لجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieure et la Recherche Scientifique

جامعة 20 أوت 1955 –سكيكدة–

Université de 20 Aout 1955 –Skikda-

كليةالتكنولوجيا

Faculté de Technologie

قسم التكنلوجيا

Département de Technologie

Support de Cours pour la Filière du Groupe « A » (2ème Année ST)

Logique Combinatoire et Séquentielle

Préparé Par : Dr INEL FOUAD

SOMMAIRE

V. Chapitre V

CIRCUIT SEQUENTIEL 'BASCULES'

VI. Chapitre VI

CIRCUIT SEQUENTIEL 'COMPTEUR'

Chapitre I

Les systèmes de numérations et Codage de L'information

1. Généralités

Presque tous les ordinateurs ne font pas le calcul en base 10. La base 10 est le plus souvent utilisée par les hommes, il faut apprendre à convertir les nombres d'une base de données à une autre.

Par définition est l'action ou la manière de représenter les chiffres. Il existe plusieurs systèmes de numération :

- Système décimal
- Système binaire
- Système octal
- Système hexadécimal

Ces systèmes sont généralement utilisés dans la programmation, pour la communication entre l'ordinateur et son utilisateur. Dans la base donnée α un nombre peut être exprimé sous la forme d'un polynôme. Supposons le nombre suivant X = ($a_n a_{n-1}$ … a_0) la forme pronominale de ce nombre est $X = \sum_{i=0}^{n} ai\alpha^n$

a_i : Coefficient
α : la base

Exemple :

Soit X = 2955 sa forme pronominale est : $X = 2x10^3 + 9x10^2 + 5x10^1 + 5x10^0$.

1.1 Le système décimal ou base 10

C'est le système le plus connu et le plus utilisé par les hommes. Il est constitué de 10 symboles ou chiffres appelés digits. Ces symboles sont : 0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9

Tout nombre de ce système s'écrit :

$(N)_{10} = a_n 10^n + a_{n-1} 10^{n-1} + \ldots\ldots\ldots\ldots + a_1 10^1 + a_0 10^0$.

Exemple:$(329)_{10} = 3*10^2 + 2*10^1 + 9*10^0$.

1.2 Le système binaire ou base 2

C'est un système composé de 2 symboles 0 et 1 encore appelé digits binaire (binary digit) d'où le nom bit. Ce système est à la base du langage machine (c'est le seul langage compréhensible par l'ordinateur).

$(N)_2 = a_n 2^n + a_{n-1} 2^{n-1} + \ldots\ldots\ldots\ldots + a_1 2^1 + a_0 2^0$.

Exemple: :$(1010)_2 = 1*2^3+ 0*2^2+ 1*2^1+ 0*2^0$.

1. 3 Le système octal ou base 8

C'est un système intermédiaire constitué de 8 symboles qui sont : 0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ;7 .

$(N)_{10} = a_n 8^n + a_{n-1} 8^{n-1} + \ldots\ldots\ldots\ldots + a_1 8^1 + a_0 8^0$.

Exemple :$(325) : 3*8^2+2*8^1 +5*8^0=(69)_{10}$.

1.4 Le système hexadécimal ou base 16

Il utilise la base 16 et les chiffres de 0 à 15,mais comme les chiffres de 10 à 15 posent confusion avec les chiffres de 10 à 15 on les remplaçants par les lettres de A à F

A = 10 ; B = 11 ; C = 12 ; D = 13 ; E = 14 ; F = 15 .

$(N)_{16} = a_n 16^n + a_{n-1} 16^{n-1} + \ldots\ldots\ldots\ldots + a_1 16^1 + a_0 16^0$.

Exemple: $(A29) = A*16^2+2*16^1 +9*16^0=(2601)_{10}$.

2. CONVERSION ENTRE LES SYSTEMES DE NUMERATION

2.1. conversion d'un nombre de base(p) en un nombre décimal :

La valeur décimale d'un nombre « N » écrit dans une base « p » s'obtient par sa forme générale :

$(N)_p=(a_n\ a_{n-1}\ldots a_0)_p = a_n*p^n + a_{n-1}*p^{n-1} +\ldots+ a_0*p^0$.

2.1.1. conversion binaire – décimal:

- $(1100101)_2 = (\ldots)_{10}$

$2.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 64 + 32 + 0 + 0 + 4 + 0 + 1 = (101)_{10}$

2.1.2. conversion octal – décimal:

- 2) $(203)_8 = (\ldots)_{10}$

$2.8^2 + 0.8^1 + 3.8^0 = 128 + 3 = (131)_{10}$

2.1.3. conversion hexadécimal – décimal:

- $(76)_{16} = (\ldots)_{10}$

$7.16^1 + 6.16^0 = 112 + 6 = (118)_{10}$

2.2. conversion d'un nombre décimal en un nombre de base(p):

L'algorithme de conversion est le suivant :

1. Diviser (division entière) le nombre décimal par la base « P » ou premier lieu.
2. Prendre le résultat de la division, et le diviser de nouveau par la base « P ».
3. Reprendre l'étape 2 jusqu'à obtient un résultat nulle.
4. Le nouveau nombre équivalant s'obtient en prenant les restes de la division de la dernière à la première, en écrivons de gauche à droite.

2.2.1.Conversion d'un nombre en base 2.

$(18)_{10} = (\ldots)_2$

18 :2 = 9 et reste = 0
9 :2 = 4 et reste = 1
4 :2 = 2 et reste = 0
2 :2 = 1 et reste = 0
1 :2 = 0 et reste = 1

$(18)_{10} = (10010)_2$

2.2.2. Conversion d'un nombre en base 8

$(207)_{10} = (\ldots)_8$

207 :8 = 25 et reste = 7
25 :8 = 3 et reste = 1
3 :8 = 0 et reste = 3

$(207)_{10} = (317)_8$

2.2.3. Conversion d'un nombre en base 16

$(128)_{10} = (\ldots)_{16}$

128 :16 = 8et reste = 0
8 :16 = 0 et reste = 8
$(128)_{10} = (80)_{16}$

2.3. CONVERSION DES NOMBRE FRACTIONNAIRES :

Un nombre fractionnaire est un nombre qui contient une partie inférieure à 1.

Exemple : 5/2 ;32 ,125 ,0,63.

2. 3.1.CONVERSION D'UN NOMBRE FRACTIONNAIRE DE BASE(P) EN UN NOMBRE DECIMAL

La valeur décimale d'un nombre fractionnaire « N » écrit dans une base « P » s'obtient par sa forme polynomiale avec les rangs négatifs de façons décroissante.

Exemple:$(18,053)_{16} = 1*16^{1}+8*16^{0}+0*16^{-1} +5*16^{-2}+3*16^{-3}$

$(18,053)_{16} = (24,0839)$

2.3.2.CONVERSION D'UN NOMBRE DECIMAL EN UN NOMBRE FRACTIONNAIRE DE BASE(P)

L'algorithme de conversion consiste à :

- Multiplier le nombre décimal fractionnaire par la base « P » en 1^{ier} lieu.
- Reprendre la partie fractionnaire du résultat et la multiplier de nouveau par la base « P ».
- Reprendre l'étape 2 jusqu'à obtenir une partie fractionnaire nulle, où arrivée à une limite de chiffre après la virgule.
- Le nouveau nombre s'obtient : on prend les parties entières des résultats de multiplications de la première à la dernière on les écrit de gauche à droite.

Exemple $(0,125)_{10}==========(0,001)_2$

2 .4 Conversion Binaire –Octal et vice versa

Les chiffres de système octal sont de zéro à sept, ces chiffres nécessitent trois bits pour être codée en binaire 0—000, 7----111. Donc pour convertir un nombre octal en binaire, il suffit de remplacer chaque chiffre par son équivalant binaire sur trois bits.

Exemple : $(5)_8=(101)_2$

De façon inverse, pour convertir un nombre binaire en octal, on regroupe les bits en groupe de trois (03) de droite à gauche pour la partie entier (on ajoute des zéros à gauche si c'est nécessaire) et de gauche à droite pour la partie fractionnaire (on ajoute des zéros à droite si c'est nécessaire) , en suite on remplace chaque groupe par son équivalant octal.

Exemple: $(101)_2=(5)_8$

2 .5 Conversion Binaire –hexadécimal et vice versa

Les chiffres de système hexadécimal sont de zéro à neuf et de A à F , ces chiffres nécessitent quatre bits pour être codée en binaire 0—0000, F----1111. Donc pour convertir un nombre hexadécimal en binaire, il suffit de remplacer chaque chiffre par son équivalant binaire sur quatre bits.

Exemple : $(1A1)_{16}=(0001\ 1010\ 0001)_2$

De façon inverse, pour convertir un nombre binaire en hexadécimal, on regroupe les bits en groupe de quatre (04) de droite à gauche pour la partie entier (on ajoute des zéros à gauche si c'est nécessaire) et de gauche à droite pour la partie fractionnaire (on ajoute des zéros à droite si c'est nécessaire) , en suite on remplace chaque groupe par son équivalant hexadécimal.

Exemple: $(0111)_2==========(7)_{16}$

3. OPERATION ARITHMETIQUE DE BASE

1. Addition binaire

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 et on retient 1
1 + 1+1 = 1et on retient 1

2. Soustraction binaire

0 – 0 = 0
0 – 1 = 1 et on retient 1
1 – 0 = 1
1 – 1 = 0
0 – 1 – 1 = 0 et on retient 1
1– 1 – 1 = 1 et on retient 1

3. Multiplication binaire :

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

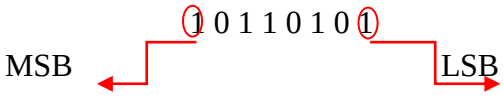
4. Division binaire :

0/1=0
1/1=1

4. Le codage de l'information

4.1. Généralités

Les processeurs sont des systèmes automatiques de traitement des informations. Ils manipulent ces informations sous forme de données binaires (groupe de bits).
Notre système conventionnel de comptage en base 10 est donc incompatible avec les processeurs et nous oblige à utiliser d'autres systèmes de numération. Nous coderons donc les signaux numériques (images, sons, textes, ...) sous forme de "0" et de "1" compréhensibles par un processeur.
Le "bit" est l'abréviation de "binary digit". Un bit peut prendre la valeur 0 ou 1
Le BIT de poids le plus faible (le plus à droite) s'appelle le LSB (Least Significant Bit)
Le BIT de poids le plus fort (le plus à gauche) s'appelle le MSB (Most Significant Bit)



Remarque

Un nombre binaire se terminant par 1 est donc un nombre IMPAIR alors qu'un nombre binaire se terminant par 0 est PAIR.

- Un groupe de 4 bits s'appelle un quartet
-Un groupe de 8 bits est nommé un octet ou un byte
- Un groupe de 16 bits s'appelle un mot ou un word
- Un groupe de 32 bits s'appelle un mot long ou un double word

4.2. LES CODES BINAIRES

4.2.1.Code binaire naturel

C'est le code le plus simple, il ne représente que des chiffres positifs.

Le tableau suivant donne le code binaire naturel pour un exemple d'un mot de 4 bits.

Tableau 1 : Code binaire naturel

Valeurs décimales	Code en binaire Naturel 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

4.2.2.Code Binaire en BCD « Décimal Codé Binaire » :

le code BCD est le code le plus utilisés. Il permet de codé chaque nombre décimal sur 4 bits. il représente les 10 chiffres décimaux (0, …,9), mais il y a 6 configurations non utilisable. Aussi ce code appelé « le code 8 4 2 1 ».

Exemple : $(271)_{10}$= 0010 0111 0001

2 7 1

Tableau 2 : Code décimal binaire

Valeurs décimales	Code en BCD 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

Remarque :

Les notations (1010,1011,1100,1101,1110,1111) sont interdites en BCD.

Addition en BCD :

- Si un nombre est >= 10, on ajoute 0110 pour trouver le résultat correcte.

Problématique :

Dans le code binaire naturel plusieurs variables peuvent changer d'état en même temps. (Exemple : pour passer de la valeur 3 à la valeur 4, 3 variables changent d'état simultanément). Dans un système automatise, il est impossible que deux variables changent au même moment.

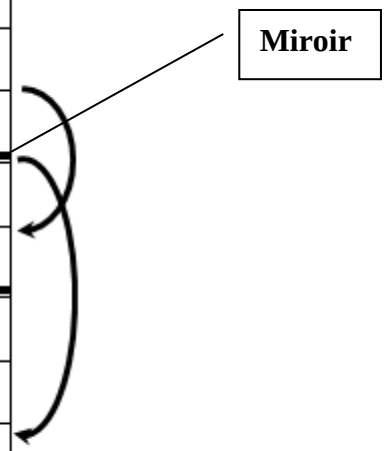
4.2.3.Code binaire réfléchi ou code GRAY :

Dans ce code, il n'y a qu'une variable qui change d'état à la fois entre deux valeurs consécutives.

II est nommé réfléchi car il faut recopier les valeurs comme si elles étaient réfléchies dans un miroir. On trouve ce code dans les roues codeuses, les capteurs numériques de position, etc.

Tableau 3 : Code binaire réfléchi

Valeurs Décimales	Code en binaire réfléchi 2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1

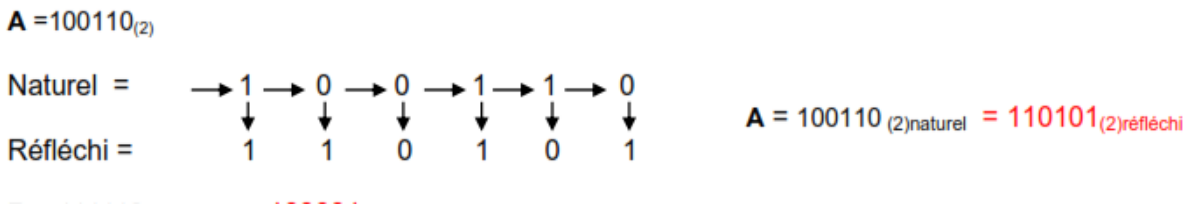


- si le nombre de 1 est pair, il faut inverser le dernier chiffre.
- si le nombre de 1 est impair, il faut inverser le chiffre situé à gauche du 1 le plus à droite.

4.3 Conversion binaire – gray et vice versa :

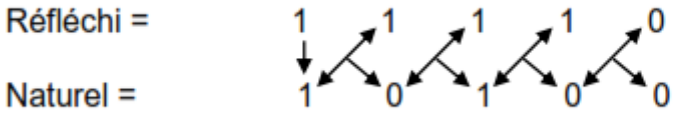
- le bit de gauche du code gray est le même que le bit du code binaire.
- Ajoute le MSB du binaire à son voisin de droite, et reporté la somme (on négligeant la retenue) sur la ligne inférieur du code gray.
- Continue l'addition des bits a leurs voisins de droite et reporter les sommes obtenu jusqu'à atteindre le LSB.
- Si le bit bn+1 et le bit binaire naturel ont même valeur, le chiffre correspondant BR est 0.
- Si le bit bn+1 et le bit binaire naturel des valeurs différentes, le chiffre correspondant BR est 1.

exemple :



4.4 Conversion gray - binaire :

- L'addition ce fait avec le MSB de binaire et son voisin de code gray.
- Même règle que la conversion binaire gray.
- Reproduire le chiffre du poids plus fort (chiffre à gauche)
- Comparer le chiffre de rang n+1 du binaire naturel à de rang n du binaire réfléchi (on écrit 1 s'ils sont différents si non on écrit 0).



4.5. code ASCII (*American Standard Code for Information Interchange*)

La mémoire de l'ordinateur conserve toutes les données sous forme numérique. Il n'existe pas de méthode pour stocker directement les caractères. Chaque caractère possède donc son équivalent en code numérique : c'est le **code ASCII** (*American Standard Code for Information Interchange* - traduisez « Code Americain Standard pour l'Echange d'Informations »). Le code ASCII de base représentait les caractères sur 7 bits (c'est-à-dire 128 caractères possibles, de 0 à 127).

- Les codes 0 à 31 ne sont pas affichables (ne sont pas des caractères). On les appelle *caractères de contrôle* car ils permettent de faire des actions telles que :
  - retour à la ligne (CR)
  - Le caractère numéro 32 est l'espace

DEC	OCT	HEX	BIN	Symbol	Description
0	000	00	00000000	NUL	Null char
1	001	01	00000001	SOH	Start of Heading
2	002	02	00000010	STX	Start of Text
3	003	03	00000011	ETX	End of Text
4	004	04	00000100	EOT	End of Transmission
5	005	05	00000101	ENQ	Enquiry
6	006	06	00000110	ACK	Acknowledgment
7	007	07	00000111	BEL	Bell
8	010	08	00001000	BS	Back Space
9	011	09	00001001	HT	Horizontal Tab
10	012	0A	00001010	LF	Line Feed
11	013	0B	00001011	VT	Vertical Tab
12	014	0C	00001100	FF	Form Feed
13	015	0D	00001101	CR	Carriage Return
14	016	0E	00001110	SO	Shift Out / X-On
15	017	0F	00001111	SI	Shift In / X-Off
16	020	10	00010000	DLE	Data Line Escape
17	021	11	00010001	DC1	Device Control 1 (oft. XON)
18	022	12	00010010	DC2	Device Control 2
19	023	13	00010011	DC3	Device Control 3 (oft. XOFF)
20	024	14	00010100	DC4	Device Control 4
21	025	15	00010101	NAK	Negative Acknowledgement
22	026	16	00010110	SYN	Synchronous Idle
23	027	17	00010111	ETB	End of Transmit Block
24	030	18	00011000	CAN	Cancel
25	031	19	00011001	EM	End of Medium
26	032	1A	00011010	SUB	Substitute
27	033	1B	00011011	ESC	Escape
28	034	1C	00011100	FS	File Separator
29	035	1D	00011101	GS	Group Separator
30	036	1E	00011110	RS	Record Separator
31	037	1F	00011111	US	Unit Separator

- Bip sonore (BEL) Les codes 32 à 127 sont communs à toutes les variations de la table ASCII, ils sont appelés caractères imprimables, représentent des lettres, des chiffres, des signes de ponctuation et quelques symboles divers. Vous trouverez presque tous les caractères sur votre clavier. Le caractère 127 représente la commande DEL.
- Les codes 65 à 90 représentent les majuscules
- Les codes 97 à 122 représentent les minuscules

(Il suffit de modifier le 6ème bit pour passer de majuscules à minuscules, c'est-à-dire ajouter 32 au code ASCII en base décimale.)

DEC	OCT	HEX	BIN	Symbol	Description
32	040	20	00100000		Space
33	041	21	00100001	!	Exclamation mark
34	042	22	00100010	"	Double quotes (or speech marks)
35	043	23	00100011	#	Number
36	044	24	00100100	$	Dollar
37	045	25	00100101	%	Procenttecken
38	046	26	00100110	&	Ampersand
39	047	27	00100111	'	Single quote
40	050	28	00101000	(	Open parenthesis (or open bracket)
41	051	29	00101001	)	Close parenthesis (or close bracket)
42	052	2A	00101010	*	Asterisk
43	053	2B	00101011	+	Plus
44	054	2C	00101100	,	Comma
45	055	2D	00101101	-	Hyphen
46	056	2E	00101110	.	Period, dot or full stop
47	057	2F	00101111	/	Slash or divide
48	060	30	00110000	0	Zero
49	061	31	00110001	1	One
50	062	32	00110010	2	Two
51	063	33	00110011	3	Three
52	064	34	00110100	4	Four
53	065	35	00110101	5	Five
54	066	36	00110110	6	Six
55	067	37	00110111	7	Seven
56	070	38	00111000	8	Eight
57	071	39	00111001	9	Nine
58	072	3A	00111010	:	Colon
59	073	3B	00111011	;	Semicolon
60	074	3C	00111100	<	Less than (or open angled bracket)
61	075	3D	00111101	=	Equals
62	076	3E	00111110	>	Greater than (or close angled bracket)
63	077	3F	00111111	?	Question mark
64	100	40	01000000	@	At symbol
65	101	41	01000001	A	Uppercase A
66	102	42	01000010	B	Uppercase B
67	103	43	01000011	C	Uppercase C
68	104	44	01000100	D	Uppercase D
69	105	45	01000101	E	Uppercase E
70	106	46	01000110	F	Uppercase F
71	107	47	01000111	G	Uppercase G
72	110	48	01001000	H	Uppercase H
73	111	49	01001001	I	Uppercase I
74	112	4A	01001010	J	Uppercase J
75	113	4B	01001011	K	Uppercase K
76	114	4C	01001100	L	Uppercase L
77	115	4D	01001101	M	Uppercase M
78	116	4E	01001110	N	Uppercase N
79	117	4F	01001111	O	Uppercase O
80	120	50	01010000	P	Uppercase P
81	121	51	01010001	Q	Uppercase Q
82	122	52	01010010	R	Uppercase R
83	123	53	01010011	S	Uppercase S

84	124	54	01010100	T	Uppercase T
85	125	55	01010101	U	Uppercase U
86	126	56	01010110	V	Uppercase V
87	127	57	01010111	W	Uppercase W
88	130	58	01011000	X	Uppercase X
89	131	59	01011001	Y	Uppercase Y
90	132	5A	01011010	Z	Uppercase Z
91	133	5B	01011011	[	Opening bracket
92	134	5C	01011100	\	Backslash
93	135	5D	01011101	]	Closing bracket
94	136	5E	01011110	^	Caret - circumflex
95	137	5F	01011111	_	Underscore
96	140	60	01100000	`	Grave accent
97	141	61	01100001	a	Lowercase a
98	142	62	01100010	b	Lowercase b
99	143	63	01100011	c	Lowercase c
100	144	64	01100100	d	Lowercase d
101	145	65	01100101	e	Lowercase e
102	146	66	01100110	f	Lowercase f
103	147	67	01100111	g	Lowercase g
104	150	68	01101000	h	Lowercase h
105	151	69	01101001	i	Lowercase i
106	152	6A	01101010	j	Lowercase j
107	153	6B	01101011	k	Lowercase k
108	154	6C	01101100	l	Lowercase l
109	155	6D	01101101	m	Lowercase m
110	156	6E	01101110	n	Lowercase n
111	157	6F	01101111	o	Lowercase o
112	160	70	01110000	p	Lowercase p
113	161	71	01110001	q	Lowercase q
114	162	72	01110010	r	Lowercase r
115	163	73	01110011	s	Lowercase s
116	164	74	01110100	t	Lowercase t
117	165	75	01110101	u	Lowercase u
118	166	76	01110110	v	Lowercase v
119	167	77	01110111	w	Lowercase w
120	170	78	01111000	x	Lowercase x
121	171	79	01111001	y	Lowercase y
122	172	7A	01111010	z	Lowercase z
123	173	7B	01111011	{	Opening brace
124	174	7C	01111100	|	Vertical bar
125	175	7D	01111101	}	Closing brace
126	176	7E	01111110	~	Equivalency sign - tilde
127	177	7F	01111111		Delete

Exemples : % trouver dans le clavier = 0100101 dans le code ASCII en binaire ;+ :0101011.

EXERCICES

Exercice 1

a) Donner la valeur numérique décimale des nombres binaires suivants :
(1100) ;(10001101) ; (101.01) ;(0.1101)

b) Convertir en binaire les nombres décimaux suivants :
(48) ; (11) ; (189) ;(0.25) ;(12.234)

Exercice 2

a) Convertir en décimal puis en binaire les nombres octaux suivants :
(534.72)
8
(15)
8
(52.392)
8

b) Convertir en octal les nombres décimaux suivants :
(77.375)
10
(20.515625)
10
(8.15625)
10

c) Convertir en octal (base 8) les nombres binaires suivants :
(11101)
2
(10010101)
2
(111.001)
2
(11000.1001)
2

Exercice 3

a) Convertir en décimal les nombres hexadécimaux suivants :
(F.4)
16
(D3.E)
16
(1111.1)
16
(EBA.C)
16

b) Convertir en hexadécimal les nombres décimaux suivants :

(204.125)
10
(613.25)
10
(255.875)
10

b) Convertir en hexadécimal les nombres binaires suivants :
(11110010)
2
(10001.11111)
2
(111110.000011)
2

LA SOLUTION

Exercice01

a) la valeur numérique décimale des nombres binaires suivants :

- $(1100)_2 = 1*2^3+1*2^2+0*2^1+0*2^0 = (12)_{10}$
- $(10001101)_2 = (141)_{10}$ avec la même méthode.
- $(101,01)_2 = 1*2^2+0*2^1+1*2^0+0*2^{-1}+0*2^{-2} = (5,25)_{10}$

b) La valeur en binaire des nombres décimaux suivants :

- $(48)_{10}=$

48 :2 = 24 et reste = 0
24 :2 = 12 et reste = 0
12:2 = 6 et reste = 0
6 :2 = 3 et reste = 0
3 :2 = 1 et reste = 1
1 :2 = 0 et reste = 1

La lecture

Le nouveau nombre équivalant s'obtient en prenant les restes de la division de la dernière à la première, en écrivons de gauche à droite.

$(48)_{10}=(110000)_2$

- $(0.25)_{10}=(???)_2$

0.25 0.50
*2 *2
=0.50 =1.00

La lecture

on prend les parties entières des résultats de multiplications de la première à la dernière on les écrit de gauche à droite.

$(0.25)_{10}=(0.01)_2$

- $(12.234)_{10}= (1100,00111011)_2$

Exercice 02

a) La conversion en décimal puis en binaire les nombres octaux suivants :

- $(534.72)_8=(??)_{10}=(??)_2$.

$(534.72)_8=(5*8^2+3*8^1+4*8^0+7*8^{-1}+2*8^{-2})_{10}=(348,90625)_{10}$

$(534.72)_8$= il suffit de remplacer chaque chiffre par son équivalant binaire sur trois bits ; Alors $=(101\ 011\ 100,111\ 010)_2$

b) Conversion en octal les nombres décimaux suivants :

- $(77.375)_{10}=(??)_8$

77/8=9 reste 5
9/8=1 reste 1

La lecture

1/8=0 reste 1

Et 0.375=

0.375
*8
=3.00

Alors $(77.375)_{10}=(115.3)_8$

- $(8.15625)_{10}=(??)_8$

  avec même méthode on a trouvé le résultat suivant :

  $(8.15625)_{10}=(10.12)_8$

c) Conversion en octal (base 8) les nombres binaires suivants :

- $(11101)_2=(??)_8$

on regroupe les bits en groupe de trois (03) de droite à gauche pour la partie entier (on ajoute des zéros à gauche si c'est nécessaire) et de gauche à droite pour la partie fractionnaire (on ajoute des zéros à droite si c'est nécessaire) , en suite on remplace chaque groupe par son équivalant octal.

$(11101)_2=011\ 101=(35)_8$

La lecture

- $(10010101)_2=010\ 010\ 101=(225)_8$
- $(111.001)_2=111.001=(7.1)_8$
- $(11000.1001)_2=011\ 000.100\ 100=(30.44)_8$

Exercice 03

a) Conversion en décimal les nombres hexadécimaux suivants :

- $(F.4)_{16}=15*16^0+4*16^{-1}=(15,25)_{10}$
- $(D3.E)_{16}=13*16^1+3*16^0+14*16^{-1}=(111,875)_{10}$
- $(1111.1)_{16}=1*16^3+1*16^2+1*16^1+1*16^0+1*16^{-1}=(4368,0625)_{10}$

b) Conversion en hexadécimal les nombres décimaux suivants :

- $(204.125)_{10}=(??)_{16}$
  204/16=12 reste 12
  12/16=0 reste 12
  Et 0.125=
  0.125
  *16
  =2

Alors $(204.125)_{10}=(CC,2)_{16}$

- $(613.25)_{10}=(265,4)_{16}$
- $(255.875)_{10}=(FF,E)_{16}$

b) Conversion en hexadécimal les nombres binaires suivants :

on regroupe les bits en groupe de quatre (04) de droite à gauche pour la partie entier (on ajoute des zéros à gauche si c'est nécessaire) et de gauche à droite pour la partie fractionnaire (on ajoute des zéros à droite si c'est nécessaire) , en suite on remplace chaque groupe par son équivalant hexadécimal.

- $(11110010)_2$ $=1111\ 0010 = (F2)_{16}$
- $(10001,11111)_2$ $= 0001\ 0001,1111\ 1000 = (11,F8)_{16}$

  La lecture
- $(111110,000011) = 0011\ 1110,\ 0000\ 1100 = (3D,0C)_{16}$

  La lecture

CHAPITRE II

ALGEBRE DE BOOLE ET SIMPLIFICATIONS DES FONCTIONS LOGIQUE

1. DEFINITION

L'algèbre de Boole, ou calcul booléen, est la partie des mathématiques qui s'intéresse aux opérations et aux fonctions sur les variables logiques. Elle fut inventée par le mathématicien britannique George **Boole (1815-1864)**. Aujourd'hui, l'algèbre de Boole trouve de nombreuses applications en informatique et dans la conception des circuits électroniques.

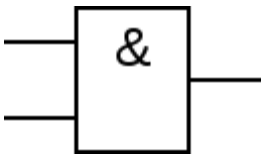
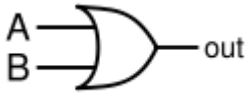
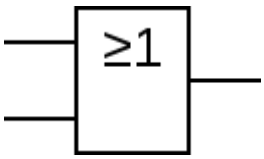
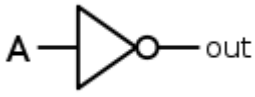
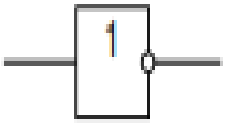
On appelle B l'ensemble constitué de deux élément s'appelés valeurs de vérité {FAUX,VRAI}. Cet ensemble est aussi noté $B = \{0, 1\}$.

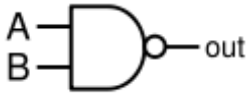
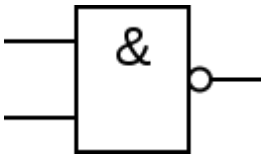
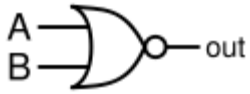
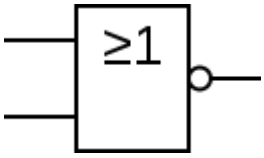
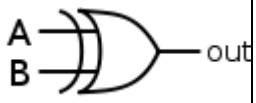
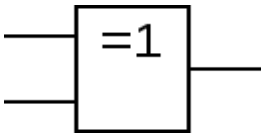
Sur cet ensemble on peut définir les lois ET et OU , …

2. Fonctions logiques et tables de vérité

Une table de vérité est un tableau qui représente des entrées (en colonne)et des états binaires(0 et 1). Le résultat, exprimé lui aussi sous forme binaire, se lit dans la dernière colonne.

Tableau 5 : les portes logiques

<table>
<tr><th>Symboles Américaines</th><th>Symboles Européenne</th><th>Opération Booléenne</th><th>Table de vérité</th></tr>
<tr>
<td>ET(AND)
 
</td>
<td> 
</td>
<td>A.B</td>
<td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><td>A</td><td>B</td><td>A AND B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
<tr>
<td>OU(OR)
 
</td>
<td> 
</td>
<td>A+B</td>
<td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><td>A</td><td>B</td><td>A OR B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
<tr>
<td>NON(NOT)
 
</td>
<td> 
</td>
<td>$\overline{A}$</td>
<td>
<table>
<tr><th>Entrée</th><th>Sortie</th></tr>
<tr><td>A</td><td>NOT A</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>(NAND)
  </td>
<td>  </td>
<td>$\overline{A \cdot B}$</td>
<td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><th>A</th><th>B</th><th>A NAND B</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>NON (NOR)
  </td>
<td>  </td>
<td>$\overline{A+B}$</td>
<td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><th>A</th><th>B</th><th>A NOR B</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>OU exclusif (XOR)
  </td>
<td>  </td>
<td>$A \oplus B$
$= A \cdot \overline{B} + \overline{A} \cdot B$</td>
<td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><th>A</th><th>B</th><th>A XOR B</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

A et **B** sont des signaux d'entrées.

3.Lois fondamentales :

3.1.Loi de commutativité :les opérations + et **.** ; sont commutativités.

A*B=B*A

A+B=B+A

3.2.Loi d'associativité :les opérations + et **.** ; sont associativités.

(A*B)*C=A*(B*C)

(A+B)+C=A+(B+C)

3.3.Loi de Distributivité : chaque des opérations + et . est distributive sur l'autre :

A+(B*C)=(A+B)*(A+C)

A*(B+C)=(A*B)+(A*C)

3.4.Loi D'impotence :

Le résultat d'une opération entre une variable *A* et elle-même est égal à cette variable.

A*A=A

A+A=A

3.5.Loi de Complémentarité :le résultat d'une opération * du complémentarité entre une variable A est égale à 0.

$A * \overline{A} = 0$

$A + \overline{A} = 1$

3.6.Loi de la Double négation :

Le complément du complément d'une variable *A* est égal à cette variable.

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

3.7. L'élément neutre :

À chaque opérateur correspond un élément neutre qui, lorsqu'il est opéré avec une variable quelconque *A*, donne un résultat identique à cette variable.

A*1=A **1** élément neutre du **ET**

A+0=A **0** élément neutre du **OU**

3.8.Loi de L'absorption :

À chaque opérateur correspond un élément nul qui, lorsqu'il est opéré avec une variable quelconque *A*, donne un résultat identique à cet élément nul.

A*0=0

A+1=1

4. Théorème de Morgan

Théorème :1
Le complémentaire du OU (Somme) des variables est égal au ET (Produit) des complémentaires de chaque variable **non (A ou B) = (non A) et (non B).**

Exemple :$\overline{A + B} = \overline{A} * \overline{B}$

Théorème :2
Le complémentaire du ET (Produit) des variables est égal au OU (Somme) des complémentaires de chaque variable. **non (A et B) = (non A) ou (non B).**

Exemple : $\overline{A * B} = \overline{A} + \overline{B}$

5. Dualité

Deux expressions se correspondent par dualité si l'on obtient l'une en changeant dans l'autre, les "ET" par des "OU", les "OU" par des "ET", les "1" par des "0" et les "0" par des "1".
La notion de duale intervient dans l'étude des montages CMOS

Exemple :A*(A+B) par l'expression duale A+(A*B)=A

A+0=A par l'expression duale A*1=A.

6. FONCTION LOGIQUE

Les fonctions logiques « combinatoires », bases du calcul booléen, qui résultent de l'analyse combinatoire des variations des grandeurs d'entrées uniquement et peut avoir dans la sortie deux états (0 et1).
Une fonction booléenne peut être défini par :

- Sa table de vérité
- Son expression
- Sa table de karnaugh
- Son logigramme

6.1.Définition par sa table de vérité :

Les fonctions de « n » variables présentent 2^n résultats. Donc elle sera décrite par une T V de 2^n lignes. Chaque ligne représente la valeur de la fonction par une combinaison binaire de « n » variables.
EX : M=f(a,b,c) : est une fonction booléenne qui prend zéro si la majorité des variables vaut zéro, et 'un' si inversement.
3 var ⟶ $2^n = 2^3 = 8$ lignes.

Tableau 6 : table de vérité (3 variable)

a	b	c	M
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

Alors M(a,b,c)=$\bar{a}bc + a\bar{b}$c+ab$\bar{c} + abc$

6.2.Définition par expression :

Il existe une notation pratique pour représenter une fonction logique :

- On spécifié toutes les combinaisons d'entrées qui fournissent un « 1 ».
- Par convention on place une barre horizontale sur la variable qui vaut un zéro « 0 ».
- L'absence de cette barre signifie qu'elle vaut un « 1 ».

Exemple :$\overline{A}\,BC + A\overline{B}C + \overline{A}\,\overline{B}C + ABC$

6.3.Définition par table de karnaugh :

La table de vérité présente un inconvénient :

Lorsque le nombre de ligne croissant ; en même temps le nombre de variables croissant c à d (n variables ⟶ 2^n lignes). C'est pour ça, on utilise la table de karnaugh.

- Les combinaisons des variables doivent être selon le code gray.
- Si 'n'est le nombre de variables ; p et q deux entier représentant les colonnes et les lignes tel que :
  - * p+q=n
  - * p= q=n /2 ; si le 'n' est pair
  - * p-q=1 ; si le 'n' est impair

  Alors on aura :
  - 2^p colonnes
  - 2^q lignes

Exemple : soit la fonction 'M' de l'exemple précédent : le variable

n= 3 ⟶ impair

p+q=3 ⟶ 2p=3+1=4 ⟶ p=2

p-q=1 ⟶ q=1

Alors :
- $2^p=2^2$= 4 colonnes
- $2^q=2^1$= 2 lignes

Tableau 7: Tableau de karnaugh

BC / A	00	01	11	10
0	0	0	1	0
1	O	1	1	1

6.4.Définitions par logigramme :

Connaissons les symboles logiques, on peut construire un logigramme.

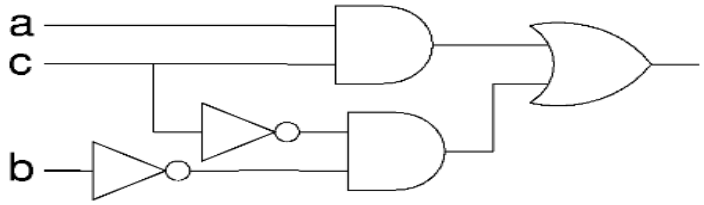


Figure : logigramme logique

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

7.Les formes canoniques :

Une fonction logique exprimée dans une forme canonique est constituée d'un ou plusieurs **termes canoniques.**

7.1.Terme canonique :

Un terme algébrique est dit canonique s'il contient une occurrence de chacune des variables impliquées dans la forme. (les occurrences de la variable a sont $a$ou $\bar{a}$).

7.2.Forme canonique :

Il existe deux formes :

La première forme canonique ou forme disjonctive (Somme des produits) , chacun des termes est constitué par une combinaison de toutes les variables c'est un minterme. La première forme canonique est une réunion de mintermes.

minterme m_i est l'un quelconque des produits des **n** variables de la fonction.

Exemple :

A	B	Minterme
0	0	$m_0(\bar{a}\bar{b})$
0	1	$m_1(\bar{a}b)$
1	0	$m_2(a\bar{b})$
1	1	$m_3(ab)$

Deuxième forme canonique ou forme canonique conjonctive (Produits des somme), chacun des termes est constitué par une combinaison de toutes les variables c'est un Maxterme. Maxterme M_i est l'un quelconque des sommes des **n** variables impliquées.

Exemple :

A	B	Minterme
0	0	$M_0\,(a+b)$
0	1	$M_1\,(a+\bar{b})$
1	0	$M_2(\bar{a}$+b)
1	1	$M_3(\bar{a}+\bar{b})$

Une fonction booléenne peut être exprimée sous forme algébrique à partir de sa table de vérité:

- Elle est égale à la somme des mintermes pour lesquels la fonction vaut 1.
- Elle est égale au produit des maxtermes pour lesquels la fonction vaut 0.

a	b	F(a,b)	mintermes	maxtermes
0	0	1	$m_0(\bar{a}\bar{b})$	$M_0\,(a+b)$
0	1	0	$m_1(\bar{a}$b)	$M_1\,(a+\bar{b})$
1	0	1	$m_2(a\bar{b})$	$M_2(\bar{a}$+b)
1	1	0	$m_3(ab)$	$M_3(\bar{a}+\bar{b})$

Expression de F par des mintermesF(a,b)=$\bar{a}\bar{b}+ a\bar{b}$

Expression de F par des maxtermesF(a,b)= $(a+\bar{b})\ (\bar{a}+\bar{b})$

Exemple :

Soit la fonction $F(a,b,c)=a+\bar{b}c$

Il est clair que F n'est pas canonique

Essayons d'exprimer cette fonction sous la première forme canonique :

Chaque terme doit contenir une occurrence de toutes les variables impliquées dans la fonction pour ce faire :

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

$=a\left(b+\bar{b}\right)(c+\bar{c})+(a+\bar{a})*\bar{b}c$

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

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

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

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

$=m_7+ m_6+ m_5+ m_4+ m_1$

Maintenant essayons d'exprimer cette fonction sous la deuxième forme canonique :

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

$(a + \bar{b})(a + c)$

$(a + \bar{b} + 0)(a + c + 0)$

On remplace le premier 0 par $\bar{c}c$ et le deuxième 0 par $b\bar{b}$ .ainsi, on aura :

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

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

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

$=M_2.M_3.M_0$

8.SIMPLIFICATION DES FONCTIONS LOGIQUES :

Simplifier une fonction revient à réduire le nombre de ses termes ou le nombre de variables dans un même terme. L'intérêt de simplifier une fonction logique apparait dans la réalisation du circuit logique qui lui correspond puisque ça réduit le nombre de portes logiques utilisées pour la réalisation.

Il existe plusieurs méthodes de simplification. Dans ce qui suit, on va en étudier deux :

- Méthode algébrique
- Méthode de karnaugh

8.1.Méthode algébrique :

Elle consiste à utiliser les propriétés de l'algèbre de Boole.

Exemple :

Soit la fonction f(a,b,c)=$(ab + \bar{a}\,c + bc)$

$(ab + \bar{a}\,c + bc)$=$(ab + \bar{a}\,c + bc.1)$

=$(ab + \bar{a}\,c + bc.(a + \bar{a}))$

= $(ab + \bar{a}\,c + bc.a + bc.\bar{a})$

=$(ab + abc + \bar{a}\,c + \bar{a}bc)$

=$(ab(1 + c) + \bar{a}\,c(1 + bc)$

=$(ab.1 + \bar{a}\,c.1)$

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

8.2.Méthode de karnaugh :

1ère règle : encercler tout ensemble de casse occupées (=1), adjacentes sur la même ligne ou sur la même colonne. Une case peut être encerclée deux fois ; si 1 est isolé.

Ex :F(a,b,c)= $a\bar{b} + \bar{a}\,b + ab$= $m_1 + m_2 + m_3$

a/b	0	1
0	m_0	m_1
1	m_2	m_3

a/b	0	1
0	0	1
1	1	1

Cases adjacentes

Cases adjacentes

Si le nombre de cases dans un même groupe est égale à quatre, alors deux variables seront éliminées (les variables qui change l'état).

ab/cd	00	01	11	10
00				1
01				1
11				1
10				1

F=c. $\bar{d}$

Il faut noter aussi que l'adjacent existe pour les extrémités de la table.

a/bc	00	01	11	10
0	1			1
1				1

Pour une fonction à 4 variables alors :

ab/cd	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Exemple :

ab/cd	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	1	1	1
10	0	0	0	0

A,b,cet **d** sont des signaux d'entrées.

F=ab+cb

EXERCICES

Exercice 01

Effectuer les opérations arithmétiques suivantes en binaire :

a) 110101+0110

b) 110111-1101

c) 10110101*1011

d) 110101,110+1011,011

e) 110111 /1011

f) 101101,110/11,010

Exercice 02

Effectuer les opérations suivantes par les méthodes des compléments à (1) et à (2) en utilisant la forme normalisée à 8 bits (bit signe compris).

1. (11) - (3)
2. (-24) - (-9)
3. (-102) - (+23)
4. (+101) - (-69)
5. (+107) – (+219)
6. (+79) + (-6 3)

Exercice 03

I. Effectuer l'addition des nombres suivants en Binaire BCD :
   A. 67+85
   B. 29+83
   C. 307+512

II. Convertir les nombres décimaux suivants en binaire naturel puis convertir en binaire réfléchi:
   - ✓ 25+36
   - ✓ 27
   - ✓ 21

III. Convertir ces nombres binaires réfléchis (GRAY) en binaire naturel :
   A. 10001101
   B. 01011110

La solution

Exercice 01

a

110101
+
0110
=
111011

d

110101,110
+
1011,011
=
1000001,001

c

10110101
*
1011
=
101 0101
+101101010
+0000000000
+ 10110101000
=1110100111

Exercice N° :3

I. Effectuer l'addition des nombres suivants en Binaire BCD :

A)

67
+
85
=
152

BCD +

0110 0111
+
1000 0101
=
1110 1100
+ 0110 0110
=0001 0101 0010

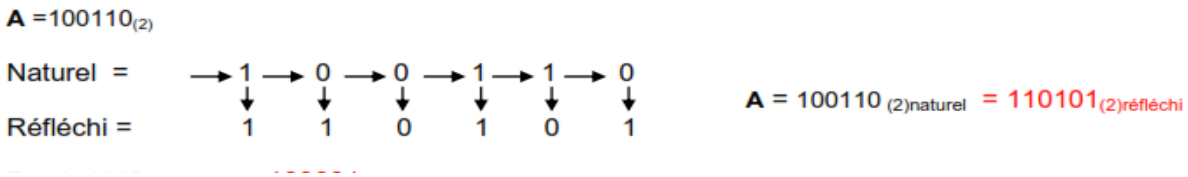
B)

307
+
512
=
819

BCD +

001100000111
+
010100010010
=
1000 0001 1001

I. II. Convertir les nombres décimaux suivants en binaire naturel puis convertir en binaire réfléchi:



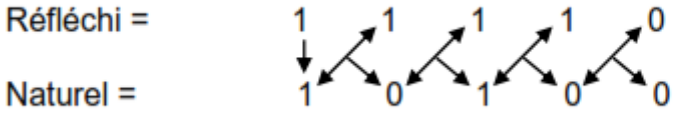
✓ 25 en Binaire Naturel =11001 et 36 en Binaire Naturel =100101

11001
+ 100101
= 111101

Binaire réfléchi 100011 (même méthode)

✓ 21 en Binaire Naturel 10101 en Binaire réfléchi 11111 (même méthode)

III. GRAY(Réfléchi) en Binaire Naturel



A. 10001101 Binaire Naturel 11110111 (même méthode)
B. 01011110 Binaire Naturel 10010101 (même méthode)

CHAPITRE III

TECHNOLOGIE DES CIRCUITS INTEGRES TTL ET CMOS

1.Introduction

Le but de ce chapitre est d'apporter une connaissance sur les deux grandes familles technologiques utilisées dans les circuits logiques en électronique. Ce cours va donc comparer les familles TTL et CMOS en donnant leurs principales caractéristiques.

1.1.Présentation

1-Définition :

Avant toute chose il faut définir ce que sont TTL et CMOS. Ce sont deux familles technologiques utilisées pour les circuits logiques en électronique.
Pour ceux qui ne comprennent pas le terme 'circuit logique, c'est en fait un circuit intégré contenant des portes logiques telles que des OR, AND, NAND etc...

2.La famille TTL :

TTL est l'abréviation de "Transistor-Transistor Logic". Elle a été inventée en 1960. Cette famille est réalisée avec des transistors bipolaires. (De nos jours, la technologie TTL tend à être remplacée par la technologie CMOS).
Les avantages de cette famille :

- Les entrées laissées en 'l'air' ont un état logique à 1 par défaut.
- Une bonne immunité au bruit.
- Un temps de propagation faible.

Les inconvénients de cette famille :

- L'alimentation doit être précise à 5V +/- 5 % sinon on risque de détruire le circuit.
- Du fait qu'elle est réalisée avec des transistors bipolaires elle consomme pas mal de courant comparer à la famille CMOS. (Car les transistors bipolaires sont commandés en courant).

Le nom des circuits de cette famille commencent par 74 suivi d'une ou plusieurs lettres représentant la série et suivi d'un code à 2 ou 3 chiffres représentant le modèle du circuit.

Exemple :

Le circuit 74LS08 (porte ET).

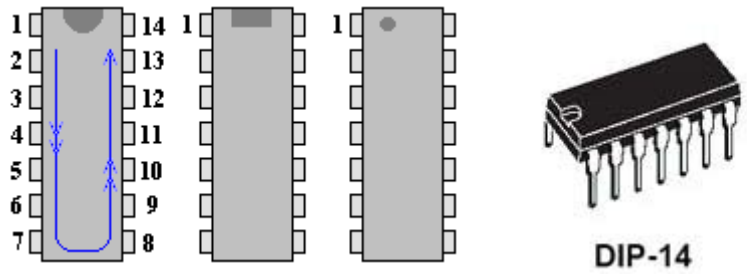
Les lettres au milieu du nom représentent donc la série. On décline plusieurs séries :
74xx : série standard.
74Lxx : série faible consommation.
74Sxx, 74ASxx, 74ALSxx 74Fxx, 74AFxx : série rapide.
74LSxx : mélange des technologies L et S, c'est une des séries la plus répandue.

Il existe aussi des séries 74HCxx et 74HCTxx qui est le résultat d'une combinaison de la technologie TTL et CMOS afin d'ajouter leurs avantages.

Comment utiliser les circuits intégrés de la famille TTL ????

2.1.Règle

Cette encoche détermine " l'avant" du CI. Pour regarder un CI, il faut le voir **du dessus** avec l'encoche en haut (comme sur le Dessin) On peut donc, à l'aide de cette encoche, trouver les numéros des pattes. En effet, la patte n°1 se situe **toujours** à gauche de cette encoche. Pour trouver les autres pattes, il suffit de descendre, puis de remonter sur la colonne de droite (en suivant la flèche bleue sur le dessin). On peut donc, de cette manière, déterminer les numéros des broches.



Exemple TTL 7400

Figure : Circuit intégrés **TTL**

3.La famille CMOS :

CMOS est l'abréviation de "ComplementaryMetalOxide Semi-conductor". Le premier dispositif MOS est apparu en 1960. Son développement a été rendu possible par les progrès réalisés par la technologie TTL. Cette famille est réalisée avec des transistors à effet de champs.

Les avantages de cette famille :

- L'alimentation peut aller de 3V à 18V.
- Le courant d'entrée est nul, car elle est réalisée avec des transistors à effet de champs. (Les transistors à effet de champs sont commandés en tension).
- Une excellente immunité au bruit.

Les inconvénients de cette famille :

- La vitesse de commutation est plus faible que pour la technologie TTL.

Le nom des circuits de cette famille commencent par 40 suivis de deux chiffres représentant le modèle du circuit.

Exemple : le circuit 4081 (porte ET).

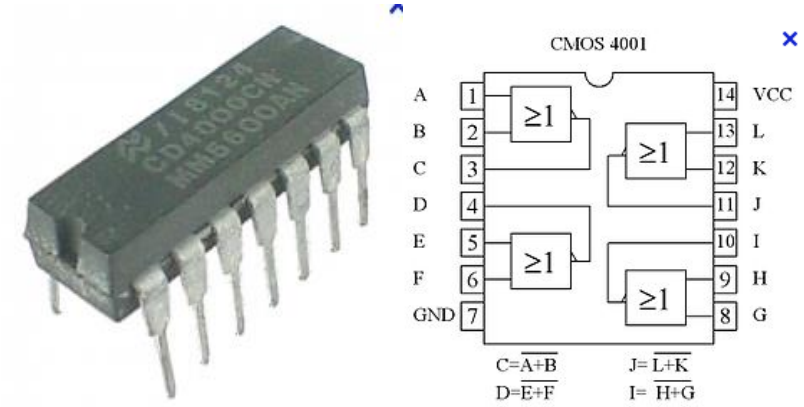


Figure : circuit intégré CMOS 4001

EXERCICES

Exercice N° :01

Soit la fonction logique à 3 variables **a,b,c** définies par la table de vérité ci-dessous :

1. Ecrire la fonction S sous forme Somme des produits ?
2. Simplifier S ?

a	b	c	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Exercice N° :02

Montrer à l'aide de table de vérité :

A. Le théorème de morgan pour 3 variables ?
B. La distributivité de la somme logique par rapport au produit logique ?

Exercice N° :0 3

I. On appliquant les propriétés le l'algèbre de boole, montrer les fonctions suivantes :
- (a.b+b.c+a.c)=(a+b). (b+c).(a+c).
- $(a + \bar{b} + a.\bar{b})(a.b + \bar{a}.c + b.c) = a.b + \bar{a}.\bar{b}.c$

II. Simplifier les fonctions suivantes :
- $f = \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + a\bar{b}c\bar{d} + abc\bar{d} + a\bar{b}\bar{c}\bar{d} + \bar{a}bc\bar{d}$
- S= $\bar{x}.\bar{y}.\bar{z} + \bar{x}.\bar{y}.z + \bar{x}.y.z + x.\bar{y}.\bar{z} + x.\bar{y}.z + x.y.\bar{z} + x.y.z$
- $f = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$
- T= (a+b+c).($\bar{a} + b + c$)+a.b+b.c
- R= a+a.b.c+$\bar{a}$.b.c+$\bar{a}$.b+a.d+a.$\bar{d}$.

Exercice N° :0 4

Exprimer la fonction suivante sous la première et la deuxième forme canonique :

F(X,Y,Z,W)= $\bar{Y}.Z + W.X.\bar{Y} + W.X.\bar{Z} + \overline{W}.\bar{X}.Z$

Exercice N° :0 5

i. Soit les tableaux de karnaugh suivant :

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

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

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

- Donnez l'expression algébrique de chaque tableau ?

ii. Soit la fonction suivante :

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

- Simplifier cette fonction avec la table du karnaugh?

La solution

Exercice 01

3. D'après la table de vérité on a :

a	b	c	S
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	**1**
1	1	1	0

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

4. La simplification

$S = \bar{a}.\bar{b}.(\bar{c} + c) + b.\bar{c}(\bar{a}. + a.)$

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

Exercice 02

C. La distributivité de la somme logique par rapport au produit logique

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

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

Nous allons vérifier ces lois en utilisant une table de vérité :

a	b	c	$\bar{a}$	$\bar{b}$	$\bar{c}$	a+b+c	$\overline{a+b+c}$	a.b.c	$\overline{a.b.c}$	$\bar{a}.\bar{b}.\bar{c}$	$\bar{a}+\bar{b}+\bar{c}$
0	0	0	1	1	1	0	1	0	1	1	1
0	0	1	1	1	0	1	0	0	1	0	1
0	1	0	1	0	1	1	0	0	1	0	1
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	0	1	1	1	0	0	1	0	1
1	0	1	0	1	0	1	0	0	1	0	1
1	1	0	0	0	1	1	0	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0

Conclusion

Les lois de MORGAN sont vérifiées pour trois variables.

D. La somme logique est distributive par rapport au produit logique, soient les trois variables logiques a,b et c ; on a : a+b.c=(a+b).(a+c).

Démontrons cette relation par la table de vérité :

a	b	c	a.b	a+b.c	a+b	a+c	(a+b).(a+c)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Conclusion
La somme logique est distributive par rapport au produit logique.

Exercice 03
La simplification

- $f = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$
- $f = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc + abc + abc$
- $f = ab(c + \bar{c}) + ac(b + \bar{b}) + bc(a + \bar{a})$
- $f = ab + ac + bc$

- $f = \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}cd + abcd + a\bar{b}c\bar{d} + \bar{a}bc\bar{d}$
- $f = \bar{a}b\bar{d}(\bar{c} + c) + a\bar{b}d(\bar{c} + c) + bc\bar{d}(\bar{a} + a)$
- $f = \bar{a}b\bar{d} + a\bar{b}d + bc\bar{d}$
- $f = \bar{b}\bar{d}(a + \bar{a}) + bc\bar{d}$
- $f = \bar{d}(\bar{b} + bc)$
- $f = \bar{d}((\bar{b} + b).(\bar{b} + c))$
- $f = \bar{d}(\bar{b} + c) = \bar{b}\bar{d} + c\bar{d}$

Les autres équations avec la même méthode.

Exercice 04

$$F = \bar{Y}.Z + W.X.\bar{Y} + W.X.\bar{Z} + \overline{W}.\bar{X}.Z$$
$$= \bar{Y}.Z.(X + \bar{X}).(W + \bar{W}) + W.X.\bar{Y}.(Z + \bar{Z}) + W.X.\bar{Z}.(Y + \bar{Y}) + \overline{W}.\bar{X}.Z.(Y + \bar{Y})$$
$$= X.\bar{Y}.Z.W + X.\bar{Y}.Z.\bar{W} + \bar{X}.\bar{Y}.Z.W + \bar{X}.\bar{Y}.Z.\overline{W}. + X.\bar{Y}.Z.W + X.\bar{Y}.\bar{Z}.W$$
$$+ X.Y.\bar{Z}.W + X.\bar{Y}.\bar{Z}.W + \bar{X}.Y.Z.\overline{W}. + \bar{X}.\bar{Y}.Z.\overline{W}.$$
$$= X.\bar{Y}.Z.W + X.\bar{Y}.Z.\bar{W} + \bar{X}.\bar{Y}.Z.W + X.\bar{Y}.\bar{Z}.W + X.Y.\bar{Z}.W + X.\bar{Y}.\bar{Z}.W$$
$$+ \bar{X}.Y.Z.\overline{W}. + \bar{X}.\bar{Y}.Z.\overline{W}.$$
$$= m_1 + m_{10} + m_3 + m_9 + m_{13} + m_6 + m_2$$

Exercice 05

Simplification avec table de karnaugh :

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

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

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

$G = D$

Les autres tableaux avec la même méthode.

- La Simplifier de cette fonction avec la table du karnaugh

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

C / AB	0	1
00	1	
01	1	
11	1	1
10	1	

2

1

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

CHAPITRE IV

LES CIRCUITS COMBINATOIRES

1. INTRODUCTION

Pour résoudre le problème de la logique combinatoire , il faut connaître l'algèbre de Boole. Cet algèbre permet de traduire des signaux en expressions mathématiques on remplaçant chaque signal élémentaire par des variables logiques et leur traitement par des fonctions logiques. Ces fonctions seront appelées fonctions combinatoires et l'étude de la logique combinatoire.

2.Un circuit combinatoire :

Un circuit combinatoire est un circuit numérique dont les sorties dépendent uniquement des entrées.

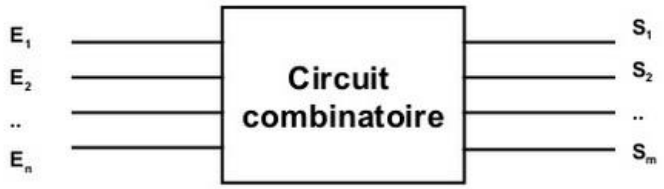


Schéma Bloc

C'est possible pour utiliser des circuits combinatoires de base pour réaliser d'autre circuits complexes.

2.1.Demi additionneur :

Un **additionneur** est un circuit logique permette de réaliser une addition. Ce circuit est très présent dans les ordinateurs pour le calcul arithmétique mais également pour le calcul d'adresses, d'indice de tableau dans le processeur.

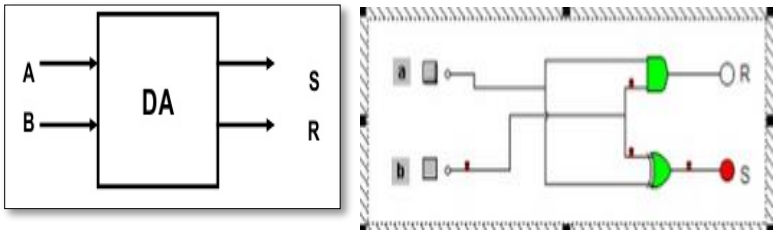


Schéma bloc logigramme

Figure 6 :demi additionneur

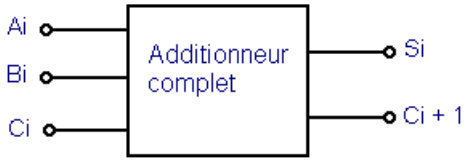
Entrées		S	R
A	B		
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S_{add}=A.\bar{B}+\bar{A}.B = A \oplus B.$

$R_{Add}=A.B$

2.2.Additionneur complet

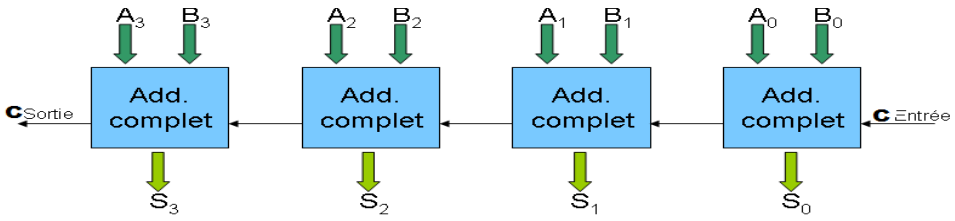
Un additionneur complet nécessite une entrée supplémentaire : une retenue. L'intérêt de celle-ci est de permettre le chaînage des circuits. La table de vérité et le schéma bloc d'un additionneur complet sont:



Représentation schématique d'un additionneur complet.

Tableau8 : table d'Additionneur complet

Additionneur Binaire Complet				
a_i	b_i	c_i	s_i	c_{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



Il y a deux méthodes pour obtenir les équations de ce circuit :

1. Méthode1
   (a) Analytique :

Alors l'expression de la somme S_i est :

$$\mathbf{S_i} = \overline{A}_i.\overline{B}_i.C_i + \overline{A}_i.B_i.\overline{C}_i + A_i.\overline{B}_i.\overline{C}_i + A_i.B_i.C_i$$

$$= \overline{A}_i.(B_i.\overline{C}_i + \overline{B}_i.C_i) + A_i.(\overline{B}_i.\overline{C}_i + B_i.C_i)$$

$$= \overline{A}_i.(B_i \oplus C_i) + A_i.\overline{(B_\iota \oplus C_\iota)}$$

$$= A_i \oplus B_i \oplus C_i$$

Et l'expression de la retenue C_{i+1} :

$$\mathbf{C_{i+1}} = \overline{A}_i.B_i.C_i + A_i.\overline{B}_i.C_i + A_i.B_i.\overline{C}_i + A_i.B_i.C_i$$

$$= (\overline{A}_i.B_i + A_i.\overline{B}_i).C_i + A_i.B_i.(C_i + \overline{C}_i) = (A_i \oplus B_i)C_i + A_i.B_i$$

(2) Méthode II :
   (b) Table de karnaugh :

Pour S_i :

B_iC_i / A_i	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	0	(1)

Pour C_{i+1} :

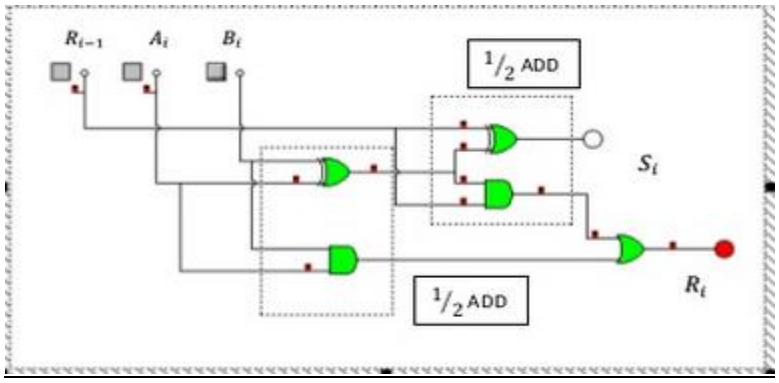
B_iC_i / A_i	00	01	11	10
0	0	0	(1)	0
1	0	(1)	(1)	(1)

Après la simplification on a trouvé que :

$$\mathbf{S_i} = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = (A_i + B_i)C_i + A_i.B_i = (A_i \oplus B_i)C_i + A_i.B_i$$

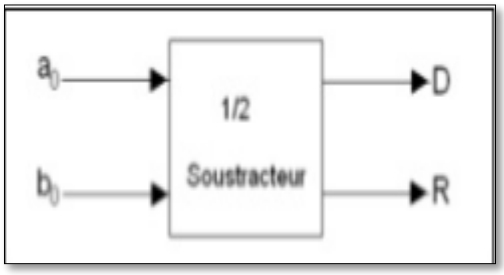
Logigramme :



Logigramme d'un additionneur complet

2.3.Demi-Soustracteur

C'est un circuit capable de faire la soustraction de deux bits binaires . Le circuit aura deux entrées A, B et deux sorties S et R.



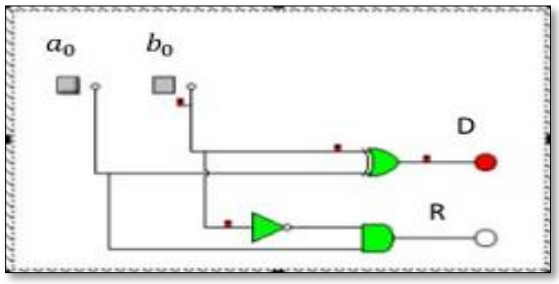


Schéma bloc logigramme

Figure :Demi Soustracteur

Entrées		S	R
B	A		
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

La table de vérité

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

R= $A.\bar{B}$

2.4.Comparateur

Le comparateur est un circuit arithmétique permettre de comparer deux nombres binaires A et B, avec A et B doivent avoir la même longueur (nombre de bits).ce dernier, on cherche à

savoir Si A>B, A<B et A=B. On comprend donc que le circuit répond à une question à trois chois.

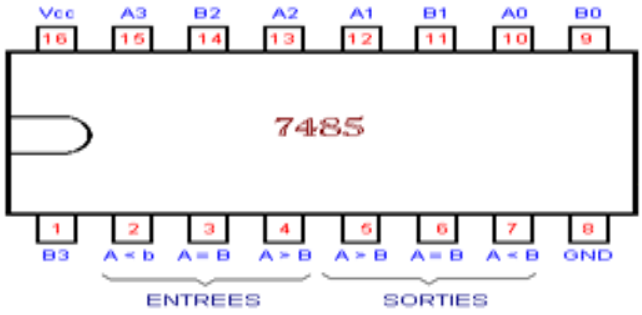


Fig. 1. - Brochage du circuit intégré 7485.

La table de vérité d'un comparateur de deux bits peut être exprimée de la manière suivante : (tableau ci-dessous).

Tableau 18: table d'un comparateur à 2 bits

Entrées		Sorties		
A	B	S_2 A<B	S_1 A>B	S_0 A=B
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Alors :

$S_1=\overline{A}.B$

$S_2= A.\overline{B}$

$S_0=\overline{A}.\overline{B}+A.B=\overline{\boldsymbol{A}.\overline{\boldsymbol{B}}+\overline{\boldsymbol{A}}.\boldsymbol{B}}$

Nous avons que $\overline{\boldsymbol{A}}.\overline{\boldsymbol{B}}+\boldsymbol{A}.\boldsymbol{B}+\boldsymbol{A}.\overline{\boldsymbol{B}}+\overline{\boldsymbol{A}}.\boldsymbol{B}=\mathbf{1}$

Et $\overline{A}.\overline{B}+A.B$ c'est le complément de $A.\overline{B}+\overline{A}.B$

Puisque la somme de ses deux expressions reçoit 1.

Alors : $\overline{\boldsymbol{A}}.\overline{\boldsymbol{B}}+\boldsymbol{A}.\boldsymbol{B}=\overline{\boldsymbol{A}.\overline{\boldsymbol{B}}+\overline{\boldsymbol{A}}.\boldsymbol{B}}$

❖ **Schéma de circuit comparateur**

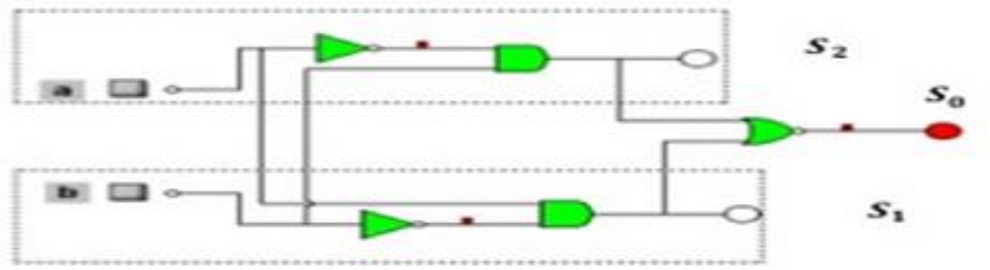


Figure1 : schéma logique d'un comparateur à 2 bits

2.6.Le multiplexeur (MUX)

Le multiplexeur est un système combinatoire ayant pour fonction de sélectionner une parmi 2^nentrées et de la transmettre à la sortie. La sélection est faite à l'aide de *n* lignes d'adresse. La notation usuelle du MUX est: MUX 2^n à 1. Par exemple, un MUX 8 à 1 aura 3 lignes d'adresse. La figure 5 présente la forme générale d'un MUX.

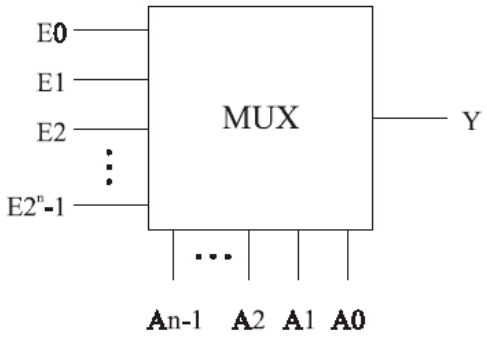
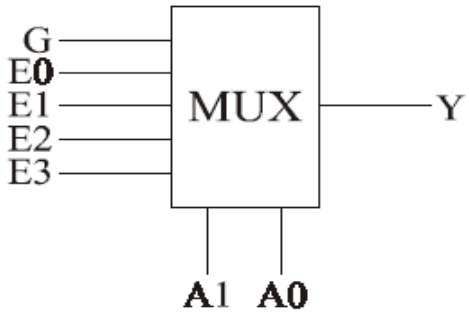


Figure05 : représentation générale d'un Multiplexeur

La figure .6 montre le composant 74153 qui contient deux MUX 4 à 1. Le signal supplémentaire *G* (*Strobe*) est un signal d'activation du composant. Si *G* est inactif, la sortie du MUX sera obligatoirement inactive.



G	A1	A0	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	E0
1	0	1	E1
1	1	0	E2
1	1	1	E3

Figure . 6: Caractéristiques du 74153.

Les applications d'un multiplexeur en électronique sont principalement :

- la génération de fonctions logiques

- la conversion parallèle / série d'informations
- la concentration de données et leur transmission parallèle
- le décodage d'un clavier matriciel
- l'affichage multiplexé sur des afficheurs 7 segments

2.7.Le démultiplexeur (DEMUX)

Le démultiplexeur est un système combinatoire ayant pour fonction de transmettre une entrée vers une des 2^nsorties. La sélection est faite à l'aide de *n* lignes d'adresse et les sorties sont mutuellement exclusives. La notation usuelle du DEMUX est: DEMUX 1 à 2^n. Par exemple, un DEMUX 1 à 8 aura 3 lignes d'adresse. La figure .7 présente la forme générale d'un DEMUX.

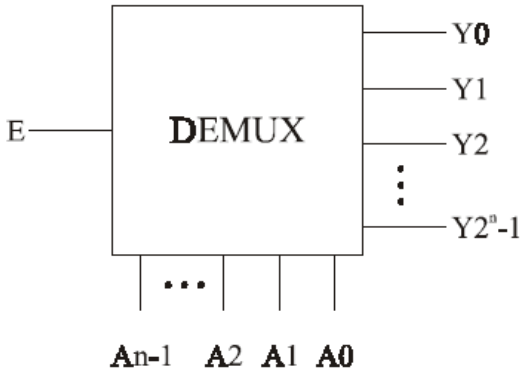
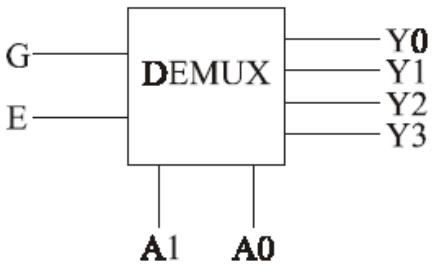


Figure . 7: Représentation générale d'un démultiplexeur.

La figure 8 montre un DEMUX 1 à 4. Le signal supplémentaire *G* (*Strobe*) est un signal d'activation du composant. Si *G* est inactif, toutes les sorties du DEMUX seront obligatoirement inactives.



G	A1	A0	Y
0	0	0	Yn inactifs
0	0	1	Yn inactifs
0	1	0	Yn inactifs
0	1	1	Yn inactifs
1	0	0	Y0=E
1	0	1	Y1=E
1	1	0	Y2=E
1	1	1	Y3=E

Figure . 8: Caractéristiques d'un démultiplexeur.

2.8.Décodeur

Les circuits de transformation des codes font la transposition des données d'un code à un autre. Ils jouent le rôle d'interprète entre l'homme et la machine (codeur) ,entre la machine et l'homme (décodeur) et entre machine et machine (transcodeur).

Le décodeur est un système combinatoire ayant pour fonction d'activer une des 2^nsorties.
La sélection est faite à l'aide de *n* lignes d'adresse et les sorties sont mutuellement exclusives.
La notation usuelle du décodeur est: décodeur 1 parmi 2^n. Le décodeur se comporte exactement comme un DEMUX avec son entrée toujours à 1.

- Convertisseur de code à un code de sortie d'entrée correspond un code de sortie.
  Exemple: Un décodeur binaire octal possède 3 bits d'entrés permettant 2^3=8 combinaisons pour activer chacun des 8 sortie de l'octal.

- Sélecteur de sortie: Une seule sortie parmi les M disponibles est activée à la fois en fonction de la valeur binaire affichée à l'entré. Ces fonctions permettent d'activer (sélectionner) un circuit intégré parmi plusieurs.

2.8.1.Principe d'un décodeur 1 parmi 4

Pour pouvoir activer toutes les 4 voies on a besoin de 2 bits à l'entrée car c'est 2^2=4.

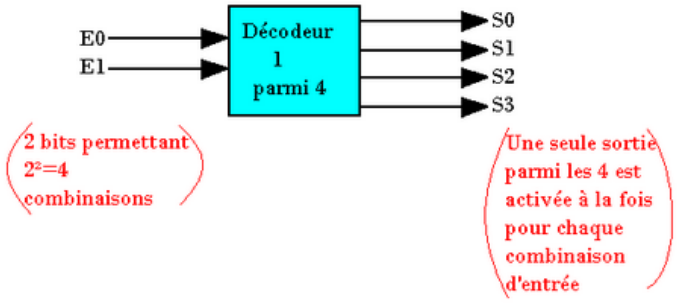


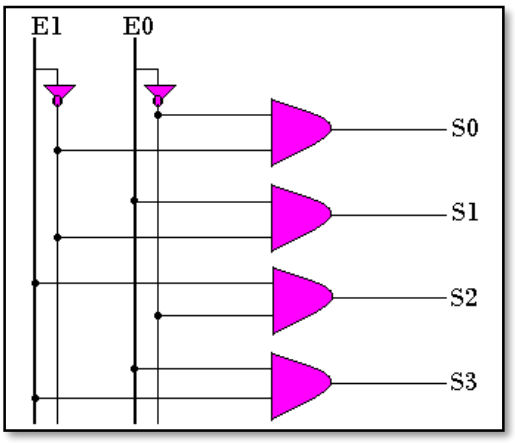
Table de fonctionnement					
Code binaire d'entrée		**Code 1 parmi 4 sorties**			
E_1	**E_0**	**S_3**	**S_2**	**S_1**	**S_0**
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Équation de sorties

$S_0=\overline{E1}.\overline{E0}$ $S_2=E1.\overline{E0}$

$S_1=\overline{E1}.E0$ $S_3=E1.E0$

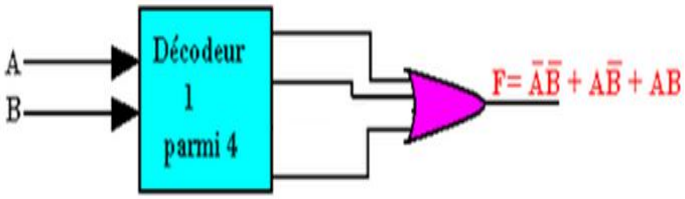
Logigramme:



Logigramme d'un décodeur

Exemple : réaliser la fonction
$F = \overline{A}\overline{B} + A\overline{B} + AB$

Solution :

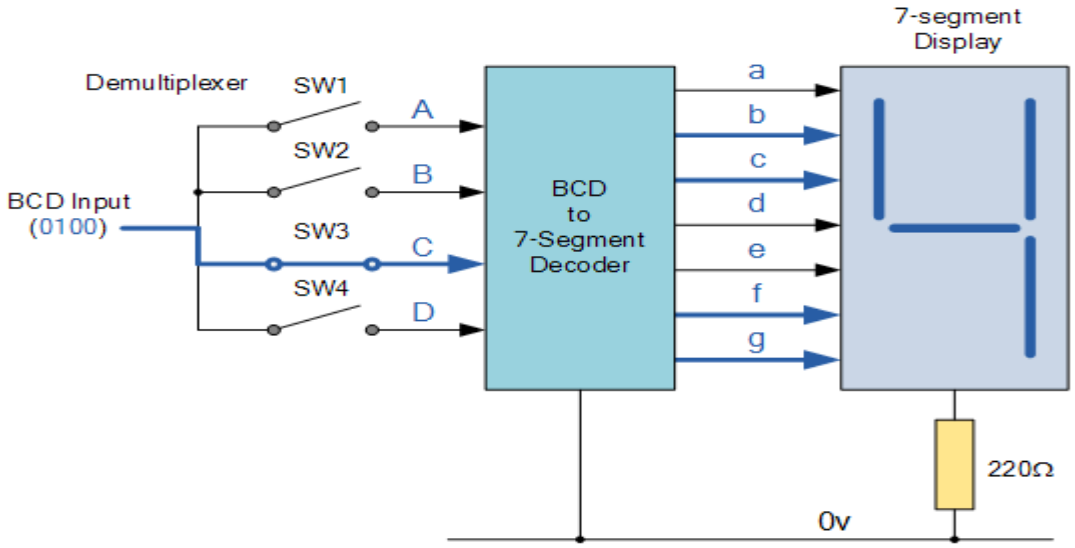


Exercice : réaliser les fonctions X1 et X2

$X1 = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + ABC$

$X2 = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + ABCD$

Exemple d'un afficheur BCD 7 segments

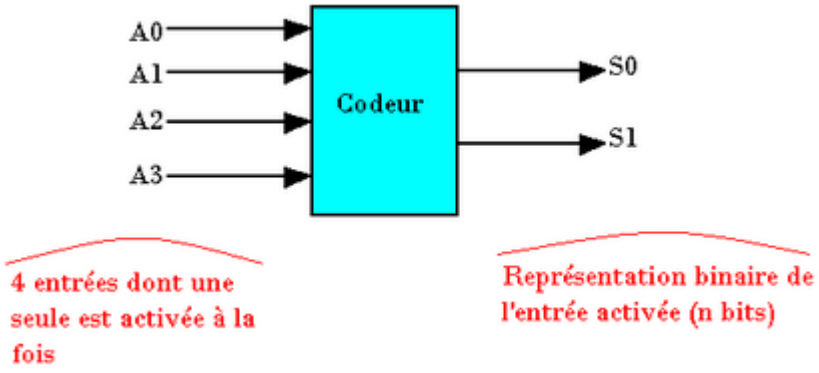


2.9.L'encodeur (Codeur) :

L'encodeur est un système combinatoire ayant pour fonction de retourner l'index d'activation d'une parmi 2^n entrées. L'index d'activation est donné sur *n* lignes d'adresse. Lorsque plusieurs entrées sont activées, l'encodeur accorde la priorité à l'entrée dont l'index est supérieur.

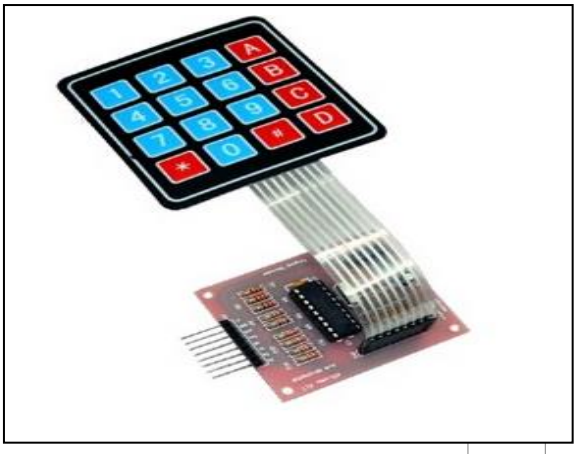
La notation usuelle de l'encodeur est: encodeur 2^n à n. Par exemple, un encodeur 8 à 3 aura 8 entrées et 3 lignes d'adresse en sortie.(il fournit en sortie le numéro de l'entrée active sur n bit)

2.9.1.Principe d'un codeur 4 voies d'entrées et 2 bits de sortie



La table de vérité :

ENTREES				SORTIE	
Codage 1 parmi 2^n				Nombre binaire de n bits	
A_3	A_2	A_1	A_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

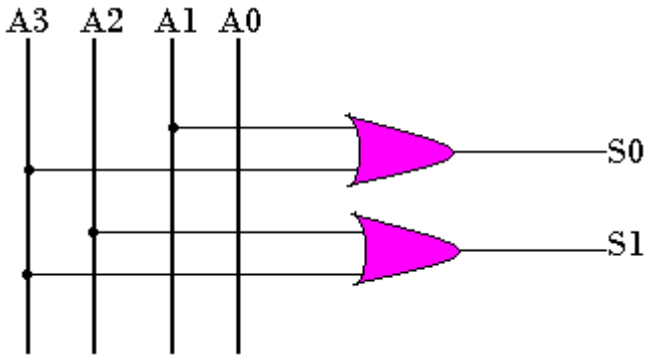


Equation des sorties

S_1=1 si (A_2=1) ou (A_3=1) ; $S_1=A_2+A_3$

S_0=1 si (A_1=1) ou (A_3=1) ; $S_0=A_1+A_3$

Logigramme



2.9.2.Codeur de priorité

Si nous activons simultanément les entrées A_1 et A_2 du codeur ci-dessus, les sorties S_1S_0 présente le nombre 11 qui ne correspond pas au code de l'une ou de l'autre des entrées activés. C'est plutôt le code qui représente l'activation de A_3
Pour résoudre ce problème on utilise un codeur de priorité qui choisit le plus grand nombre lorsque plusieurs entrées sont activées à la fois. Exemple lorsque A_1 et A_2 sont activées simultanément S_1S_0 sera égale à 10 qui représentent l'activation de A_2.

Tableau 9 :Exemple d'un codeur de priorité octal-binaire

ENTREES (OCTAL)								**SORTIES**		
A_7	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	**S_2**	**S_1**	**S_0**
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

2.10.Transcodeur

Ce sont des circuits qui transforment une donnée en code machine en un autre code machine.
Exemple: Transformation binaire pur binaire réfléchi ou binaire pur ASCII.

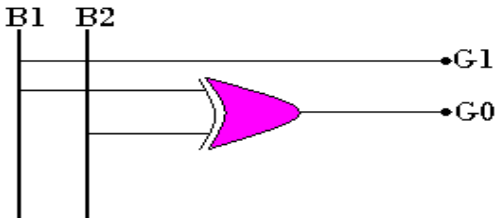
Table de vérité:

ENTREES		SORTIES	
B_1	B_0	G_1	G_0
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Equation de sortie:

$$G1 = B1\overline{B0} + B1B0 = B1$$
$$G0 = \overline{B1}B0 + B1\overline{B0} = B1 \oplus B0$$

Logigramme:



EXERCICES

Exercice N° :01

Soit la fonction suivante :

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

1. Donner sa table de vérité ?
2. Simplifier F?
3. Réalisez cette fonction avec :
   - D'un minimum portes logiques ?
   - D'un décodeur 3 à 8 et d'un minimum portes logiques ?
   - D'un DEMUX 1 à 8 et d'un minimum portes logiques ?

Exercice N° :02

Le multiplexeur peut être utilisé comme générateur des fonctions logiques, avec 8 entrées

(3 lignes d'adresse), on aura pour le signal de sortie :

$S(A,B,C) = \bar{A}\bar{B}\bar{C}D_0 + \bar{A}\bar{B}CD_1 + \cdots + ABCD_7$

Avec :D_0,D_1,D_2,…,D_7 : entrées des multiplexeur,

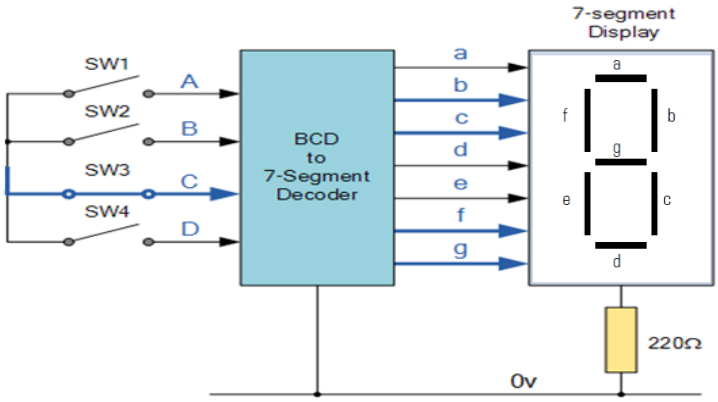
A,B,C : ligne d'adresse du multiplexeur.

I. Réaliser à l'aide de multiplexeur à 8 entrées les fonctions logiques suivantes :
   - $G(X,Y,Z) = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + \bar{X}YZ + XY\bar{Z}$
   - $H = AC + \bar{A}\bar{B} + AB$
   - $E = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\overline{D} + \bar{A}B\bar{C}\overline{D} + A\bar{B}\bar{C}\overline{D} + \bar{A}BCD + A\bar{B}CD + AB\bar{C}D + ABC\overline{D}$.

Exercice N° :03

Le circuit suivant est un pilote/décodeur 7 segments destiné à être utilisé avec un affichage LED à 7 segments. L'entrée **A** correspond au bit le plus significatif. Les sorties commandent l'allumage des segments d'un afficheur constitué de diodes a,b,…,g figure(1), avec les règles suivant :

- Mètre l'entrée **C** du décodeur à **1**, les sorties (segments) **b,c,f et g** sont allumés (à **1)** , les règles sont indiquées sur la figure (2).



Figure(1)

Figure (2)

I. Dresser la table de vérité de ce circuit pour les valeurs d'entrée allant de **0** à **9** ?
II. Déterminer les fonctions simplifiées de **a** à **g** avec la table de karnaugh ?
III. Donner le schéma permettant de réaliser la fonction **e,** le plus simplement possible avec des portes NAND à deux entrées ?

La solution

Exercice 01

Soit la fonction suivante :

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

La table de vérité :

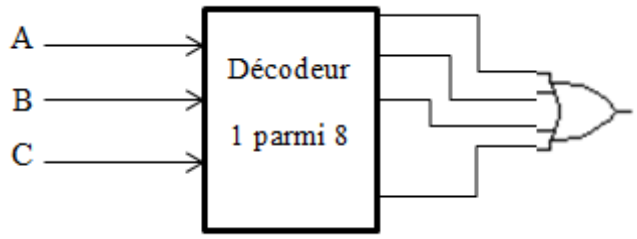
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- $F(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C}$

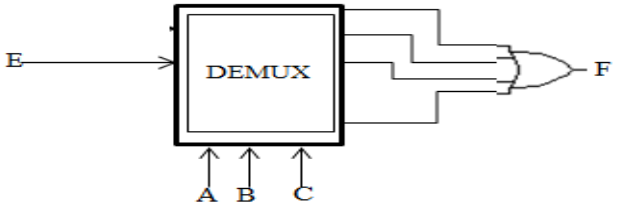
  $= \bar{A}(\bar{B}\bar{C} + BC) + A(\bar{B}C + B\bar{C})$

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

- $F(A,B,C) = \overline{A \oplus (B \oplus C)}$
- Avec un décodeur



- Avec DEMUX



Exercice 03

I. table de vérité

regroupons dans un table l'état désiré pour les sorties dans chaque cas. L'ordre n'a pas d'importance

décimal	Binaire					a	b	c	d	e	f	g
0	0	0	0	0		0	0	0	0	0	0	1
1	0	0	0	1		1	0	0	1	1	1	1
2	0	0	1	0		0	0	1	0	0	1	0
3	0	0	1	1		0	0	0	0	1	1	0
4	0	1	0	0		1	0	0	1	1	0	0
5	0	1	0	1		0	1	0	0	1	0	0
6	0	1	1	0		0	1	0	0	0	0	0
7	0	1	1	1		0	0	0	1	1	1	1
8	1	0	0	0		0	0	0	0	0	0	0
9	1	0	0	1		0	0	0	0	1	0	0

On peut maintenant analyser chaque sortie indépendamment, pour déterminer les équations. Nous allons utiliser des tableaux de Karnaugh.

Avec e_0=D,e_1=C, e_2=A et e_3=B.

II. **Fonction** (**a)**

e1e0

a	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	0	0
10	0	0	1	0

e3e2

Fonction (g)

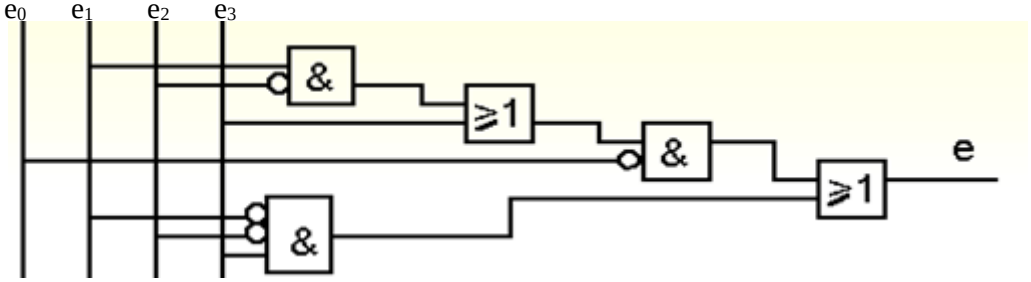
e1e0

g	00	01	11	10
00	1	1	0	0
01	0	0	1	0
11	1	0	0	0
10	0	0	0	0

e3e2

$$g = \overline{e_3}\,(\overline{e_2}\,\overline{e_1} + e_2 e_1 e_0) + e_3 e_2 \overline{e_1}\,\overline{e_0}$$

III. Schéma (fonction (**e**)).



CHPITRE V

CIRCUIT SEQUENTIEL 'BASCULES'

1. INTRODUCTION

La mémoire de l'ordinateur stocke les données et les instructions du programme en cours d'exécution. Pour réaliser cette mémoire, des circuits sont utilisés dits circuits de mémorisation. Pour mémoriser un bit (0 ou 1), il faut utiliser un circuit capable de se souvenir de la valeur qu'il a enregistrée.

Ces circuits sont appelés circuits séquentiels. A l'inverse des circuits combinatoires, la valeur de sortie d'un circuit séquentiel ne dépend pas que des variables logiques d'entrée, mais dépend aussi de la valeur de sortie antérieure.

Dans ce chapitre, on va étudier les éléments de base d'un circuit séquentiel et leurs fonctionnements.

2. Définition d'un circuit séquentiel

Un circuit séquentiel est composé d'un circuit combinatoire et d'éléments mémoire appelés BASCULES.

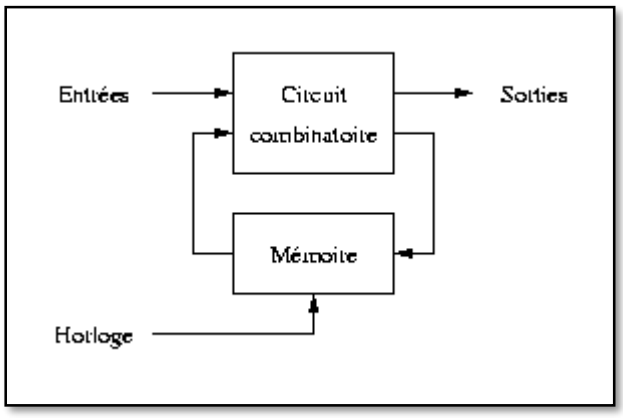


Schéma d'un circuit séquentiel

On distingue deux types de circuits séquentiels :

- Les circuits séquentiels synchrones
- Les circuits séquentiels asynchrones

Dans un circuit séquentiel synchrone, l'état de variable de sortie dépend des signaux appliqués en entrée à des périodes de temps régulières. Ces circuits utilisent généralement des horloges et sont, donc, appelés séquentiels à horloge.

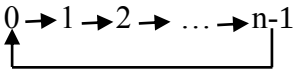
Par contre, un circuit séquentiel asynchrone n'est pas une horloge. L'état d'un tel circuit dépend de l'ordre d'apparition des signaux en entrée.

3. Définition d'une bascule

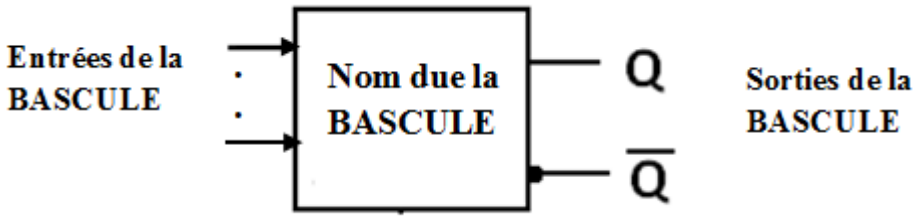
Une bascule est un élément capable de stocker un bit et le restituer au temps voulu. Les bascules sont utilisées pour réaliser des circuits de mémorisation.

Exemple :

1. Pour réaliser un compteur modulo n, on a besoin de mémoriser la valeur précédente du compteur afin de l'incrémenter à chaque fois.



4. **Schéma fonctionnel d'une bascule :**illustrées dans la figure ci-dessous.



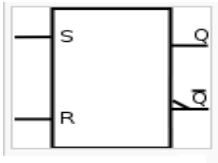
Q : est la variable de sortie de la bascule. Elle fournit l'état de la bascule. Si Q=1, on dit que l'état de la bascule est à 1. Sinon, l'état de la bascule est 0.

$\overline{Q}$: C'est l'inverse de la variable de sortie Q.

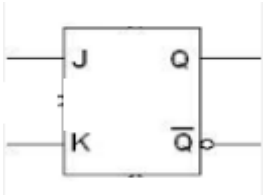
Nom de la bascule : un nom est associé à chaque bascule. Il indique les entrées de la bascule en question. On distingue quatre bascules de base :

- La bascule R-S
- La bascule J-K
- La bascule T
- La bascule D

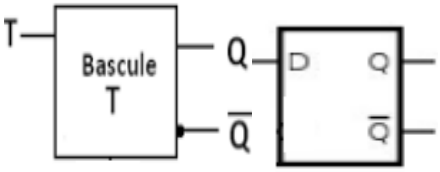
Ainsi, les bascules de base sont représentées par les schémas suivants :



La bascule R-S



la bascule J-K



La bascule T la bascule D

Dans ce qui suit, on va étudier ces bascules de base, l'une après l'autre, dans le but de comprendre le principe de leur fonctionnement et leurs points de différence.

4.LES BASCULES DE BASE :

4.1.La bascule R-S :

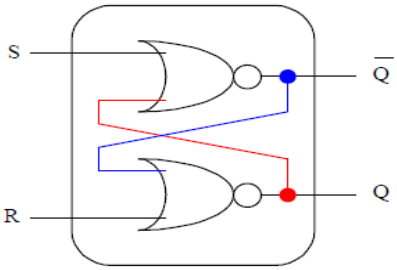
La bascule R-S est un circuit formé de deux portes logiques NOR(non ou) ou NAND (non et). Ce circuit possède deux entrées et deux sorties :

Les entrées : S pour la mise à 1 de la bascule

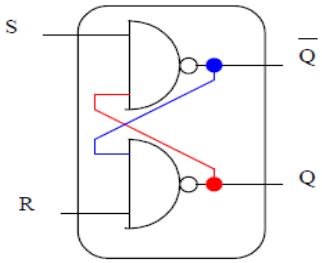
R pour la mise à 0 de la bascule

Les sorties :Q et $\overline{\mathbf{Q}}$

Le shéma de la bascule R-S est le suivant :



Technologie NON-OU



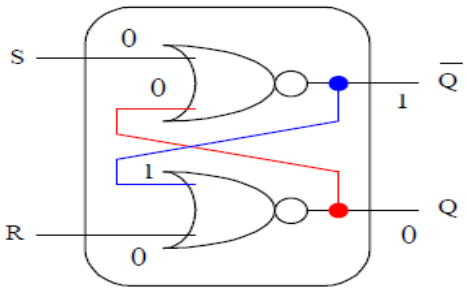
Technologie NON-ET

Pour comprendre le fonctionnement de la bascule R-S, on va étudier le comportement des variables de sortie (Q et $\overline{\mathbf{Q}}$) en fonction des variables d'entrée (R-S).

Pour cela, on désigne par :

- Q_t : la variable de sortie à l'instant t (l'état présent de la variable).
- Q_{t+1} : la variable de sortie à l'instant t+1 (l'état futur de la variable).

Pour un exemple : quand S=R=0 ; on suppose que Q_t=0. Dans ce cas, on aura le schéma suivant :



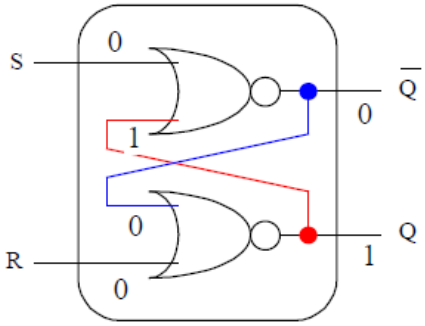
Ce schéma montre bien que, lorsque Q_t=0, la variable de sortie de la porte NOR supérieure est à 1 ($\overline{S + Q_t} = \overline{0 + 0} = 1$). Donc, on aura $\overline{Q_t}$=1.

Du moment que $\overline{Q_t}$=1, la variable de sortie de la porte NOR inférieure sera à 0 ($\overline{\overline{Q_t} + R} = \overline{1 + 0} = 0$) et on a Q_{t+1}=0.

C'est-à-dire $Q_{t+1}=Q_t$

On dit que l'état de la bascule stable.

- On ne supposant que Q_t=1. Dans ce cas, on aura le schéma suivant :



($S = 0; Q_t = 1 \rightarrow \overline{S + Q_t} = \overline{0 + 1} = 0$). Donc, on aura $\overline{Q_t}$=0.

$\overline{Q_t} = 0; R = 0 \rightarrow \overline{\overline{Q_t} + R} = \overline{0 + 0}$=1, et on aura :$Q_{t+1}$=1

Donc dans ce cas aussi, on a $Q_t = Q_{t+1}$.

On peut résumer le comportement de la bascule R-S par la table vérité suivante :

Etats présents / Etats futurs

R	S	Q_t	Q_{t+1}	
0	0	0	0	$Q_t = Q_{t+1}$
0	0	1	1	
0	1	0	1	$Q_{t+1}=1$ (Mise à 1)
0	1	1	1	
1	0	0	0	$Q_{t+1}=0$ (Remise à 0)
1	0	1	0	
1	1	0	X	Etats indéterminés
1	1	1	X	

Cette table est appelée, **table caractéristique** de la bascule R-S.

A partir de cette table, on peut déduire les expressions algébriques de Q_{t+1}et $\overline{Q_{t+1}}$.

Pour obtenir des fonctions simplifiées, on utilise une table de karnaugh :

SQ_t / R	00	01	11	10
0		1	1	1
1			ϕ	ϕ

L'expression simplifiée de Q_{t+1} est $Q_{t+1}=\bar{R}Q_t + S$.

Cette équation est appelée équation caractéristique de la bascule R-S

4.2.La bascule J-K :

La bascule J-K diffère de la bascule R-S du fait que quand les deux variables d'entrée passent simultanément à 1 l'état de la bascule n'est pas indéterminé.

En effet, quand J=K=1, on obtient la fonction de complémentation $Q_{t+1}=\overline{Q_t}$

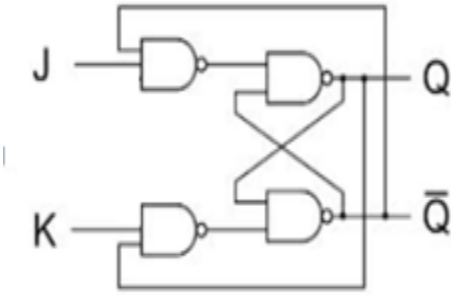


Schéma d'une bascule J-K

Tableau 10 :Table caractéristique de la bascule J-K :

J	K	Q_t	Q_{t+1}	
0	0	0	0	$Q_{t+1}=Q_t$
0	0	1	1	
0	1	0	0	$Q_{t+1}=0$ (Remise à 0)
0	1	1	0	
1	0	0	1	$Q_{t+1}=1$ (Mise à 1)
1	0	1	1	
1	1	0	1	$Q_{t+1}=\overline{Q_t}$
1	1	1	0	

A partir de la table de karnaugh, on obtient l'expression de Q_{t+1} dans le cas de la bascule J-K :

J \ KQ_t	00	01	11	10
0		1		
1	1	1		1

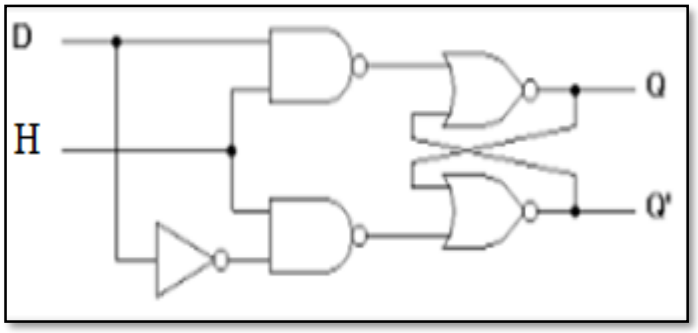
L'expression simplifiée de Q_{t+1} est $Q_{t+1}=\overline{K}Q_t + J\overline{Q_t}$.

A partir de cette expression, on peut élaborer la table suivante qui est d'une très grande utilité lors de réalisation d'un circuit séquentiel à base de bascule J-K.

Q_t	Q_{t+1}	J	K	
0	0	0	X	La valeur que peut prendre **K** n'a aucun effet sur l'état de la bascule.
0	1	1	X	
1	0	X	1	La valeur que peut prendre **J** n'a aucun effet sur l'état de la bascule.
1	1	X	0	

4.3.Bascule D :

Une autre manière de résoudre le problème d'ambiguité rencontrée ave la bascule R-S lorsque R=S=1, est de faire en sorte que ce cas ne se présente jamais à l'entrée de la bascule. Pour cela, on n'utilise seule variable d'entrée externe D et on parle alors de la bascule D. elle est illustrée par le schéma suivant :



La bascule D

La variable d'entrée de la porte AND inferieure étant égale à $\overline{D}$, les entrées internes de la bascule ne peuvent jamais identiques.

- Si D=1, alors la bascule passe à l'état 1.
- Si D=0, alors la bascule passe à l'état 0.

Il faut noter toutefois, que l'entrée D n'a d'effet que si H est à l'état 1.

D	Q_t	Q_{t+1}	
0	0	0	Q_{t+1}=D=0, D qui passe
0	1	0	
1	0	1	Q_{t+1}=D=1, D qui passe
1	1	1	

A partir de cette table, on peut déterminer l'expression algébrique de Q_{t+1}.

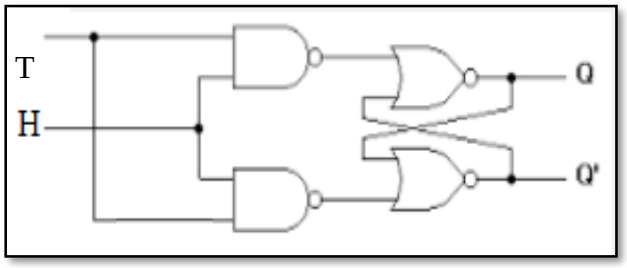
L'équation caractéristique de la bascule D

Q_t / D	0	1
0	0	0
1	1	1

L'expression simplifiée de Q_{t+1} est $Q_{t+1}=D$

4.4.Bascule T :

La bascule T ressemble assez à une bascule J-K à une seule entrée. Son schéma est le suivant :



La Bascule T

Chaque fois qu'une impulsion arrive, les états de la bascule (sorties) sont inversés (complémentés).

- Si T=0, alors, pas de changement.
- Si T=1, à ce moment- là, il y a complémentation des variables de sortie.

T	Q_t	Q_{t+1}	
0	0	0	$Q_{t+1}=Q_t$, pas de changement
0	1	1	
1	0	1	$Q_{t+1}=\overline{Q_t}$, complémentation
1	1	0	

Cette table nous permet de simplifier la fonction de l'état de sortie Q_{t+1} par la table de karnaugh, comme suit :

Q_t / T	0	1
0	0	1
1	1	0

Donc, L'expression simplifiée de Q_{t+1} est $Q_{t+1}=T\overline{Q_t}+\overline{T}Q_t=T\oplus Q_t$

5.Voici les tables d'excitation des différents types de Bascules :

5.1.Tables d'excitation de la Bascule R-S

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

5.2.Tables d'excitation de la Bascule J-K :

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

5.3.Tables d'excitation de la Bascule D :

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

5.4.Tables d'excitation de la Bascule T :

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

EXERCICES

Exercice N° :01:

1. A l'aide d'une bascule RS réaliser une bascule JK ?
2. A l'aide d'une bascule D réaliser une bascule JK ?

Exercice N° :02:

1. A l'aide d'une bascule RS réaliser une bascule D ?
2. A l'aide d'une bascule JK réaliser une bascule T ?

Exercice N°:03:

1. A l'aide d'une bascule JK réaliser une bascule RS ?
2. A l'aide d'une bascule T réaliser une bascule RS ?

LA SOLUSION

Exercice 01

A l'aide d'une bascule RS réaliser une bascule JK

D'après la table d'excitation de la bascule J K on peut construire cette table.

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

table d'excitation de la bascule J K

R	S	Q_t	Q_{t+1}	J	K
0	0	0	0	0	*
0	0	1	1	*	1
0	1	0	1	1	*
0	1	1	1	*	1
1	0	0	0	0	*
1	0	1	0	*	1
1	1	0	X	/	/
1	1	1	X	/	/

*** : 0 ou 1**

X : cas indéterminé.

.

RS / Q_t	00	01	11	10
0	*	*		*
1				1

$K = R\bar{S}$

RS / Q_t	00	01	11	10
0		1		
1	*	*		*

$J = \bar{R}S$

Avec les deux nouveaux fonctions logiques ,on peut réaliser la bascule J K

Exercice 02

A. A l'aide d'une bascule JK réaliser une bascule T :

D'après la table d'excitation de la bascule T on peut construire cette table.

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

Table d'excitation de la bascule T.

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

$T = KQt + J$

KQ_t / J	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Avec la nouvelle fonction logique ,on peut réaliser la bascule T

Exercice 03

B. A l'aide d'une bascule T réaliser une bascule RS

D'après la table d'excitation de la bascule T on peut construire cette table.

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

Table d'excitation de la bascule T.

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

table de karnaugh

Q_t / T	0	1
0	X	0
1	0	1

$J = TQ$

Q_t / T	0	1
0	0	X
1	1	0

$S = T\overline{Q}t$

CHAPITRE VI

CIRCUIT SEQUENTIEL 'COMPTEURS'

1. Objectif du chapitre

Le présent chapitre étudie et décrit le fonctionnement des compteurs asynchrones réalisés à partir de bascules qui ne changent pas d'état au même moment, puisqu'elles n'ont pas le même signal d'horloge. Ce chapitre est enchainé par l'étude des compteurs synchrones réalisés à partir de bascules synchronisées avec le même signal d'horloge. Des exemples de compteurs asynchrones et synchrones en circuits intégrés commercialisés sont étudiés pour donner au chapitre un intérêt pratique. A la fin de ce chapitre, l'étudiant sera en mesure de faire la différence entre un compteur asynchrone et synchrone et d'analyser et de synthétiser n'importe quel compteur.

2. Compteurs asynchrones

Un compteur asynchrone est constitué de plusieurs bascules en cascade. La première bascule reçoit le signal d'horloge CLK, la deuxième reçoit comme signal d'horloge le signal de sortie de la précédente et ainsi de suite. D'une manière générale, le signal d'horloge d'une bascule de rang i n'est autre que le signal de sortie de la bascule de rang i-1.

Compteur asynchrone à cycle complet (modulo 2^N)

Un exemple de compteur pouvant être réalisé de manière asynchrone est celui du compteur binaire qui compte de 0 jusqu'à 15.

Soient QD, QC,QB et QA les sorties des bascules utilisées pour réaliser un tel compteur. Le tableau ci-dessous montre l'évolution des différents états des bascules de ce compteur après chaque impulsion d'horloge. D'après la séquence de comptage à réaliser, on note que :

- La bascule de sortie QA doit changer sur chaque front de l'horloge.
- La bascule de sortie QB doit changer à chaque fois que la sortie QA de la bascule précédente passe de 1 à 0 (front)
- La bascule de sortie QC doit changer à chaque fois que la sortie QB de la bascule précédente passe de 1 à 0 (front)
- La bascule de sortie QD doit changer à chaque fois que la sortie QC de la bascule précédente passe de 1 à 0 (front).

Sorties du compteur					Sorties du compteur				
Qd	Qc	Qb	Qa		Qd	Qc	Qb	Qa	
0	0	0	0	0	1	0	0	0	8
0	0	0	1	1	1	0	0	1	9
0	0	1	0	2	1	0	1	0	10
0	0	1	1	3	1	0	1	1	11
0	1	0	0	4	1	1	0	0	12
0	1	0	1	5	1	1	0	1	13
0	1	1	0	6	1	1	1	0	14
0	1	1	1	7	1	1	1	1	15

3.FONCTIONNEMENT DES COMPTEURS/DECOMPTEURS

Il existe deux types de comptage/décomptage: BINAIRE ou DECIMAL, et deux modes de comptage/décomptage: SYNCHRONE ou ASYNCHRONE.

4.Compteurs binaires

Un compteur est dit "compteur binaire" lorsque le compteur effectue un comptage binaire, c'est à dire que l'ensemble des états logiques que peuvent prendre les sorties du compteur forme des mots ou des nombres binaires 11001101.

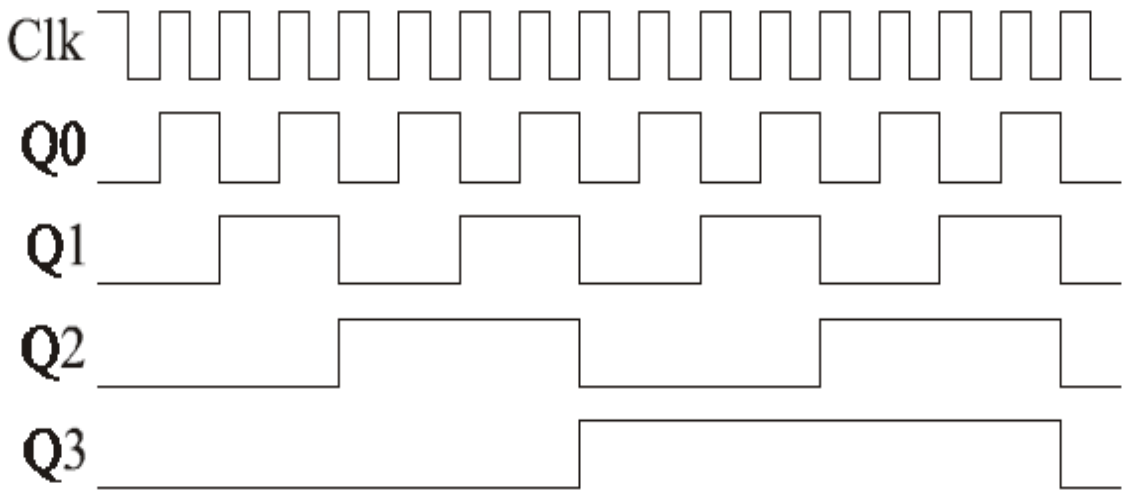
Exemple:

Si un compteur possède 4 sorties **Qa=Q0, Qb=Q1, Qc =Q2et Qd=Q3**, le nombre de code possible sera 24 = 16.

Le compteur peut compter jusqu'à 15: 0, 1, 2, 3, ..., 15.

Tableau des valeurs pour un compteur à 4 sorties.

4.1.Chronogrammes associés



5.Compteurs décimaux (compteur bcd)

Un compteur est dit "compteur décimal" ou "compteur BCD" lorsque le nombre de mots binaires possibles fournis par ses sorties est au plus de dix, cela signifie que le compteur ne pourra compter au-delà de la valeur 9, et donc la prochaine valeur correspondra à la valeur de départ 0.

Un compteur décimal possède 4 sorties binaires Qa,Qb, Qc et Qd comme un compteur binaire. Mais le nombre de codes possibles en sortie ne sera que de 10, du code 0000bin=0déc au code 1001bin=9déc.

Nota: Compteur BCD signifie Compteur Binaire Codé Décimal.

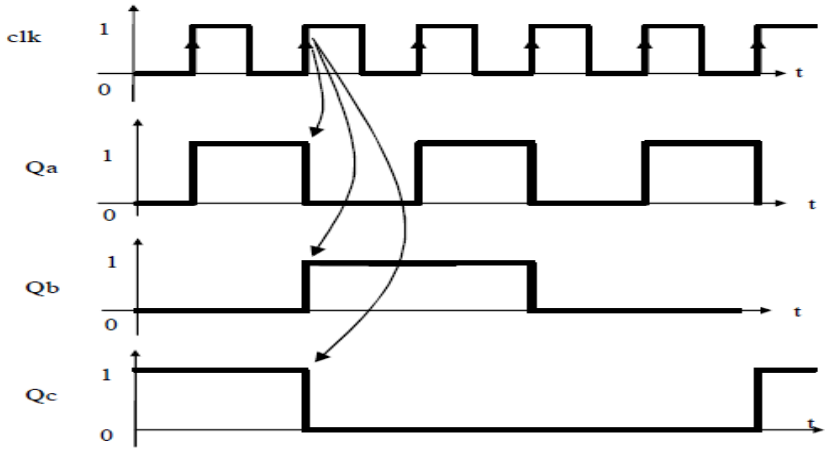
Tableau des valeurs pour un compteur BCD à 4 sorties.

Sorties du compteur					Sorties du compteur				
Qd	Qc	Qb	Qa		Qd	Qc	Qb	Qa	
0	0	0	0	0	0	1	1	0	6
0	0	0	1	1	0	1	1	1	7
0	0	1	0	2	1	0	0	0	8
0	0	1	1	3	1	0	0	1	9
0	1	0	0	4	0	0	0	0	0
0	1	0	1	5	0	0	0	1	1

6.Mode synchrone

Un "Compteur Synchrone" signifie que les bascules qui composent le compteur sont synchronisées par le même signal (signal d'horloge) et donc "basculent" au même instant. Par conséquence le changement d'état des différentes sorties (Qa, Qb, ..., Qn) que composent le compteur ne peut s'effectuer qu'à des instants identiques.

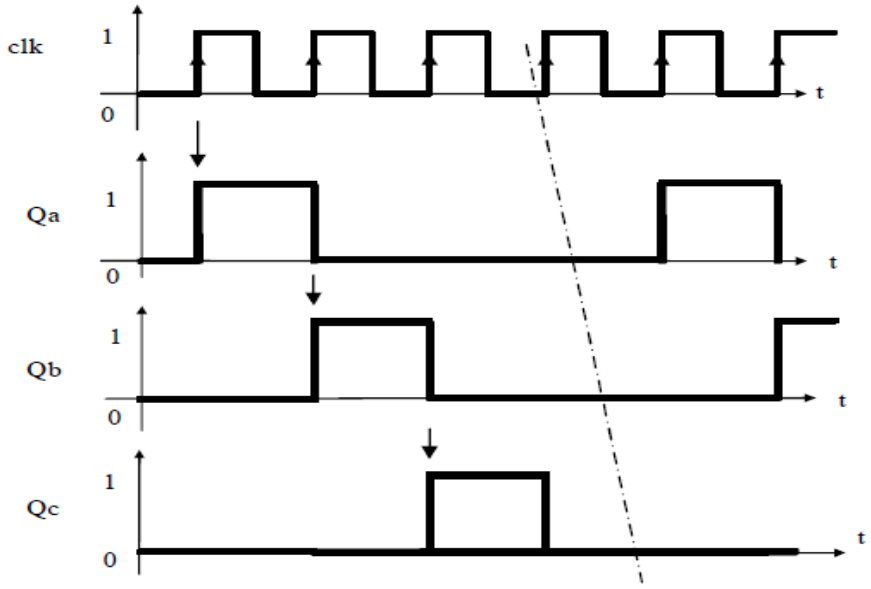
Chronogrammes.



7.Mode asynchrone

Un "Compteur Asynchrone" signifie que les "basculements" des bascules du compteur s'effectuent les uns après les autres. Le changement d'état de la sortie d'une bascule autorisera le changement d'état de la sortie de la bascule suivante et ainsi de suite. C'est un fonctionnement dit en cascade. La conséquence de ce type de fonctionnement est que le changement d'état des sorties du compteur ne s'effectue pas à des instants réguliers.

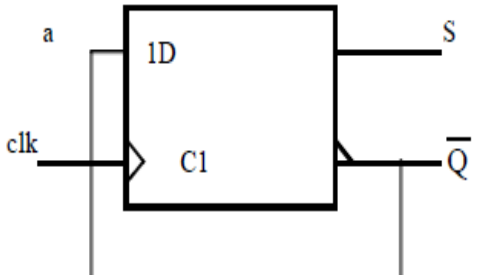
Chronogrammes :



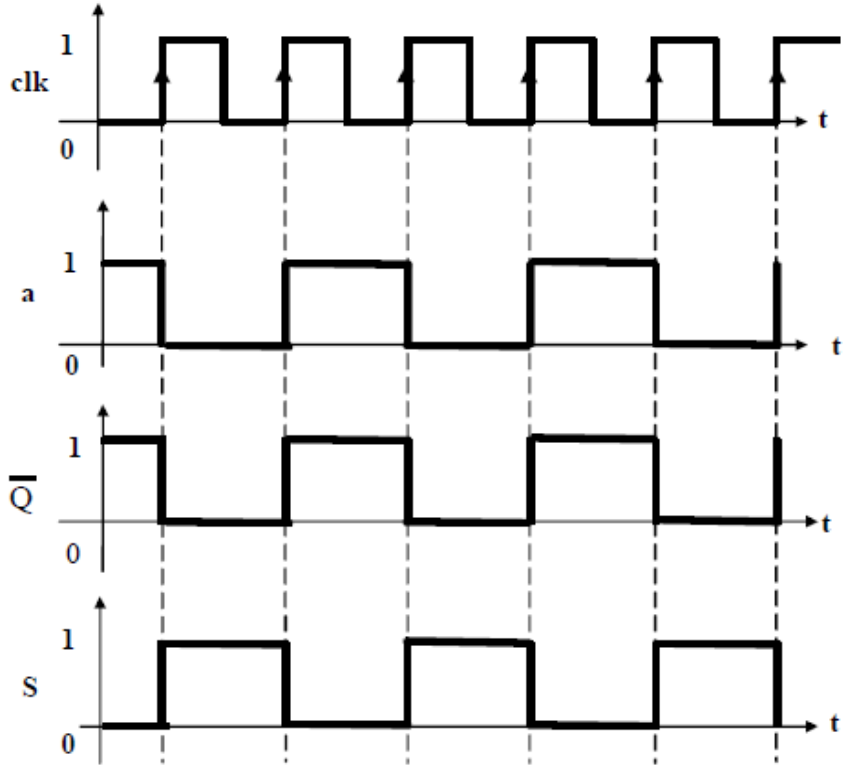
8.APPLICATIONS

8.1. Diviseur par 2 avec bascule d.

8.1.1. Schéma.

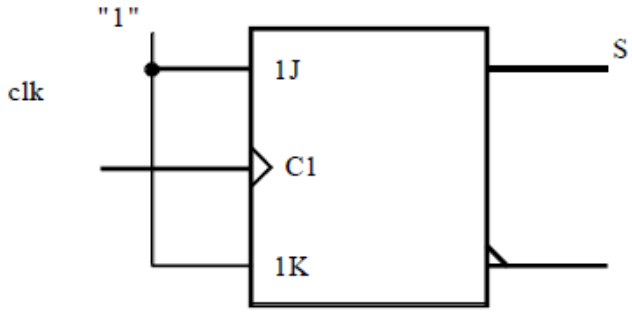


8.1.2Chronogrammes associes

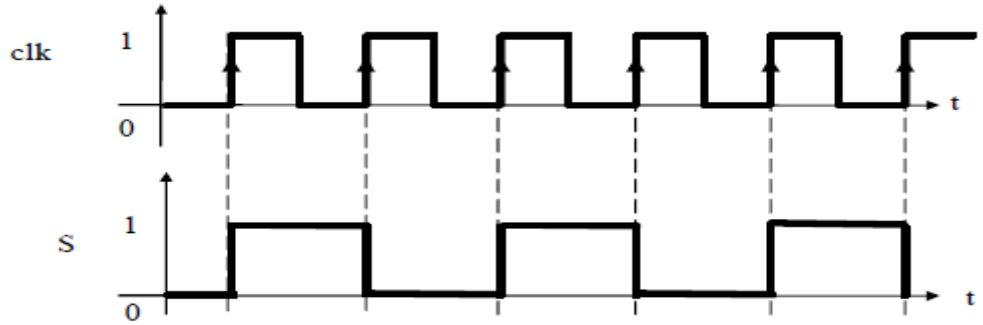


8.3.DIVISEUR PAR 2 AVEC BASCULE JK.

8.3.1.Schéma



8.3.2.Chronogrammes associes



9.Résumé sur les méthodes de conception des compteurs

On a vu que la méthode à employer pour concevoir un compteur dépend de la catégorie du compteur. On distingue déjà les compteurs synchrones des compteurs asynchrones. Si à première vue la réalisation d'un compteur asynchrone semble plus simple, ils'avère que la méthode de conception des compteurs synchrones permet une plus grande souplesse dans la réalisation de cycles évolués : ceci est dû au fait que les entrées des bascules constituant un compteur synchrone sont configurables, ce qui n'est pas le cas avec un compteur asynchrone.

Au sein d'une catégorie de compteurs, on trouve également des différences dans la démarche de conception selon la nature du cycle de comptage (modulo n ou $2n$, cycle régulier ou non, etc.).

TP N° :01et 02

Etude et réalisation des différents circuits intégrés

I. Le but de ce TP

- ✓ Appréhender et tester les différentes portes logiques.
- ✓ Apprendre les circuits combinatoires arithmétiques.
- ✓ Apprendre la structure de quelque circuit combinatoire souvent utilisés.
- ✓ Apprendre comment utiliser des circuits combinatoires pour concevoir d'autres circuits plus complexes.
- ✓ Réaliser des opérations arithmétiques de base (addition, soustraction).

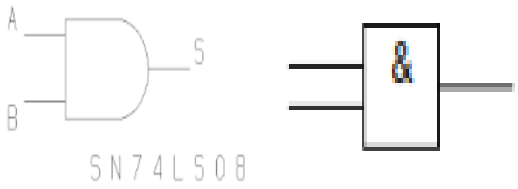
II. Matériels utilisés :

- ✓ Les circuits intégrés :74SL04 ; 74SL08 ; 74SL32 ; 74SL86.
- ✓ plaque d'essai électronique
- ✓ fils

III. Travail demandé

A. Réaliser les circuits suivants puis remplir les résultats de tests dans les tableaux suivant ?

ET(AND) **Opération booléenne**



Entrées		Sortie
A	B	
0	0	
0	1	
1	0	
1	1	

OU(OR)

Entrées		Sortie
A	B	
0	0	
0	1	
1	0	
1	1	

NON(NOT)

Entrée	Sortie
A	
0	
1	

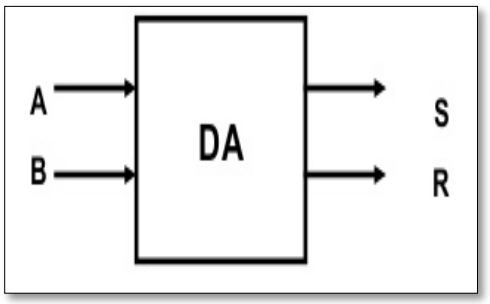
OU Exclusif (XOR)

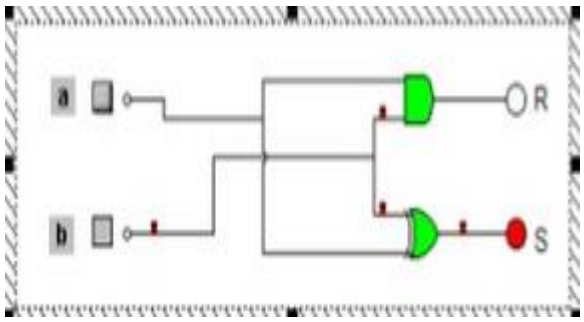
Entrées		Sortie
A	B	
0	0	
0	1	
1	0	
1	1	

B. Réalisations les circuits arithmétiques suivants et remplir les tableaux ci-dessous :

a. <u>Demi-additionneur</u>

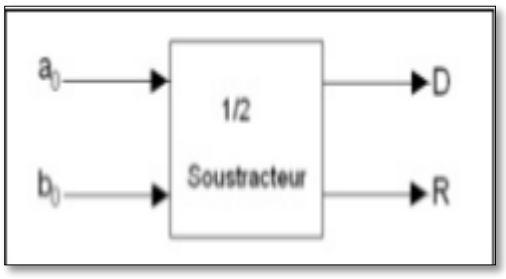
Un **additionneur** est un circuit logique permettant de réaliser une addition. Ce circuit est très présent dans les ordinateurs pour le calcul arithmétique mais également pour le calcul d'adresses, d'indice de tableau dans le processeur.





b. Demi-Soustracteur

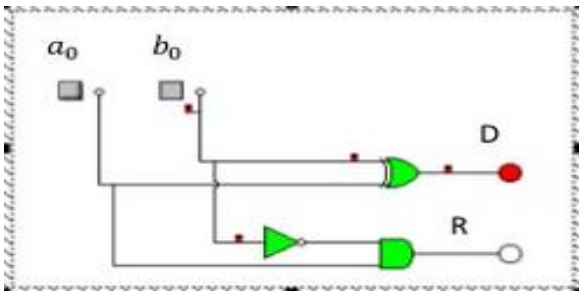
C'est un circuit capable de faire la soustraction de deux nombre binaires d'un bit chacun. Le circuit aura deux entrées A, B et deux sorties S et R.



Entrées		S	R
A	B		
0	0		
0	1		
1	0		
1	1		

S_{soustr}=…… …….

R_{soustr}=……



Entrées		S	R
A	B		
0	0		
0	1		
1	0		
1	1		

S_{addi}=……

R_{addi}=……

III. Conclusion (Comparaison entre les résultats théorique et pratique)

TP N° :03 et 04

(**Réalisation une fonction logique et Réalisation un comparateur)**

Le but de ce TP :

- ✓ Connaitre comment peut réaliser un circuit compatible à partir une fonction logique.
- ✓ Réaliser un comparateur avec plusieurs circuits de base.

Matériels utilises :

Les circuits :74..32 ;74..04 ;74..08 ;74..86.

Manipulation :

I. Soit la fonction logique suivante :

F(z,y,z)=$\bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$

- ✓ Simplifier cette fonction **F** et noté **F'** ?
- ✓ Réaliser la nouvelle fonction **F'** avec les circuits intégrés puis remplir le tableau ci-dessous ?

x	**y**	**z**	**F(théorique)**	**F'(pratique)**

II. Comparateur :

Le comparateur est un circuit arithmétique permettant de comparer deux nombres binaires A et B. A et B doivent voir la même longueur (nombre de bits). On cherche à savoir Si A>B,A<B ou A=B. on comprend donc que le circuit répond à une question à trois choix.

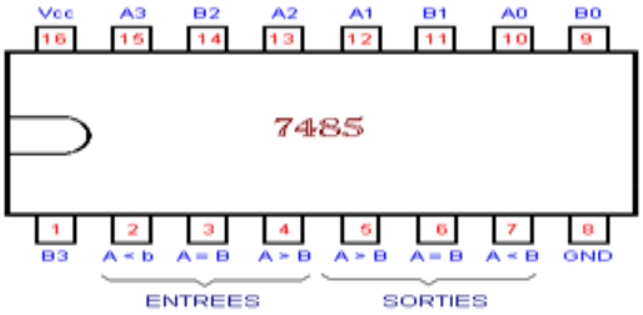


Fig. 1. - Brochage du circuit intégré 7485.

⊙ *Réalisation d'un comparateur de 2 nombres de 1 bit (a, b) :*

Soit 2 entrées a et b, et 3 sorties $\{S_0, S_1, S_2\}$.

$$: Avec : \begin{cases} S_0 = 1, & Si\ a = b \\ S_1 = 1, & Si\ a > b \\ S_2 = 1, & Si\ a < b \end{cases}$$

On peut dresser la table de vérité de ce circuit. On a 2 entées alors est possibilité { 2^2 } des résultats, résumé on table ci-dessous.

a	b	S_0	S_1	S_2
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

A partir cette table de vérité on peut déduire les expressions de ces circuits logique :

$$\begin{cases} S_0 = \overline{a \oplus b} = \overline{a\bar{b} + \bar{a}b} = \overline{S_1 + S_2} \\ S_1 = a\,\bar{b} \\ S_2 = \bar{a}\,b \end{cases}$$

❖ *Schéma de circuit comparateur :*

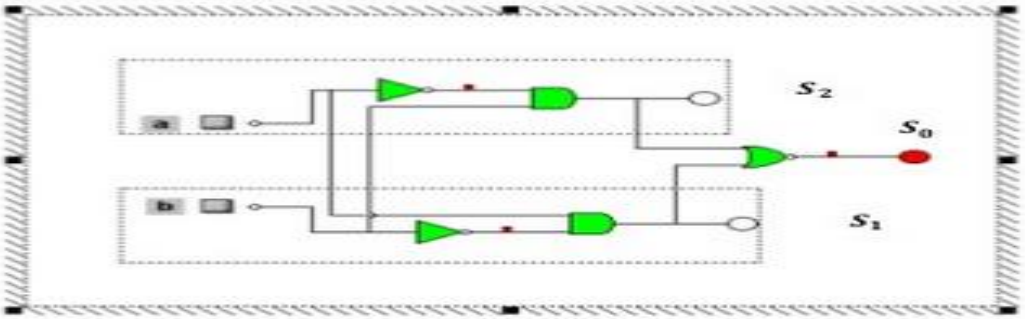


Figure_1 Schéma de circuit logique comparateur 2 bits

Réaliser ce circuit puis remplir cette table ?

a	b	S_0	S_1	S_2

Avec : S_0=égale

S_1=inferieure

S_2=supérieure

Conclusion ?

LES REFERENCES

1. « Cours d'électronique numérique » Camille Diou,Maître de Conférences Laboratoire Interfaces Capteurs et Microélectronique Université Paul Verlaine–Metz. Version du 24 février 2009.
2. « SYSTÈMES LOGIQUES I » Jean-François Harvey Mohamad Sawan Département de génie électrique et de génie informatique Quatrième édition école polytechnique de Montréal septembre 1999
3. « Introduction aux circuits logiques » LETOCHA ,Edition Mc-Graw Hill.
4. « logique combinatoire et technologie » M. GINDRE, Electronique numérique , Mc Graw Hill, 1987.
5. « logique combinatoire » A.BESSAID office des publications Universitaires 1, place centrale de ben Aknoun (alger).
6. « Algebre de Boole et Circuits Logiques » M.BELAID, Editions Universitaires ,carrefour 10102 Bouira.
7. INTERNET .