_{16} = ?_2$
23) $93.4_{16} = ?_{10}$
24) $F1B.C_{16} = ?_8$

Chapitre 2 logique combinatoire

Introduction

Les informations traitées par les ordinateurs sont codées sous forme binaire. Un système binaire (signal, circuit, etc…) est un système qui ne peut exister que dans deux états autorisés. Le circuit de la figure suivante est un exemple plus que simpliste de circuit binaire : selon que l'interrupteur S est ouvert ou fermé la tension V0 ne peut être égale qu'à +5 V ou 0 V.



La réalité technique est un peu plus complexe avec des interrupteurs commandés réalisés par des transistors. Diverses notations peuvent être utilisées pour représenter ces deux états :

numérique : 1 et 0 (bit : binary digit)
logique : vrai et faux (true et false)
oui et non (yes et no)
physique : ouvert et fermé ON et OFF, haut et bas (HI et LO, H et L, H et B)

La logique combinatoire utilise les lois de l'algèbre de Boole développée au XIXème siècle par Georges Boole; philosophe et mathématicien Anglais (1815 -1864). L'algèbre de Boole permet d'établir des règles pour l'étude des circuits logiques ou des circuits d'automatisme.
Une variable logique ne peut prendre que 02 (deux) valeurs distinctes (les chiffres de la numération binaire 0 et 1).
Exemple : une lampe L est allumée.
Vrai : L=1
Faux : L=0
Deux types de logiques existent :
*la logique positive :
État bas (0v) : logique « 0 »
État haut(5v) : logique « 1 »
.
*la logique négative :
État bas (0v) : logique « 1 ».
État haut (5v) : logique « 0 ».

Les valeurs 0 et 1 ne sont des symboles représentant un état électrique et qu'elles n'ont aucune signification numérique.

L'algèbre de Boole concerne la logique des systèmes binaires. Une variable booléenne ne peut prendre que deux valeurs possibles 0 ou 1. En électronique les deux états d'une telle variable peuvent être associés à deux niveaux de tension : V(0) et V(1) pour les états 0 et 1 respectivement. On distingue les logiques positive et négative selon que V(1) > V(0) ou V(1) < V(0).
En pratique un niveau est défini par un domaine en tension ou en courant. Par exemple en technologie TTL, un niveau sera dit haut s'il est compris entre +2 V et +5 V et un niveau sera bas s'il est inférieur à +0.8 V. Dans la plage intermédiaire, l'état est indéterminé. Si les transitions sont inévitables, il est indispensable de traverser cette plage intermédiaire le plus rapidement possible. D'autre part, les signaux doivent être stabilisés avant d'être pris en compte par les circuits. Il y a donc des contraintes temporelles, spécifiées par les chronogrammes des feuilles de données (data sheets) fournis par les constructeurs.

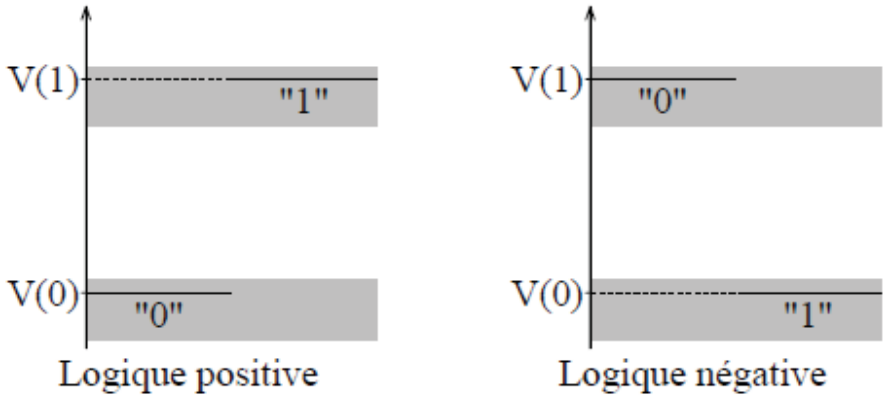
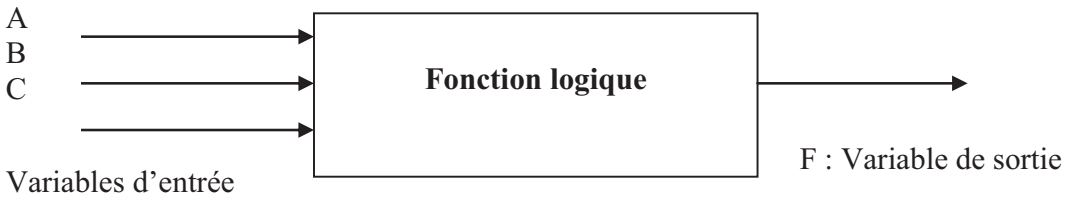


Figure 7. Matérialisation de la logique positive et négative

2.1 Fonctions logiques

2.1.1 Définition

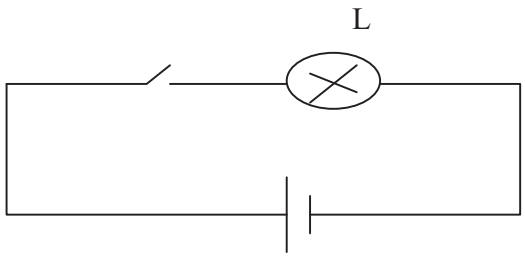
Une fonction logique f de plusieurs variables a,b,c est une application qui définit une variable de sortie binaire (prenant ses valeurs sur l'ensemble [0,1]) à partir de variables d'entrée également binaires .



Ainsi, pour une fonction de deux variables a et b ; il y'a 04 combinaisons possibles de valeurs binaires des variables :
(ab=00 , ab=01 , ab=10 , ab=11).
Pour une fonction de 03 variables : il y'a 08 combinaisons possibles.

En général ; pour une fonction de n variables ; il y'a 2^n combinaisons possibles des valeurs des variables :
F est donc définie par un ensemble de 2^n « 0 » ou « 1 ».

Exemple : considérons une lampe et un interrupteur disposés en série :



L'état de la lampe (allumée ou éteinte) dépend de l'état de l'interrupteur (fermé ou ouvert) .
L'état de la lampe est une fonction logique de l'état de l'interrupteur.

Le même exemple peut être traité en utilisant des diodes à jonction comme l'illustre le schéma suivant :
Si l'anode est connectée à la borne positive ; la diode est passante et la tension aux bornes de la résistance est mesurable (égale approximativement à Ve)
Si l'anode est connectée à la borne négative ; la diode est bloquée et la tension au niveau de la résistance est nulle.
La tension de la résistance est donc fonction de la tension d'entrée.

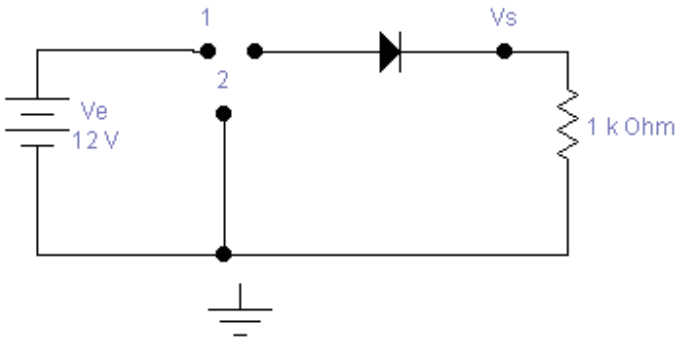


Figure 8. Matérialisation de la fonction égalité

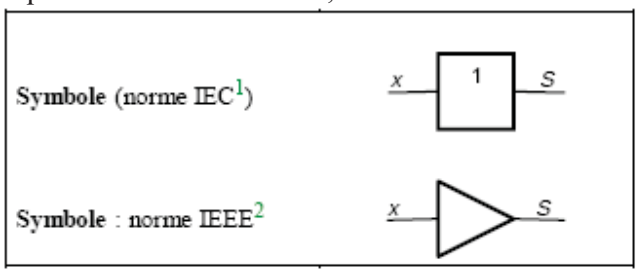
La tension Vs est approximativement égale à Ve si l'interrupteur est connecté à la position 1 (position fermé).

Il existe 04 fonctions logiques élémentaires :
- la fonction égalité.
- la fonction négation (Pas, Non, No, inverse).
- La multiplication logique (ET, And).
- L'addition logique (OU, OR).
- La fonction complément.

On peut réaliser des fonctions combinaisons de ces fonctions élémentaires :
-la fonction OU exclusif.
La fonction NOR (NON_OU)
La fonction NAND (NON_ Et)
La fonction NON OU exclusif.

2.1.2 Opérateur OUI

L'opération (ou opérateur) OUI est dite **unaire** (ne s'applique qu'à un seul opérande). Elle affecte à la variable de sortie l'état logique de la variable d'entrée.
Equation : x est la l'entrée, S la sortie : S = x.



1 IEC, International Electrotechnical Commission (CEI en français).
2 IEEE, Institute of Electrical and Electronics Engineers.

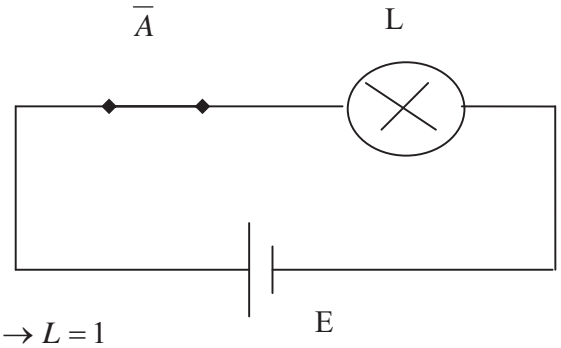
Remarque : les anglo-américains notent H (*High*) le niveau haut et L (*Low*) le niveau bas.
Table de vérité

x	S
0	0
1	1

2.1.3 Opérateur NON, négation : (fonction complément)

L'opération (ou opérateur) NON est la fonction unaire qui affecte à la variable de sortie l'état complémentaire de la variable d'entrée.

Equation : x est la l'entrée, S la sortie, $S = \overline{x}$.
Cette fonction est appelée fonction NON ; inverse ou NOT.



Si $\overline{A} = 1 \rightarrow L = 1$
$\overline{A} = 0 \rightarrow L = 0 \quad \Rightarrow L = \overline{A}$

En résumé ; on aura : $si\ f = 0 \rightarrow \overline{f} = 1$
$et\ si\ f = 1 \rightarrow \overline{f} = 0$

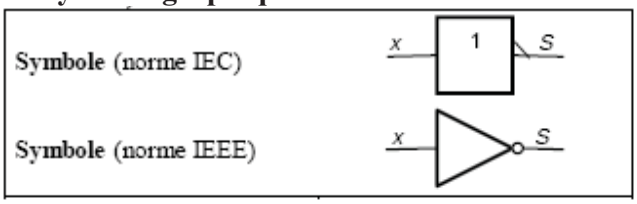
a- Notation

NONA = NOTA= $\overline{A}$

b- Table de vérité

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

c- symbole graphique :



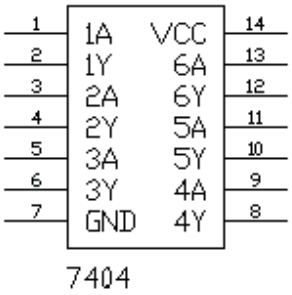
d- Propriétés

$A + \overline{A} = 1$

$A * \overline{A} = 0$

$\overline{\overline{A}} = A$

e- Exemple de circuit intégré



2.1.4 Fonction ET

L'opération ET est le **produit logique**. C'est un opérateur **binaire** qui affecte à la variable de sortie l'état 1 si et seulement si les variables d'entrée sont à 1 simultanément.

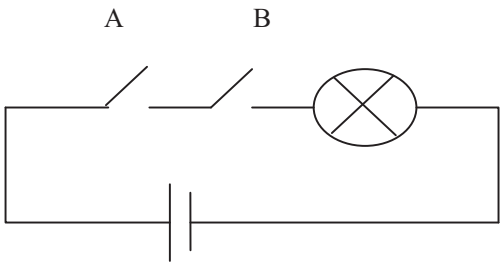


Figure 9. Matérialisation de la fonction ET

La lampe est allumée si A et B sont fermés simultanément soit :

L=1 si A=1 et B=1.
On écrira alors L= A.B ou C=A.B.

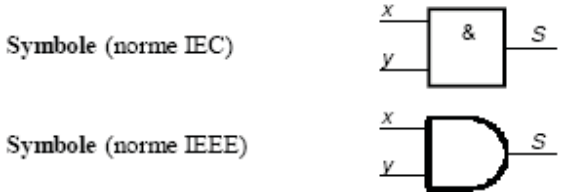
a- Notation :

A and B = $A \wedge B = A.B = C$

b- table de vérité :

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

C-symbole logique : porte logique And à deux entrées.



Cas de plusieurs variables :

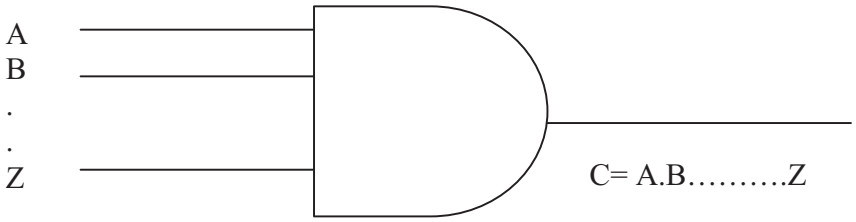
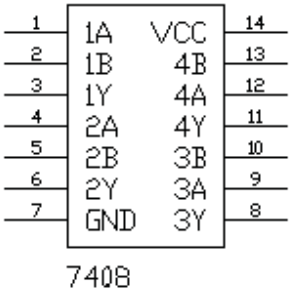


Table de vérité

x	y	S
0	0	0
0	1	0
1	0	0
1	1	1

d- Exemple de circuit intégré réalisant cette fonction



e- Propriétés

La fonction ET « And » est :
-commutative : A.B= B.A.
-Associative : A(BC) = (AB). C
-Distributive : A (B +C) = AB + AC.
-Complémentaire : $A.\overline{A} = 0$
- Elément neutre : A.1= A.
- Identité : A.A= A

2.1.5 Fonction OU

L'opération OU est la **somme logique**. C'est un opérateur binaire qui affecte à la variable de sortie l'état 1 si et seulement si une variable d'entrée est à 1.

Cette fonction est appelée somme logique ; réunion ou OR (fonction de plusieurs variables binaires).

Soit l'exemple suivant :

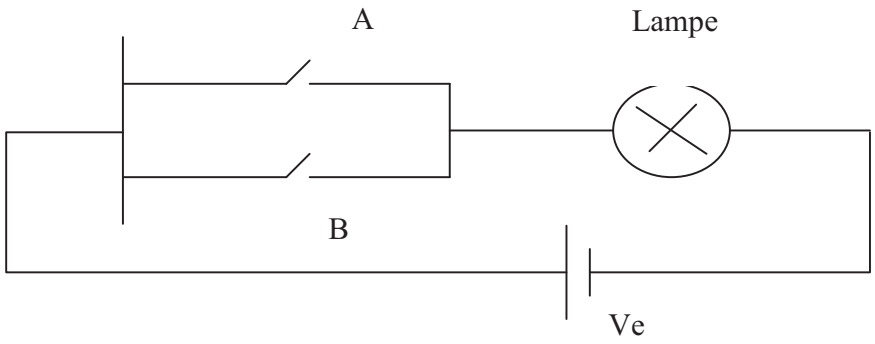


Figure 10. Matérialisation de la fonction OU

La lampe est allumée si on ferme A **OU** si on ferme B ; soit :
L=1 (ou C=1) si A=1 ou B=1.
On écrit alors : L=A.

a- Notation

A OU B = A OR B = $A \cup B$ = A +B.

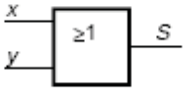
b- Table de vérité:

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

Remarque : cette algèbre est différente de l'algèbre des nombres.

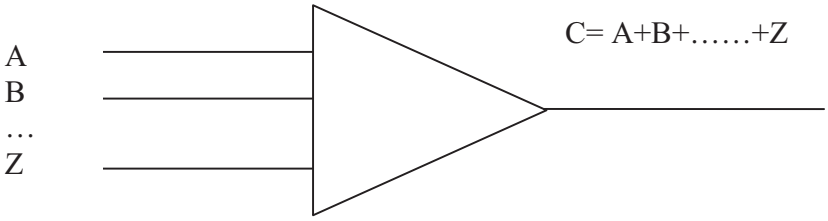
c-Symbole logique

Symbole (norme IEC)



Symbole (norme IEEE)

Cas de plusieurs variables :



d- Propriétés

La fonction OU est :

-commutative: A + B =B +A

-Associative: A + (B+C) = (A +B) +C

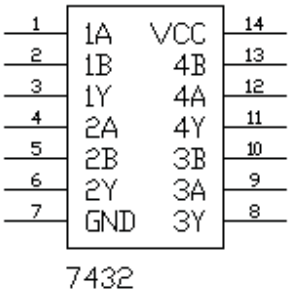
-Distributivité : A(B+C)=AB+AC

-Complémentaire: $A + \overline{A} = E = 1$

-Elément neutre : A+0=A

- Identité : A+A=A.

e- Exemple de circuit intégré réalisant le OU



2.2 Théorème de DEMORGAN

De Morgan a exprimé deux théorèmes qui peuvent se résumer sous la forme suivante :

$\overline{a.b.c.....} = \overline{a}. + \overline{b} + ...$

$\overline{a + b + c + ...} = \overline{a}.*\overline{b}.....$

2.2.1 Théorème 1

Le complément d'une somme de variables logiques est égal au produit des compléments de ces variables :

$\overline{a + b + c + ... + z} = \overline{a}.*\overline{b}*.....*\overline{z}$

Exemple à deux variables :

$\overline{a + b} = \overline{a}*\overline{b}$

Vérification :

a	b	$\bar{a}$	$\bar{b}$	(a+b)	$\overline{a+b}$	$\bar{a}*\bar{b}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

2.2.2 Théorème 2

Le complément d'un produit de variables logiques est égal à la somme des compléments de ces variables.

$$\overline{a*b*c.....*z} = \bar{a}+\bar{b}+.....+\bar{z}$$

exemple : $\overline{a*b} = \bar{a}+\bar{b}$

a	b	$\bar{a}$	$\bar{b}$	(a*b)	$\overline{a*b}$	$\bar{a}+\bar{b}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Le complément d'une expression logique s'obtient en remplaçant :

- chaque variable par son complément.
- Chaque signe + par le signe * et inversement.

Les théorèmes de De Morgan montrent qu'une fonction ET peut être fabriquée à partir des fonctions OU et NON. De même une fonction OU peut être obtenue à partir des fonctions ET et NON. La figure suivante montre la conversion d'une porte OU en porte ET et réciproquement, utilisant le fait que :

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

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

De même, à partir des théorèmes de De Morgan nous pouvons montrer qu'une porte ET en logique positive fonctionne comme une porte OU en logique négative et vice versa.

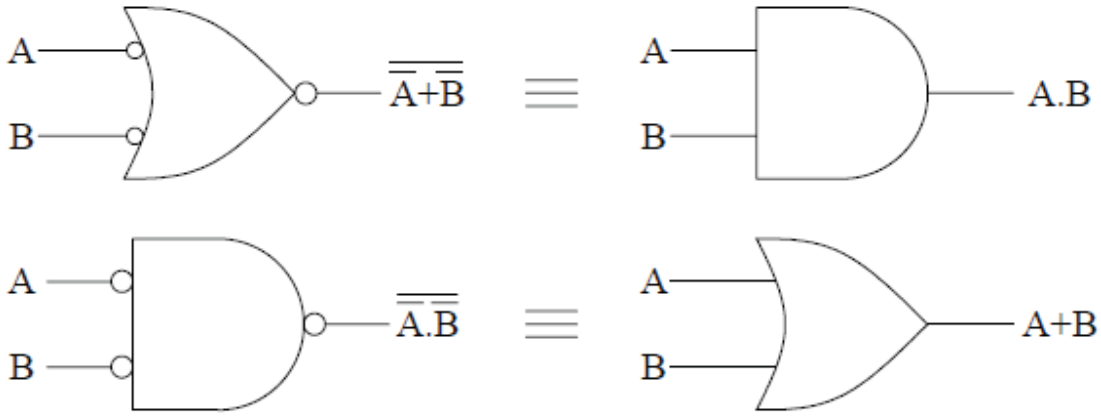


Figure 11. Conversion de portes ET en OU et vice versa

3 Autres portes logiques

Les portes logiques sont des circuits fondamentaux qui traitent les signaux binaires afin de réaliser les différentes fonctions logiques réalisées à partir de semi conducteurs sous forme de circuit intégré (voir chapitre 3)).

Remarque :

Une porte logique peut comporter plusieurs lignes d'entrée mais n'a qu'une seule ligne de sortie.

A partir de portes logiques élémentaires (OU , ET, NON) on peut constituer d'autres portes tel que :

1-NON_ET « NAND »

2- NON_OU « NOR »

3- OU exclusif ou « XOR »

4-NON OUexclusif ou « XNOR ».

2.3.1 Porte logique NON_ET (NAND)

Cette fonction logique est le résultat de l'association d'un NON et d'un ET. C'est un opérateur binaire qui affecte à la variable de sortie l'état 0 si et seulement si les variables d'entrée sont à 1 simultanément.

Elle résulte de la combinaison des 02 portes « ET » et « NON »

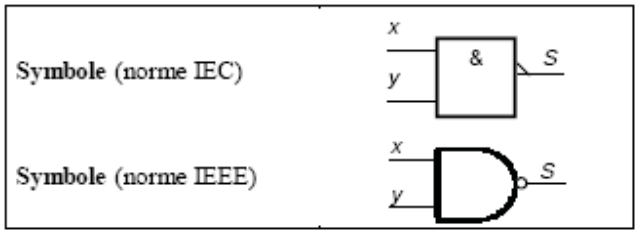
a- Notation

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

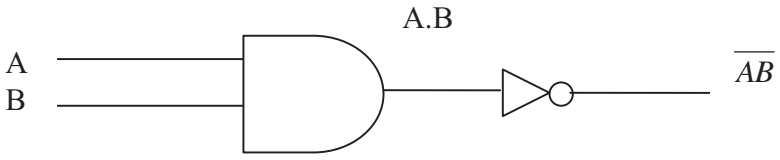
b- Table de vérité

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

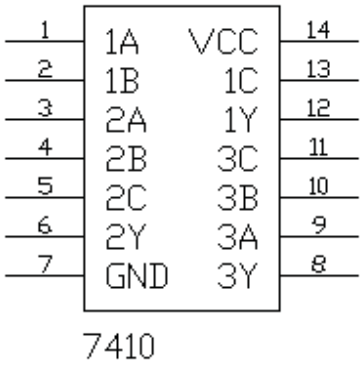
c- Symbole



Ou une autre représentation :



d- Exemple de circuit intégré



2.3.2 Porte logique NON OU :(NOR)

Cette fonction logique est le résultat de l'association d'un NON et d'un OU. C'est un opérateur binaire qui affecte à la variable de sortie l'état 1 si et seulement si les variables d'entrée sont à 0 simultanément.

Elle résulte de la combinaison des 02 portes « OU » et « NON », appelée fonction inverse de « OU ».

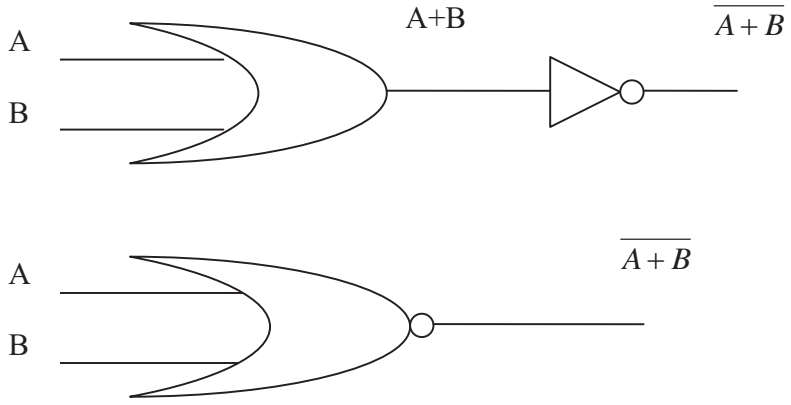
a- Notation

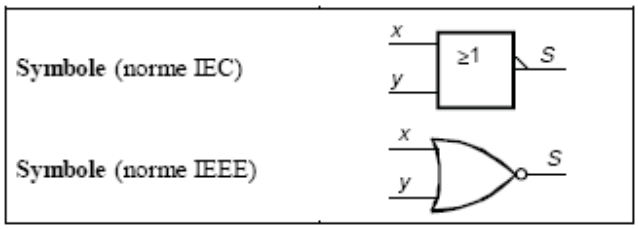
$C = \overline{A+B} = \overline{A} * \overline{B}$

b- Table de vérité

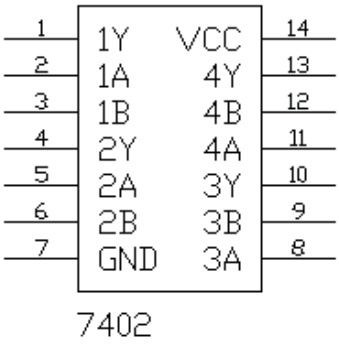
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

c- Symbole logique





d- Exemple de circuit intégré



2.3.3 Porte logique « OU exclusif »

Cette porte est la combinaison des 03 portes élémentaires : ET, OU et NON. Sa sortie est au niveau haut si une et une seule entrée est au niveau haut.

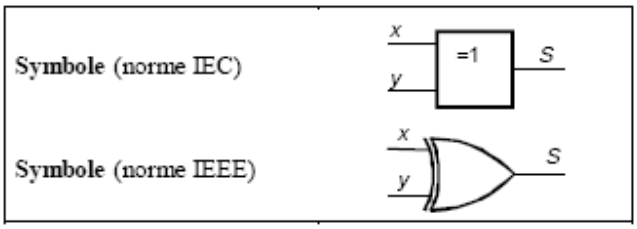
a- Notation

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

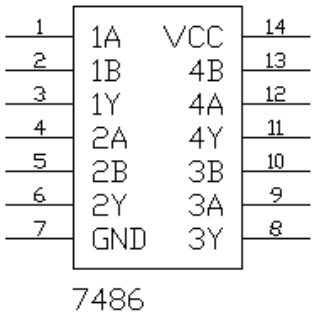
b- Table de vérité

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

c-Symbole logique



c- Exemple de circuit intégré



Nous pouvons formuler de diverses manières la définition précédente : $C = A \oplus B$ est égal à 1 si et seulement si A = 1 ou B = 1 mais pas simultanément. Ce que nous pouvons écrire :

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

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

Une fonction XOR fournit un comparateur d'inégalité : $C = A \oplus B$ ne vaut 1 que si A et B sont différents. Si A et B sont égaux à 1 ou si A et B sont égaux à 0 alors Y = 0. Ce qui permet d'écrire :

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

La fonction $Y = \overline{C} = \overline{A \oplus B}$ correspond à un détecteur d'égalité. Nous avons encore la relation suivante qui peut être démontrée en utilisant les théorèmes de De Morgan :

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

A ces quatre relations logiques correspondent quatre circuits réalisant la fonction XOR à partir de portes OR et AND.

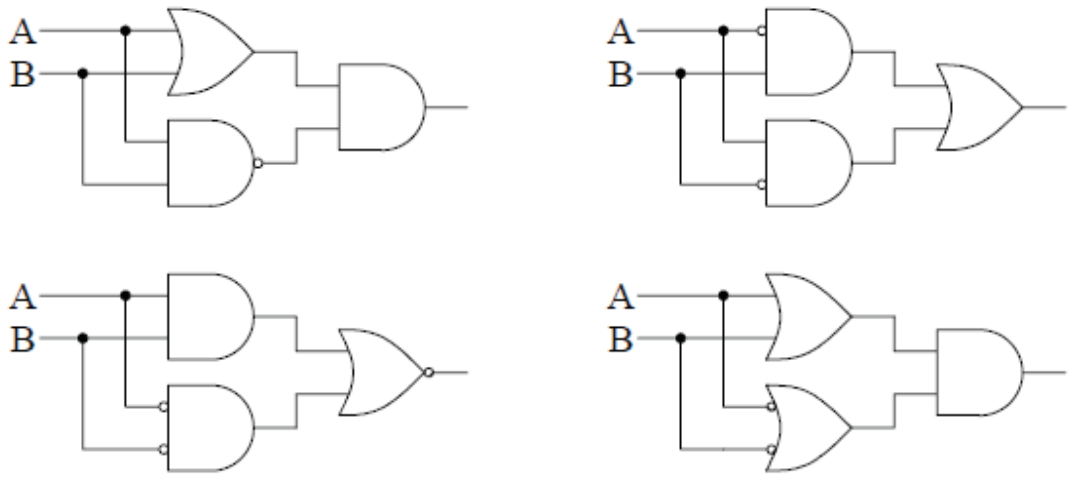


Figure 12. Réalisation de la fonction XOR à partir de portes OR et AND

Mentionnons également quelques propriétés faciles à vérifier :

$A \oplus A = 0$

$\overline{A} \oplus A = 1$

$A \oplus 0 = A$

$A \oplus 1 = \overline{A}$

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

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

Généralisation

L'opérateur XOR se généralise à un ensemble de *n* variables d'entrée par la définition suivante :
La sortie vaut 1 si et seulement si le nombre d'entrées à 1 est impair.

2.3.4 Porte logique NON OUexclusif : (XNOR)

C'est la combinaison de deux portes « XOR » et « NON »
C'est une fonction inverse de XOR.

a- Notation

$C = \overline{A \oplus B} = A \bullet B$

b- Table de vérité :

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

Autres propriétés

a est une variable logique

$a + \overline{a} = 1 \qquad a + 0 = a \qquad a + 1 = 1 \qquad a + a = a$

$a.\overline{a} = 0 \qquad a.0 = 0 \qquad a.1 = a \qquad a.a = a$

Application

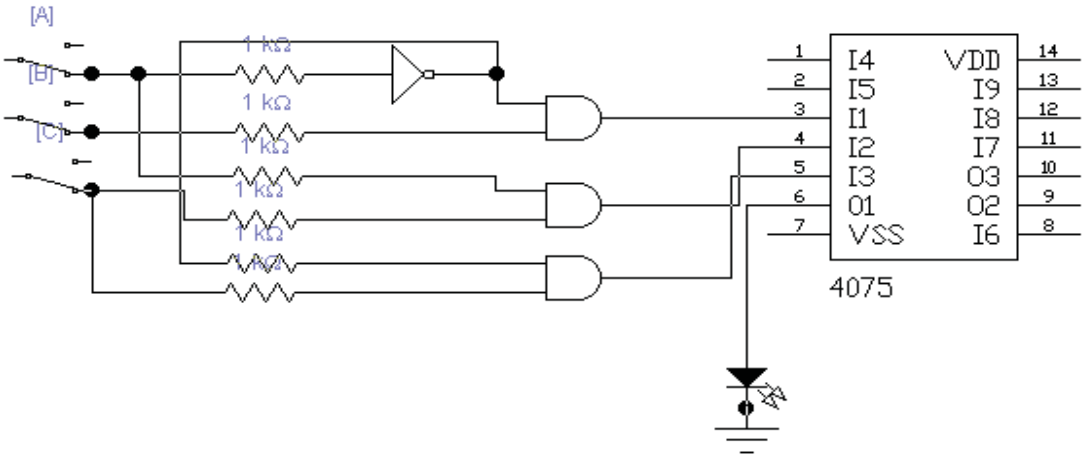
1-Transformer les expressions suivantes en utilisant les théorèmes de DEMORGAN :

a- $\overline{AB + C}$

b- $\left[\overline{aAB + AD).(\overline{B} + \overline{C})}\right]$

donner alors le logigramme correspondant (présentation sous forme de portes logiques)

2-soit le circuit logique suivant



Donner l'expression de la sortie logique O1 .
Remarque : **le CI 7475** est composé de 03 portes OR à 03 entrées.

2.4 Table de vérité et forme canonique de la fonction logique

La table de vérité est un tableau qui permet d'exprimer les valeurs de la fonction étudiée (0 ou1) pour les différentes valeurs des variables.
Cette table comporte autant de colonnes que le système comporte de variables (A,B,C…….) avec une colonne réservée à la sortie et autant de lignes que le nombre de combinaisons possibles.

(N)= nombre de lignes =nombre de combinaisons
(n) =nombre de colonnes =nombre de variables.
Exemple=
Pour une variable a : 2lignes pour combinaison de a
Pour 3 variables a,b,c : 8 lignes pour les combinaisons.

2.4.1 Forme canonique d'une fonction logique

Soit la table de vérité suivante :

A	**B**	**C**	**X(F)**	$\overline{X}$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

2.4.1.1 Première forme canonique

Elle est donnée par la réunion d'intersection, c'est-à-dire que c'est une somme de produits (SP)

La fonction F est égale à 1 lorsque :

- A=0 ; B=1 ; C=1 ; ou lorsque $\overline{A}BC = 1$

OU lorsque

- A=1 ; B=0 ; C=1 ; ou lorsque $A\overline{B}C = 1$

OU lorsque

- A=1 ; B=1 ; C=0 , ou lorsque $AB\overline{C} = 1$

OU lorsque

- A=1 ; B=1 ; C=1 ou lorsque ABC=1.

On obtient finalement 04 termes équivalents binaires des valeurs « 1 » de la fonction qu'on peut donc représenter par l'équation suivante :

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

Pour ce type d'expression (somme logique de plusieurs produits logiques) : les portes utilisées sont des portes NAND.
La fonction F devient donc :

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

2.4.1.2 Deuxième forme canonique

Elle est donnée par l'intersection des réunions ; c'est-à-dire un produit de somme.
La fonction $\overline{X}(\overline{F})$ est égale à 1 pour 04 combinaisons :

- A=0 ; B=0 ; C=0 ; c-a-d lorsque : $\overline{A}\,\overline{B}\,\overline{C} = 1$

OU

- A=0 ; B=0 ; C=1 ; c-a-d lorsque : $\overline{A}\overline{B}C = 1$

OU

- A=0 ; B=1 ; C=0 ; c-a-d lorsque : $\overline{A}B\overline{C} = 1$

OU

- A=1 ; B=0 ; C=0 , c-a-d lorsque $A\overline{B}\overline{C} = 1$

Donc ; on peut écrire que :

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

par conséquent

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

Les portes logiques utilisées sont des portes NOR

Ce 2éme type d'expression s'appelle la 2éme forme canonique de f .
Remarque : les combinaisons qui ne sont pas définies ; sont dites indifférentes ou interdites présentant une fonction logique incomplète.

2.5 Simplification des fonctions logiques

2.5.1 Introduction

La simplification est la recherche des termes ou variables inutiles pour satisfaire la fonction recherchée (minimisation de la fonction logique).

Exemple :

$F = \overline{a}\,\overline{b} + \overline{a}\,b = \overline{a}(b + \overline{b}) = \overline{a}$

On a plusieurs possibilités pour simplifier une équation logique :

- simplification algébrique.
- Simplification graphique.

2.5.2 Simplification algébrique

La simplification algébrique ne fait appel à aucune méthode précise mais nécessite une connaissance parfaite des théorèmes fondamentaux de l'algèbre de Boole.

2.5.2.1 Utilisation des relations classiques

Exemple1

Z=A+AB= A(1+B)=A : d'où la variable B est inutile.

Exemple2 :

$Z = \overline{A}B\overline{C} + AB\overline{C} = B\overline{C}(A + \overline{A}) = B\overline{C}$ **:** A est inutile

Exemple3 :

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

$= \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\,\overline{B} + \overline{B}\,\overline{C} = \overline{B}(1 + \overline{C}) + \overline{A}\,\overline{C}$

$donc \quad Z = \overline{B} + \overline{A}\,\overline{C}$

De la même manière :

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

2.5.2.2 utilisation de la fonction complément $\overline{Z}$

Soit l'expression suivante :

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

$\overline{Z} = \overline{A + \overline{A}B} = \overline{A} * (\overline{\overline{A}B}) = \overline{A}(A + \overline{B}) = \overline{A}A + \overline{A}\,\overline{B} = \overline{A}\,\overline{B}$

$\overline{\overline{Z}} = Z = \overline{\overline{A}\,\overline{B}} = A + B$

2.5.2.3 Addition de termes existants :

On ne change pas la valeur d'une expression booléenne en dédoublant un ou plusieurs de ses termes , cela permet quelques fois des mises en facteur.

Exemple :

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

$= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} = \overline{A}\,\overline{B}\left[C + \overline{C}\right] + \overline{B}\,\overline{C}\left[A + \overline{A}\right]$

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

2.5.2.4 Multiplication par un facteur égal à 1

On ne change pas la valeur d'une expression booléenne en multipliant un ou plusieurs de ses termes par un terme égal à 1 . ce terme peut être la somme complémentaire d'une variable quelconque ($X + \overline{X}$)

Exemple1 :

Simplifier $Z = AB + AC + B\overline{C}$

En multipliant AB par ($C + \overline{C}$)

$Z = AB(C + \overline{C}) + AC + B\overline{C} = ABC + AB\overline{C} + AC + B\overline{C}$

$Z = AC(B + 1) + B\overline{C}(A + 1) = AC + B\overline{C}$

exemple2 :

$X = (A + B + C + \overline{A}\,\overline{B}\,\overline{C})DE$

$on \quad remarque\ \overline{A}\,\overline{B}\,\overline{C} = \overline{A + B + C}$

$on\ écrit\ alors\ X = \left[(A + B + C) + (\overline{A + B + C})\right]DE$

$d'ou \quad X = DE$

2.5.3 Simplification graphique « utilisation du tableau de Karnaugh »

On appelle tableau de Karnaugh une grille comportant un nombre de case égal au nombre de combinaison des variables de la fonction booléenne à étudier ; le tableau est organisé de telle façon qu'une seule variable change entre les cases voisines ; ceci est obtenu en énumérant les vecteurs dans un ordre particulier (code Gray).

Le nombre de cases pour n variables = 2^n

Cas d'une seule variable a

Cas de 02 variables A et B

A / B	0	1
0		
1		

Cas de 03 variables A, B et C

AB / C	00	01	11	10
0				
1				

Cas de 04 variables A,B,C et D : 16 combinaisons :

AB CD	00	01	11	10
00				
01				
11				
10				

2.5.3.1 Simplification des expressions

On groupe les « 1 » contenues dans les cases adjacentes ; c'est à dire pour lesquelles une seule variable change d'état et on ne retient que les variables qui conservent leur valeur lorsqu'on passe d'une case à l'autre.

Exemple1 :
Simplifier l'équation suivante :
$X = \bar{a}\bar{b}\bar{c} + abc + ab\bar{c} + a\bar{b}c + a\bar{b}\bar{c}$
Représentant les un de cette fonction dans un tableau à 08 variables :

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

1er groupement de quatre un : la variable qui ne change pas d'état est : a
2éme groupement de deux un : les variables qui ne changent pas d'état : $\bar{b}\bar{c}$
D'où $X = a + \bar{b}\bar{c}$
Exemple 2 :
Simplifier l'équation suivante :
$X = abc\bar{d} + ab + \bar{c}\bar{d} + abcd$

ab cd	00	01	11	10
00	**1**	**1**	**1**	**1**
01			**1**	
11			**1**	
10			**1**	

1er groupement de 04 un : les variables qui ne changent pas leur états est $\bar{c}\bar{d}$
2éme groupement de quatre un : les variables qui ne changent pas leurs états :ab
D'où $X = ab + \bar{c}\bar{d}$

2.5.3.2 Inverse d'une expression

Si on représente dans un diagramme de karnaugh la onction X=f(a,b,c,d).
Le diagramme de la fonction inverse $\bar{X}$ s'obtient en remplaçant les un par des zéro et réciproquement.

Exemple :
Simplifier la fonction suivante :
$X = \bar{a}\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}\bar{c}d$

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**

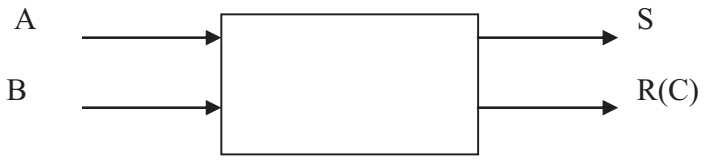
La fonction X simplifiée est
$X = \bar{b}\bar{d}$
la fonction inverse simplifiée est :
$\overline{X} = b + d$
$\overline{\overline{X}} = \bar{b}\bar{d}$

2.6 Fonctions arithmétiques usuelles et comparateurs

2.6.1 Semi additionneur ou demi- additionneur(semi-adder)

Le problème posé est de concevoir un circuit qui réalise l'addition de 02 nombres binaires à un bit chacun qui génère une somme S et une retenue R.

a- Table de vérité



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

D'où

l'équation de S et de R :
$S = \bar{a}b + a\bar{b} = a \oplus b$
R=ab

b- logigramme : (demi additionneur)

Le logigramme de notre demi additionneur est donné comme suit :

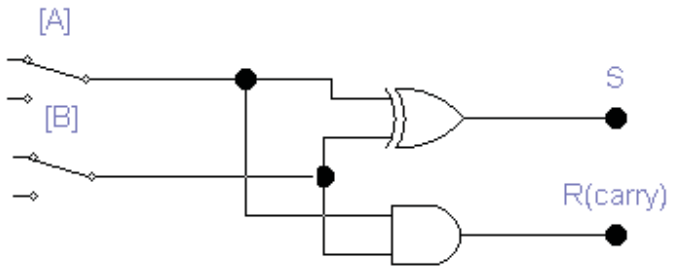


Figure 13. Organigramme d'un demi additionneur

1- Additionneur complet :(Full adder)

Concevoir un circuit qui réalise l'addition de 02 nombres de n bits chacun :

$A = a_{n-1}\ a_{n-2}\ \ldots\ldots\ldots\ldots a_1\ a_0$

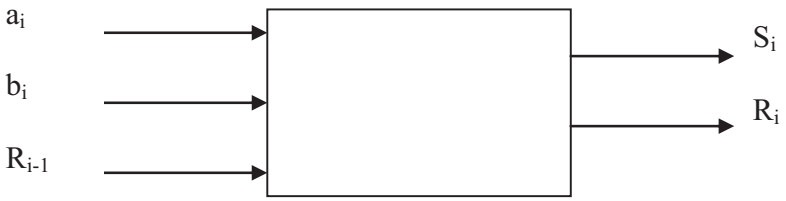
$B = b_{n-1}\ b_{n-2}\ \ldots\ldots\ldots\ldots\ b_1\ b_0$

$$\begin{array}{lcccccc} & R_{n-1} & R_{n-2} & & & R_1 & R_0 \\ A= & a_{n-1} & a_{n-2} & \ldots\ldots\ldots\ldots & & a_1 & a_0 \\ B= & b_{n-1} & b_{n-2} & \ldots\ldots\ldots\ldots & & b_1 & b_0 \\ \hline S= & S_{n-1} & S_{n-2} & & & S_1 & S_0 \end{array}$$

$S_i = (a_i , b_i , R_{i-1})$

a- Table de vérité

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



Equation logique

Soit le tableau de karnaugh pour la représentation de la somme S_i :

$a_i\ b_i$ / R_{i-1}	00	01	11	10
0		**1**		**1**
1	**1**		**1**	

$$S_i = \overline{a_i}\,\overline{b_i}\,R_{i-1} + \overline{a_i}\,b_i\,\overline{R_{i-1}} + a_i b_i R_{i-1} + a_i\overline{b_i}\,\overline{R_{i-1}}$$
$$= R_{i-1}(\overline{a_i \oplus b_i}) + \overline{R_{i-1}}(a_i \oplus b_i)$$
$$S_i = R_{i-1} \oplus (a_i \oplus b_i)$$

Soit le tableau de Karnaugh pour la représentation de la retenue R_i :

$a_i b_i$ / R_{i-1}	00	01	11	10
0			**1**	
1		**1**	**1**	**1**

L'équation de la retenus est :

$$R_i = a_i b_i + b_i R_{i-1} + a_i R_{i-1} = a_i b_i + R_{i-1}(a_i + b_i)$$

le circuit logique de l'additionneur complet est donné par la figure14.

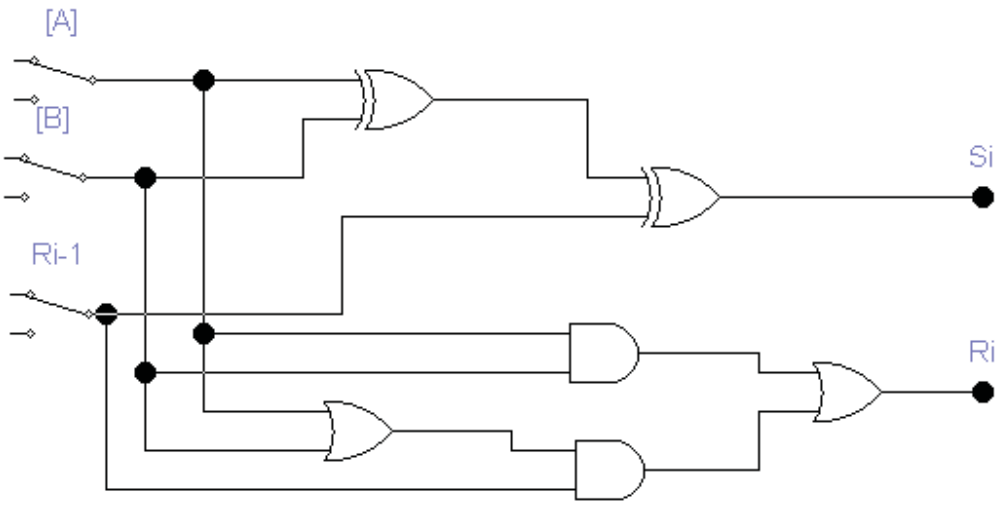


Figure 14. Organigramme d'un additionneur complet

Si on désigne par A et B les nombres binaires à additionner te que :

A= $a_{n-1}\ a_{n-2}$ ……$a_1\ \ a_0$

B= $b_{n-1}\ b_{n-2}$ ……$b_1\ \ b_0$.

Le schéma synoptique d'un additionneur complet est le suivant :

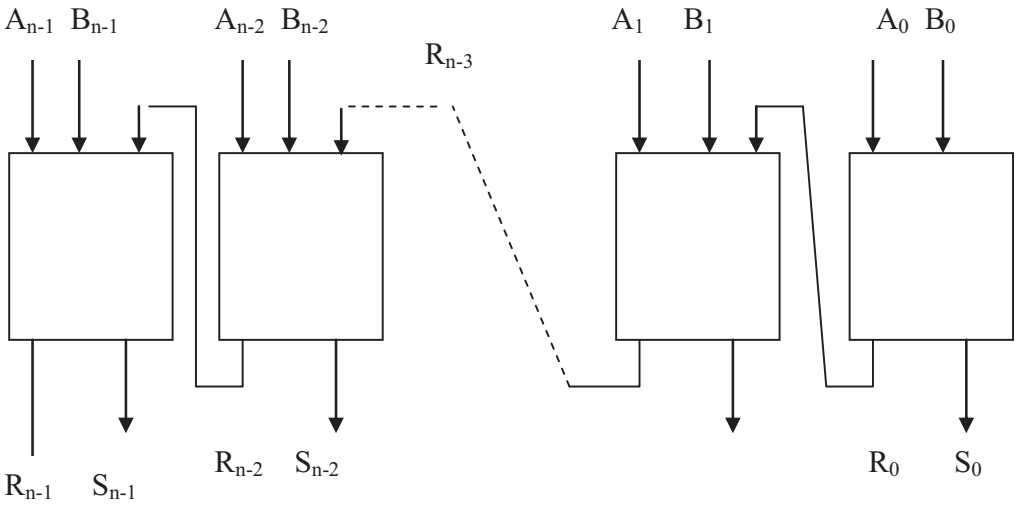


Figure 15. Schéma général d'un additionneur complet

Addition en parallèle

L'addition de nombres comptant plusieurs bits peut se faire en série (bit après bit) ou en parallèle (tous les bits simultanément). La figure 16 montre l'exemple d'un additionneur 4 bits comptant quatre "Full Adders", montés en parallèle ou en cascade. Chaque additionneur FAi est affecté à l'addition des bits de poids i. L'entrée correspondant au report de retenue pour FA0 est imposée à 0 (en logique positive). La retenue finale C indique un dépassement de capacité si elle est égale à 1. Le temps d'établissement du résultat correspondant au temps de propagation des retenues au travers des diverses cellules. Si δt est le temps réponse d'une cellule, la sortie S_0 et la retenue R_0 sont valables après un retard δt, la sortie S_1 et la retenue R_1 ne sont correctes qu'après un retard 2 δt, et ainsi de suite.

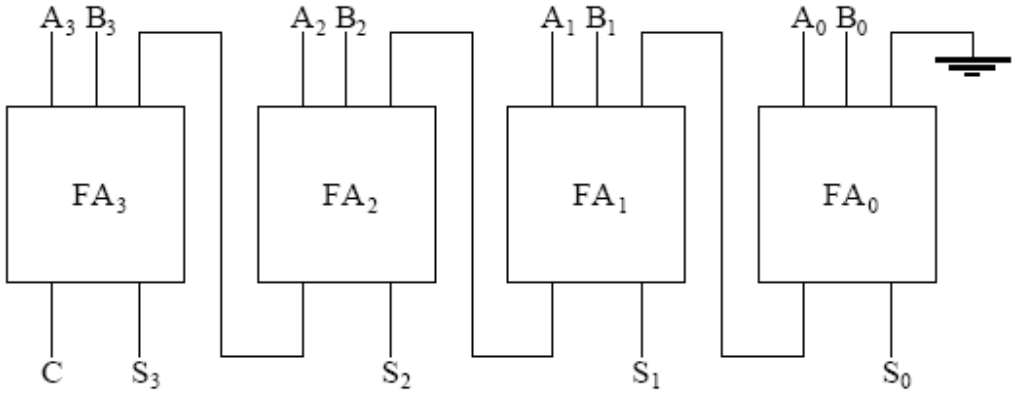


Figure 16. Schéma détaillé d'un additionneur complet à 4 bits

Addition séquentielle

Dans un additionneur séquentiel chacun des nombres A et B est représenté par un train d'impulsions (figure 17) synchrones par rapport à un signal d'horloge. L'ordre chronologique d'arrivée des impulsions correspond à l'ordre croissant des poids : le bit le moins significatif se présentant le premier. Ces impulsions sont injectées sur les deux lignes d'entrée d'un additionneur. A chaque cycle d'horloge, la retenue provenant des bits de poids inférieurs doit être mémorisée (par exemple, à l'aide d'une bascule D qui sera traitée au chapitre 4).

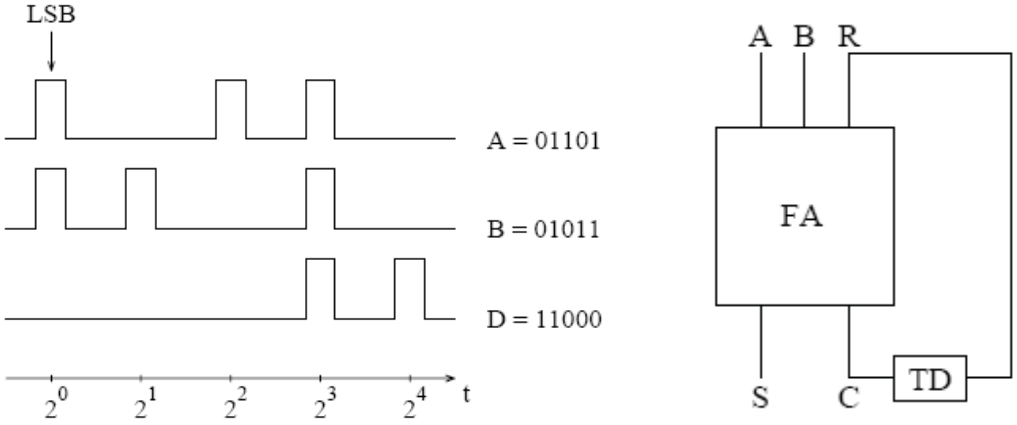


Figure 17. Schéma détaillé d'un additionneur série

Un additionneur parallèle est plus rapide mais nécessite plus de composants.

2.6.2 Semi soustracteur (demi soustracteur)

La soustraction de nombres à 1 bit chacun est présentée de la même manière qu'un demi additionneur.

a- Table de vérité

A_0	B_0	D	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Equation :

$D = a_0 \oplus b_0$

$R = \overline{a_0}\, b_0$

D est la différence et R est la retenue.

La figure 18 montre le logigramme d'un demi additionneur.

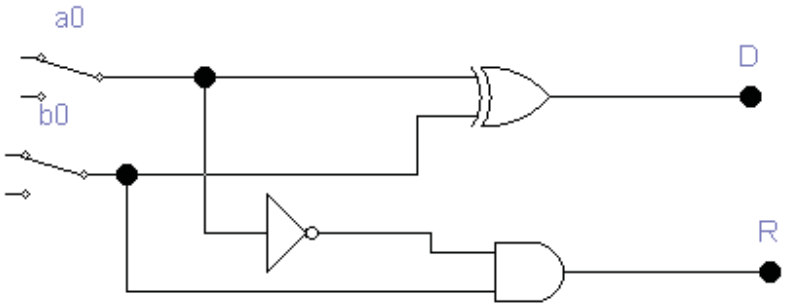


Figure 18. Logigramme d'un demi soustracteur

2.6.3 Comparateur

On rencontre très souvent la nécessité de comparer deux entiers (A = B, A > B ou A < B). soit la table de vérité correspondante à ces trois fonctions de comparaison de 2 bits. La fonction S (supérieur) doit être égale à 1 si et seulement si A > B, la fonction I (inférieur) si et seulement si A < B et la fonction E (égalité) si et seulement si A = B. Ce qui se résume par le tableau suivant :

A	B	S ($A \succ B$)	I($A \succ B$)	E(A=B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Nous en déduisons les expressions logiques de S, I et E :

$$S = A\overline{B}$$

$$I = \overline{A}B$$

$$E = \overline{A \oplus B} = \overline{A\overline{B} + \overline{A}B} = \overline{S + I}$$

La figure 19 présente le logigramme d'un tel comparateur :

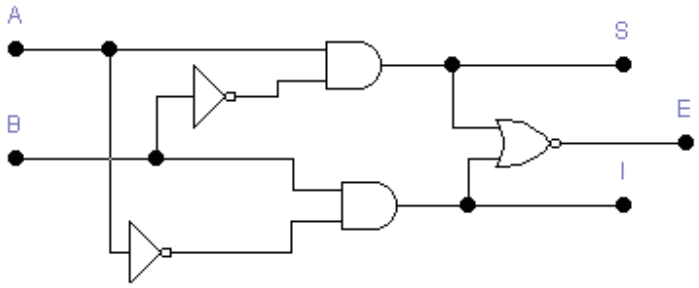


Figure 19. Logigramme d'un comparateur de 02 nombres à 01 bit chacun

Une généralisation de comparateur de deux nombres à 02 bits peut être réalisée comme suit :

a_1	a_0	b_1	b_0	S	E	I
0	0	0	0	**0**	**1**	**0**
0	0	0	1	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**
0	0	1	1	**0**	**0**	**1**
0	1	0	0	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**
0	1	1	0	**0**	**0**	**1**
0	1	1	1	**0**	**0**	**1**
1	0	0	0	**1**	**0**	**0**
1	0	0	1	**1**	**0**	**0**
1	0	1	0	**0**	**1**	**0**
1	0	1	1	**0**	**0**	**1**
1	1	0	0	**1**	**0**	**0**
1	1	0	1	**1**	**0**	**0**
1	1	1	0	**1**	**0**	**0**
1	1	1	1	**0**	**1**	**0**

En analysant la table de vérité, nous remarquons clairement que l'équation $E = \overline{A \oplus B} = \overline{S + I}$ est vérifiée.

2.6.4 Contrôle de parité

La parité d'un mot binaire est définie comme la parité de la somme des bits, soit encore :
- parité paire (ou 0) : nombre pair de 1 dans le mot;
- parité impaire (ou 1) : nombre impair de 1 dans le mot.

La fonction OU-exclusif donne la parité d'un sous-ensemble de deux bits. Le contrôle de parité est basé sur la constatation que le mot de n+1 bits formé en adjoignant à un mot de n bits son bit de parité est toujours de parité 0. La figure 20 représente le diagramme logique d'un générateur contrôleur de parité pour 4 bits. Si l'entrée P' est imposée à 0 ce circuit fonctionne comme générateur de parité : la sortie P représente la parité du mot composé par les bits A, B, C et D.

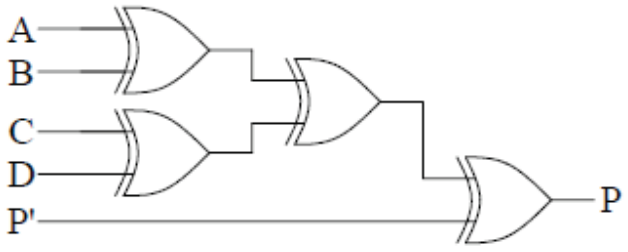


Figure 20. Parité du mot composé par les bits A, B, C et D

Le contrôle de la parité est utilisé, par exemple, pour augmenter la fiabilité d'un système de transmission ou de stockage de données. La figure 21 montre l'utilisation du circuit précédent en générateur de parité du côté de l'émission et contrôleur de parité du côté de la réception. La sortie P2 doit être à 0 pour chaque mot transmis, sinon cela indique un problème de transmission.

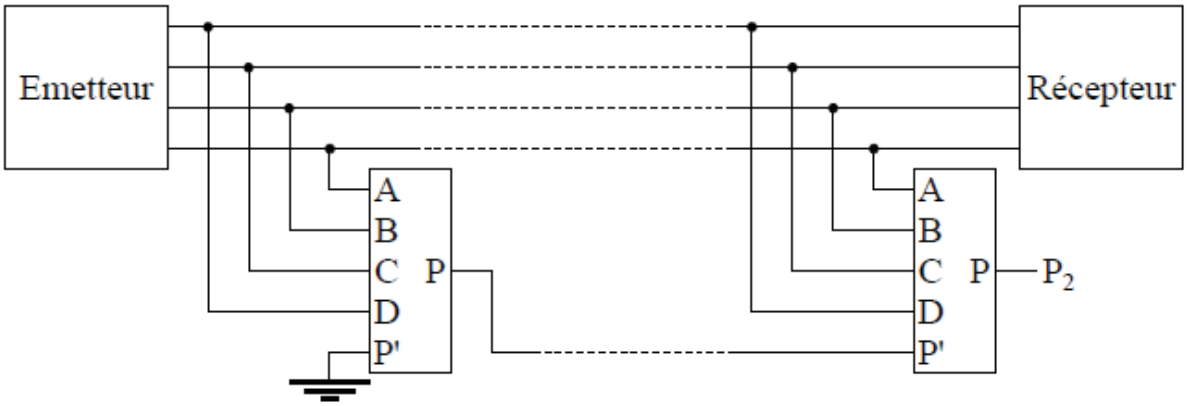


Figure 21. Génération de parité en émission et contrôle de parité en réception

Remarquons cependant que la parité ne permet de détecter qu'un nombre impair de bits en erreur dans un mot. Par ailleurs il ne permet pas corriger les erreurs détectées. Pour ce faire il faut utiliser des codes correcteurs d'erreur qui nécessitent plusieurs bits supplémentaires.

2.7 Codage- Décodage – transcodage

Les codeurs /décodeurs sont des circuits qui permettent d'exprimer un nombre décimal en son équivalent binaire et réciproquement.

Les transcodeurs sont des circuits qui réalisent un changement de code (passage du code binaire : code binaire réfléchi ; code binaire- code Ai Ken …….).

2.7.1 Les codeurs

2.7.1.1 Codage décimal en binaire naturel

Le codeur qui est un cas particulier des transcodeurs est un circuit logique qui traduit un nombre écrit en décimal vers son équivalent binaire

a- Table de vérité du codeur :

N	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

N est le nombre décimal à coder ,(A B C D) est le code binaire correspondant te que D est le poids faible. Les équations d'un te codeur sont :

D=1+3+5+7+9

C=2+3+6+7

B=4+5+6+7

A=8+9.

$\overline{D} = 0 + 2 + 4 + 6 + 8$

$\overline{C} = 0 + 1 + 4 + 5 + 8 + 9$

$\overline{B} = 0 + 1 + 2 + 3 + 8 + 9$

$\overline{A} = 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7$

Le logigramme du codeur est donné par la figure 22.

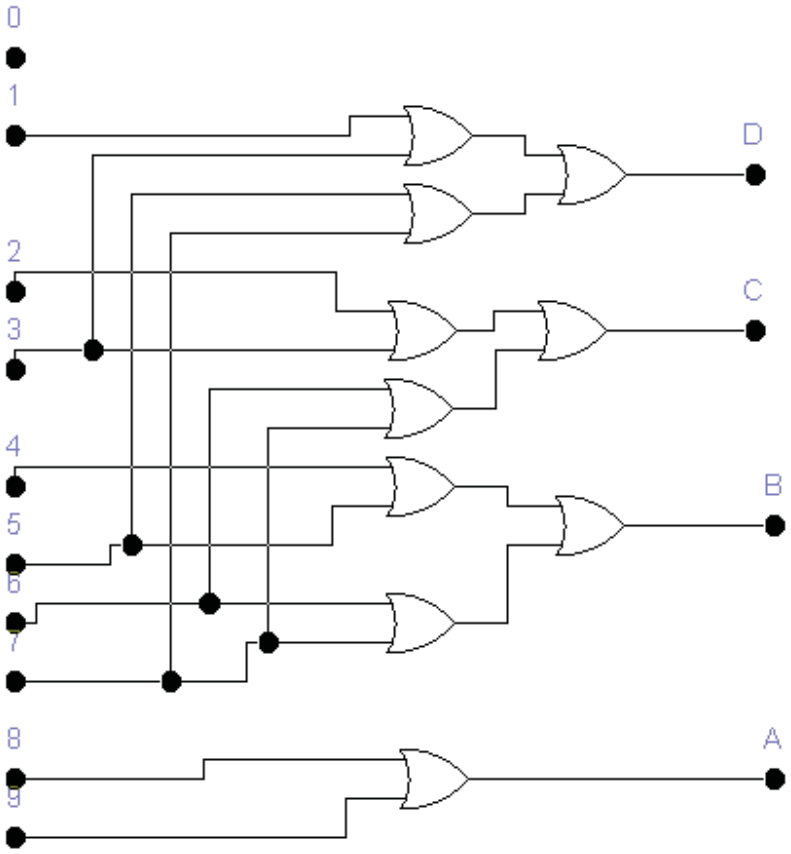
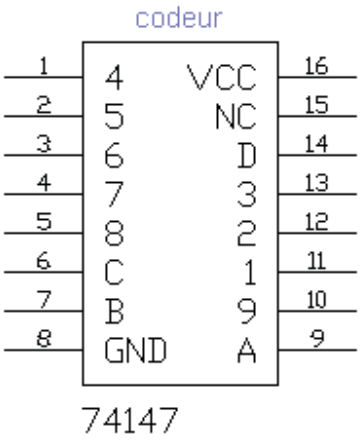


Figure 22. Logigramme du codeur binaire d'un nombre décimal

Plusieurs circuits intégrés délivrent le nombre codé en BCD à partir d'un nombre décimal.

Exemple : **le 74LS147** de la série TTL : ce codeur est en même temps un codeur de priorité de « 10 » lignes « entrées » vers 04 lignes sorties.



Le circuit **intégré « 74LS148 »** représente aussi un codeur de priorité comme pour le 74147 mais avec 8 lignes d'entées (du nombre 1 au nombre7) et 03 lignes de sortie (A, B, C) . L'état inactif de toutes les entrées donne le nombre « 0 ».

2.7.2 Les décodeurs

2.7.2.1 Décodeur BCD- décimal

Le décodeur BCD- décimal est un circuit logique qui traduit un nombre binaire vers son équivalent décimal.

a- Table de vérité

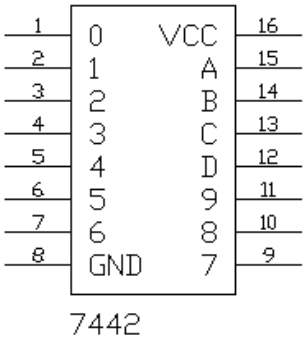
A	B	C	D	N
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

b- Equations

$$0 = \overline{A}.\overline{B}.\overline{C}.\overline{D} \quad ; 1 = \overline{A}.\overline{B}.\overline{C}.D \; ; 2 = \overline{A}.\overline{B}.C.\overline{D} \; ; 3 = \overline{A}.\overline{B}.C.D \; ; 4 = \overline{A}.B.\overline{C}.\overline{D}$$

$$5 = \overline{A}.B.\overline{C}.D \; ; 6 = \overline{A}.B.C.\overline{D} \; ; 7 = \overline{A}.B.C.D \; ; 8 = A.\overline{B}.\overline{C}.\overline{D} \; ; 9 = A.\overline{B}.\overline{C}.D$$

Parmi les circuits intégrés qui délivrent le nombre décimal codé en BCD ; on trouve **le « 7442 » de la série TTL-74.**



les pins (12 ;13 ;14 ;15) correspondent aux entrées (inputs) ; les pins (1 ;2 ;3 ;4 ;5 ;6 ;7 ;9 ;10 ;11) correspondent aux sorties (output). Le logigramme d'un tel décodeur est donné par la figure suivante :

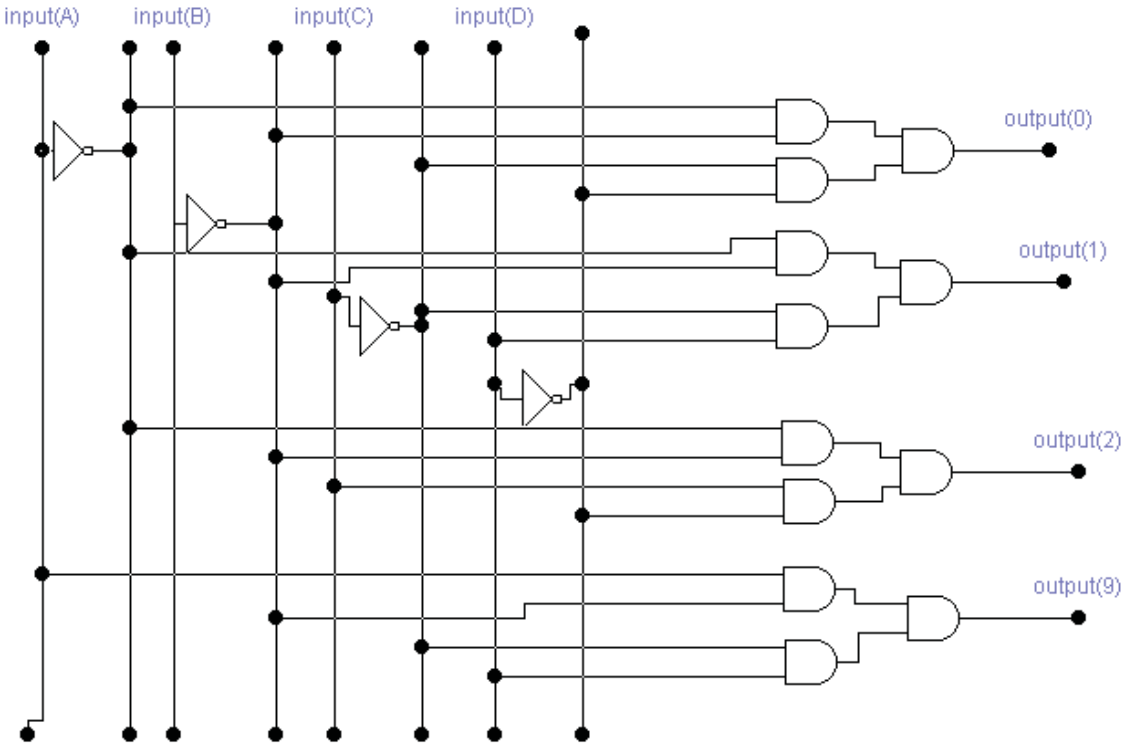


Figure 23. Logigramme du décodeur BCD- décimal

2.7.2.2 Le décodeur binaire

Décodeur de lignes ou sélecteur de sorties

Le décodeur de lignes est un circuit à n entrées et 2^n sorties ; il permet de sélectionner une seule sortie parmi les 2^n

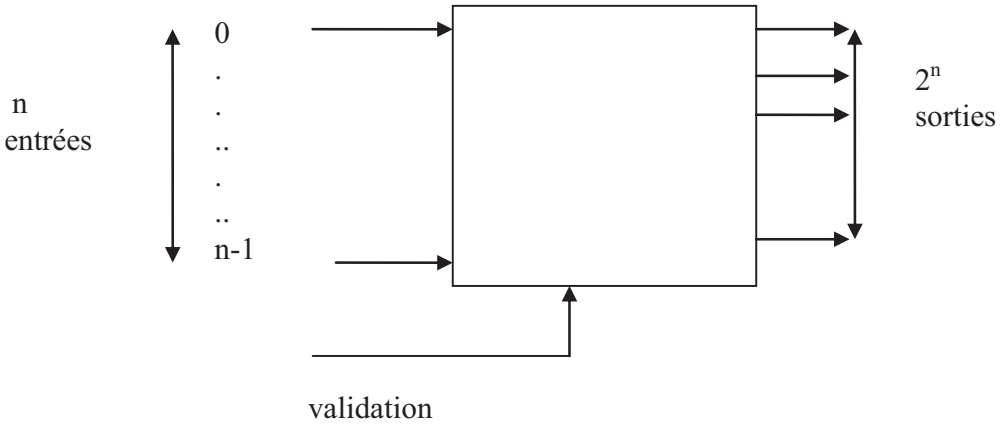


Figure 24. Schéma général du décodeur de lignes

Le rang de la sortie active correspond à la valeur binaire affichée sur les entrées.
Exemple : décodeur 2 vers 4.
Décodeur 3 vers 8
Décodeur 4 vers 16
Problème :
Réaliser un décodeur 3 lignes vers 8. : n=3
$N=2^3$ =8 sorties.
V : entrée de validation (permission de décoder).
On suppose : V=1 : validation
V=0 : inhibition (non validation).

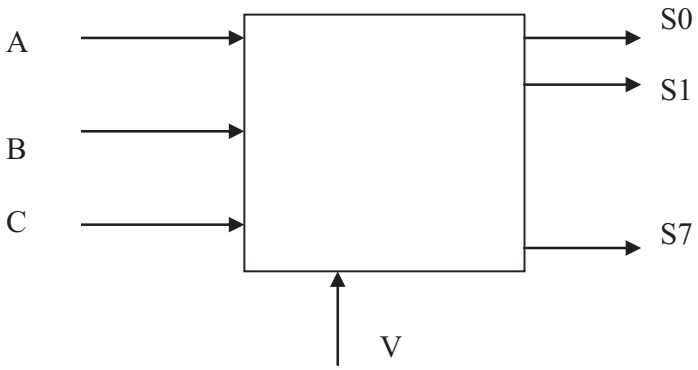


Figure 25 . Schéma synoptique du décodeur de lignes 3 vers 8

a- Table de vérité

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

b- Equations

Soient les équations d'un tel décodeur de lignes :

$$S_0 = V.\overline{A}.\overline{B}.\overline{C}; \quad S_1 = V.\overline{A}.\overline{B}.C; \quad S_2 = V.\overline{A}.B.\overline{C}; \quad S_3 = V.\overline{A}.B.C$$

$$S_4 = V.A.\overline{B}.\overline{C}; \quad S_5 = V.A.\overline{B}.C; \quad S_6 = V.AB.\overline{C}; \quad S_7 = V.A.B.C$$

b- logigramme :

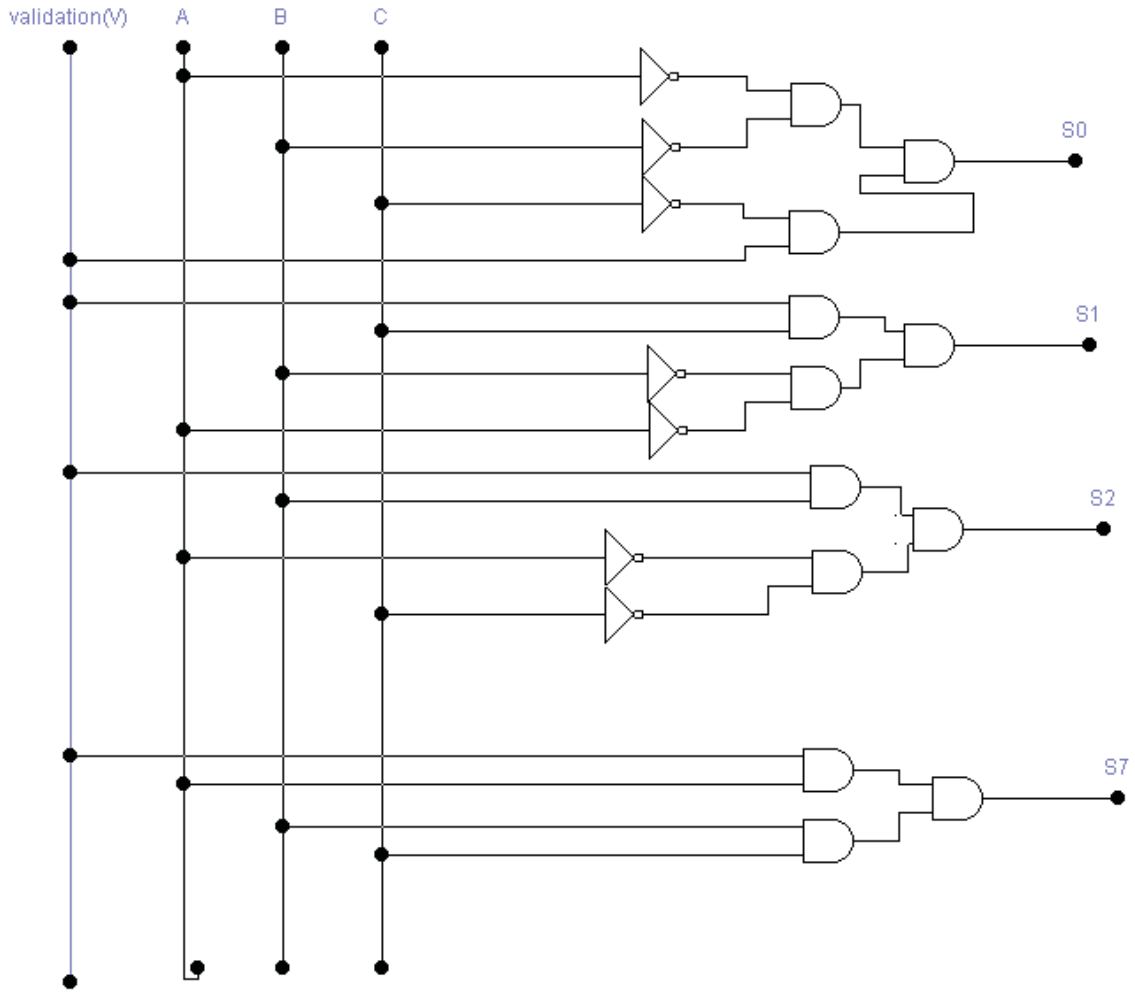


Figure 26 . Logigramme du décodeur de lignes 3 vers 8

2.7.3 Les transcodeurs

Un transcodeur permet de passer d'un code binaire à un autre code binaire.
Exemple : code Aiken – code BCD :
Le code Aiken est connu sous le nom de code 2421 relatifs au poids de chaque bit.

a- Table de vérité

N	A	B	C	D	X	Y	Z	T
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	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	0	1	0	1
6	0	1	1	0	0	1	1	0
7	0	1	1	1	0	1	1	1
8	1	1	1	0	1	0	0	0
9	1	1	1	1	1	0	0	1

Les combinaisons de 10 à 15 sont prises comme interdites.(=ϕ)

b- Equations de sorties

Équation de la sortie X

AB / **CD**	00	01	11	10
00	0	0	ϕ	ϕ
01	0	0	ϕ	ϕ
11	0	0	**1**	ϕ
10	0	0	**1**	ϕ

X=A

Equation de la sorti Y :

AB / **CD**	00	01	11	10
00	0	**1**	ϕ	ϕ
01	0	**1**	ϕ	ϕ
11	0	**1**	0	ϕ
10	0	**1**	0	ϕ

$Y = \overline{A}.B$

Equation de la sortie Z :

AB / **CD**	00	01	11	10
00	0	0	ϕ	ϕ
01	0	0	ϕ	ϕ
11	**1**	**1**	0	ϕ
10	**1**	**1**	0	ϕ

$Z = \overline{A}.C$

Equation de la sortie T :

AB / **CD**	00	01	11	10
00	0	0	ϕ	ϕ
01	**1**	**1**	ϕ	ϕ
11	**1**	**1**	**1**	ϕ
10	**0**	**0**	0	ϕ

T=D.

c- logigramme :

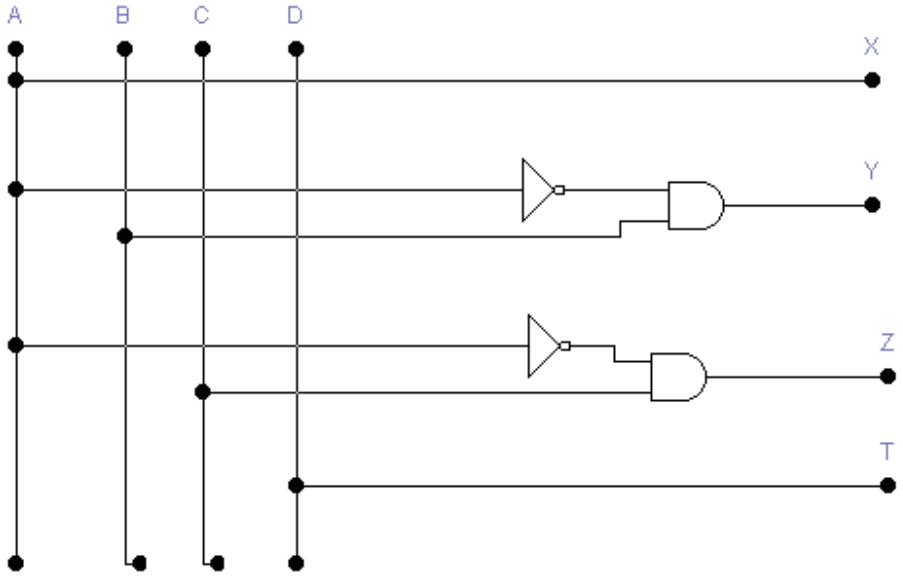
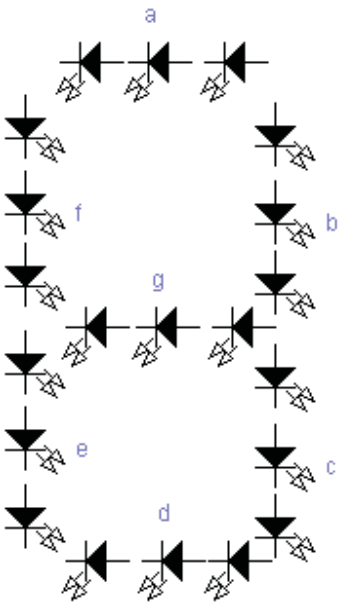


Figure 27 . Logigramme du transcodeur

2.7.4 Les afficheurs

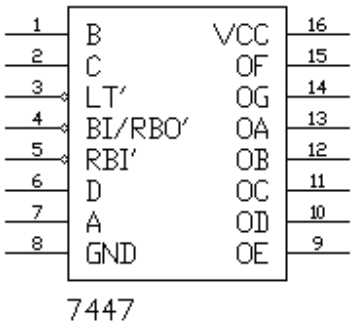
2.7.4.1 Afficheurs à LED

Les afficheurs 7 segments (7 segment display) sont constituées de sept barres de diodes disposées comme suit :



Ils peuvent être de type LED ; LCD ou fluorescent.
L'entrée binaire ou BCD dans un afficheur à 07 traits doit être décodée pour pouvoir afficher le nombre décimal convenable.

Le circuit intégré 7447 est un circuit transcodeur BCD / 7 segments ; son schéma symbolique est comme suit :



les entrées BCD se trouvent sur les pins (1 ;2 ;6 ;7) ; les 07 sorties du décodeur sur les pins (9 ;10 ;11 ;12 ;13 ;14 ;15).

Le circuit complet décodeur afficheur est donné comme suit :

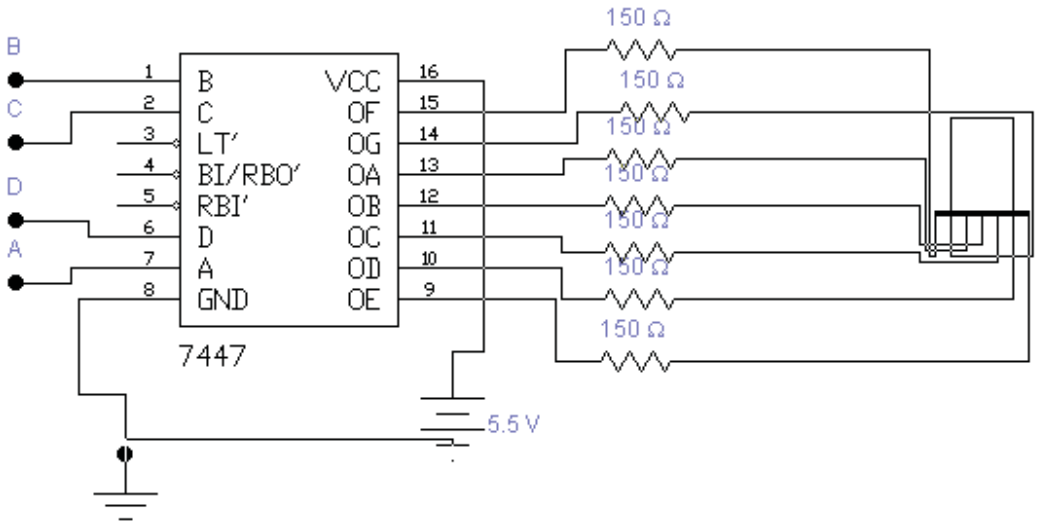
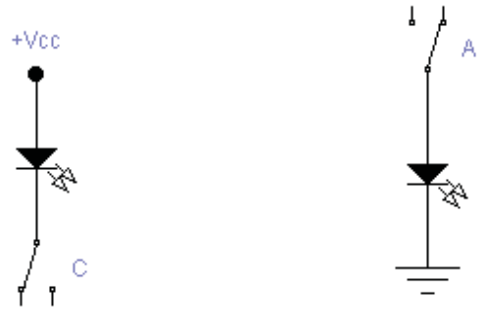


Figure 28 . Circuit du décodeur afficheur

On place des résistances (150 Ω environ) de limitation de courant (intensité de courant de 20mA correspondant au fonctionnement d'une LED).

Il existe 2 types d'afficheurs à 7 segments à LED : Afficheur 7 segments à anode commune, et des anafficheur 7 segments à cathode commune. Les afficheurs à anode commune sont actifs par des niveaux bas, tandis que les afficheurs à cathode commune sont actifs par des niveaux hauts.



a- Table de vérité d'un afficheur cathode commune

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

b- Diagramme de Karnaugh et équations :

AB / CD	00	01	11	10
00	1	0	Φ	1
01	0	1	Φ	1
11	1	1	Φ	Φ
10	1	1	Φ	Φ

D'où $\overline{a} = \overline{A}.B\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D = \overline{A}.\overline{C}.(B.\overline{D} + \overline{B}.D) = \overline{A}\overline{C}.(B \oplus D)$

AB / CD	00	01	11	10
00	1	1	Φ	1
01	1	0	Φ	1
11	1	1	Φ	Φ
10	1	0	Φ	Φ

D'où $\overline{b} = \overline{A}.B.\overline{C}.D + \overline{A}.B.C.\overline{D} = \overline{A}B(C \oplus D)$

AB / CD	00	01	11	10
00	1	1	Φ	1
01	1	1	Φ	1
11	1	1	Φ	Φ
10	0	1	Φ	Φ

D'où $\bar{c} = \overline{A}.\overline{B}.C.\overline{D}$

AB / CD	00	01	11	10
00	1	0	Φ	1
01	0	1	Φ	1
11	1	0	Φ	Φ
10	1	1	Φ	Φ

$\bar{d} = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.B.CD = \overline{A}.\overline{B}.\overline{C}..D + \overline{A}.B(\overline{C \oplus D})$

AB / CD	00	01	11	10
00	1	0	Φ	1
01	0	0	Φ	0
11	0	0	Φ	Φ
10	1	1	Φ	Φ

D'où $e = C.\overline{D} + \overline{B}.\overline{C}.\overline{D}$

AB / CD	00	01	11	10
00	1	1	Φ	1
01	0	1	Φ	1
11	0	0	Φ	Φ
10	0	1	Φ	Φ

D'où $f = A + \overline{C}.\overline{D} + B\overline{C}.D + B.C.\overline{D} = A + \overline{C}.\overline{D} + B(C \oplus D)$

AB / CD	00	01	11	10
00	0	1	Φ	1
01	0	1	Φ	1
11	1	0	Φ	Φ
10	1	1	Φ	Φ

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

Le logigramme d'un tel afficheur est le suivant.

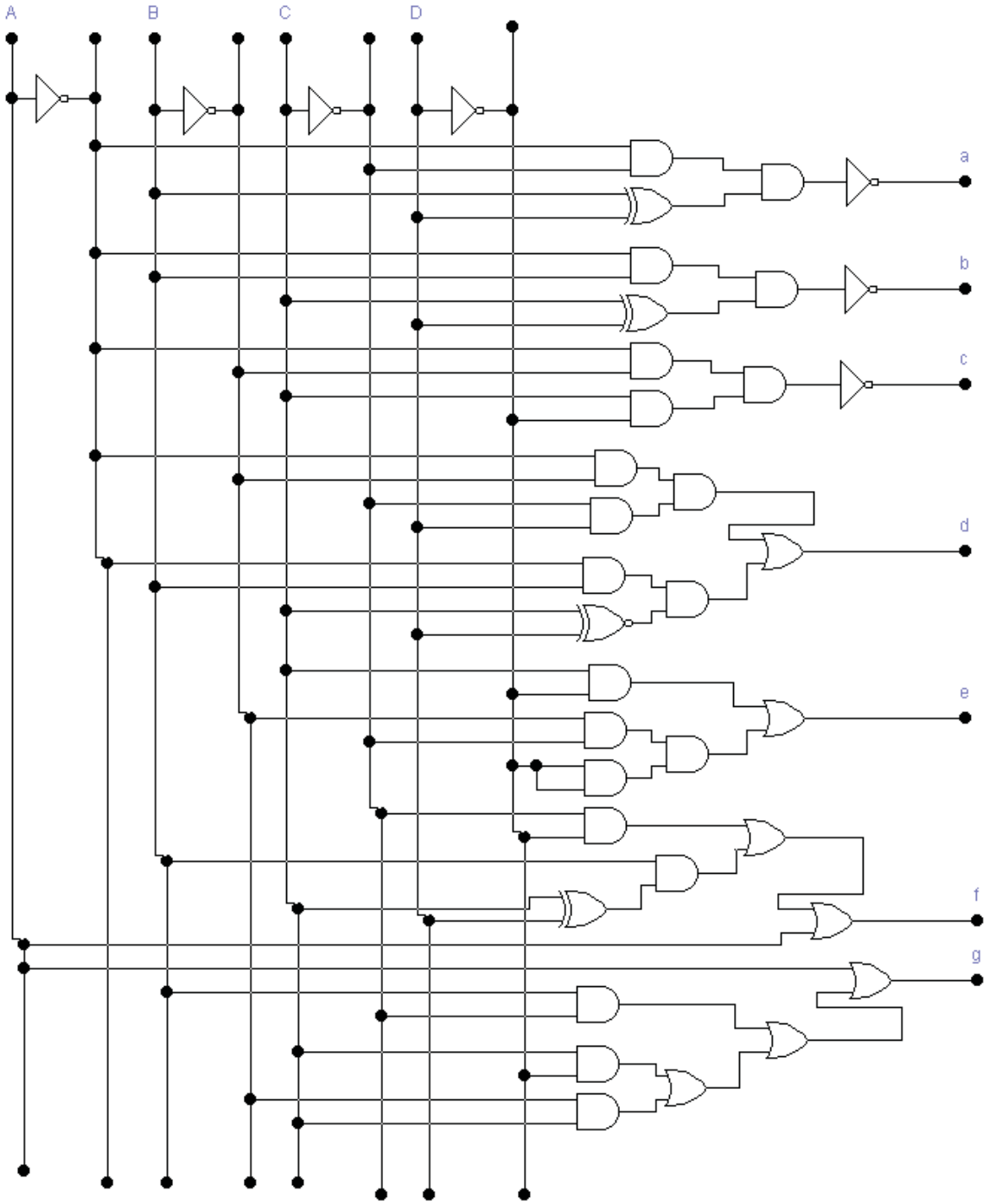


Figure 29 . Logigramme du décodeur BCD 7 segments

2.7.4.2 Afficheurs à cristaux liquides : (Liquid Cristal Display)

Définitions

Les afficheurs à cristaux liquides sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Ils sont relativement bons marchés et s'utilisent avec beaucoup de facilité.

Plusieurs afficheurs sont disponibles sur le marché et ne diffèrent les uns des autres, non seulement par leurs dimensions, (de 1 à 4 lignes de 6 à 80 caractères), mais aussi par leurs caractéristiques techniques et leurs tensions de service. Certains sont dotés d'un rétro éclairage de l'affichage. Cette fonction fait appel à des LED montées derrière l'écran du module, cependant, cet éclairage est gourmand en intensité (250 mA max.).

a- Introduction

Cristaux liquides : liquides à l'état mésomorphe ; c'est-à-dire un état intermédiaire entre l'état amorphe et l'état cristallin ; ils sont en particulier utilisés dans des fonctions d'affichage.

Propriétés et structures :

L'état mésomorphe fut découvert en 1888. on l'atteint par chauffage (cristaux liquides thermotropes) ou par dissolution (composés lyotropes) . Les molécules des cristaux liquides peuvent se déplacer les unes par rapport aux autres relativement facilement, comme les molécules d'un liquide. Cependant les molécules d'un cristal liquide ont tendance à s'orienter de la même façon, comme dans un cristal solide. le double comportement (liquide et solide) des cristaux ne s'observe que dans un certain domaine de température et de pression.

A des températures suffisamment élevées ou à de faibles pressions, l'orientation d »es molécules disparaît, provoquant la transformation du cristal liquide en liquide, A des températures suffisamment basses ou à des pressions suffisamment élevées ; les molécules d'un cristal liquide se déplacent difficilement les unes par rapport aux autres : le cristal liquide se solidifie.

Il existe plusieurs types de cristaux liquides dans les substances organiques : les cristaux liquides myéliniques, smectiques (structure en couche) et nématiques (liquides biréfringents), chacune de ces phases étant caractérisé par un arrangement différent des molécules dans l'espace.

On peut mettre en évidence les propriétés optiques d'un cristal liquide en lui appliquant un champ magnétique ou électrique qui modifie l'orientation de ses molécules. Par exemple, lorsqu'on applique un faible champ électrique à certains cristaux liquides, on observe un changement de teinte du cristal, qui passe d'une teinte claire à une teinte foncée . le cristal peut également acquérir la propriété de faire tourner le plan de polarisation de la lumière.

Utilisation :

Les cristaux liquides sont utilisés pour l'affichage , dans les écrans des montres digitales , des calculatrices , des petits téléviseurs , des ordinateurs portables , etc. les écrans à cristaux liquides utilisent souvent moins d'énergie que les écrans classiques , comme ceux qui utilisent des diodes.

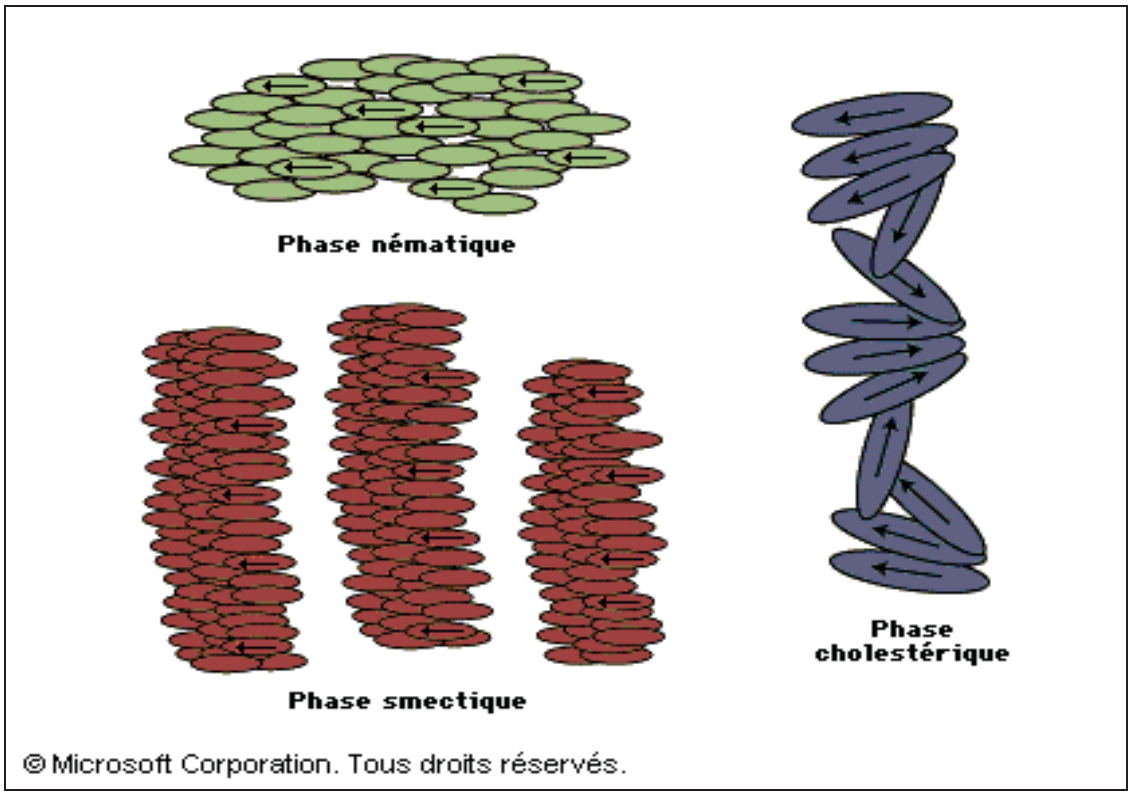
Certains cristaux liquides réfléchissent différentes longueurs d'onde de la lumière en fonction de l'orientation de leurs molécules, orientation qui dépend de la température.

Ainsi, ils sont utilisés pour la conception de thermomètres affichant des couleurs différentes selon la température du corps avec lequel le cristal liquide est en contact.

Parmi toutes les technologies d'écrans plats de visualisation, les écrans à cristaux liquides (Liquid Cristal Display) présentent un avantage majeur : ne consomment que peu d'énergie donc commandables par des circuits à transistor CMOS (Complementary Metal Oxide Silicium) basse tension ; de plus leurs excellente qualités électro-optiques, même à forte luminosité ambiante sont un attrait supplémentaire.

ces deux avantages ont conduit de nombreuses équipes à travers le monde à participer aux développement des LCD dés les années 1970 . dans cette première phase de développement (1970 à 1980) , le cristal utilisé est le nématique en hélice (ou TN : Twisted Nematic) et les propriétés des LCD sont simples : afficheurs pour montres et calculatrices de poches puis pour jeux électroniques à la fin des années 1970. Ces écrans sont directement multiplexés (ou écrans dits passifs).

Dans une deuxième phase à partir des années 1980 , les performances électro- optiques du cristal liquide sont améliorés , un nouveau type du cristal liquide dit STN (Super Twisted Nematic) est mis au point . il permet d'augmenter de façon importante le taux de multiplexage des écrans LCD passifs.



Les afficheurs à cristaux liquides sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Ils sont relativement bons marchés et s'utilisent avec beaucoup de facilité.

Un exceptionnel microprocesseur "pilote" de la famille C-MOS diminue considérablement leur consommation (inférieur à 0.1 mW). Ils sont pratiquement les seuls à être utilisés sur les appareils à alimentation par piles.

Plusieurs afficheurs sont disponibles sur le marché et ne diffèrent les uns des autres, non seulement par leurs dimensions, (de 1 à 4 lignes de 6 à 80 caractères), mais aussi par leurs caractéristiques techniques et leurs tension de service. Certains sont dotés d'un rétro éclairage de l'affichage. Cette fonction fait appel à des LED montées derrière l'écran du module, cependant, cet éclairage est gourmand en intensité (250 mA max.).

b) Principe de fonctionnement.

Schéma fonctionnel

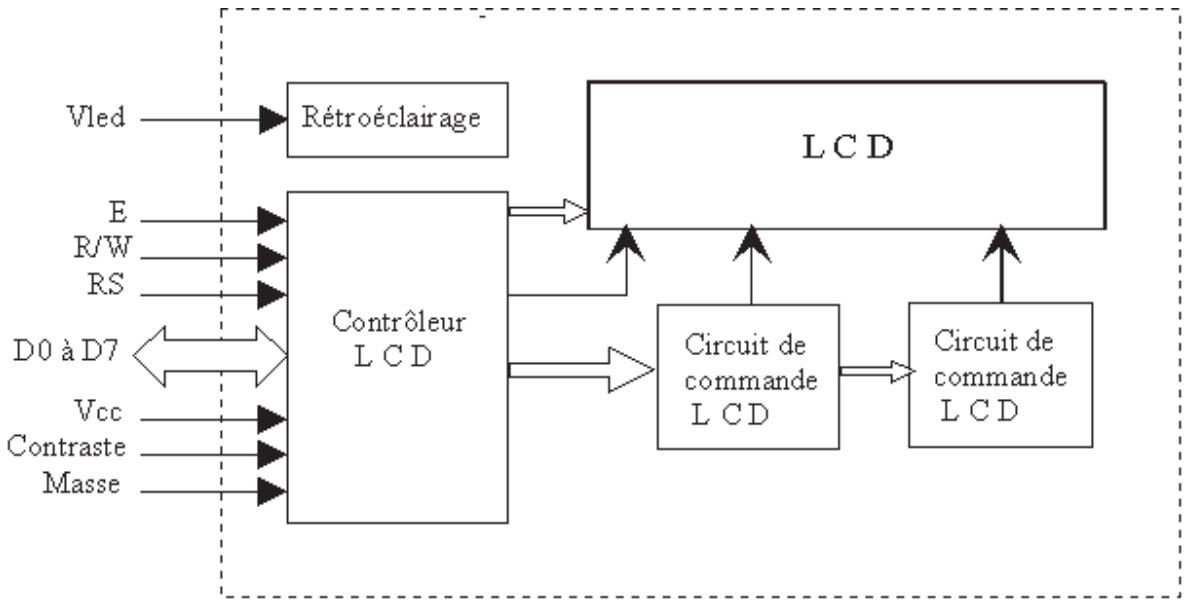


Figure 30. Composants de l'afficheur à cristaux liquides

Comme le montre le schéma fonctionnel, l'affichage comporte d'autres composants que l'afficheur à cristaux liquides (LCD) seul. Un circuit intégré de commande spécialisé, le LCD-controller, est chargé de la gestion du module. Le "contrôleur" remplit une double fonction: d'une part il commande l'affichage et de l'autre se charge de la communication avec l'extérieur.

c) Connexions

Les connexions à réaliser sont simples puisque l'afficheur LCD dispose de peu de broches. Il faut, évidement, l'alimenter, le connecter à un bus de donnée (4 ou 8 bits) d'un microprocesseur, et connecter les broches Enable (validation), Read/Write (écriture/lecture) et Register Select (instruction/commande).

2.7.4.3 Principe des cristaux liquides

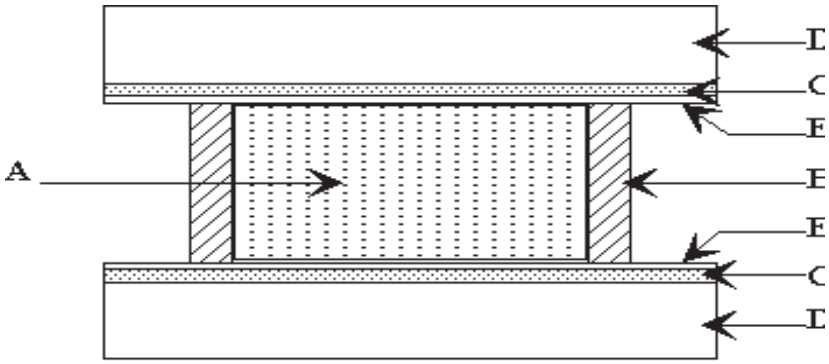
L'afficheur est constitué de deux lames de verre, distantes de 20 µm environ, sur lesquelles sont dessinées les mantisses formant les caractères. L'espace entre elles est rempli de cristal liquide normalement réfléchissant (pour les modèles réflexifs).

L'application entre les deux faces d'une tension alternative basse fréquence de quelques volts (3 à 5 V) le rend absorbant. Les caractères apparaissent sombres sur fond clair.

N'émettant pas de lumière, un afficheur à cristaux liquides réflexif ne peut être utilisé qu'avec un bon éclairage ambiant. Sa lisibilité augmente avec l'éclairage.

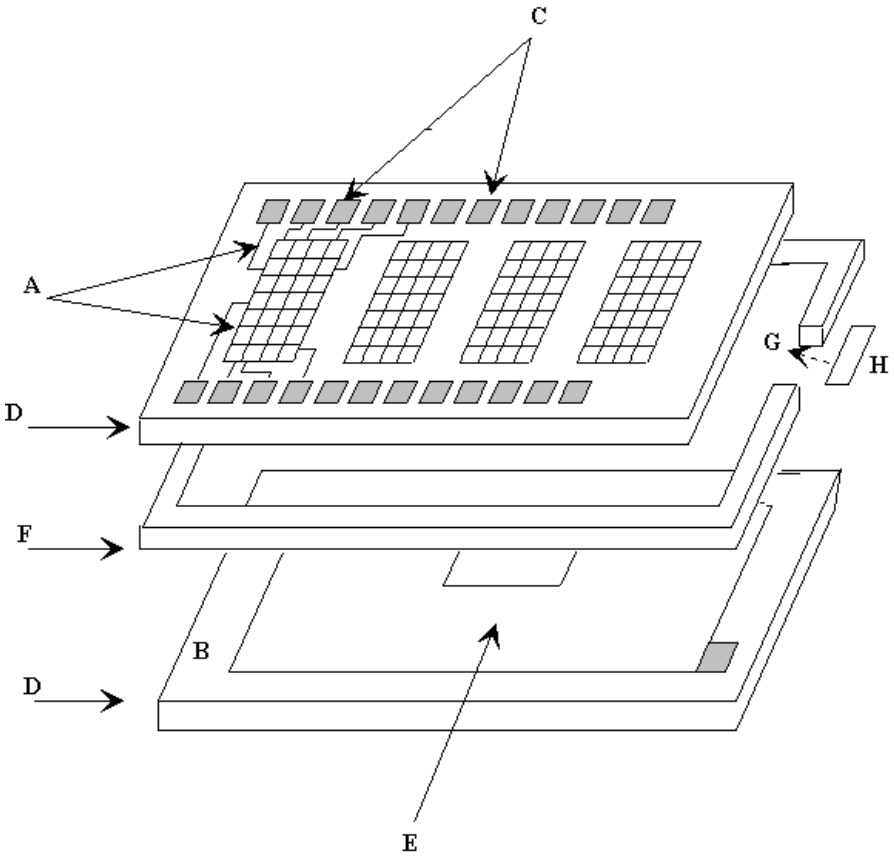
Les modèles transmissifs fonctionnent différemment: normalement opaque au repos, le cristal liquide devient transparent lorsqu'il est excité; pour rendre un tel afficheur lisible, il est nécessaire de l'éclaire par l'arrière.

a) Constitution d'une cellule à cristal liquide

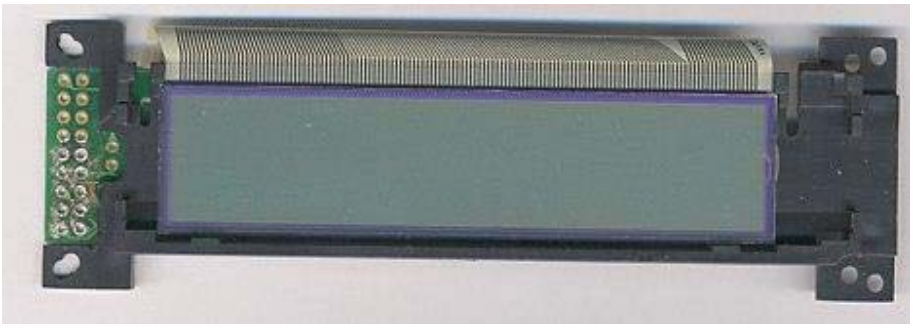


A: cristal liquide
B: cales d'épaisseur
C: électrodes transparentes
D: substrat (verre)
E: couche d'orientation (polyimide frotté)

b) Constitution d'un afficheur à cristaux liquides



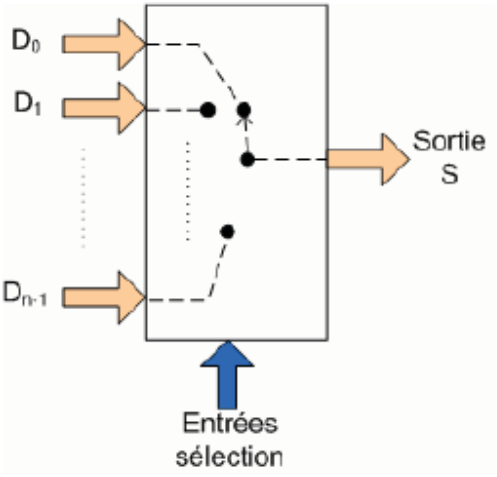
A: électrodes en ITO
B: cristal liquide
C: contacts
D: couches d'alignements
E: contre élecctrode
F: joint organique
(cale d'épaisseur)
G: orifice de remplissage
H: bouchon



2.7.5 Multiplexage / Démultiplexage

2.7.5.1 Les multiplexeurs

Le multiplexeur permet de sélectionner une variable logique choisie parmi 2n variables logique. C'est pour cette raison qu'il est aussi appelé sélecteur de données. L'aiguillage de l'entrée de données qui nous intéresse sur la sortie est commandé par des entrées de sélection appelées des entrées d'adresse.



Le multiplexage consiste donc à sélectionner une voix parmi plusieurs par laquelle transitent les informations à transmettre. Le circuit sélectionneur de données ou multiplexeur (MUX) est donc analogue à un commutateur unidirectionnel (le transfert n'est possible que des entrées du circuit vers l'unique sortie). L'aiguillage de l'information est assuré par un décodeur d'adresse interne au circuit.

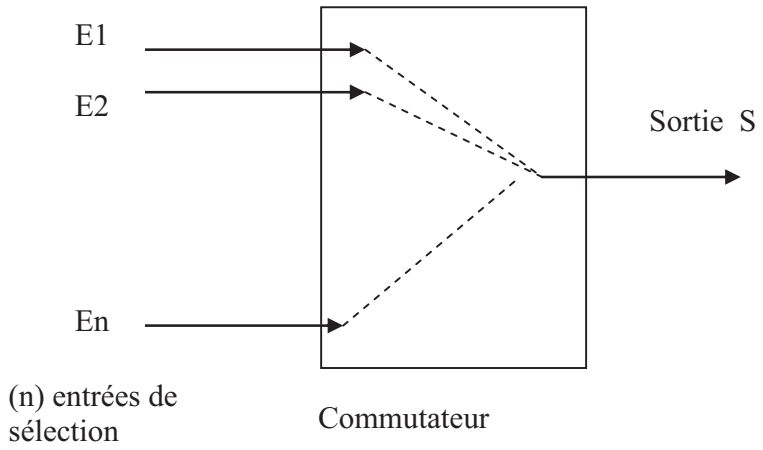
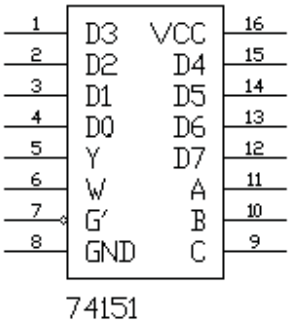


Figure 31. Schéma synoptique d'un multiplexeur à n entrées

Exemple :

Soit le circuit **intégré 74LS 151** (multiplexeur à 8 voies)



Les entrées de données du circuit multiplexeur se trouvent sur les pins (1 ;2 ;3 ;4 ; 12 ;13 ; 14 ; 15) , les entrées de sélection aux pins (9 ;10 ; 11)

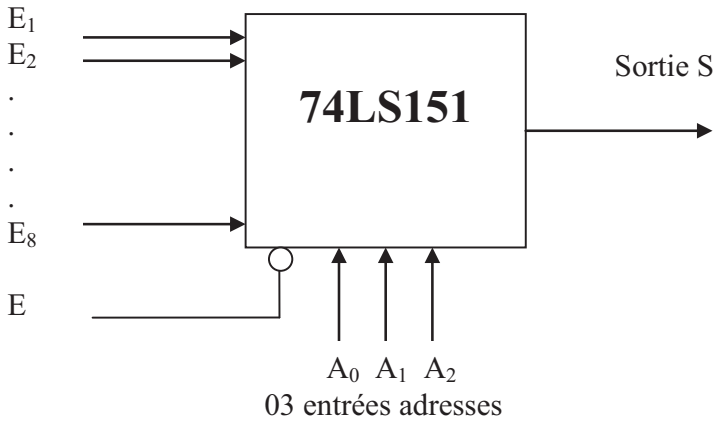


Figure 32. Schéma synoptique d'un multiplexeur à 8 entrées

a- Table de vérité

Entrées sélection A B C	Strobe S (validation)	Sorties Y
* * *	H	L
L L L	L	D_0
L L H	L	D_1
L H L	L	D_2
L H H	L	D_3
H L L	L	D_4
H L H	L	D_5
H H L	L	D_6
H H H	L	D_7

b- Equation

$$Y = V[\overline{A}.\overline{B}.\overline{C}.D_0 + \overline{A}.\overline{B}.C.D_1 + \overline{A}.B.\overline{C}.D_2 + \overline{A}.B.C.D_3 + A.\overline{B}.\overline{C}.D_4 + A.\overline{B}.C.D_5 + AB.\overline{C}.D_6 + ABC.D_7]$$

c- **Logigramme**

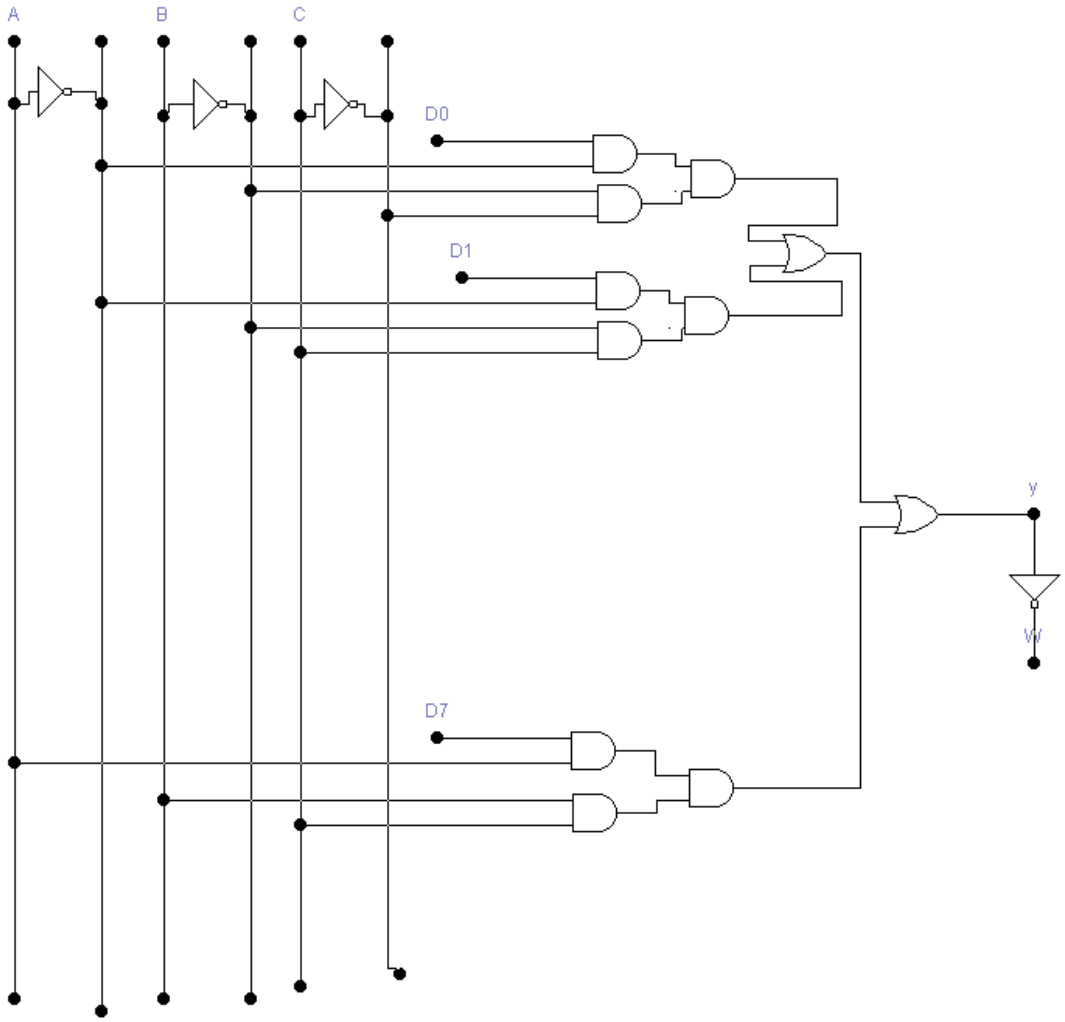


Figure 33. Logigramme du multiplexeur 8 vers 1

Remarque :

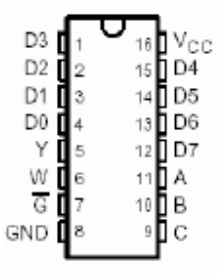
Le multiplexeur possède plusieurs applications ; parmi lesquelles :

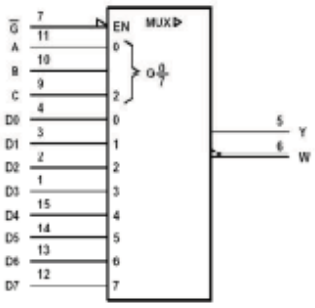
a- un multiplexeur peut être utilisé pour transmettre une donnée binaire parallèle sous forme sérielle. ceci est réalisé en branchant un compteur sur les entrées de sélection de voies.

b- résolution des problèmes de logique combinatoire.

Exemple pratique :

Extrait de la documentation technique du décodeur 74HC151 (Texas Instrument)



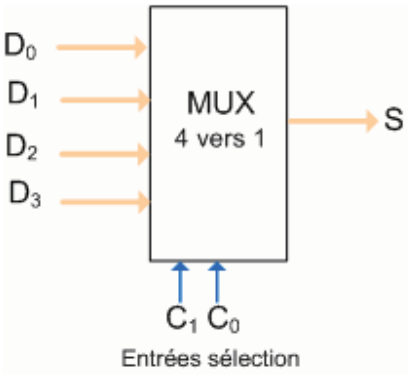


FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE $\overline{G}$	Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

D0, D1 . . . D7 = the level of the respective D input

Description d'un multiplexeur 4 entrées / 1 sortie



C_1	C_0	S
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

2.7.5.2 les démultiplexeurs

Le démultiplexeur est un circuit logique qui comporte une seule entrée information ; n entrées adresse et 2n sorties. L'information présente à l'entrée est dirigée vers une seule sortie, celle déterminée par les lignes de sélection. L'aiguillage de l'information est aussi assuré par un décodeur d'adresse interne au circuit, comme pour le multiplexeur.

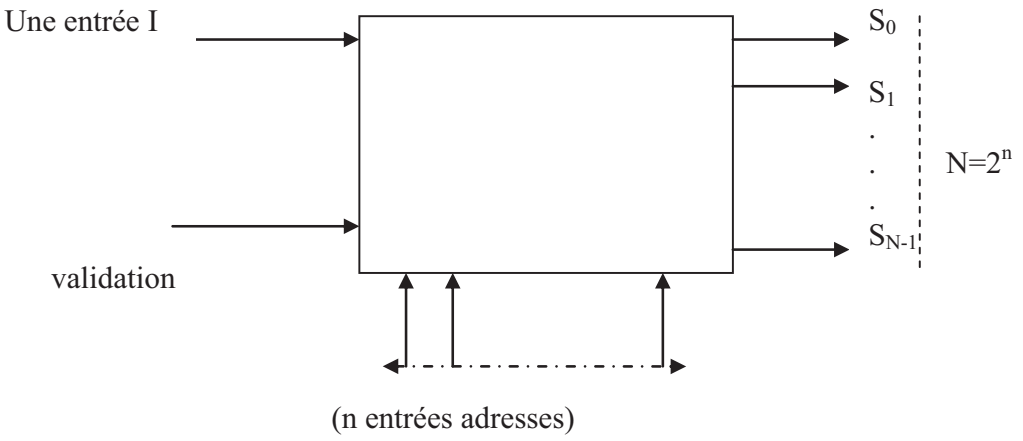


Figure 34. Schéma synoptique d'un démultiplexeur 1 vers N sorties

Exemple
Faire la synthèse d'un démultiplexeur 1 vers 4 (1 entrée vers 04 sorties) : le nombre de lignes d'adresses =2.

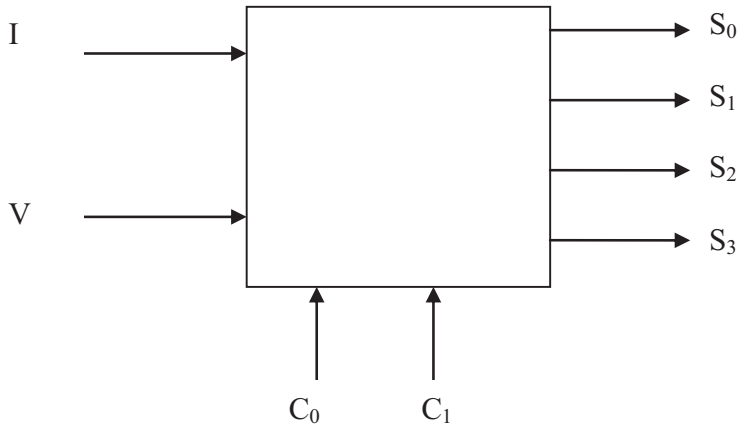


Figure 35. Schéma synoptique d'un démultiplexeur 1 vers 4

a- Table de vérité

V	C_1	C_0	I	S_0	S_1	S_2	S_3
1	0	0		1			
1	0	1			1		
1	1	0				1	
1	1	1					1

b- Equations

$S_0 = \overline{C_0}.\overline{C_1}.I.V$

$S_1 = C_0.\overline{C_1}.I.V$

$S_2 = \overline{C_0}.C_1.I.V$

$S_3 = C_0.C_1.I.V$

c- Logigramme

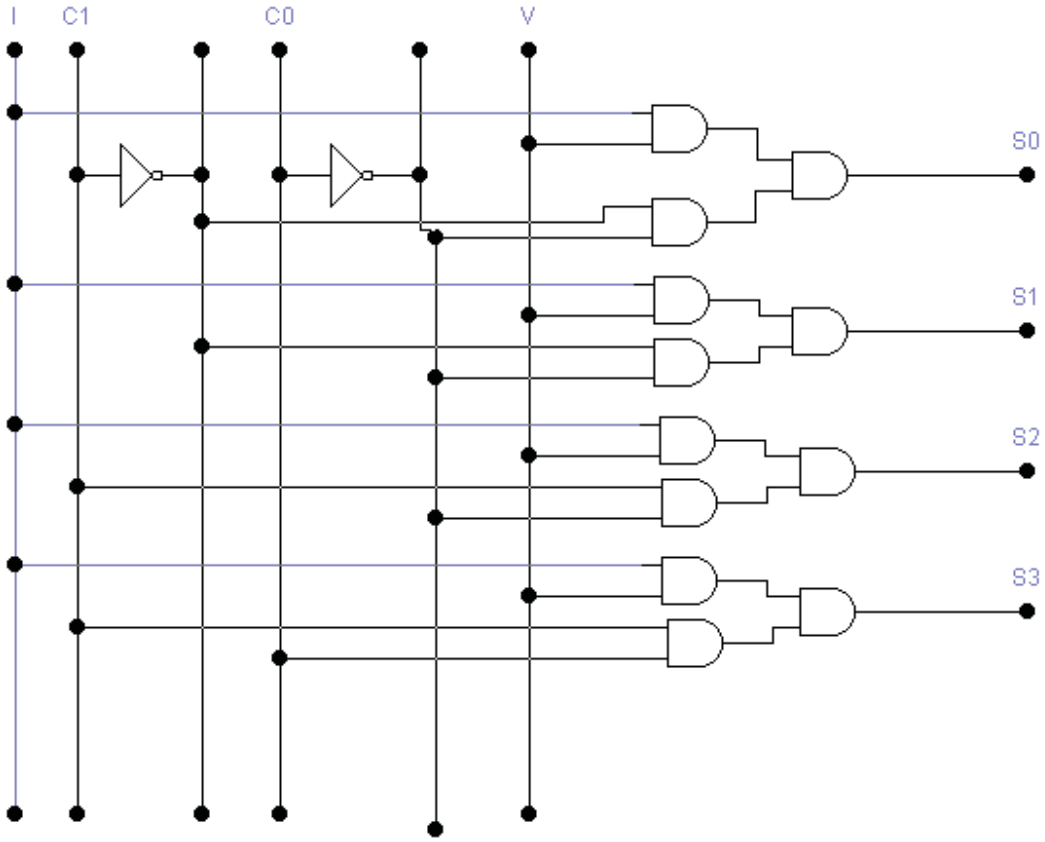
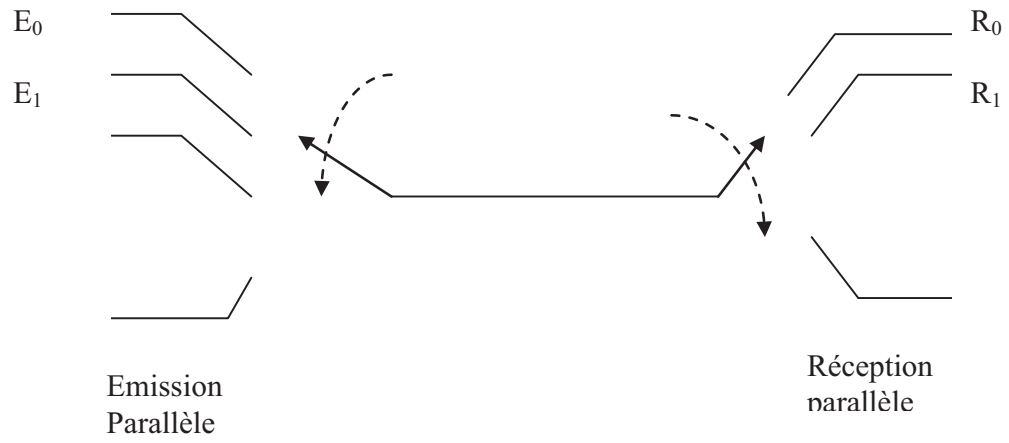


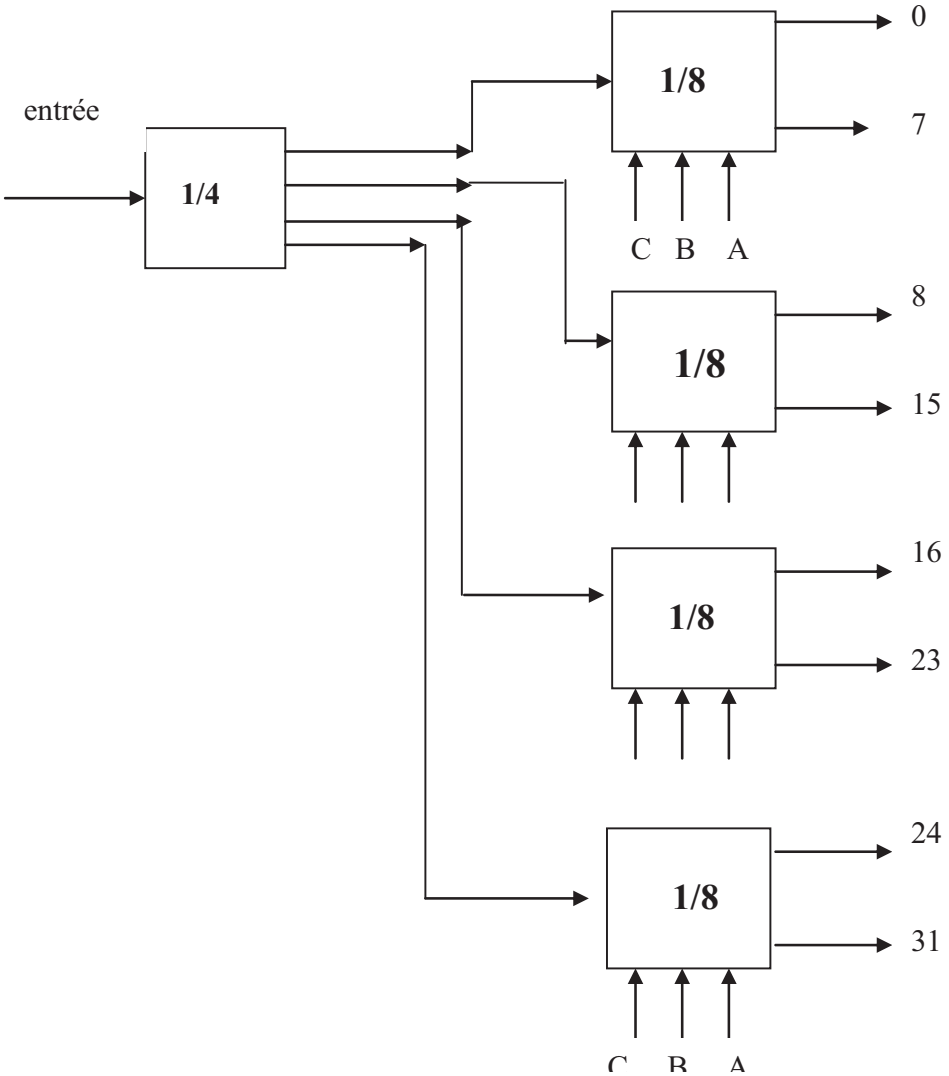
Figure 36. Logigramme d'un démultiplexeur 1 vers 4

2.7.5.3 Application des multiplexeurs / démultiplexeurs

Si l'on envoie à distance les informations issues d'un grand nombre de sources différentes ; on multiplexe ces informations pour les transmettre en série sur une seule ligne (conversion parallèle – série) ; à l'autre extrémité de la ligne ; il faut démultiplexer ; c'est-à-dire restituer les informations sur les récepteurs homologues des émetteurs ; le démultiplexeur est réalisé à l'aide des décodeurs.



La figure suivante montre le schéma d'un démultiplexeur 1 voie d'entrée, 32 voies de sorties et 5 entrées d'adresses EDCBA : un décodeur 1 parmi 4 aiguille l'information vers l'entrée de l'un des 4 décodeurs 1 parmi 8.
Chacune des 4 combinaisons de ED choisit un de ces décodeurs c'est-à-dire un groupe de 8 sorties, ensuite chacune des 8 combinaisons de CBA choisit l'une de ces huit sorties. Reste à résoudre le problème de la distribution des horloges, ce qui est un problème séquentiel.



- **Application des multiplexeurs**

a- Conversion parallèle - série

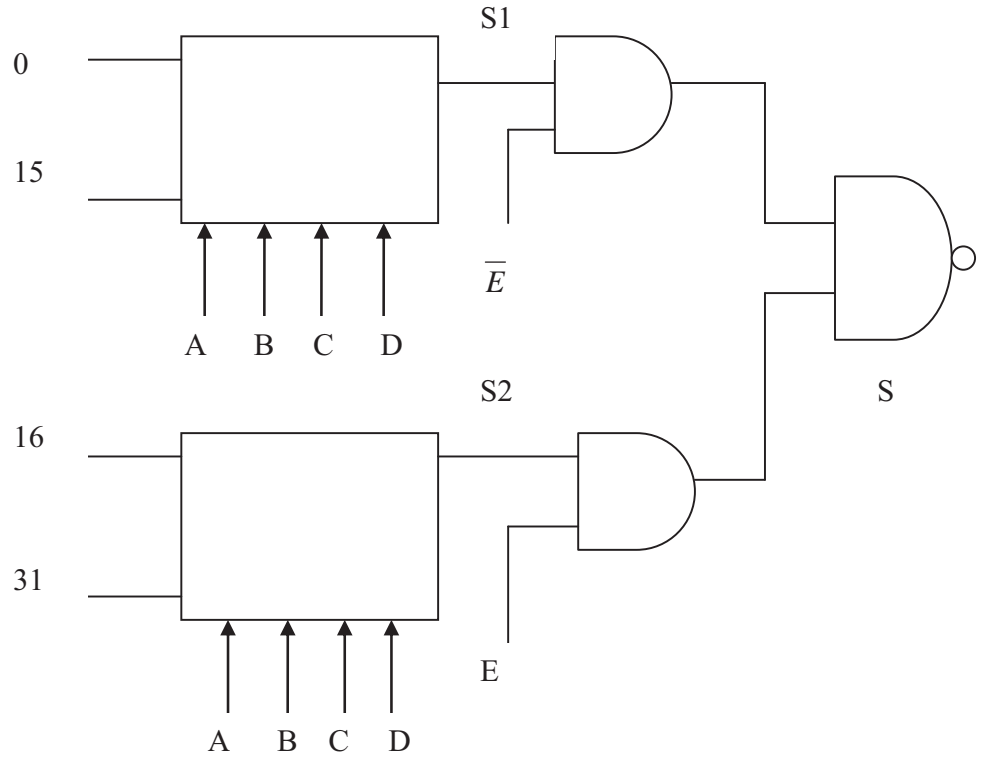


Figure 37. Conversion parallèle série

La sortie S peut s'écrire : $S = S_1.\overline{E} + S_2.E$

On dispose de 32 bits en parallèle et on veut les transmettre en série ; il suffit de placer les 32 bits aux entrées (0,1,…….31) de 02 multiplex à 16 entrées et d'utiliser 5 bits d'adresses A , B, C, D (E et $\overline{E}$) : les poids forts (E et $\overline{E}$) sélectionnent l'un des 02 multiplexeurs ; les poids faibles (DCBA) sélectionnent l'une des 16 adresses du MUX concerné .
Pour obtenir successivement les 32 adresses on utilise un compteur à 5 bits qui donne EDCBA= 0,1,2……31

b- Génération de fonctions

Un MUX peut être utilisé pour générer des fonctions logiques.
Avec un multiplexeur à 8 entrées (SN 74151) on a pour le signal de sortie :

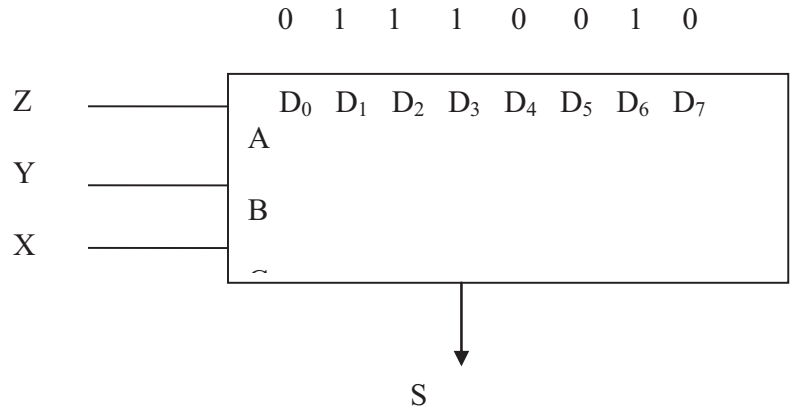
$$S = \overline{C}.\overline{B}.\overline{A}.D_0 + \overline{C}.\overline{B}.A.D_1 + \overline{C}.B.\overline{A}.D_2 + \overline{C}.BA.D_3 + C\overline{B}.\overline{A}.D_4 + C\overline{B}.A.D_5 + CB.\overline{A}.D_6 + CBA.D_7$$

(on a 8 entrées informations D_0 ; D_1 ; ……D_7 et 3 entrées adresses CBA)

Pour réaliser la fonction :

$$S = \overline{X}.\overline{Y}.Z + \overline{X}.Y.\overline{Z} + \overline{X}.Y.Z + XY.\overline{Z}$$

il suffit de placer :
X sur l'entrée C
Y sur l'entrée B
Z sur l'entrée A
1 sur les entrées D_1 ; D_2 ; D_3 ; D_6
0 sur les entrées D_0 ; D_4 ; D_5 ; D_7



Remarque :

Multiplexer des données (se présentent en parallèle) ; consiste à un instant donné ; à dissocier dans le temps l'envoi de ces données sur un seul conducteur. Le multiplexage vise à économiser les voies des transmissions.
Démultiplexer les données consiste à trier ces données en fonction du temps pour les affecter aux récepteurs qui leurs sont destinées.

Exercices d'application
Algèbre de Boole et simplification des fonctions logiques

EXERCICE N°1

Démontrez les propriétés suivantes

1- X(X'+Y)= XY
2- (X+Y). (X+Z)= X+YZ
3- XY+XY'=X
4- (X+Y). (X+Y')=X

EXERCICE N°2

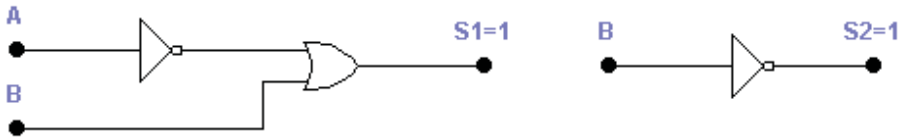
Appliquer les fonctions logiques suivantes sur les deux nombres X et Y donnés sur 16 bits en base 2 :

X= 0111 1101 1111 0011
Y= 1010 1011 1100 1101

a) ET (AND)
b) OU (OR)
c) Ou exclusif
d) Addition des nombres : X+Y
e) Soustraction des deux nombres : X-Y

EXERCICE N°3

Quelles doivent être les valeurs de A et B pour que les sorties des 2 circuits suivants possèdent la même valeur 1 ?



EXERCICE N°4

1- Réduire les fonctions suivantes:

$F1 = (A\overline{B} + C).(A + \overline{B})C$

$F2 = C + AB + AD(B + \overline{C}) + CD$

$F3 = A\overline{B} + A\overline{C} + \overline{BC}$

$F4 = A + ABC + \overline{A}C$

2. Démontrer :

A+AB=A

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

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

$AC + \overline{A}BC = AC + BC$

$AB + AC + \overline{B}C = AB + \overline{B}C$

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

3. Simplifier :

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

4. Réaliser les logigrammes des fonctions suivantes

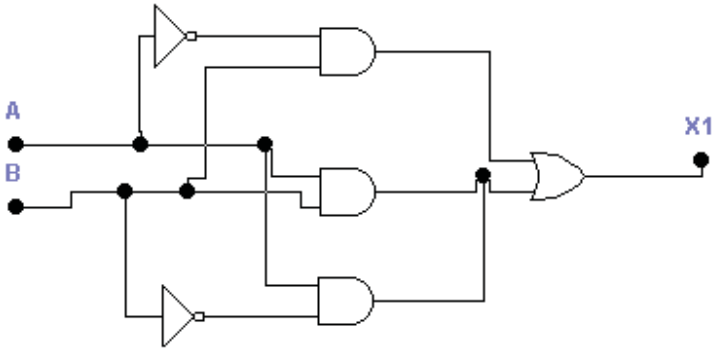
$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{C}\,D$ avec 3 portes NOR à 2 entrées

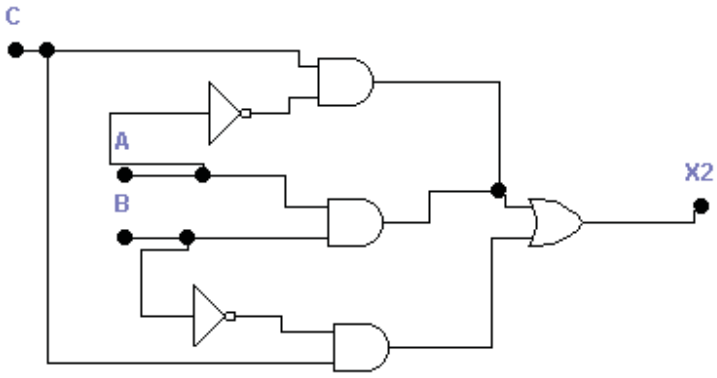
G=A(B+C) avec 3 portes NAND à 2 entrées

$H = AB + BC + AC$ avec des portes Nand à 2 entrées

EXERCICE N°5

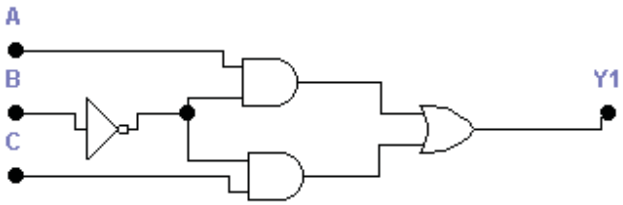
Simplifier les circuits combinatoires suivants :

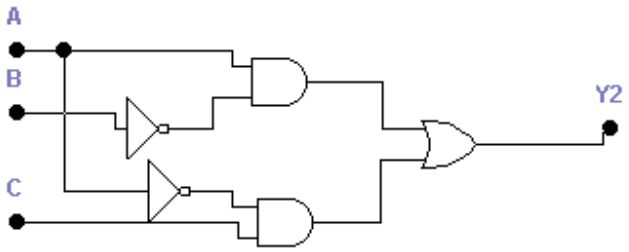




EXERCICE N°6

Soit les circuits suivants :

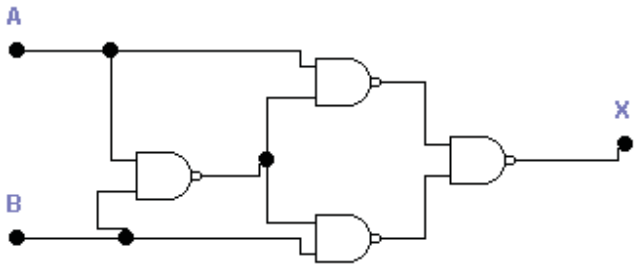




1- Donnez les expressions booléennes que représentent ces circuits ?
2- Calculez la table de vérité de chacun ?

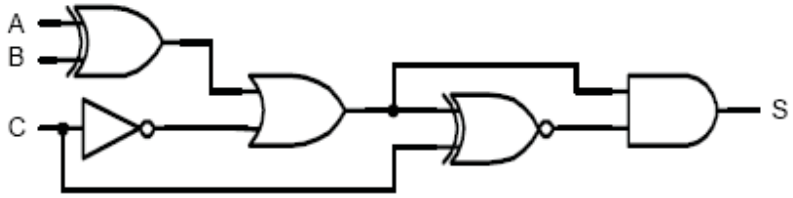
EXERCICE N°7

a- Montrez que le circuit logique suivant formé de portes Nand est une représentation graphique du OU-exclusif (XOR).
b- Remplacer les portes NAND par des portes NOR et montrez que la fonction ainsi obtenue n'est autre que l'inverse du OU-exclusif.



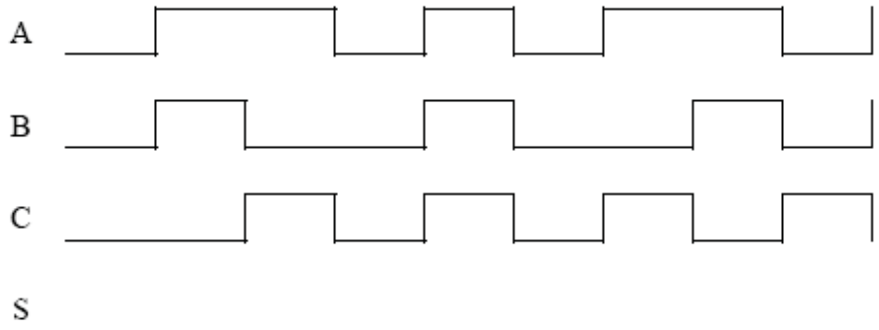
Exercice N°8

1. Complétez la table de vérité correspondante au circuit logique suivant :



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

2. Extraire l'équation de S à partir de la table de vérité.
3. Complétez le chronogramme suivant :



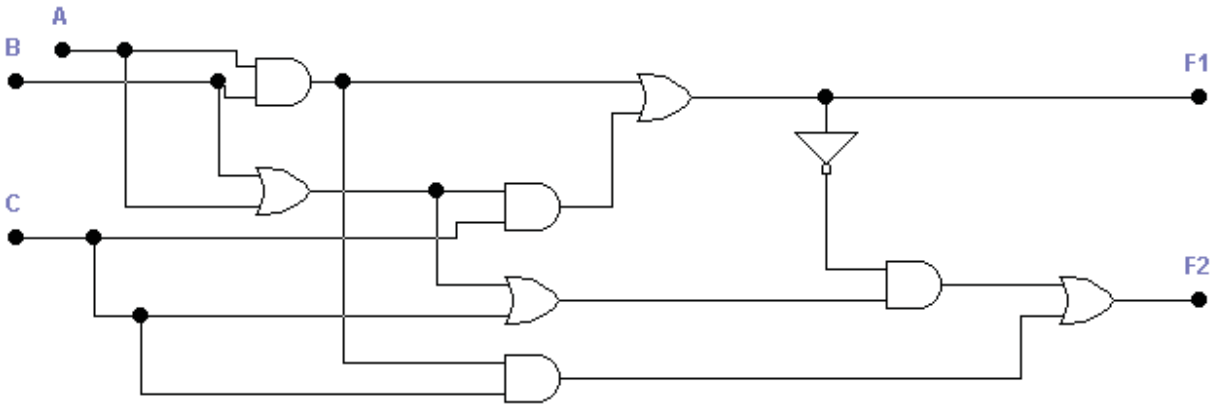
Partie II : Simplification par Karnaugh

EXERCICE N°9

1. Soit la fonction $F = AB\overline{C}D + AC\overline{D} + \overline{AB}C + \overline{AB}\,\overline{D} + \overline{AB}\,\overline{C} + A\,\overline{BD}$

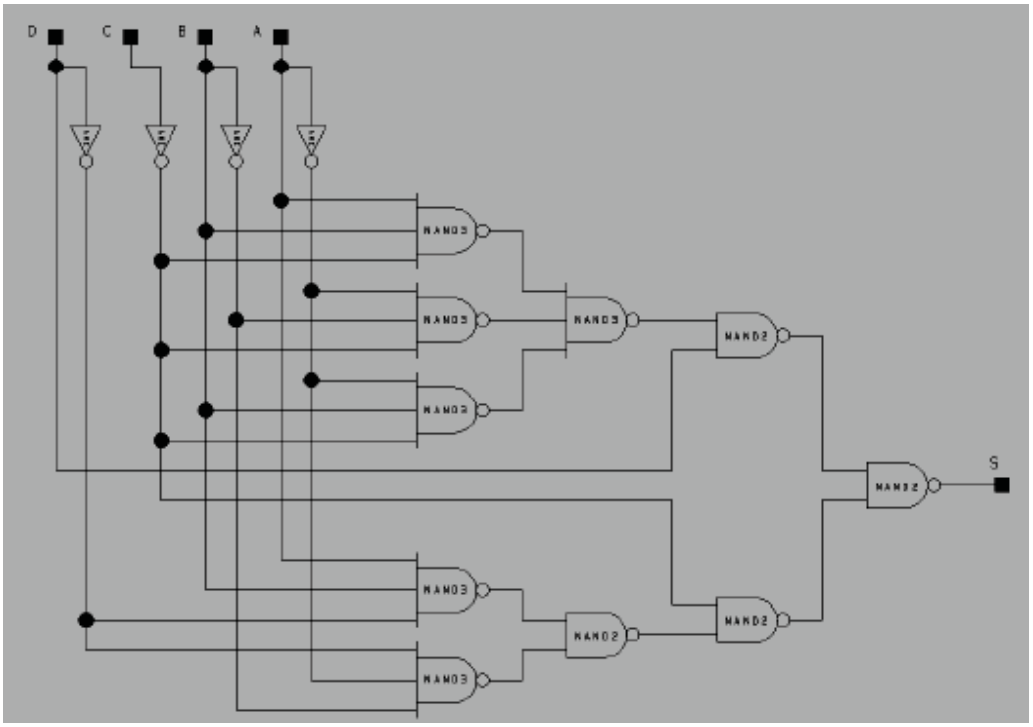
a- Ecrire l'équation sous forme canonique.
b- Simplifiez avec une table de Karnaugh.

2. Soit le circuit suivant :



a- Ecrire les équations des sorties.
b- Réduire les fonctions et dessiner le nouveau schéma.

3. Soit le schéma suivant :



a) Écrire la fonction logique sous forme de somme de produits.
b) Simplifier la fonction avec une table de Karnaugh sachant que ABCD ne prennent jamais la valeur 1001.
c) Dessiner le schéma de la fonction S.

EXERCICE N°10

1. En utilisant les diagrammes de Karnaugh, simplifier les fonctions complètement définies :

$F1 = ab\bar{c} + \bar{a}\bar{b}c + a\bar{b}\bar{c} + a\bar{b}c$

$F2 = abc + \bar{a}bc + \bar{a}\bar{b}c + ab\bar{c}$

$F3 = \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}\bar{d} + a\bar{b}c\bar{d} + \bar{a}\bar{b}\bar{c}\bar{d} + ab\bar{c}d + \bar{a}bcd + a\bar{b}\bar{c}\bar{d}$

$F4 = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}\bar{b}cd + \bar{a}\bar{b}c\bar{d} + ab\bar{c}d + abcd + a\bar{b}\bar{c}d + a\bar{b}\,cd + a\bar{b}c\bar{d}$

$F5 = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + abc\bar{d} + abcd + a\bar{b}\bar{c}\bar{d} + a\bar{b}c\bar{d}$

EXERCICE N°11

Soit la fonction suivante, donnée sous forme décimale :

$f(a,b,c,d) = \sum_{1}(0,1,2,7,8,9,10,15) + \sum_{\phi}(5,6,12)$

Simplifiez par la méthode de Karnaugh ?
Dessinez le logigramme ?

EXERCICE N°12

Soit la fonction incomplète définie par :

$F(A,B,C,D) = \sum(0,3,6,7,8,10,12,13)$

et

$\overline{F}(A,B,C,D) = \sum(4,5,9,14,15)$

1) Simplifiez la fonction F en utilisant la méthode de karnaugh ?
2) Réalisez la fonction F avec des portes Nand à 2 entrées ?

EXERCICE N°13

Simplifiez à l'aide des tableaux de Karnaugh les expressions suivantes :

$F(A,B,C,D) = \sum_{1}(2,6,7,8,10) + \sum_{\phi}(0,12,13,15)$

$F = \prod(0,2,4,6)$

$F = \prod(2,5,6,7)$

EXERCICE N°14

Soient

$F1(A,B,C,D) = (A.\overline{D}).(D+B).(A+\overline{B}+\overline{C})$

$F2(A,B,C,D) = (\overline{A}+B).(\overline{A}+\overline{B}+D).(A+B+\overline{C}+\overline{D})$

a- donner les deux formes canoniques de F1 et F2, puis $\overline{F1}$ *et* $\overline{F2}$
b- simplifiez F1 et F2 à l'aide du diagramme de karnaugh ?
c- Donnez les formes simplifiées de $\overline{F1}$, $\overline{F2}$ *et* $f = F1 + F2$?