CHAPITRE I – SYSTEMES DE NUMERATION

I- GÉNÉRALITÉS

Un système de numération est un système de comptage et de représentation des nombres. Il est généralement pondéré.
La définition d'un système de numération pondéré repose sur trois notions :

- **La base** : C'est un entier naturel supérieur ou égal à 2 désignant le type de système de numération.
- **Les digits** : Ce sont les symboles, tous différents d'une base B donnée. Dans une base B on dénombre au total B digits.
- **Le poids** : Il est déterminé en fonction de la position de chaque digit lorsqu'on écrit un nombre dans une base B donnée.

I-1- Notion de base

Les systèmes de numération pondérés les plus utilisés sont :

Base	Symboles ou Digits	Autre appellation
Base 2	(0; 1)	Système Binaire
Base 8	(0; 1; 2; 3; 4; 5; 6; 7)	Système Octal
Base 10	(0; 1; 2; 3; 4; 5; 6; 7; 8; 9)	Système Décimal
Base 16	(0; 1; 2; 3; 4; 5; 6; 7; 8; 9; A; B; C; D; E; F)	Système Hexadécimal

Compte tenu du fait que certains digits appartiennent à plusieurs bases, il est indispensable lorsqu'on travaille sur différentes bases, d'indiquer la base dans laquelle chaque nombre est exprimé.
Les différentes notations utilisées pour spécifier la base dans laquelle un nombre (par exemple pour le nombre 1001) est exprimé sont :

En binaire : $(1001)_2$ ou 1001B ou %1001.

En octal : $(1001)_8$ ou 1001@.

En hexadécimal : $(1001)_{16}$ ou 1001H ou $1001.

En décimal : $(1001)_{10}$ ou 1001 sans aucune indication.

I-2- Notion de digit

Par exemple :

- Le nombre $(574)_8$ est formé de 3 digits octaux (ou 3 éléments octaux ou 3 chiffres octaux).
- Le nombre $(9862)_{10}$ est formé de 4 digits décimaux (ou 4 éléments décimaux ou 4 chiffres décimaux).
- Le nombre $(30EA7)_{16}$ est formé de 5 digits hexadécimaux (ou 5 éléments hexadécimaux ou 5 chiffres hexadécimaux).
- Le nombre $(1011)_2$ est formé de 4 digits binaires (ou 4 éléments binaires ou 4 chiffres binaires).

*NB : Un digit binaire est conventionnellement appelé **bit** (venant de binary digit). On dira alors du nombre binaire de l'exemple ci-dessus qu'il est formé de **4 bits**.*

I-3- Notion de poids

L'écriture d'un nombre dans une base donnée consiste à associer plusieurs digits de cette base dans un ordre bien déterminé.
Considérons une base B et **xyzt,uv**, un nombre à virgule de cette base. Chaque digit de ce nombre intervient avec un **poids** selon son rang (sa position). La numérotation positionnelle se fait pour la partie entière, de la droite vers la gauche évoluant par 0, 1, 2, 3, 4, 5, ... et pour la partie après la virgule de la gauche vers la droite en évoluant par -1, -2, -3, -4, -5, ...

Pour chaque rang *r* le poids *p* correspondant est : $p = B^r$

Ainsi pour le nombre xyzt,uv écrit dans la base B on a :

Nombre :	x	y	z	t,	u	v
Rang :	3	2	1	0	-1	-2
Poids	B^3	B^2	B^1	B^0	B^{-1}	B^{-2}

Exemples :

- Soit le nombre $(CAFE, 5)_{16}$. Les différents poids sont : 16^3 ; 16^2 ; 16^1 ; 16^0 ; 16^{-1} c'est-à-dire 4096 ; 256 ; 16 ; 1 ; 0,0625.
  Dans ce nombre le digit C a un poids de 4096, c'est le **digit le plus significatif** (en anglais, **Most Significant Digit** ; **MSD**) et le digit 5 a un poids de 0,0625, c'est le **digit le moins significatif** (en anglais, **Less** ou **Least Significant Digit** ; **LSD**)
- Soit le nombre $(1100)_2$ on a :

Nombre :	1	1	0	0
Poids :	2^3	2^2	2^1	2^0
	MSB			LSB

MSB : Most Significant Bit (car le digit binaire est nommé bit)
LSB : Least Significant Bit

II- COMPTAGE DANS LES BASES

Lorsqu'on compte au moyen du système décimal par exemple, on commence par 0 et on progresse en utilisant chacun des symboles jusqu'à 9. Après 9 on revient à 0 et on augmente d'un pas le digit de rang immédiatement supérieur.
Pour compter dans n'importe quelle base, c'est le même principe qui est adopté. Tout est une question d'utilisation entière des symboles de la base pour le rang inférieur avant que le rang immédiatement suivant ne fasse un pas.

Scanned by CamScanner

Le tableau ci- dessous présente la suite des valeurs de 0 à 33 dans les systèmes décimal, hexadécimal, octal et binaire

DECIMAL	HEXADECIMAL	OCTAL	BINAIRE
00	00	00	000000
01	01	01	000001
02	02	02	000010
03	03	03	000011
04	04	04	000100
05	05	05	000101
06	06	06	000110
07	07	07	000111
08	08	10	001000
09	09	11	001001
10	0A	12	001010
11	0B	13	001011
12	0C	14	001100
13	0D	15	001101
14	0E	16	001110
15	0F	17	001111
16	10	20	010000

DECIMAL	HEXADECIMAL	OCTAL	BINAIRE
17	11	21	010001
18	12	22	010010
19	13	23	010011
20	14	24	010100
21	15	25	010101
22	16	26	010110
23	17	27	010111
24	18	30	011000
25	19	31	011001
26	1A	32	011010
27	1B	33	011011
28	1C	34	011100
29	1D	35	011101
30	1E	36	011110
31	1F	37	011111
32	20	40	100000
33	21	41	100001

Remarque

- Tout nombre s'écrit **10** dans sa propre base.
- Le plus grand nombre formé de n digits dans une base B donnée est $B^n - 1$ et l'intervalle des nombres entiers représentables est **[0 ; $B^n - 1$]** ; soit alors B^n nombres possibles.

Exemples :

- Le plus grand nombre binaire représenté sur 5 bits est 11111. D'après le tableau de comptage, $(11111)_2 = (31)_{10}$.
On a bien $31 = 2^5 - 1$.
Tous les nombres binaires de 5 bits qu'il est possible d'écrire sont dans l'intervalle $[(0000)_2 ; (11111)_2]$ c'est à dire [0 ; 31] ; soit au total $2^5 = 32$ nombres possibles.

- En base 10, sur 3 digits le plus grand nombre est 999. On note bien que $999 = 10^3 - 1$.
L'intervalle de tous les nombres décimaux de 3 digits est [0 ; 999] soit au total $10^3 = 1000$ nombres possibles

III- TRANSCODAGE OU CHANGEMENT DE BASE

Le transcodage c'est la conversion d'un nombre exprimé dans une base B_1 vers sont équivalent dans une base B_2.

III-1- Conversion de la base décimale vers une autre base

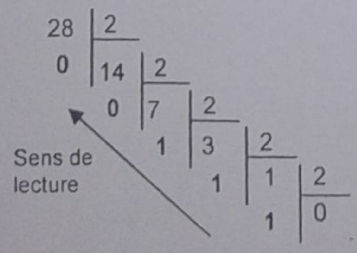
Cette opération est particulièrement appelée ***codage.***

III-1-1- Cas des nombres entiers

Pour convertir l'expression décimale d'un nombre entier positif vers une autre base, on procède par des divisions entières successives par la base jusqu'à obtenir un quotient nul. Le dernier reste obtenu est le digit de poids fort (MSD) et le premier reste, le digit de poids faible (LSD).

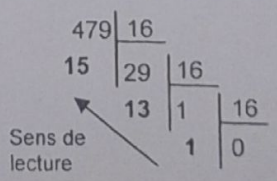
Exemples :

- Soit à coder en binaire le nombre 28.



Donc $(28)_{10} = (11100)_2$

- Soit à trouver l'équivalent hexadécimal de 479.



On sait que $(15)_{10} = (F)_{16}$ et $(13)_{10} = (D)_{16}$.

Donc $(479)_{10} = (1DF)_{16}$

III- 1- 2- Cas des nombres fractionnaires

a- Nombres fractionnaires inférieurs à 1

Soit N un nombre fractionnaire positif inférieur à 1 dont l'expression décimale est donnée.
Soit E_0 la partie entière de N. On a alors $E_0 = 0$ car $0 < N < 1$.
Pour convertir le nombre fractionnaire N de la base décimale vers une autre base B, on procède par des multiplications successives

Scanned by CamScanner

- Multiplication de N par la base B. (Posons P_1, le résultat du produit et E_{-1}, la partie entière de P_1).
- Multiplication de $(P_1 - E_{-1})$ par la base B. (Posons P_2, le résultat du produit et E_{-2}, la partie entière de P_2).
- Multiplication de $(P_2 - E_{-2})$ par la base B. (Posons P_3, le résultat du produit et E_{-3}, la partie entière de P_3).
- Multiplication de $(P_3 - E_{-3})$ par la base B. (Posons P_4, le résultat du produit et E_{-4}, la partie entière de P_4).
- Ce processus continu jusqu'à la précision souhaitée...

Ainsi l'équivalent dans la base B du nombre décimal N est $E_0,E_{-1}E_{-2}E_{-3}E_{-4}$ ...
On peut donc écrire $(N)_{10} = (E_0,E_{-1}E_{-2}E_{-3}E_{-4})_B$ si l'on admet comme précision, 4 digits après la virgule.

Exemples :

- Soit à coder vers la base 2, le nombre 0,625.

Sens de lecture ↓

0,	625
	x 2
1	250
	x 2
0	500
	x 2
1	000

Donc $(0,625)_{10} = (0,101)_2$

- Soit à Coder en hexadécimal le nombre 0,417.

Sens de lecture ↓

0,	417
	x 16
6	672
	x 16
10	752
	x 16
12	032
	x 16
0	512
	x 16
8	192
	x 16
3	072

On sait que $(10)_{10} = (A)_{16}$ et $(12)_{10} = (C)_{16}$

Donc $(0,417)_{10} = (0,6AC083)_{16}$ pour une précision de 6 digits après la virgule.

b- Nombres fractionnaires supérieurs à 1

Soit à trouver l'équivalent hexadécimal du nombre décimal 479,417.
On a 479,417 = 479 + 0,417.
Il s'agira alors de convertir chacun des deux termes de cette addition en hexadécimal et de faire une juxtaposition des deux résultats obtenus.
D'après les exemples précédents on a trouvé que $(479)_{10} = (1DF)_{16}$ et $(0,417)_{10} = (0,6AC083)_{16}$.

Donc $(479,417)_{10} = (1DF,6AC083)_{16}$.

III-2- Conversion d'une base B quelconque vers la base décimale

Cette opération est appelée **décodage**.
Pour effectuer un décodage, on multiplie les différents digits formant le nombre dans la base B donnée par leur poids respectif, puis on additionne les différents produits obtenus.

Exemples :

- Soit à décoder le nombre $(11001,101)_2$

On a donc : $(11001, 101)_2 = 1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$
$= 16 + 8 + 1 + 0,5 + 0,125$
$= 25,625$

D'où $(11001, 101)_2 = (25,625)_{10}$

- Soit à effectuer le transcodage du nombre hexadécimal AF en décimal.

On sait que $(A)_{16} = (10)_{10}$ et $(F)_{16} = (15)_{10}$. Donc $(AF)_{16} = 10x16^1 + 15x16^0 = 175$.

D'où $(AF)_{16} = (175)_{10}$

III-3- Conversion d'une base B_1 vers une base B_2 quelconques

B_1 et B_2 sont deux bases toutes différentes de la base 10.
Pour effectuer le transcodage d'un nombre exprimé dans la base B_1 vers la base B_2, plusieurs cas de figure et méthodes sont possibles.

III-3-1- Première méthode (méthode universelle)

C'est une méthode applicable quelles que soient les bases B_1 et B_2.
Elle comporte deux étapes :

- **Première étape** : on effectue un décodage (c'est-à-dire la conversion du nombre de la base B_1 vers la base décimale).
- **Deuxième étape** : on effectue un codage (c'est-à-dire la conversion du nombre obtenu en décimal vers la base B_2).

Exemple : Soit à déterminer l'équivalent **hexadécimal** du nombre **octal** 567.

Le décodage de $(567)_8$ donne : $(567)_8 = 5x8^2 + 6x8^1 + 7x8^0 = 5x64 + 6x8 + 7 = (375)_{10}$.

Le codage en hexadécimal de $(375)_{10}$ donne :

375	16	
7	23	16
	7	1
		1

Soit alors $(375)_{10} = (177)_{16}$

Ainsi le transcodage donne : $(567)_8 = (177)_{16}$

Scanned by CamScanner

III-3-2- Deuxième méthode (méthode particulière)

Cette méthode n'est seulement applicable que si et seulement si la base B_1 est une puissance $n^{ième}$ de la base B_2 ou vis versa. Deux cas de figure se présentent donc :

a - Cas 1 : La base de départ B_1 est une puissance $n^{ième}$ de la base d'arrivée B_2 ($B_1 = B_2^n$)

La méthode dans ce cas de figure consiste à convertir chaque digit de la base de **départ B_1** sur **n digits** de la base d'arrivée B_2. Pour ce faire, on se sert du tableau de comptage.

Exemple : Soit à donner l'équivalent **binaire** du nombre **hexadécimal** AF3

On a $16 = 2^4$. Donc on **transcrit** chaque **digit hexadécimal sur 4 bits**.
D'après le tableau de comptage on a $(A)_{16} = (1010)_2$, $(F)_{16} = (1111)_2$ et $(3)_{16} = (0011)_2$. Donc $(AF3)_{16} = (1010\ 1111\ 0011)_2$

b- Cas 2 : La base d'arrivée B_2 est une puissance $n^{ième}$ de la base de départ B_1 ($B_1^n = B_2$)

La méthode consiste à **regrouper** les digits du nombre dans la base de **départ B_1** par **groupe de n** puis à **convertir chaque groupe** obtenu en son équivalent dans la base **d'arrivée B_2**

NB : *Le regroupement des digits se fait de la droite vers la gauche pour la partie entière et de la gauche vers la droite pour la partie après la virgule.*

Exemple : Soit à donner l'équivalent **octal** du nombre **binaire** 11101010,10111.

On a $2^3 = 8$. Donc on regroupe convenablement par groupe de 3 bits les éléments du nombre en binaire.

On obtient : **011 101 010, 101 110**

Zéro non significatif mais permettant d'uniformiser les groupes

Zéros non significatifs mais permettant d'uniformiser les groupes

D'après le tableau de comptage on a $(011\ 101\ 010, 101\ 110)_2 = (352,56)_8$. Donc $(11101010,10111)_2 = (352,56)_8$

Remarque : Pour effectuer le transcodage d'un nombre d'une base B_1 vers une base B_2, on peut **si B_1 et B_2 sont tous les deux des puissances de 2**, utiliser la **base 2** comme **base relais ou base de transition**.

Exemple : Soit à convertir $(C3F)_{16}$ vers la base 8.

16 et 8 sont tous deux des puissances de 2. Plutôt que d'utiliser la base 10 comme base relais (décodage et codage : voir méthode universelle), il serait plus intéressant d'utiliser la base 2 comme base relais car la conversion est aisée d'une part entre les bases 16 et 2, et d'autre part entre les bases 2 et 8.
En effet $16 = 2^4$. Donc chaque digit hexadécimal sera converti sur 4 bits. Ce qui donne $(C3F)_{16} = (1100\ 0011\ 1111)_2$
Par ailleurs $2^3 = 8$. Donc on regroupe les bits du nombre binaire obtenu par groupe de 3, puis on trouve la valeur octale de chaque groupe. On obtient $(110\ 000\ 111\ 111)_2 = (6077)_8$.

Donc $(C3F)_{16} = (6077)_8$.

EXERCICES

Exercice 1
Effectuer le transcodage de :
1/ $(917,34)_{10}$ en octal avec 4 digits après la virgule.
2/ $(175,0825)_{10}$ en hexadécimal avec 12 chiffres après la virgule.
3/ $(65536)_{10}$ en binaire.

Exercice 2
Trouver les équivalents décimaux de :
1/ $(1001000,001)_2$
2/ $(473)_8$
3/ $(109A)_{16}$
4/ $(CACA0,CAFE)_{16}$

Exercice 3
Convertir les nombres suivants vers les bases indiquées :
1/ $(111100)_2 = (\ ?\ )_{16}$
2/ $(1010111,01)_2 = (\ ?\ )_8$
3/ $(7510)_8 = (\ ?\ )_{16}$
4/ $(FC105,02)_{16} = (\ ?\ )_2$
5/ $(507,104)_8 = (\ ?\ )_2$
6/ $(5BF,DAE)_{16} = (\ ?\ )_8$
7/ $(CEA3)_{16} = (\ ?\ )_8$

Exercice 4
1/ Que vaut le nombre $(65530)_{10}$ en hexadécimal ? En octal ?
2/ Faire un tableau de comptage de $(65530)_{10}$ à $(65537)_{10}$ dont les colonnes sont les évolution en décimal, hexadécimal, et octal. On indiquera le poids des différentes positions des digits dans chaque colonne.

Exercice 5
1/ Dresser un tableau de comptage de 0 à 15 en base 16, base 10, base 9, base 4, base 3 et base 2.
2/ Transcoder les nombres vers la base indiquée :
a/ $(642)_{10} = (\ ?\ )_9 = (\ ?\ )_3$
b/ $(2103)_4 = (\ ?\ )_2$
c/ $(AF,CD)_{16} = (\ ?\ )_4$
d/ $(2212,2)_3 = (\ ?\ )_9$
e/ $(32768,625)_{10} = (\ ?\ )_4$

Scanned by CamScanner

CORRECTION

Exercice 1

1/ $(917,34)_{10} = (917)_{10} + (0,34)_{10}$

917	8		
5	114	8	
	2	14	8
		6	1	8
			1	0

$(917)_{10} = (1625)_8$

0,	34
	x 8
2	72
	x 8
5	76
	x 8
6	08
	x 8
0	64

$(0,34)_{10} = (0,2560)_8$

On déduit que $\mathbf{(917,34)_{10} = (1625,2560)_8}$

2/ $(175,0825)_{10} = (175)_{10} + (0,0825)_{10}$

175	16
F	10
	A

16	
0	

$(175)_{10} = (AF)_{16}$

0,	0825
	x16
1	3200
	x16
5	1200
	x16
1	9200
	x16
E	7200
	x16
B	5200
	x16
8	3200
	x16
5	1200

Récurrence

Ainsi $(0,0825)_{10} = (0,151EB851EB85)_{16}$

D'où $\mathbf{(175,0825)_{10} = (AF,151EB851EB85)_{16}}$

3/ Faire des divisions successives par 2 du nombre $(65536)_{10}$ jusqu'à obtenir un quotient nul serait laborieux.
Utilisons plutôt une transition par la base 16 car elle nous éviterait une trop longue suite de divisions.

65536	16			
0	4096	16		
	0	256	16	
		0	16	16
			0	1	16
				1	0

Donc $\mathbf{(65536)_{10} = (10000)_{16}}$

Nous passons de la base 16 à la base 2 en transcodant chaque digit du nombre dans la base 16 sur 4 bits (car $16 = 2^4$)

$(10000)_{16} = (0001\ 0000\ 0000\ 0000\ 0000)_2$
$= (1\ 0000\ 0000\ 0000\ 0000)_2$

D'où $\mathbf{(65536)_{10} = (1\ 0000\ 0000\ 0000\ 0000)_2}$

Exercice 2

1/ $(1001000,001)_2 = 2^6 + 2^3 + 2^{-3} = 64 + 8 + 0,125 = 72,125$
$\mathbf{(1001000,001)_2 = (72,125)_{10}}$

2/ $(473)_8 = 4 \times 8^2 + 7 \times 8^1 + 3 \times 8^0 = 256 + 56 + 3 = 315$
$\mathbf{(473)_8 = (315)_{10}}$

3/ $(109A)_{16} = 16^3 + 9 \times 16 + 10 = 4250$
$\mathbf{(109A)_{16} = (4250)_{10}}$

4/ $(CACA0,CAFE)_{16} = 12 \times 16^4 + 10 \times 16^3 + 12 \times 16^2 + 10 \times 16 + 12 \times 16^{-1} + 10 \times 16^{-2} + 15 \times 16^{-3} + 14 \times 16^{-4}$
$= 830624,7929$
$\mathbf{(CACA0,CAFE)_{16} = (830624,7929)_{10}}$

Exercice 3

1/ $(111100)_2 = (0011\ 1100)_2$
$\mathbf{(111100)_2 = (3C)_{16}}$

2/ $(1010111,01)_2 = (001\ 010\ 111,\ 010)_2$
$\mathbf{(1010111,01)_2 = (127,2)_8}$

3/ En utilisant la base 2 comme base relais on a :
$(7510)_8 = (111\ 101\ 001\ 000)_2$
$= (1111\ 0100\ 1000)_2$
$\mathbf{(7510)_8 = (F48)_{16}}$

4/ $(FC105,02)_{16} = (1111\ 1100\ 0001\ 0000\ 0101,\ 0000\ 0010)_2$
$\mathbf{(FC105,02)_{16} = (1111\ 1100\ 0001\ 0000\ 0101,\ 0000\ 001)_2}$

5/ $(507,104)_8 = (101\ 000\ 111,\ 001\ 000\ 100)_2$
$\mathbf{(507,104)_8 = (101\ 000\ 111,\ 001\ 000\ 1)_2}$

6/ $(5BF,DAE)_{16} = (0101\ 1011\ 1111,\ 1101\ 1010\ 1110)_2$
$= (010\ 110\ 111\ 111,110\ 110\ 101\ 110)_2$
$\mathbf{(5BF,DAE)_{16} = (2677,6656)_8}$

7/ $(CEA3)_{16} = (1100\ 1110\ 1010\ 0011)_2$
$= (001\ 100\ 111\ 010\ 100\ 011)_2$
$\mathbf{(CEA3)_{16} = (147243)_8}$

Exercice 4

1/ En hexadécimal :

65530	16			
A	4095	16		
	F	255	16	
		F	15	16
			F	0

Donc $\mathbf{(65530)_{10} = (FFFA)_{16}}$

En octal :

$(FFFA)_{16} = (1111\ 1111\ 1111\ 1010)_2$
$= (001\ 111\ 111\ 111\ 111\ 010)_2$
$(FFFA)_{16} = (177772)_8$

Donc $\mathbf{(65530)_{10} = (177772)_8}$.

Scanned by CamScanner

2/ Tableau de comptage.

Bases	Décimal					Hexadécimal					octal					
Poids	10^4	10^3	10^2	10^1	10^0	16^4	16^3	16^2	16^1	16^0	8^5	8^4	8^3	8^2	8^1	8^0
	6	5	5	3	0		F	F	F	A	1	7	7	7	7	2
	6	5	5	3	1		F	F	F	B	1	7	7	7	7	3
	6	5	5	3	2		F	F	F	C	1	7	7	7	7	4
	6	5	5	3	3		F	F	F	D	1	7	7	7	7	5
	6	5	5	3	4		F	F	F	E	1	7	7	7	7	6
	6	5	5	3	5		F	F	F	F	1	7	7	7	7	7
	6	5	5	3	6	1	0	0	0	0	2	0	0	0	0	0
	6	5	5	3	7	1	0	0	0	1	2	0	0	0	0	1

Exercice 5

1/ Tableau de comptage.

Base 16	Base 10	Base 9	Base 4	Base 3	Base 2
0	0 0	0 0	0 0	0 0 0	0 0 0 0
1	0 1	0 1	0 1	0 0 1	0 0 0 1
2	0 2	0 2	0 2	0 0 2	0 0 1 0
3	0 3	0 3	0 3	0 1 0	0 0 1 1
4	0 4	0 4	1 0	0 1 1	0 1 0 0
5	0 5	0 5	1 1	0 1 2	0 1 0 1
6	0 6	0 6	1 2	0 2 0	0 1 1 0
7	0 7	0 7	1 3	0 2 1	0 1 1 1
8	0 8	0 8	2 0	0 2 2	1 0 0 0
9	0 9	1 0	2 1	1 0 0	1 0 0 1
A	1 0	1 1	2 2	1 0 1	1 0 1 0
B	1 1	1 2	2 3	1 0 2	1 0 1 1
C	1 2	1 3	3 0	1 1 0	1 1 0 0
D	1 3	1 4	3 1	1 1 1	1 1 0 1
E	1 4	1 5	3 2	1 1 2	1 1 1 0
F	1 5	1 6	3 3	1 2 0	1 1 1 1

2/

a/

642	9	
3	71	9
8	7	9
7	0	

Donc $(642)_{10} = (783)_9$

On a $9 = 3^2$. On va alors transcoder chaque digit de la base 9 sur 2 digits ternaires (base 3). Ainsi $(783)_9 = (21\ 22\ 10)_3$.
D'où $(642)_{10} = (212210)_3$

b/ $4 = 2^2$. On transcodera chaque digit de la base 4 sur 2 bits. Ainsi $(2103)_4 = (10\ 01\ 00\ 11)_2$

c/ $16 = 4^2$. On transcode chaque digit hexadécimal sur 2 digits de la base 4. Ainsi $(AF,CD)_{16} = (22\ 33,\ 30\ 31)_4$

d/ $3^2 = 9$. On va faire dans l'expression du nombre en ternaire, des groupes de deux digits, puis transcoder chaque groupe obtenu en base 9. Ainsi $(2212,2)_3 = (22\ 12,20)_3 = (85,6)_9$

e/ $(32768,625)_{10} = (32768)_{10} + (0,625)_{10}$. Utilisons la base 16 comme base relais.

32768	16	
0	2048	16
0	128	16
0	8	16
8	0	

Donc $(32768)_{10} = (8000)_{16}$

0,	625
	x 16
A	000

Donc $(0,625)_{10} = (0,A)_{16}$

D'où $(32768,625)_{10} = (8000,A)_{16}$

Puisque $16 = 4^2$, on transcrit chaque digit hexadécimal sur 2 digits de la base 4. Ce qui donne que $(8000,A)_{16} = (20\ 00\ 00\ 00,22)_4$
Soit alors $(32768,625)_{10} = (20000000,22)_4$.

Scanned by CamScanner

CHAPITRE II – CODES BINAIRES

Lorsque nous devons faire un traitement automatique d'informations, il importe de définir un code (de traitement) reconnu par tous les équipements de traitement. Par exemple si, le système décimal est utilisé comme code de traitement, alors il faut que les dispositifs de traitement puissent fonctionner avec dix niveaux de tensions différentes destinées à représenter les dix symboles décimaux. Malheureusement de tels dispositifs seraient difficiles à concevoir. Par contre, il est très facile d'imaginer des systèmes électroniques simples et précis fonctionnant seulement avec deux niveaux de tensions (la masse et le plus de la tension).
Pour ce faire la plus part des systèmes numériques ont recours aux systèmes binaires comme système de numération de base, au sein duquel plusieurs autres codes ont été inventés pour rendre plus facile ou plus rapide les différents traitements dans les équipements électroniques.

I - CODES BINAIRES PONDERES

Un code est dit pondéré, si pour un nombre écrit dans ce code, chaque position de digit a une valeur intrinsèque nommée **poids**. Par exemple pour le système décimal, les poids positionnels, sont dans l'ordre croissant (de la droite vers la gauche) 1, 10, 100, 1000,...

I-1- Code binaire naturel (ou code binaire pur)

I-1-1- Différents poids du code binaire pur

Pour le code binaire pur ou binaire naturel, les positions ont dans l'ordre croissant, les poids 2^0, 2^1, 2^2, 2^3, 2^4, ... soit 1, 2, 4, 8, 16, ...
Tableau des codes binaires naturels de 4 bits et leurs correspondants décimaux.

Décimal	Binaire			
	2^3	2^2	2^1	2^0
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
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Remarque :
Pour obtenir facilement un tableau de comptage en binaire naturel, il suffit pour chaque position de bit (chaque rang), de faire une alternance d'une suite de 0 et d'une suite de 1 successivement jusqu'à remplir toute la colonne correspondante de ce rang. Le nombre de zéros dans une suite de 0 est identique au nombre de uns dans une suite de 1. Ce nombre d'éléments dans une suite étant égal à la valeur du poids de la position.
Par exemple :
- Pour le rang 0, le poids étant $2^0 = 1$, on remplit alors la colonne de ce rang par une alternance d'un seul 0 et d'un seul 1 successivement.
- Pour le rang 2, le poids étant $2^2 = 4$, on remplit alors la colonne de ce rang par une alternance de quatre 0 et de quatre 1 successivement.

I-1-2- Traitements des bits

Les équipements numériques (ordinateur, modem, caméra numérique, téléphone mobile, calculateur,...) traitent des informations codées sur d'énormes quantités de bits. Pour rendre plus fluide le traitement ou pour avoir une parfaite maîtrise sur la quantité de bits manipulés, les traitements dans ces équipements se font par groupe de 4, 8, 16 bits, ...

Un nombre de **4 bits** est un **quartet** (ou un **quadruplet** , un **tétrade** , un **nibble**) et un nombre de **8 bits**, un **octet**.

Lorsqu'on prend un ensemble de bits dont la taille atteint $2^{10} = 1024$ **bits**, on dit que cet ensemble est un **kilobit**.
Le **kilobit** est **l'unité de capacité** de traitement numérique.
Des multiples de cette unité ont été définis :

1 kilobit (Kb) = 2^{10} bits 1 Mégabit (Mb) = 2^{10} Kb = 2^{20} bits 1 Gigabit (Gb) = 2^{10} Mb = 2^{20} Kb = 2^{30} bits.

Lorsqu'on actionne une touche sur un clavier d'ordinateur par exemple, on envoie une information constituée de 8 bits vers l'unité centrale. L'action sur plusieurs touches permet d'y envoyer une série d'octets. Ainsi une série de 1024 octets constitue un **kilo-octet**. On définit alors :

1 kilo-octet (ko) = 2^{10} octets = 2^{13} bits 1 Mégaoctet (Mo) = 2^{10} ko 1 Gigaoctet (Go) = 2^{10} Mo.

Exemple: Un disque dur de 10 Go peut enregistrer 10×2^{30} octets = 10 737 418 240 octets = 85 899 345 920 bits.

I-2- Code Décimal Codé Binaire (DCB)

Le codage DCB (Décimal Codé Binaire) d'un entier naturel consiste à coder séparément chaque digit décimal sur **quatre bits.**
Pour désigner ce code on utilise aussi très souvent le sigle anglais **BCD** (Binary Coded Decimal).
Les digits décimaux étant 0, 1, 2, 3,4, 5, 6, 7, 8, 9, alors les codes binaires de 4 bits (tétrades) correspondants sont 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001.
Ainsi le code BCD est un sous-ensemble de 10 combinaisons parmi les 16 combinaisons possibles de 4 bits du code binaire naturel.

Exemple : $(6971)_{10} = (0110\ 1001\ 0111\ 0001)_{BCD}$ $(103)_{10} = (0001\ 0000\ 0011)_{BCD}$

Il va de soit que pour décoder un nombre exprimé en BCD vers son équivalent décimal il suffit alors de réaliser des groupes de 4 bits au sein de l'expression du nombre donné en BCD et de donner l'équivalent décimal de chaque groupe de 4 bits.

Exemple : $(100100001010000)_{BCD} = (0100\ 1000\ 0101\ 0000)_{BCD} = (4850)_{10}$.

Scanned by CamScanner

Toute fois, il est à noter que le code BCD est un code pondéré dont les coefficients de pondération (poids) de la droite vers la gauche (sens des poids croissants) sont :

- Pour le premier tétrade (unité) : 2^0, 2^1, 2^2, 2^3, c'est-à-dire 1, 2, 4, 8.
- Pour le deuxième tétrade (dizaine) : 10×2^0, 10×2^1, 10×2^2, 10×2^3, c'est-à-dire 10, 20, 40, 80.
- Pour le troisième tétrade (centaine) : 100×2^0, 100×2^1, 100×2^2, 100×2^3, c'est-à-dire 100, 200, 400, 800.
- Etc.

Ainsi les poids des bits de tous les tétrades sont : 1, 2, 4, 8,10, 20, 40, 80, 100, 200, 400, 800, etc.
On peut donc également trouver l'équivalent décimal d'un nombre BCD en procédant par la méthode de multiplication de chaque bit par son poids.

Exemple : Soit à trouver l'équivalent décimal du nombre $(100100110111)_{BCD}$

1	0	0	1	0	0	1	1	0	1	1	1
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
800		200		80		***20***		8		2	
	400		***100***		40		***10***		***4***		***1***

Poids

Donc $(100100110111)_{BCD} = 800+100+20+10+4+2+1 = (937)_{10}$.

Le codage en BCD est beaucoup utilisé en électronique numérique car il permet d'obtenir les affichages numériques (montre, calculatrice, indication du niveau d'un ascenseur...).

II – CODES BINAIRES NON PONDERES

Pour ces codes chaque position de bit dans l'expression d'un nombre binaire n'a pas de poids.

II-1- Code excédent 3

Le codage excédent 3 (ou majoré 3) consiste à ajouter 3 à chaque chiffre décimal puis à coder en binaire sur 4 bits chaque nouveau chiffre obtenu.

Exemple : Soit à donner l'équivalent en excédent 3 du nombre $(6971)_{10}$.

6+3 = 9 9+3 = 12 7+3 = 10 1+3 = 4 Donc $(6971)_{10} = (1001\ 1100\ 1010\ 0100)_{EXC3}$.

NB : le code majoré 3 permet d'effectuer des opérations de soustraction décimale par des circuits électroniques dédiés aux opérations arithmétiques.
Ces opérations de soustraction se fondent sur l'expression du complément à 9 des digits. Cela permet de transformer l'opération de soustraction en une opération d'addition donnant le même résultat.
Voyons comment se forme le complément à 9 des digits dans le code excédent 3 d'après l'exemple suivant :

- On sait que le complément à 9 de 2 est 7. Or le code excédent trois de 2 est 0101 et celui de 7 est 1010. On remarque alors que dans le code excédent trois de 2, si on change les 0 par des 1 et les 1 par des 0 (on dit qu'on complémente chaque bit), on obtient le code excédent trois de 7.
- Le complément à 9 de 5 est 4. Or $(5)_{10} = (1000)_{EXC3}$ et $(4)_{10} = (0111)_{EXC3}$. On constate bien que si 4 et 5 sont exprimés en excédent 3, on obtient le code du complément à 9 du code de 4 en complémentant chaque bit du code de 5.

Ainsi à partir du code excédent 3 d'un chiffre décimal, on peut obtenir le code excédent 3 de son complément à 9 en complémentant chaque bit du code excédent 3 de ce chiffre. Pour ce faire on dit du code excédent 3 qu'il est **auto complémentaire.**

II-2- Code binaire réfléchi ou code Gray

Pour ce code, entre deux combinaisons successives, un seul bit change de valeur. C'est-à-dire que la représentation précédente ne diffère de la suivante que d'un seul bit. On dit alors de ce code qu'il est à **distance minimale.**

Tableau du code Gray pour les valeurs décimales de 0 à 15 et leurs correspondants en binaire naturel.

Décimal	Binaire naturel				Code Gray			
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

1er axe de symétrie
2ème axe de symétrie
3ème axe de symétrie
4ème axe de symétrie éventuelle

Remarques :

- Comme nous l'avons vu au chapitre précédent, un code binaire de **n** bits peut avoir au total 2^n combinaisons possibles. Dans notre tableau n étant égale à 4 donc nous avons 16 combinaisons de quatre bits (quartets) possibles ; que ce soit en binaire naturel ou en binaire réfléchi.

Ainsi tous les quartets du binaire naturel sont également des quartets pour le binaire réfléchi. Mais la différence c'est qu'ils n'ont pas la même valeur correspondante décimale (par exemple $(1000)_2 = (8)_{10}$ et $(1000)_{GRAY} = (15)_{10}$).

On peut donc retenir que la différence entre le code binaire naturel (binaire pur) et le code Gray ne porte pas sur l'ensemble des combinaisons possibles mais plutôt sur la succession des combinaisons.

Scanned by CamScanner

Si le passage du binaire pur en décimal se fait en tenant compte des poids binaires, il n'en est pas de même pour le code Gray car ce code n'est pas pondéré.

- Pour former plus facilement le tableau du code Gray, on peut faire recours à l'astuce suivante :
  - Numéroter les positions 1, 2, 3, 4, ... de la droite vers la gauche.
  - Tracer des axes de symétrie dans chaque colonne de position tels que :
    - Le 1er axe se limitant à la 1ère colonne ait au-dessus 2^1=2 lignes.
    - Le 2e axe se limitant à la 2e colonne ait au-dessus 2^2=4 lignes.
    - Le 3e axe se limitant à la 3e colonne ait au-dessus 2^3=8 lignes.
    - Etc. (pour autant de bits qu'il y a).
  - Remplir les lignes au-dessus de chaque axe par autant de 0 que de 1 en commençant par la suite de 0.
  - Remplir les lignes au-dessous de chaque axe en adoptant l'évolution suivante :
    - Les lignes au-dessus d'un axe sont symétriques aux lignes au-dessous de cet axe jusqu'à l'axe suivant. (La symétrie par rapport à un axe revient à recopier au-dessous de cet axe jusqu'à l'axe prochain, dans le sens inverse, tous les bits situés au-dessus).
    - Recommencer l'opération en considérant ce nouvel axe croisé.
    - Par exemple, pour la colonne 2 les 4 premières lignes sont symétriques par rapport à l'axe 2, aux 4 autres lignes situées entre l'axe 2 et l'axe 3. De même les 8 lignes au-dessus de l'axe 3 sont symétriques par rapport à cet axe, aux 8 autres lignes situées ente l'axe 3 et l'axe 4.

Ce procédé très pratique peut s'avérer délicat lorsqu'on travaille sur un nombre de bits supérieur à 4 car il peut facilement s'introduire une erreur lors des réalisations des lignes symétriques (du moins lorsqu'on le fait manuellement). Cependant, une autre méthode très intéressante et qui se veut pratiquement infaillible quel que soit le nombre de bits de codage, serait bien à connaître. Cette méthode, je l'ai baptisée méthode de ***"pseudo pondération alternative"****. Découvrons la :*

Code Gray			
2^3	2^2	2^1	2^0
2^4	2^3	2^2	2^1
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

← *Valeur indiquant le nombre de 0 initiaux pour chaque colonne*

← *Valeur indiquant le nombre de 1 et de 0 dans une suite de uns et de zéros évoluant alternativement pour chaque colonne.*

Contrairement au binaire naturel où l'on peut grâce au poids unique de chaque position de bit savoir combien de 0 et 1 vont se recopier alternativement, j'ai pu définir pour le code Gray, deux "poids" différents pour chaque position de bit.
La première série de "poids", 2^0, 2^1, 2^2, 2^3, ..., c'est-à-dire 1, 2, 4, 8, ... (semblable au poids binaire naturel) permet de savoir combien de 0 il faut inscrire au début de chaque colonne (soit un seul 0 au début de la 1ère colonne, deux 0 au début de la 2e colonne, quatre 0 au début de la 3e colonne, huit 0 au début de la 4e colonne, ...).
La deuxième série de "poids", 2^1, 2^2, 2^3, 2^4, ..., c'est-à-dire 2, 4, 8, 16, ... permet de savoir combien de 1 il faut aligner d'abord, suivie du même nombre de 0 que l'on va répéter alternativement jusqu'à la dernière ligne ; (soit inscrire une alternance de deux 1 et de deux 0, cela jusqu'à la dernière ligne de la 1ère colonne, inscrire une alternance de quatre 1 et de quatre 0 pour la 2e colonne, etc.).
Malheureusement ce sympathique "jeu" est limité par le nombre de lignes, étroitement liées au nombre de bits de codage. En effet, si l'on code sur 3 bits on a 8 lignes, sur 4 bits on a 16 lignes, sur 5 bits on a 32 lignes etc.

III- AUTRES TYPES DE CODES BINAIRES

III-1- codes alphanumériques

Un code est dit alphanumérique s'il permet de coder des nombres, des lettres alphabétiques et des symboles particuliers. Les codes alphanumériques sont destinés à la transmission d'informations quelconques vers des circuits de traitement logique.
Le code alphanumérique le plus connu est le Code ASCII (American Standard Code for Information Interchange) ; entendons "code *standard américain pour l'échange d'informations*". C'est le code qu'utilise la majorité des fabricants des micro-ordinateurs.
Le code ASCII comprend les 26 lettres de l'alphabet (en minuscule et en majuscule), les 10 chiffres décimaux et au moins 25 caractères spéciaux. Soit un total de 87 symboles au minimum. Puisque $2^6 \leq 87 \leq 2^7$, alors il faut 7 bits pour coder chacun de ces symboles. A ces 7 bits un 8ème bit est ajouté permettant d'effectuer un contrôle sur l'information transmise. C'est le bit de parité.

Scanned by CamScanner

Parité	00	01	10	11
- -- 00000	Codes réservés	espace	à	
- -- 00001		!	A	a
- -- 00010		"	B	b
- -- 00011		£	C	c
- -- 00100		$	D	d
- -- 00101		%	E	e
- -- 00110		&	F	f
- -- 00111		'	G	g
- -- 01000		(	H	h
- -- 01001		)	I	i
- -- 01010		*	J	j
- -- 01011		+	K	k
- -- 01100		,	L	l
- -- 01101		-	M	m
- -- 01110		.	N	n
- -- 01111		/	O	o
- -- 10000		0	P	p
- -- 10001		1	Q	q
- -- 10010		2	R	r
- -- 10011		3	S	s
- -- 10100		4	T	t
- -- 10101		5	U	u
- -- 10110		6	V	v
- -- 10111		7	W	w
- -- 11000		8	X	x
- -- 11001		9	Y	y
- -- 11010		:	Z	z
- -- 11011		;	°	é
- -- 11100		<	ç	ù
- -- 11101		=		è
- -- 11110		>	^	~
- -- 11111		?	_	effacement

Tableau du code ASCII

III-2- Code avec bit de parité (détecteur d'erreur)

L'ajout d'un bit de parité à un code permet de détecter d'éventuelles erreurs pouvant s'entacher dans le code à transmettre. Un bit de parité est un bit supplémentaire associé à un groupe de bits d'un code destinés être transférés d'un endroit à un autre.

III-2-2 Bit de contrôle de parité paire

On fixe le bit de parité pour que le nombre total de 1 dans le code (y compris le bit de parité ajouté) soit un nombre pair.
Si dans la représentation du code à transmettre le nombre de 1 est pair alors le bit de parité est 0 sinon le bit de parité est 1.

Ainsi à la réception, en effectuant un contrôle sur le nombre de bit 1, on pourra détecter s'il y a erreur (nombre de bits à 1 du code reçu est impair) ou non (nombre de bits à 1 du code reçu est pair).

III-2-2 Bit de contrôle de parité impaire

C'est exactement la même chose que précédemment mais cette fois la valeur du bit de parité est telle que le nombre de 1 est impair.

EXERCICES

Exercice1

1/ Si l'intervalle des adresses des cases mémoires dans un micro-ordinateur va de $(0000)_{16}$ à $(FFFF)_{16}$, combien cet ordinateur a-t-il de cases mémoires?
2/ On suppose que chaque case est un octet. Convertir l'ensemble de bits formé par toutes les cases mémoires en Ko (Kilo-octet) et en Mb (Mégabit).

Exercice2

Combien y a-t-il d'octets dans cinq gigabits ?

Exercice3

1/ Donner le tableau du code Gray correspondant aux valeurs binaires naturels de 0 à 15.
2/ Sachant que les nombres suivants sont des nombres exprimés en BCD, trouver leur équivalent décimal.
a/ 1000000000010010
b/ 11010110000011
3/ Répondre la question précédente, si les nombres sont exprimés en excédent 3.
4/ Exprimer en ASCII l'instruction suivante sachant que les codes des lettres et des symboles ont un bit de parité pair et ceux des chiffres un bit de parité impair : MOVE $81,D1

Scanned by CamScanner

CORRECTION

Exercice 1

1/ L'intervalle en base 16 $[(0000)_{16} ; (FFFF)_{16}]$ correspond en base 10 à $[(0)_{10} ; (65535)_{10}]$ car $(FFFF)_{16} = 15\ (16^3+16^2+16+1) = (65535)_{10}$. Puisque chaque valeur de cet intervalle correspond à l'adresse d'une case mémoire, alors le nombre N de cases mémoires est : **N = 65535+1 = 65536 cases mémoires.**

2/ Si chaque case mémoire est un octet, alors l'ensemble E des bits formés par toutes les cases mémoires vaut :
E = 65536 x 1 octet = **65536 octets**
Or 1 Ko = 2^{10} octets ⇒ **E = 65536 / 2^{10}= 64 Ko**
On sait que 1 octet = 8 bits ⇒ **E = 65536 x 8 = 524288 bits**
On a 1 Mb = 2^{20} bits ⇒ **E = 524288 / 2^{20} = 0,5 Mb**

Exercice 2

5 Gb = 5 x 2^{30} bits = 5 x 2^{27} octets = 671 088 640 octets

Exercice 3

1/

BINAIRE PUR				CODE GRAY			
d	c	b	a	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

2/
a/ $(1000000000010010)_{BCD} = (1000\ 0000\ 0001\ 0010)_{BCD} = (8012)_{10}$
b/ $(11010110000011)_{BCD} = (11\ 0101\ 1000\ 0011)_{BCD} = (3583)_{10}$

3/
a/ Dans l'expression en exc 3, chaque groupe de 4 bits doit être supérieur ou égale à 3. Donc le code 1000 000 0001 0010 n'est pas un nombre exprimé en exc 3.
b/ $(11010110000011)_{exc3} = (11\ 0101\ 1000\ 0011)_{exc3} = (0250)_{10}$.

4/ MOVE $81,D1 a pour code ASCII selon l'énoncé :

01001101	11001111	01010110	11000101	10100000	00100100	00111000	00110001	10101100	01000100	00110001
(M)	(O)	(V)	(E)	(espace)	-(-($)	(8)	(1)	(,)	(D)	(1)

Scanned by CamScanner

CHAPITRE III – ARITHMETIQUE BINAIRE

I- REPRÉSENTATION DES NOMBRES BINAIRES

Dans les équipements de traitement arithmétique, les nombres binaires manipulés sont rangés dans des cases mémoires (ou registres). Pour effectuer des opérations arithmétiques sur ces nombres, il existe deux manières de les représenter dans ces cases mémoires.

I-1- Représentation à virgule fixe

Soit à mettre en mémoire dans un registre à 10 cases par exemple, le nombre 5,75 qui s'écrit en binaire 101,11.

0	0	0	0	1	0	1	1	1	0

Dans cette représentation la position de la virgule est d'emblée fixée à la troisième 3e case par la droite. D'où le terme **représentation à virgule fixe**. Cette représentation suppose la multiplication du contenu du registre par 2^{-3}.
Ainsi **101,11 = 0000101110 X 2^{-3}.**
Il ressort donc que dans cette représentation à virgule fixe, la grandeur maximum qu'il est possible d'inscrire dans les 10 cases est :

1	1	1	1	1	1	1	1	1	1

Ce qui correspond à une valeur de $1111111111 \times 2^{-3} = (1023 / 8) = 127,875$

Supposons que nous devons représenter le nombre 6,8125, soit en binaire 110,1101 dans les 10 cases précédentes. Ce qui donne

0	0	0	0	1	1	0	1	1	0

L'équivalent décimal du contenu du registre est : $110110 \times 2^{-3} = (54 / 8) = 6,75$.
$6,75 \neq 6,8125$: la représentation à virgule fixe aura apporté une erreur. C'est un inconvénient de cette représentation.

I- 2- Représentation à virgule flottante

Le nombre 6,8125 valant en binaire 110,1101 peut s'écrire :

$110,1101 \times 2^0$
$11,01101 \times 2^1$
$1,101101 \times 2^2$
$0,1101101 \times 2^3$

La dernière expression donne un nombre inférieur à 1 que multiplie une puissance 2 dont l'exposant est représentatif de la position de la virgule. Tout nombre peut se représenter ainsi. Pour ce faire, il suffit de deux termes pour définir totalement toute grandeur.
Dans notre cas ces termes sont :

- 1101101 : Il est appelé la **mantisse.**
- 3, soit en binaire (11) : Il est appelé la **caractéristique** ou l'**exposant.**

Si nous gardons toujours 3 cases pour la partie représentative de la position de la virgule comme dans le cas précédant, nous avons :

1	1	0	1	1	0	1	0	1	1

Mantisse **Caractéristique**

Cette représentation est appelée représentation à **virgule flottante.**
Elle suppose une valeur implicite qui vaut : **$0,1101101 \times 2^3$** ; soit en décimal

$\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{16}+\frac{1}{32}+\frac{1}{128}\right)\times 2^3 = 6,8125$: exactement le nombre mis en mémoire

Dans cette représentation à virgule flottante, la grandeur maximum qu'il est possible d'inscrire dans les 10 cases est :

1	1	1	1	1	1	1	1	1	1

Soit $0,1111111 \times 2^{7}$. C'est-à-dire en décimal 127

La représentation à virgule flottante est la plus utilisée car pour le même nombre de cases, la représentation à virgule fixe introduit très rapidement des erreurs.
Pour pourvoir travailler sur les nombres signés une autre case mémoire est prévue pour représenter le signe.

Signe **Mantisse** **Caractéristique**

Remarque :

- La mantisse d'un nombre est un entier. Ainsi la différence entre la représentation à virgule flottante de par exemple – 6,8125 et + 6,8125 sera nécessairement le bit de signe.
- Si nous avons des nombres dont les expressions à virgule flottante donnent des valeurs différentes d'exposant, nous pourrons toujours les exprimer en gardant comme exposant commun le plus grand des exposants des expressions de ces nombres.

Exemple : Soit à exprimer les nombres binaires suivants avec une mantisse de 8 bits en uniformisant les exposants.
(i) 0,1 (ii) 10,111 (iii) 10111,01

On a : (iii) $10111,01 = 0,1011101 \times 2^5 =$ **$0,10111010 \times 2^5$** (ii) $10,111 =$ **$0,00010111 \times 2^5$** (i) $0,1 =$ **$0,00000100 \times 2^5$**

Scanned by CamScanner

D'où les représentations :

	Mantisse	Caractéristique
(i)	00000100	101
(ii)	00010111	101
(iii)	10111010	101

Vu ces deux points, dans tout ce qui va suivre, nous allons entièrement travailler sur des nombres **entiers signés** (puisque d'une part la mantisse est un entier et que d'autre part la caractéristique peut être uniformisée).

II- OPÉRATION SUR LES ENTIERS SIGNÉS

Un entier signé est un entier qui porte un signe **(+)** ou **(-)**. Par convention le signe **(+)** est codé par **0** et le signe **(-)** par **1**.
Dans la représentation binaire des entiers signés, le signe est le bit situé à l'extrême gauche.
On utilise généralement trois systèmes de **représentation** (ou de **codage**) **binaire** pour exprimer un nombre signé :

- Codage en grandeur exacte ou valeur absolue et signe (nous adopterons l'abréviation **VAS** pour désigner ce type de codage).
- Codage en complément à 1 ou complément restreint (nous adopterons l'abréviation **CR** pour désigner ce type de codage).
- Codage en complément à 2 ou complément vrai (nous adopterons l'abréviation **CV** pour désigner ce type de codage).

Lors du codage des entiers signés il est impératif d'indiquer le nombre de bits sur lesquels le codage s'effectue et le système de codage utilisé (VAS, CR ou CV).

II-1- Codage en valeur absolue et signe (VAS) sur n bits

- La représentation d'un entier signé (entier relatif) sur n bits en VAS n'est possible que s'il appartient à l'intervalle $[-2^n+1 ; 2^n-1]$; **bien entendu, le bit de signe n'étant pas compté.**
- La représentation en VAS d'un **entier relatif** est **identique au code binaire naturel de la valeur absolue** de cet entier auquel on adjoint le bit de signe.

Exemple:

- Pour n = 4 ; les entiers relatifs représentables en VAS doivent appartenir à l'intervalle $[-2^4+1 ; 2^4-1] = [-15 ; 15]$.
- En VAS sur 4 bits +5, +12, -3 et -14 sont représentés par :

+5 → **0** 0101 en VAS/4 +12 → **0** 1100 en VAS/4 -3 → **1** 0011 en VAS/4 -14 → **1** 1110 en VAS/4.

NB : *Nous adopterons par la suite la notation VAS/4 pour dire VAS sur 4 bits. Aussi verra-t-on une notation semblable pour les autres types de codage. Toutes ces représentations sont sous entendues contenir un 5ième bits pour le signe.*

II-2- Codage en complément Restreint (CR) sur n bits

- L'intervalle des entiers relatifs représentables en CR sur n bits est aussi $[-2^n+1 ; 2^n-1]$.
- La représentation d'un **entier positif** en CR est **identique** à sa représentation en **VAS**.
- La représentation d'un **entier négatif** en CR s'obtient à partir de la représentation en **VAS** en **remplaçant** les 0 par des 1 et les 1 par des 0 sauf le bit du signe qui reste inchangé (on dit qu'on complémente tous les bits sauf celui du signe).

Exemple : Soit à représenter +10 et -3 en CR sur 4 bits.

+10 → (**0** 1010)$_{CR/4}$ -3 → (**1** 0011)$_{VAS/4}$ → (**1** 1100)$_{CR/4}$

II-3- Codage en complément vrai (CV) sur n bits

- L'intervalle des entiers relatifs représentables en CV sur n bits est également $[-2^n+1 ; 2^n-1]$.
- La représentation d'un **entier positif** en CV est **identique** à sa présentation en **CR**.
- La représentation d'un **entier négatif** en CV s'obtient à partir de sa représentation en **CR** auquel on **ajoute 1**.

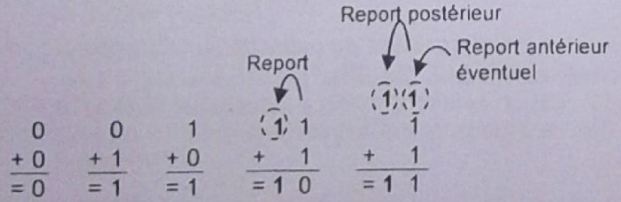
Exemple : Soit à représenter -15 en CV sur 4 bits.

-15 → (**1** 1111)$_{VAS/4}$ → (**1** 0000)$_{CR/4}$ → (**1** 0001)$_{CV/4}$

NB : *Astuce pour obtenir plus rapidement le CV d'un entier négatif :*
a/ Dans l'expression du nombre en VAS on conserve le LSB (le bit de plus faible poids).
b/ Si le LSB est égal à 0, en progressant de la droite vers la gauche, on conserve tous les bits jusqu'au prochain bit 1. Au-delà de ce bit, tous les autres changent sauf le bit de signe.
c/ Si le LSB est égal à 1, on change tous les bits sauf le bit de signe.

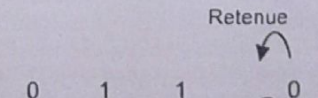
II-4- Règles de calcul pour les opérations élémentaires

- **Additions élémentaires**



0	0	1	1 1	1
+0	+1	+0	+1	+1
=0	=1	=1	=10	=11

- **Soustractions élémentaires**



0	1	1	0
-0	-0	-1	-1 1
=0	=1	=0	=11

- **Multiplications élémentaires**

0	0	1	1
x0	x1	x0	x1
=0	=0	=0	=1

- **Divisions élémentaires**

0	1	1	1	0	0	1	0
0	0	0	1	impossible		impossible	

Scanned by CamScanner

III- ADDITION – SOUSTRACTION BINAIRE

III-1- Addition – Soustraction en valeur absolue et signe (VAS)

➢ **Règles**

- Si les nombres sont de même signe, on effectue la somme de leur valeur absolue et le résultat porte le signe commun.
- Si les nombres sont de signes différents, on effectue la différence de leur valeur absolue et le résultat porte le signe du plus grand nombre en valeur absolue.

Exemple : Soit à effectuer en VAS/4 les opérations suivantes : (i) 5+3 ; (ii) -5-3 ; (iii) -5+3 ; (iv) -5+5.

(i) 0 0101 + 0 0011 = 0 1000 → +8

(ii) 1 0101 + 1 0011 = 1 1000 → -8

(iii) 1 0101 - 0 0011 = 1 0010 → -2

(iv) 1 0101 - 0 0101 = 1 0000 ⇒ -0
ou
= 0 0000 ⇒ +0

➢ **Remarque**

Dans les calculs en VAS les bits de signe n'étant pas affectés pas l'opération, il ressort que zéro a deux représentations (-0 ou +0) puisque les nombres dont la différence donne zéro sont toujours identique en valeur absolue.

III-2- Addition en complément restreint (CR)

➢ **Règles**

- Quels que soient leurs signes, on effectue la somme des nombres jusqu'aux bits de signe.
- Le report obtenu en additionnant les bits de signe est réinjecté (additionné) au résultat obtenu.

Exemple : Soit à effectuer en CR/4 les opérations suivantes : (i) -5+3 ; (ii) -5-3 ; (iii) -5+8 ; (iv) -5+5.

(i) 1 1010 + 0 0011 = 1 1101

(ii) 1 1010 + 1 1100 = 1 0110 ; + 1 = 1 0111 (Réinjection du report sur l'addition des bits de signe)

(iii) 1 1010 + 0 1000 = 0 0010 ; + 1 = 0 0011

(iv) 1 1010 + 0 0101 = 1 1111

➢ **Remarque**

- Les opérations de soustraction sont ramenées à des opérations d'additions dans les calculs en complément restreint.
- Pour vérifier l'exactitude d'un résultat négatif dans une opération en CR, il faut trouver la valeur VAS correspondante. Par exemple pour l'opération (i) on a bien $(1\ 1101)_{CR/4} = (1\ 0010)_{VAS/4} = (-2)_{10}$.

III-3- Addition en complément vrai (CV)

➢ **Règles**

- Quels que soient leurs signes, on effectue la somme des nombres jusqu'aux bits de signe.
- Le report obtenu en additionnant les bits de signe est négligé.

Exemple : Soit à effectuer en CV/4 les opérations suivantes : (i) -5-2 ; (ii) -2+12 ; (iii) 5-5

(i) 1 1011 + 1 1110 = 1 1001 ① Le report sur l'addition des signes est négligé

(ii) 1 1110 + 0 1100 = 0 1010

(iii) 0 0101 + 1 1011 = 0 0000

➢ **Remarque**

- Les opérations de soustraction sont ramenées à des opérations d'addition dans les calculs en complément vrai.
- Pour vérifier l'exactitude d'un résultat négatif dans une opération en CV, il faut trouver la valeur VAS correspondante. Par exemple pour l'opération (i) on a bien $(1\ 1001)_{CV/4} = (1\ 0111)_{VAS/4} = (-7)_{10}$.

III-4- Notion de débordement

Effectuons par exemple les calculs suivants en CV sur 4 bits : (i) -9-8 ; (ii) 9+8

(i) 1 0111 + 1 1000 = 0 1111 **Résultat faux et même positif**

(ii) 0 1001 + 0 1000 = 1 0001 **Résultat faux et même négatif**

Le changement de signe obtenu au cours de ces opérations, indique qu'il y a **dépassement de capacité** ou **débordement** ; c'est à dire que nous tentons de représenter un nombre ne faisant pas parti de l'intervalle $[-2^n + 1 ; 2^n - 1]$.
En effet, pour l'opération (ii) par exemple le résultat de 9+8 est +17 ; soit en binaire 0 10001 : il nécessite alors **cinq bits non compris le bit de signe** pour représenter ce résultat. Donc chaque nombre dans ce calcul doit être exprimé avec au moins cinq bits non compris le bit de signe.

Ce qui donne pour l'opération (ii) 0 01001 + 0 01000 = 0 10001 ⇒ +17

Ce dépassement de capacité est appelé **débordement** ou **overflow**. Quand un calculateur (Ordinateur, Calculatrice...) effectue des opérations arithmétiques, si les nombres manipulés sont trop grand pour son format maximum d'expression des nombres, alors on obtient un dépassement de capacité qui est en général signalé par un bit appelé **Overflow Flag**.

Scanned by CamScanner

III-5- Addition des nombres positifs exprimés en BCD

- **Règles**
- L'opération d'addition porte sur l'ensemble de tous les bits.
- Dans le résultat obtenu tous les quartets qui sont supérieurs à 9 c'est-à-dire 1001, sont défaillants et nécessitent une correction.
- Dans le résultat obtenu tous les quartets qui sont issus de la somme de quartets ayant engendrés la propagation d'un report vers des quartets de rang supérieur, sont défaillants et nécessitent une correction.
- La correction consiste à ajouter 6 c'est-à-dire 0110 aux quartets défaillants.
- Lors de la correction la propagation d'un report n'est plus à corriger.

Exemple : Soit à effectuer les opérations suivantes des nombres en BCD : (i) 25+34 ; (ii) 63+17 ; (iii) 98+28

(i)
```
  0010 0101
+ 0011 0100
= 0101 1001
   5    9
```

(ii)
```
  0110 0011
+ 0001 0111
  0111 1010
+      0110
= 1000 0000
   8    0
```

(iii)
```
    1001 1000
+   0010 1000
    1100 0000
+   0110 0110
= 1 0010 0110
  1  2    6
```

IV – MULTIPLICATION – DIVISION BINAIRE

Lorsqu'on effectue une multiplication ou une division de nombres entiers relatifs, on effectue l'opération sur la valeur absolue des nombres puis on affecte au résultat le bit de signe négatif (c'est-à-dire 1) si les nombres sont de signes opposés, sinon on y affecte le bit de signe positif (c'est-à-dire 0).

IV-1- Multiplication

Exemple : Soit à effectuer les opérations suivantes en exprimant les nombres en VAS : (i) +9 x (+2) ; (ii) -11x4

(i)
```
  0 1001
  0 x  10
     0000
    1001
= 0 10010
```

(ii)
```
  1 1011
  0 x 100
     0000
    0000
   1011
= 1 101100
```

Remarque

Lorsqu'on effectue des multiplications on doit prévoir un nombre assez suffisant de cases mémoires pour contenir le résultat. Par exemple si les opérations portent sur des nombres de 3 bits alors l'opération la plus grande qu'on puisse effectuer est 111x111 =110001. On doit donc prévoir 6 cases mémoires, plus éventuellement une case pour le bit de signe. C'est-à-dire au total 7 cases.

IV-2- Division

Exemple : Soit à effectuer les opérations suivantes : (i) +9 ÷ (+2) ; (ii) -11 ÷ 4

(i)
```
0 1001 | 0 10
  00   |-----
   01  | 0 100
    1  |
```

(ii)
```
1 1011 | 0 100
  0011 |-----
   011 | 1 10
```

EXERCICES

Exercice 1

Sachant que l'on dispose de 7 bits pour écrire les nombres, non compris le bit de signe, donner les expressions des nombres suivants : +8, +31, +15, +24, -8, -31, -15 , -24

a/ en valeur absolue plus signe (VAS)
b/ en complément restreint (CR)
c/ en complément vrai (CV)

Exercice 2

Trouver la valeur décimale des nombres binaires suivants exprimés en complément à 2 :
a/ 011011, b/ 111011, c/ 01111011, d/ 10011001, e/ 0111111, f/ 1100000.

Exercice 3

Combien de bits au moins faut-il pour représenter les nombres compris entre – 65535 et + 65535 ?

Exercice 4

Effectuer sur 5 bits les opérations suivantes :
31 – 24, 15 + 8, 8 – 8, – 8 – 15, – 24 – 31.
a/ en VAS, b/ en CR, c/ en CV

Exercice 5

Les nombres binaires suivants sont exprimés VAS.
Effectuer les calculs en CV.
a/ 11011011 – 01101011, b/ 1011 – 0011,
c/ 01101011 – 11011011.

Exercice 6

Effectuer les opérations suivantes (les nombres sont exprimés en sans signe) :
a/ 1110 x 101, b/ 1011 x 1011, c/ 111111 ÷ 1011,
d/ 1100 ÷ 100

Exercice 7

Effectuer les opérations suivantes en BCD :
a/ 174 + 46, b/ 95 + 55, c/ 103 + 358.

Scanned by CamScanner

CORRECTION

Exercice 1

a/ Expression en VAS

Entier relatif	Expression en VAS	Entier relatif	Expression en VAS
+8	0 0 0 0 1 0 0 0	-8	1 0 0 0 1 0 0 0
+31	0 0 0 1 1 1 1 1	-31	1 0 0 1 1 1 1 1
+15	0 0 0 0 1 1 1 1	-15	1 0 0 0 1 1 1 1
+24	0 0 0 1 1 0 0 0	-24	1 0 0 1 1 0 0 0

b/ Expression en CR

Entier relatif	Expression en CR	Entier relatif	Expression en CR
+8	0 0 0 0 1 0 0 0	-8	1 1 1 1 0 1 1 1
+31	0 0 0 1 1 1 1 1	-31	1 1 1 0 0 0 0 0
+15	0 0 0 0 1 1 1 1	-15	1 1 1 1 0 0 0 0
+24	0 0 0 1 1 0 0 0	-24	1 1 1 0 0 1 1 1

c/ Expression en CV

Entier relatif	Expression en CV	Entier relatif	Expression en CV
+8	0 0 0 0 1 0 0 0	-8	1 1 1 1 1 0 0 0
+31	0 0 0 1 1 1 1 1	-31	1 1 1 0 0 0 0 1
+15	0 0 0 0 1 1 1 1	-15	1 1 1 1 0 0 0 1
+24	0 0 0 1 1 0 0 0	-24	1 1 1 0 1 0 0 0

Exercice 2

a/$(011011)_{CV}$ → +27
b/$(111011)_{CV}$ → $(111010)_{CR}$ → $(100101)_{VAS}$ → -5
c/ $(01111011)_{CV}$ → +123
d/ $(10011001)_{CV}$ → $(11100111)_{VAS}$ → -103
e/ $(0111111)_{CV}$ → +63
f/ $(1100000)_{CV}$ → $(1100000)_{VAS}$ → -32

Exercice 3

On sait que, non compris le bit de signe; un nombre signé de n bits est compris dans l'intervalle $[-2^n+1 ; 2^n-1]$.

Posons $2^n-1=65535 \Rightarrow 2^n=65536 \Rightarrow n=\frac{\ln 65536}{\ln 2}=16$ bits.

Ainsi si nous comptons le bit de signe, pour représenter les nombres compris entre – 65535 et + 65535, il faut 17 bits.

Exercice 4

a/ opération en VAS sur 5 bits

31–24

0	11111
1	–11000
= 0	00111

15+8

0	01111
0	+ 01000
= 0	10111

8 – 8

0	01000
0	– 01000
= 0	00000

–24 – 31

1	11000
1	+ 11111
= 1	10111

–8 – 15

1	01000
1	+ 01111
= 1	10111

Résultat faux : débordement

b/ opération en CR sur 5 bits.

31–24 = 31+ (– 24)

0	11111
+ 1	00111
	0 00110
+	1
= 0	00111

15+8

0	01111
+ 0	01000
= 0	10111

8 – 8 = 8+ (– 8)

0	01000
+ 1	10111
= 1	11111

–24 – 31= –24+ (–31)

1	00111
+ 1	00000
	0 00111
+	1
= 0	01000

–8 – 15= –8+ (–15)

1	10111
+ 1	10000
	1 00111
+	1
= 1	01000

Résultat faux : débordement

c/ opération en CV sur 5 bits.

31–24 = 31+ (–24)

0	11111
+ 1	01000
= 0	00111

15+8

0	01111
+ 0	01000
= 0	10111

8 – 8= 8+ (–8)

0	01000
+ 1	11000
= 0	00000

–24 – 31= –24+ (–31)

1	00111
+ 1	00001
= 0	01000

–8 – 15 = –8 + (– 15)

1	11000
+ 1	10001
= 1	01001

Résultat faux : débordement

Exercice 5

a/ Exprimons tout d'abord les nombres en CV.
$(11011011)_{VAS}$ → **(10100101)**$_{CV}$
$-(01101011)_{VAS}$ → $(11101011)_{VAS}$ → **(10010101)**$_{CV}$.
D'où le calcul suivant :

	10100101
+	10010101
=	00111010

Il y a débordement.

b/ Expression des nombres en CV.
$(1011)_{VAS}$ → **(1101)**$_{CV}$
$-(0011)_{VAS}$ → $(1011)_{VAS}$ → **(1101)**$_{CV}$.
D'où le calcul suivant :

	1101
+	1101
=	1010

c/ Expression des nombres en CV.
$(01101011)_{VAS}$ → **(01101011)**$_{CV}$
$-(11011011)_{VAS}$ → $(01011011)_{VAS}$ → **(01011011)**$_{CV}$.
D'où le calcul suivant :

	01101011
+	01011011
=	11000110

Il y a débordement

Exercice 6

a/

	1110
x	101
	1110
	1110
=	1000110

b/

	1011
x	1011
	1011
	1011
	1110
=	1111001

c/

111111	1011
01001	101
10011	
1000	

d/

1100	100
0100	11
000	

Exercice 7

a/

	0001 0111 0100
+	0000 0100 0110
	0001 1011 1010
+	0110 0110
=	0010 0010 0000

b/

	1001 0101
+	0101 0101
	1110 1010
+	0110 0110
=	0001 0101 0000

c/

	0001 0000 0011
+	0011 0101 1000
	0100 0101 1011
+	0110
=	0100 0110 0001

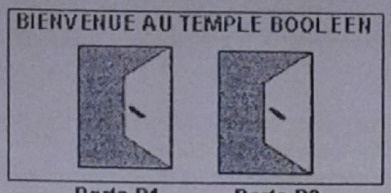
Scanned by CamScanner

CHAPITRE IV – ALGEBRE DE BOOLE - FONCTIONS LOGIQUES

I- INTRODUCTION AUX FONCTIONS LOGIQUES

I-1- Fonctions logiques et variables logiques

Soit le schéma ci-après sur lequel on étudie la relation R, définie par "**passage possible**" ; tel que si le passage est possible alors R vaut "**oui**", sinon R vaut "**non**".



On constate que la valeur de R dépend donc des états (ouvert ou fermé) des portes P1 et P2. Les différents cas de figure sont :

P1	P2	R
Fermée	Fermée	Non
Fermée	Ouverte	Oui
Ouverte	Fermée	Oui
Ouverte	Ouverte	Oui

En effet, tant que les deux portes ne sont pas simultanément fermées, le passage est possible, donc R est Oui.
La relation R ne pouvant prendre que deux valeurs (***oui*** ou ***non***) selon les étapes des portes P1 et P2 est donc ***une fonction logique***.
P1 et P2 sont les variables de la fonction logique R. Ces variables ne peuvent prendre que deux valeurs possibles : *ouverte* ou *fermée*. Ce sont donc des ***variables logiques*** (ou variables binaires ou variables booléennes).
Les fonctions logiques sont des fonctions associées à l'étude des systèmes utilisant deux états stables distincts représentés par 0 et 1. Une variable logique peut prendre deux états, représentés par 0 et 1.
Pour l'exemple précédent, si on choisit que pour les variables, l'état *fermé = 0*, l'état *ouvert =1* et pour la fonction R l'état *non = 0* et l'état *oui =1*, on a le tableau des différents cas de figure qui devient tout simplement :

P1	P2	R
0	0	0
0	1	1
1	0	1
1	1	1

Pour définir une fonction logique associée à des variables logiques, il faut déterminer l'état ou la valeur de la fonction pour chaque combinaison des variables logiques.

I-2- Combinaisons des variables logiques

Pour **n variables logiques** on a 2^n **combinaisons binaires possibles**.

Par exemple

pour 2 variables x, y on a $2^2 = 4$ combinaisons possibles :

x	y
0	0
0	1
1	0
1	1

pour 3 variables x, y, z on a $2^3 = 8$ combinaisons possibles :

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Remarque :
Pour obtenir les différentes combinaisons de n variables logiques, on suit généralement l'évolution du comptage en binaire naturel afin de se prémunir d'éventuelles répétitions ou omissions de combinaisons.
Pour n variables logiques il peut exister $(2^n)^2$ fonctions logiques possibles.
Par exemple, les différentes fonctions logiques possibles associées à deux variables binaires x, y sont au nombre de $(2^2)^2 = 16$; ce sont :

Variables binaires		Fonctions logiques possibles															
x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Scanned by CamScanner

I-3- Table de vérité d'une fonction logique

La table de vérité est un tableau qui présente la valeur logique que prend une fonction logique pour chaque combinaison de ses variables.

Exemple : Soit la fonction h portant sur les variables a et b telle que h =1 si au moins l'une des variables est égales à 1.
La table de vérité de h est :

a	b	h
0	0	0
0	1	1
1	0	1
1	1	1

I-4- Opérateurs logiques simples de base

Ce sont des signes et des symboles permettant d'exprimer ou de représenter 3 fonctions logiques essentielles. Ce sont :

- l'opérateur **non** (encore nommé **pas, complémentaire, inverseur, not**)
- l'opérateur **ou** (encore nommé **somme logique, or**)
- l'opérateur **et** (encore nommé **produit logique, and**)

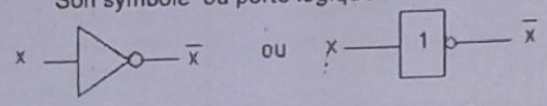
I-4-1- Opérateur NON

Il s'écrit " ¯ " . C'est un opérateur auquel est associée une fonction à une seule variable. Ceci $\bar{x}$, se lit **"x barre"** ou **"non x"**.

Sa table de vérité

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

Son symbole ou porte logique :



Remarque : la fonction $\bar{x} = 1$ si $x = 0$ et $\bar{x} = 0$ si $x = 1$

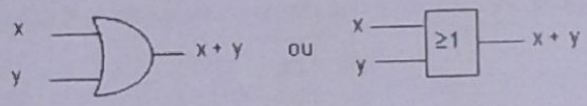
I-4-2- Opérateur OU

Il s'écrit " + ". C'est un opérateur auquel est associée une fonction à au moins deux variables. Ceci x + y, se lit **"x ou y"**.

Sa table de vérité

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

Son symbole ou porte logique :



Remarque : la fonction x + y = 1 si au moins une des variables est égale à 1.

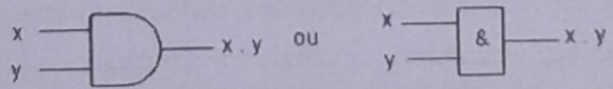
I-4-3- Opérateur ET

Il s'écrit ".". C'est un opérateur auquel est associée une fonction à au moins deux variables. Ceci "x . y", se lit **"x et y"** ou **"xy"**.

Sa table de vérité

x	y	x . y
0	0	0
0	1	0
1	0	0
1	1	1

Son symbole ou porte logique :



Remarque : la fonction x . y = 1 si toutes les variables sont égales à 1.

NB : x . y peut tout simplement s'écrire **xy**.

I-5- Opérateurs logiques complexes de base

Ce sont des signes et des symboles permettant d'exprimer ou de représenter tout simplement quatre fonctions logiques composées essentielles. Ce sont :

- l'opérateur non ou (encore nommé nor)
- l'opérateur non et (encore nommé nand)
- l'opérateur ou exclusif (encore nommé non coïncidence, dilemme, xor)
- l'opérateur non ou exclusif (encore nommé coïncidence, xnor)

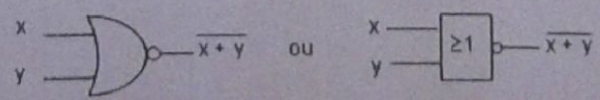
I-5-1- Opérateur NON OU

C'est un opérateur auquel est associée une fonction logique à au moins deux variables. Il résulte de l'association des opérateurs ou et non. Ceci $\overline{x+y}$, se lit "**non x ou y**" ou bien "**x ou y barre**" ou bien "**x NOR y**"

Sa table de vérité

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

Son symbole ou porte logique :



Remarque : la fonction $\overline{x+y} = 1$ si aucune des variables n'est égale à 1.

Scanned by CamScanner

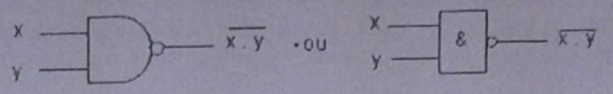
I-5-2- Opérateur NON ET

C'est un opérateur auquel est associée une fonction logique à au moins deux variables. Il résulte de l'association des opérateurs et et non. Ceci $\overline{x \cdot y}$, se lit "non xy" ou bien "xy barre" ou bien "**x NAND y**"

Sa table de vérité

x	y	$\overline{x \cdot y}$
0	0	1
0	1	1
1	0	1
1	1	0

Son symbole ou porte logique :



Remarque : la fonction $\overline{x \cdot y}$ = 1 si au moins une des variables est égale à 0.

I-5-3- Opérateur OU EXCLUSIF

C'est un opérateur auquel est associée une fonction logique à au moins deux variables.

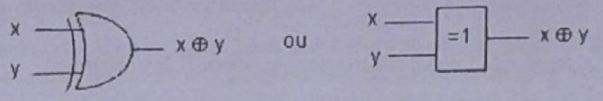
L'expression de cette fonction est $x\overline{y} + \overline{x}y$. Elle se note tout simplement $x \oplus y$.

L'écriture $x \oplus y$ se lit "**x non coïncidence y**" ou bien "**x dilemme y**" ou bien "**x ou exclusif y**" ou bien "**x Xor y** "

Sa table de vérité

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Son symbole ou porte logique :



Remarque : la fonction $x \oplus y = 1$ si et seulement si les variables sont différentes.

I-5-4- Opérateur NON OU EXCLUSIF

C'est un opérateur auquel est associée une fonction logique à au moins deux variables

L'expression de cette fonction est $xy + \overline{x}\overline{y}$.

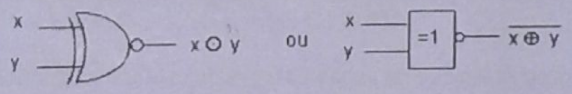
C'est la fonction complémentaire de la fonction ou exclusif. Elle se note $\overline{x \oplus y}$ ou tout simplement $x \odot y$.

L'écriture $x \odot y$ se lit "**x coïncidence y**" ou bien "**x non ou exclusif y**" ou bien "**x XNor y**"

Sa table de vérité

x	y	$x \odot y$
0	0	1
0	1	0
1	0	0
1	1	1

Son symbole ou porte logique :

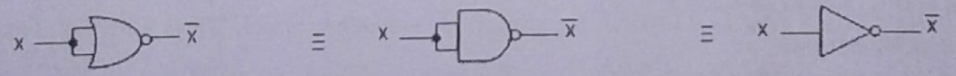


Remarque : la fonction $x \odot y = 1$ si et seulement si les variables sont égales.

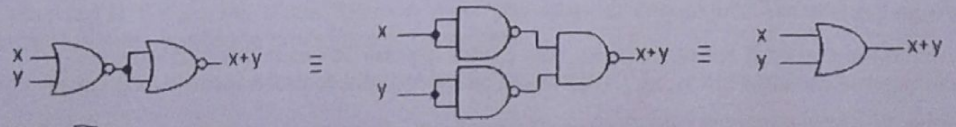
I-6- Universalité des portes NON OU et NON ET.

Les portes **NON OU** et **NON ET** peuvent servir à la réalisation de toutes les autres portes de base. On dit donc de ces deux portes qu'elles sont universelles.

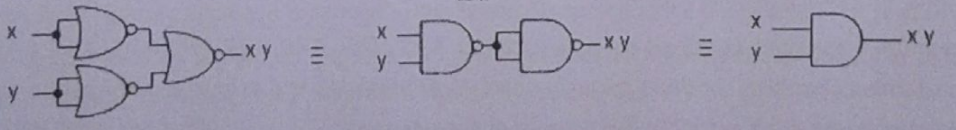
- Porte NON :



- Porte OU :



- Porte ET :



II- ALGÈBRE DE BOOLE

II-1- Propriétés

Soit l'ensemble binaire E= {0 ; 1} sur lequel on définit les lois
"+" (Somme logique)
"." (Produit logique) et
" ¯ " (Complément).

L'ensemble E muni de ces trois lois ({ E, +, . , ¯ }) est un algèbre de Boole ou algèbre binaire si les trois lois vérifient les propriétés suivantes pour toutes variables x, y, z éléments de E :

Scanned by CamScanner

a- Commutativité

Du produit logique : $x\,y = y\,x$

De la somme logique : $x + y = y + x$

b- Associativité

Du produit logique : $(x\,y)\,z = x\,(y\,z) = x\,y\,z$

De la somme logique : $(x + y) + z = x + (y + z) = x + y + z$

c- Distributivité

Du produit logique sur la somme : $x\,(y+z) = x\,y + x\,z$

De la somme logique sur le produit logique : $x + y\,z = (x + y)\,(x + z)$

d- Idempotence

Du produit logique : $x\,x = x$ (***une variable n'admet pas de puissance***)

De la somme logique : $x + x = x$ (***une variable n'admet pas de multiple***)

e- Elément absorbant

Du produit logique : $x\,.\,0 = 0$ (***0 est l'élément absorbant du produit logique***)

De la somme logique : $x + 1 = 1$ (***1 est l'élément absorbant de la somme logique***)

f- Elément neutre

Du produit logique : $x\,.\,1 = x$ (***1 est l'élément neutre du produit logique***)

De la somme logique : $x + 0 = x$ (***0 est l'élément neutre de la somme***)

g- Complément

Le complément de x est $\bar{x}$. Ainsi le complément du complément de $\bar{x}$ est $\bar{\bar{x}} = x$.

Pour le produit logique : $x\bar{x} = 0$ Pour la somme logique : $x + \bar{x} = 1$

h- Dualité

Pour les variables : $\bar{1} = 0$ et $\bar{0} = 1$ Pour les lois : $\overline{\bullet} = +$ et $\overline{+} = \bullet$

II-2-Théorèmes

II-2-1- Théorème de De Morgan

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

Démonstration : $\overline{x + y} = \bar{x}\,\bar{+}\,\bar{y} = \bar{x} \cdot \bar{y}$

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

Démonstration : $\overline{x \cdot y} = \bar{x}\,\bar{\cdot}\,\bar{y} = \bar{x} + \bar{y}$

II-2-2- Théorème de la fausse inhibition

$x + \bar{x}y = x + y$

Démonstration : Par la distributivité de la somme logique sur le produit logique on a : $x + \bar{x}y = (x + \bar{x})(x + y) = 1.(x + y) = x + y$.

III- ETUDE DES FONCTIONS LOGIQUES

III-1- Mintermes et Maxtermes

Soient $x_1, x_2, \ldots, x_n \in \{0\,;1\}$.

- On appelle minterme construit sur $x_1, x_2, \ldots, x_n$, tout produit logique de ces n variables logiques.
- On appelle maxterme construit sur $x_1, x_2, \ldots, x_n$, toute somme logique de ces n variables logiques.

Exemple : Soient a, b, c trois variables binaires.

- Tous les mintermes possibles construits sur a, b, c sont : $\bar{a}\bar{b}\bar{c},\ \bar{a}\bar{b}c,\ \bar{a}b\bar{c},\ \bar{a}bc,\ a\bar{b}\bar{c},\ a\bar{b}c,\ ab\bar{c},\ abc$
- Tous les maxtermes possibles construits sur a, b, c sont : $\bar{a}+\bar{b}+\bar{c},\ \bar{a}+\bar{b}+c,\ \bar{a}+b+\bar{c},\ \bar{a}+b+c,\ a+\bar{b}+\bar{c},\ a+\bar{b}+c,\ a+b+\bar{c},\ a+b+c$

On constate bien que quand n = 3, on a $2^3 = 8$ mintermes et 8 maxtermes.

D'une manière générale, pour n variables on a 2^n mintermes et 2^n maxtermes possibles.

III-2- Représentation d'une fonction logique

Une fonction logique de n variables peut être représentée par :

- Une table de vérité
- Une représentation algébrique
- Une représentation numérique
- Une représentation graphique

Scanned by CamScanner

III- 2-1- Représentation algébrique d'une fonction logique

a- Représentation d'une fonction logique sous la première forme canonique

La représentation d'une fonction logique sous la première forme canonique est l'expression de celle-ci sous forme de **sommes logiques de mintermes.** Cette représentation est aussi appelée forme **disjonctive** ou forme **sommes de produits canoniques.** On la note simplement forme **SP canonique** ou **Σπ.**

La première forme canonique d'une fonction s'obtient à partir de la table de vérité dans laquelle on ne s'intéresse qu'aux combinaisons des variables pour lesquelles la fonction prend la valeur 1; puis on écrit l'expression de F sous forme de sommes des différents mintermes pour lesquels F vaut 1 . En effet, dans un minterme on complémente une variable quand elle vaut 0 et on ne la complémente pas quand elle vaut 1.

Exemple :

On veut construire une machine à voter pour 3 électeurs A, B, C. Chacun exprime son suffrage en appuyant sur le bouton poussoir qui lui est destiné. Le vote se fait par OUI (1) ou par NON (0). Une lampe L doit s'allumer (L=1) si le OUI l'emporte à la majorité absolue.

La table de vérité définissant l'état de L pour chaque combinaison des variables d'entrée A, B, C est ci-contre :

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

Il ressort que la forme SP canonique est :

$$L = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

b- Représentation d'une fonction logique sous la deuxième forme canonique

La représentation d'une fonction logique sous la deuxième forme canonique est l'expression de celle-ci sous forme de **produits logiques de maxtermes.** Cette représentation est aussi appelée forme **conjonctive** ou forme **produits de sommes canoniques.** On la note simplement forme **PS canonique** ou **πΣ.**
Pour déterminer la deuxième forme canonique d'une fonction logique F, on peut utiliser deux méthodes :

- **1ère méthode :**
  - Dresser la table de vérité de $\bar{F}$ et trouver la première forme canonique de $\bar{F}$
  - Complémenter $\bar{F}$ (c'est-à-dire $\bar{\bar{F}}$) et lui appliquer le théorème de De Morgan.

Exemple : Soit à déterminer la deuxième forme canonique de la fonction L précédente.

La table de vérité de $\bar{L}$ est déduite de celle de L en complémentant chaque valeur de L :

A	B	C	L	$\bar{L}$
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

Forme SP canonique de $\bar{L}$: $\bar{L} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$

Complément de $\bar{L}$: $\bar{\bar{L}} = L = \overline{\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}}$

D'après le théorème de De Morgan :

$$L = \overline{\bar{A}\bar{B}\bar{C}}\cdot\overline{\bar{A}\bar{B}C}\cdot\overline{\bar{A}B\bar{C}}\cdot\overline{A\bar{B}\bar{C}} = (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$$

- **2ème méthode :**

  Dans la table de vérité F, s'intéresser aux combinaisons pour lesquelles la fonction prend la valeur 0 et écrire l'expression de F sous forme de produits des différents maxtermes pour lesquels F vaut 0. En effet, dans un maxterme on complémente une variable quand elle vaut 1 et on ne la complémente pas quand elle vaut 0.

Ainsi, de la table de vérité de L, on déduit la 2ème forme canonique directement par : $L = (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$

Remarque: L'expression sous forme SP ou PS d'une fonction n'est pas toujours canonique. Toute fois, quelle que soit son expression, on pourra toujours la mettre sous forme canonique si besoin est.

➢ ***Passage d'une forme SP non canonique à une forme SP canonique***

Exemple : Soit à mettre sous la 1ère forme canonique la fonction sous forme SP non canonique suivante : $F = \bar{A} + B + AC$.

La méthode consiste à transformer tous les termes qui ne sont pas des mintermes en mintermes, en faisant le produit logique de chacun de ces termes par $(x+\bar{x})(y+\bar{y})$ telles que x et y soient les variables manquantes.

Ainsi, la valeur de F ne change pas si l'on écrit :

$$F = \bar{A}(B+\bar{B})(C+\bar{C}) + B(A+\bar{A})(C+\bar{C}) + AC(B+\bar{B}) = \bar{A}B\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$$

En éliminant les termes redondants (répétitifs), on a $F = \bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + ABC + AB\bar{C} + A\bar{B}C$

Passage d'une forme PS non canonique à une forme PS canonique

Exemple : Soit à mettre sous la 2e forme canonique la fonction sous forme PS non canonique suivante : $H = \bar{A}B(A+C)$

La méthode la plus simple est de passer par sa **forme SP** en procédant comme suit :

Scanned by CamScanner

- A partir de la forme PS non canonique, complémenter la fonction et appliquer le théorème de De Morgan : *on obtient la forme SP non canonique de la fonction complémentée.*
- Rendre sous forme SP canonique la fonction complémentée.
- Complémenter à nouveau la fonction complémentée obtenue sous forme SP canonique : *on obtient alors la forme PS canonique de la fonction.*

Ainsi, $\overline{H} = \overline{\overline{A}B(A+C)} = A + \overline{B(A+C)} = A + \overline{B} + \overline{(A+C)} = A + \overline{B} + \overline{A}\,\overline{C}$ *(Forme SP non canonique de $\overline{H}$)*

Il ressort que : $\overline{H} = A(B+\overline{B})(C+\overline{C}) + \overline{B}(A+\overline{A})(C+\overline{C}) + \overline{A}\,\overline{C}(B+\overline{B})$

Après distribution et élimination des termes redondants on a :

$\overline{H} = A\,B\,C + A\,B\,\overline{C} + A\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C}$ *(Forme SP Canonique de $\overline{H}$)*

Ce qui donne : $\overline{\overline{H}} = H = \overline{A\,B\,C + A\,B\,\overline{C} + A\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C}}$

D'où $H = (\overline{A}+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)(\overline{A}+B+\overline{C})(\overline{A}+B+C)(A+B+\overline{C})(A+B+C)(A+\overline{B}+C)$ ***(Forme PS canonique de H)***

III-2-2- Représentation numérique d'une fonction logique

a- Représentation numérique par écriture indicielle

Cette représentation est surtout utilisée pour représenter les fonctions logiques données sous forme Σπ canonique ou sous forme de table de vérité.
Après avoir pondéré les variables suivant les puissances croissantes de deux, chaque minterme se voit affecter la valeur décimale correspondante à la combinaison des variables. On l'appelle aussi **représentation à base 10.**

Exemple :

- Soit à donner l'écriture indicielle de la machine à voter étudiée au paragraphe III-2-1.

La pondération positionnelle donne pour les trois variables les poids 1, 2, 4 :

Variables / Poids / valeurs décimales	A 4	B 2	C 1	L
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

On effectue la somme logique des mintermes pour lesquels la fonction L = 1.
L'écriture indicielle de L est donc notée **L = Σ(3, 5, 6, 7).**
La somme logique n'étant rien d'autre que la réunion dans le langage des ensembles, l'écriture indicielle de L se note aussi **L = R(3, 5, 6, 7).**

- Soit à donner la représentation numérique par écriture indicielle de la fonction F donnée sous forme SP canonique suivante : $F = DC\overline{B}A + D\overline{C}\,\overline{B}A + \overline{D}C\overline{B}\,\overline{A} + \overline{D}\,\overline{C}\,\overline{B}A + DCBA$

On écrit sous chaque terme la combinaison binaire correspondante puis on attribut à chaque combinaison obtenue la valeur décimale correspondante.

Ainsi, $F = DC\overline{B}A + D\overline{C}\,\overline{B}A + \overline{D}C\overline{B}\,\overline{A} + \overline{D}\,\overline{C}\,\overline{B}A + DCBA$

↓ ↓ ↓ ↓ ↓
1101 1001 0100 0001 1111 (*Combinaisons binaires correspondantes aux mintermes*)
↓ ↓ ↓ ↓ ↓
13 9 4 1 15 (*Valeurs décimales de chaque combinaison binaire*)

D'où sous forme indicielle **F = R (1, 4, 9, 13, 15).**

Remarque : La représentation numérique par écriture indicielle est canonique par définition. Il est indispensable de s'en tenir toujours à un ordre de pondération croissante des variables. C'est pour cela qu'il est préférable de préciser le plus souvent l'ordre choisi pour les poids des variables.

b- Représentation numérique par image caractéristique

On appelle **image caractéristique** ou **vecteur caractéristique**, la suite des valeurs prises par la fonction pour toutes les combinaisons des variables dans l'ordre binaire naturel. C'est en fait le résultat de la table de vérité écrite en ligne.
L'image caractéristique d'une fonction F, se note ÷F.
Lorsqu'on représente une fonction par image caractéristique, on regroupe généralement ses bits par groupe de 4 bits pour en faciliter l'écriture et la lecture. La représentation par image caractéristique est implicitement canonique.

Exemple : Soit à donner la représentation par image caractéristique de la fonction L étudiée :

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

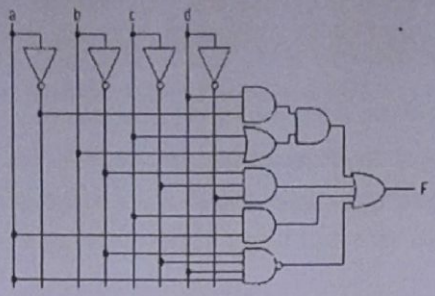
La représentation par image caractéristique de L est : **÷L = 0001 0111**

Scanned by CamScanner

III-3- Réalisation des fonctions logiques avec des portes logiques

La réalisation d'une fonction logique avec des portes logiques revient à construire **le diagramme logique** ou **logigramme** de ladite fonction.
Un logigramme est donc une représentation graphique d'une fonction à l'aide des symboles des opérateurs logiques qui interviennent dans son expression algébrique.

Exemple : Soit à construire le logigramme de la fonction $F = d\bar{a}(b+c) + \bar{b}\bar{c}\bar{d} + ac + \overline{\bar{a}\bar{b}\bar{c}d}$ telle qu'elle est donnée.



III-4- Simplification des fonctions logiques

La simplification d'une fonction logique répond à un souci d'écrire l'expression de la fonction plus simplement avec moins de termes ou moins de variables par terme, tout en gardant le même résultat logique que l'expression initiale plus complexe.
L'expression obtenue avec le minimum de termes et (ou) de variables par terme est appelée **forme minimale** de la fonction logique.
La forme minimale d'une fonction permet généralement de construire un logigramme avec moins de portes logiques. Ce dernier est donc moins encombrant et certainement peu coûteux à réaliser.
Lorsque la forme minimale d'une fonction est écrite sous forme de sommes de produits, chaque terme est appelé **monôme premier** ou **impliquant** de la fonction.
Il existe plusieurs méthodes pour obtenir la forme minimale d'une fonction logique. Mais nous allons en étudier deux, d'ailleurs les plus utilisées. La première méthode dite méthode algébrique, est basée sur des calculs se fondant sur les théorèmes et propriétés de l'algèbre de boole. La seconde méthode dite de Karnaugh, est une méthode graphique basée sur une démarche assez simple.

III-4-1- Simplification algébrique de fonctions logiques

Exemple :

- soit à simplifier algébriquement la fonction $F = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$

1er cas d'approche :

$F = \bar{x}yz + xy\bar{z} + xz(y+\bar{y})$
$F = \bar{x}yz + xy\bar{z} + xz$
$F = xy\bar{z} + x(y\bar{z}+z)$
$F = \bar{x}yz + x(y+z)$
$F = \bar{x}yz + xy + xz$
$F = y(\bar{x}z+x) + xz$
$F = y(z+x) + xz$
$F = yz + yx + xz$

2ème cas d'approche :

$F = \bar{x}yz + x\bar{y}z + xy(\bar{z}+z)$
$F = \bar{x}yz + x\bar{y}z + xy$
$F = \bar{x}yz + x(\bar{y}z+y)$
$F = \bar{x}yz + x(z+y)$
$F = \bar{x}yz + xz + xy$
$F = z(\bar{x}y+x) + xy$
$F = z(y+x) + xy$
$F = zy + zx + xy$

- soit à simplifier algébriquement $H = (a+b+c+\overline{abc})d\bar{e}$

1er cas d'approche

$H = [(a+b+c) + \overline{(a+b+c)}]d\bar{e}$
$H = 1 \cdot d\bar{e}$
$H = d\bar{e}$

2ième cas d'approche

$H = [(a+\overline{abc}) + b + c]d\bar{e}$
$H = [a+b+\bar{b}\bar{c}+c]d\bar{e}$
$H = [a+b+\bar{c}+c]d\bar{e}$
$H = 1 \cdot d\bar{e}$
$H = d\bar{e}$

Remarque : En utilisant les règles de l'algèbre booléenne nous pouvons simplifier l'expression d'une fonction logique. Mais la simplification par les règles de calcul (simplification algébrique) n'est pas toujours évidente. Elle nécessite beaucoup d'intuitions, une parfaite connaissance des postulats des théorèmes fondamentaux booléens et parfois de l'expérience.
L'une des méthodes les plus simples pour simplifier l'expression d'une fonction logique est la méthode de Karnaugh.

III-4-2- Simplification de fonctions logiques par la méthode de Karnaugh

La méthode de Karnaugh est une méthode graphique qui facilite la simplification de l'équation d'une fonction logique obtenue ou non à partir d'une table de vérité. Cette méthode est basée sur la construction d'un tableau dit tableau ou diagramme de Karnaugh dans lequel chaque case se voit affecter la valeur correspondante de la fonction. Et en procédant par des regroupements convenables de cases, on obtient la forme minimale donnant les monômes premiers de la fonction.

Dans un tableau de karnaugh, la disposition des cases est basée sur l'ordre d'évolution des combinaisons binaires du code binaire réfléchi (encore nommé code reflex ou code gray) et les regroupements ne portent que sur des cases **adjacentes** *(la définition de ce terme par la suite).*

Scanned by CamScanner

a - Propriétés du code binaire réfléchi

- **Propriété d'adjacence :**

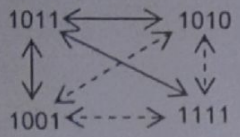
Deux minternes sont adjacents si l'on passe de l'un à l'autre en complémentant seulement une seule des variables qui les composent. De même deux mots binaires sont adjacents s'ils ne diffèrent que d'un seul bit.

Exemple :

- Soit le minterme suivant : ABCDE

Les mintermes $\bar{A}BCDE$ et $A\bar{B}CDE$ sont tous les deux adjacents au premier mais pas entre eux.

- Soient les mots binaires suivants : 1011, 1010, 1001, 1111. En établissant les relations d'adjacence entre eux on obtient :



<--> : Ne sont pas adjacents

<--> : Sont adjacents

- **Code continu :**

Un code est dit continu si et seulement si deux combinaisons successives sont adjacentes.

- **Code cyclique :**

Un code est dit cyclique si et seulement si la dernière combinaison est adjacente à la première.

Remarque :
Le code binaire réfléchi est un code continu et cyclique.
On le constate bien d'après le tableau du code gray pour quatre variables dressé ci-contre. (Pour plus de détail, voir chapitre II).

Tableau du code Gray pour quatre Variables W, X, Y, Z

Z	Y	X	W
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

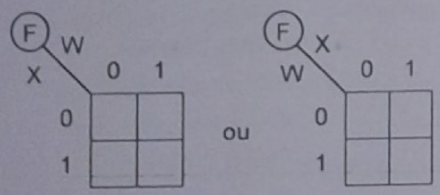
b- Tableau de karnaugh

Disposition du tableau de karnaugh

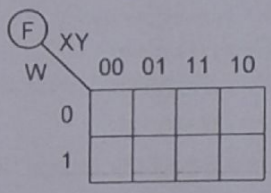
Soit F une fonction logique à **n variables**. Alors le tableau de karnaugh nécessaire pour la simplification de l'équation de F comporte 2^n **cases.**

On a généralement la disposition suivante :

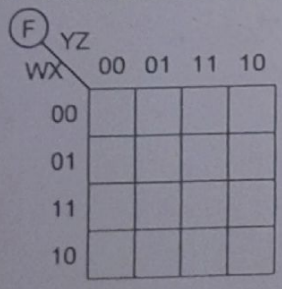
- **Pour une fonction F à deux variables W, X :**



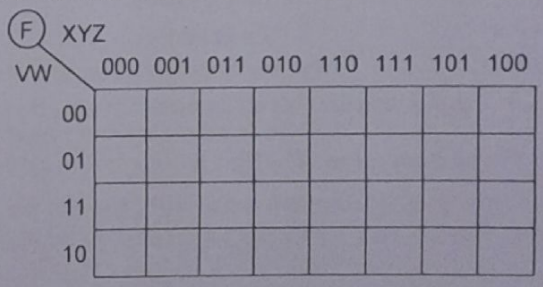
- **Pour une fonction F à trois variables W, X, Y :**



- **Pour une fonction F à quatre variables W, X, Y, Z :**



- **Pour une fonction F à cinq variables V, W, X, Y, Z :**



Remarque :

Un tableau de karnaugh comprend des axes de symétrie.
Dans un tableau de karnaugh,
- Les cases ayant un côté commun sont adjacentes car elles correspondent à des combinaisons adjacentes.
- Les cases symétriques par rapport à un axe quelconque sont adjacentes.

Exemple : soit le tableau ci-dessous dans lequel quelques cases sont marquées $\alpha, \beta, \gamma, \delta, \varepsilon, \rho, \omega$.

F cde / ab	000	001	011	010	110	111	101	100
00		α					δ	ρ
01		β					ε	
11		γ						
10								ω

- Les traits en gras sont les axes de symétrie principaux.
- Tous les autres traits sont chacun un axe de symétrie secondaire.
- Parmi les cases marquées, celles qui sont adjacentes sont :

α et β, α et δ, β et ε, β et γ, δ et ε, δ et ρ, ω et ρ. En effet, $\alpha = \bar{a}\bar{b}\bar{c}de = 00001$, $\beta = \bar{a}\bar{b}\bar{c}de = 01001$, $\delta = \bar{a}\bar{b}\bar{c}de = 00101$, ...

- Par ailleurs l'ensemble de cases (α , β) est adjacent à l'ensemble (δ , ε)

Scanned by CamScanner

▸▸ Utilisation du tableau de karnaugh pour la simplification d'une fonction logique.

Pour obtenir l'équation logique simplifiée d'une fonction à partir du tableau de karnangh , il faut :

- *Remplir le tableau de karnangh à partir de la table de vérité de la fonction ou à partir de son expression logique non simplifiée.*
- *Faire les regroupements judicieux :*
  - ✓ Un regroupement est une boucle renfermant le **maximum** de cases adjacentes contenant 1.
  - ✓ Un regroupement peut contenir une case, deux cases, quatre cases,..., 2^n cases.
  - ✓ Dans le but d'obtenir les regroupements les plus vastes possibles, des cases contenant 1 peuvent appartenir à plusieurs regroupements : on dit qu'il y a recouvrement de regroupement.
  - ✓ Un tableau est fait de regroupements judicieux s'il contient le minimum de regroupements possibles.
- ***Ecrire l'expression simplifiée de la fonction sous la forme de sommes de produits en appliquant la règle suivante :***

  Chaque regroupement judicieux donne un terme de la fonction simplifiée, c'est-à-dire un monôme premier.
  - ✓ Si une variable quelconque **x garde la valeur 1** pour toutes les cases contenues dans un regroupement, elle s'écrit **x** dans le terme correspondant.
  - ✓ Si une variable quelconque **x garde la valeur 0** pour toutes les cases contenues dans un regroupement, elle s'écrit $\overline{x}$ dans le terme correspondant.
  - ✓ Si une variable quelconque **x change de valeur (0 ou 1)** pour quelques cases contenues dans un regroupement, elle s'écrit **1** dans le terme correspondant : ***soit donc à omettre cette valeur dans le terme car 1 est l'élément neutre du produit logique.***

c- Simplification par tableau de karnaugh à partir de la table de vérité d'une fonction

Soit à déterminer l'équation simplifiée de la fonction L dont la table de vérité est la suivante :

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

Il en ressort le tableau de karnaugh suivant :

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

D'après les regroupements l'expression de L est :

$$L = \mathbf{BC} + \mathbf{AC} + AB$$

d- Simplification par tableau de karnaugh à partir de l'expression algébrique non simplifiée d'une fonction

Soit à simplifier à partir d'un digramme de karnaugh, la fonction suivante :

$$H = A\overline{B}CD + \overline{A}\overline{B}\overline{C}D + ABCD + AB\overline{C} + \overline{A}D$$

1er terme, 2ème terme, 3ème terme, 4ème terme, 5ème terme

On constate bien que H est une fonction à 4 variables A, B, C, D. Alors les trois premiers termes dans l'expression ci-dessus correspondent chacun à une seule case dans le tableau de karnaugh car ce sont des minternes de H. Donc dans chacune de ces cases, H vaut 1. Pour le 4ème terme, il suffit que la condition $AB\overline{C}$ soit réalisée, quelle que soit la valeur de D, pour que H prenne la valeur 1.

On a donc $AB\overline{C}\emptyset_D = AB\overline{C}D$ ou $AB\overline{C}\,\overline{D}$ (*pour l'écriture $ABC\emptyset_D$, lire ABC quelle que soit la valeur de D*). Ainsi le terme $AB\overline{C}$ porte sur deux cases, et pour celles-ci H vaut 1.

Le 5ème terme donne $\overline{A}\emptyset_B\emptyset_C D = \overline{A}\,\overline{B}\,\overline{C}D$ ou $\overline{A}\,\overline{B}CD$ ou $\overline{A}B\overline{C}D$ ou $\overline{A}BCD$. Donc ce terme porte sur 4 cases pour lesquelles H vaut alors 1.

D'où le diagramme de karnaugh suivant :

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

On déduit alors que $H = AB\overline{C} + CD + \overline{A}D$

e- Simplification par tableau de Karnaugh d'une fonction ayant des valeurs non connues.

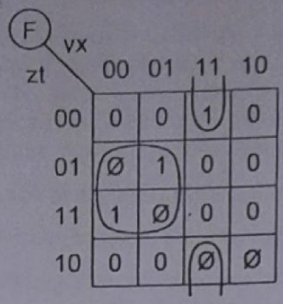
Pour n variables on a 2^n combinaisons possibles. Si une fonction à n variables n'est définie que pour quelques unes des 2^n combinaisons possibles (c'est-à-dire que F prend explicitement 0 ou 1 que pour seulement certaines combinaisons des variables), alors les autres combinaisons pour lesquelles l'état de la fonction n'est pas connu peuvent être :

Scanned by CamScanner

- Soit des combinaisons ***interdites*** des variables d'entrée ; c'est-à-dire que d'après la définition de la fonction (donc du système), ces combinaisons ne doivent jamais apparaître ou être obtenues.
- Soit des combinaisons ***indifférentes*** des variables d'entrée ; c'est-à-dire que ces combinaisons n'influent pas sur le fonctionnement normal du système.
- Soit des combinaisons ***inutilisées*** des variables d'entrée ; c'est-à-dire que ces combinaisons ne font pas partie des combinaisons normalement utiles pour l'étude de la fonction.

Dans tous les cas, quelles que soient leurs origines, pour ces combinaisons, l'état inconnu de la fonction est matérialisé par Ø (ou par **x**) dans la table de vérité et dans le tableau de Karnaugh correspondant.
Par ailleurs, on cherchera donc toujours à inclure les Ø (ou x) dans des regroupements si et uniquement si cela permet d'effectuer des regroupements plus intéressants. Dans ce cas les Ø (ou x) dans les regroupements sont vus comme des 1.

Exemple : Soit à donner l'expression de la fonction F dont le tableau de Karnaugh est ci-contre :

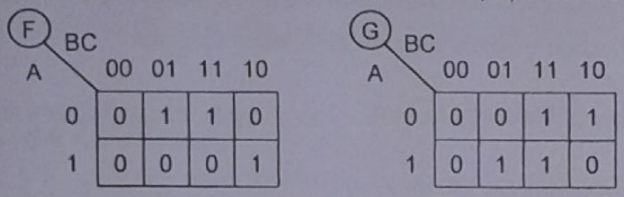


$\Rightarrow F = \bar{t}\bar{y} + \bar{t}yx$

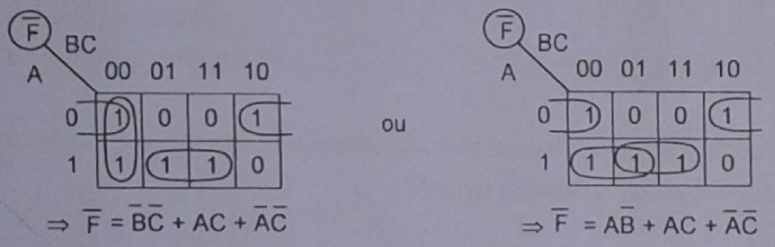
f- Opération logique sur les tableaux de karnaugh

On utilise les propriétés matricielles du tableau de Karnaugh pour obtenir le complément, la réunion (somme logique, OU), l'intersection (produit logique, ET), le dilemme (XOR) etc.... de deux fonctions représentées sur des tableaux de même dimension. En effet, on effectue case par case la somme logique (OU), le produit logique (ET), le dilemme (XOR), le complémentaire de chaque minterne correspondant.

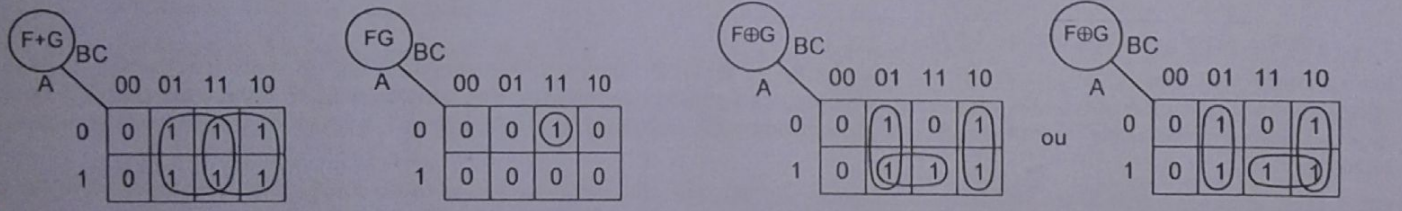
Exemple : soient Fet G, deux fonctions définies sur les mêmes variables A, B, C dont les tableaux de Karnaugh sont ci-dessous.



A partir de ces tableaux, on demande de déterminer les fonctions $\overline{F}$, F+G, FG, F⊕G



$\Rightarrow \overline{F} = \bar{B}\bar{C} + AC + A\bar{C}$ ou $\Rightarrow \overline{F} = A\bar{B} + AC + A\bar{C}$



$\Rightarrow F+G = B + C$ $\Rightarrow FG = \bar{A}BC$ $\Rightarrow F\oplus G = \bar{B}C + B\bar{C} + AC$ ou $\Rightarrow F\oplus G = \bar{B}C + B\bar{C} + AB$

g- Quelques insuffisances des tableaux de Karnaugh

Les tableaux de karnaugh permettent de simplifier les fonctions logiques en les écrivant avec des opérateurs ET, OU, et NON. Malheureusement dans le cas où les équations se simplifieraient très bien à l'aide d'opérateurs NON ET, NON OU, OU EXCLUSIF, NON OU EXCLUSIF, il est souvent difficile de le voir directement sur le tableau de karnaugh. Il est donc indispensable quelque fois, après avoir sortie l'expression d'une fonction par tableau de karnaugh d'y appliquée les règles de calcul de base ou celle de De Morgan afin d'obtenir un logigramme beaucoup plus simple. (Voir l'exercice 6 de ce chapitre pour une illustration).

Scanned by CamScanner

EXERCICES

Exercice 1
Simplifier algébriquement les fonctions suivantes :

$F_1 = (a\bar{b}+c)(a+\bar{b})c$ $F_2 = (x+y)z+\bar{x}(\bar{y}+z)+\bar{y}$ $F_3 = a+\bar{a}b+\bar{a}\bar{b}c+\bar{a}\bar{b}\bar{c}d+\bar{a}\bar{b}\bar{c}\bar{d}e$ $F_4 = (\bar{a}b+a\bar{b})(ab+\bar{a}\bar{b})$ $F_5 = ab\bar{c}+b(a+\bar{c})+\overline{\bar{a}+b+\bar{a}c}$

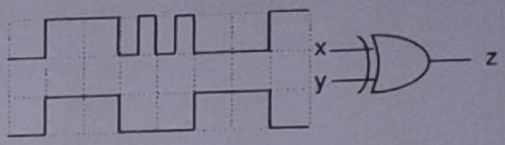
$F_6 = (\bar{a}+b)(a+\bar{b})+(a+b)(\bar{a}+\bar{b})$

Exercice 2
Mettre les fonctions suivantes sous la première et la deuxième forme canonique :

$H_1 = (x+y)(\bar{x}+y+z)$ $H_2 = \bar{x}\bar{y}+xyz\bar{t}$ $H_3 = \overline{B\bar{C}}+A\bar{B}C$

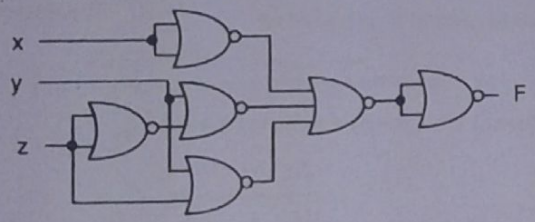
Exercice 3
Une association est constituée de quatre membres dont un président. Face à un problème épineux ils décident de voter. Etant donné leur nombre, il est décidé que le président (A) aura seul droit à un vote de poids double. Les autres membres (B, C et D) ont une seule voix chacun.
Réaliser le logigramme de la machine à voter dont la sortie S = 1 si la majorité de oui est obtenue. (On donne que un oui correspond à 1 et un non à 0).

Exercice 4
1/ Donner le signal de sortie du circuit suivant sachant les signaux d'entrée :



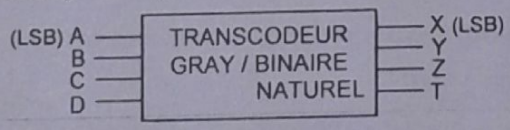
2/ Montrer que $a(b \oplus c) = ab \oplus ac$.

Exercice 5
Soit le logigramme ci-dessous :

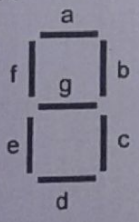


1/ Donner l'expression algébrique de F sous forme de sommes de produits.
2/ Simplifier F par la méthode de karnaugh.
3/ Réaliser le logigramme simplifier de F à l'aide de portes NOR, puis de portes NAND.

Exercice 6
Réaliser le logigramme de transcodage sur 4 quatre bits du code gray en binaire naturel.



Exercice 7



Donner le logigramme du transcodage nécessaire pour obtenir à partir du code BCD, les chiffres décimaux correspondants sur un afficheurs 7 segments.
On précise que chaque segment du dispositif d'affichage des chiffres décimaux ci-contre s'allume pour un niveau logique 1.

Exercice 8
Soient les fonctions logiques suivantes : $F = \bar{A}\bar{B}+\bar{A}\bar{C}+ABC+\bar{B}D$ et $H = \bar{A}\bar{D}+\bar{B}C+\bar{A}\bar{B}\bar{C}\bar{D}$
1/ Simplifier chaque fonction par un diagramme de karnaugh
2/ Réaliser le logigramme donnant F et H à l'aide :
a/ de portes logiques hybrides.
b/ de portes Non Et uniquement.
c/ de portes Non Ou uniquement.

Exercice 9
Soit la fonction complémentée suivante : $\bar{F} = (\bar{A}+\bar{C})\,[(\bar{A}+\bar{B})\overline{(B+\bar{D})}+AB(B+\bar{D})]$
1/ Trouver la forme minimale de F en utilisant les propriétés matricielles du tableau de karnaugh.
On posera $G = \bar{A}+\bar{C}$, $H = \bar{A}+\bar{B}$, $K = B+\bar{D}$
2/ Construire le logigramme de F.

Exercice 10
La représentation numérique indicielle d'une fonction logique F à cinq variables A, B, C, D, E dont dans leur combinaison A est le LSB, est la suivante :
$F(A, B, C, D, E) = R(0,4,9,13,17,18,26,30,31) + Ø(1,2,3,5,16,20,21,27)$
Donner l'expression logique simplifiée de F.

Scanned by CamScanner

Exercice 11

On considère un dispositif de régulation de température T dans une pièce, utilisant trois capteurs de température Z, Y, X correspondant respectivement à trois températures de référence TM (température maximum), TI (température intermédiaire) et Tm (température minimum) , avec TM > TI > Tm.

Les capteurs donnent les informations suivantes : Z = 1 si T > TM, Y = 1 si T > TI, X = 1 si T > Tm.

On détecte également si la température croît ou décroît, à l'aide d'un détecteur de pente P tel que P=1 si T croît.

Le dispositif permet d'actionner l'ouverture d'un volet d'aération V (tel que V = 1 ⇒ volet ouvert) et la mise en marche d'un radiateur de chauffage R (tel que R = 1 ⇒ chauffage).

Les conditions de fonctionnement du volet sont les suivantes : le volet est fermé si T croît et T< TM ou si T décroît et T< TI.

Les conditions de fonctionnement du radiateur sont les suivantes : le radiateur chauffe si T croît et T < TI ou si T décroît et T< Tm.

1/ Réaliser le dispositif avec des protes NAND à deux entrées.

2/ Le circuit intégré 4011 contient quatre portes NAND à deux entrées. Combien faut-il de circuits intégrés 4011 pour réaliser le dispositif ?

CORRECTION

Exercice 1

$F_1 = (a\bar{b}+c)(a+\bar{b})c = (a\bar{b}+a\bar{b}+ac+\bar{b}c)c = (a\bar{b}+ac+\bar{b}c)c = a\bar{b}c+ac+\bar{b}c = ac+\bar{b}c$

$F_2 = (x+y)z+\bar{x}(\bar{y}+z)+\bar{y} = xz+yz+\bar{x}\bar{y}+\bar{x}z+\bar{y} = (xz+\bar{x}z)+(\bar{x}\bar{y}+\bar{y})+yz = z+\bar{y}+yz = (z+yz)+\bar{y} = z+\bar{y}$

$F_3 = a+\bar{a}b+\bar{a}\bar{b}c+\bar{a}\bar{b}\bar{c}d+\bar{a}\bar{b}\bar{c}\bar{d}e = a+b+\bar{a}\bar{b}(c+\bar{c}d+\bar{c}\bar{d}e) = (a+b)+\overline{(a+b)}(c+d+\bar{c}\bar{d}e) = a+b+c+\bar{c}e+d = a+b+c+d+e$

$F_4 = (\bar{a}b+a\bar{b})(ab+\bar{a}\bar{b}) = (\bar{a}b+a\bar{b})\overline{(\bar{a}b+a\bar{b})} = 0$

$F_5 = ab\bar{c}+b(a+\bar{c})+\overline{\bar{a}+b+\bar{a}c} = ab\bar{c}+ab+b\bar{c}+a\bar{b}(a+\bar{c}) = ab+b\bar{c}+a\bar{b}+a\bar{b}\bar{c} = a(b+\bar{b})+\bar{c}(b+a\bar{b}) = a+\bar{c}(b+a) = a+a\bar{c}+b\bar{c} = a+b\bar{c}$

$F_6 = (\bar{a}+b)(a+\bar{b})+(a\uparrow b)(\bar{a}+\bar{b}) = \bar{a}\bar{b}+ab+a\bar{b}+\bar{a}b = \overline{a\oplus b}+a\oplus b = 1$

Exercice 2

Première forme canonique:

$H_1 = (x+y)(\bar{x}+y+z) = xy+xz+\bar{x}y+y = xy(z+\bar{z})+xz(y+\bar{y})+xy(z+\bar{z})+y(x+\bar{x})(z+\bar{z}) = xyz+xy\bar{z}+x\bar{y}z+\bar{x}yz+\bar{x}y\bar{z}$

$H_2 = \bar{x}\bar{z}+xyz\bar{t} = \bar{x}\bar{z}(y+\bar{y})(t+\bar{t})+xyz\bar{t} = \bar{x}y\bar{z}t+\bar{x}y\bar{z}\bar{t}+\bar{x}\bar{y}\bar{z}t+\bar{x}\bar{y}\bar{z}\bar{t}+xyz\bar{t}$

$H_3 = \overline{B\bar{C}}+A\bar{B}C = \bar{B}+C+A\bar{B}C = \bar{B}(A+\bar{A})(C+\bar{C})+C(A+\bar{A})(B+\bar{B})+A\bar{B}C = A\bar{B}C+A\bar{B}\bar{C}+\bar{A}\bar{B}C+\bar{A}\bar{B}\bar{C}+ABC+\bar{A}BC$

Deuxième forme canonique :

$H_1 = (x+y)(\bar{x}+y+z)$. Cette valeur ne change pas si on écrit $H_1 = [(x+y)+z\bar{z}](\bar{x}+y+z)$.

En appliquant la distributicn de la somme sur le produit logique on a : $H_1 = [(x+y)+z][(x+y)+\bar{z}](\bar{x}+y+z) = (x+y+z)(x+y+\bar{z})(\bar{x}+y+z)$

On a la table de vérité suivante de H_2 d'après sa première forme canonique plus haut :

x	y	z	t	H_2
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

En tenant compte des combinaisons pour lesquelles $H_2 = 0$ et par écriture de produits de sommes, on a la deuxième forme canonique :

$H_2 = (x+y+\bar{z}+t)(x+y+\bar{z}+\bar{t})(x+\bar{y}+\bar{z}+t)(x+\bar{y}+\bar{z}+\bar{t})(\bar{x}+y+z+t)(\bar{x}+y+z+\bar{t})(\bar{x}+y+\bar{z}+t)(\bar{x}+y+\bar{z}+\bar{t})(\bar{x}+\bar{y}+z+t)(\bar{x}+\bar{y}+z+\bar{t})(\bar{x}+\bar{y}+\bar{z}+t)$.

D'après la première forme canonique de H_3, en considérant que A = MSB, sa représentation indicielle est : $H_3 = R(0, 1, 3, 4, 5, 7)$

On déduit donc que la représentation indicielle de $\bar{H}_3$ est : $\bar{H}_3 = R(2, 6)$. Ce qui donne $\bar{H}_3 = \bar{A}B\bar{C}+AB\bar{C}$.

Donc $H_3 = \overline{\bar{A}B\bar{C}+AB\bar{C}} = (A+\bar{B}+C)(\bar{A}+\bar{B}+C)$

Scanned by CamScanner

Exercice 3

La table de vérité est :

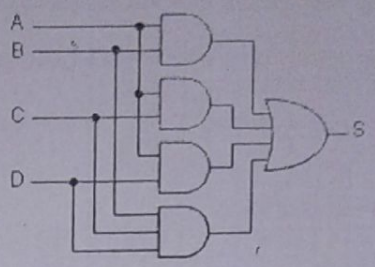
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Son expression logique

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

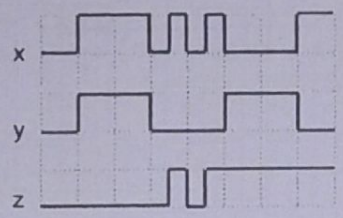
S =AB+AD+AC+BCD

Son logigramme :



Exercice 4

1/



2/ On a : $a(b\oplus c) = a(b\bar{c} + \bar{b}c) = ab\bar{c} + a\bar{b}c$.

Par ailleurs, $ab\oplus ac = ab.\overline{ac} + \overline{ab}\,ac$

$= ab(\bar{a} + \bar{c}) + ac(\bar{a} + \bar{b})$

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

Donc $a(b\oplus c) = ab\oplus ac$.

Exercice 5

1/ $F = \overline{\bar{x} + \overline{y+\bar{z}} + \overline{y+z}} = \bar{x} + \overline{y+\bar{z}} + \overline{y+z} = \bar{x} + \bar{y}z + \bar{y}\bar{z}$

2/

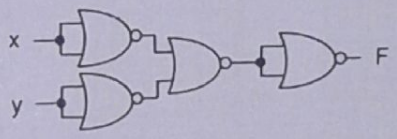
F	yz 00	01	11	10
x 0	1	1	1	1
1	1	1	0	0

$\Rightarrow F = \bar{x} + \bar{y}$

3/

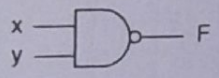
Logigramme à portes NOR :

On a $F = \bar{x} + \bar{y} = \overline{\bar{x} + \bar{y}}$



Logigramme à portes NAND :

On a $F = \bar{x} + \bar{y} = \overline{x\,y}$



Exercice 6

Table de vérité du transcodeur code Gray / code binaire naturel

D	C	B	A	T	Z	Y	X
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1'	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Equations logiques des sorties

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

$\Rightarrow T = D$

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

$\Rightarrow Z = \bar{D}C + D\bar{C}$

$= D\oplus C$

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

$\Rightarrow Y = \bar{D}\bar{C}B + DCB + D\bar{C}\bar{B} + \bar{D}C\bar{B}$

$= B\overline{(D\oplus C)} + \bar{B}(D\oplus C)$

$= (D\oplus C)\oplus B$

$= D\oplus C\oplus B$

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

$\Rightarrow X = \bar{B}\bar{A}(\bar{D}C + D\bar{C}) + BA(\bar{D}C + D\bar{C}) + \bar{B}A(\bar{D}\bar{C} + DC) + B\bar{A}(\bar{D}\bar{C} + DC)$

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

$= (D\oplus C)\oplus(B\oplus A)$

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

Logigramme

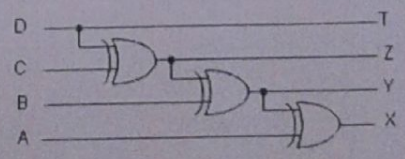
On constate que les équations des sorties suivent une évolution linéaire.
On peut ainsi exprimer les unes en fonction des autres :

$T = D$

$Z = D\oplus C \Rightarrow Z = T\oplus C$

$Y = D\oplus C\oplus B = (D\oplus C)\oplus B \Rightarrow Y = Z\oplus B$

$X = D\oplus C\oplus B\oplus A = (D\oplus C\oplus B)\oplus A \Rightarrow X = Y\oplus A$

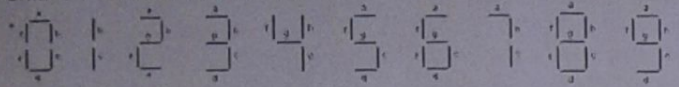


Entrée Code Gray — Sortie Code binaire naturel

Scanned by CamScanner

Exercice 7

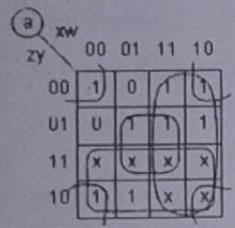
Avec 4 variables il y a 16 combinaisons possibles allant de 0000 à 1111. Or le code BCD va de 0000 à 1001. Donc le système doit être indifférent aux 6 dernières combinaisons binaires.
Par ailleurs, les segments qui s'allument pour donner chaque chiffre décimal de 0 à 9 sont :



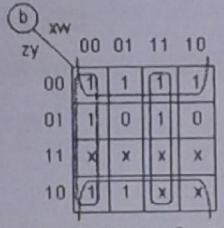
D'où la table de vérité ci-dessous :

z	y	x	w	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

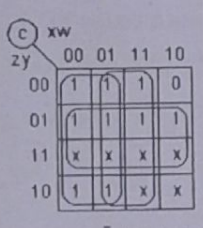
Expression des sorties du circuit de transcodage :



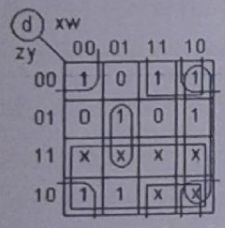
a = $\bar{y}\bar{w}+yw+x+z = y \odot w+x+z$



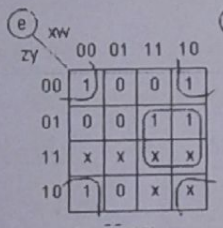
b = $\bar{y}+\bar{x}\bar{w}+xw = \bar{y}+x \odot w$



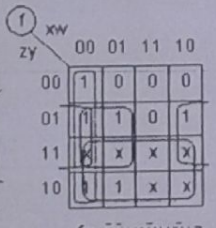
c = $\bar{x}+w+y$



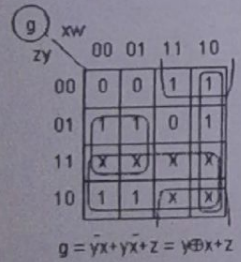
d = $z+x\bar{w}+\bar{y}\bar{w}+\bar{y}x+y\bar{x}w$



e = $\bar{y}\bar{w}+yx$

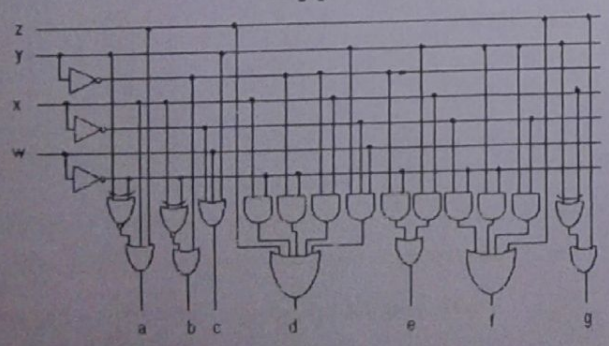


f = $\bar{x}\bar{w}+y\bar{w}+y\bar{x}+z$



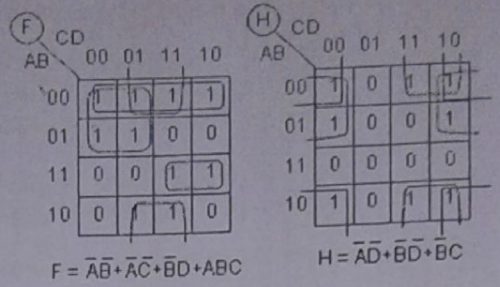
g = $\bar{y}x+y\bar{x}+z = y \oplus x+z$

Logigramme



Exercice 8

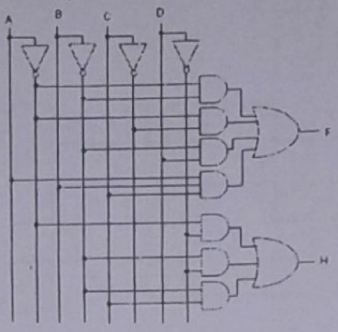
1/ Simplification des fonctions F et H par diagramme de karnaugh :



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

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

2/
a/ Logigramme à portes hybrides



b/ Logigramme à portes NON ET uniquement

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

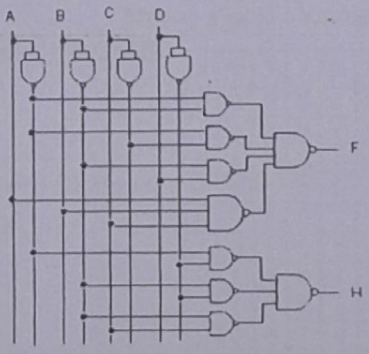
$\overline{\overline{F}} = \overline{\overline{\bar{A}\bar{B}+\bar{A}\bar{C}+\bar{B}D+ABC}}$

$F = \overline{\bar{A}\bar{B}.\ \bar{A}\bar{C}.\ \bar{B}D.\ ABC}$ *(écriture en NON ET de F)*

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

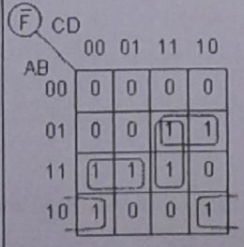
$\overline{\overline{H}} = \overline{\overline{\bar{A}\bar{D}+\bar{B}\bar{D}+\bar{B}C}}$

$H = \overline{\bar{A}\bar{D}.\ \bar{B}\bar{D}.\ \bar{B}C}$ *(écriture en NON ET de H)*



c/ Logigramme à portes NON OU uniquement

D'après le tableau de karnaugh de F on ressort celui de $\bar{F}$ comme ci-dessous :



D'où on déduit que :

$\bar{F} = \bar{A}BC+BCD+AB\bar{C}+A\bar{B}\bar{D}$

$\Rightarrow \overline{\bar{F}} = \overline{\bar{A}BC+BCD+AB\bar{C}+A\bar{B}\bar{D}}$

$= \overline{\bar{A}BC}\ \overline{BCD}\ \overline{AB\bar{C}}\ \overline{A\bar{B}\bar{D}}$

$= (A+\bar{B}+\bar{C})(\bar{B}+\bar{C}+\bar{D})(\bar{A}+\bar{B}+C)(\bar{A}+B+D)$

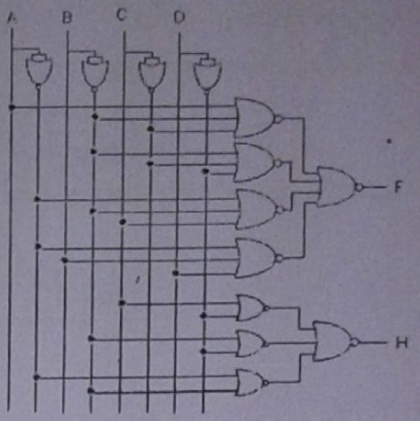
$\overline{\overline{F}} = \overline{(A+\bar{B}+\bar{C})(\bar{B}+\bar{C}+\bar{D})(\bar{A}+\bar{B}+C)(\bar{A}+B+D)}$

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

(écriture en NON OU de F)

Scanned by CamScanner

Appliquons une autre méthode pour l'écriture de H sous forme de produits de sommes : ***l'exploitation d'un diagramme de karnaugh par regroupements de 0.***



Règle : lorsqu'on veut donner l'expression simplifiée d'une fonction logique sous forme de produits de sommes on peut dans le tableau de karnaugh de ladite fonction procéder par des regroupements pertinents de 0.
Ainsi chaque regroupement est un terme représenté par une somme logique des variables correspondantes.
En définitive, l'équation de la fonction est exprimée sous forme de produits des différents termes tels que dans un regroupement :

- *Si une variable x garde la valeur 0, elle s'écrit x dans le terme correspondant.*
- *Si une variable x garde la valeur 1, elle s'écrit $\bar{x}$ dans le terme correspondant.*
- *Si une variable x change de valeur (0 ou 1), elle s'écrit 0 dans le terme correspondant. Comme 0 est l'élément neutre de la somme logique, il est donc omis dans l'écriture du terme.*

H

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

$$H = \overline{(C+\bar{D})(\bar{B}+\bar{D})(\bar{A}+\bar{B})}$$
$$= (C+\bar{D})(\bar{B}+\bar{D})(\bar{A}+\bar{B})$$
$$= \overline{(C+\bar{D})}+\overline{(\bar{B}+\bar{D})}+\overline{(\bar{A}+\bar{B})}$$ *(écriture en NON OU de H)*

Exercice 9

$\bar{F} = (\bar{A}+\bar{C})[(\bar{A}+\bar{B})\overline{(B+\bar{D})} + AB(B+\bar{D})]$

En posant $G = (\bar{A}+\bar{C})$, $H = \overline{AB}$, $K = B+\bar{D}$ on a $\bar{F} = G(H\bar{K}+\bar{H}K) = G(H \oplus K)$

La fonction $\bar{F}$ a comme variables G, H et K ; elles mêmes qui sont des fonctions dépendantes variables A, B, C et D.
Les tableaux de karnaugh de G, H et K sont donc :

G

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

H

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

K

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

Tableau de H⊕K

H⊕K

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

Tableau de $\bar{F} = G(H \oplus K)$

$\bar{F}$

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

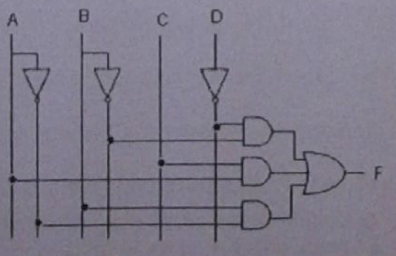
On déduit le tableau de karnaugh de F

F

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

$\Rightarrow F = \bar{D}\bar{B} + CA + B\bar{A}$

2 / Logigramme



Scanned by CamScanner

Exercice 10

1/ F(A, B, C, D, E) = R(0, 4, 9, 13, 17, 18, 26, 30, 31) + Ø(1, 2, 3, 5, 16, 20, 21, 27) avec A = LSB

Tableau de karnaugh de F :

F CBA / ED	000	001	011	010	110	111	101	100
00	1	ϕ	ϕ	ϕ	0	0	ϕ	1
01	0	1	0	0	0	0	1	0
11	0	0	ϕ	1	1	1	0	0
10	ϕ	1	0	1	0	0	ϕ	ϕ

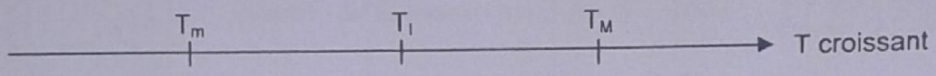
$\Rightarrow F = \overline{DB} + \overline{EBA} + EDB + \overline{DCA}$

Exercice 11

1/ Les variables d'entrée : ce sont les informations issues des capteurs : Z, Y, X et P.
Les variables de sortie : ce sont les actions effectuées par le volet et le radiateur : V et R.

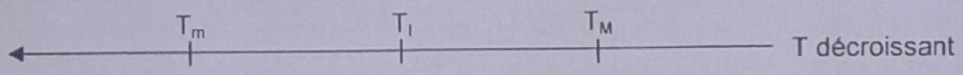
D'après l'énoncé nous pouvons ressortir le schéma d'évolution suivant :

- Lorsque la température croît alors P = 1. ainsi on a :



Comportement des entrées selon l'évolution de la température	P = 1 Z = 0 Y = 0 X = 0	P = 1 Z = 0 Y = 0 X = 1	P = 1 Z = 0 Y = 1 X = 1	P = 1 Z = 1 Y = 1 X = 1
Etat des sorties par intervalle de température	V = 0 R = 1	V = 0 R = 1	V = 0 R = 0	V = 1 R = 0

- Lorsque la température décroît alors P = 0. ainsi on a :



Comportement des entrées selon l'évolution de la température	P = 0 Z = 0 Y = 0 X = 0	P = 0 Z = 0 Y = 0 X = 1	P = 0 Z = 0 Y = 1 X = 1	P = 0 Z = 1 Y = 1 X = 1
Etat des sorties par intervalle de température	V = 0 R = 1	V = 0 R = 0	V = 1 R = 0	V = 0 R = 1

On constate que le fonctionnement du dispositif n'est explicitement défini que pour les 8 combinaisons ci-dessus des variables d'entrée. Les autres combinaisons ne pouvant jamais apparaître à cause de la condition $T_m < T_I < T_M$.
Il ressort la table de vérité suivante :

P	Z	Y	X	V	R
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	Ø	Ø
0	0	1	1	1	0
0	1	0	0	Ø	Ø
0	1	0	1	Ø	Ø
0	1	1	0	Ø	Ø
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	Ø	Ø
1	0	1	1	0	0
1	1	0	0	Ø	Ø
1	1	0	1	Ø	Ø
1	1	1	0	Ø	Ø
1	1	1	1	1	0

Expressions des sorties

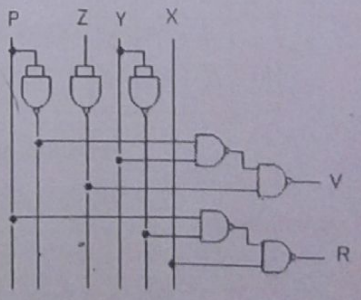
V / PZ \ YX	00	01	11	10
00	0	0	1	Ø
01	Ø	Ø	1	Ø
11	Ø	Ø	1	Ø
10	0	0	0	Ø

$V = \overline{P}Y + \overline{Z} = \overline{\overline{P}Y + Z} = \overline{\overline{P}Y \cdot Z}$

R / PZ \ YX	00	01	11	10
00	1	0	0	Ø
01	Ø	Ø	0	Ø
11	Ø	Ø	0	Ø
10	1	1	0	Ø

$R = P\overline{Y} + \overline{X} = \overline{P\overline{Y} + X} = \overline{P\overline{Y} \cdot X}$

Logigramme



2/ On a au total eu besoin de 7 portes logiques NAND à deux entrées parmi lesquelles certaines sont utilisées en inverseuse. Puisque le circuit intégré 4011 contient 4 portes NAND à deux entrées, alors la réalisation du dispositif nécessite deux circuits intégrés.

Scanned by CamScanner

CHAPITRE V – TECHNOLOGIE DES CIRCUITS LOGIQUES

I- DIFFERENTES TECHNOLOGIES DES CIRCUITS LOGIQUES

Les fonctions logiques peuvent être matérialisées par l'intermédiaire de circuits intégrés logiques. Ces circuits sont réalisés selon différentes technologies dont les plus fréquemment croisées sont :

- La technologie à transistor bipolaire : TTL (Transistor Transistor Logic)
- La technologie à transistor à effet de champ : CMOS (Complementary Metal Oxide Semiconductor)

Pour un fonctionnement logique identique, chaque technologie offre des performances différentes sur le plan électrique (tensions, courants, puissances) et temporel (rapidité).
Dans chaque technologie on dénombre plusieurs familles se distinguant par la référence.

I-1- Familles de la technologie TTL

Familles	Référence
TTL Standard	74xx
TTL Low power	74Lxx
TTL Fast	74Fxx
TTL Schottky	74Sxx
TTL Low power Schottky	74LSxx
TTL Advanced Schottky	74ASxx
TTL Advanced Low power Schottky	74ALSxx

I-2- Familles de la technologie CMOS

Familles	Référence
CMOS Classique	74Cxx ou série 4000
CMOS High Speed	74HCxx 74HCTxx
CMOS Advanced	74ACxx 74ACTxx
CMOS Low voltage	74LVxx 74LVCxx

II- CARACTERISTIQUES TECHNOLOGIQUES DES CIRCUITS LOGIQUES

II-1- Tension d'alimentation

	Familles	Tensions d'alimentation
Technologie TTL	Toutes les familles	5 V
Technologie CMOS	CMOS Classique (74Cxx ou série 4000)	3 V à 18 V
	CMOS High Speed (74HCxx et 74HCTxx)	2 V à 6 V
	CMOS Advanced (74ACxx et 74ACTxx)	3 V à 5,5 V
	CMOS Low Voltage (74LVxx et 74LVCxx)	2 V à 3,6 V

Remarque : Parmi les familles de la technologie CMOS, la famille CMOS Classique série 4000 est la plus couramment croisée.

II-2- Niveaux logiques d'un circuit logique

Les états ou niveaux logiques sont caractérisés par des valeurs de tensions dont les limites sont précisées dans les documents constructeurs :
État 0 : niveau bas, absence de tension (Low level, L)
État 1 : niveau haut, présence de tension (High level, H)

V_{CC} ou V_{DD} - tension d'alimentation : niveau de tension nécessaire pour alimenter le circuit.

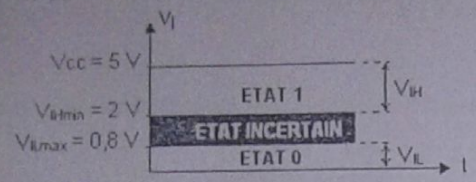
V_{IH} - tension d'entrée niveau HAUT : niveau de tension nécessaire pour avoir un 1 logique en entrée.
V_{IL} - tension d'entrée niveau BAS : niveau de tension nécessaire pour avoir un 0 logique en entrée.
V_{OH} - tension de sortie niveau HAUT : niveau de tension de la sortie d'un circuit logique correspondant à l'état logique 1.
V_{OL} - tension de sortie niveau BAS : niveau de tension de la sortie d'un circuit logique correspondant à l'état logique 0.

I_{IH} - courant d'entrée niveau HAUT : le courant qui traverse une borne d'entrée quand une tension niveau haut est appliquée à cette entrée.
I_{IL} - courant d'entrée niveau BAS : le courant qui traverse une borne d'entrée quand une tension niveau bas est appliquée à cette entrée.
I_{OH} - courant de sortie niveau HAUT : le courant qui traverse une borne de sortie placée au niveau logique 1 dans des conditions de charge spécifiées.
I_{OL} - courant de sortie niveau BAS : le courant qui traverse une borne de sortie placée au niveau logique 0 dans des conditions de charge spécifiées.

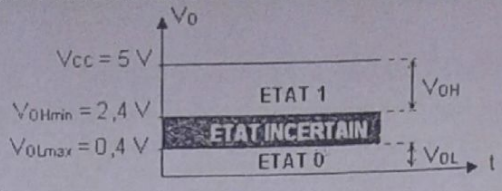
Scanned by CamScanner

Par exemple les documents constructeurs précisent :

- Pour la famille TTL standard (famille de référence de cette technologie), les niveaux de tensions suivantes :



Les niveaux logiques d'entrées



Les niveaux logiques de sorties

- Pour la famille CMOS Classique :

Les niveaux logiques d'entrées :
- Entrée au niveau haut : $(V_{IHmin} = 70\%V_{DD}) \leq V_{IH} \leq V_{DD}$
- Entrée au niveau bas : $0 \leq V_{IL} \leq (V_{ILmax} = 30\%\ V_{DD})$

Les niveaux logiques de sorties :
- Sortie au niveau haut : $V_{OH} = V_{DD}$
- Sortie au niveau bas : $V_{OL} = 0$

II-3- Immunité aux bruits

Un bruit est une variation intempestive (ou aléatoire) d'une grandeur physique autour de la valeur déterminée (permanente, attendue ou nominale).
L'immunité aux bruits est la marge de sécurité que l'on peut observer sur les variations d'entrées sans que cela entraîne un changement d'état logique non désiré en sorties d'un circuit logique.

- Marge de sensibilité aux bruits état haut : $V_{NH} = V_{OHmin} - V_{IHmin}$
- Marge de sensibilité aux bruits état bas : $V_{NL} = V_{ILmax} - V_{OLmax}$

Par exemple pour la famille TTL standard, $V_{NH} = 2,4 - 2 = 0,4$ V et $V_{NL} = 0,8 - 0,4 = 0,4$ V

II-4- Performances dynamiques

Les performances dynamiques principales sont :
- les temps de transition d'un signal entrant ou sortant d'un circuit logique.
- les temps de propagation d'un signal entre l'entrée et la sortie d'un circuit logique.

Temps de transition
La transition de l'état bas à l'état haut ou la transition contraire d'un signal logique n'est pas rigoureusement une transition raide.
Les temps de transition sont les temps que mettent les signaux sortant d'un circuit logique pour passer d'un niveau logique à un autre.
Deux temps de transition sont définis :
t_r – temps de montée (transition bas–haut) : temps mis par un signal logique pour passer de 10% à 90% de sa valeur maximale.
t_f – temps de descente (transition haut–bas) : temps mis par un signal logique pour passer de 90% à 10% de sa valeur maximale.

Temps de propagation
Un signal logique qui traverse un circuit subit toujours un retard.
Le temps de propagation ou **retard de propagation** est le temps que met un signal logique pour traverser un circuit logique.
Deux retards de propagation sont définis :
t_{PHL} : retard pour passer du niveau logique 1 au niveau logique 0.
t_{PLH} : retard pour passer du niveau logique 0 au niveau logique 1.

Exemple avec une porte inverseuse :

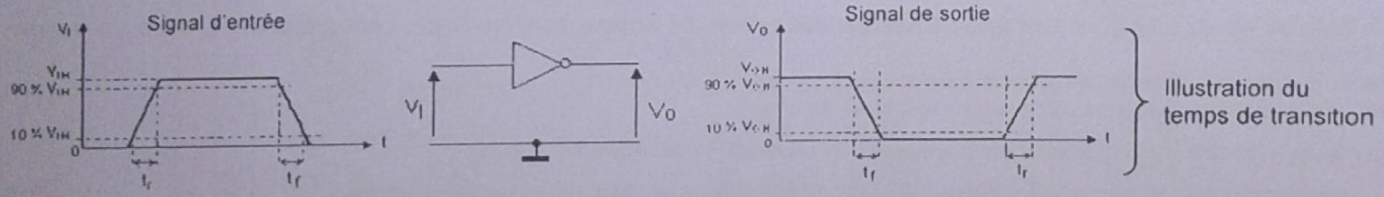


Illustration du temps de transition

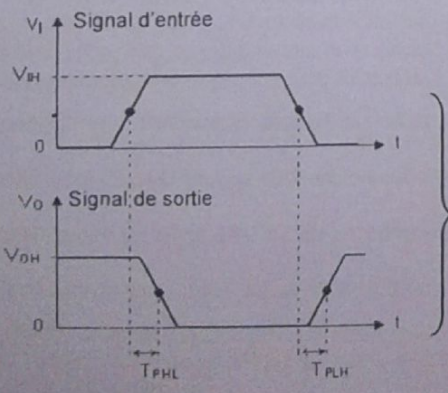


Illustration du temps de propagation

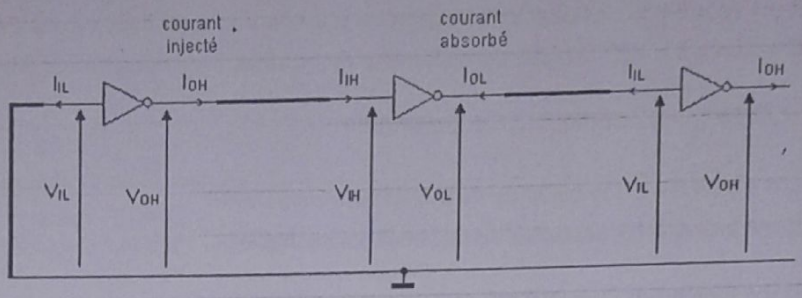
Scanned by CamScanner

II-5- Courant injecté et courant absorbé

Lorsqu'une sortie au niveau haut attaque une entrée, ce niveau haut n'est garanti que si la sortie arrive à fournir le courant nécessaire à l'entrée pilotée. C'est-à-dire si le **courant injecté** à cette entrée est suffisant.

Lorsqu'une sortie au niveau bas attaque une entrée, ce niveau bas n'est garanti que si la sortie arrive à absorber tout le courant sortant de l'entrée pilotée. C'est-à-dire si le **courant absorbé** par cette sortie ne dépasse pas sa capacité d'absorption de courant

Exemple d'indication des différents courants :



Les documents constructeurs précisent pour la famille TTL standard (famille de référence de cette technologie), les valeurs de courants suivants :

I_{OHmax} = **0,4 mA** : courant maximal que peut fournir (injecter) une sortie au niveau haut.
I_{OLmax} = **16 mA** : courant maximal que peut recevoir (absorber) une sortie au niveau bas.
I_{IH} = **40 µA** : courant consommé par une entrée maintenue au niveau haut.
I_{IL} = **1,6 mA** : courant extrait d'une entrée maintenue au niveau bas.

II-6- Sortance

Normalement, la sortie d'un circuit logique doit pouvoir piloter plusieurs entrées logiques.
La sortance (appelée également **facteur de charge**) est définie comme le nombre maximal d'entrées logiques **standards** qui peuvent être pilotées sans problème par une sortie.
Par exemple, quand il est indiqué qu'une porte logique a une sortance de 10, cela signifie qu'elle peut piloter 10 entrées logiques standards. Si on dépasse ce nombre, il n'est pas assuré que les tensions des niveaux logiques des sorties seront exactes.

II-7- Règles d'association des circuits TTL et CMOS

D'usage, on ne mélange pas plusieurs technologies entre elles. Cependant il existe des structures d'interface permettant de relier deux technologies différentes. Ces structures peuvent être matérialisées par l'intermédiaire de résistances et de transistors afin que les niveaux des tensions exigés en entrée et fournis en sortie pour les états hauts et bas soient compatibles.

II-8- Technologies internes des portes logiques

II-8-1- Technologies internes des portes logiques de la technologie TTL

Les portes logiques de la technologie TTL existent en trois modèles différents :

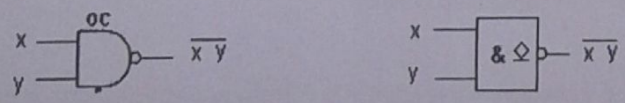
- **Le modèle à sortie totem pôle**

Le modèle à sortie totem pôle est le modèle le plus répandu dans toutes les familles de la technologie TTL. La sortie totem pôle a la particularité d'avoir un temps de propagation faible et de dissiper une faible puissance. Deux sorties totem pôle ne doivent jamais être reliées sinon les circuits dissiperont une très forte puissance destructrice lorsque ces deux sorties n'ont pas le même niveau logique.
Les symboles de telles portes sont tous les symboles exploités au chapitre précédent.

- **Le modèle à sortie collecteur ouvert (open collector output)**

Le modèle à sortie collecteur ouvert ne fonctionne que si une résistance est mise entre la sortie et la tension d'alimentation. C'est un avantage permettant selon la valeur de la résistance de disposer sur la sortie d'un courant assez important et donc de commander de fortes charges comme des afficheurs à LED, moteurs, relais, etc.
Contrairement à la sortie totem pôle, plusieurs sorties à collecteur ouvert peuvent être reliées afin de réaliser un "ET câblé". C'est-à-dire si plusieurs sorties à collecteur ouvert S_1, S_2, S_3,... sont reliées alors la sortie S résultante vaut $S = S_1 . S_2 . S_3 . \ldots$
Pour faire la distinction entre une porte à sortie totem pôle et une porte à sortie collecteur ouvert on indique OC à côté du symbole pour la symbolisation anglo-saxonne et pour la symbolisation française un losange à l'intérieur.

Exemple : porte NAND à deux entrées à sortie collecteur ouvert.



- **Le modèle à sortie trois états (3-state output)**

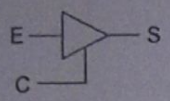
Les modèles à sortie trois états (3-state output) encore appelé **tampons trois états (tri-state buffers)** sont des portes dont on se sert pour contrôler le passage d'un signal logique entre une entrée et une sortie. Pour assurer ce contrôle, les sorties de ces portes peuvent avoir, en fonction d'une entrée de commande, l'un des trois états suivants :

- **l'état 1**
- **l'état 0**
- **l'état flottant ou Haute impédance** noté HZ

Scanned by CamScanner

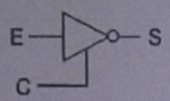
L'état haute impédance est un état pour lequel la sortie de la porte est flottante. C'est-à-dire que la sortie se trouve comme déconnectée, donc ne pouvant véhiculer aucun signal électrique. Cet état haute impédance permet de raccorder plusieurs sorties sur un même fil formant un bus. Il faut cependant respecter le fait qu'à chaque fois, une seule sortie doit être à un niveau logique normal quand toutes les autres sont à l'état haute impédance (HZ).
Les tampons trois états sont utilisés quand on veut isoler une partie d'un circuit à un moment bien précis.

Il existe 4 symboles possibles :

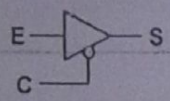


- Si C = 1 on a S = E : le buffer est dit ouvert ou transparent. Il est dans ce cas équivalent à un interrupteur fermé
- Si C = 0 on a S = HZ : le buffer est dit fermé ou verrouillé. Il est dans ce cas équivalent à un interrupteur ouvert.

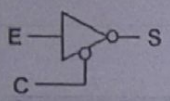
E : Entrée S : Sortie C : Commande



- Si C = 1 on a S = $\overline{E}$: le buffer est dit ouvert ou transparent.
- Si C = 0 on a S = HZ : le buffer est dit fermé ou verrouillé.



- Si C = 0 on a S = E : le buffer est dit ouvert ou transparent.
- Si C = 1 on a S = HZ : le buffer est dit fermé ou verrouillé.



- Si C = 0 on a S = $\overline{E}$: le buffer est dit ouvert ou transparent.
- Si C = 1 on a S = HZ : le buffer est dit fermé ou verrouillé.

II-8-2- Technologies internes des portes logiques de la technologie CMOS

Les portes logiques de la technologie CMOS couramment croisées sont celles de la famille CMOS classique ou série 4000. On distingue :

- **Le modèle à sortie non amplifiée**

Ce modèle est couramment appelé série 4000UB (UB : Unbuffered, non amplifiée).

- **Le modèle à sortie amplifiée**

Ce modèle est couramment appelé série 4000B (B : Buffered, amplifiée). Les portes de ce modèle sont beaucoup plus rapides que celles du modèle précédent.

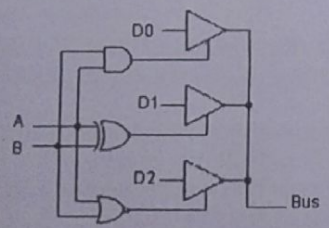
- **Le modèle à sortie trois états**

Son fonctionnement est identique à celui de la technologie TTL.

EXERCICES

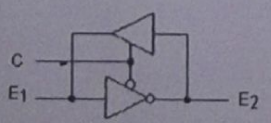
Exercice 1

Déterminer la donnée transmise par le bus pour les différentes combinaisons des entrées de commande A et B



Exercice 2

Montrer que le montage ci-dessous permet de réaliser une ligne bidirectionnelle.



Exercice 3

Le circuit intégré 7405 contient six portes inverseuses à sortie collecteur ouvert.
1/ Faire le schéma du câblage permettant de réaliser un ET câblé avec les six inverseuses.
2/ Déterminer l'expression de la sortie S. Quelle fonction logique obtient-on ?
3/ Calculer la valeur de la résistance aux collecteurs si la sortie est destinée à piloter 4 autres circuits TTL standards.

Scanned by CamScanner

CORRECTION

Exercice 1

A	B	Bus
0	0	D_2
0	1	D_1
1	0	D_1
1	1	D_0

Exercice 2

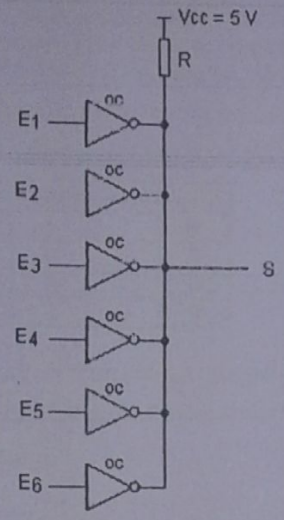
Si C = 0 on a le transfert du complément de E_1 vers E_2 : $E_2 = \overline{E_1}$

Si C = 1 on a le transfert de E_2 vers E_1 : $E_1 = E_2$

Ainsi selon l'état de C, les données circulent dans un sens ou dans l'autre. On a donc une ligne bidirectionnelle.

Exercice 3

1/



2/ $S = \overline{E_1}\,\overline{E_2}\,\overline{E_3}\,\overline{E_4}\,\overline{E_5}\,\overline{E_6} \Rightarrow S = \overline{E_1 + E_2 + E_3 + E_4 + E_5 + E_6}$

On obtient une porte nor à 6 entrées

Par ailleurs la sortie au niveau bas a une valeur

$0 \le V_{OL} \le V_{OLmax}$

Or $V_{OL} = Vcc - RIc$.

$\Rightarrow \quad 0 \le Vcc - RIc \le V_{OLmax}$

$\Rightarrow \quad Vcc - V_{OLmax} \le RIc \le Vcc$

$\Rightarrow \dfrac{V_{CC} - V_{OL\,max}}{I_C} \le R \le \dfrac{V_{CC}}{I_C}$

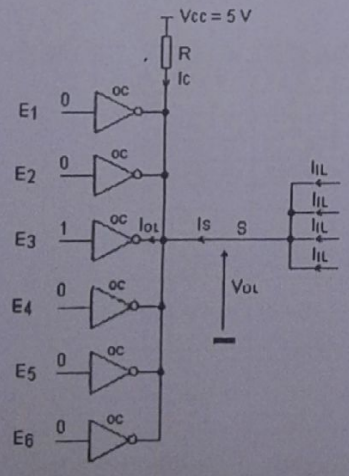
Or $V_{OLmax} = 0{,}4$ V

$\Rightarrow \dfrac{5 - 0{,}4}{9{,}6.10^{-3}} \le R \le \dfrac{5}{9{,}6.10^{-3}}$

$\Rightarrow 479\ \Omega \le R \le 520{,}8\ \Omega$

On choisira donc une résistance $R = 510\ \Omega$ dans la série E24.

3/



Si une seule sortie est au niveau bas elle doit être toute seule capable d'absorber la somme des courants Ic et Is.

On a donc $I_{OL} = Ic + Is \Rightarrow Ic = I_{OL} - Is$.

La sortie au niveau bas peut absorber un courant $I_{OLmax} = 16$ mA.

Les 4 entrées pilotées étant au niveau bas fournissent chacun un courant $I_{ILmax} = 1{,}6$ mA. Ce qui donne que $Is = 4 \times 1{,}6$ mA $= 6{,}4$ mA.

D'où $Ic = 16$ mA $- 6{,}4$ mA $= 9{,}6$ mA.

Scanned by CamScanner

CHAPITRE VI – CIRCUITS ARITHMETIQUES LOGIQUES

De nombreux systèmes numériques sont souvent amenés à effectuer des opérations arithmétiques (addition, soustraction, comparaison, multiplication,...) au cours des traitements. Ces opérations sont réalisées par des circuits arithmétiques logiques. Ces circuits sont à la base de la conception des calculateurs numériques. Grâce à leur vitesse de fonctionnement, ils permettent le traitement ultra rapide de calculs volumineux.

I- CIRCUITS LOGIQUES DE COMPARAISON DE DONNEES

La comparaison de deux informations binaires revient à déterminer si ces informations sont égales ou si l'une est inférieure ou supérieure à l'autre.

I-1- Fonction égalité

I-1-1- Principe

Deux mots binaires A et B codés avec n+1 éléments binaires chacun sont égaux si tous les éléments homologues (éléments de même position) sont égaux. C'est-à-dire si $A = a_n a_{n-1} a_{n-2} \ldots a_0$ et $B = b_n b_{n-1} b_{n-2} \ldots b_0$ alors A=B si et seulement si $(a_0=b_0)$ **et** $(a_1=b_1)$ **et** $(a_2=b_2)$ **et ... et** $(a_n=b_n)$.

I-1-2- Equation logique de la fonction égalité

a- Egalité élémentaire

Soient deux bits quelconques a_i et b_i.
La table de vérité de la fonction $(a_i=b_i)$ est :

a_i	b_i	$a_i=b_i$
0	0	1
0	1	0
1	0	0
1	1	1

Expression logique : $(a_i = b_i) = \overline{a_i \oplus b_i} = a_i \odot b_i$

b- Généralisation

Si $A = a_n a_{n-1} \ldots a_0$ et $B = b_n b_{n-1} \ldots b_0$, alors l'équation logique de la fonction (A=B) est : $(A=B) = (a_0 \odot b_0)(a_1 \odot b_1)(a_2 \odot b_2) \ldots (a_n \odot b_n)$.

I-2- Fonction supériorité

I-2-1- Principe

Si deux mots binaires A et B à n+1 bits chacun sont à comparer, alors A>B si et seulement si :
$(a_n>b_n)$
ou $(a_n=b_n$ **et** $a_{n-1}>b_{n-1})$
ou $(a_n=b_n$ **et** $a_{n-1}=b_{n-1}$ **et** $a_{n-2}>b_{n-2})$
⋮
ou $(a_n=b_n$ **et** $a_{n-1}=a_{n-1}$ **et** $a_{n-2}=b_{n-2}$ **et** ... **et** $a_1=b_1$ **et** $a_0>b_0)$

I-2-2- Equation logique de la fonction supériorité

a- supériorité élémentaire

Soient deux bits quelconques a_i et b_i.
La table de vérité de la fonction $(a_i>b_i)$ est :

a_i	b_i	$a_i>b_i$
0	0	0
0	1	0
1	0	1
1	1	0

Expression logique : $(a_i > b_i) = a_i \overline{b_i}$

b- Généralisation

Si $A = a_n a_{n-1} \ldots a_0$ et $B = b_n b_{n-1} \ldots b_0$, alors l'équation logique de la fonction (A>B) est :

$$(A>B) = a_n\overline{b_n} + (a_n \odot b_n)a_{n-1}\overline{b_{n-1}} + (a_n \odot b_n)(a_{n-1} \odot b_{n-1})a_{n-2}\overline{b_{n-2}} + \ldots + (a_n \odot b_n)(a_{n-1} \odot b_{n-1})(a_{n-2} \odot b_{n-2}) \ldots (a_2 \odot b_2)(a_1 \odot b_1)a_0\overline{b_0}$$

I-3- Fonction infériorité

I-3-1- Principe

Si deux mots binaires A et B à n+1 bits chacun sont à comparer, alors A<B si et seulement si :
$(a_n<b_n)$
ou $(a_n=b_n$ **et** $a_{n-1}<b_{n-1})$
ou $(a_n=b_n$ **et** $a_{n-1}=b_{n-1}$ **et** $a_{n-2}<b_{n-2})$
⋮
ou $(a_n=b_n$ **et** $a_{n-1}=a_{n-1}$ **et** $a_{n-2}=b_{n-2}$ **et** ... **et** $a_0<b_0)$

Scanned by CamScanner

I-3-2- Equation logique de la fonction infériorité

a- Infériorité élémentaire

Soient deux bits quelconques a_i et b_i.

La table de vérité de la fonction ($a_i<b_i$) est :

a_i	b_i	$a_i<b_i$
0	0	0
0	1	1
1	0	0
1	1	0

Expression logique : $(a_i < b_i) = \overline{a_i} b_i$

b- Généralisation

Si $A = a_n a_{n-1} \ldots a_0$ et $B = b_n b_{n-1} \ldots b_0$, alors l'équation logique de la fonction (A<B) est :

$$(A<B)= \overline{a_n} b_n + (a_n \odot b_n)\overline{a_{n-1}} b_{n-1} + (a_n \odot b_n)(a_{n-1} \odot b_{n-1})\overline{a_{n-2}} b_{n-2} + \ldots + (a_n \odot b_n)(a_{n-1} \odot b_{n-1})(a_{n-2} \odot b_{n-2}) \ldots (a_2 \odot b_2))(a_1 \odot b_1)\overline{a_0} b_0$$

I-4- Comparateur en circuits intégrés

On constate que la construction discrète d'un comparateur de plusieurs bits est complexe, vu le nombre d'opérateurs logiques à mettre en œuvre. Pour ce faire, on emploie généralement des circuits intégrés comparateurs. Ce sont des circuits qui appartiennent à la catégorie des circuits intégrés MSI (Medium Scale Intégration : *intégration à moyenne échelle*).

I-4-1- Logigramme d'un comparateur élémentaire (comparateur de deux mots d'un bit).

Table de vérité

a	b	(a=b)	(a>b)	(a<b)
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

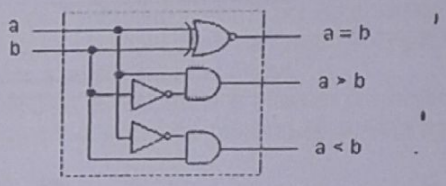
Equations logiques

$(a=b) = a \odot b$

$(a>b) = a\overline{b}$

$(a<b) = \overline{a} b$

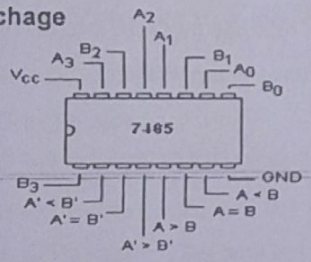
Logigramme



I-4-2- Utilisation d'un circuit intégré comparateur : le 7485

a- Description

Brochage



Symbole logique

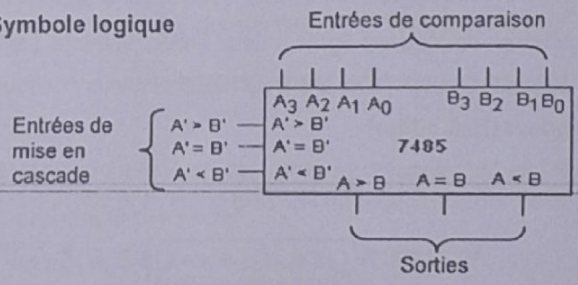
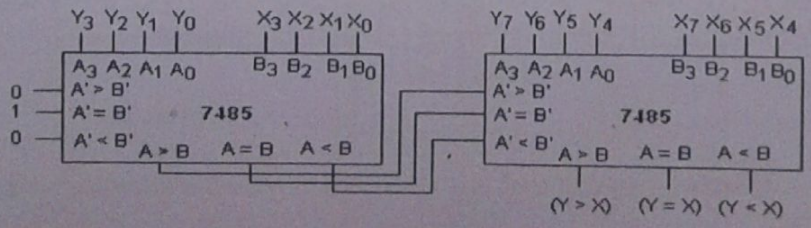


Table de vérité

Entrées de comparaison				Entrées de mise en cascade			sorties		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	A'> B'	A'= B'	A'< B'	A > B	A = B	A< B
$A_3 > B_3$	X	X	X	X	X	X	1	0	0
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X			
	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X			
	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X			
$A_3 < B_3$	X	X	X	X	X	X	0	0	1
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X			
	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X			
	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X			
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	1	0	0	1	0	0
				0	0	1	0	0	1
				0	1	0	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	X	1	X	0	1	0
				1	0	1	0	0	0
				0	0	0	1	0	0

b- Comparaison de deux mots binaires de 8 bits avec des circuits 7485

Soit à comparer Y à X tels que $X=X_7X_6X_5X_4X_3X_2X_1X_0$ et $Y= Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$

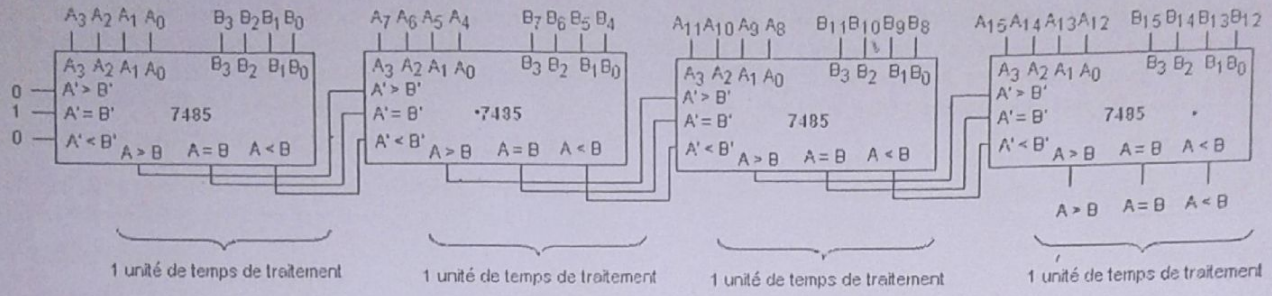


Scanned by CamScanner

c- Comparaison de deux mots binaires de 16 bits avec des circuits intégrés 7485

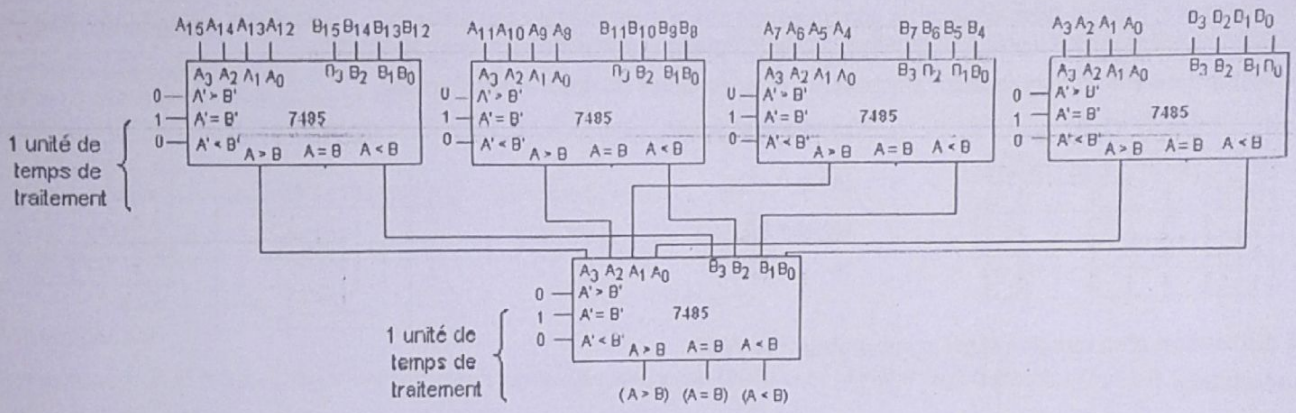
Soit à comparer $A = A_{15}A_{14}A_{13}....A_3A_2A_1A_0$ et $B = B_{15}B_{14}B_{13}....B_3B_2B_1B_0$

- **Comparaison lente**



A cause de la mise en cascade des circuits, le temps de réponse à la sortie du montage est obtenu en faisant la somme du temps de réponse de chaque circuit ; ce qui donne 4 unités de temps de traitement. On peut adopter une autre configuration qui bien que faisant intervenir un nombre élevé de circuits intégrés permet d'obtenir un temps de réponse total beaucoup plus faible.

- **Comparaison rapide**



Désormais, on a un temps de réponse égale à deux unités de temps (deux fois le temps de réponse du circuit intégré 7485).

II- CIRCUITS LOGIQUES D'ADDITION DE DONNEES

II-1- Demi additionneur (Half adder)

Soient a_i et b_i deux bits homologues de rang i de deux mots binaires donnés. La somme a_i+b_i (lire a_i **plus** b_i ; car il s'agit d'une somme arithmétique et non d'une somme logique) observe le principe suivant :

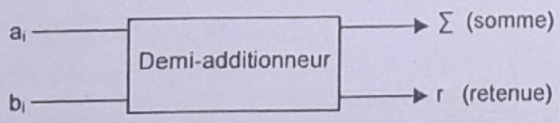


Table de vérité

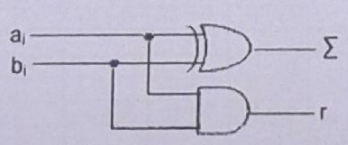
a_i	b_i	Σ	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Equations logiques

$\Sigma = \bar{a_i}b_i + a_i\bar{b_i} = a_i \oplus b_i$

$r = a_i b_i$

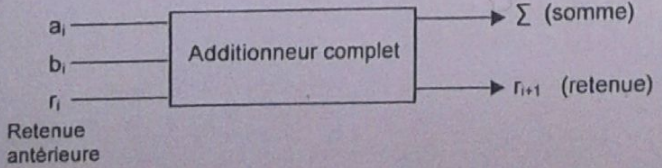
Logigramme



L'addition ainsi réalisée ne tient pas compte d'une éventuelle retenue pouvant provenir d'une précédente addition ; d'où le nom de demi-additionneur ou semi-additionneur. Pour réaliser un additionneur complet, il faut donc tenir compte non seulement des variables d'entrée a_i et b_i mais aussi de la retenue éventuelle pouvant provenir d'une addition précédente. On obtient ainsi un additionneur complet.

II-2- Additionneur complet (Full adder)

Schéma de principe



Scanned by CamScanner

Table de vérité

a_i	b_i	r_i	$\sum_i$	r_{i+1}
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

Equations logiques

$$\sum_i = \bar{a}_i\bar{b}_i r_i + \bar{a}_i b_i \bar{r}_i + a_i\bar{b}_i\bar{r}_i + a_i b_i r_i$$

$$\sum_i = r_i(\bar{a}_i\bar{b}_i + a_i b_i) + \bar{r}_i(\bar{a}_i b_i + a_i\bar{b}_i)$$

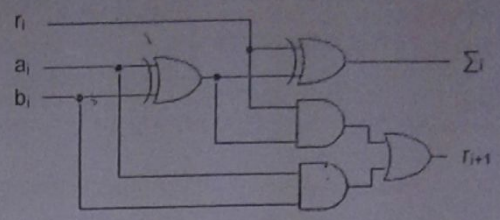
$$\sum_i = r_i \oplus a_i \oplus b_i$$

$$r_{i+1} = \bar{a}_i b_i r_i + a_i\bar{b}_i r_i + a_i b_i\bar{r}_i + a_i b_i r_i$$

$$r_{i+1} = r_i(\bar{a}_i b_i + a_i\bar{b}_i) + a_i b_i(\bar{r}_i + r_i)$$

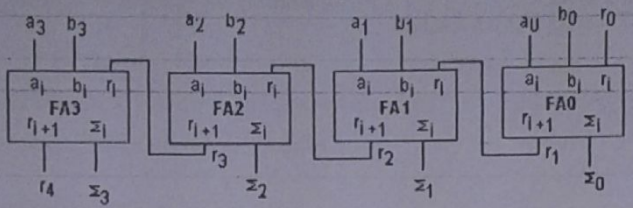
$$r_{i+1} = r_i(a_i \oplus b_i) + a_i b_i$$

Logigramme



II-3- Addition de deux variables binaires de quatre bits

II-3-1- Utilisation d'un additionneur à retenue propagée



Le résultat de l'opération ne sera juste qu'après que l'additionneur complet FA_0 aura fini son opération, suivi de FA_1 qui a besoin de la retenue issue de FA_0, puis de FA_2 qui doit recevoir la retenue de FA_1 et enfin de FA_3 qui doit recevoir la retenue de FA_2. D'où le nom d'additionneur à retenue propagée.

Ainsi, si le temps de réponse de chaque circuit additionneur est tr, alors le résultat de l'opération n'est exact qu'après un temps égale à 4 tr. Tout se passe comme si l'addition se fait de manière sérielle. Les délais de réalisation de l'opération pour un tel additionneur peuvent devenir assez élevé lorsque le nombre de bits de chaque variable à additionner est grand.

Si nous effectuons par exemple l'addition de deux mots binaires de n bits chacun avec un additionneur à retenue propagée, il faut attendre un temps t = ntr avant d'obtenir un résultat exact. Pour ce faire, si l'opération est déclenchée par un signal d'horloge alors ce signal doit avoir une période $T_H \geq ntr$.

II-3-2- Utilisation d'un additionneur à retenue anticipée

Puisque l'équation générale de la retenue est connue ($r_{i+1} = r_i(a_i \oplus b_i) + a_i b_i$), alors il est possible d'exprimer l'équation de toute retenue r_{i+1} en fonction de tous les bits a_i et b_i antérieurs et de r_0. Ce qui permet de réduire considérablement les délais de traitement.

Par exemple pour un additionneur quatre bits, on a les équations suivantes des retenues :

$r_1 = r_0(a_0 \oplus b_0) + a_0 b_0$

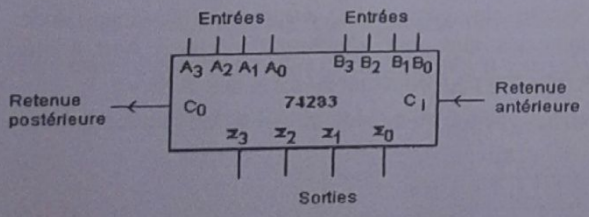
$r_2 = r_1(a_1 \oplus b_1) + a_1 b_1 = (r_0(a_0 \oplus b_0) + a_0 b_0)(a_1 \oplus b_1) + a_1 b_1$

$r_3 = r_2(a_2 \oplus b_2) + a_2 b_2 = ((r_0(a_0 \oplus b_0) + a_0 b_0)(a_1 \oplus b_1) + a_1 b_1)(a_2 \oplus b_2) + a_2 b_2$

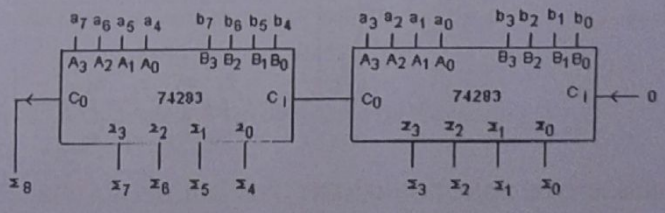
$r_4 = r_3(a_3 \oplus b_3) + a_3 b_3 = (((r_0(a_0 \oplus b_0) + a_0 b_0)(a_1 \oplus b_1) + a_1 b_1)(a_2 \oplus b_2) + a_2 b_2)(a_3 \oplus b_3) + a_3 b_3$

Un tel circuit appelé additionneur à retenue anticipée est très généralement disponible en circuit intégré de la catégorie MSI (Medium Scale Integration). C'est par exemple le cas du circuit intégré **74283 : additionneur quatre bits.**

Son symbole logique :



Exemple de réalisation d'un additionneur 8 bits avec des 74283



II-4- Réalisation d'un additionneur- soustraction

II-4-1- Principe de la soustraction binaire

a- Demi-soustraction

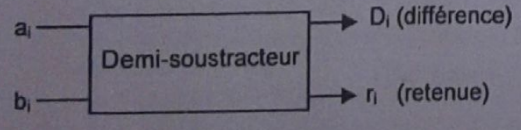


Table de vérité

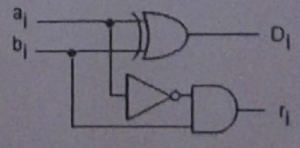
a_i	b_i	D_i	r_i
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Equations logiques

$D_i = a_i \oplus b_i$

$r_i = \bar{a}_i b_i$

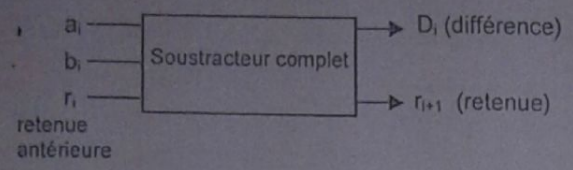
Logigramme



Ce circuit ne tient pas compte d'une éventuelle retenue antérieure. Cette défaillance est corrigée par le soustracteur complet.

Scanned by CamScanner

b- Soustracteur complet



Telle que mathématiquement
$D_i = a_i - (b_i + r_i)$
et r_{i+1}, le report sur le résultat

Table de vérité

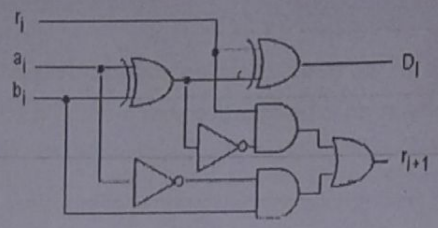
a_i	b_i	r_i	D_i	r_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Équations logiques

$$D_i = \bar{a}_i\bar{b}_i r_i + \bar{a}_i b_i \bar{r}_i + a_i\bar{b}_i\bar{r}_i + a_i b_i r_i$$
$$D_i = r_i(\bar{a}_i\bar{b}_i + a_i b_i) + \bar{r}_i(\bar{a}_i b_i + a_i\bar{b}_i)$$
$$D_i = r_i \oplus a_i \oplus b_i$$

$$r_{i+1} = \bar{a}_i\bar{b}_i r_i + \bar{a}_i b_i \bar{r}_i + \bar{a}_i b_i r_i + a_i b_i r_i$$
$$r_{i+1} = r_i(\bar{a}_i\bar{b}_i + a_i b_i) + \bar{a}_i b_i(\bar{r}_i + r_i)$$
$$r_{i+1} = r_i\overline{(a_i \oplus b_i)} + \bar{a}_i b_i$$

Logigramme



Remarque :
La comparaison des équations des sorties de l'additionneur complet et du soustracteur complet montre que la différence porte seulement sur les équations des sorties de retenue. Ce qui permet d'utiliser un circuit additionneur en même temps comme un soustracteur. En effet, une opération de soustraction peut être ramenée à une addition à l'aide du **complément à 2**. Pour soustraire une variable binaire $B = b_3b_2b_1b_0$ d'une variable binaire $A = a_3a_2a_1a_0$, on effectue la somme de $A = a_3a_2a_1a_0$ et du complément à 2 de B (la formation du complément à 2 est donnée par $\bar{b}_3\bar{b}_2\bar{b}_1\bar{b}_0 + 1$).

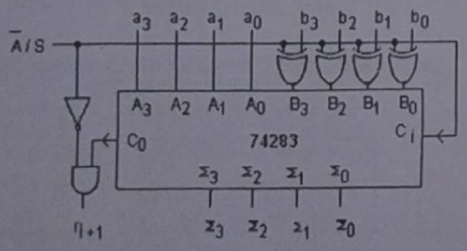
Exemple : soit à effectuer l'opération 7 – 5 ; les nombres étant codés sur 4 bits y compris le signe.
A = 0111 et B = 0101 ⇒ en complément à deux – B = 1010+1= 1011.

$$\begin{array}{r} 0\ 111 \\ +\ 1\ 011 \\ \hline = (1)\ 0\ 010 \end{array}$$

(1) ← Bit de report négligé

Supposons que nous devons exploiter le circuit intégré 74283 comme additionneur-soustracteur.
Le complément à 2 de B nécessite l'addition de 1. Cela peut être bien réalisé si on porte l'entrée de retenue antérieure C_i à 1. Ainsi pour une addition on doit imposer $C_i = 0$ et pour une soustraction $C_i = 1$.
Par ailleurs dans le cas d'une addition : A + B, les bits b_i de B arrivent directement sur l'additionneur. C'est-à-dire si $C_i = 0$ alors on doit disposer directement sur l'additionneur, des bits b_i du nombre B. Mais dans le cas d'une soustraction : A – B les bits b_i de B arrivent complémentés sur l'additionneur. C'est-à-dire si $C_i = 1$ alors on a les $\bar{b}_i$. Aussi faut-il remarquer que dans le cas de la soustraction la retenue sur les bits de poids fort est négligée (c'est-à-dire que C_0 doit fournir une valeur égale à 0).

D'où le schéma suivant :



L'entrée de commande nommée $\bar{A}/S$ indique donc clairement que pour effectuer l'opération A+B il faut $\bar{A}/S = 0$ et pour effectuer A – B il faut $\bar{A}/S = 1$.

III- PRINCIPE DE FONCTIONNEMENT D'UN MULTIPLICATEUR BINAIRE

La multiplication binaire se résume en une succession d'addition et de décalage de bits. Ainsi un circuit multiplicateur est essentiellement formé d'additionneurs et de registre à décalages.

Soit à multiplier 9 par 13 c'est-à-dire en binaire 1001 par 1101.
L'opération se déroule manuellement comme suit :

$$\begin{array}{r} 1001 \\ \times\ 1101 \\ \hline 1001 \\ 0000 \\ 1001 \\ 1001 \\ \hline = 1110101 \end{array}$$

1001 ← Multiplicande (MD)
x 1101 ← Multiplicateur (MT)
Produits partiels
= 1110101 ← Produit (P)

Scanned by CamScanner

Pour obtenir le produit (P) nous sommes amené à faire la somme des quatre produits partiels. Les circuits électroniques ne pouvant faire que l'addition de deux nombres à la fois, il s'agira si nous devons utiliser de tels circuits, de faire plusieurs additions deux par deux. Il va s'en dire que nous disposions alors d'un registre destiné à stocker les sommes partielles et en définitive le résultat final c'est-à-dire le produit (P). Pour le rôle de stockage d'informations que ce registre joue, nous le nommons accumulateur (AC). Par ailleurs deux autres registres sont utiles pour contenir au départ le multiplicande (MD) et le multiplicateur (MT). Le registre du multiplicande sera nommé RMD et celui du multiplicateur RMT.
Si le circuit multiplicateur en est un qui effectue le produit de deux nombres binaires de 4 bits, alors l'accumulateur (AC) doit être un registre d'au moins 8 bits car l'opération maximale effectuable est en décimale 15 x 15 = 225. On a $(225)_{10} = (11100001)_2$.
Le registre RMT du multiplicateur sera un registre de 4 bits.
Quant au registre du multiplicande RMD, il sera un de 8 bits car son contenu doit être décalé 4 fois.

Le principe des différentes étapes de fonctionnement du circuit multiplicateur est le suivant :

Au départ :
- Charger le registre RMD avec la valeur de MD : (RMD) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
- Charger le registre RMT avec la valeur de MT : (RMT) | 1 | 1 | 0 | 1 |
- Initialiser le registre AC (remise à zéro) : (AC) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Etape 1 : Le 1er bit de MT (LSB) = 1, alors :
- Additionner AC à RMD puis stocker dans AC : (AC) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
- Décaler RMD d'un rang à gauche : (RMD) | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

Etape 2 : Le 2e bit de MT = 0, alors :
- Ne pas changer AC : (AC) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
- Décaler RMD d'un rang à gauche : (RMD) | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Etape 3 : Le 3e bit de MT = 1, alors :
- Additionner AC à RMD puis stocker dans AC : (AC) | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
- Décaler RMD d'un rang à gauche : (RMD) | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Etape 4 : Le 4e bit de MT (MSB) = 1, alors :
- Additionner AC à RMD puis stocker dans AC : (AC) | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
- Opération terminée car examen du MSB : **Produit (P) 0 1 1 1 0 1 0 1**

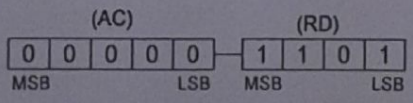
IV- PRINCIPE DE FONCTIONNEMENT D'UN DIVISEUR BINAIRE

Dans les circuits logiques de division binaire, les calculs de division sont effectués par une succession d'additions, de soustractions et de décalages de bits.

Soit à diviser 13 par 11 c'est-à-dire en binaire 1101 par 1011.
Le processus de l'opération amène à définir :

- Un registre Rd qui contient le diviseur : (Rd) | 1 | 0 | 1 | 1 |
- Un registre RD qui contient le dividende : (RD) | 1 | 1 | 0 | 1 |
- Un accumulateur AC qui reçoit des opérations partielles. Le registre AC doit pouvoir contenir un bit en plus par rapport au registre dividende RD : (AC) | 0 | 0 | 0 | 0 | 0 |

Pour effectuer l'opération de division on fait une concaténation de l'accumulateur AC et du registre du dividende RD :



Le processus de la division est le suivant :

- Au départ charger Rd et RD, initialiser AC.
- Décaler l'ensemble (AC↔RD) d'un rang vers la gauche.
- Soustraire Rd de AC (on obtient un reste partiel) et mettre le résultat dans AC.
- Vérifier le MSB de AC.
  - *Si le MSB de AC =1, alors additionner seulement AC et Rd puis mettre le résultat dans AC et conserver la valeur de RD.*
  - *Si le MSB de AC =0, alors conserver AC et changer le LSB de RD à 1.*
- Répéter ce processus autant de fois qu'il y a de bits du dividende.
- Enfin de processus, le contenu de AC est le reste et le contenu de RD est le quotient.

Maintenant effectuons la division de l'exemple.

Scanned by CamScanner

Au départ :

Charger Rd et RD, initialiser AC

(AC) 0 0 0 0 0 — 1 1 0 1 (RD)

1 0 1 1 (Rd)

Etape 1 :

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 0 0 1 — 1 0 1 0 (RD)

\- 1 0 1 1 (Rd)

(AC) 1 0 1 1 0 — 1 0 1 0 (RD)

- Vérifier le MSB de AC : Il vaut 1 ; donc additionner : AC + Rd, maintenir RD

\+ 1 0 1 1 (Rd)

(AC) 0 0 0 0 1 — 1 0 1 0 (RD)

Etape 2 :

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 0 1 1 — 0 1 0 0 (RD)

\- 1 0 1 1 (Rd)

(AC) 1 1 0 0 0 — 0 1 0 0 (RD)

- Vérifier le MSB de AC : Il vaut 1 ; donc additionner : AC + Rd, maintenir RD

\+ 1 0 1 1 (Rd)

(AC) 0 0 0 1 1 — 0 1 0 0 (RD)

Etape 3 :

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 1 1 0 — 1 0 0 0 (RD)

\- 1 0 1 1 (Rd)

(AC) 1 1 0 1 1 — 1 0 0 0 (RD)

- Vérifier le MSB de AC : Il vaut 1 ; donc additionner : AC + Rd, maintenir RD

\+ 1 0 1 1 (Rd)

(AC) 0 0 1 1 0 — 1 0 0 0 (RD)

Etape 4 : (dernière étape car tout le contenu de RD a été décalé)

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 1 1 0 1 — 0 0 0 0 (RD)

\- 1 0 1 1 (Rd)

(AC) 0 0 0 1 0 — 0 0 0 0 (RD)

- Vérifier le MSB de AC : Il vaut 0 ; donc conserver AC et changer le LSB de RD à 1

(AC) 0 0 0 1 0 — 0 0 0 1 (RD)

Reste **Quotient**

Résultat : **quotient = 1 reste = 10**

Remarque :

La division de deux nombres tout comme la multiplication de deux nombres, porte essentiellement sur les valeurs absolues de ces nombres. En effet, le signe du résultat peut être traité à part.

Soient : -S_D le signe du dividende et S_d le signe du diviseur

-S_M le signe du multiplicande et S_m le signe du multiplicateur

-S_R le signe du Résultat de la division ou de la multiplication

S_R est donné par la table de vérité suivante :

S_D ou S_M	S_d ou S_m	S_R
0	0	0
0	1	1
1	0	1
1	1	0

Il ressort que :

$S_R = S_D \oplus S_d$ (pour la division) ou $S_R = S_M \oplus S_m$ (pour la multiplication).

Avec 0 ≡ + (plus) et 1 ≡ – (moins)

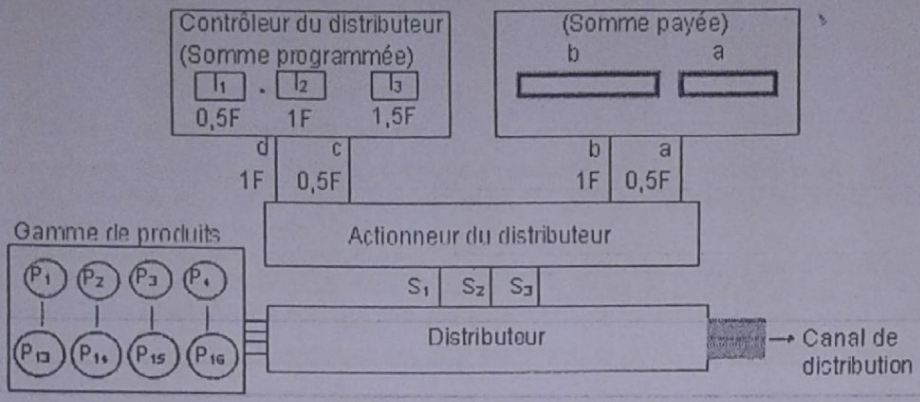
Les circuits pouvant effectuer des divisions sont des circuits complexes destinés à réaliser toutes les opérations arithmétiques simples ou complexes et même d'autres types d'opérations telles que les opérations logiques (Ou, Et, Non, Non Ou, Non Et, ect.). Ce type de circuit peut traiter des nombres exprimés avec un nombre élevé de bits. Ils sont appelés UAL (Unité Arithmétique et Logique).

Scanned by CamScanner

EXERCICES

Exercice 1

Le système à étudier permet de faire des achats automatiques dans un magasin.



Les prix des produits disponibles varient entre 1,5 F ; 1F et 0,5 F.
Les boutons poussoirs I_3, I_2, I_1 du contrôleur de distribution correspondent respectivement à 1,5 F ; 1F et 0,5 F.
La sortie d du contrôleur compte pour 1F et la sortie c compte pour 0,5 F.
Si on actionne par exemple I_3 alors d = 1 et c = 1 (car 1,5 F = 1F + 0,5 F).
Le fonctionnement du contrôleur est tel que si plusieurs boutons poussoirs sont actionnés, c'est seulement l'information de sortie du bouton dont la somme est plus grande qui est prise en compte.
Le client pour payer, doit introduire dans les fentes a et b les pièces dont la somme correspond au prix du produit. Lorsqu'une pièce est introduite, la sortie correspondante de la fente est à 1 sinon elle est à 0.

Le fonctionnement de l'actionneur du distributeur est le suivant :
- Si la somme payée est inférieure à la somme programmée, $S_1 = 1$
- Si la somme payée est identique à la somme programmée, $S_2 = 1$
- Si la somme payée est supérieure à la somme programmée, $S_3 = 1$

Il y a livraison du produit lorsque le client procède comme suit :
- Choisir dans la gamme de produits, le produit désiré en actionnant le poussoir Pi correspondant.
- Actionner l'un des poussoirs du contrôleur correspondant au prix du produit choisi.
- Payer en introduisant dans les fentes les pièces nécessaires. Dans ce cas si $S_2 = 1$, il y a livraison du produit.

Travail à faire :
1/ Construire le logigramme du contrôleur de distribution uniquement à l'aide de portes logiques NAND à deux entrées.
2/ Construire le logigramme de l'actionneur de distribution à l'aide de portes logiques. Faire un câblage à l'aide d'un comparateur (le 7485) pour obtenir le même fonctionnement. S'il nécessite de portes logiques, utiliser des portes NAND à deux entrées.

Exercice 2

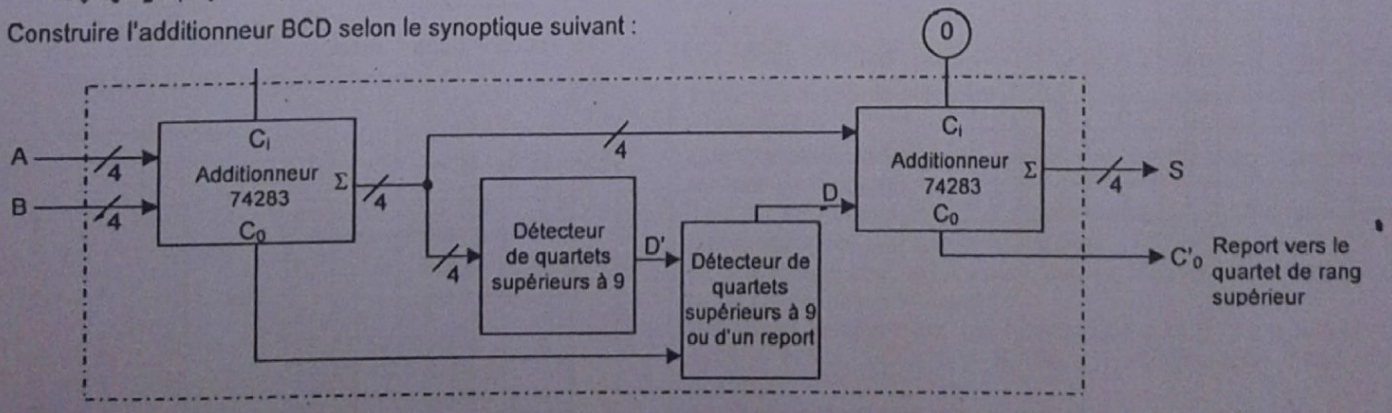
Réalisation d'un additionneur BCD.

Un additionneur BCD doit être en mesure de :
- Additionner deux nombres donnés dans le code BCD en additionnant deux à deux les quartets homologues.
- Reconnaître si la somme de deux quartets est supérieure à 1001 (c'est-à-dire 9) ou si cette somme a généré un report pour le quartet immédiatement supérieur, afin dans ces deux cas, d'y ajouter la valeur 0110 (c'est-à-dire 6) : cela s'appelle correction.

Soient deux nombres BCD, $A = a_3a_2a_1a_0$ et $B = b_3b_2b_1b_0$ avec $\Sigma = \Sigma_3\Sigma_2\Sigma_1\Sigma_0$ le résultat de A + B sans le report éventuel.

On a : $a_3\ a_2\ a_1\ a_0$ ⟵ nombre BCD
$+\ b_3\ b_2\ b_1\ b_0$ ⟵ nombre BCD
$= C_0\ \Sigma_3\ \Sigma_2\ \Sigma_1\ \Sigma_0$ ⟵ résultat net où C_0 est le report éventuel sur la somme de a_3 et b_3.

Construire l'additionneur BCD selon le synoptique suivant :

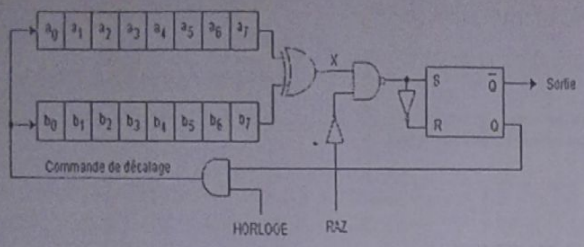


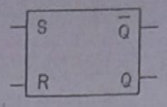
NB : ⟋4 : un ensemble de 4 liaisons.

Scanned by CamScanner

Exercice 3

On donne $A = a_7a_6a_5a_4a_3a_2a_1a_0$ et $B = b_7b_6b_5b_4\ b_3b_2b_1b_0$ inscrits dans deux registres à décalage.
Montrer que le montage ci-dessous permet d'identifier si $A \neq B$ (A différent de B).



NB : le circuit  est appelé bascule.

Son fonctionnement est tel que :

Si S = 0 et R =1 alors Q = 0 et $\overline{Q}$ =1

Si S = 1 et R = 0 alors Q = 1 et $\overline{Q}$ = 0

Exercice 4

1/ Donner les étapes de la multiplication 13 par 11 par la méthode étudiée.
2/ Donner les étapes de la division 13 par 5 par la méthode étudiée.

CORRECTION

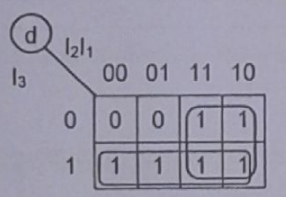
Exercice 1

1/ Etude du contrôleur de distribution

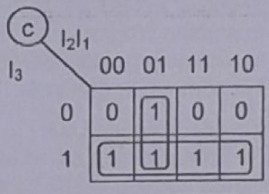
Table de vérité

I_3 (1,5F)	I_2 (1F)	I_1 (0,5F)	d (1F)	c (0,5F)
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Expressions Logiques

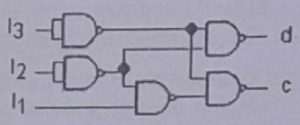


$d = I_3 + I_2 = \overline{\overline{I_3} + \overline{I_2}} = \overline{\overline{I_3}}\,\overline{\overline{I_2}}$



$c = I_3 + \overline{I}_2I_1 = \overline{\overline{I_3} + \overline{I}_2I_1} = \overline{\overline{I_3}}\,\overline{\overline{I}_2I_1}$

Logigramme en NAND

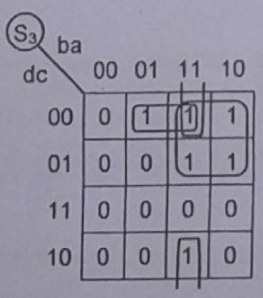


2/ Etude de l'actionneur de distribution

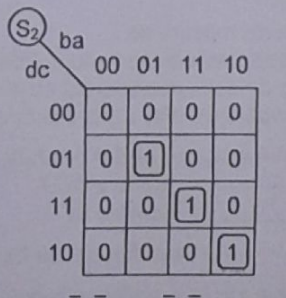
Table de vérité

d (1F)	c (0,5F)	b (1F)	a (0,5F)	S_3 (>)	S_2 (<)	S_0 (=)
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

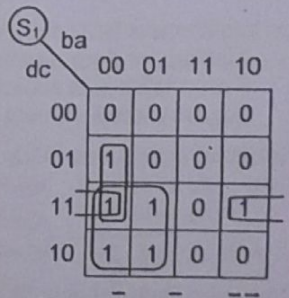
Expressions Logiques des sorties



$S_3 = \overline{d}b + \overline{d}\,\overline{c}a + \overline{c}ba$

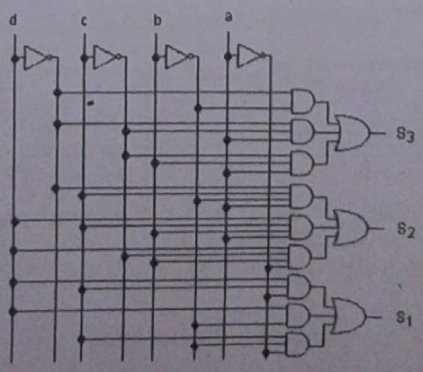


$S_2 = \overline{d}c\overline{b}a + d\overline{c}b\overline{a} + dcba$



$S_1 = dc\overline{a} + d\overline{b} + c\overline{b}\,\overline{a}$

Logigramme



Scanned by CamScanner

Construction autour du circuit 7485 :

Pour l'actionneur de distribution, il y a comparaison des informations binaires ba aux informations binaires dc. Donc on portera sur les entrées du comparateur, dc = B_1B_0 en maintenant B_3B_2 = 00 et ba = A_1A_0 en maintenant A_3A_2 = 00. Ce qui donne que la récupération des sorties du circuit revient à S_3 = (A>B), S_2 = (A<B), S_1 = (A=B).
Il ne peut y avoir une livraison que si **S_2 = 1 et S_1 = S_3 = 0** c'est-à-dire si le couplet dc est égal au couplet ba.
En effet, Il y a égalité des bits homologues si :
- dc = ba = 11 ou
- dc = ba = 10 ou
- dc = ba = 01 ou
- dc = ba = 00 (état de repos des entrées).

Dans ce dernier cas l'actionneur devra être inactif c'est-à-dire que S_2 doit être égale à 0 afin qu'il n y ait pas de livraison. De même **S_1 et S_3 doivent être égales à 0** pour respecter les mêmes conditions de fonctionnement du dispositif construit par le logigramme (au repos S_3 =S_2 = S_1 = 0).
D'après la table de vérité du 7485, on déduit que :
- Si dc = ba = 00 alors pour les entrées de mise en cascade il faut **(A'>B') = (A'<B') = 1** et **(A'=B') = 0** afin d'avoir S_3 = S_2 = S_1 = 0.
- Si dc = ba ≠ 00 alors il faut **(A'=B') = 1, quelles que soient ce que valent (A'>B') et (A'<B')** afin d'avoir S_2 = 1 et S_3 = S_1 = 0.

Il ressort que ces deux conditions sont réalisées si et seulement si **(A'>B') = (A'<B') = 1** et que l'entrée **(A'=B')** vérifie l'une des tables de vérité suivantes :

d	c	(A'=B')
0	0	0
0	1	1
1	0	1
1	1	1

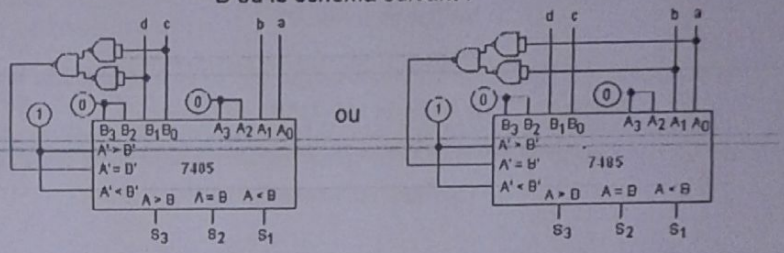
$\Rightarrow (A'=B') = d + c = \overline{d+c} = \overline{dc}$

ou

b	a	(A'=B')
0	0	0
0	1	1
1	0	1
1	1	1

$\Rightarrow (A'=B') = b + a = \overline{b+a} = \overline{ba}$

D'où le schéma suivant :



Exercice 2

La sortie D' permettant de détecter uniquement les quartets supérieurs à 9, on a :

Table de vérité

Valeur	$\sum_3$	$\sum_2$	$\sum_1$	$\sum_0$	D'
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Expression de la sortie D'

D'	$\sum_1\sum_0$ 00	01	11	10
$\sum_3\sum_2$ 00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

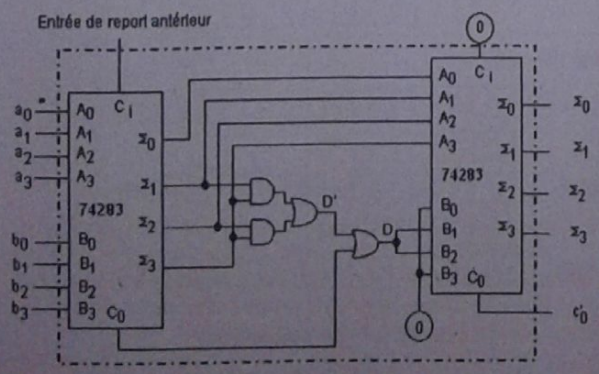
$D' = \sum_3\sum_2 + \sum_3\sum_1$

La sortie de D permettant de détecter D' ou C_0

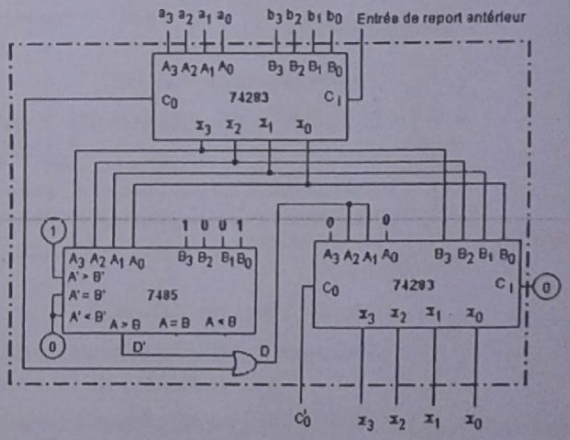
$\Rightarrow D = D' + C_0$

$\Rightarrow D = \sum_3\sum_2 + \sum_3\sum_1 + C_0$

D'où le schéma logique ci-dessous :



Un autre câblage peut être obtenu par l'approche suivante :



Additionner A=$a_3a_2a_1a_0$ et B=$b_3b_2b_1b_0$. S'il y a report sur le résultat alors on ajoute 6 (c'est-à-dire 0110) au résultat. D'autre part si ce résultat est supérieur à 9 (c'est-à-dire 1001), alors on ajoute également 6 au résultat.

Exercice 3

Au début, on peut initialiser la sortie en envoyant une brève impulsion montante sur l'entrée RAZ. Dans ce cas **Q = 1** et $\overline{Q}$ = 0 **⇒ Sortie = 0**
Q = 1, alors le décalage du contenu des registres est possible en fonction de l'horloge.
Tout en maintenant RAZ = 0 (après initialisation), tant que deux bits homologues a_i et b_i sont égaux alors X = 0 ⇒ **Q = 1 et Sortie = 0** : le décalage peut se faire pour mettre à disposition les bits a_{i-1} et b_{i-1} et ce, jusqu'aux bits a_0 et b_0
Par contre lorsque deux bits homologues a_i et b_i sont différents alors X = 1 ⇒ Q = 0 et **Sortie = 1** : le décalage n'est plus possible.
Ainsi dès que **Sortie = 1** on conclut que **A ≠ B**, sinon après 8 impulsions d'horloge si **Sortie = 0**, on conclut que A = B. Le montage est donc un comparateur série.

Scanned by CamScanner

Exercice 4

1/ multiplication de 13 par 11 c'est-à-dire de 1101 par 1011.

Au départ : - Charger le registre RMD avec la valeur de MD : (RMD) 0 0 0 0 1 1 0 1

- Charger le registre RMT avec la valeur de MT : (RMT) 1 0 1 1

- Initialiser le registre AC (remise à zéro) : (AC) 0 0 0 0 0 0 0 0

Etape 1 : Le 1er bit de MT (LSB) = 1, alors :

- Additionner AC à RMD puis stocker dans AC : (AC) 0 0 0 0 1 1 0 1

- Décaler RMD d'un rang à gauche : (RMD) 0 0 1 1 0 1 0 0

Etape 2 : Le 2e bit de MT = 1, alors :

- Additionner AC à RMD puis stocker dans AC : (AC) 0 0 1 0 0 1 1 1

- Décaler RMD d'un rang à gauche : (RMD) 0 1 0 0 1 0 0 0

Etape 3 : Le 3e bit de MT = 0, alors :

- Ne pas changer AC : (AC) 0 0 1 0 0 1 1 1

- Décaler RMD d'un rang à gauche : (RMD) 0 1 1 0 1 0 0 0

Etape 4 : Le 4e bit de MT (MSB) = 1, alors :

- Additionner AC à RMD puis stocker dans AC : (AC) 1 0 0 0 1 1 1 1

- Opération terminée car examen du MSB : **Produit (P) 1 0 0 0 1 1 1 1**

2/ Division de 13 par 5 c'est-à-dire de 1101 par 101.

Au départ :

Charger Rd et RD, initialiser AC

(AC) 0 0 0 0 0 — 1 1 0 1 (RD)

0 1 0 1 (Rd)

Etape 1 :

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 0 0 1 — 1 0 1 0 (RD)

\- 0 1 0 1 (Rd)

(AC) 1 1 1 0 0 — 1 0 1 0 (RD)

- Vérifier le MSB de AC : Il vaut 1 ; donc additionner : AC + Rd, maintenir RD

\+ 0 1 0 1 (Rd)

(AC) 0 0 0 0 1 — 1 0 1 0 (RD)

Etape 2 :

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 0 1 1 — 0 1 0 0 (RD)

\- 1 0 1 1 (Rd)

(AC) 1 1 1 1 0 — 0 1 0 0 (RD)

- Vérifier le MSB de AC : Il vaut 1 ; donc additionner : AC + Rd, maintenir RD

\+ 0 1 0 1 (Rd)

(AC) 0 0 0 1 1 — 0 1 0 0 (RD)

Etape 3 :

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 1 1 0 — 1 0 0 0 (RD)

\- 0 1 0 1 (Rd)

(AC) 0 0 0 0 1 — 1 0 0 0 (RD)

- Vérifier le MSB de AC : Il vaut 0 ; donc conserver AC et changer le LSB de RD à 1

(AC) 0 0 0 0 1 — 1 0 0 1 (RD)

Etape 4 : (dernière étape car tout le contenu de RD à été décalé)

- Décaler à gauche (AC↔RD)
- Soustraire : AC – Rd et mettre le résultat dans AC

(AC) 0 0 0 1 1 — 0 0 1 0 (RD)

\- 0 1 0 1 (Rd)

(AC) 1 1 1 1 0 — 0 0 1 0 (RD)

- Vérifier le MSB de AC : Il vaut 1 ; donc additionner : AC + Rd, maintenir RD

\+ 0 1 0 1 (Rd)

(AC) 0 0 0 1 1 — 0 0 1 0 (RD)

Reste Quotient

Résultat : quotient = 10 reste = 11

Scanned by CamScanner

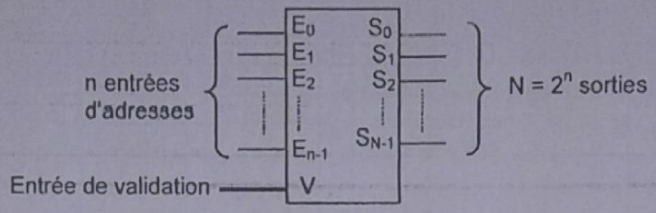
CHAPITRE VII – CIRCUITS LOGIQUES DE TRANSFORMATION DE CODE ET D'AIGUILLAGE DE DONNEES

I- CIRCUITS LOGIQUES DE TRANSFORMATION DE CODE

I-1- Décodeurs

I-1-1- Définition - Présentation

Un décodeur est un dispositif de logique combinatoire qui a **n entrées** (nommées **entrées d'adresses**) et généralement **$N=2^n$ sorties.** Son rôle est de faire correspondre à chaque valeur du code binaire pur admis sur les **n entrées, une seule sortie** parmi les **2^n sorties,** tel que le numéro de la sortie active ou (valide) est la valeur décimale (en base 10) du code binaire présent sur les entrées.



DECODEUR 1 parmi N ou DECODEUR n vers N

Le décodeur est mis en état de marche lorsque son boîtier est sélectionné c'est-à-dire lorsqu'on envoie le niveau logique adéquat sur l'entrée de validation V (ou de sélection ou enable) du circuit.
Lorsque le boîtier n'est pas sélectionné, les sorties sont toutes au même état logique : soit à l'état logique 0 soit à l'état logique 1, soit à l'état haute impédance, selon les caractéristiques de chaque circuit, spécifiées par le constructeur. Dans tous ces cas les sorties sont dites inactives.

I-1-2- Réalisations diverses de circuits décodeurs

a- Réalisation d'un décodeur 1 parmi 4

Soit à réaliser un décodeur 1 parmi 4 (ou décodeur 2 vers 4) tels que :
- Une sortie active est à l'état haut
- Le circuit dispose d'une entrée de validation (V) active à l'état bas

Table de vérité

V	E_1	E_0	S_0	S_1	S_2	S_3
0	0	0	1	0	0	0
	0	1	0	1	0	0
	1	0	0	0	1	0
	1	1	0	0	0	1
1	X	X	0	0	0	0

Equations des sorties

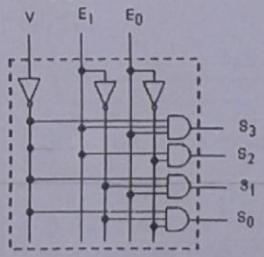
$S_0 = \overline{V}\,\overline{E_1}\,\overline{E_0}$

$S_1 = \overline{V}\,\overline{E_1}E_0$

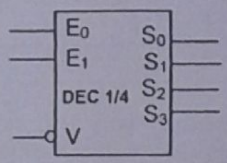
$S_2 = \overline{V}E_1\overline{E_0}$

$S_3 = \overline{V}E_1E_0$

Logigramme



Symbole logique



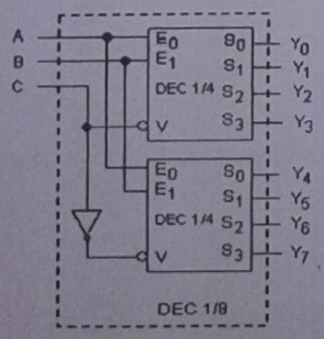
La boucle sur l'entrée V indique que V est active à l'état bas

b- Réalisation d'un décodeur 1 parmi 8

Soit à réaliser un décodeur 1 parmi 8 avec des boîtiers du décodeur 1 parmi 4 réalisé précédemment.

Le décodeur 1 parmi 8 nécessite deux boîtiers de décodeur 1 parmi 4. Par ailleurs le décodeur 1 parmi 8 a 3 entrées d'adresses. Chaque circuit décodeur 1 parmi 4 ayant 2 entrées d'adresses, alors la 3ème entrée sera l'entrée de validation des boîtiers.
Nommons A, B, C ses entrées d'adresses (avec A = LSB et C = MSB) et $Y_0, Y_1, Y_2, \ldots, Y_7$ ses sorties.

Le schéma du montage donne :



Il en ressort la table de fonctionnement ci-dessous :

C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
	0	1	0	1	0	0	0	0	0	0
	1	0	0	0	1	0	0	0	0	0
	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
	0	1	0	0	0	0	0	1	0	0
	1	0	0	0	0	0	0	0	1	0
	1	1	0	0	0	0	0	0	0	1

↑ Sélection alternative de chaque boîtier

Tel qu'il est réalisé, le décodeur 1 parmi 8 ainsi obtenu ne dispose pas d'entrée de validation. Pour cela il existe toujours une sortie parmi les 8 qui est valide ; ce qui est un inconvénient.
Modifions le montage ci-dessus pour obtenir un décodeur 1 parmi 8 ayant une entrée de validation E_N active au niveau haut.

Posons : V_1 l'entrée de validation du premier boîtier (B_1),
V_2 l'entrée de validation du deuxième boîtier (B_2),

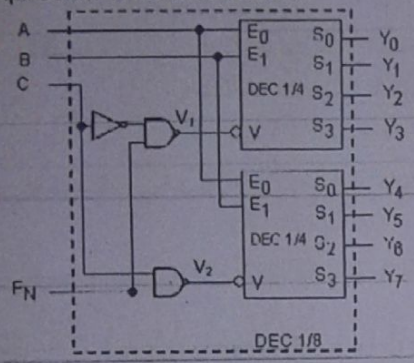
Scanned by CamScanner

A l'aide de l'entrée C on sélectionne l'un ou l'autre des deux boîtiers et cela, conditionné par l'état de E_N
On en ressort la table de vérité suivante :

E_N	C	V_1	V_2	
0	0	1	1	} Aucun boîtier n'est sélectionné tant que $E_N = 0$
0	1	1	1	
1	0	0	1	← Sélection du boîtier B_1
1	1	1	0	← Sélection du boîtier B_2

D'où les équations des entrées de validations suivantes : $V_1 = \overline{E_N \bar{C}}$ $\quad V2 = \overline{E_N C}$

Ce qui donne le schéma suivant :



Il en ressort la table de fonctionnement ci-dessous :

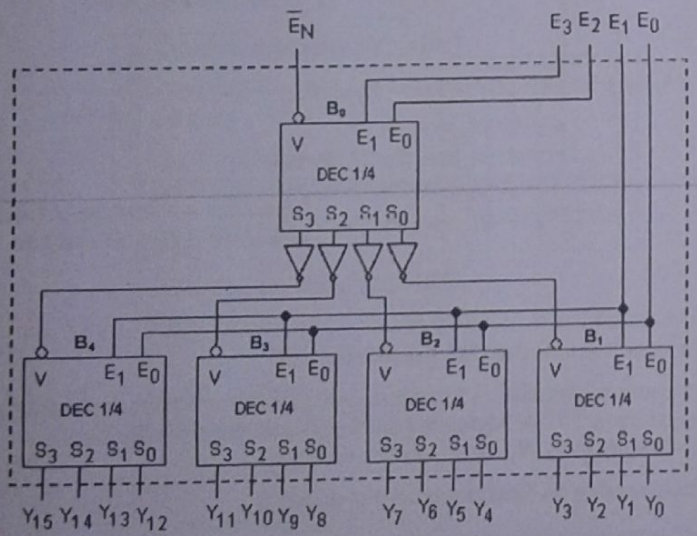
E_N	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	
0	X	X	X	0	0	0	0	0	0	0	0	← Sorties inactives tant que $E_N = 0$
1	0	0	0	1	0	0	0	0	0	0	0	
		0	1	0	1	0	0	0	0	0	0	
		1	0	0	0	1	0	0	0	0	0	
		1	1	0	0	0	1	0	0	0	0	
	1	0	0	0	0	0	0	1	0	0	0	
		0	1	0	0	0	0	0	1	0	0	
		1	0	0	0	0	0	0	0	1	0	
		1	1	0	0	0	0	0	0	0	1	

↑ Validation des sorties (sélection du circuit DEC 1/8)

c- Réalisation d'un décodeur 4 vers 16

Soit à réaliser un décodeur 1 parmi 16 ayant une entrée de validation notée $\bar{E}_N$, active à l'état bas avec des boîtiers du décodeur 1 parmi 4 réaliser précédemment.

Le circuit à réaliser a 4 entrées et 16 sorties. Ainsi 4 boîtiers fournissent les 16 sorties et 2 entrées. Un 5ème boîtier assurera la sélection des boîtiers et fournira les 2 autres entrées.



Par exemple la sortie Y_9 est donnée par : $Y_9 = \bar{V}_{B3}\bar{E}_1 E_0$.

Or $V_{B3} = \bar{S}_{2B0} \Rightarrow \bar{V}_{B3} = S_{2B0}$.

Par ailleurs $S_{2B0} = \overline{[\bar{E}_N]E_3\bar{E}_2}$.

Donc $Y_9 = \overline{[\bar{E}_N]}E_3\bar{E}_2\bar{E}_1E_0$

Remarque :

- *Généralisation de l'expression des sorties d'un décodeur :*

Si le décodeur dispose d'une entrée de validation V, active à l'état 1(état haut), alors l'expression d'une sortie Y_i est $Y_i = Vm_i$ avec m_i, le minterme d'adresses d'entrée dont l'expression binaire vaut l'indice i.

Pour un décodeur à 3 entrées E_0, E_1, E_2 par exemple, le minterme $m_5 = E_2\bar{E}_1E_0$.

Donc l'expression de la sortie Y_5 est $Y_5 = VE_2\bar{E}_1E_0$.

Si le décodeur dispose d'une entrée de validation V, active à l'état 0 (état bas), alors l'expression d'une sortie Y_i est $Y_i = \bar{V}m_i$.

Si les sorties du décodeur sont complémentées, avec une entée de validation V active à l'état 1 par exemple, alors $Y_i = \overline{Vm_i}$.

- Un décodeur avec un nombre élevé de sorties pourra toujours être réalisé avec des décodeurs disposant moins de sorties car la réalisation discrète à partir de portes logiques fait appel à un nombre pléthorique de portes.

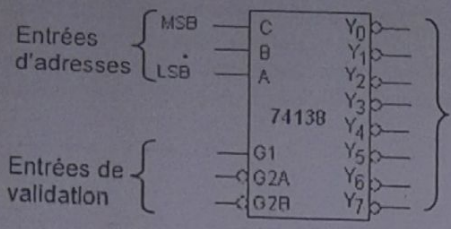
Scanned by CamScanner

I-1-3- Réalisation de fonctions logiques avec les décodeurs

Exemple du circuit intégré ***74138 : Décodeur 3 vers 8 à sorties complémentées***

a- Présentation du 74138

Symbole logique



Sorties complémentées avec $Y_i = \overline{(G_1 \bar{G}_{2A} \bar{G}_{2B})\, m_i}$

Minterme de validation du circuit

Minterme correspondant au code binaire des entrées d'adresses

Table de vérité

Entrées						Sorties							
Validation			Adresses										
G_1	G_{2A}	G_{2B}	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x								
x	x	1	x	x	x								
1	0	0	0	0	0	0	1	1	1	1	1	1	1
			0	0	1	1	0	1	1	1	1	1	1
			0	1	0	1	1	0	1	1	1	1	1
			0	1	1	1	1	1	0	1	1	1	1
			1	0	0	1	1	1	1	0	1	1	1
			1	0	1	1	1	1	1	1	0	1	1
			1	1	0	1	1	1	1	1	1	0	1
			1	1	1	1	1	1	1	1	1	1	0

b- Réalisation avec le 74138, la fonction $F = zyx + z\bar{y}x + \bar{z}y\bar{x} + \bar{y}\bar{x}$

La mise de F sous forme canonique donne $F = zyx + z\bar{y}x + \bar{z}y\bar{x} + z\bar{y}\bar{x} + \bar{z}\bar{y}\bar{x}$.

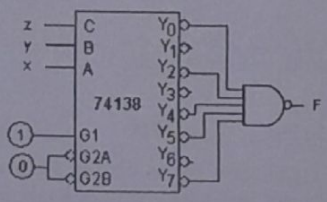
L'expression d'une sortie Yi du circuit intégré si nous supposons le circuit valide (**c'est-à-dire $G_1 = 1$, $G_{2A} = 0$, $G_{2B} = 0$**), est $Y_i = \overline{(CBA)_i}$; où $(CBA)_i$ est le minterme d'adresses correspondant à l'indice i.

Pour les variables de la fonction, posons z = MSB et x = LSB $\Rightarrow$ z = C, y = B, x = A.

Il ressort que $F = CBA + C\bar{B}A + \bar{C}BA + C\bar{B}\bar{A} + \bar{C}\bar{B}\bar{A}$

La valeur de F ne change pas si on écrit $F = \overline{\overline{CBA} + \overline{C\bar{B}A} + \overline{\bar{C}BA} + \overline{C\bar{B}\bar{A}} + \overline{\bar{C}\bar{B}\bar{A}}} \Rightarrow F = \bar{Y}_7 + \bar{Y}_5 + \bar{Y}_2 + \bar{Y}_4 + \bar{Y}_0 \Rightarrow F = \overline{Y_7 Y_5 Y_2 Y_4 Y_0}$

Le schéma de la fonction est donc :



I-1-4- Autres types de décodeurs

• Certains décodeurs n'utilisent pas toutes les 2^n combinaisons possibles du mot binaire d'entrée, mais seulement un sous-ensemble de celles-ci. C'est le cas par exemple du décodeur **1 parmi 10** (ou décodeur **4 vers 10** ou décodeur **BCD–Décimal**) pour lequel le décodage ne concerne seulement que les combinaisons d'entrées d'adresses $E_3E_2E_1E_0$ = 0000 à 1001 et portant sur les sorties S_0 à S_9. Pour les autres combinaisons d'entrées d'adresses $E_3E_2E_1E_0$ =1010 à 1111, les sorties restent non valides.

Un exemple ce type de circuit : le circuit ***7442 : décodeur BCD – Décimal à sorties complémentées***

Symbole logique

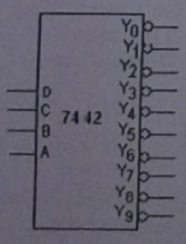


Table de vérité

Entrées d'adresse				Sorties									
D	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1										
1	1	0	0										
1	1	0	1										
1	1	1	0										
1	1	1	1										

• Un autre décodeur, un peu plus particulier est le decodeur **BCD vers 7 segments**. Ce décodeur permet pour les combinaisons binaires BCD c'est-à-dire $E_3E_2E_1E_0$ = 0000 à 1001 d'activer un sous-ensemble de sorties parmi un ensemble de 7 sorties dont les sorties actives permettent de commander les segments sur un afficheur à 7 segments afin de donner directement en décimal la valeur correspondante au code BCD d'entrée (voir chapitre 4, exercice 7).

Un exemple ce type de circuit : le circuit ***7449 : décodeur BCD – 7 segments à sorties collecteurs ouverts.***

Symbole logique

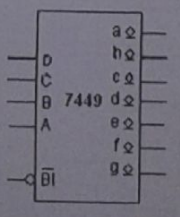


Table de vérité

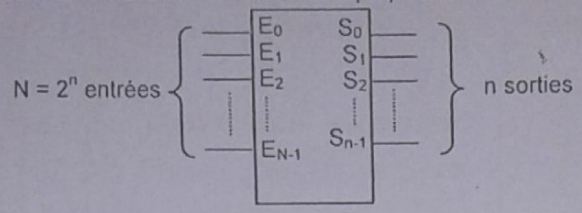
Validation	Entrées				Sorties							Identification des segments
$\overline{BI}$	D	C	B	A	a	b	c	d	e	f	g	
0	x	x	x	x	0	0	0	0	0	0	0	Eteint
1	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
1	0	0	1	0	1	1	0	1	1	0	1	2
1	0	0	1	1	1	1	1	1	0	0	1	3
1	0	1	0	0	0	1	1	0	0	1	1	4
1	0	1	0	1	1	0	1	1	0	1	1	5
1	0	1	1	0	0	0	1	1	1	1	1	6
1	0	1	1	1	1	1	1	0	0	0	0	7
1	1	0	0	0	1	1	1	1	1	1	1	8
1	1	0	0	1	1	1	1	0	0	0	1	9
1	1	0	1	0	0	0	0	1	1	1	1	Affichage erroné
1	1	0	1	1	0	0	1	1	0	0	1	
1	1	1	0	0	0	1	0	0	0	0	1	
1	1	1	0	1	1	0	0	1	0	0	1	
1	1	1	1	0	0	0	0	1	1	1	1	
1	1	1	1	1	0	0	0	0	0	0	0	

Scanned by CamScanner

I-2- Codeurs

I-2-1- Définition – présentation

Contrairement au décodeur, le codeur a 2^n entrées et n sorties. Il a la propriété de coder en binaire pur le numéro de l'entrée active.



CODEUR 2^n vers n

I-2-2- Réalisations diverses de circuits codeurs

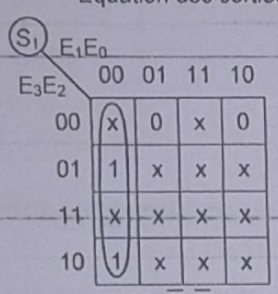
a- Réalisation d'un codeur 4 vers 2

soit à réaliser un codeur 4 vers 2 à entrées et sorties actives à l'état haut.

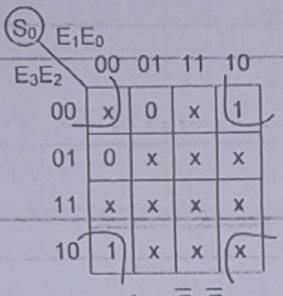
Table de vérité

E_3	E_2	E_1	E_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Equation des sorties

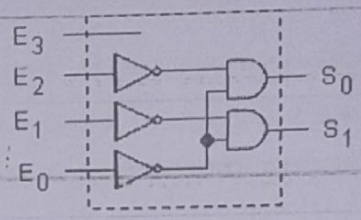


$S_1 = \overline{E}_1 \overline{E}_0$



$S_0 = \overline{E}_2 \overline{E}_0$

Logigramme



E_3 n'a d'utilité que d'adapter l'entrée du codeur

Remarque :
L'étude est effectuée pour seulement une entrée valide à la fois. Analysons le fonctionnement du montage si plusieurs entrées sont validées.

- Pour $E_3E_2E_1E_0 = 1100$, d'après le logigramme, on a $S_1S_0 = 10$.

Ce code est normalement destiné à la combinaison normale $E_3E_2E_1E_0 = 0100$.

- Pour $E_3E_2E_1E_0 = 0011$, on a $S_1S_0 = 00$.

Ce code est normalement destiné à la combinaison normale $E_3E_2E_1E_0 = 0001$.

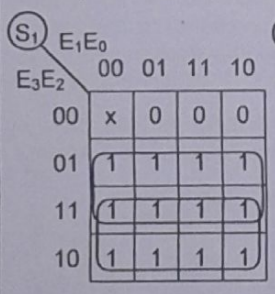
Il ressort de cette analyse une certaine ambiguïté dans l'utilisation de ce circuit. Pour éviter cet incovénient, on doit définir une priorité d'une entrée par rapport aux autres quand plusieurs entrées sont validées à la fois. Ce type de circuit est appelé **codeur de propriété** ou **encodeur**. Très généralement pour un encodeur lorsque plusieurs entrées sont activées, la priorité est donnée à l'entrée de numéro supérieur.

b – Réalisation d'un codeur de priorité 4 vers 2

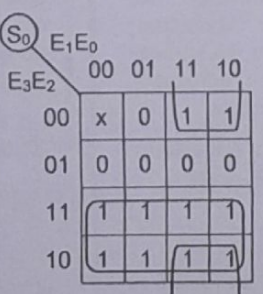
Table de vérité

E_3	E_2	E_1	E_0	S_1	S_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Equation des entrées

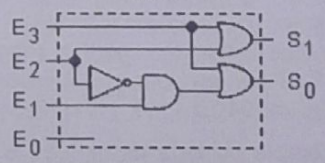


$S_1 = E_3 + E_2$



$S_0 = E_3 + \overline{E}_2 E_1$

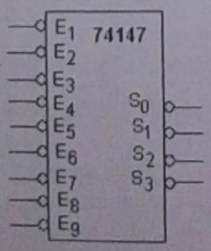
Logigramme



Lorsque aucune entrée n'est activée, on a $S_1S_0 = 00$. Tout se passe comme si l'entrée E_0 était activée.
Pour éviter cette ambiguïté soit :

- les circuits codeurs disposent très souvent d'une sortie additionnelle Co (carry out) permettant, d'après son niveau logique, de distinguer ces deux cas de figures.
- l'entrée E_0 est omise.

Remarque: Certains encodeurs ne fournissent pas la relation linéaire entre le nombre (2^n) d'entrées et le nombre (n) de sorties. C'est le cas par exemple de l'encodeur ***74147: codeur de priorité décimal–BCD à entrées et sorties complémentées***, dont le symbole logique et la table de vérité sont les suivants :



Entrées									Sorties			
E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	S_3	S_2	S_1	S_0
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

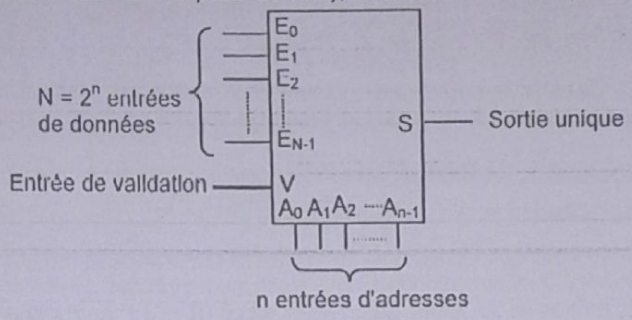
Scanned by CamScanner

II - CIRCUITS LOGIQUES D'AIGUILLAGE DE DONNEES

II-1- Multiplexeurs

II-1-1 Définition – Présentation

Un multiplexeur (ou Sélecteur de données) est un circuit de logique combinatoire qui possède **n entrées d'adresses, 2^n entrées de données et une sortie unique.**
Son rôle est de relier à la sortie unique, une entrée de données parmi les 2^n. Le numéro de l'entrée de données reliée à la sortie est la valeur décimale du code binaire pur appliqué sur les n entrées d'adresses.
Le multiplexeur possède également une entrée de validation du boîtier. Lorsque le boîtier n'est pas validé, la sortie reste invalide (soit 0 logique, soit 1 logique, ou soit l'état haute impédance : HZ), selon les caractéristiques spécifiées par le constructeur.



MULTIPLEXEUR 2^n voies

II -1-2- Réalisations diverses de circuits multiplexeurs

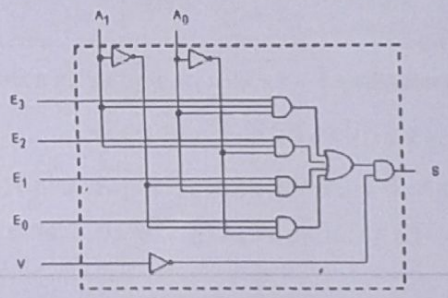
a- Réalisation d'un multiplexeur 4 voies avec une entrée de validation V active à l'état bas

Soit à réaliser un multiplexeur 4 vers1 disposant d'une entrée de validation active à l'état bas et à sortie non complémentée.
Le circuit a 4 voies de données (E_0, E_1, E_2, E_3), donc 2 entrées d'adresses (A_0, A_1) et sa sortie S est une fonction des entrées.

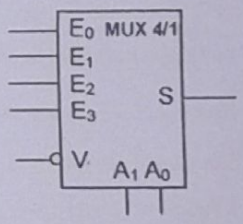
Table de vérité

V	A_1	A_0	S
0	0	0	E_0
	0	1	E_1
	1	0	E_2
	1	1	E_3
1	X	X	0

Logigramme



Symbole logique



Multiplexeur 4 vers 1
ou multiplexeur 4 voies
ou Multiplexeur 4 canaux

Expression de la sortie

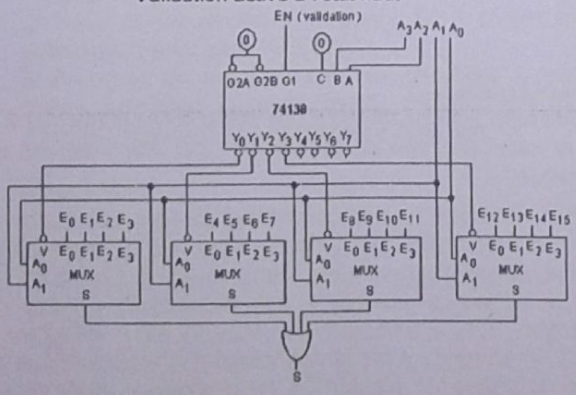
$$S = \overline{V}\left[(\overline{A_1}\,\overline{A_0})E_0 + (\overline{A_1}A_0)E_1 + (A_1\overline{A_0})E_2 + (A_1A_0)E_3\right]$$

b- Réalisation d'un multiplexeur 16 vers 1

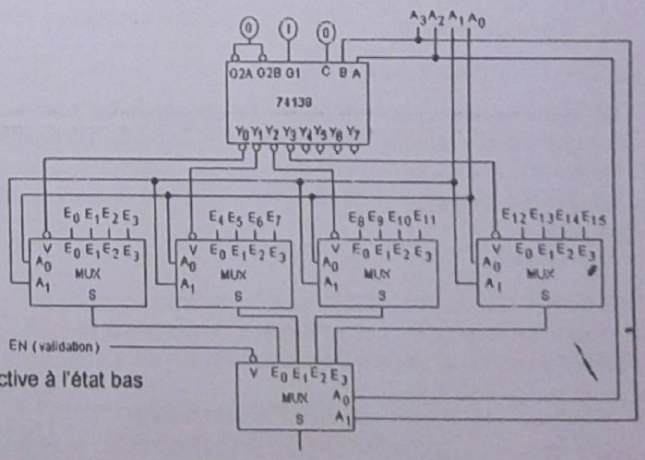
Soit à réaliser un multiplexeur 16 vers1 à l'aide de circuits multiplexeurs 4 voies ci-dessus.

Le circuit à réaliser fera nécessairement intervenir 4 boîtiers de multiplexeurs 4 voies fournissant les 16 entrées. Par ailleurs un circuit décodeur (le 74138 par exemple) devra servir pour la sélection alternative de chacun de ces boîtiers.

Validation active à l'état haut



OU



Validation active à l'état bas

On a bien dans les deux cas :

$$S = E_N\left[(\overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0})E_0 + (\overline{A_3}\,\overline{A_2}\,\overline{A_1}A_0)E_1 + (\overline{A_3}\,\overline{A_2}A_1\overline{A_0})E_2 + (\overline{A_3}\,\overline{A_2}A_1A_0)E_3 + (\overline{A_3}A_2\overline{A_1}\,\overline{A_0})E_4 + (\overline{A_3}A_2\overline{A_1}A_0)E_5 + (\overline{A_3}A_2A_1\overline{A_0})E_6 + (\overline{A_3}A_2A_1A_0)E_7 +$$

$$(A_3\overline{A_2}\,\overline{A_1}\,\overline{A_0})E_8 + (A_3\overline{A_2}\,\overline{A_1}A_0)E_9 + (A_3\overline{A_2}A_1\overline{A_0})E_{10} + (A_3\overline{A_2}A_1A_0)E_{11} + (A_3A_2\overline{A_1}\,\overline{A_0})E_{12} + (A_3A_2\overline{A_1}A_0)E_{13} + (A_3A_2A_1\overline{A_0})E_{14} + (A_3A_2A_1A_0)E_{15}\right]$$

Scanned by CamScanner

Remarques :

- *Généralisation de l'expression de sortie d'un multiplexeur*

- Si le multiplexeur dispose d'une entrée de validation E_N, active à l'état haut par exemple, l'expression de sa sortie est : $S = E_N[m_0E_0 + m_1E_1 + ... + m_iE_i + ... + m_{n-1}E_{n-1}]$.

Avec E_i, l'entrée d'indice i et m_i, le minterme d'adresses dont l'expression binaire vaut l'indice i.

On peut tout simplement écrire

$$S = E_N \sum_{i=0}^{n-1} m_i E_i$$ (où $\sum$ représente la somme logique)

-Si la sortie du multiplexeur est complémentée, alors $S = E_N \sum_{i=0}^{n-1} m_i \overline{E_i}$

- Un multiplexeur avec un nombre élévé d'entrées pourra toujours être réalisé par association de, multiplexeurs diposant de moins d'entrées car la réalisation discrète à partir de portes logiques fait appel à un nombre important de portes.

II-1-3- Réalisation de fonctions logiques avec les multiplexeurs.

Exemple du circuit intègré *74151: multiplexeur 8 vers 1 à double sortie (S et $\overline{S}$)*

a- présentation du 74151

Symbole logique

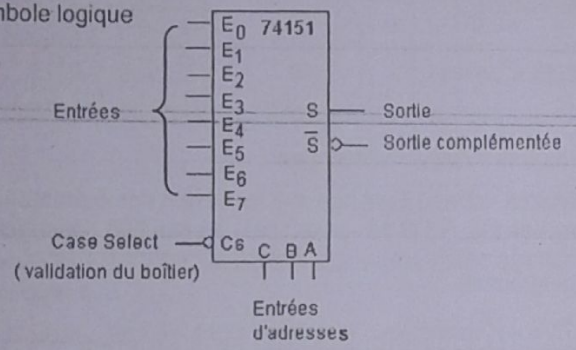


Table de vérité

Entrées				Sorties	
Adresses			Validation		
C	B	A	Cs	S	$\overline{S}$
X	X	X	1	0	1
0	0	0	0	E_0	$\overline{E_0}$
0	0	1		E_1	$\overline{E_1}$
0	1	0		E_2	$\overline{E_2}$
0	1	1		E_3	$\overline{E_3}$
1	0	0		E_4	$\overline{E_4}$
1	0	1		E_5	$\overline{E_5}$
1	1	0		E_6	$\overline{E_6}$
1	1	1		E_7	$\overline{E_7}$

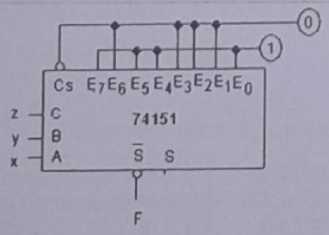
b- Réalisation de la fonction logique $F = \overline{z}\,\overline{y}\,x + \overline{z}\,y\,\overline{x} + z\,\overline{y}\,x + z\,y\,\overline{x}$ avec le 74151 en utilisant sa sortie complémentée

Posons C = z, B = y, A = x $\Rightarrow F = \overline{C}\,\overline{B}A + \overline{C}B\overline{A} + C\overline{B}A + CB\overline{A}$

Comparativement à $\overline{S}$ on aura : $\overline{S} = (\overline{C}\,\overline{B}A)\overline{E_1} + (\overline{C}B\overline{A})\overline{E_2} + (C\overline{B}A)\overline{E_3} + (CB\overline{A})\overline{E_6}$

Ainsi $F = \overline{S}$ si $\overline{E_1} = \overline{E_2} = \overline{E_3} = \overline{E_6} = 1$ et $\overline{E_0} = \overline{E_4} = \overline{E_5} = \overline{E_7} = 0$. C'est-à-dire si $E_1 = E_2 = E_3 = E_6 = 0$ et $E_0 = E_4 = E_5 = E_7 = 1$.

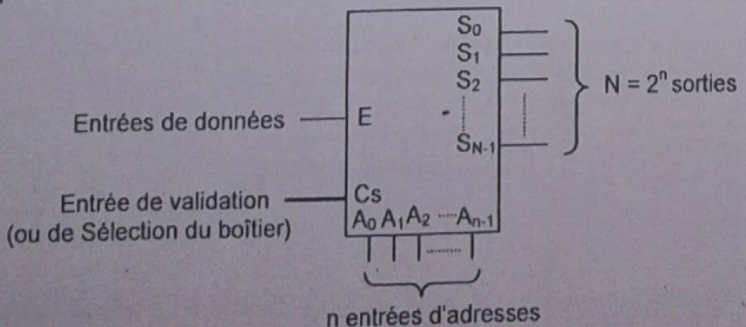
D'où le câblage ci-contre :



II-2- Démultiplexeur

II-2-1- Définition – présentation

Un démultiplexeur (ou distributeur de données) réalise la fonction inverse du multiplexeur .
Il a une **entrée unique, 2^n sorties et n entrées d'adresses**. Son rôle est de relier l'unique entrée à une seule sortie parmi les 2^n sorties telles que le numéro de la sortie connectée à l'unique entrée est la valeur décimale du code binaire pur admis sur les n entrées d'adresses. Il possède éventuellement une entrée de validation du circuit.



DEMULTIPLEXEUR 1 vers 2^n

Scanned by CamScanner

II-2-2- Réalisations diverses de circuits démultiplexeurs

Soit à réaliser un démultiplexeur 1 vers 4 disposant d'une entrée de validation active à l'état haut et à entrée non complémentée.

Table de vérité

Cs	A_1	A_0	S_3	S_2	S_1	S_0
0	X	X	0	0	0	0
1	0	0	0	0	0	E
	0	1	0	0	E	0
	1	0	0	E	0	0
	1	1	E	0	0	0

Equations des sorties

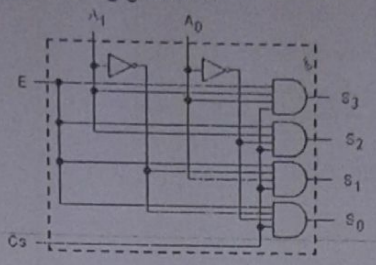
$S_0 = C_S(\overline{A_1}\,\overline{A_0})E$

$S_1 = C_S(\overline{A_1}A_0)E$

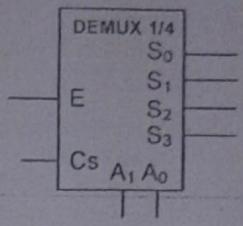
$S_2 = C_S(A_1\overline{A_0})E$

$S_3 = C_S(A_1A_0)E$

Logigramme



Symbole logique



Remarque:

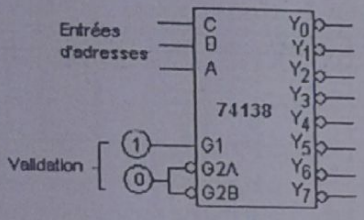
- *Généralisation de l'expression des sorties d'un démultiplexeur.*

$S_i = k_i E$ avec k_i le minterme d'adresses dont l'expression binaire vaut l'indice i.

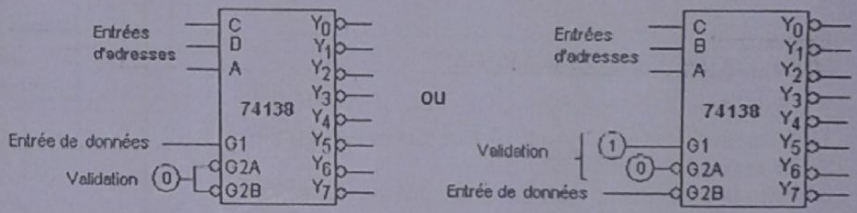
(par exemple pour i =5, $k_5 = C\overline{B}A \Rightarrow S_5 = (C\overline{B}A)E$).

- D'une manière générale les démultiplexeurs sont réalisés à l'aide de décodeurs car leurs fonctionnements sont analogues. Le démultiplexeur apparaît comme un décodeur muni d'une entrée conditionnelle. C'est-à-dire au lieu que la sortie sélectionnée du décodeur prenne la valeur logique 1 par exemple, elle prend plutôt la valeur se trouvant sur l'entrée conditionnelle. En technologie des circuits intégrés presque tous les décodeurs sont en même temps des démultiplexeurs.
D'où le nom **décodeur-démultiplexeur 3 vers 8** donné par exemple au **74138**. Un tel circuit peut donc être utilisé en décodeur ou en démultiplexeur.

Mode décodeur



Modes démultiplexeurs



EXERCICES

Exercice 1

1/ Réaliser la fonction $F = U(VW\overline{X} + \overline{V}\,\overline{W}X)$ avec le circuit 74LS138.

2/ Réaliser un décodeur 1 parmi 32 à sorties inversées uniquement avec des 74LS138.

3/ Proposer un autre schéma de décodeur 1 parmi 32 à sorties inversées avec moins de circuits 74LS138 que le schéma précédant et à validation à l'état bas.

Exercice 2

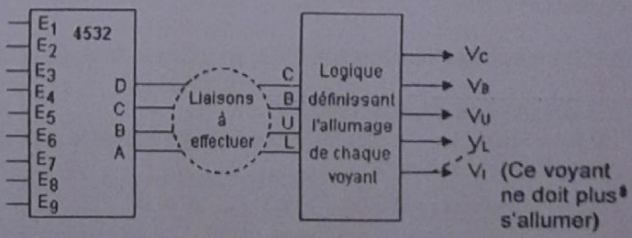
Un distributeur de boissons permet d'obtenir du café (C) ou du bouillon (B) et d'y ajouter éventuellement du sucre (U) ou du sel (L). Il dispose donc de 4 boutons rangés dans l'ordre CBUL, représentant les 4 ingrédients possibles et de 4 voyants (V_C,V_B,V_U,V_L) représentant les signaux de commande des robinets distribuant les ingrédients. De plus le dispositif comporte un 5e voyant V_i indiquant qu'il y a incompatibilité dans les choix effectués.

En effet, il y a incompabilité dans les cas suivants :

- Choix de plusieurs boissons simultanées,
- Choix de sel ou de sucre sans boissons,
- Choix de café salé (café et sel) ou de bouillon sucré,

1/ Déterminer la fonction logique définissant l'allumage de chaque voyant puis réaliser le logigramme du distributeur.

2/ On contourne la logique des choix incompatibles en adjoignant un codeur de priorité, le 4532 au dispositif, selon le synoptique suivant :



En s'inspirant de la table de vérité précédemment élaborée pour la question 1 , mentionner les entrées correspondantes du codeur de priorité permettant d'avoir la possibilité d'obtenir soit :

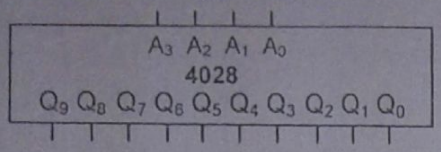
- Du café simple (café sans sucre)
- Du café sucré (café et sucre)
- Du bouillon simple
- Du bouillon salé

NB: le 4532 se présente fonctionnellement comme le 74147, mais les entrées et les sorties ne sont pas complémentées.

Scanned by CamScanner

Exercice 3

On dispose d'un circuit intégré 4028 : un décodeur BCD - décimal dont le symbole logique est ci-après.

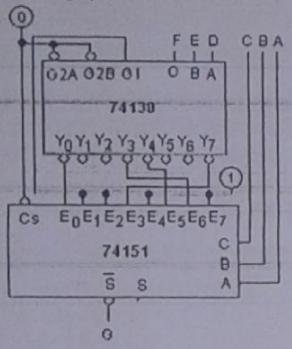


Le circuit, comme on le constate bien, ne possède pas d'entrée de validation.

Réaliser avec ce circuit un décodeur 1 parmi 8 avec une entrée de validation active à l'état bas.

Exercice 4

Déterminer l'expression logique de la fonction G.



Exercice 5

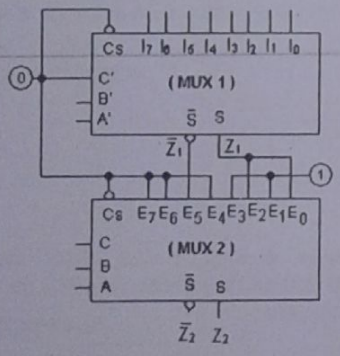
Réaliser la fonction logique suivante :

$F = D\overline{C}BA + \overline{D}\overline{C}BA + CBA + DCB\overline{A} + D\overline{C}\overline{B}\overline{A} + D\overline{C}B\overline{A}$

1/ Avec un décodeur 1 parmi 16 à sorties complémentées.
2/ Avec un multiplexeur 16 voies.
3/ Avec un décodeur 1 parmi 4 et un multiplexeur 4 voies.
4/ Avec un multiplexeur 8 voies.

Exercice 6

Soit le montage ci-dessous



1/ Donner l'équation logique de Z_2 en fonction de A, B, C et Z_1.
2/ Simplifier l'expression de Z_2
3/ En utilisant les circuits MUX1 et MUX2, réaliser le câblage de la fonction suivante :

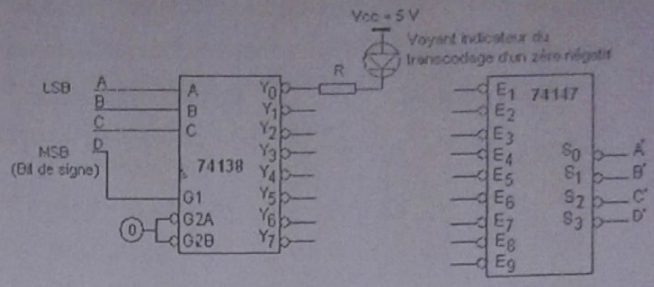
$Z_2 = \overline{A}\,\overline{C} + \overline{A}B\overline{Z_1} + \overline{B}C\overline{Z_1}$ avec $Z_1 = \overline{A'} \oplus \overline{B'}$

Exercice 7

On désire réaliser avec le 74138 et le 74147 un transcodeur des nombres binaires signés négatifs de 4 bits en complément vrai.

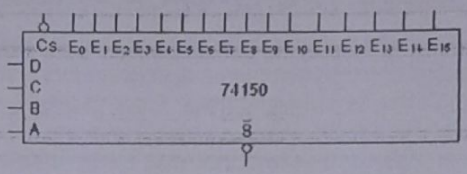
1/ Dresser la table présentant les nombres binaires négatifs de 4 bits DCBA (dont le MSB "D" est le bit de signe) et leur équivalent en complément vrai D'C'B'A'. (0 n'étant pas considéré comme pouvant être négatif).

2/ Réaliser le câblage sur le schéma suivant permettant d'obtenir le transcodeur binaire signé négatif-complément vrai.



Exercice 8

On veut réaliser un transcodeur binaire pur 5 bits-BCD avec uniquement des circuits multiplexeurs; notamment des ***74150 : multiplexeur 16 vers 1 à sortie inversée, à entrée de validation active au niveau bas*** et ***74151*** (dont on exploitera la sortie inversée).



Présentation du 74150

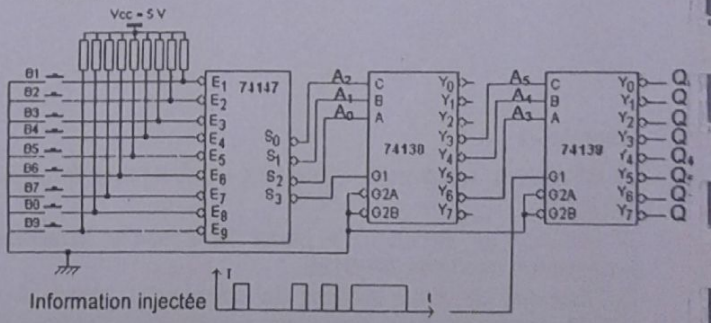
1/ Dresser la table de vérité du transcodeur suivant le modèle ci-dessous puis déterminer les expressions simplifiées des sorties.

Valeur	Entrées Binaires					Sorties Dizaines				Sorties Unité		
	z	y	x	w	v	D_3	D_2	D_1	D_0	U_3	U_2	U_1
0												
1												
2												

2/ Monter d'après les expressions obtenues qu'il nécessitera 4 circuits 74150 et 1 circuit 74151 puis réaliser la câblage nécessaire.

Exercice 9

Un circuit d'aiguillage d'informations est donné par le schéma ci-dessous :



1/ Remplir le tableau suivant :

Bouton actionné	Adresses basses			Adresses hautes			Sortie où est aiguillée l'information
	A_0	A_1	A_2	A_3	A_4	A_5	
B_1							
B_4							
B_3 et B_7							
B_6							

2/ Tracer en concordance de temps le signal fourni par l'information I et sa réponse sur une sortie Q_i aiguillée.

Scanned by CamScanner

CORRECTION

Exercice 1

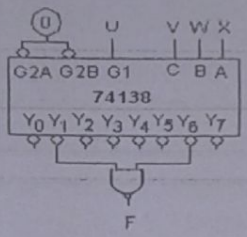
1/ $F = U(VW\bar{X} + \bar{V}\bar{W}X) = U(VW\bar{X}) + U(\bar{V}\bar{W}X)$

Posons $U = G_1$, $V = C$, $W = B$, $X = A$, $G_{2A} = G_{2B} = 0$

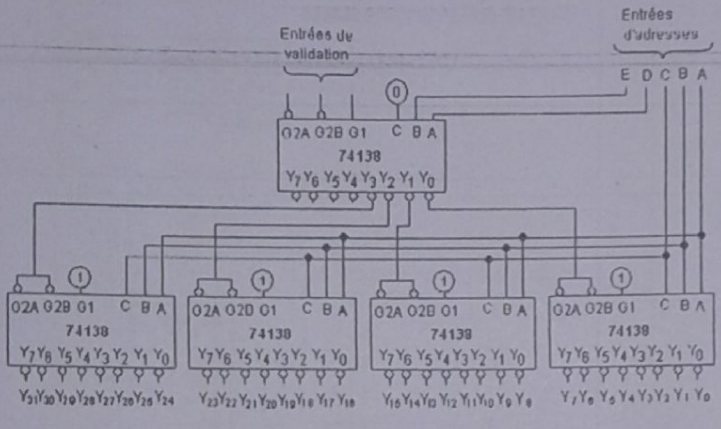
$\Rightarrow F = G_1(CB\bar{A}) + G_1(\bar{C}\bar{B}A) = G_1\overline{(CB\bar{A})} + G_1\overline{(\bar{C}\bar{B}A)}$

Ce qui donne que : $F = \bar{Y}_6 + \bar{Y}_1 = \overline{Y_6 Y_1}$

D'où le schéma suivant :

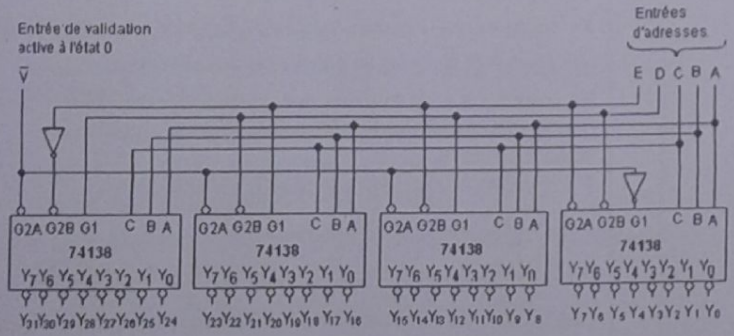


2/



On sélectionne un boîtier parmi 4 avec le code disponible sur ED puis on scrute les sorties du boîtier sélectionné avec le code CBA. Ce qui donne les 32 possibilités.

3/ A l'aide des entrées de validation G_1, G_{2A}, G_{2B}, on va assurer la sélection de chaque boîtier.



Exercice 2

1/ Table de vérité

C	B	U	L	V_C	V_B	V_U	V_L	V_I
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	0	1
0	0	1	1	0	0	0	0	1
0	1	0	0	0	1	0	0	0
0	1	0	1	0	1	0	1	0
0	1	1	0	0	0	0	0	1
0	1	1	1	0	0	0	0	1
1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	1
1	0	1	0	1	0	1	0	0
1	0	1	1	0	0	0	0	1
1	1	0	0	0	0	0	0	1
1	1	0	1	0	0	0	0	1
1	1	1	0	0	0	0	0	1
1	1	1	1	0	0	0	0	1

Equations logiques des sorties

$V_C = C\bar{B}\bar{U}\bar{L} + C\bar{B}U\bar{L} = C\bar{B}\bar{L}$

$V_B = \bar{C}B\bar{U}\bar{L} + \bar{C}BU\bar{L} = \bar{C}B\bar{U}$

$V_U = C\bar{B}U\bar{L}$

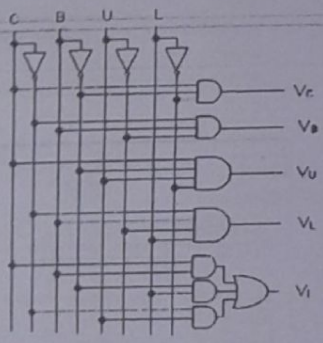
$V_L = \bar{C}B\bar{U}L$

V_I (UL / CB)

CB \ UL	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	1	1	1	1
10	0	1	1	0

$V_I = CB + \bar{B}L + \bar{C}U$

Logigramme



2/ D'après la table de vérité élaborée précédemment pour avoir :

- Du café simple (Ca), il faut CBUL=1000
- Du café sucré (Cs), il faut CBUL=1010
- Du bouillon simple (Bo), il faut CBUL=0100
- Du bouillon salé (Bs), il faut CBUL=0101

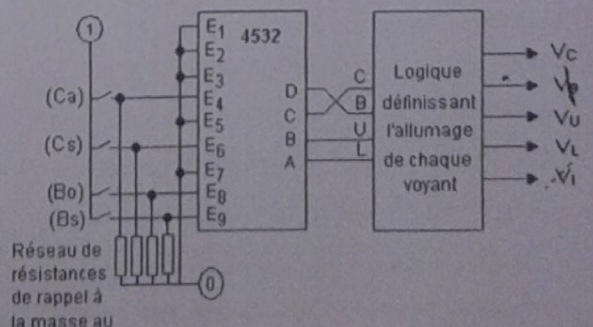
La combinaison **CBUL=1010** qui doit donner du café sucré (Cs) ne correspond à aucune combinaison d'entrée du codeur de priorité.

Il faut donc trouver la disposition des variables C, B, U, L permettant d'avoir des combinaisons inférieures à **1010**.

La disposition la plus pertinente est l'ordre **BCUL**. Ainsi :

- Pour (Ca) on a BCUL = 0100 : l'entrée numéro 4 doit être activée.
- Pour (Cs) on a BCUL = 0110 : l'entrée numéro 6 doit être activée.
- Pour (Bo) on a BCUL = 1000 : l'entrée numéro 8 doit être activée.
- Pour (Bs) on a BCUL = 1001 : l'entrée numéro 9 doit être activée.

D'où le montage suivant :



Scanned by CamScanner

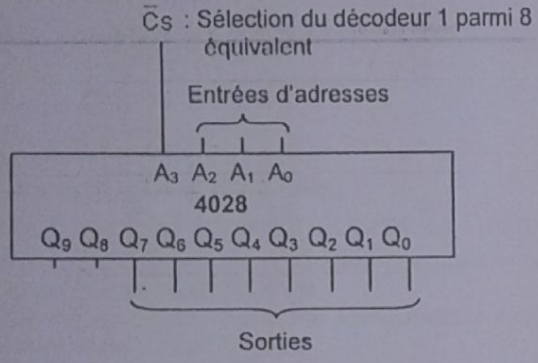
Exercice 3

Table de vérité du 4028 :

Adresses				Sorties									
A_3	A_2	A_1	A_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Un décodeur 1 parmi 8 a 3 entrées d'adresses.
Pour ce faire, il faudra utiliser sur le boîtier du 4028, les 3 entrées d'adresse A_2, A_1, A_0 comme les entrées d'adresses du décodeur à réaliser . Ce qui induit que ses 8 sorties sont les sorties Q_0 à Q_7.
Ce conditionnement impose que $\overline{A_3}$ doit être à 0.
En effet si $A_3 = 1$, ce sont les sorties inutilisées Q_8 ou Q_9 qui sont susceptibles d'être valides.

D'où le schéma suivant :



Exercice 4

Sorties utilisées du décodeur :

$Y_0 = \overline{\overline{F}\overline{E}\overline{D}} = E_0$ $\qquad$ $Y_4 = \overline{F\overline{E}\overline{D}} = E_5$

$Y_3 = \overline{\overline{F}ED} = E_6$ $\qquad$ $Y_7 = \overline{FED} = E_3$

Sortie du multiplexeur :

$$G = (\overline{C}\overline{B}\overline{A})\overline{E}_0 + (\overline{C}BA)\overline{E}_3 + (C\overline{B}A)\overline{E}_5 + (CB\overline{A})\overline{E}_6$$

On déduit l'expression logique de la fonction G :

$$G = (\overline{C}\overline{B}\overline{A})\overline{F}\overline{E}\overline{D} + (\overline{C}BA)FED + (C\overline{B}A)F\overline{E}\overline{D} + (CB\overline{A})\overline{F}ED$$

$$\Rightarrow G = \overline{F}\overline{E}\overline{D}\overline{C}\overline{B}\overline{A} + FED\overline{C}BA + F\overline{E}\overline{D}C\overline{B}A + \overline{F}EDCB\overline{A}$$

Exercice 5

1/ On a $F = D\overline{C}BA + \overline{D}\overline{C}BA + CBA + DCB\overline{A} + D\overline{C}\overline{B}\overline{A} + D\overline{C}B\overline{A}$

Sous forme canonique on obtient :

$$F = D\overline{C}BA + \overline{D}\overline{C}BA + DCBA + \overline{D}CBA + DCB\overline{A} + D\overline{C}\overline{B}\overline{A} + D\overline{C}B\overline{A}$$

En posant $D = A_3$, $C = A_2$, $B = A_1$, $A = A_0$, on a :

$$F = A_3\overline{A}_2A_1A_0 + \overline{A}_3\overline{A}_2A_1A_0 + A_3A_2A_1A_0 + \overline{A}_3A_2A_1A_0 + A_3A_2A_1\overline{A}_0 + A_3\overline{A}_2\overline{A}_1\overline{A}_0 + A_3\overline{A}_2A_1\overline{A}_0$$

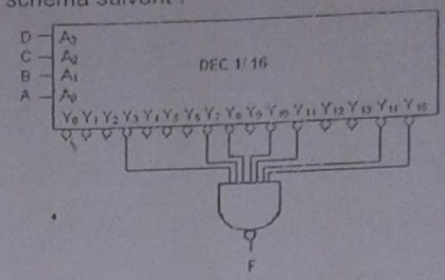
La valeur de F ne change pas si on écrit :

$$F = \overline{\overline{A_3\overline{A}_2A_1A_0}} + \overline{\overline{\overline{A}_3\overline{A}_2A_1A_0}} + \overline{\overline{A_3A_2A_1A_0}} + \overline{\overline{\overline{A}_3A_2A_1A_0}} + \overline{\overline{A_3A_2A_1\overline{A}_0}} + \overline{\overline{A_3\overline{A}_2\overline{A}_1\overline{A}_0}} + \overline{\overline{A_3\overline{A}_2A_1\overline{A}_0}}$$

Ainsi en fonction des sorties du décodeur on a :

$$F = \overline{Y}_{11} + \overline{Y}_3 + \overline{Y}_{15} + \overline{Y}_7 + \overline{Y}_{14} + \overline{Y}_8 + \overline{Y}_{10} \Rightarrow F = \overline{Y_{11}\,Y_3\,Y_{15}\,Y_7\,Y_{14}\,Y_8\,Y_{10}}$$

D'où le schéma suivant :



2/ Soient A_3 , A_2, A_1, A_0, les entrées d'adresses du multiplexeur, E_0, E_1, E_2, ..., E_{15} ses entrées de données et S sa sortie.
Comparativement aux mintermes de F on a :

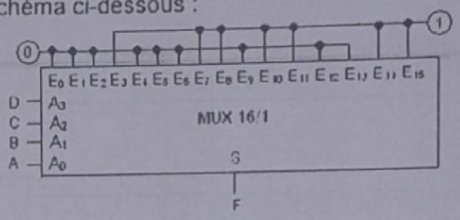
$$S = (A_3\overline{A}_2A_1A_0)E_{11} + (\overline{A}_3\overline{A}_2A_1A_0)E_3 + (A_3A_2A_1A_0)E_{15} + (\overline{A}_3A_2A_1A_0)E$$

$$(A_3A_2A_1\overline{A}_0)E_{14} + (A_3\overline{A}_2\overline{A}_1\overline{A}_0)E_8 + (A_3\overline{A}_2A_1\overline{A}_0)E_{10}$$

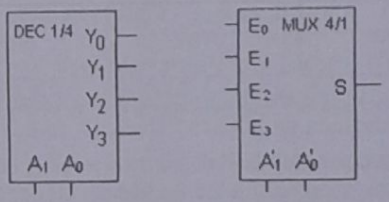
En posant $D = A_3$, $C = A_2$, $B = A_1$, $A = A_0$,

on obtient $F = S$ si $E_3 = E_7 = E_8 = E_{10} = E_{11} = E_{14} = E_{15} = 1$, les autres entrées étant bien sûres mises à 0.

D'où le schéma ci-dessous :



3/ Soient les symboles logiques du décodeur et du multiplexeur suivants :



En tout état de cause c'est la sortie du multiplexeur qui sera utile pour donner F
Posons $D = A_1$, $C = A_0$, $B = A'_1$, $A = A'_0$. On a donc par comparaison aux mintermes de F :

$$F = (A_1\overline{A}_0)(A'_1A'_0) + (\overline{A}_1\overline{A}_0)(A'_1A'_0) + (A_1A_0)(A'_1A'_0) + (\overline{A}_1A_0)(A'_1A'_0) +$$
$$(A_1A_0)(A'_1\overline{A}'_0) + (A_1\overline{A}_0)(\overline{A}'_1\overline{A}'_0) + (A_1\overline{A}_0)(A'_1\overline{A}'_0)$$

En exprimant F en fonction des sorties du décodeur correspondantes aux mintermes d'adresses de décodage, on a :

$$F = Y_2(A'_1A'_0) + Y_0(A'_1A'_0) + Y_3(A'_1A'_0) + Y_1(A'_1A'_0) +$$
$$Y_3(A'_1\overline{A}'_0) + Y_2(\overline{A}'_1\overline{A}'_0) + Y_3(A'_1\overline{A}'_0)$$

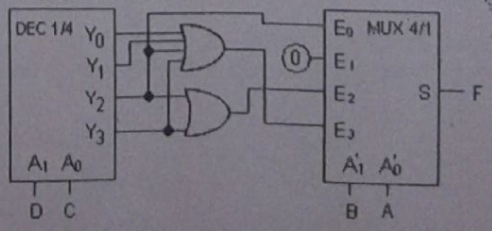
$$\Rightarrow F = (A'_1A'_0)(Y_0 + Y_1 + Y_2 + Y_3) + (A'_1\overline{A}'_0)(Y_2 + Y_3) + (\overline{A}'_1\overline{A}'_0)Y_2$$

Les mintermes d'adresses du multiplexeur qui interviennent dans l'expression de F permettent de déduire que $S = F$ si

$$S = (A'_1A'_0)E_3 + (A'_1\overline{A}'_0)E_2 + (\overline{A}'_1\overline{A}'_0)E_0$$

Il ressort que $F = S$ si $E_3 = Y_0 + Y_1 + Y_2 + Y_3$
$E_2 = Y_2 + Y_3$
$E_0 = Y_2$

D'où le montage ci-dessous :



Scanned by CamScanner

4/ L'expression initiale

$F = \overline{D}CBA + D\overline{C}BA + CBA + DCB\overline{A} + D\overline{C}B\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$

donne après simplification $F = \overline{C}BA + CBA + D\overline{C}A + DCB\overline{A}$

Si on considère que les variables C, B, et A sont les trois entrées d'adresse du multiplexeur, alors en introduisant dans tous les termes de l'expression de F ces trois variables d'adresse, on obtient :

$F = \overline{C}BA + CBA + D\overline{C}BA + D\overline{C}B\overline{A} + DCB\overline{A}$

Comparativement aux mintermes d'adresse qui interviennent dans cette expression de F, la sortie S du multiplexeur pour être semblable à F doit donner :

$S = E_3\overline{C}DA + E_7CBA + E_0\overline{C}B\overline{A} + E_2\overline{C}B\overline{A} + E_0CB\overline{A}$

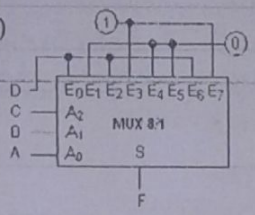
Il ressort que la sortie du multiplexeur donne F si :

$E_3 = E_7 = 1$

$E_0 = E_2 = E_6 = D$ (4ème variable de F)

$E_1 = E_4 = E_5 = 0$

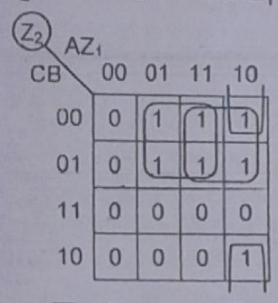
D'où le montage ci-contre :



Exercice 6

1/ $Z_2 = \overline{C}BA + C\overline{B}A + CB\overline{A} Z_1 + C\overline{B}\,\overline{A} Z_1 + CB\overline{A}\,\overline{Z_1}$

2/



$Z_2 = \overline{C}Z_1 + \overline{C}A + \overline{B}A\overline{Z_1}$

3/ $Z_1 = \overline{A'} \oplus \overline{B'} \Rightarrow Z_1 = \overline{A}B' + A'\overline{B'}$

Explicitement Z_1 est exprimée en fonction des entrées de MUX1.

Soit alors $Z_1 = \overline{A'}B'I_1 + A'\overline{B'}I_2$.

C'est-à-dire implicitement $I_1 = I_2 = 1$et $I_0 = I_3 = I_4 = I_5 = I_6 = I_7 = 0$.

On a $Z_2 = \overline{C}\,\overline{A} + B\overline{A}\,\overline{Z_1} + C\overline{B}\,\overline{Z_1}$.

Faisons intervenir dans tous les termes de Z_2 toutes les variables C, B, A :

$Z_2 = \overline{C}B\overline{A} + \overline{C}\overline{B}A + CB\overline{A}\,\overline{Z_1} + \overline{C}B\overline{A}\,\overline{Z_1} + \overline{C}\overline{B}A\overline{Z_1} + C\overline{B}\,\overline{A}\,\overline{Z_1}$

$\Rightarrow Z_2 = \overline{C}B\overline{A} + \overline{C}\overline{B}A(1 + Z_1) + CB\overline{A}\,\overline{Z_1} + C\overline{B}A\overline{Z_1} + C\overline{B}\,\overline{A}\,\overline{Z_1}$

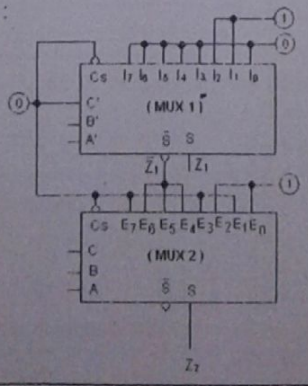
$\Rightarrow Z_2 = \overline{C}B\overline{A} + \overline{C}\overline{B}A + CB\overline{A}\,\overline{Z_1} + C\overline{B}A\overline{Z_1} + C\overline{B}\,\overline{A}\,\overline{Z_1}$

Z_2 peut être explicitement exprimée en fonction des entrées de MUX2 par :

$Z_2 = C\overline{B}\,\overline{A}E_0 + \overline{C}\overline{B}AE_2 + \overline{C}B\overline{A}E_6 + \overline{C}B\overline{A}E_5 + C\overline{B}\,\overline{A}E_4$

D'où $E_0 = E_2 = 1$, $E_6 = E_5 = E_4 = \overline{Z_1}$ et $E_1 = E_3 = E_7 = 0$

On a donc le montage suivant :



Exercice 7

1/Table des binaires signés négatifs de 4 bits et leurs équivalents en complément vrai :

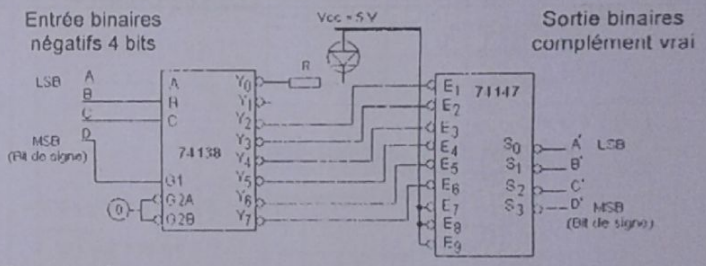
Valeurs	Binaires signés négatifs				Complément vrai			
	D	C	B	A	D'	C'	B'	A'
-1	1	0	0	1	1	1	1	1
-2	1	0	1	0	1	1	1	0
-3	1	0	1	1	1	1	0	1
-4	1	1	0	0	1	1	0	0
-5	1	1	0	1	1	0	1	1
-6	1	1	1	0	1	0	1	0
-7	1	1	1	1	1	0	0	1

2/ D'après le schéma :

- Pour le code binaire de -1 on a la sortie $Y_1 = 0$ (sortie active) pendant que le 74147 réagit comme si toutes ses entrées sont à un (inactive). Donc Y_1 n'est reliées à aucune entrée de l'encodeur.
- Pour le code binaire de -2 on a la sortie $Y_2 = 0$ (sortie active) pendant que le 74147 réagit comme si son entrée E_1 seule est activée. Donc Y_2 doit être reliée à l'entrée E_1 de l'encodeur.
- Pour le code binaire de -3 on a la sortie $Y_3 = 0$ (sortie active) pendant que le 74147 réagit comme si son entrée E_2 seule est activée. Donc Y_3 doit être reliée à l'entrée E_2 de l'encodeur.
- Par analyse similaire on trouve que : $Y_4 = E_3$, $Y_5 = E_4$, $Y_6 = E_5$, $Y_7 = E_6$.

Ainsi toues les autres entrées E_7, E_8, E_9 doivent être portées à 1 afin de les rendre Inactives pulsqu'elles ne sont connectées à aucune sortie du décodeur.

On a donc le câblage suivant :



Exercice 8

1/ Table de vérité du transcodeur :

Valeurs	Entrées Binaires					Sorties Dizaines				Sorties Unités			
	z	y	x	w	v	D_3	D_2	D_1	D_0	U_3	U_2	U_1	U_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	0	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	0	0	0	1	1
4	0	0	1	0	0	0	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	0	0	0	1	0	1
6	0	0	1	1	0	0	0	0	0	0	1	1	0
7	0	0	1	1	1	0	0	0	0	0	1	1	1
8	0	1	0	0	0	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	0	0	0	1	0	0	1
10	0	1	0	1	0	0	0	0	1	0	0	0	0
11	0	1	0	1	1	0	0	0	1	0	0	0	1
12	0	1	1	0	0	0	0	0	1	0	0	1	0
13	0	1	1	0	1	0	0	0	1	0	0	1	1
14	0	1	1	1	0	0	0	0	1	0	1	0	0
15	0	1	1	1	1	0	0	0	1	0	1	0	1
16	1	0	0	0	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	0	0	0	1	0	1	1	1
18	1	0	0	1	0	0	0	0	1	1	0	0	0
19	1	0	0	1	1	0	0	0	1	1	0	0	1
20	1	0	1	0	0	0	0	1	0	0	0	0	0
21	1	0	1	0	1	0	0	1	0	0	0	0	1
22	1	0	1	1	0	0	0	1	0	0	0	1	0
23	1	0	1	1	1	0	0	1	0	0	0	1	1
24	1	1	0	0	0	0	0	1	0	0	1	0	0
25	1	1	0	0	1	0	0	1	0	0	1	0	1
26	1	1	0	1	0	0	0	1	0	0	1	1	0
27	1	1	0	1	1	0	0	1	0	0	1	1	1
28	1	1	1	0	0	0	0	1	0	1	0	0	0
29	1	1	1	0	1	0	0	1	0	1	0	0	1
30	1	1	1	1	0	0	0	1	1	0	0	0	0
31	1	1	1	1	1	0	0	1	1	0	0	0	1

Scanned by CamScanner

Expressions simplifiées des sorties :

La table de vérité laisse clairement apparaître que les dizaines s'expriment avec deux bits car $D_3 = D_2 = 0$.

NB : Les cases vides dans tous les tableaux sont des 0.

D_1 (x w v / z y)

zy \ xwv	000	001	011	010	110	111	101	100
00								
01								
11	1	1	1	1	1	1	1	1
10					1	1	1	1

$D_1 = zy + zx$

D_0 (x w v / z y)

zy \ xwv	000	001	011	010	110	111	101	100
00								
01		1	1		1	1	1	
11					1	1		
10	1	1	1	1				

$D_0 = \bar{z}yw + \bar{z}yx + yxw + z\bar{y}\bar{x}$

U_3 (x w v / z y)

zy \ xwv	000	001	011	010	110	111	101	100
00								
01	1	1						
11						1	1	
10			1	1				

$U_3 = \bar{z}y\bar{x}\bar{w} + z\bar{y}\bar{x}w + zyx\bar{w}$

U_2 (x w v / z y)

zy \ xwv	000	001	011	010	110	111	101	100
00					1	1	1	1
01					1	1		
11	1	1	1	1				
10	1	1						

$U_2 = \bar{z}\bar{y}x + \bar{z}xw + zy\bar{x} + z\bar{x}\bar{w}$

U_1 (x w v / z y)

zy \ xwv	000	001	011	010	110	111	101	100
00		1	1		1	1		
01							1	1
11		1	1					
10	1	1			1	1		

$U_1 = \bar{z}\bar{y}w + \bar{y}xw + z\bar{y}x\bar{w} + zy\bar{x}w + z\bar{y}\bar{x}\bar{w}$

U_0 (x w v / z y)

zy \ xwv	000	001	011	010	110	111	101	100
00	1	1				1	1	
01	1	1				1	1	
11	1	1				1	1	
10	1	1				1	1	

$U_0 = v$

2/ D'après les expressions des sorties on note que :

- U_0 ne dépend que de la seule varible v ; donc sa réalisation ne nécessite aucun circuit.
- D_1 dépend des trois variables z, y, x ; donc sa réalisation nécessite un 74151.
- D_0, U_3, U_2, U_1 dépendent des quatre variables z, y, x, w ; donc la réalisation de chacune de ces sorties nécessite un 74150. Ce qui donne un total de quatre circuits 74150.

On conclut alors que la réalisation du transcodeur nécessitera un seul circuit 74151 et quatre circuits 74150.

Le câblage est précédé par la mise sous la première forme canonique de chacune des expressions des sorties en fonction des varibles qui leur sont justes nécessaires.

Pour obtenir ces formes canoniques assez rapidement sans passer par la methode algébrique nous allons utiliser une autre méthode avec des tableaux de karnaugh. Cette méthode consiste à remplir un tableau de karnaugh à partir de l'expression de la fonction puis, sans faire de regroupements, à écrire le minterme correspndant à chaque case du tableau contenant 1.

Ainsi, d'après l'expression de $D_1 = zy + zx$, on a le tableau de karnaugh suivant :

D_1 (y x / z)

z \ yx	00	01	11	10
0				
1	1	1	1	

$\Rightarrow$ La première forme canonique de D_1 est

$D_1 = z\bar{y}x + zy\bar{x} + zyx$

En posant par rapport aux variables d'adresses du 74151, z = C, y = B, x = A, on obtient $D_1 = C\bar{B}A + CB\bar{A} + CBA$.

Comparativement aux mintermes d'adresse qui interviennent dans cette expression de D_1, la sortie $\bar{S}$ du multiplexeur 74151 pour être semblable à D_1 doit donner quand il est validé ($C_S = 0$) :

$\bar{S} = C\bar{B}A\bar{E}_5 + CB\bar{A}\bar{E}_7 + CBA\bar{E}_6$.

Il ressort que cette sortie du multiplexeur donne D_1 si :

$$\begin{cases} \bar{E}_5 = \bar{E}_6 = \bar{E}_7 = 1 \Rightarrow E_5 = E_6 = E_7 = 0 \\ \text{et} \\ \bar{E}_0 = \bar{E}_1 = \bar{E}_2 = \bar{E}_3 = \bar{E}_4 = 0 \Rightarrow E_0 = E_1 = E_2 = E_3 = E_4 = 1 \end{cases}$$

Par ailleurs, d'après l'expression de D_0 on déduit le tableau de karnaugh suivant :

D_0 (x w / z y)

zy \ xw	00	01	11	10
00				
01		1	1	1
11			1	
10	1	1		

$\Rightarrow$ La première forme canonique de D_1 est

$D_0 = \bar{z}y\bar{x}w + \bar{z}yxw + \bar{z}yx\bar{w} + zyxw + z\bar{y}\bar{x}\bar{w} + z\bar{y}\bar{x}w$

En posant par rapport aux variables d'adresses du 74150, z = D, y = C, x = B, w = A, on obtient

$D_0 = \bar{D}C\bar{B}A + \bar{D}CBA + \bar{D}CB\bar{A} + DCBA + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A$.

Comparativement aux mintermes d'adresse qui interviennent dans cette expression de D_0, la sortie $\bar{S}$ du multiplexeur 74150 pour être semblable à D_0 doit donner quand il est validé ($C_S = 0$) :

$\bar{S} = \bar{D}C\bar{B}A\bar{E}_5 + \bar{D}CBA\bar{E}_7 + \bar{D}CB\bar{A}\bar{E}_6 + DCBA\bar{E}_{15} + D\bar{C}\bar{B}\bar{A}\bar{E}_8 + D\bar{C}\bar{B}A\bar{E}_9$.

Il ressort que la sortie du multiplexeur donne D_0 si :

$$\begin{cases} E_5 = E_6 = E_7 = E_8 = E_9 = E_{15} = 0 \\ \text{et} \\ E_0 = E_1 = E_2 = E_3 = E_4 = E_{10} = E_{11} = E_{12} = E_{13} = E_{14} = 1 \end{cases}$$

Par rapport aux variables z, y, x, w, on note que U_3 est déjà sous forme canonique. Ainsi en fonction des variables d'adresses du 74150 on a : $U_3 = \bar{D}C\bar{B}\bar{A} + D\bar{C}\bar{B}A + DCB\bar{A}$.

La sortie $\bar{S}$ du second multiplexeur 74150 pour être semblable à U_3 doit donner quand il est validé ($C_S = 0$) :

$\bar{S} = \bar{D}C\bar{B}\bar{A}\bar{E}_4 + D\bar{C}\bar{B}A\bar{E}_9 + DCB\bar{A}\bar{E}_{14}$.

Il ressort que la sortie du second multiplexeur 74150 donne U_3 si pour ce circuit on a :

$$\begin{cases} E_4 = E_9 = E_{14} = 0 \\ \text{et} \\ E_0 = E_1 = E_2 = E_3 = E_5 = E_6 = E_7 = E_8 = E_{10} = E_{11} = E_{12} = E_{13} \\ = E_{15} = 1 \end{cases}$$

D'après son expression on déduit le tableau de karnaugh suivant de U_2 :

U_2 (x w / z y)

zy \ xw	00	01	11	10
00			1	1
01			1	
11	1	1		
10	1			

$\Rightarrow$ La première forme canonique de D_1 est

$U_2 = \bar{z}\bar{y}xw + \bar{z}y\bar{x}\bar{w} + \bar{z}yxw + zyx\bar{w} + zy\bar{x}w + z\bar{y}\bar{x}\bar{w}$

Ainsi en fonction des variables d'adresses du troisième circuit 74150 on a :

Scanned by CamScanner

$U_2 = \bar{D}\bar{C}BA + \bar{D}\bar{C}B\bar{A} + \bar{D}CBA + DCB\bar{A} + DC\bar{B}A + DC\bar{B}\bar{A}$.

La sortie $\bar{S}$ du troisième multiplexeur 74150 pour être semblable à U_2 doit donner quand il est validé ($C_S = 0$) :

$\bar{S} = \bar{D}\bar{C}BA\bar{E}_3 + \bar{D}\bar{C}B\bar{A}\bar{E}_2 + \bar{D}CBA\bar{E}_7 + DCB\bar{A}\bar{E}_{12} + DC\bar{B}A\bar{E}_{13} + DC\bar{B}\bar{A}\bar{E}_8$

Il ressort que la sortie du toisième multiplexeur 74150 donne U_2 si pour ce circuit on a :

$$\begin{cases} E_2 = E_3 = E_7 = E_8 = E_{12} = E_{13} = 0 \\ \text{et} \\ E_0 = E_1 = E_4 = E_5 = E_6 = E_9 = E_{10} = E_{11} = E_{14} = E_{15} = 1 \end{cases}$$

D'après son expression on déduit le tableau de karnaugh suivant de U_1 :

U_1 : z y \ x w	00	01	11	10
00		1	1	
01				1
11		1		
10	1		1	

$\Rightarrow$ La première forme canonique de U_1 est

$U_1 = \bar{z}\bar{y}\bar{x}w + \bar{z}\bar{y}xw + \bar{z}yx\bar{w} + zy\bar{x}w + z\bar{y}\bar{x}\bar{w} + \bar{z}\bar{y}xw$

On peut en fonction des variables d'adresses du qutrième circuit 74150 que

$U_1 = \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}BA + \bar{D}CB\bar{A} + DC\bar{B}A + D\bar{C}\bar{B}\bar{A} + D\bar{C}BA$.

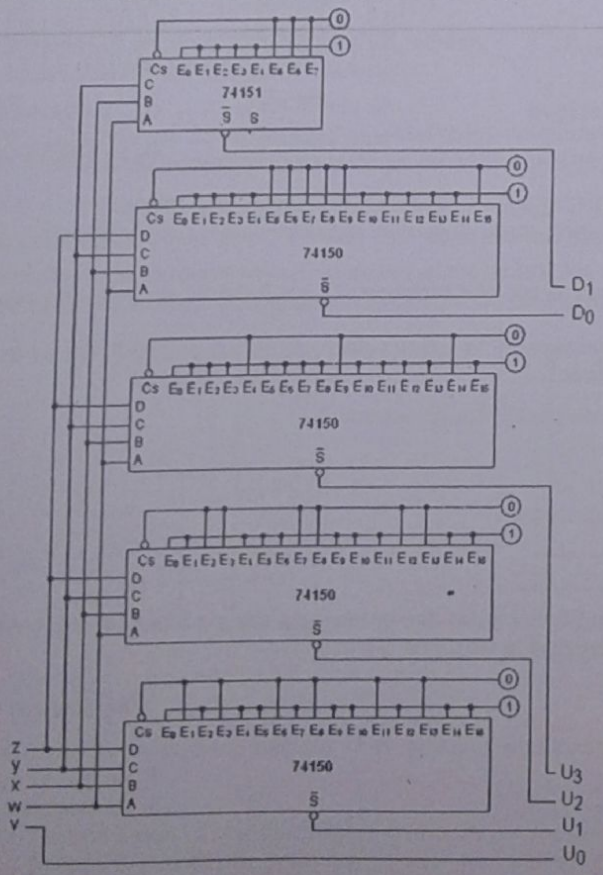
La sortie $\bar{S}$ du quatrième multiplexeur 74150 pour être semblable à U_1 doit donner quand il est validé ($C_S = 0$) :

$\bar{S} = \bar{D}\bar{C}\bar{B}A\bar{E}_1 + \bar{D}\bar{C}BA\bar{E}_3 + \bar{D}CB\bar{A}\bar{E}_6 + DC\bar{B}A\bar{E}_{13} + D\bar{C}\bar{B}\bar{A}\bar{E}_8 + D\bar{C}BA\bar{E}_{11}$

Il ressort que la sortie du quatrième multiplexeur 74150 donne U_1 si pour ce circuit on a :

$$\begin{cases} E_1 = E_3 = E_6 = E_8 = E_{11} = E_{13} = 0 \\ \text{et} \\ E_0 = E_2 = E_4 = E_5 = E_7 = E_9 = E_{10} = E_{12} = E_{14} = E_{15} = 1 \end{cases}$$

Il ressort de toute cette analyse le schéma de câblage suivant :

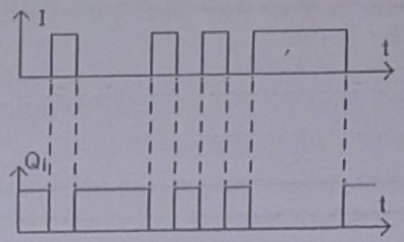


Exercice 9

1/

Bouton actionné	Adresses basses			Adresses hautes			Sortie où est aiguillée l'information
	A_0	A_1	A_2	A_3	A_4	A_5	
B_1	1	1	0	1	1	0	Q_3
B_4	0	1	1	0	1	1	Q_6
B_3 et B_7	0	0	0	1	1	1	Q_7
B_6	0	0	1	1	0	1	Q_5

2/



Scanned by CamScanner

CHAPITRE VIII – BASCULES

I- DIFFERENTS TYPES DE BASCULES ET LEUR REALISATION

Une bascule est un circuit de la logique séquentielle qui, pour une combinaison d'états logiques de ses entrées présente en sortie deux états complémentaires stables dont les valeurs dépendent à la fois des variables d'entrées et des états précédents en sortie. Une bascule se présente comme une mémoire élémentaire qui ne peut mémoriser qu'un seul bit.
Il existe essentiellement 4 types de bascules qui sont :

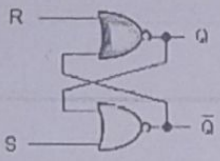
- La bascule RS (Reset - Set)
- La bascule JK (sans signification particulière)
- La bascule D (Data)
- La bascule T (Trigger)

I-1- Bascule RS asynchrone et bascule RS synchrone

I-1-1- Bascule RS asynchrone

Les bascules asynchrones sont des bascules qui ne disposent d'aucune entrée d'autorisation de fonctionner. En effet, dès qu'une information logique est présente en entrée, automatiquement, la bascule réagit.

La bascule RS asynchrone est constituée de deux portes NOR montées comme ci-après :



Les équations des sorties lorsque le circuit est au repos : $Q_n = \overline{R + \overline{Q}_n}$ $\quad \overline{Q}_n = \overline{S + Q_n}$ (n étant l'instant d'observation).

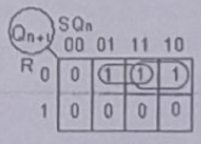
A l'instant où l'on donne une nouvelle valeur à R et (ou) à S on a : $Q_{n+1} = \overline{R + \overline{Q}_n}$ $\quad \overline{Q}_{n+1} = \overline{S + Q_n}$.

En remplaçant Q_n et $\overline{Q}_n$ par leur valeur de repos (valeur antérieure), on obtient que : $Q_{n+1} = \overline{R}(S + Q_n)$ $\quad \overline{Q}_{n+1} = \overline{S}(R + \overline{Q}_n)$.

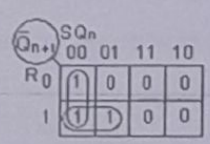
D'après ces équations on déduit la table de vérité ou table de fonctionnement suivante :

R	S	Q_n	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	0	0	1	$Q_{n+1} = Q_n$ *(Mémorisation)*
0	0	1	1	0	
0	1	0	1	0	$Q_{n+1} = 1$ *(Mise à Un ou Set)*
0	1	1	1	0	
1	0	0	0	1	$Q_{n+1} = 0$ *(Mise à Zéro ou Reset)*
1	0	1	0	1	
1	1	0	0	0	$Q_{n+1} = \overline{Q}_{n+1}$ *(Combinaison ambiguë de R et S à éviter)*
1	1	1	0	0	

Remarque : D'après la table de vérité, on peut ressortir les équations de Q_{n+1} et $\overline{Q}_{n+1}$ par diagrammes de karnaugh.



$Q_{n+1} = \overline{R}S + \overline{R}Q_n$
$Q_{n+1} = \overline{R}(S + Q_n)$



$\overline{Q}_{n+1} = \overline{S}R + \overline{S}\overline{Q}_n$
$\overline{Q}_{n+1} = \overline{S}(R + \overline{Q}_n)$

Du commentaire sur la table de fonctionnement il ressort que :

- Si on a R = 0 et S = 0, alors la sortie ne change pas d'état : $Q_{n+1} = Q_n$. On dit qu'il y a une mémorisation
- Si on a R = 0 et S = 1, alors la sortie passe à l'état 1 quel que soit son état antérieur : $Q_{n+1} = 1$. On dit qu'il y a une **mise à un ou SET**
- Si on a R = 1 et S = 0, alors la sortie passe à l'état 0 quel que soit son état antérieur : $Q_{n+1} = 0$. On dit qu'il y a une **mise à zéro ou RESET**
- Si on a R = 1 et S = 1, alors la sortie directe et son complémentaire ont la même valeur ; ce qui est ambiguë. On dit qu'il y indétermination. Cette combinaison ***(R = S = 1) est donc à éviter.***

De tout ce qui précède on déduit la table de vérité simplifiée ou **table de vérité réduite** suivante :

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	indétermination

Par ailleurs on peut ressortir la **table de transition ou table de conditionnement des entrées** qui est une table qui présente les valeurs logiques à donner aux variables d'entrée pour conserver ou changer la valeur de la sortie.

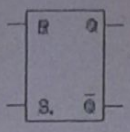
Q_n	R	S	Q_{n+1}
0	ø	0	0
0	0	1	1
1	1	0	0
1	0	ø	1

Scanned by CamScanner

Remarque :

- Dès qu'une information logique est présente sur l'une ou l'autre des entrées, automatiquement, la bascule réagit. C'est-à-dire que sa sortie est instantanément influencée par toute variation en entrée. On dit dans ce cas que son mode de fonctionnement est asynchrone. D'où le nom de bascule asynchrone.

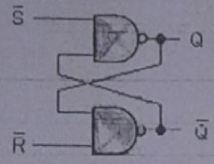
Le symbole logique d'une bascule RS asynchrone est :



- *Bascule RS asynchrone à portes NAND*

Elle est généralement appelée bascule $\bar{R}\bar{S}$ asynchrone.

Son logigramme :

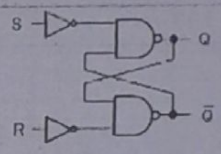


Sa table de vérité :

$\bar{R}$	$\bar{S}$	Q_{n+1}	
0	0	indétermination	
0	1	0	Reset
1	0	1	Set
1	1	Q_n	Mémorisation

On remarque d'après sa table de vérité, qu'une bascule RS à portes NAND donne pour les mêmes combinaisons d'entrées, la réaction complémentaire d'une bascule RS à portes NOR. C'est pourquoi la bascule RS à portes NAND est, nommée bascule $\bar{R}\bar{S}$. C'est-à-dire que pour qu'une telle bascule réagisse comme une bascule RS normale (bascule RS à portes NOR), il faut inverser ses entrées.

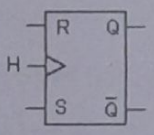
Ainsi on croise très souvent le logigramme suivant pour une bascule RS :



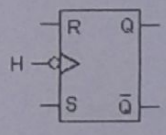
I-1-2- Bascule RS synchrone ou bascule RSH

Pour une bascule synchrone, le changement d'état de la sortie qui correspond à une nouvelle combinaison d'états des entrées ne peut s'effectuer que sur le front actif (front montant ou front descendant) d'un signal de synchronisation ou signal d'horloge.
Le front d'un signal d'horloge est une **transition raide** c'est à dire **le passage rapide d'un niveau logique à un autre**.

Symbole logique d'une bascule RS synchrone :



Bascule RS à fonctionnement sur front montant (⌐↑) ou bascule RS active sur front montant

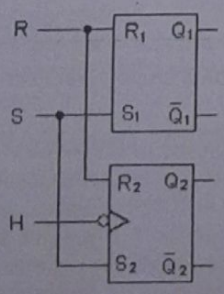


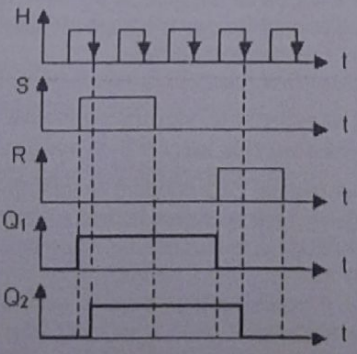
Bascule RS à fonctionnement sur front descendant (⌐↓) ou bascule RS active sur front descendant

Remarque :

- Une bascule active sur front montant réagit **uniquement à l'instant** de la **brusque transition** montante du signal d'horloge mais reste insensible toute la durée du niveau haut du signal d'horloge.
- Une bascule active sur front descendant réagit **uniquement à l'instant** de la **brusque transition** descendant du signal d'horloge mais reste insensible toute la durée du niveau bas du signal d'horloge.

Exemple : Les chronogrammes ci-dessous présentent les allures des sorties Q_1 et Q_2 du montage ci-contre dont on connaît l'évolution des entrées de données et d'horloge. On considère au départ que les sorties sont à 0.





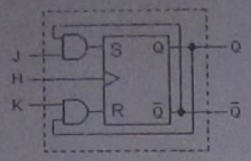
I-2- Bascule JK

La bascule JK dérive de la bascule RSH avec une amélioration dont **l'objectif est de résoudre le problème de la combinaison interdite R = S = 1.**
Pour la bascule JK :

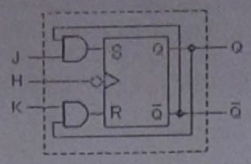
- l'entrée J joue un rôle assimilable à celui de l'entrée S de la bascule RSH.
- l'entrée K joue un rôle assimilable à celui de l'entrée R de la bascule RSH.

Scanned by CamScanner

Schémas logiques :

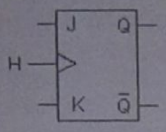


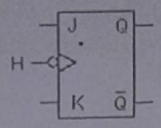
Bascule JK active sur front montant



Bascule JK active sur front descendant

Symboles logiques :





Equation logique de la sortie Q :

D'après le schéma logique, on a $R = KQ_n$ et $S = J\bar{Q}_n$. Or pour la bascule RS on a $Q_{n+1} = \bar{R}(S + Q_n)$

On obtient donc : $Q_{n+1} = \overline{KQ_n}(J\bar{Q}_n + Q_n) \Rightarrow Q_{n+1} = (\bar{K} + \bar{Q}_n)(J\bar{Q}_n + Q_n) = \bar{K}J\bar{Q}_n + \bar{K}Q_n + J\bar{Q}_n + \bar{Q}_nQ_n = J\bar{Q}_n(\bar{K} + 1) + \bar{K}Q_n + 0$

$$\Rightarrow Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

On en déduit les différentes tables de fonctionnement suivantes :

Table de vérité

J	K	Q_n	Q_{n+1}	
0	0	0	0	$Q_{n+1} = Q_n$ *(Mémorisation)*
0	0	1	1	
0	1	0	0	$Q_{n+1} = 0$ *(Mise à zéro ou Reset)*
0	1	1	0	
1	0	0	1	$Q_{n+1} = 1$ *(Mise à un ou Set)*
1	0	1	1	
1	1	0	1	$Q_{n+1} = \bar{Q}_n$ *(Mode basculement ou Trigger)*
1	1	1	0	

Table de vérité réduite

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

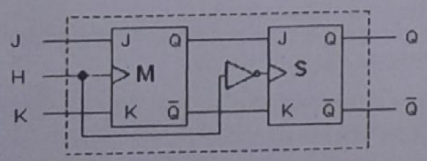
Table de transition

Q_n	J	K	Q_{n+1}
0	0	ø	0
0	1	ø	1
1	ø	1	0
1	ø	0	1

Remarque : Bascule JK Maître-Esclave

C'est une bascule qui réagit avec les deux fronts (montant et descendant) du signal d'horloge.

Schéma logique d'une bascule JK maître-esclave active sur front montant.



Les deux bascules M (Master) et S (Slave) sont de type JK à fonctionnement sur front montant. Au front montant de l'horloge H, la bascule M réagit en fonction des informations extérieures disponibles sur ses entrées J, K alors que la bascule S est insensible à cause de la porte inverseuse. Au front descendant de l'horloge la bascule S réagit en fonction des données reçues de la bascule M alors que la bascule M reste insensible à toutes données extérieures.

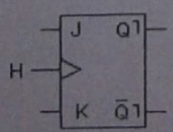
En somme, au front montant de l'horloge la bascule M communique avec l'extérieur puis fournit la réponse à la bascule S et au front descendant, la bascule S réagit en copiant sur sa sortie la donnée qui est fournie par M (réaction en bascule D car J=Q et K=$\bar{Q}$).

On dit alors que la bascule M réagit en maître (Master) et la bascule S en esclave (Slave).

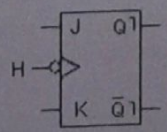
La bascule étudiée est dite **active sur front** montant car c'est pendant ce front qu'elle communique avec l'extérieur.

On déduit ainsi aisément que pour une bascule JK maître-esclave active sur front descendant, la communication avec les données extérieures se fait sur le front descendant de l'horloge et la réponse sur sa sortie s'obtient sur le front montant.

Symboles logiques :



Bascule JK maître-esclave active sur front montant



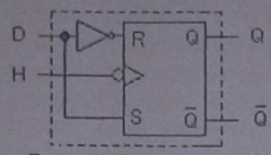
Bascule JK maître-esclave active sur front descendant

Avec une bascule Maître-Esclave on peut écrire une donnée et la conserver à volonté tant que l'horloge ayant permis la copie ne change pas de niveau. On les appelle aussi **data lock out**.

Scanned by CamScanner

I-3- Bascule D

La bascule D n'a qu'une unique entrée et elle dérive de la bascule RSH ou de la bascule JK selon le même principe schématique.

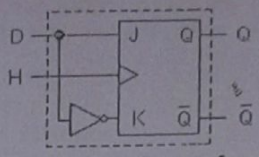


Bascule D active sur front descendant

Equation binaire de la sortie Q :

On a $R = \bar{D}$ et $S = D$. Or $Q_{n+1} = \bar{R}(S + Q_n)$.

Donc $Q_{n+1} = D(D + Q_n) \Rightarrow Q_{n+1} = D$



Bascule D active sur front montant

Equation binaire de la sortie Q :

On a $J = D$ et $K = \bar{D}$. Or $Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$.

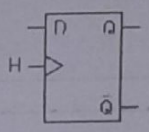
Donc $Q_{n+1} = D\bar{Q}_n + DQ_n \Rightarrow Q_{n+1} = D$

Table de vérité

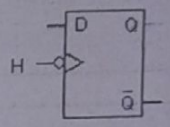
D	Q_{n+1}
0	0
1	1

Une bascule D copie directement à la sortie au front actif de l'horloge la donnée disponible sur son entrée.

Symboles logiques



Bascule D active sur front montant



Bascule D active sur front descendant

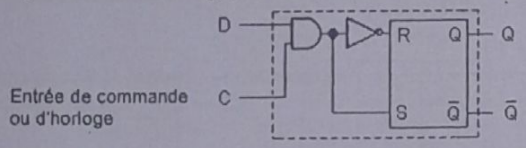
Remarque : Bascule D à verrouillage

Une bascule D à verrouillage (ou **verrou D**, ou **D-Latch**) est une bascule D dont le déclenchement se fait par le niveau logique actif (haut ou bas) du signal d'horloge.

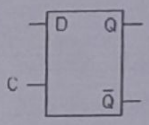
En effet, un verrou D actif au niveau haut est **transparent** ou **ouvert** tant que le signal d'horloge est à l'état haut *(c'est-à-dire que la sortie prendra constamment la valeur de la donnée entrée tant que le signal d'horloge est à l'état haut)* ; et il est **verrouillé** ou **fermé** tant que le signal d'horloge est à l'état bas *(c'est-à-dire que la sortie ne change pas tant que le signal d'horloge est à l'état bas)*.

On a le fonctionnement contraire pour un verrou D actif au niveau bas.

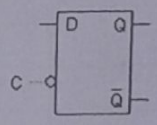
Exemple de schéma logique d'un D-Latch actif au niveau haut :



symboles logiques



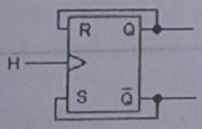
D-Latch actif au niveau haut



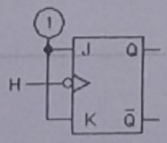
D-Latch actif au niveau bas

I-4- Bascule T

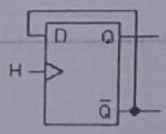
Elle dérive des bascules RSH, JK ou D selon les schémas suivants :



Exemple de bascule T active sur front montant construite avec une bascule RSH



Exemple de bascule T active sur front descendant construite avec une bascule JK



Exemple de bascule T active sur front montant construite avec une bascule D

Une bascule T n'existe pas en tant que telle en circuit intégré mais se construit en pratique à l'aide des autres types de bascules.

D'après les équations des bascules RS, JK ou D, il ressort que pour chacun des schémas ci-dessus on a : $Q_{n+1} = \bar{Q}_n \Rightarrow$ la sortie change constamment d'état à chaque front actif du signal d'horloge : on dit qu'on a un fonctionnement en **mode basculement** ou en **Trigger.**

Remarque : La version bascule T réalisée avec une bascule JK peut s'exploiter autrement en ne mettant pas en permanence au niveau haut, les deux entrées J et K liées.

En effet, les deux entrées J et K liées peuvent être nommées **T (Toggle)** et être utilisées pour contrôler le changement d'état de la bascule. Si **T=0**, la sortie **garde sa valeur** au front actif du signal d'horloge. Si **T=1**, la bascule **change d'état** au front actif du signal d'horloge.

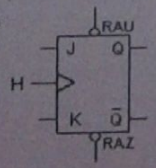
II- ENTREES DE FORÇAGE " PRESET " ET "CLEAR" D'UNE BASCULE

Il existe très souvent sur les bascules en plus des entrées de données et d'horloge, deux entrées de forçages prioritaires sur les autres entrées, qui sont :

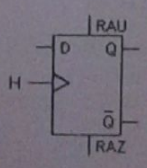
- **L'entrée de forçage à 1** ou entrée **preset** ou **RAU** (Remise à Un) : lorsque cette entrée est active, la sortie **Q demeure à 1** quel que soit ce qu'il y a sur les autres entrées (entrées de donnée et d'horloge)
- **L'entrée de forçage à 0** ou entrée **clear** ou **RAZ** (Remise à Zéro) : lorsqu'elle est active la sortie **Q demeure à 0**.

Les entrées de forçage sont des entrées qui s'activent avec un niveau logique.

Exemples



Bascule JK active sur front montant avec RAZ et RAU actives au niveau bas



Bascule D active sur front montant avec RAZ et RAU actives au niveau haut

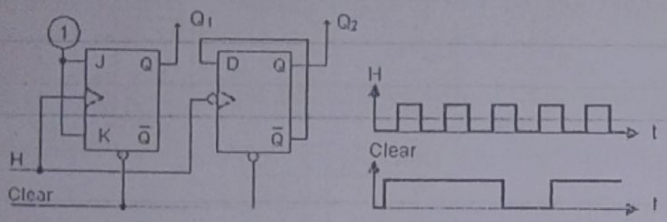
Scanned by CamScanner

Remarque :
Les entrées de forçages sont prioritaires sur les autres entrées, mais il n'existe pas une sélection de priorité entre elles-mêmes. Donc pour une bascule, si l'on active la RAU alors la RAZ doit être inactivée. Et vice-versa. C'est-à-dire qu'il ne faut pas activer simultanément pour la même bascule les entrées RAZ et RAU sinon la sortie devient instable.

La plupart des bascules usuelles actuellement disposent d'entrées de forçage. Si sur un schéma contenant des bascules ces entrées ne sont pas mentionnées, c'est sous entendu qu'elles sont mises au niveau logique qui les rend inactives ; donc permettant aux bascules de fonctionner normalement. Par contre si elles sont mentionnées, il faut alors tenir compte de leur présence dans toutes analyses.

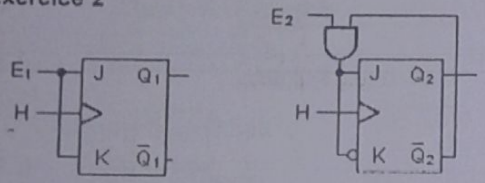
EXERCICES

Exercice 1



On suppose initialement $Q_1=Q_2=0$
Déterminer leurs allures sachant que les allures de H et Clear sont celles ci-dessus :

Exercice 2

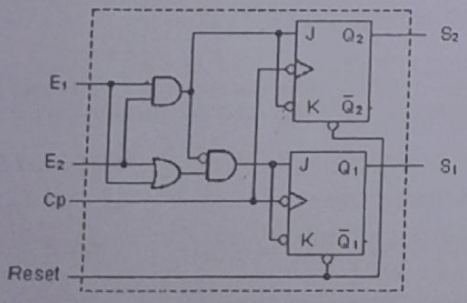


Montage 1 Montage 2

a/ Dresser la table de vérité de chaque montage.
b/ On désire obtenir sur la sortie d'une bascule un signal carré lorsque $E_i=1$ et un niveau logique 0 tant que $E_i = 0$, (i = 0 ;1). Lequel des deux montages est le plus adapté. Justifier le choix.

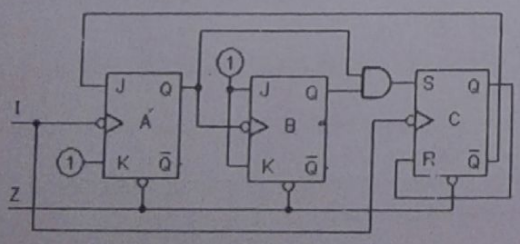
Exercice 3

Soit le dispositif électronique suivant :



a/ Quel niveau logique faut-il appliquer sur l'entrée Reset pour que le couple (S_2 ; S_1) soit égale à (0 ; 0) ?
b/ Dresser la table des séquences de ce dispositif si l'entrée Reset est désactivée.

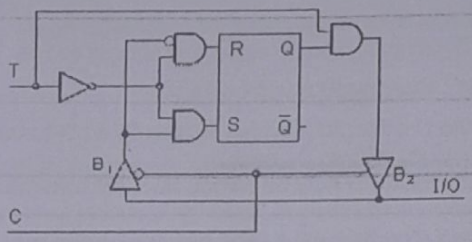
Exercice 4



a / On envoie le signal suivant ⊓ (brève impulsion) sur l'entrée Z. Quels sont les états des bascules ?
b / Dresser sa table d' implication séquentielle en considérant Qc comme le MSB.
c/ Commenter la table.

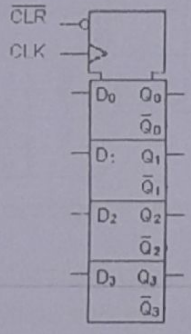
Exercice 5

Expliquer en fonction de T et C que ce montage permet d'écrire et de lire une donnée à partir du même conducteur ligne I/0.

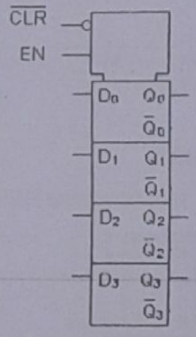


Exercice 6

On dispose des deux circuits intégrés ci-dessous.

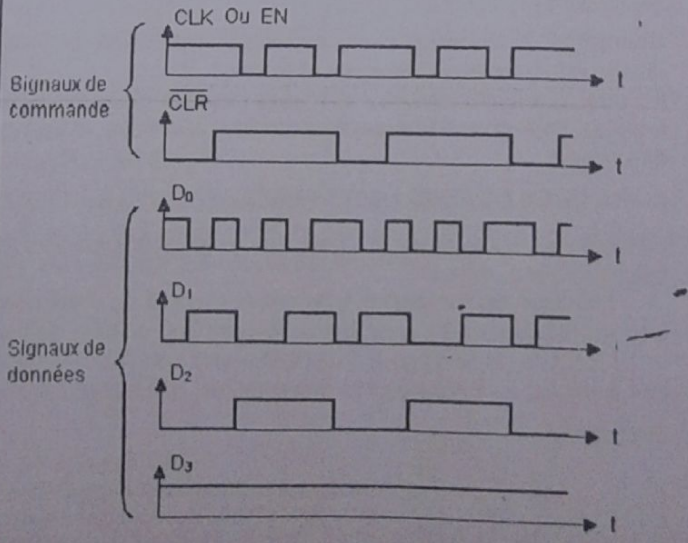


Quadruple bascule D à horloge (CIK) communes et remise à zéro ($\overline{CLR}$) communes.



Quadruple verrou D à validation (EN) communes et remise à zéro ($\overline{CLR}$) communes.

Tracer les chronogrammes des sorties de chaque circuit si à chacun d'eux on envoie les signaux de commande et de données suivants :

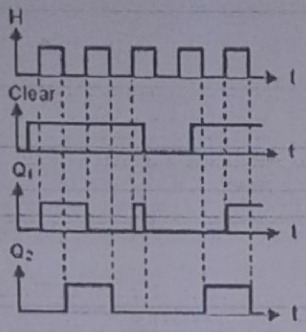


Scanned by CamScanner

CORRECTION

Exercice 1

Toutes les deux bascules sont câblées en bascule T. Elles changent donc constamment d'état si une impulsion d'horloge adéquate arrive sur leur entrée d'horloge (fonctionnement en mode basculement). Les deux bascules reçoivent le même signal d'horloge mais la bascule Q1 est active sur front montant et la bascule Q2 est active sur front descendant. Par ailleurs, les entrées de remise à zéro sont toutes reliées et actives au niveau bas. D'après les chronogrammes de l'horloge et de la remise à zéro (clear) on a :



Exercice 2

a/ Table de vérité du montage 1

E_1	Q_1^+
0	Q_1
1	$\overline{Q_1}$

Table de vérité du montage 2

E_2	Q_2	Q_2^+	
0	0	0	Si $E_2 = 0$ alors quelque soit Q_2 on a $Q_2^+ = 0$
0	1	0	
1	0	1	Si $E_2 = 1$ alors on a $Q_2^+ = \overline{Q_2}$
1	1	0	

On déduit la table de vérité réduite du montage 2 comme suit :

E_2	Q_2^+
0	0
1	$\overline{Q_2}$

Q_1^+ et Q_2^+ sont les états des sorties à l'arrivée d'une impulsion d'horloge active (front montant).

b/ D'après les tables de vérité, le montage 2 est le plus adapté car si $E_2 = 0$ alors $Q_2^+ = 0$ en permanence et $E_2 = 1$ alors $Q_2^+ = \overline{Q_2}$ donc change constamment d'état (d'où le signal carré) aux fronts montants du signal d'horloge.

Par contre pour le montage 1, si $E_1 = 1$ alors $Q_1^+ = \overline{Q_1}$, on a bien la production d'un signal carré ; mais si $E_1 = 0$ on a $Q_1^+ = Q_1$; c'est-à-dire qu'il y a conservation de l'état antérieur de la sortie : elle peut dans ce cas être soit en permanence à 1 soit en permanence à 0.

Exercice 3

a/ Pour avoir $(S_2 ; S_1) = (0 ; 0)$, il faut appliquer sur l'entrée reset un niveau bas.

b/ Table des séquences :

E_2	E_1	S_2	S_1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Exercice 4

a/ Toutes les bascules sont remises à 0.

b/ Les bascules A et C reçoivent la même horloge I tandis que la bascule B reçoit comme horloge la sortie Q_A de la bascule A. Le montage est donc à fonctionnement asynchrone. On obtient par analyse la table d'implication séquentielle suivante :

N° de l'impulsion I	Etat antérieur des sorties			Etat des entrées						Etat postérieur des sorties		
	Q_C	Q_B	Q_A	S_C	R_C	$J_B=K_B$	J_A	K_A		Q_C^+	Q_B^+	Q_A^+
1	0	0	0	0	0	1	1	1		0	0	1
2	0	0	1	0	0	1	1	1		0	1	0
3	0	1	0	0	0	1	1	1		0	1	1
4	0	1	1	1	0	1	1	1		1	0	0
5	1	0	0	0	1	1	0	1		0	0	0

c/ Les sorties des bascules évoluent de manière cyclique (indiqué par la flèche).
Lorsqu'on considère Q_C comme le MSB, les différents états d'un cycle d'évolution sont : 0, 1, 2, 3, 4.

Exercice 5

Lorsque C = 0, le buffer B_1 est transparent et le buffer B_2 est verrouillé. On a dans ce cas $R = \overline{(I/O)}$ et $S = (I/O)\overline{T}$.

Ainsi :

- Si T = 0 alors $R = \overline{(I/O)}$ et $S = (I/O) \Rightarrow Q = (I/O)$: écriture de la donnée dans la bascule.
- Si T = 1 alors R = 0 et S = 0 $\Rightarrow Q^+ = Q = (I/O)$: mémorisation de la donnée copiée.

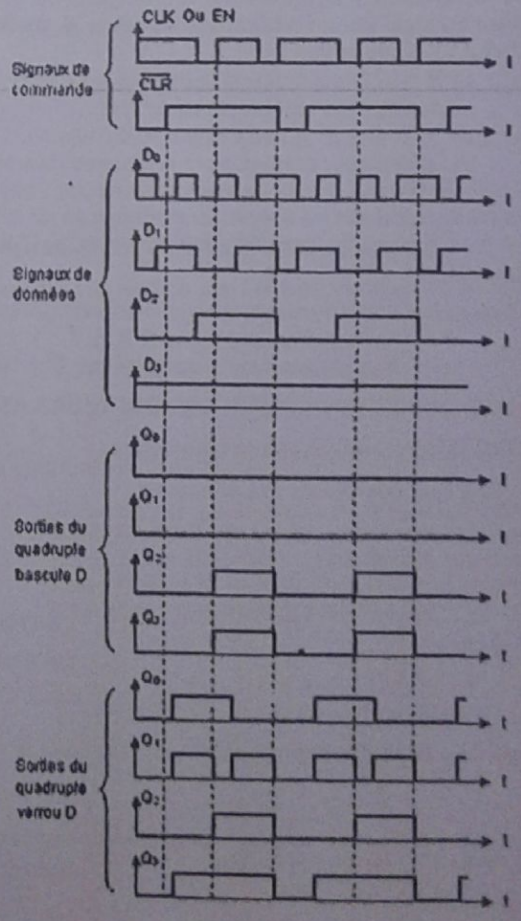
Lorsque C = 1, B_1 est verrouillé et B_2 transparent.

- Si T = 1 alors R = 0 et S = 0 $\Rightarrow$ la donnée copiée reste inchangée.
  Puisque B_2 est transparent, on obtient sur la ligne le contenu de la bascule : lecture du contenu de la bascule.
- Si T = 0 alors les entrées R et S deviennent instables car B_1 étant verrouillé, aucune information n'est disponible sur sa sortie. Donc les entrées des portes qui sont reliées à la sortie de B_1 sont flottantes. Par contre pendant ce temps la ligne est maintenue à 0.
  On peut résumer le fonctionnement du système dans le tableau ci-dessous :

C	T	Etat des buffers B1	Etat des buffers B2	R	S	Q	commentaire
0	0	Transparent	verouillé	$\overline{(I/O)}$	(I/O)	(I/O)	Ecriture
0	1	Transparent	verouillé	0	0	(I/O)	Mémorisation
1	1	verouillé	Transparent	0	0	(I/O)	Lecture
1	0	verouillé	Transparent	$\overline{x}$	x	x	Instabilité

NB : x : état inconnu

Exercice 6



Scanned by CamScanner

CHAPITRE IX – COMPTEURS

I- CLASSIFICATION DES COMPTEURS

Un compteur est un circuit séquentiel qui permet de dénombrer des impulsions appliquées sur son entrée d'horloge et de restituer sur ces sorties des informations binaires évoluant le plus souvent de manière cyclique.
L'élément de base pour la synthèse d'un compteur est une bascule.
L'état d'un compteur est défini par l'ensemble des sorties des bascules qui le constituent. Ainsi, un compteur peut être caractérisé par le nombre d'états (ou de séquences) que donne ses sorties.
On distingue principalement deux critères de classification des compteurs :

- Le mode de commande
- Le mode d'évolution

I-1- Classification suivant le mode de commande

I-1-1- Compteurs asynchrones

Ce sont des compteurs pour lesquels les bascules qui les constituent ne reçoivent pas le même signal d'horloge :

- Le signal d'horloge de commande est appliqué sur la bascule de plus faible poids (LSB).
- Les bascules sont placées en cascade telle que l'entrée d'horloge de la bascule suivante est connectée à la sorite Q ou $\overline{Q}$ de la précédente (selon que les bascules fonctionnent sur front montant ou descendant).

N.B : *Le comptage asynchrone est basé sur la propriété trigger des bascules. C'est-à-dire que la bascule doit être câblée de sorte qu'à l'arrivée du front d'horloge adéquat, sa sortie prenne l'état contraire de l'état antérieur. Autrement dit, dans un compteur asynchrone toutes les bascules utilisées doivent être câblées en bascule T.*

I-1-2- Compteurs synchrones

Contrairement aux compteurs asynchrones, les compteurs synchrones sont conçus de telle sorte que toutes les bascules soient commandées par le même signal d'horloge. Ce qui assure la réaction simultanée de toutes les bascules.

I-2- Classification suivant le mode d'évolution

On distingue :

- Les compteurs qui s'incrémentent à chaque coup d'horloge ; on les appelle tout simplement **"compteurs"**.
- Les compteurs qui se décrémentent à chaque coup d'horloge ; on les appelle **"décompteurs"**.
- Les compteurs qui ont la possibilité d'évoluer en compteur ou en décompteur ; on les appelle **"compteurs réversibles"**.
- Les compteurs qui décrivent un cycle d'états (ou séquences) quelconques ; on les appelle **"automates finis"**.

I-3- Constitution d'un compteur

Qu'il soit asynchrone ou synchrone, un compteur est défini par ses séquences (ou états) de comptage.

- Un compteur (ou un décompteur) **modulo N**, de façon classique, est un compteur qui compte de **0** à **N-1** (ou qui décompte de **N-1** à **0**). Dans ce cas :
  - Si $N = 2^n$ (c'est-à-dire si le modulo est une puissance exacte de 2), on dit que le compteur (ou le décompteur) est à **cycle complet**. Pour réaliser un tel compteur, il nécessite **n bascules**.
  - Si $2^n < N \leq 2^m$ (c'est-à-dire si le modulo n'est pas une puissance exacte de 2), on dit que le compteur (ou le décompteur) est à **cycle incomplet**. Pour réaliser un tel compteur, il nécessite **m bascules**.

  Lorsque les séquences de comptage ne sont pas **consécutives**, ou n'évoluent pas de **0** à **N-1**, le cycle de comptage est entièrement défini ou donné. Le modulo d'un tel compteur est le nombre **d'états qu'il y a dans un cycle**. Dans ce cas si **G** est le plus grand état dans le cycle, on effectue l'encadrement $2^n < G \leq 2^m$. Pour réaliser un tel compteur, il nécessite **m bascules**.

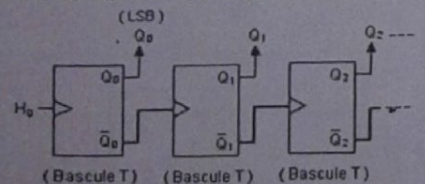
  Exemple : soit à réaliser un compteur décrivant les nombres premiers compris entre 0 et 20. Le cycle décrit par ce compteur est constitué des états 2, 3, 5, 7, 11, 13, 17, 19. Il est de modulo 8 car il y a 8 états dans son cycle de comptage et il se réalise avec 5 bascules car $2^4 < 19 \leq 2^5$.

II- ETUDE DES COMPTEURS ET DECOMPTEURS ASYNCHRONES

II-1- Différentes configurations classiques

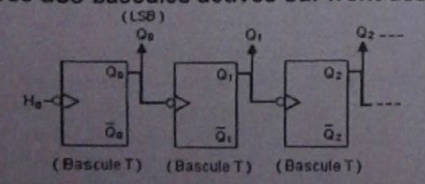
II-1-1- Cas des compteurs asynchrones

a- Avec des bascules actives sur front montant



L'entrée horloge (H_{n+1}) de la bascule de rang n+1 reçoit le signal issu de la sortie complémentée ($\overline{Q}_n$) de la bascule de rang n : ($H_{n+1} = \overline{Q}_n$)

b- Avec des bascules actives sur front descendant

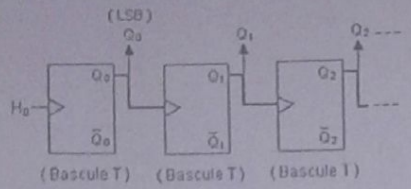


L'entrée horloge (H_{n+1}) de la bascule de rang n+1 reçoit le signal issu de la sortie directe (Q_n) de la bascule de rang n : ($H_{n+1} = Q_n$)

Scanned by CamScanner

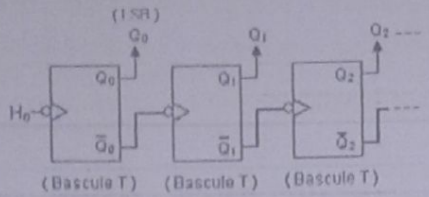
II-1-2- Cas des décompteurs asynchrones

a- Avec des bascules actives sur front montant



L'entrée horloge (H_{n+1}) de la bascule de rang n+1 reçoit le signal issu de la sortie directe (Q_n) de la bascule de rang n : ($H_{n+1} = Q_n$)

b- Avec des bascules actives sur front descendant



L'entrée horloge (H_{n+1}) de la bascule de rang n+1 reçoit le signal issu de la sortie complémentée ($\overline{Q}_n$) de la basculo do rang n : ($H_{n+1} = \overline{Q}_n$)

Remarque :

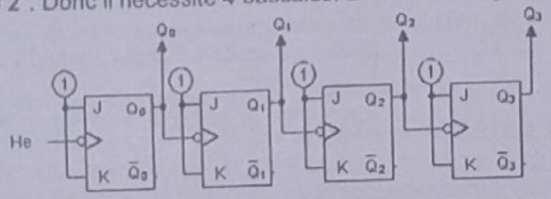
- *Lorsqu'on réalise un compteur dont les sorties utiles sont les Q_i, on obtient automatiquement un décompteur décrivant le même cycle en inverse, si on considère les sortles $\overline{Q}_i$.*
- *Lorsqu'on réalise un décompteur dont les sorties utiles sont les sorties Q_i, on obtient automatiquement un compteur décrivant le même cycle en inverse, si on considère les sortles $\overline{Q}_i$.*

II-2- Réalisation discrète do compteurs asynchrones à cycle complet

Un compteur est dit à cycle complet quand son modulo $N = 2^n$ et que le cycle qu'il décrit contient les états de 0 à N-1.

Exemple : Soit à réaliser un compteur asynchrone modulo 16 avec des bascules JK actives sur front descendant.

Nombre de bascules nécessaires : $16 = 2^4$. Donc il nécessite 4 bascules. D'où le montage ci-dessous.



Remarque :

- Un compteur modulo 16 est aussi appelé **compteur binaire 4 bits** car il décrit naturellement la suite binaire pure de 4 bits de 0 à 15
- Pour un compteur à cycle complet, la fréquence du signal sur une sortie Q_i quelconque est :

$$F_{Qi} = \frac{F_e}{2^{i+1}}$$ Avec Fe : La fréquence du signal d'entrée.

i = 0, 1, 2, 3, ...: les positions des bascules

II-3- Réalisation discrète de compteurs asynchrones à cycle incomplet

Un compteur est dit à cycle incomplet quand il évolue de 0 à N-1 tel que le modulo N soit $2^n < N \leq 2^m$

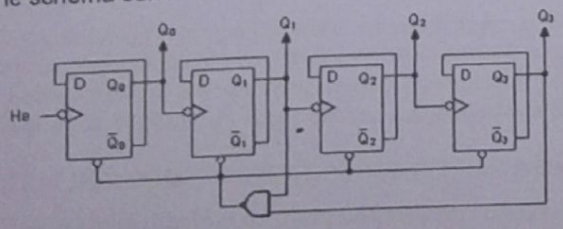
Exemple : Soit à réaliser un compteur asynchrone modulo 10 avec des bascules D actives sur front descendant et à entrée de forçage active au niveau bas.

Nombre de bascules nécessaires : $2^3 < 10 \leq 2^4$. Donc il nécessite 4 bascules.

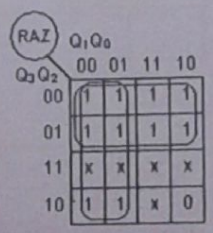
Méthode de construction :

- On réalise un compteur modulo 2^m (ici $2^4 = 16$)
- On utilise les sorties à 1 quand l'état 10 (soit en binaire 1010) est présent pour remettre à zéro les bascules.

D'où le schéma suivant :



On remarquera que l'équation de la remise à zéro peut se déterminer à l'aide d'un tableau de Karnaugh. En effet, sur les RAZ on doit avoir un niveau 1 tant que l'état 10 (c'est-à-dire en binaire 1010) n'est pas en sortie du compteur.



$RAZ = \overline{Q}_3 + \overline{Q}_1 \Rightarrow RAZ = \overline{Q_3 Q_1}$

Remarque :

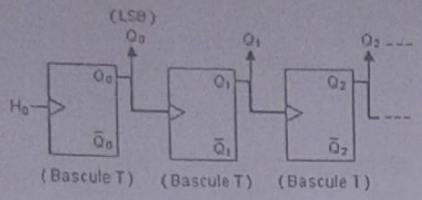
- Un compteur modulo 10 qui compte la suite binaire pur de 0 à 9 est aussi appelé **compteur décade** ou **compteur BCD**.
- La fréquence du signal sur la sortie Q_M de plus fort poids est :

$$F_{QM} = \frac{F_e}{\text{modulo}}$$ Avec F_e la fréquence du signal d'entrée

Scanned by CamScanner

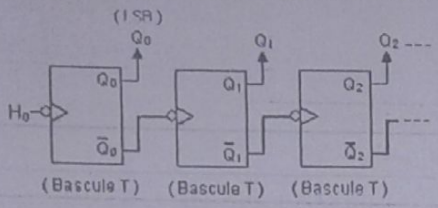
II-1-2- Cas des décompteurs asynchrones

a- Avec des bascules actives sur front montant



L'entrée horloge (H_{n+1}) de la bascule de rang n+1 reçoit le signal issu de la sortie directe (Q_n) de la bascule de rang n : ($H_{n+1} = Q_n$)

b- Avec des bascules actives sur front descendant



L'entrée horloge (H_{n+1}) de la bascule de rang n+1 reçoit le signal issu de la sortie complémentée ($\overline{Q}_n$) de la bascule de rang n : ($H_{n+1} = \overline{Q}_n$)

Remarque :

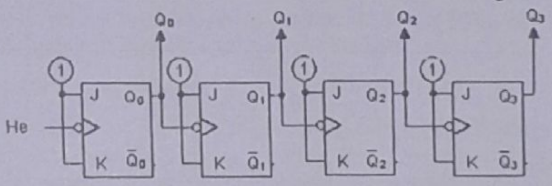
- *Lorsqu'on réalise un compteur dont les sorties utiles sont les Q_i, on obtient automatiquement un décompteur décrivant le même cycle en inverse, si on considère les sorties $\overline{Q}_i$.*
- *Lorsqu'on réalise un décompteur dont les sorties utiles sont les sorties Q_i, on obtient automatiquement un compteur décrivant le même cycle en inverse, si on considère les sorties $\overline{Q}_i$.*

II-2- Réalisation discrète de compteurs asynchrones à cycle complet

Un compteur est dit à cycle complet quand son modulo **N = 2^n** et que le cycle qu'il décrit contient les états de **0 à N-1**.

Exemple : Soit à réaliser un compteur asynchrone modulo 16 avec des bascules JK actives sur front descendant.

Nombre de bascules nécessaires : **16 = 2^4**. Donc il nécessite **4** bascules. D'où le montage ci-dessous.



Remarque :

- Un compteur modulo 16 est aussi appelé **compteur binaire 4 bits** car il décrit naturellement la suite binaire pure de 4 bits de 0 à 15
- Pour un compteur à cycle complet, la fréquence du signal sur une sortie Q_i quelconque est :

$$F_{Qi} = \frac{F_e}{2^{i+1}}$$ Avec Fe : La fréquence du signal d'entrée.

i = 0, 1, 2, 3, ...: les positions des bascules

II-3- Réalisation discrète de compteurs asynchrones à cycle incomplet

Un compteur est dit à cycle incomplet quand il évolue de **0 à N-1** tel que le **modulo N** soit **$2^n < N \leq 2^m$**

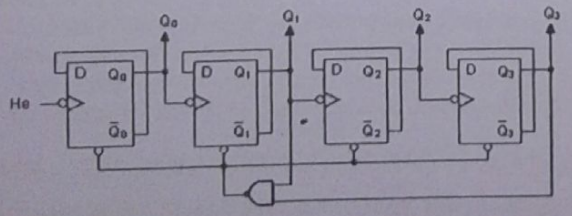
Exemple : Soit à réaliser un compteur asynchrone modulo 10 avec des bascules D actives sur front descendant et à entrée de forçage active au niveau bas.

Nombre de bascules nécessaires : **$2^3 < 10 \leq 2^4$**. Donc il nécessite **4** bascules.

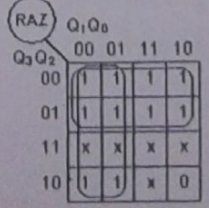
Méthode de construction :

- On réalise un compteur modulo 2^m (Ici $2^4 = 16$)
- On utilise les sorties à 1 quand l'état 10 (soit en binaire **1010**) est présent pour remettre à zéro les bascules.

D'où le schéma suivant :



On remarquera que l'équation de la remise à zéro peut se déterminer à l'aide d'un tableau de Karnaugh. En effet, sur les RAZ on doit avoir un niveau 1 tant que l'état 10 (c'est-à-dire en binaire **1010**) n'est pas en sortie du compteur.



$RAZ = \overline{Q}_3 + \overline{Q}_1 \Rightarrow RAZ = \overline{Q_3 Q_1}$

Remarque :

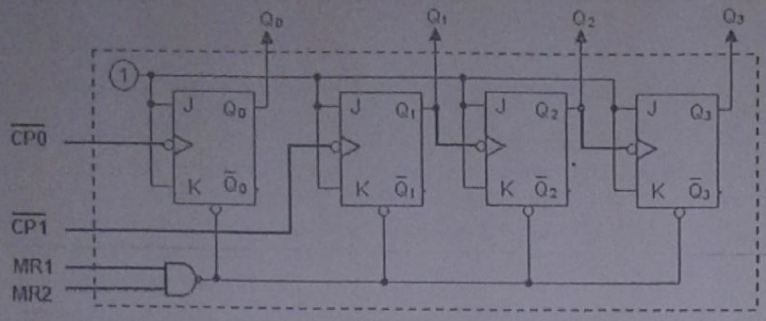
- Un compteur modulo 10 qui compte la suite binaire pur de 0 à 9 est aussi appelé **compteur décade** ou **compteur BCD**.
- La fréquence du signal sur la sortie Q_M de plus fort poids est :

$$F_{QM} = \frac{F_e}{modulo}$$ Avec F_e la fréquence du signal d'entrée

Scanned by CamScanner

II-4- Réalisation de compteurs asynchrones avec un circuit intégré : le 74293

II-4-1- Structure interne du circuit intégré 74293



MR : Master Reset ou Main Reset
CP0 , CP1: Clock Pulse 0, Clock Pulse 1

Symbole logique

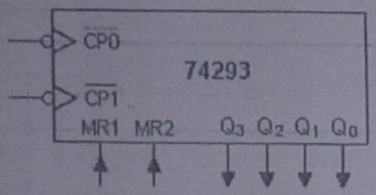


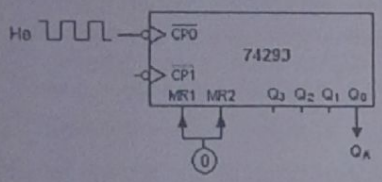
Table de fonctionnement

Entrées				Réponse
MR1	MR2	CP0	CP1	
H	H	X	X	$Q_3=Q_2=Q_1=Q_0=0$: RAZ
H	L	↓	X	(Q_0) : compteur par 2
L	H	X	↓	(Q_3, Q_2, Q_1) : compteur par 8
L	L	↓	↓	(Q_0) : compteur par 2 et (Q_3, Q_2, Q_1) : compteur par 8
		↓	Q_0	(Q_3, Q_2, Q_1, Q_0) : compteur par 16

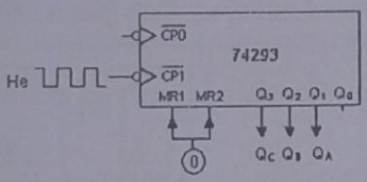
II-4-2- Utilisation du 74293 pour la réalisation de compteurs divers

a- Réalisation des trois modulos triviaux du 74293

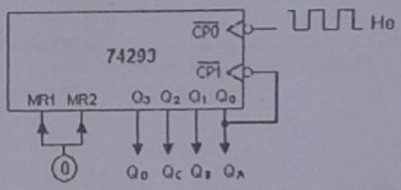
Compteur modulo 2



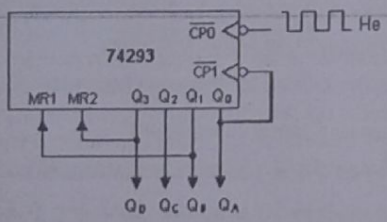
Compteur modulo 8



Compteur modulo 16



b- Compteur modulo 10 (ou compteur décade)

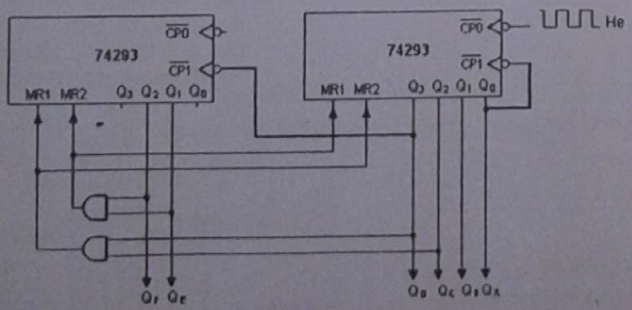


c- Compteur modulo 60

On peut compter suivant trois modes d'évolution en général :

- **comptage par le mode d'évolution en binaire pur**
  Pour ce mode d'évolution les séquences dans le cycle de comptage évoluent selon le code binaire pur.
  Dans ce cas, dans le code binaire pur de 60, on utilisera les bits à 1 pour effectuer la remise à zéro des bascules.

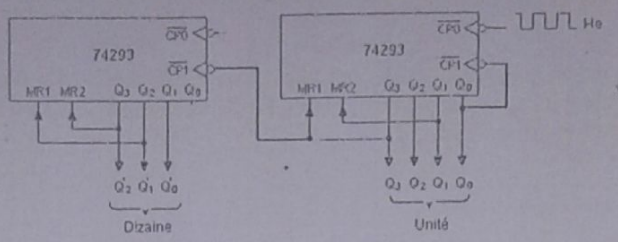
  On a $(60)_{10} = (111100)_2$. Il nécessite 6 bascules ; soit alors deux circuits intégrés.



Scanned by CamScanner

- Comptage par le mode d'évolution en décade (mode BCD)

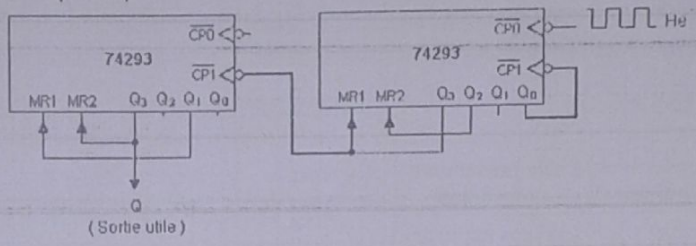
60 = 6x10. La réalisation nécessite donc 2 décades : modulo10 (unité) et modulo 6 (dizaine).



On adopte ce mode d'évolution quand le résultat du comptage est destiné à être affiché sur des afficheurs 7 segments.

- **Comptage par le mode d'évolution quelconque pour la division de fréquence**

On utilise ce mode d'évolution pour diviser la fréquence du signal d'horloge. Il n'est donc pas nécessaire que le compteur dénombre une suite binaire utile (binaire pur ou BCD) tant que le modulo recherché est réalisé.
Par exemple modulo 60 = modulo (5 x 12)



Remarque : On utilise souvent un compteur pour diviser un signal d'horloge de haute fréquence afin d'obtenir un signal de basse fréquence.
Quand on utilise un compteur essentiellement pour la division de fréquence, il n'est pas nécessaire qu'il évolue selon une suite binaire utile (évolution en binaire pur ou en BCD par exemple). On décomposera le modulo en un produit de plusieurs modulos à mettre en cascade tel qu'on ait à utiliser moins de composants extérieurs. Si en particulier l'un de ces modulos est une puissance exacte de 2 *(mais pour le circuit 74293, un de ses modulos triviaux)* alors en récupérant la sortie utile sur ce modulo, on obtient une onde carrée symétrique. C'est-à-dire un signal de rapport cyclique ½.

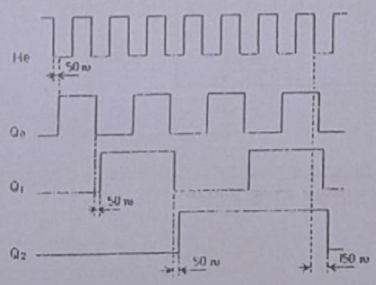
II-5- Temps de propagation du signal dans un compteur asynchrone

Dans les compteurs asynchrones, les bascules ne commutent pas simultanément. Chaque bascule est déclenchée par la transition de la sortie de la bascule qui la précède. Comme tout circuit logique, une bascule ne réagit pas immédiatement à un changement sur son entrée d'horloge. Il existe un temps de transfert (quelques nano secondes) entre l'arrivée du signal d'horloge et la réponse à la sortie de la bascule. IL se produit alors des délais de propagation où temps de propagation (T_p) d'une bascule à l'autre. Ainsi la $n^{ième}$ bascule ne réagit que $n.T_p$ unités de temps après l'arrivée de l'impulsion d'horloge initiale. Dans ce cas pour avoir un bon fonctionnement du compteur, il faut que la période T_e du signal d'horloge soit beaucoup plus grande que le retard de propagation total du compteur.

Soit alors $Te \geq n \cdot Tp \Rightarrow Fe \leq \dfrac{1}{n \cdot Tp}$

On définit alors la fréquence maximale du signal d'horloge $Femax = \dfrac{1}{n \cdot Tp}$ n : nombre de bascules mises en cascade.

Exemple illustratif du temps de propagation.



Lorsqu'elle change d'état, la sortie Q_2 effectue sa transition 150 ns après l'arrivée de l'impulsion d'horloge de commande He.

Remarque :
- Compte tenu du temps de propagation dans les compteurs asynchrones dû à la commande des bascules les unes après les autres, ces compteurs sont aussi appelés **compteurs à propagation.**
- Les compteurs asynchrones, faciles à constituer, sont très utilisés mais leurs emplois trouvent rapidement une limite avec la fréquence d'utilisation. Dans ce cas, l'utilisation des compteurs synchrones est préférable.

III- ETUDE DES COMPTEURS ET DECOMPTEURS SYNCHRONES

III-1- Réalisation discrète de compteurs synchrones

On les réalise avec des bascules D ou JK suivant plusieurs méthodes qui sont :

- La méthode avec des bascules D
- La méthode avec des bascules JK par conditionnement des entrées (méthode universelle)
- La méthode avec des bascules JK par identification de l'équation caractéristique
- La méthode avec des bascules JK câblées pour fonctionner en bascule T.

Scanned by CamScanner