République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Dr. Tahar Moulay de Saida
Faculté de Technologie

Département d'Electrotechnique

Support de Cours

Logique Combinatoire et Séquentielle

Présenté par :

Dr. MOHAMMED CHIKOUCHE Tarik

Maître de conférences « A » en Electrotechnique

Mars 2023

Avant-propos

Ce polycopie de cours de logique combinatoire et séquentielle est à l'intention des étudiants de deuxième année licence Electrotechnique et Automatique dans le cadre des programmes officiels.

Cet ouvrage comprend six chapitres, le premier chapitre traitera les systèmes de numérations et codage de l'information. Le second chapitre entamera l'algèbre de BOOLE et simplification des fonctions logiques. Le troisième chapitre développera la technologie des circuits logiques intégrés. Le quatrième chapitre est dédié aux circuits combinatoires. Quant au cinquième chapitre traitera la partie séquentielle en commençant par les bascules. Finalement le dernier chapitre définira les différentes classes des compteurs.

Le minimum que l'on puisse dire de ce fascicule , c'est qu'il présente les notions d'une manière pédagogique et méthodique en se basant sur l'évaluation progressive. Il sera un cours type enseigné au département d'Electrotechnique, à l'université de Saida pour les étudiants en formation LMD en Electrotechnique et Automatique.

Enfin, nous espérons que le présent ouvrage aura le mérite d'être un bon support pédagogique pour l'enseignant et un document permettant une concrétisation théorique pour l'étudiant.

Dr. MOHAMMED CHIKOUCHE Tarik

Sommaire

Chapitre III
Technologie des Circuits Intégrés

Chapitre IV
Les Circuits combinatoires

Chapitre V
Les Bascules

Chapitre VI
Les Compteurs

Introduction générale

L'anglais Georges Boole (1815-1864) philosophe et mathématicien irlandais, eut l'idée de traduire le raisonnement logique ayant ses propre règles de calcul appelée depuis algèbre de Boole, algèbre logique et enfin algèbre binaire ; celle-ci s'avère actuellement le meilleur outil mathématique pour l'automatisation des machines et la programmation des ordinateurs.

Toute proposition ou fonction logique ne pouvant être que fausse ou vraie, on peut convenir de représenter par deux valeurs numériques '**0**' et '**1**'. Ainsi chaque variable ou grandeurs de Boole **a**,**b**,**c** etc.…ne peut prendre qu'une seul valeur (ou état) à la fois, '**0**' si elle est fausse, '**1**' si elle est vrai que nous conviendrons d'appeler 'Logique positive', l'inverse est appelé 'Logique négative'. L'algèbre de Boole est donc une algèbre de variables et de fonctions des deux valeurs numériques.

Concrètement, lors de la réalisation de circuits électroniques numériques, les 2 niveaux logiques sont constitués par 2 tensions différentes. La tension correspondant au niveau **0** est en général **0V**. La tension correspondant au niveau **1** dépend de la technologie utilisée. Une norme couramment répandue est la norme TTL :

niveau logique **0** ↔ **0** Volts
niveau logique **1** ↔ **5** Volts

Les opérateurs logiques de base et d'autres fonctions logiques plus évoluées existent sous forme de circuits intégrés.

Dans le domaine de la logique, on distingue en général deux grandes catégories, la logique combinatoire et la logique séquentielle. Dans les circuits combinatoires, les sorties sont déterminées uniquement en fonction des variables d'entrée. Le temps n'intervient pas dans les fonctions logiques.

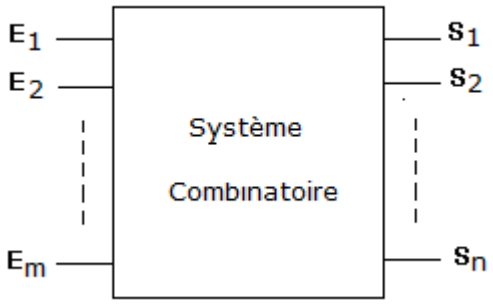


Figure. 1 : *Système Combinatoire à m entrées (E_1, E_2 ,…., E_m), n sorties (S_1 , S_2 ,….., S_m)*

Dans un système séquentiel, l'état de sortie ne dépend pas uniquement de la combinaison des entrées à un instant donné, mais aussi des valeurs passées d'entrées, de sorties ou des variables internes. Dans notre cas, il s'agira de valeurs précédentes des sorties. En plus de dépendre des entrées présentes, les sorties présentes seront exprimées en fonction de leur valeur précédente, disponible grâce à la présence d'éléments de mémorisation. Les systèmes séquentiels sont donc des systèmes bouclés. Le schéma suivant illustre ces propriétés :

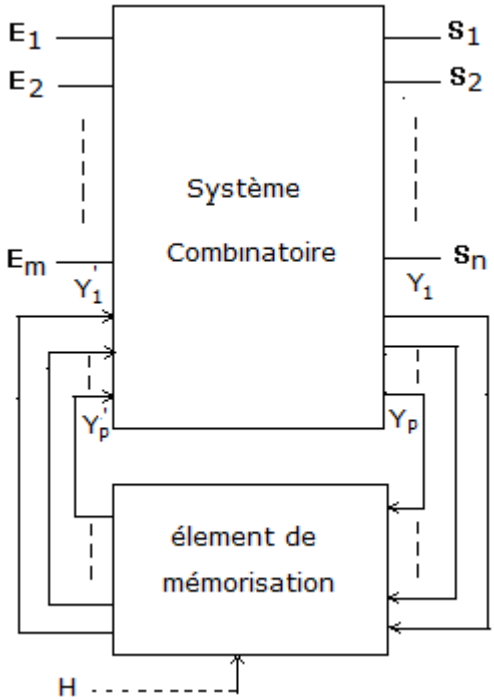


Figure. 2 : *Système Séquentiel à m entrées ($E_1, E_2, ..., E_m$), n sorties ($S_1, S_2, ..., S_n$) et p variables internes ($Y_1, Y_2, ..., Y_P$).*

Sur ce schéma, $y'_1, y'_2, ..., y'_p$ représentent les valeurs présentes de variables internes et $y_1, y_2, ..., y_p$ leur valeur passées. La notion de présent et de passé dépend du type de système séquentiel utilisé. Il existe les systèmes séquentiels synchrones et les systèmes séquentiels asynchrones.

Dans les systèmes séquentiels synchrones, c'est un signal d'horloge (une alternance, dans le temps, de 0 et de 1) qui définit le passage du passé au présent , la mise à jour des variables présentes $y'_1, y'_2, ..., y'_p$ se produit à chaque "coup" d'horloge. Ce signal H est représenté en pointillé sur le schéma de la figure (2).

Dans les systèmes séquentiels asynchrones, les éléments de mémorisation sont constitués par un retour direct. La mise à jour des variables présentes $y'_1, y'_2, ..., y'_p$ est donc quasi-instantanée après la mise à jour des variables passées $y_1, y_2, ..., y_p$.

Par exemple, un compteur est un système séquentiel qui ne possède comme entrée qu'un signal d'horloge. A chaque coup d'horloge, sa valeur de sortie binaire s'incrémente. A un instant donné, sa sortie dépend de sa valeur au coup d'horloge précédent.

Chapitre I

Systèmes de Numération et Codage de L'information

I.1./La représentation des nombres :

La fonction des systèmes de numération est de représenter les valeurs numériques le plus simplement et le plus clairement possible. En numération décimal en utilise dix signe de 0, 1,2,…….9, alors dans la numération binaire en utilise uniquement les deux signes '0' et '1'.

Exemple :

Décimale	0	1	2	3	4	5	6	7	8		
Binaire	0	1	01	11	100	101	110	111	1000		
Décimale	9	10	11	12	13	14	15	16	32	64	
Binaire	1001	1010	1011	1100	1101	1110	1111	10000	100.000	1000.000	

Le nombre décimale 1978 est une notation abrégée de :

$$1978 = 1000+900+70+8$$
$$= 1*1000+ 9*100+7*10+8*1$$
$$= 1*10^3+9*10^2+7*10^1 +8*10^0$$

On obtient la valeur numérique en multipliant chaque chiffre par son poids, et on additionnant les résultats obtenus. Cette forme s'appelle la forme polynomiale.

Comme le système décimal représente la numération en base '10', le système binaire représente la numération dans la base '2'. Plus généralement dans le système de numération de base B on a :

- Si N est un nombre entier :

$$N_B= (a_n a_{n-1}…..a_1)_B=a_nB^{n-1}+a_{n-1}B^{n-2}+…..+…..a_1B^0 \quad (1)$$

- Si N est fractionnel :

$$N_B=a_nB^{-1}+a_{n-1}B^{-2}+……+a_1B^{-n} \quad (2)$$

Exemples :

- Base 10 : Si $N=(365)=3.10^2+6.10^1+5.10^0$
  Si $N=0.365=3.10^{-1}+6.10^{-2}+5.10^{-3}$
- Base 2 : Si $N=101101101=1.2^8+0.2^7+1.2^6+1.2^5+0.2^4+1.2^3+1.2^2+0.2^1+1.2^0$

Remarque: On utilise les systèmes de numérations suivants :

- Base 10 avec les chiffres décimaux : 0,1,2,……..9.
- Base 2 avec les chiffres binaires : 0,1 (ou élément binaire ou bit)
- Base 8 avec les chiffres octales : 0,1,……7.
- Base 16 avec les chiffres hexadécimaux : 0,1,2…….9,A,B,C,D,E,F.

I.2/ Conversion d'un système de numération à un autre

A) Base B vers Base 10 :

Trois exemples suffiront pour expliquer la méthode :

A_1) Exemples entiers :

B=2, $(100101)_2=1.2^5+1.2^2+1.2^0=32+4+1=(37)_{10}$
B=8, $(23)_8=2.8^1+3.8^0=16+3=(19)_{10}$
B=16, $(A3D)_{16}=A.16^2+3.16^1+D.16^0=2560+48+13=(2621)_{10}$

A_2) Exemples fractionnelles :

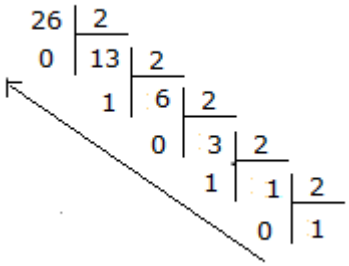
$(0.101)_2 = 1.2^{-1}+0.2^{-2}+1.2^{-3}$
$=1/2+0/4+1/8$
$=0.5+0+0.125= (0.3125)_{10}$
$(0.24)_8=2.8^{-1}+4.8^{-2}$
$=2/8+4/64$
$=0.25+0.0625= (0.3125)_{10}$
$(0.C8)_{16}=C.16^{-1}+8.16^{-2}$
$=12/16+8/256$
$=0.75+0.03125= (0.78125)_{10}$

B) Base 10 vers Base B :

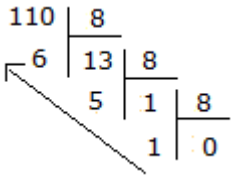
B_1) Nombres entiers : On divise le nombre (en système décimale) par la base B autant de fois qu'il est nécessaire, pour obtenir un quotient nul, le nombre s'obtient en relevant les restes de chaque division en partant de la dernière division vers la première (sens de lecture vers le haut).

Exemples :

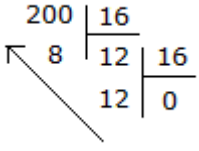
$(26)_{10}$ ⟶ $(11010)_2$



$(110)_{10}$ ⟶ $(156)_8$

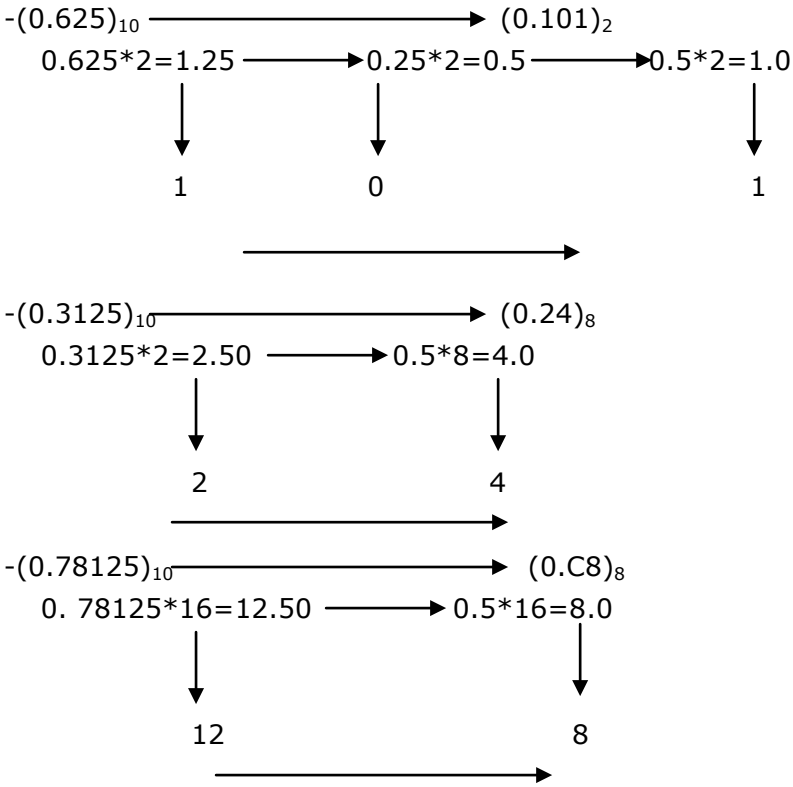


$(200)_{10}$ ⟶ $(C8)_{16}$



B_1) Nombres fractionnaires :'On multiplie plusieurs fois le nombre après la virgule par la base B. Avant chaque multiplication en supprime la partie entière du nombre précédent obtenu. La suite des parties entières supprimées écrites de gauche vers la droite, donne la fraction dans le système choisi. On s'arrête jusqu'à ce qu'il n'ait plus de partie fractionnaire ou que la précision obtenue soit jugée suffisante.

Exemples :



I.3/ Conversion direct entre système binaire, octal et hexadécimal

Les systèmes octal et hexadécimal, utilisent comme base de puissances de 2 ($8=2^3$,$16=2^4$), c'est pourquoi la conversion des nombres du système binaire en système octal ou hexadécimal est particulièrement facile.

Chiffres octales	Nombres Binaires
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Nombres binaires à trois digits (Triades)

Chiffres Hexadécimaux	Nombres Binaires	Chiffres Hexadécimaux	Nombres Binaires
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Nombres binaires à quatre digits (Tétrades)

- octal ⟶ binaire : chaque chiffre est remplacé par un nombre binaire à 3 digits

$(57)_8=(101111)_2$

- hexadécimal ⟶ binaire : chaque chiffre est remplacé par un nombre binaire à 4 digits

$(A4)_{16}=(10100100)_2$

- binaire ⟶ octal: le nombre binaire doit être divisé en triades à partir de la droite

A chaque triades correspond un chiffre.

$(110'010'101)_2=(625)_8$

- binaire ⟶ hexadécimal: le nombre binaire doit être divisé en tétrades à partir de

La droite. A chaque tétrade correspond un chiffre.

$(1001'1010')_2= (9A)_{16}$
$(111'1111)_2=(0111'1111)_2=(7E)_{16}$

I.4/ Conversion du système décimale vers code BCD (Décimal Codé Binaire) :

Le code BCD est utilisé pour les afficheurs lumineux, son principe repose sur le codage de chaque digit décimal (chiffre) en son équivalent en binaire sur 4 bits (et inversement).

Exemple 1 :

- $(127)_{10} = (0001\ 0010\ 0111)_{BCD}$
- $(850)_{10}= (1000\ 0101\ 0000)_{BCD}$

Exemple 2 : Convertir le nombre BCD 1011100000101 en décimal

- $(10\ 0111\ 0000\ 0101)_{BCD} = (0010\ 0111\ 0000\ 0101)_{BCD} = (2705)_{10}$

Remarque **:**

Les codes binaires compris entre (1010)2 et (1111)2 n'existe pas en BCD.

I.5/ Autres codes usuels :

I.5.1) 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.

Décimal	Code Binaire Naturel			
	A_8	A_4	A_2	A_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
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

I.5.2) Code Binaire réfléchi (Code Gray):

C'est le système de codage qui, contrairement au code binaire naturel est arrangé de Manière à ne faire changer d'état qu'une variable à la fois d'une ligne à l'autre. C'est Un code non pondéré, c'est-à-dire que les positions binaires ne sont affectées d'aucun poids.

Décimal	Code Binaire Naturel				Code Binaire Réfléchi			
	A_8	A_4	A_2	A_0	A_r	B_r	C_r	D_r
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0

13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

I.5.2.1) Conversion binaire naturel-binaire réfléchi:

On compare les bits Bn+1 et Bn du nombre écrit en binaire naturel:

- Reproduire le chiffre du poids le plus fort (gauche),
- Si Bn+1 et Bn ont même valeur, le chiffre correspondant en B.R est **0**,
- Si Bn+1 et Bn ont des valeurs différentes, le chiffre correspondant en B.R est **1**.

Exemple **:**

$A=(100110)_{2\ naturel} = (110101)_{2\ réfléchi}$

Naturel = 1 0 0 1 1 0

Réfléchi = 1 1 0 1 0 1

I.5.2.2) Conversion binaire réfléchi - binaire naturel:

- Reproduire le chiffre du poids le plus fort (gauche),
- Comparer le chiffre de rang n du binaire naturel à celui de rang n+1 du binaire réfléchi (on écrit **1** s'ils sont différents si non on écrit **0**).

Exemple **:**

$(11110)_{2\ réfléchi} = (10100)_{2\ naturel}$

Réfléchi = 1 1 1 1 0

Naturel = 1 0 1 0 0

I.5.3) Code ASCII (**A**mérican **S**tandard **C**ode **f**or **I**nformation **I**nterchange):

Un ordinateur doit être capable de traiter une information non numérique. C'est-à-dire il doit reconnaître des codes qui correspondent à des nombres, des lettres, des signes de ponctuation et des caractères spéciaux : Les codes de ce genre sont dit ***alphanumériques.***

Le code alphanumérique reproduit tous les caractères et les diverses fonctions que l'on trouve sur un clavier d'ordinateur, chaque caractère étant codé par un mot à **8** bits appelé **octet.** C'est un code utilisé pour communiquer entre le clavier d'un ordinateur et l'unité centrale. Le code ASCII standard est un code à 7 éléments, on peut donc représenter 2^7=128 groupe code contenu dans un tableau de conversion.

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
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

I.6/ Représentation des nombres signés en binaire:

Jusqu'a présent, nous n'avons parlé que de nombres positifs. Il peut s'avérer indispensable de traiter également des nombre négatifs. Le langage binaire ne connaît pas **le signe -** .

Il existe 3 conventions pour exprimer les nombres signés dans le système binaire:

- ✓ Représentation de la valeur et du signe indépendamment (binaire signé),
- ✓ Représentation en complément à 1,
- ✓ Représentation en complément à 2.

I.6.1/ Le binaire signé:

L'une des méthodes est de réserver **un bit pour indiquer le signe** du nombre, d'où l'appellation de binaire signé. Le bit réservé au signe est toujours le bit le plus à gauche. Pour le bit de signe et par convention, le **0** représente le **+** et le **1** le **-** .

Exemple**:**

- $(-23)_{10}$ = (1 0010111)
- $(+23)_{10}$ = (0 0010111)

I.6.2/ Complément à 1 d'un nombre binaire:

Pour calculer le complément à 1 (CA1) d'un nombre binaire, il suffit de complémenter chaque bit de ce nombre c'est-à-dire remplacé les 1 par des 0 et les 0 par des 1.

Attention: En notation signé, le bit de signe reste inchangé (ne pas complémenter).

I.6.3/ Complément à 2 d'un nombre binaire:

Pour calculer le complément à 2 (CA2) d'un nombre binaire N, on ajoute la valeur **1** au CA1 de N.

Exemple 1:

- Nombre binaire N (non signé) : N = 0101
- Complément à 1 de N : $(N)_{CA1}$ = 1010
- Complément à 2 de N : $(N)_{CA2}$ = 1011

Exemple 2:

- $(-23)_{10}$ = s'écrit (1 0010111) en binaire signé,
- Son complément à 1 est (1 1101000),
- Son complément à 2 est (1 1101001).

I.7/ Calcul arithmétique:

I.7.1/ Addition binaire:

L'addition de 2 nombres binaires est parfaitement analogue à l'addition de 2 nombres décimaux. Il faut commencer par le bit de poids le plus faible en utilisant l'algorithme suivant :

0 + 0 = 0
1 + 1 =10 (= 0 + report de 1 sur la gauche)

1 + 0 = 1
1+ 1 +1 = (= 1 + report de 1 sur la gauche)

Exemple : Faire l'addition suivante en binaire

15	1111	$^{1}1\ ^{1}1\ ^{1}1\ 1$
+ 11	+ 1011	+ 1 0 1 1
= 26	= 11010	= 11 0 1 0

I.7.2/ Soustraction binaire:

Dans le cas de la soustraction de deux nombres binaire non signés on peut utiliser L'algorithme suivant :

0 - 0 = 0
1 - 0 = 1

0 - 1 = 1 (avec report de 1 à retrancher au chiffre inferieur)
1 – 1 =0

Exemple : Opération 13 – 7

1101	$1\ ^{1}1\ ^{1}0\ 1$
- 0111	$-{}_{1}0\ {}_{1}1\ \ 1\ 1$
= 0110	= 0 1 1 0

I.7.3/ Multiplication binaire:

0*0 = 0

0*1 = 0

1*0 = 0

1*1 = 1

Exemple **:** Faire la multiplication suivante en binaire

```
        1110110
*       11011
---------------
        1110110
       1110110
      1110110
     1110110
```

$= (110001110010)_2$

I.7.3/ Division binaire:

0 / 1 = 0

1 / 1 = 1

Exemple **:** Faire la division suivante en binaire

```
110000000110 | 1110010
- 1110010    |--------
 10011100    |  11011
- 1110010    |
  10101011   |
- 1110010    |
   1110010   |
```

Chapitre II

Algèbre de Boole et Simplification des Fonctions Logiques

II.1/ Algèbre de Boole Logique

L'algèbre de Boole, ou calcul booléen, est la partie des mathématiques qui s'intéresse aux opérations et aux fonctions sur les variables logiques. C'est un cas particulier de l'algèbre de Boole des ensembles.

II.1.1/Axiomes:

A)Table de d'addition de deux variables Booliennes:

0+0=0 que l'on peut exprimer par une Table de vérité :

0+1=1

1+0=1

1+1=1

b \ a	0	1
0	0	1
1	1	1

Pour 3 variables : (a+b+c)=(a+b)+c

B)Table de multiplication de deux variables Booliennes:

0*0=0 que l'on peut exprimer par une Table de vérité :

0*1=0

1*0=0

1*1=1

b \ a	0	1
0	0	0
1	0	1

C)Complément d'une variable Boolienne:

On appelle '$\overline{a}$' le complément de la variable 'a', $\overline{a}$ se lit 'a barre'.

Si a=1, $\overline{a} = 0$

a=0, $\overline{a} = 1$

D'où la table de vérité :

a	$\overline{a}$
0	1
1	0

Remarque:

On peut alors écrire :

$$a + \overline{a} = 1 \quad (3)$$

$$a.\overline{a} = 0 \quad (4)$$

$$\overline{\overline{a}} = a \quad (5)$$

II.1.2/Propriétés de l'addition:

- Commutative : a+b=b+a
- Associative : (a+b)+c=a+(b+c)
- Distributive (a+b.c)=(a+b).(a+c)
- Admet '0' comme élément neutre: a+0=a
- L'addition de (+1) absorbe tous les éléments : a+1=1
- Non répétitive : a+a=a

II.1.3/Propriétés de la multiplication:

- Commutative : a.b=b.a
- Associative : (a.b).c=a.(b.c)
- Distributive a.(b+c)=(a.b)+(a.c)
- Admet '1' comme élément neutre: a.1=a
- Non répétitive : a.a=a

II.1.4/Absorption dans une égalité:

- $a \times b + a \times \overline{b} = a$ (6)

$$a \times b + a \times \overline{b} = a \times \left(b + \overline{b}\right) = a \times 1 = a$$

- $\begin{matrix} a + a \times b = a \\ a + a \times \overline{b} = a \end{matrix}$ (7)

$$a + a \times b = a \times (1 + b) = a \times 1 = a$$

II.1.5/Premier théorème de DEMORGAN :

'Le complément d'une somme logique de variables binaires est égal au produit des compléments de ses termes.'

$$\overline{a + b + c + ...} = \overline{a} \times \overline{b} \times \overline{c} ..... \quad (8)$$

Montrons que c'est vrai pour deux variables.

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

Il faut donc prouver que $(a + b) \times \overline{a}.\overline{b} = 0$ et que $(a + b) + \overline{a}.\overline{b} = 1$

- $(a + b) \times \overline{a}.\overline{b} = a \times \overline{a} \times \overline{b} + \overline{a} \times \overline{b} \times b = 0 \times \overline{b} + \overline{a} \times 0 = 0$

-

$$(a + b) + \overline{a}.\overline{b} = a + b + \overline{a} \times \overline{b} = a + a \times \overline{b} + b + \overline{a} \times \overline{b} = a + b + (a + \overline{a}) \times \overline{b} = a + b + 1 \times \overline{b} = a + b + \overline{b} = a + 1 = 1$$

Donc c'est vrai.

Supposons que la propriété est vraie pour n variables.

$$\overline{a_1 + a_2 + ... + a_n} = \overline{a}_1 \times \overline{a}_2 \times ..... \times \overline{a}_n$$

Montrons qu'elle est aussi vraie pour n+1 variables.

$$\overline{a_1 + a_2 + \ldots\ldots\ldots\ldots + a_n + a_{n+1}} = \overline{a_1 + a_2 + \ldots\ldots + a_n} \times \overline{a_{n+1}}$$

- Car c'est vrai pour deux variables.
- Comme c'est vrai pour n variables, alors vrai pour n+1 variables.

II.1.6/Deuxième théorème de DEMORGAN :

Enoncé : 'Le complément d'un produit logique de variables binaires est égal à la somme des compléments de ses facteurs.'

$$\overline{a \times b \times c} = \overline{a} + \overline{b} + \overline{c} \ldots\ldots\ldots \quad (9)$$

On peut déduire directement du premier théorème de DEMORGAN

$$\overline{a \times b \times c} = \overline{\overline{a} \times \overline{b} \times \overline{c} \ldots\ldots\ldots} = \overline{\overline{a} + \overline{b} + \overline{c} + \ldots\ldots\ldots} = \overline{a} + \overline{b} + \overline{c} \ldots\ldots\ldots$$

II.1.7/Règle de complémentation:

Elle résulte directement des théorèmes de DEMORGAN.

Enoncé : ' On obtient le complément d'une expression logique en remplaçant chaque variable par son complément, chaque signe $(+)$ par le signe $(\times)$ et chaque signe $(\times)$ par le signe $(+)$ sans oublier les parenthèses.'

Exemple :

Ecrire le complément de F :

$$F = \overline{a} \times \overline{b} + \overline{a} \times \overline{c} + a \times b + b \times c + a \times c$$

$$\overline{F} = \overline{\overline{a} \times \overline{b} + \overline{a} \times \overline{c} + a \times b + b \times c + a \times c}$$

Soit $\overline{F} = (a+b) \times (a+c) \times (\overline{a} + \overline{b}) \times (\overline{b} + \overline{c}) \times (\overline{a} + \overline{c})$

II.1.8/Conséquences:

- $a \times (a+b) = a$ (10)

En effet car : $a \times (a+b) = a \times a + a \times b = a + a \times b = a$

- $a + \overline{a} \times b = a + b$ (11)

Soit : $a + \overline{a} \times b = a + a \times b + \overline{a} \times b = a + (a + \overline{a}) \times b = a + b$

- $a \times b + \overline{a} \times c = a \times b + \overline{a} \times c + b \times c$ (12)

Soit : $a \times b + \overline{a} \times c = a \times b + a \times b \times c + \overline{a} \times c + \overline{a} \times b \times c = a \times b + \overline{a} \times c + (a + \overline{a}) \times b \times c$
$= a \times b + \overline{a} \times c + b \times c$

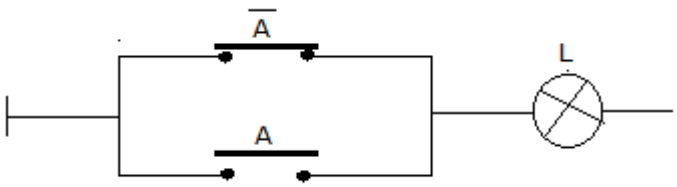
II.2/Différentes portes logiques (Opérations Booléenne):

II.2.1/La complémentation (Fonction Non ou Opérateur Non (NOT):

Soit un interrupteur A ouvert (A=0), la lampe L est éteinte (L=0). Ou l'interrupteur A fermé (A=1), la lampe est allumée.

La lampe L étant la sortie S, S=L, la fonction Non ou l'opérateur Non s'effectue ; ` la lampe L s'allume c'est-à-dire, la sortie S=1 si l'interrupteur n'est pas ouvert ou l'inverse.

S ou L est l'inverse de A ou bien 'non A' notée $\overline{A}$.



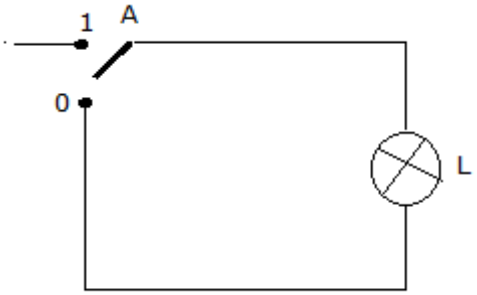
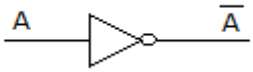


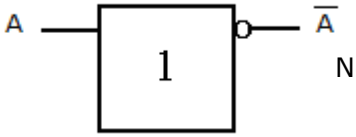
Table de vérité :

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

On représente ou on symbolise graphiquement l'opérateur 'Non' par :



Normes Américaines



Nouvelles Normes Européennes

Définition : La fonction 'Non' est la fonction dont la valeur est le complément de celle de la variable, elle est traduite par $S = \overline{A}$.

II.2.2/ Fonction ou Opérateur ET(AND):

On voit que la lampe L s'allume si les deux interrupteurs A et B sont fermés. C'est-à-dire A=1 et B=1, c'est donc L=S=A et B. On peut noter le produit L=S=A.B ou L'intersection L=S= $A \cap B$.

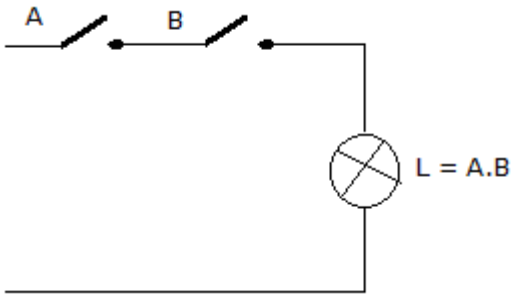
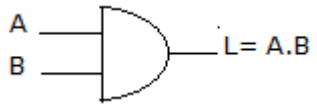


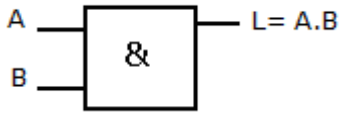
Table de vérité :

A	B	L= A.B
0	0	0
0	1	0
1	0	0
1	1	1

Le symbole logique est le suivant :



Normes Américaines



Nouvelles Normes Européennes

Définition : La fonction 'ET' est une fonction d'un nombre fini de variables binaires (A,B,C…..). Elle est égale à 1 si toutes les variables sont égales à 1, elle est égale à 0 dans tous les autres cas.

II.2.3/ Fonction ou Opérateur OU (OR):

La lampe L s'allume si les interrupteurs sont dans l'un des états suivants : (A fermé et B ouvert) ou bien (A fermé et B ouvert) ou bien (A et B fermés). On peut noter la somme L=A+B ou la réunion $L = A \cup B$.

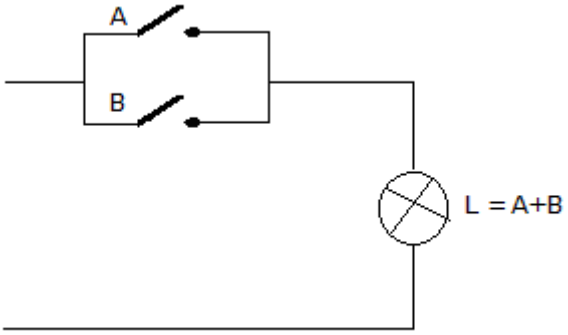
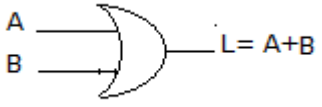


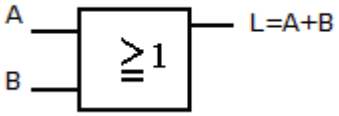
Table de vérité :

A	B	L=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Le symbole logique est le suivant :



Normes Américaines



Nouvelles Normes Européennes

Définition : La fonction OU (inclusif) est une fonction de variables binaires. Elle est égale à 1 si au moins une des variables est égale à 1. Elle est égale à 0 si toutes les variables sont nulles.

II.2.4/ Fonction Non-ET (NAND):

C'est le complément de ET.

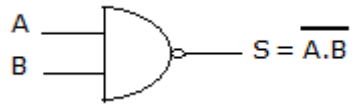
Table de vérité :

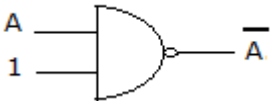
A	B	$S=\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

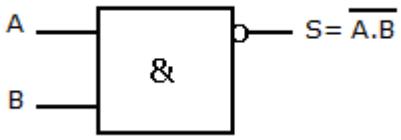
$A/B = \overline{A.B} = \overline{B.A} = \overline{A} + \overline{B}$ (DEMORGAN)

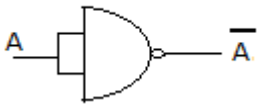
Remarque : $\overline{A.A} = \overline{A} + \overline{A} = \overline{A}$

Les symboles logiques sont les suivants :









II.2.5/ Fonction Non-OU (NOR):

C'est le complément de OU.

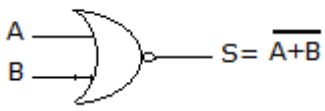
Table de vérité :

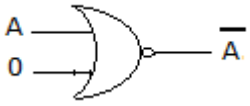
A	B	$S=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

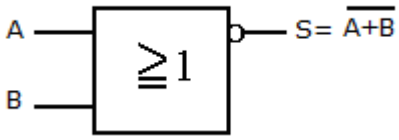
$A \downarrow B = \overline{A+B} = \overline{B+A} = \overline{A} \times \overline{B}$ (Commutative et DEMORGAN)

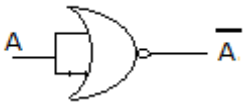
Remarque : $\overline{A+A} = \overline{A} \times \overline{A} = \overline{A}$

Les symboles logiques sont les suivants :









II.2.6/ Fonction OU Exclusif (XOR) :

La lampe L s'allume si l'un des interrupteurs est fermé. On note $A \oplus B$

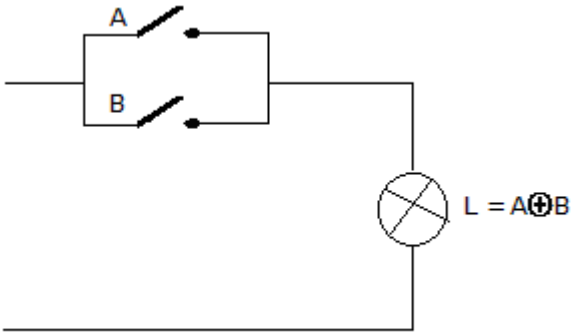


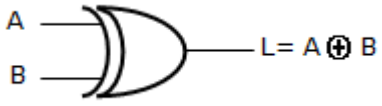
Table de vérité :

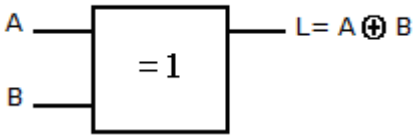
A	B	L= A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

$A \oplus B = 1$ Si $\overline{A}.B = 1$ ou $A.\overline{B} = 1$

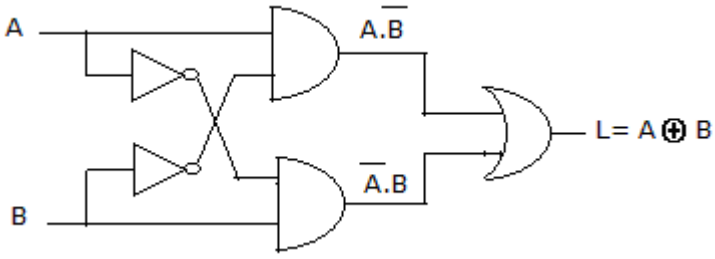
On écrit alors : $A \oplus B = \overline{A} \times B + A \times \overline{B}$

Le symbole logique est le suivant :





Remarque : Il est possible de représenter un ou exclusif à partir des fonctions Non ET, Ou.



Définition : C'est une fonction de variables binaires dont la valeur est 1 si et seulement si une variable est égale à 1.

II.2.7/ Le complément du OU Exclusif (Comparateur d'identité) :

Table de vérité :

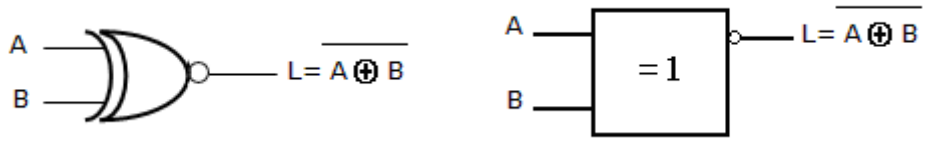
A	B	$L=\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

$\overline{A \oplus B} = 1$ Si $A \times B = 1$ et $\overline{A} \times \overline{B} = 1 \Rightarrow A = B = 1$ ou $A = B = 0$

Ce que l'on écrit : $\overline{A \oplus B} = A \otimes B = A \times B + \overline{A} \times \overline{B}$

Comparateur d'identité : $\overline{A \oplus B} = 1$ Si $A \equiv B$ (identique)

Le symbole logique est le suivant :



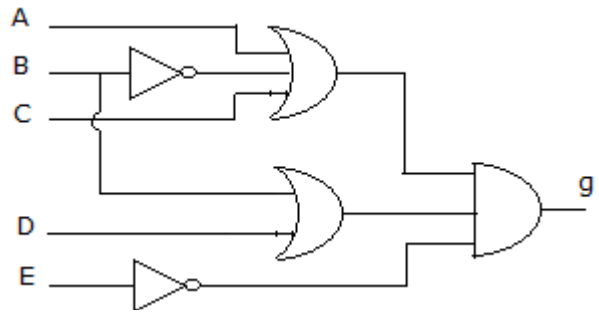
II.3/Réalisation des opérations logiques :

II.3.1/Opérateurs ET,OU,NON :

Ces opérateurs conviennent pour réaliser des fonctions mises sous les formes de produit de somme ou la somme des produits : les circuits ont trois couches de portes.

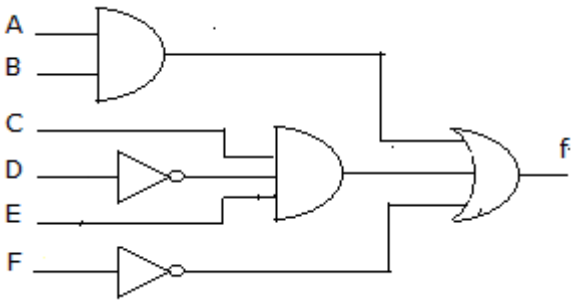
$$g = (A + \overline{B} + C).(B + D).\overline{E}$$

Le logigramme :



$$f = A.B + C.\overline{D}.E + \overline{F}$$

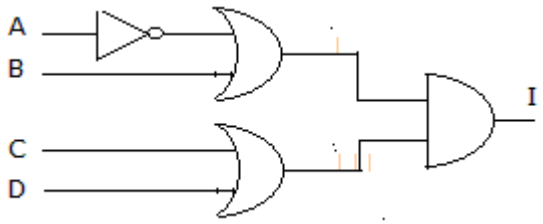
Le logigramme :



II.3.2/Opérateur NON.ET:

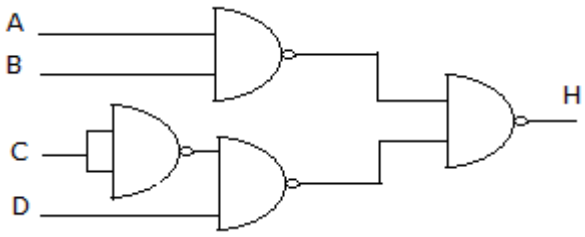
La fonction à réaliser doit être mise sous la forme d'une somme de produits :ET, Non et Ou ;

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



Uniquement avec les portes NON-ET :

$$\overline{\overline{H}} = H = \overline{\overline{A.B + \overline{C}.D}} = \overline{\left(\overline{A \times B}\right) \times \left(\overline{\overline{C} \times D}\right)}$$

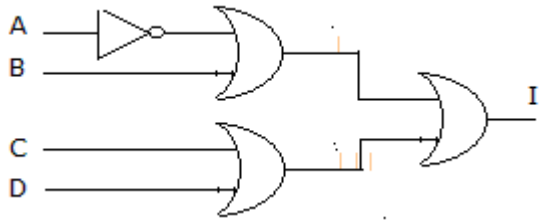


II.3.3/Opérateur NON-OU:

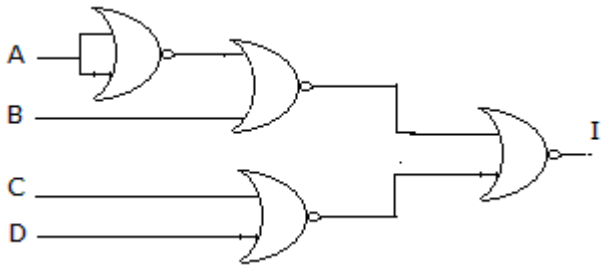
On met la fonction à réaliser sous forme d'un produit de sommes :OU, Non et ET ;

$$I = \left(\overline{A} + B\right)(C + D)$$

Uniquement avec les por



$$\overline{\overline{I}} = I = \overline{\left(\overline{A} + B\right)(C + D)} = \overline{\left(\overline{A} + B\right)} + \overline{(C + D)}$$



II.4/Synthèse des systèmes combinatoires (Représentation des fonctions Booléennes:

II.4.1/Développement d'une somme de produit (Forme canonique disjonctive):

Enoncé : 'Dans une somme de produit, chaque terme doit contenir chaque variable ou bien son complément. On obtient une somme de produit, on multipliant chaque terme par la somme de la variable manquante et de son complément, puis en suppriment les termes répétés après développement.'

Exemple1 :

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

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

$$X = \overline{A}BC\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}CD + A\overline{B}\overline{C}D + \cancel{AB\overline{C}D} + A\overline{B}\overline{C}D + ABC\overline{D} + A\overline{B}C\overline{D} \quad \left[B\overline{D} + \overline{B}D + B\overline{D} + \overline{B}\overline{D}\right]$$

Dans cet exemple, on obtient sept termes des 16 combinaisons (2^n combinaisons, n=4) possibles des quatre variables.

Ces fonctions ET qui contiennent toutes les variables logiques, sont appelles des **mintermes**. Chaque minterme ne prend la valeur '1' que pour une seule combinaison de la table de vérité, pour toutes les autres combinaisons, il a la valeur 0.

Exemple 2 : $Z = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

Nr	A B C	Z	Mintermes
0	0 0 0	0	
1	0 0 1	1	$\overline{A}\overline{B}C$
2	0 1 0	1	$\overline{A}B\overline{C}$
3	0 1 1	0	
4	1 0 0	1	$A\overline{B}\overline{C}$
5	1 0 1	0	
6	1 1 0	0	
7	1 1 1	1	ABC

Z peut donc être établi comme une fonction OU de fonctions logiques qui prennent respectivement la valeur '1' que pour une seule combinaison d'entrée (mintermes).

Remarque : La somme de produit est aussi appelle la **forme canonique disjonctive** ou **première forme canonique**.

II.4.2/Développement d'un produit de somme (Forme canonique conjonctive):

Enoncé : ' D'une manière similaire à la somme d'un produit, on obtient le produit d'une somme on additionne à chaque somme de variable, la variable manquante et son complément.

Exemple1 :

$$Y = \left(\overline{A} + \overline{B} + \overline{C}\right).\left(\overline{A} + \overline{C} + D\right).\left(A + C\right).\left(A + \overline{D}\right)$$
$$= \left[\left(\overline{A} + \overline{B} + \overline{C}\right) + D\overline{D}\right].\left[\left(\overline{A} + \overline{C} + D\right) + B\overline{B}\right].\left[\left(A + C\right) + B\overline{B} + D\overline{D}\right].\left[\left(A + \overline{D}\right) + B\overline{B} + C\overline{C}\right]$$

$$= (\overline{A}+\overline{B}+\overline{C}+D).(\overline{A}+\overline{B}+\overline{C}+\overline{D}).(\overline{A}+B+\overline{C}+D).(\overline{A}+\overline{B}+\overline{C}+D).[(A+B+C).(A+\overline{B}+C)+D\overline{D}].[(A+B+\overline{D}).(A+\overline{B}+\overline{D})+C\overline{C}]$$
$$= (\overline{A}+\overline{B}+\overline{C}+D).(\overline{A}+\overline{B}+\overline{C}+\overline{D}).(\overline{A}+B+\overline{C}+D).(\overline{A}+\overline{B}+\overline{C}+D).[(A+B+C+D).(A+B+C+\overline{D}).(A+\overline{B}+C+D).(A+\overline{B}+C+\overline{D})]$$
$$.[(A+B+C+\overline{D}).(A+B+\overline{C}+\overline{D}).(A+\overline{B}+\overline{C}+\overline{D}).(A+\overline{B}+\overline{C}+\overline{D})]$$

On obtient alors 9 sommes parmi les 16 combinaisons possibles. Ces fonctions qui contiennent toutes les variables logiques sont appelées les **maxtermes.** Chaque maxterme ne prend la valeur '0' que pour la combinaison de la table vérité pour toutes les autres combinaisons il a la valeur '1'.

Exemple 2 : $W = (A+B+C).(A+\overline{B}+\overline{C}).(\overline{A}+B+\overline{C}).(\overline{A}+\overline{B}+C)$

Table de vérité:

Nr	A B C	W	Maxtermes
0	0 0 0	0	$(A+B+C)$
1	0 0 1	1	
2	0 1 0	1	
3	0 1 1	0	$(A+\overline{B}+\overline{C})$
4	1 0 0	1	
5	1 0 1	0	$.(\overline{A}+B+\overline{C})$
6	1 1 0	0	$.(\overline{A}+\overline{B}+C)$
7	1 1 1	1	

La sortie W ne prendra la valeur '0' que pour la combinaison d'entrée 0,3,5 ou 6(pour toutes les autres W=1).

Remarque : Le produit de somme est aussi appelle la **forme canonique conjonctive** ou **deuxième forme canonique**.

II.5/Simplification des fonctions logique par le diagramme de KARNAUGH:

II.5.1/Introduction:

La méthode de simplification des fonctions logiques par le diagramme de Karnaugh est une méthode graphique dont on va expliquer l'emploi de cette méthode dans ce paragraphe.

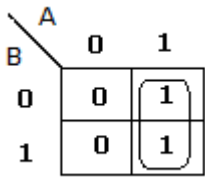
II.5.2/Termes adjacents et diagramme de Karnaugh:

Deux termes sont adjacents quand ils différent l'un de l'autre par une variable logique.

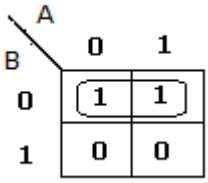
Exemple : $A\overline{B}C$ et ABC
101 111

La méthode de Karnaugh consiste à représenter graphiquement les combinaisons d'entrée de façon à ce que toutes les combinaisons d'entrée adjacentes soient regroupées. Les variables sont disposées dans un tableau Π_{xi}, Π_{yi}, les états des variables X_i ou Y_i sont disposés dans des cases, le passage d'une case à l'autre se fait de telle manière qu'on ne change qu'une seule variable à la fois.

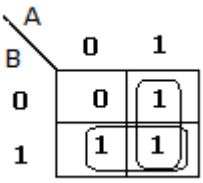
II.5.3/Fonctions à deux variables:



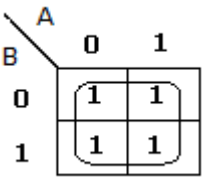
$F = A$



$F = \overline{B}$



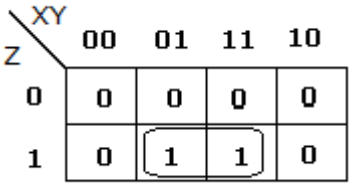
$F = A + B$



$F = 1$

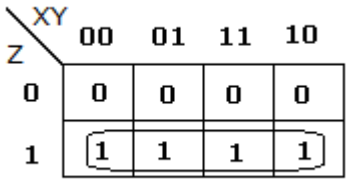
II.5.4/Fonctions à trois variables:

Exemple1 : $F = \overline{X}YZ + XYZ$



$F = XY$

Exemple2 : $F = \overline{X}.\overline{Y}Z + \overline{X}YZ + XYZ + X\overline{Y}Z$



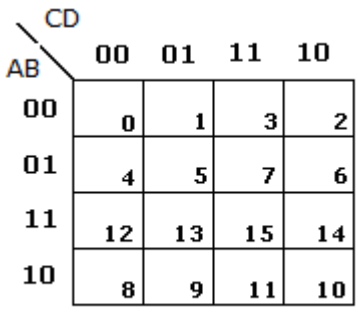
$F = Z$

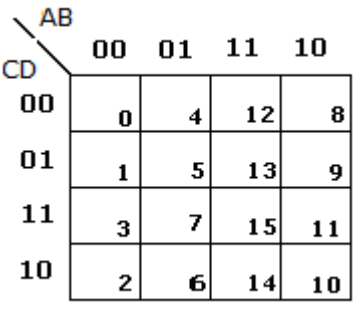
II.5.5/Fonctions à quatre variables:

Pour remplir un diagramme de Karnaugh à plus de 03 variables, il vaut mieux numéroter les cases ainsi que nous allons l'expliquer dans le cas de 04 variables.

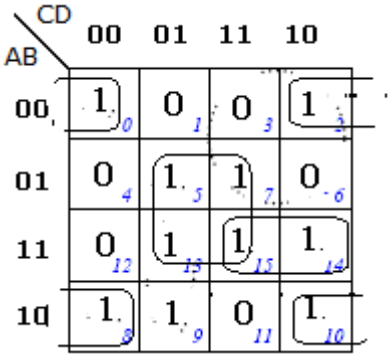
II.5.5.1/Numérotation des cases:

Le numéro de chaque case est l'équivalent décimal du nombre binaire (DCBA ou ABCD) avec le poids respectifs $2^3,2^2,2^1,2^0$. Ainsi $D\overline{C}\overline{B}A$ ou $A\overline{B}\overline{C}D$ signifie 9 et on met le 9 dans la case 1001.





Exemple1 : $F(A,B,C,D) = \sum(0,2,5,7,8,10,11,14,15)$

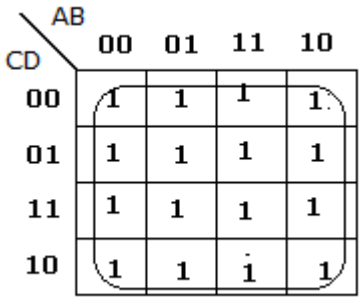


$F = BD + \overline{B}\overline{C} + ABC$

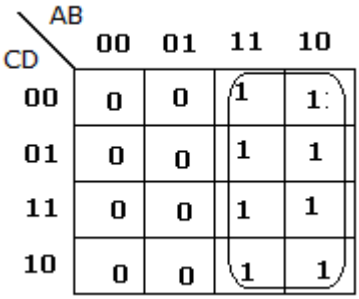
I.5.5.2/Simplification de la fonction:

- 8 cases expriment un produit d'une variable,
- 4 cases expriment un produit de deux variables,
- 2 cases expriment un produit de trois variables,
- 1 case exprime un produit de quatre variables.

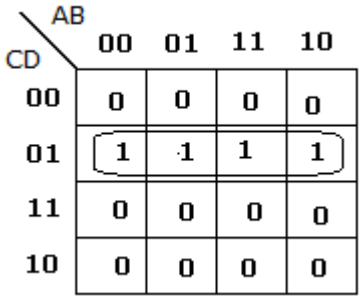
Exemples :



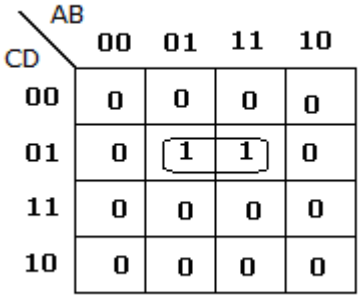
$F = 1$



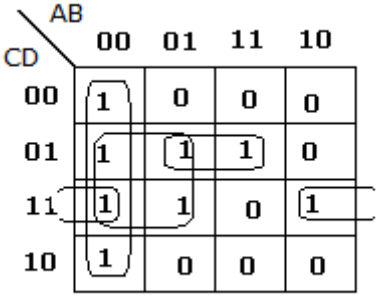
$F = A$



$F = \overline{C}D$



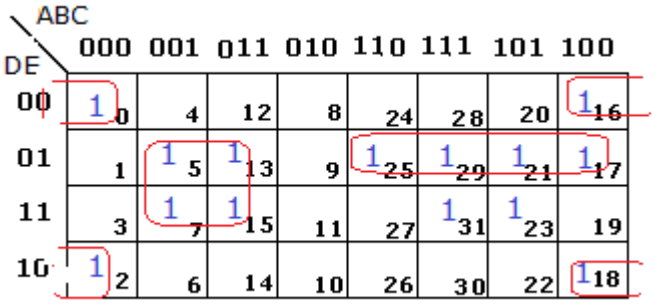
$F = B\overline{C}D$



$F = A.B + A.D + BCD + BCD$

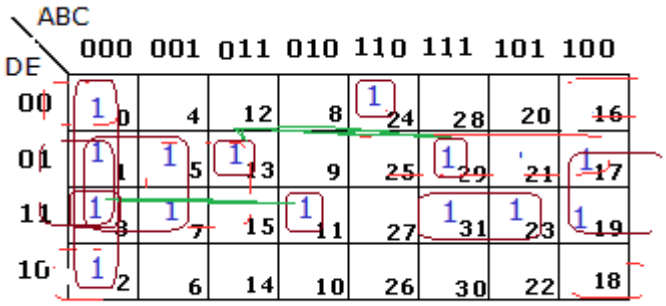
II.5.6/Fonctions à cinq variables:

Exemple1 :



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

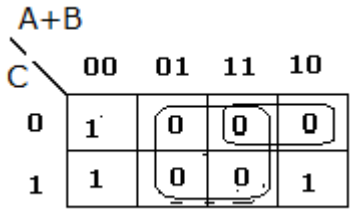
Exemple2 :



$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}E + \overline{B}\,\overline{C}E + \overline{A}\,\overline{C}DE + BC\overline{D}E + ACDE + AB\overline{C}\,\overline{D}\,\overline{E}$$

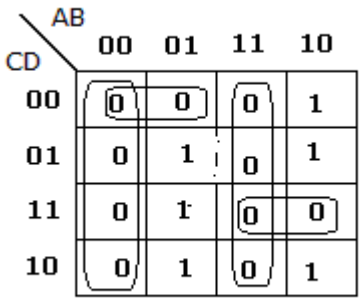
II.5.7/Simplification desMaxtermes:

II.5.7.1/ Fonctions à trois variables:



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

II.5.7.2/ Fonctions à quatre variables:

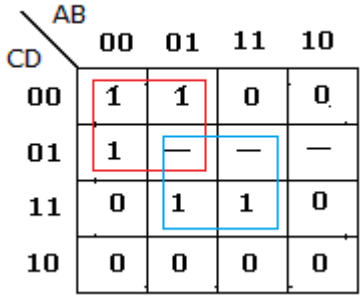


$$F = (A + B).(A + \overline{B}).(A + C + D).(\overline{A} + C + \overline{D})$$

II.5.8/ Présence d'états indifférents (interdites) :

Quand certaines combinaisons des variables sont sans effet sur la valeur de la fonction F, on dit que ce sont des états indifférents. On les notes par le signe – ou Φ (0 ou 1) dans le diagramme de Karnaugh et les utilise partiellement ou totalement pour simplifier F.

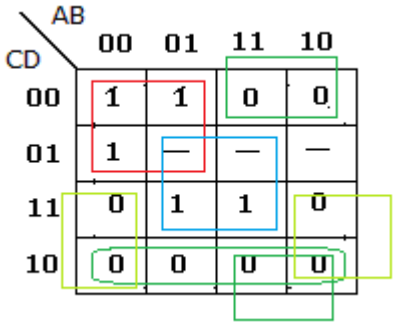
Exemple1 :



$$F = \overline{A}\,\overline{C} + BD$$

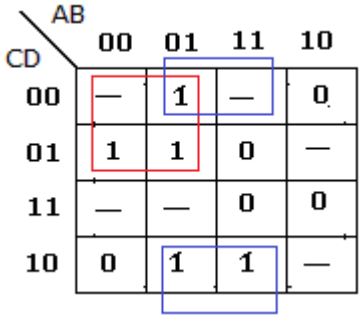
Intersection minimale F_{min} ?

Trouver d'abord $\overline{F}$



$$\overline{F} = A\,\overline{D} + C\,D + B\,C \Rightarrow F = F_{min} = (A + D)(C + D)(B + \overline{C})$$

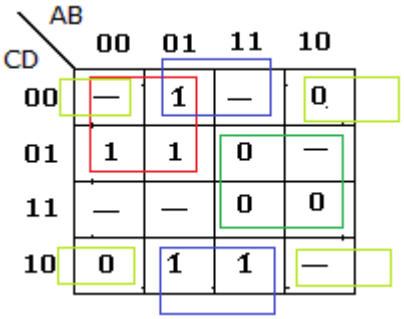
Exemple2 :



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

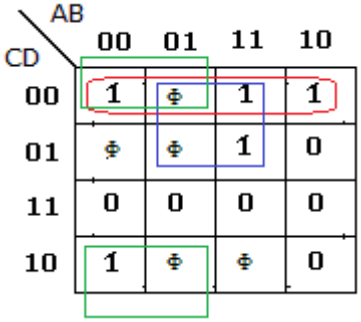
Intersection minimale F_{min} ? (Produit de somme)

Trouver d'abord $\overline{F}$



$$\bar{F} = \bar{B}\,\bar{D} + AD \quad \bar{\bar{F}} = F_{min} = (B + D)\left(\bar{A} + \bar{D}\right)$$

Exemple3 : $F(A, B, C, D) = \sum_1 (0, 2, 8, 12, 13) + \sum_\Phi (1, 4, 5, 6, 14)$



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

Chapitre III

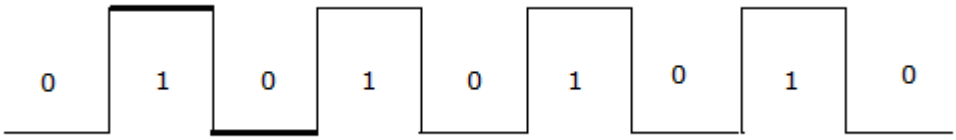
Technologie des Circuits Intégrés

III.1/Les familles de circuits intégrés:

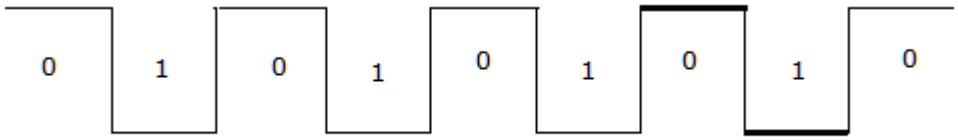
En logiques électronique plusieurs familles de circuits intégrés permettent d'obtenir les fonctions logiques désirées ; NON ;ET ;OU ; NON ET ;NON OU ce sont les familles :

DL : Diode Logic ; **RTL :** Résistance Transistor Logic ; **DTL :** Diode Transistor Logic ; **TTL :** Transistor Transistor Logic ;……..

Ces fonctions accomplissent des commutations entre deux niveaux : un niveau bas et un niveau haut par convention, on désigne par logique positive, la logique dont le niveau haut ou « **1** » logique correspondant à la tension la plus positif et le niveau bas ou « **0** » logique correspond à la tension la plus négative (le plus souvent **0**V).



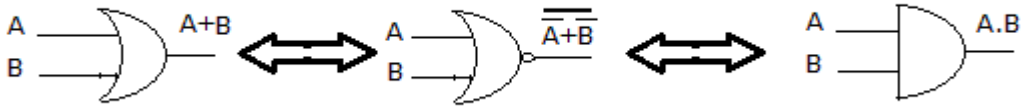
Logique positive



Logique négative

Dans les circuits modernes, on représente le plus souvent le niveau bas ou **0** par la tension nulle et le niveau haut par une tension positive (Vcc). Changer de logique pour un circuit particulier équivaut à inverser les entrées et la sortie. Le théorème de Demorgan permet alors de déterminer la nouvelle fonction logique.

Exemple :



En logique positive | En logique négative | Par Demorgan en logique négative

III.1.1/Portes a diode et a Transistors:

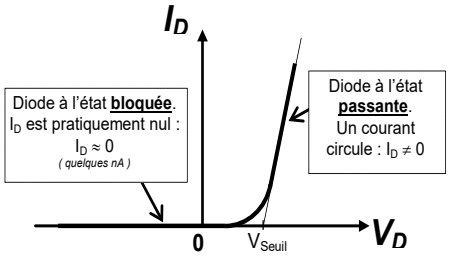
Les portes passives n'amplifient pas les signaux et d'utilisent que des diodes. Les portes actives comportent des transistors qui fonctionnent en commutation (saturé ou bloqué).

III.1.1.1/Logique à diodes (DL):

Cette famille de circuits intègres ne permet de fabriquer que deux types de portes : porte OU ET la porte ET.

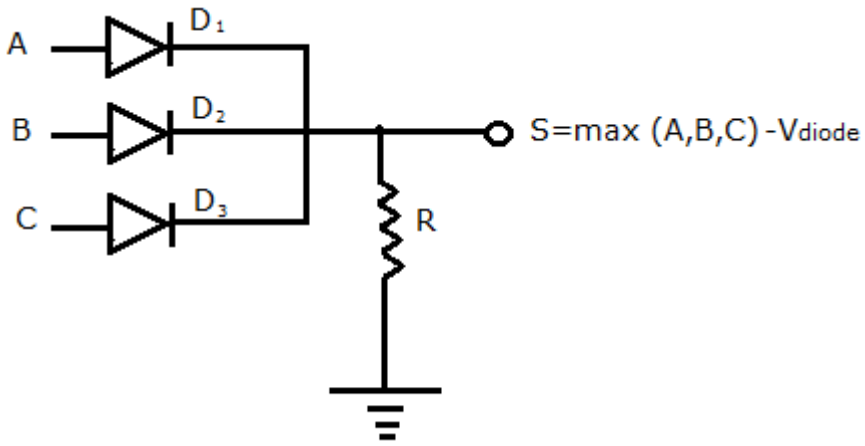
a) Porte 'OU' :

La caractéristique d'une diode est la suivante :



- ✓ Elle est passante si $V_{Anode} \rangle V_{Cathode}$
- ✓ Elle est bloquée dans le cas contraire, le seuil V_{Seuil}=0.7V pour le silicium.
- ✓ Elle se comporte donc comme un interrupteur fermé dans le sens direct (passant) et ouvert dans le sens inverse (bloqué).
- ✓ L'interrupteur est ouvert si la tension appliquée est inferieur au seuil, il est fermé si la tension est supérieur au seuil V_{Seuil}=0.7V.

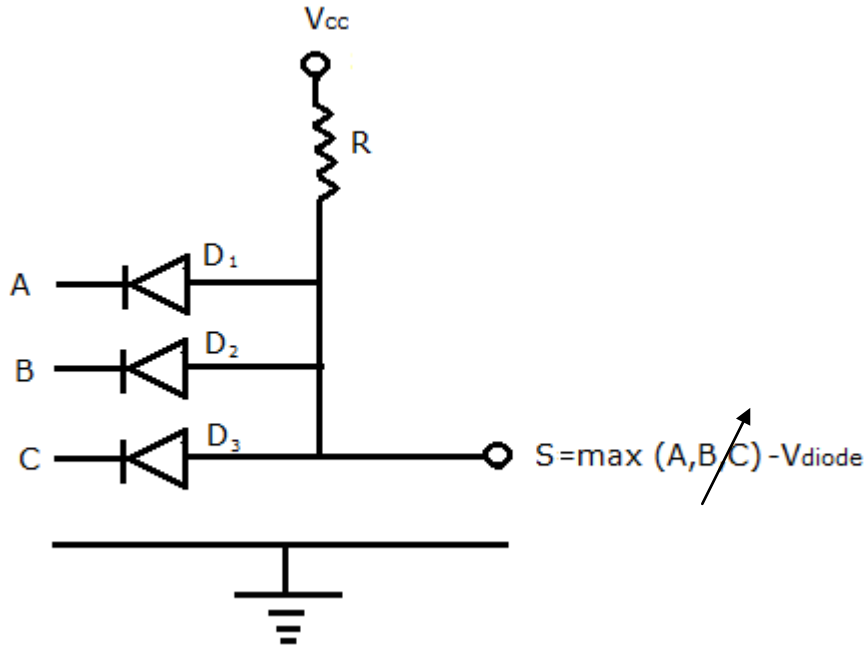
Soit la porte OU :



Pour que la diode soit passante, il faut que $V_{cc} \geq V_{diode}$

b) Porte 'ET' :

La sortie $S = V_S$ est nulle, cela veut dire que la diode conduit si au moins l'une des entrées est mise à la masse, elle sera égale à V_{cc} dans l'autre cas.

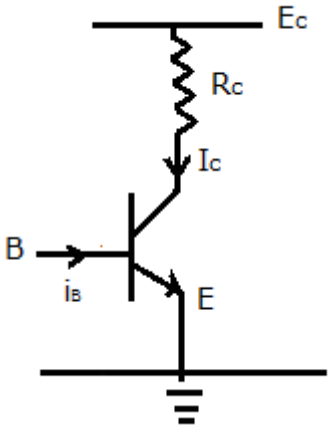


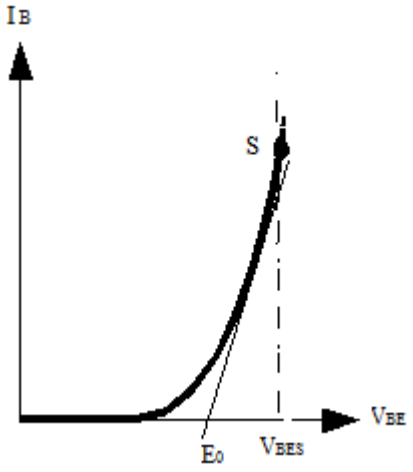
Pourque $S = V_{cc}$, il ne faut pas qu'une diode conduise c'est-à-dire $A = B = C = V_{cc}$ et $V_D = 0$

III.1.1.2/Logique à diodes et à Transistor :

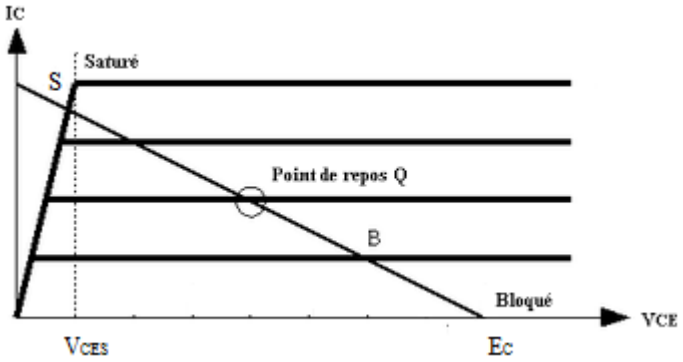
a) Fonction 'NON' :

Le transistor branché en Emetteur commun possède un régime de commutation de deux états possible. L'état bloquée et l'état saturé ce qui définit la fonction NON.

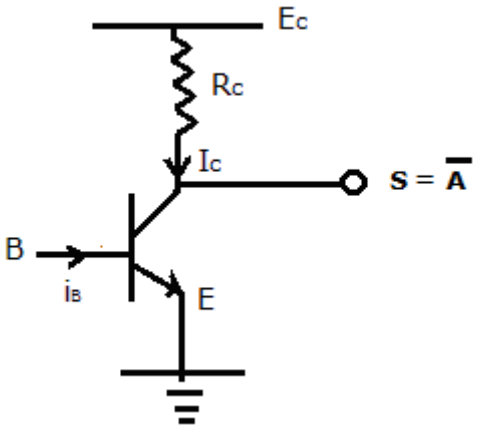




-Si $V_{BE} \langle E_0$, $I_B = 0$ (courant de base est nul) $\Rightarrow$ le transistor est bloqué, le point de fonctionnement se trouve sur le point B.

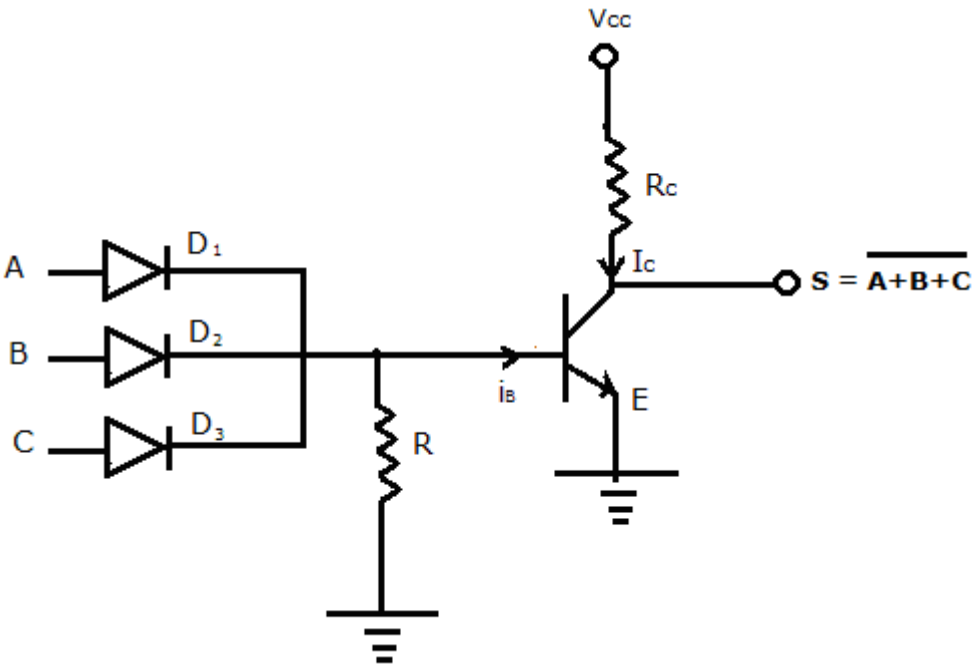


- $V_{CE} = E_C - R_C I_C$
- Si $V_{BE} \rangle E_0$, I_B commence à circuler, $I_C \uparrow$ et $V_{CE} \downarrow$, le point de fonctionnement et sur le point S(partie saturée).



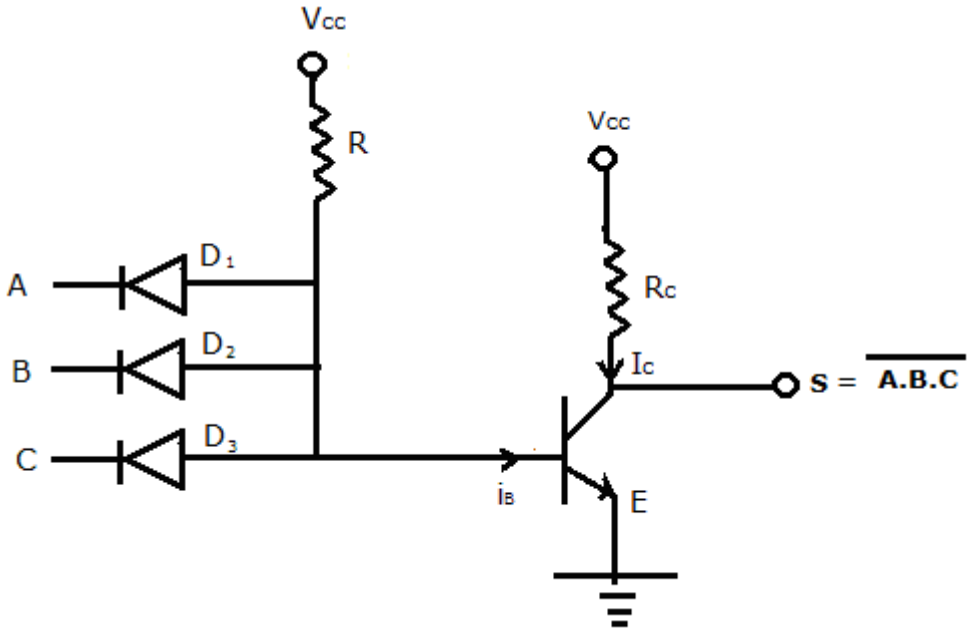
b) <u>Fonction 'NON-OU (NOR)'</u> :

Dès qu'une entrée d'une porte Non-Ou passe au niveau logique 1, le transistor devient saturé et la sortie passe au niveau logique 0. Si toutes les entrées sont au niveau logique 0, le transistor se bloque et la sortie est au niveau logique 1.



c) Fonction 'NON-ET (NAND)'

Tant que les entrées A,B,C restent en l'air autrement dit à +Vcc ou au niveau logique 1, la sortie est nulle car le transistor est saturé. Dés qu'une des entrées est mise à la masse, autrement dit au niveau logique 0, le transistor se bloque et la sortie S passe au niveau logique 1.



III.1.2/ 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).

Exemple : Le circuit 74LS08 (porte ET).

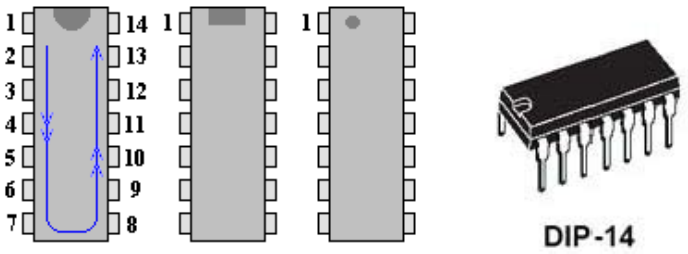


Figure 3.1: Circuit intégrés **TTL**

III.1.3/ La famille CMOS :

CMOS est l'abréviation de ''Complementary MetalOxide 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.

Exemple : le circuit 4081 (porte ET).

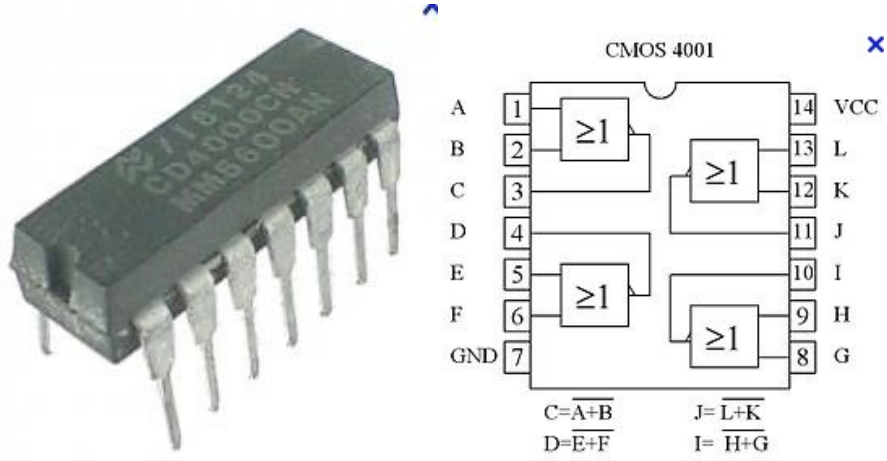


Figure 3.2 : Circuit intégré **CMOS** 4001

Chapitre IV

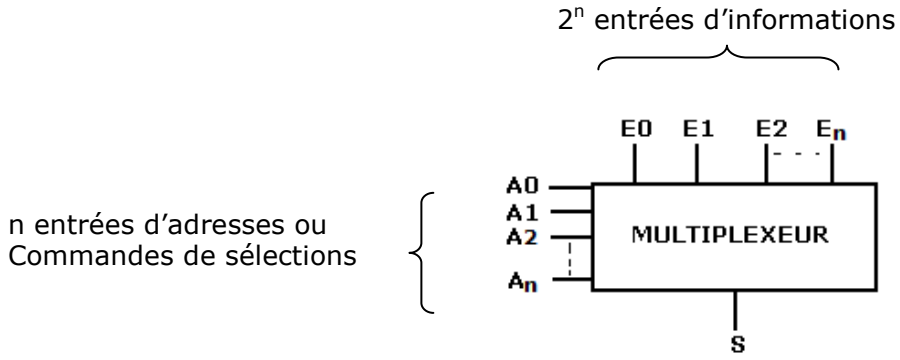
Les Circuits combinatoires

IV.1/Introduction:

On dit qu'un circuit logique est un circuit combinatoire, si l'état logique de la sortie dans un temps précis est une équation donnée en fonction des variables d'entrées et ne change d'état que si un changement apparaît sur l'état de ces variables. La fonction du circuit combinatoire est de réaliser les fonctions logiques.

IV.2/Les Multiplexeurs:

Le multiplexeur est un dipôle combinatoire constitué d'une sortie et de plusieurs entrées.



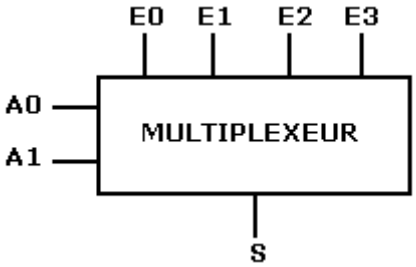
Une sortie

Si on a n entrées, alors on doit avoir P commandes de sélections telque : $n \leq 2^p$

Exemple : Réaliser un multiplexeur à 4 entrées d'informations.

On aura besoin de 2 commandes de sélections (A_0 et A_1) car : $4 \leq 2^2$

Le schéma bloc du multiplexeur :

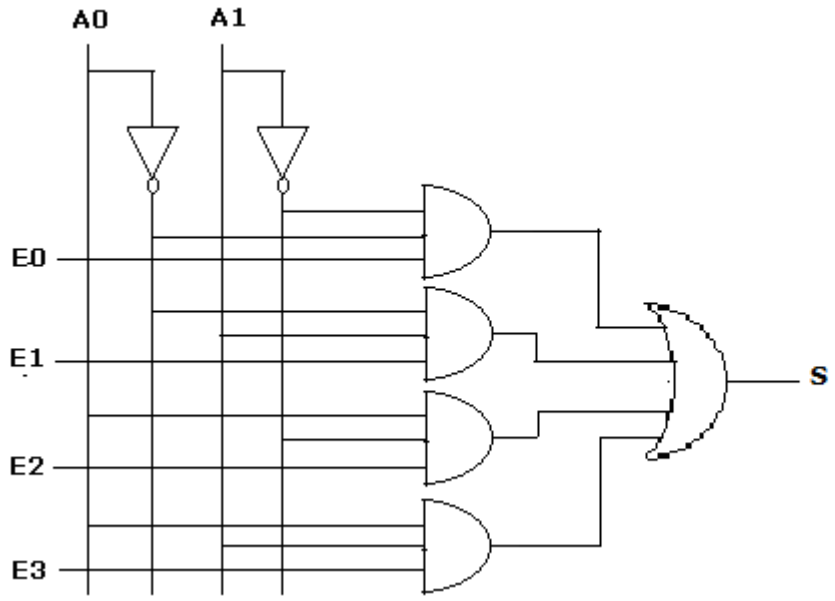


La table de vérité:

A0	A1	S
0	0	E0
0	1	E1
1	0	E2
1	1	E3

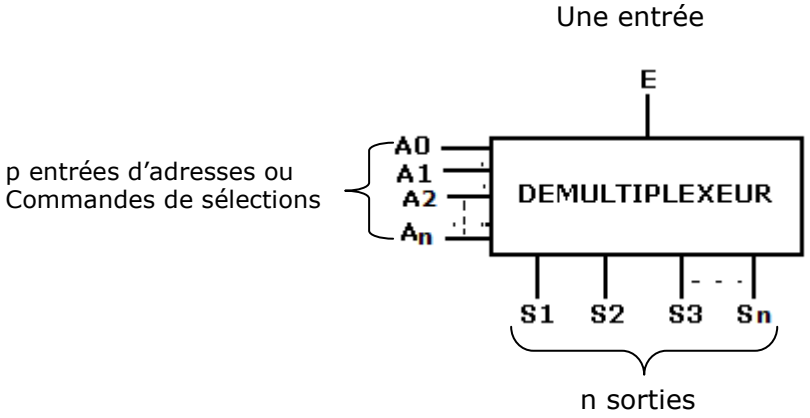
$$S = E_0.\overline{A}_0.\overline{A}_1 + E_1.\overline{A}_0.A_1 + E_2.A_0.\overline{A}_1 + E_3.A_0.A_1$$

Le circuit logique :



IV.3/ Les Démultiplexeurs:

Le démultiplexeur est multi pôle combinatoire, constitué d'une seul entrée et plusieurs sorties.

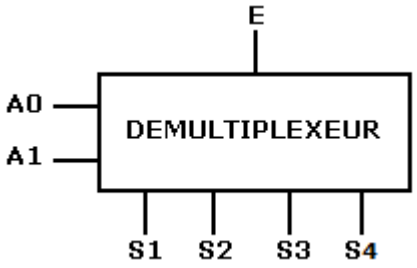


Si on a n sorties, alors on doit avoir P commandes de sélections : $2^{p-1} < n \leq 2^p$

Exemple : Réaliser un multiplexeur à 4 sorties.

On aura besoin de 2 commandes de sélections (A_0 et A_1) car : $2^{2-1} < 4 \leq 2^2$

Le schéma bloc du démultiplexeur :

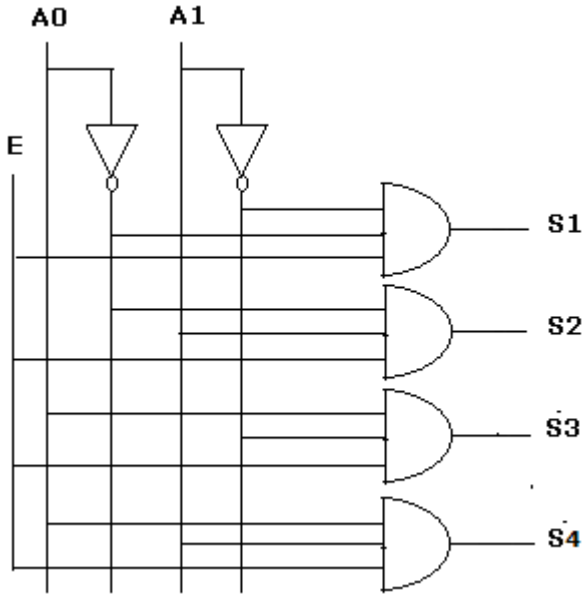


La table de vérité:

A0	A1	S1	S2	S3	S4
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

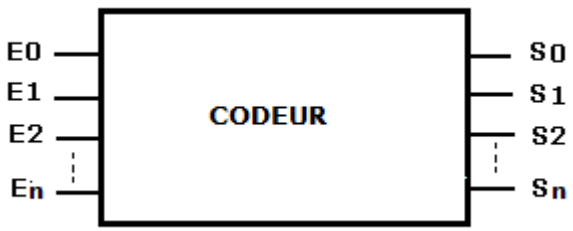
$$S_1 = \overline{A}_0.\overline{A}_1.E\,, S_2 = \overline{A}_0.A_1.E\,,$$
$$S_3 = A_0.\overline{A}_1.E\ ,\ S_4 = A_0.A_1.E$$

Le circuit logique :



IV.4/Les Codeurs:

Le Codeur est convertisseur numérique/numérique qui a 2^n entrées et n sorties.



Le Codeur est nommé aussi convertisseur au code BCD. Le Code BCD est un code qui contient les dix premières combinaisons des 16 combinaisons.

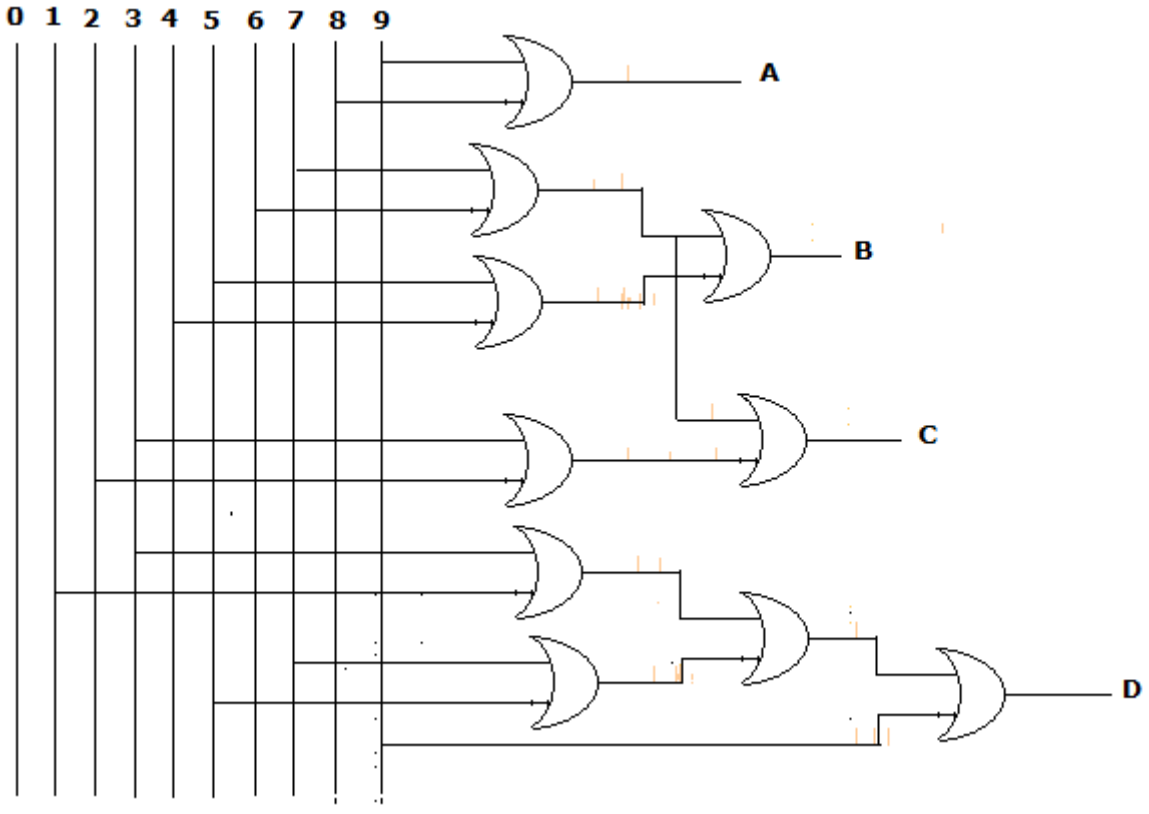
Table de Vérité:

Décimal	Code BCD
Nr	A B C D
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

On partant de la table de vérité, on obtiendra les équations de sorties suivantes.

A = 8+9
B = 4+5+6+7
C = 2+3+6+7
D = 1+3+5+7+9

Le circuit logique :



IV.5/Les Décodeurs:

Le décodeur est aussi un convertisseur numérique/numérique. Il est constitué de n entrées et 2^n sorties.

Le Décodeur est aussi un convertisseur au code BCD vers le décimal.

Table de Vérité :

Code BCD	Décimal
A B C D	Nr
0 0 0 0	**0**
0 0 0 1	**1**
0 0 1 0	**2**
0 0 1 1	**3**
0 1 0 0	**4**
0 1 0 1	**5**
0 1 1 0	**6**
0 1 1 1	**7**
1 0 0 0	**8**
1 0 0 1	**9**

On partant de la table de vérité, on obtiendra les équations de sorties à partir des tableaux de Karnaugh.

Tableaux de Karnaugh :

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

$$S_0 = \overline{A}.\overline{B}.\overline{C}.\overline{D}$$

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

$S_1 = \overline{A}.\overline{B}.\overline{C}.D$

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

$S_2 = .\overline{B}.C.\overline{D}$

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

$S_3 = .\overline{B}.C.D$

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

$S_4 = .B.\overline{C}.\overline{D}$

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

$S_5 = .B.\overline{C}.D$

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

$S_6 = .B.C.\overline{D}$

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

$S_7 = .B.C.D$

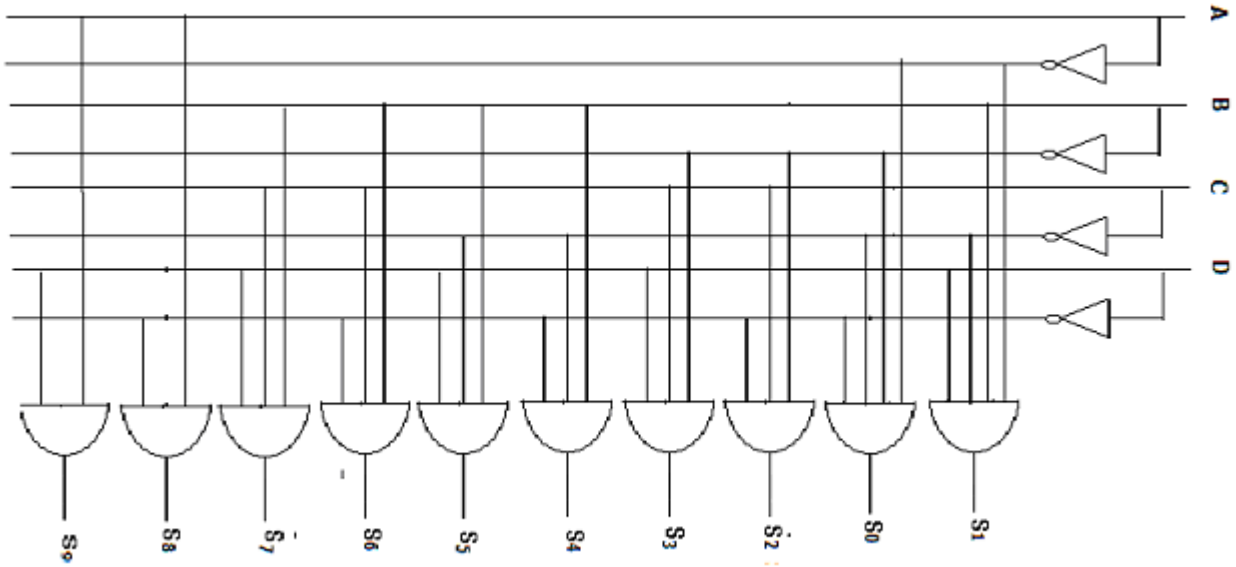
CD \ AB	00	01	11	10
00	0	0	Φ	1
01	0	0	Φ	0
11	0	0	Φ	Φ
10	0	0	Φ	Φ

$S_8 = .A.\overline{D}$

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

$S_9 = .A.D$

Le circuit logique :



IV.6/Les Transcodeurs:

Le transcodeur est convertisseur de code, qui a n entrée et n sorties.

Exemple : Réaliser un transcodeur Code Binaire Naturel ⟶ Code Binaire Réfléchi (Gray)

Table de vérité :

Code Binaire				Code Gray			
A8	A4	A2	A1	Ar	Br	Cr	Dr
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Tableaux de Karnaugh :

A2 A1 \ A8 A4 (CD)	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$A_r = A_8$

A2 A1 \ A8 A4	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$B_r = \overline{A}_8 A_4 + A_8 \overline{A}_4 = A_8 \oplus A_4$

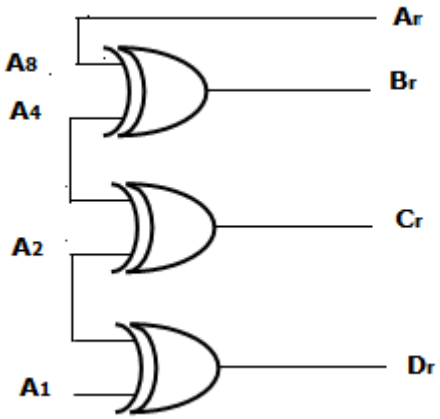
A2 A1 \ A8 A4	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

$C_r = A_4\overline{A}_2 + \overline{A}_4A_2 = A_4 \oplus A_2$

A2 A1 \ A8 A4	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$D_r = A_1\overline{A}_2 + A_2\overline{A}_1 = A_2 \oplus A_1$

Le circuit logique :



IV.6.1/Les Codes décimaux-binaires

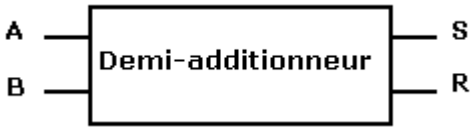
Décimal	Aiken 2 4 2 1	Code majoré de trois	Code 2 parmi 5
0	0 0 0 0	0 0 1 1	1 1 0 0 0
1	0 0 0 1	0 1 0 0	0 0 0 1 1
2	0 0 1 0	0 1 0 1	0 0 1 0 1
3	0 0 1 1	0 1 1 0	0 0 1 1 0
4	0 1 0 0	0 1 1 1	0 1 0 0 1
5	1 0 1 1	1 0 0 0	0 1 0 1 0
6	1 1 0 0	1 0 0 1	0 1 1 0 0
7	1 1 0 1	1 0 1 0	1 0 0 0 1
8	1 1 1 0	1 0 1 1	1 0 0 1 0
9	1 1 1 1	1 1 0 0	0 0 1 0 0

IV.7/Les Opérations Arithmétique D'addition-Soustraction:

IV.7.1/Le demi additionneur:

Soit la table d'addition binaire pour deux bits :

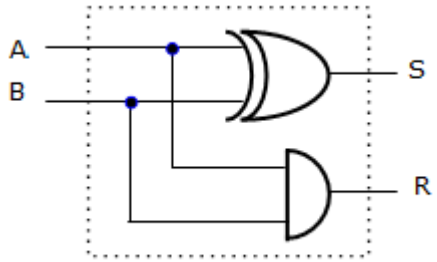
A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Les équations des fonctions de sorties sont :

$S = \overline{A}.B + A.\overline{B} = A \oplus B$; $R = A.B$

Le logigrame :



IV.7.2/L' additionneur binaire:

Ce circuit additionne les éléments binaires A_n , B_n de même rang et la retenue R_{n-1} venant de l'addition du rang précédent. Son fonctionnement obéit à la table d'implication suivante :

$R_n\ R_{n-1}$ .....R_0 0

+..... A_n.....$A_1\ A_0$

+..... B_n.....$B_1\ B_0$

=.....S_n.....$S_1\ S_0$

Table de vérité:

A_n	B_n	R_{n-1}	S_n	R_n
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

Les équations de sorties par Karnaugh :

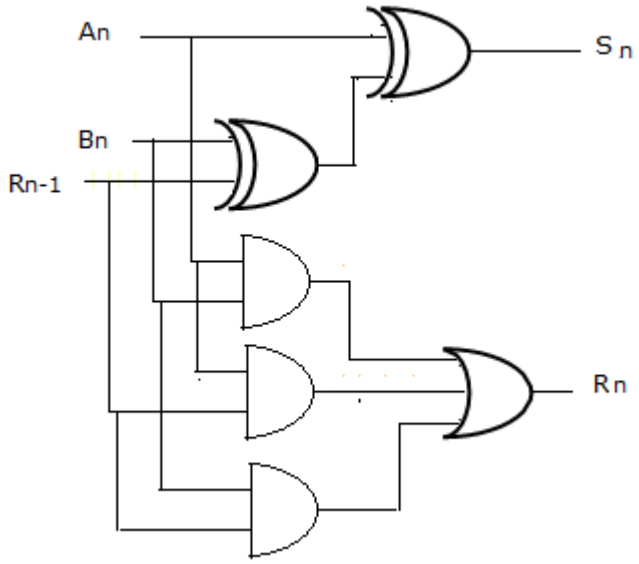
R_{n-1} \ $A_n B_n$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S_n = \overline{A}_n\overline{B}_nR_{n-1} + \overline{A}_nB_n\overline{R}_{n-1} + A_nB_nR_{n-1} + A_n\overline{B}_n\overline{R}_{n-1}$$
$$S_n = \overline{A}_n\left(\overline{B}_nR_{n-1} + B_n\overline{R}_{n-1}\right) + A_n\left(B_nR_{n-1} + \overline{B}_n\overline{R}_{n-1}\right)$$
$$S_n = \overline{A}_n\left(B_n \oplus R_{n-1}\right) + A_n\left(\overline{B_n \oplus R_{n-1}}\right) = A_n \oplus \left(B_n \oplus R_{n-1}\right)$$

R_{n-1} \ A_nB_n	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$R_n = A_n.B_n + A_n.R_{n-1} + B_n.R_{n-1}$$

Le circuit logique de l'additionneur complet :



IV.7.3/Le demi soustracteur:

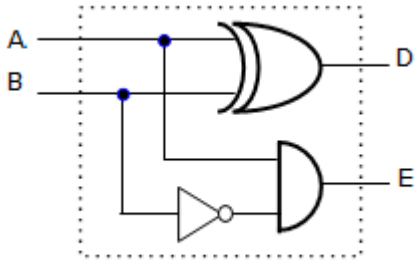
Soit la table de la soustraction binaire pour deux bits :

A	B	D	E
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Les équations des fonctions de sorties sont :

$D = \overline{A}.B + A.\overline{B} = A \oplus B$, $E = \overline{A}.B$

Le logigrame :



IV.7.4/Le soustracteur binaire:

Il reçoit les éléments binaires A_n ,B_n et l'emprunt précédente pour élaborer la différence suivante : $D_n = A_n - (B_n+E_{n-1})$.

Table de vérité.

A_n	B_n	E_{n-1}	D_n	E_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Les équations de sorties par Karnaugh :

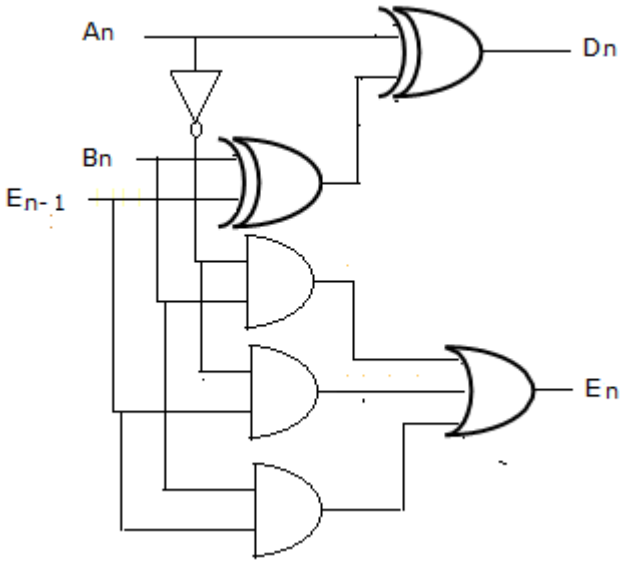
E_{n-1} \ $A_n B_n$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$D_n = A_n \oplus (B_n \oplus E_{n-1})$$

E_{n-1} \ $A_n B_n$	00	01	11	10
0	0	1	0	0
1	1	1	1	0

$$E_n = \overline{A}_n.B_n + \overline{A}_n.E_{n-1} + B_n.E_{n-1}$$

Le circuit logique du soustracteur complet :



IV.8/Les Comparateurs:

Ces circuits permettent de détecter l'égalité de deux nombres binaires A=(An…….A_k……A_0) et B=(An…….B_k…….B_0). Ces circuits permettent aussi de détecter, Si A est supérieur ou bien inferieur à B.

Table de vérité:

a b	a>b	a=b	a<b
0 0	0	1	0
0 1	0	0	1
1 0	1	0	0
1 1	0	1	0

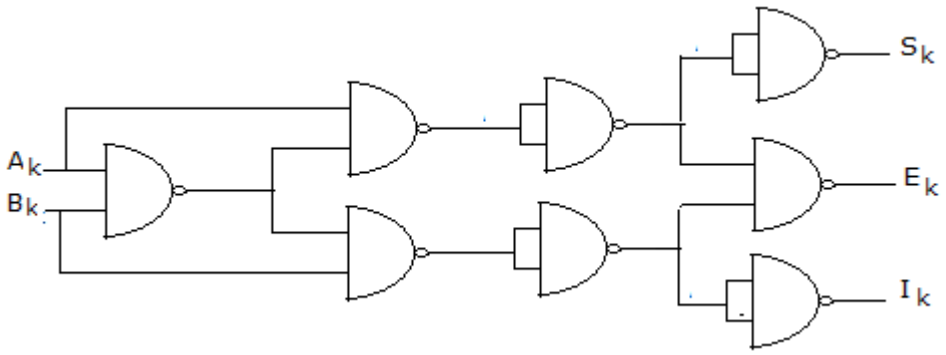
Les équations de sorties :

$(a\rangle b) = a.\overline{b} = S \quad (a = b) = \overline{a \oplus b} = E \qquad (a\langle b) = \overline{a}.b = I$

Si on pose :

$a = A_k$, $b = B_k$, $S = S_k$, $E = E_k$, $I = I_k$

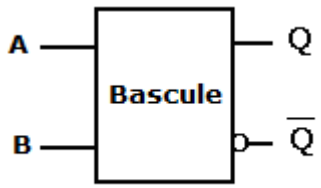
Le circuit logique est le suivant :


A_k
B_k
S_k
E_k
I_k

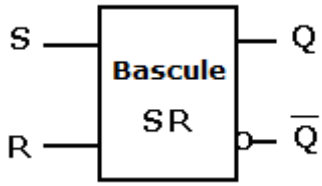
Chapitre V
Les Bascules

V.1/ Définition :

Tous les dispositifs qui peuvent commuter entre deux états, appartiennent à la famille des bascules. La transition d'un état à un autre survient brusquement et dénommée processus de commutation. En règle générale, les bascules sont dotées de deux sorties complémentaires $(Q \text{ et } \overline{Q})$et prennent deux états stables $(Q = 0 \text{ et } Q = 1)$ d'où l'appellation de bascule bistable (Q=0, état de repos),(Q=1, état opérationnel).



V.2/ Bascule S.R :



La bascule SR possède deux entrées S et R
S : Set ou mise à 1
R : Reset ou remise à 0
Et deux sorties complémentaires $(Q \text{ et } \overline{Q})$.

V.2.1/ Comportement de commutation des bascules SR :

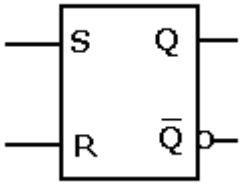
Tout comme les fonctions de commutation des éléments logiques sont représentées dans des tables de vérité, les caractéristiques de commutation des bascules peuvent elles aussi, être illustrées par des tables.

- Au moyen d'une table de vérité
- Au moyen d'une table d'états.

Contrairement aux tables de vérité relatives aux éléments logiques, la table de vérité d'une bascule doit prendre en considération le décalage existant entre l'excitation et l'apparition de l'état de l'état de sortie. Les raisons sont dans les caractéristiques de mémoire de bascules, l'état produit par la commutation est conservé après la fin de celle-ci.

V.2.2/ Table de vérité d'une bascule SR:

t		t+1	
R	S	Q	$\overline{Q}$
0	0	Q_t	$\overline{Q}_t$
0	1	1	0
1	0	0	1
1	1	Indéfini	



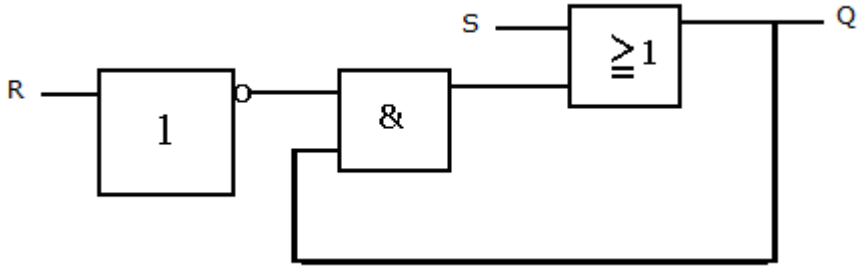
V.2.3/ Table d'états(Karnaugh):

Q_t \ SR	00	01	11	10
0	0	0	—	1
1	1	0	—	1

$$Q_{t+1} = S + \overline{R}Q_t$$

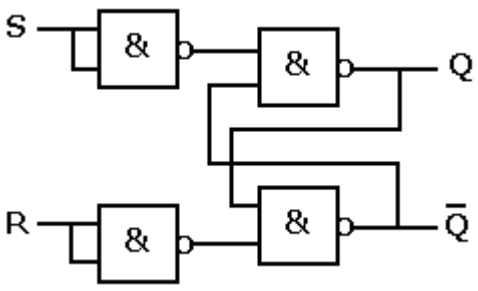
V.2.4/ Réalisation de la bascule SR avec les portes NAND et NOR:

$Q_{t+1} = S + \overline{R}Q_t$



Avec les portes NAND seulement :

$Q_t = \overline{\overline{Q}}_{t+1} = \overline{\overline{S + \overline{R}Q_t}} \Rightarrow \overline{\overline{Q}}_{t+1} = \overline{\overline{S}.\overline{\overline{R}.Q_t}}$



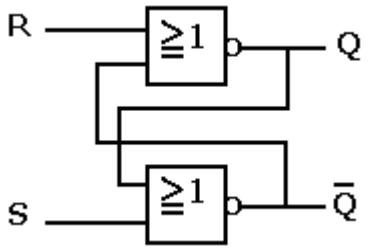
On va monter que : $\overline{Q} = \overline{\overline{R}.Q} = R + \overline{Q}$

Si R=1 et Q=0 $\Rightarrow \overline{Q} = 1 + \overline{Q} = 1$

Si R=0 $\Rightarrow$ $\overline{Q} = \overline{Q}$

Avec les portes NOR seulement :

$$\overline{Q}_{t+1} = \overline{S + \overline{R}Q_t} = \overline{S + \overline{\overline{R}Q_t}} = \overline{S + \left(\overline{R + \overline{Q}_t}\right)}$$



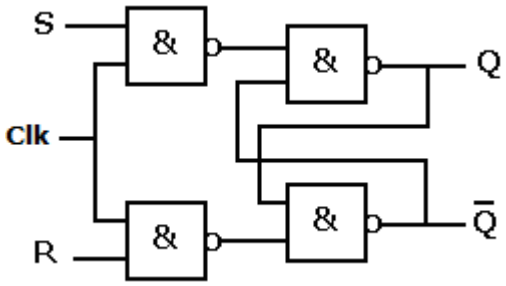
On va monter que : $Q = \overline{R + \overline{Q}}$

Si R=0, Q=Q

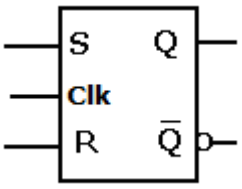
Si R=1 et Q=0, Q=Q

V.2.5/ Bascule SR Synchrone Temporisée (R.S.T):

La bascule R.S.T est une bascule pour laquelle les entrées S et R ne sont prises en compte qu'en coïncidence avec un signal de commande. Ce signal peut être fourni par une horloge, nous avons alors une bascule synchrone. Ce circuit peut être réalisé par le schéma logique suivant :



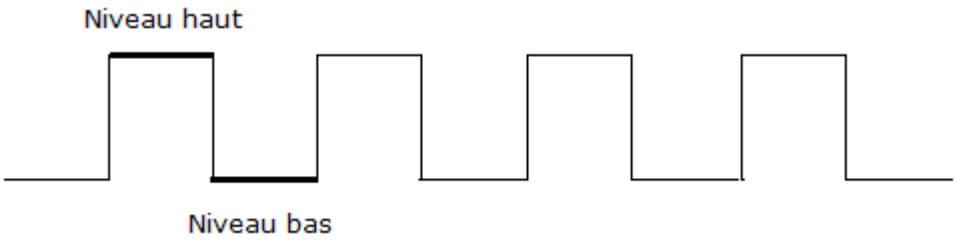
Lorsque le signal de commande, noté Clk (Clock), est à 1 la bascule fonctionne comme indiqué précédemment et les sorties suivent les variations des entrées S et R. Par contre, lorsque le signal de commande est à 0, la bascule est bloquée : Q est indépendant des éventuels changements de S et R. Le schéma bloc de la bascule RST est le suivant :



Remarque :

- L'état de la bascule ne peut changer que lorsque l'horloge est à 1, dans ce cas, on dira que la bascule est sensible au niveau haut.
- Il existe aussi des bascules sensibles au niveau bas, son état va changer lorsque la bascule est à 0.

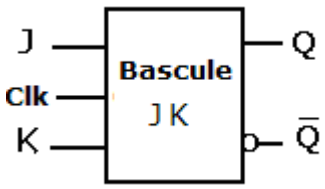
Chronogramme :



- Pour obtenir une bascule sensible au niveau bas, il suffit d'inverser l'horloge.

V.3/ Bascule J.K :

L'inconvénient des bascules RS consiste en l'état pseudo-stable. Cet inconvénient peut être éliminé au moyen de circuits spéciaux. On obtient alors une bascule JK.



La bascule JK possède deux entrées J et K.
J : Set ou mise à 1
K : Reset ou remise à 0
Une entrée d'horloge et deux sorties complémentaires $(Q \text{ et } \overline{Q})$.

V.3.1/ Table de vérité d'une bascule JK:

t_n		t_{n+1}	
J	K	Q	$\overline{Q}$
0	0	Q_n	$\overline{Q}_n$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}_n$	Q_n

Cela permet donc de déduire que la bascule JK se comporte comme une bascule RS. Sauf pour J=1 et K=1 dans cette configuration, la bascule JK passe d'un état à un autre à chaque impulsion d'horloge (basculement).

V.3.2/ Table d'états(Karnaugh):

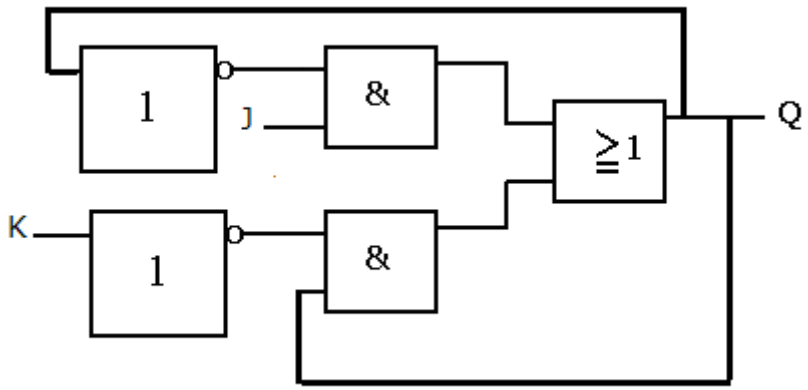
On utilise parfois l'expression logique Q_{n+1} en fonction de J_n , K_n et Q_n .Pour cela nous pouvons par exemple construire le tableau de karnaugh à partir de la table de vérité de la bascule JK :

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

D'où nous tirons l'équation caractéristique qui exprime l'état futur en fonction de l'état présent et des entrées : $Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$

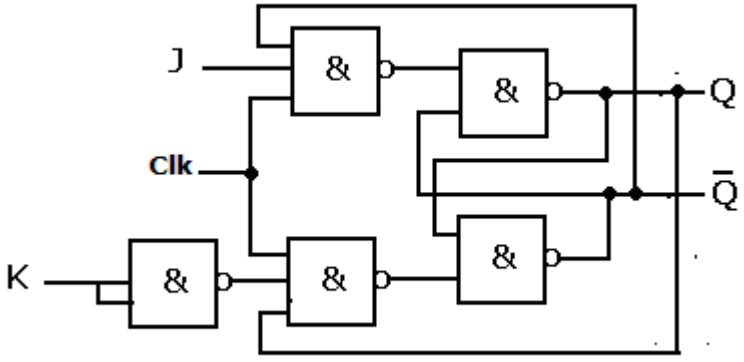
V.3.3/ Réalisation de la bascule JK avec les portes NAND:

$Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$



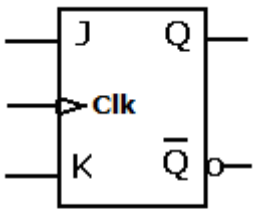
Avec les portes NAND seulement :

$$Q_{n+1} = \overline{\overline{Q}}_{n+1} = \overline{\overline{J\overline{Q}_n + \overline{K}Q_n}} = \overline{\left(\overline{J\overline{Q}_n}\right)\left(\overline{\overline{K}Q_n}\right)}$$

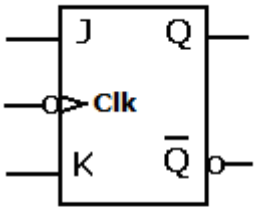


V.3.4 / Déclenchement sur front montant ou descendant du signal d'horloge:

- Lorsque la bascule change d'état pendant la transition 0 ⟶ 1, dans ce cas, on dira que la bascule est sensible au front montant.
- Lorsque la bascule change d'état pendant la transition 1 ⟶ 0, dans ce cas, on dira que la bascule est sensible au front descendant.

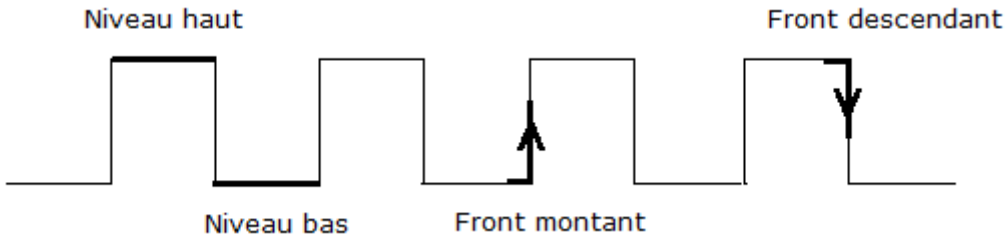


Bascule déclenchée par front montant



Bascule déclenchée par front descendant

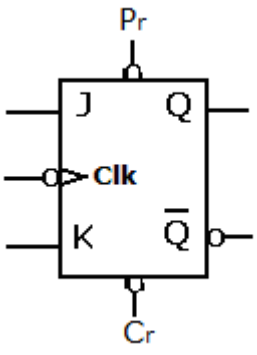
Chronogramme :

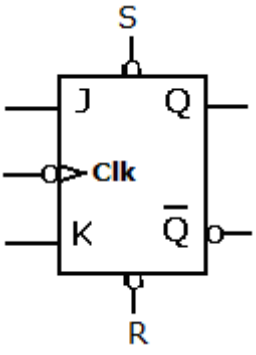


V.3.5/Preset et Clear:

Les entrées **asynchrones** (car à utiliser en absence de signal d'horloge, lorsque Clk=0) **Pr** (Preset) ou **S** (Set) et **Cr** (Clear) ou aussi **R** (Reset), permettent d'assigner l'état initial de la bascule, par exemple juste après la mise sous tension pour éviter tout aléa. En fonctionnement normal ces deux entrées doivent être maintenues à **1**. Lorsque le signal d'horloge est à **0** nous avons la table de vérité suivante :

P_r	Cr	Q
1	1	Q
0	1	1
1	0	0





V.3.6/ Réalisation de la bascule JK à partir de SR:

On peut construire cette bascule soit à partir de SR

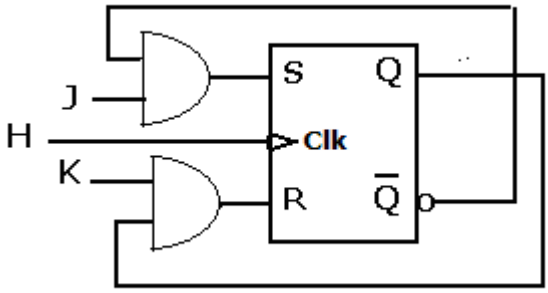
Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Q_n \ SR	00	01	11	10
0	0	0	—	1
1	1	0	—	1

La mise à 1 correspond à : $S = J.\overline{Q}$

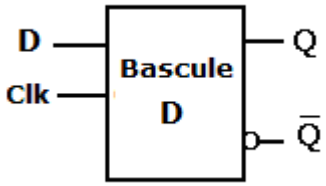
La remise à 0 correspond à : $R = K.Q$

Le schéma bloc :



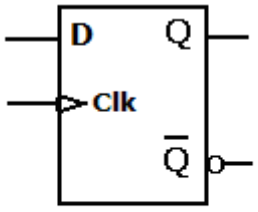
V.4/ Bascule D :

Une bascule D (Delay : retard) est une bascule dont la sortie va reproduire l'entée D avec un certain retard, c'est-à-dire : $Q_{t+\tau} = D_t$. La bascule D ne possédant qu'une entrée pour l'information. Cette dernière ne peut jamais intervenir d'une manière indépendante étant donné quelle est constamment subordonnée a une entrée d'horloge Clk. Les bascules D se présentent donc toujours sous forme de bascule temporisée.



V.4.1/ Table de vérité d'une bascule D:

t_n	t_{n+1}	
D	Q	$\overline{Q}$
0	0	1
1	1	0



Bascule déclenchée par front montant

Quand l'entrée change, la sortie change c'est-à-dire il ya un effet de **mémorisation.**

V.4.2/ Table de Karnaugh:

Q_n \ D	0	1
0	0	1
1	0	1

$Q_{n+1} = D$

V.4.3/ Construction de la bascule D à partir de JK et SR:

On peut construire cette bascule soit à partir d'une bascule JK soit à partir de SR.

-Construction de la bascule D à partir de JK

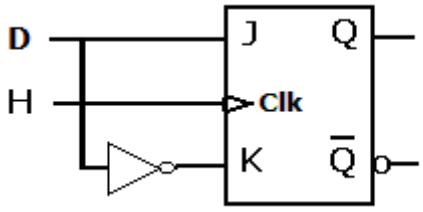
Q_n \ D	0	1
0	0	1
1	0	1

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Par analogie, d'après les deux tableaux, on peut déduire que :

$J = D$ et $K = \overline{D}$

Le schéma bloc :



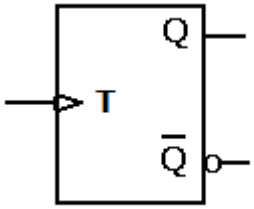
Remarque :

- La bascule D est sensible au front montant.

V.5/ Bascule T :

Les bascules T différent des autres types des bascules déjà décrites, étant donné qu'elles n'ont qu'une entrée d'horloge et ne possèdent aucune entrée pour l'information. Toute excitation survenant sur l'entrée d'horloge produit une commutation de l'état de mise 1(set) à l'état de remise à 0 (reset) et vice versa.

Les entrées qui mènent à un changement d'état à chaque apparition d'une excitation sont désignées par la lettre T.

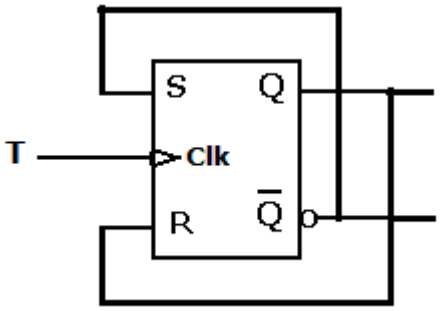


Les bascules T sont aussi dénommées diviseurs binaires. En effet, une impulsion d'excitation rectangulaire étant appliquée a l'entrée T, un changement d'état au niveau de la sortie se produira pour chaque période de la fonction d'entrée. Cela veut dire qu'il ya division de la fréquence.
Une bascule T ne possède ni table de vérité ni table d'états, étant donné que le changement d'état survient à chaque impulsion.

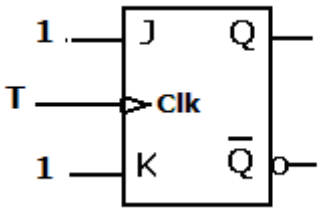
V.5.1/ Construction de la bascule T à partir de JK et SR:

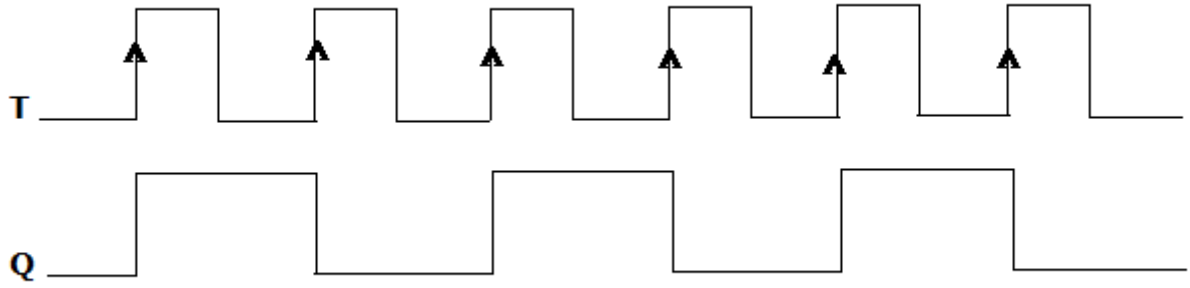
Une bascule T peut aussi être constituée à partir d'une bascule RS ou JK, moyennant le renvoi de variables de sortie aux entrées des informations.

A partir de bascules SR :



A partir de bascules JK :





Chapitre VI

Les Compteurs

VI.1/ Introduction :

Ce sont des systèmes logiques séquentiels composés de plusieurs bascules bistables en cascade dont la fonction consiste à diviser les fréquences et compter les impulsions. Le rapport de division de chaque bascule est de 1/2. Par exemple, on obtient un rapport de 1/8=1/2.1/2.1/2 en utilisant trois bascules.

Pour compter les impulsions (compteur d'impulsion), on affecte à chaque bascule un poids spécifique. On peut par exemple, leur affecter la valeur des positions binaires, soit $1=2^0$,$2=2^1$,$4=2^2$,$8=2^4$ etc. Les résultats du compteur sont alors affichés directement en code binaire.

Les compteurs binaires peuvent être classés:

a) En fonction de leur mode de fonctionnement en
- compteurs asynchrones,
- compteurs synchrones.

b) En fonction de leur sens de comptage en
- compteurs progressifs (compteurs),
- compteurs régressifs (décompteurs).

c) En fonction du codage de leur résultat en
- compteurs binaires (poids : 1, 2, 4,8, 16…),
- compteurs DCB pour le code 8-4-2-1, le code 2-4-2-1, etc.

VI.2/ Compteurs Synchrones :

Dans un compteur synchrone toutes les bascules reçoivent en parallèle le même signal d'horloge.

VI.2.1/ Conception et Réalisation d'un Compteur Synchrone :

Soit un compteur synchrone modulo M, comment le réaliser à l'aide des bascules JK ?

- On définie d'abord le nombre de bascules nécessaires pour réaliser le compteur suivant la formule $\mathbf{M} \leq 2^{\mathbf{n}}$ avec **n** le nombre de bascule pour réaliser un compteur Modulo **M**.
- Ecrire le Tableau des séquences successives (Table de transition) des bascules A,B,C,D ou 1,2,3,4 …….. etc.
- Etablir à partir du tableau de karnaugh les entrées de J, K ou T de chaque bascule en fonction des sorties Q_A, Q_B, Q_C, Q_D.

VI.2.2/ Les Compteurs Synchrones Binaires :

VI.2.2.1/ Compteur Modulo 4 :

M=4 ⟹ n=2 (on a besoin de deux bascules J.K ou T)

Donc on a deux entrées : $J_{A,\,KA}$; $J_{B,\,KB}$ et deux sorties : Q_A ; Q_B

a) Tableau des séquences successives :

M	Q_B	Q_A	$J_B\,K_B$	$J_A\,K_A$
0	0	0	0 Φ	1 Φ
1	0	1	1 Φ	Φ 1
2	1	0	Φ 0	1 Φ
3	1	1	Φ 1	Φ 1

Explication :

L'exemple de la sortie Q_B :

Première ligne : elle a passé à zéro, deux combinaisons peuvent se présenter :

1) Q_B=0 ⟹ J_B=0, K_B=0 (mémorisation)

2) Q_B=0 ⟹ J_B=0, K_B=1 (remise à 0)

On écrit alors : J_B=0, K_B= Φ

Deuxième ligne : Q_B a passé de zéro à 1

1) Q_B=1 ⟹ J_B=1, K_B=1 (basculement)

2) Q_B=1 ⟹ J_B=1, K_B=0 (mise à 1)

On écrit alors : J_B=1, K_B= Φ

Troisième ligne : Q_B garde la même valeur 1

1) Q_B=1 ⟹ J_B=0, K_B=0 (mémorisation)

2) Q_B=1 ⟹ J_B=1, K_B=0 (mise à 1)

On écrit alors : J_B= Φ, K_B= 0

Quatrième ligne : Q_B a passé de 1 à zéro

1) Q_B=0 ⟹ J_B=1, K_B=1 (basculement)

2) Q_B=1 ⟹ J_B=0, K_B=1 (remise à 0)

On écrit alors : J_B= Φ, K_B= 1

b) Tableaux de karnaugh :

Q_A \ Q_B	0	1
0	0	Φ
1	1	Φ

J_B= Q_A

Q_A \ Q_B	0	1
0	Φ	0
1	Φ	1

K_B= Q_A

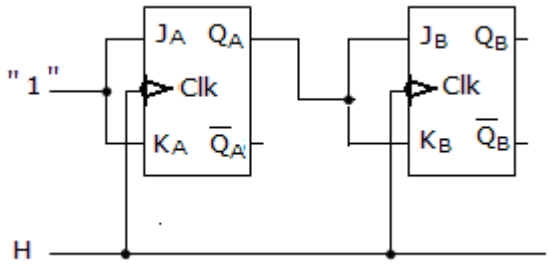
Q_A \ Q_B	0	1
0	1	1
1	Φ	Φ

J_A=1

Q_A \ Q_B	0	1
0	Φ	Φ
1	1	1

K_A= 1

c) Le Logigramme :



VI.2.2.2/ Compteur Modulo 8 :

M=8 ⟹ n= 3 on a besoin de trois bascules J.K ou T

Donc on a trois entrées : $J_{A,\,KA}$; $J_{B,\,KB}$; $J_{C,\,KC}$ et trois sorties : Q_A ; Q_B Q_C

a) Tableau des séquences successives :

M	Q_C	Q_B	Q_A	J_C K_C	J_B K_B	J_A K_A
0	0	0	0	0 Φ	0 Φ	1 Φ
1	0	0	1	0 Φ	1 Φ	Φ 1
2	0	1	0	0 Φ	Φ 0	1 Φ
3	0	1	1	1 Φ	Φ 1	Φ 1
4	1	0	0	Φ 0	0 Φ	1 Φ
5	1	0	1	Φ 0	1 Φ	Φ 1
6	1	1	0	Φ 0	Φ 0	1 Φ
7	1	1	1	Φ 1	0 Φ	Φ 1

b) Tableau de karnaugh :

Q_A \ $Q_C Q_B$	00	01	11	10
0	0	0	Φ	Φ
1	0	1	Φ	Φ

$J_C = Q_B\,Q_A$

Q_A \ $Q_C Q_B$	00	01	11	10
0	Φ	Φ	0	0
1	Φ	Φ	1	0

$K_C = Q_B\,Q_A$

Q_A \ $Q_C Q_B$	00	01	11	10
0	0	Φ	Φ	0
1	1	Φ	Φ	1

$J_B = Q_A$

Q_A \ $Q_C Q_B$	00	01	11	10
0	Φ	0	0	Φ
1	Φ	1	Φ	Φ

$K_B = Q_A$

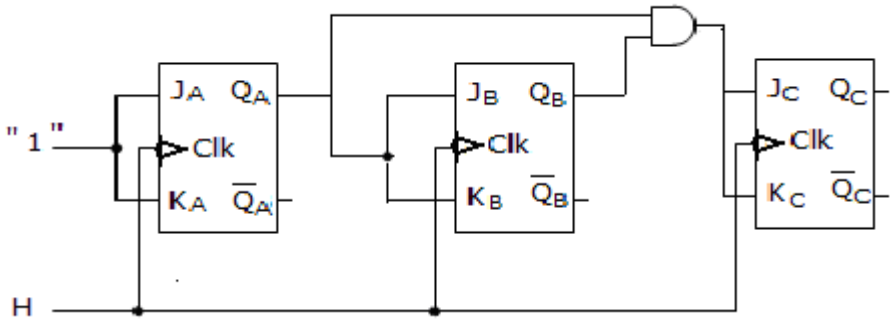
Q_A \ $Q_C Q_B$	00	01	11	10
0	1	1	1	1
1	Φ	Φ	Φ	Φ

$J_A = 1$

Q_A \ $Q_C Q_B$	00	01	11	10
0	Φ	Φ	Φ	Φ
1	1	1	1	1

$K_A = 1$

c) Le Logigramme:



De manière générale nous pouvons donner les équations de commutation d'un compteur asynchrone modulo 2^n avec n bascules JK de la manière suivante :

$J_A = K_A = 1$
$J_B = K_B = Q_A$
$J_C = K_C = Q_B . Q_A$
$J_D = K_D = Q_C . Q_B . Q_A$
……
……
……
……
$J_Z = K_Z = Q_Y . Q_X$…… $. Q_C . Q_B . Q_A$

VI.2.3/ Les Décompteurs Synchrones Binaires :

VI.2.3.1/ Décompteur Modulo 4 :

a) Tableau des séquences successives :

M	Q_B	Q_A	$J_B K_B$	$J_A K_A$
3	1	1	Φ 0	Φ 1
2	1	0	Φ 1	1 Φ
1	0	1	0 Φ	1 Φ
0	0	0	1 Φ	Φ 1

b) Tableaux de karnaugh :

Q_A \ Q_B	0	1
0	1	Φ
1	0	Φ

$J_B = \overline{Q}_A$

Q_A \ Q_B	0	1
0	Φ	1
1	Φ	0

$K_B = \overline{Q}_A$

Q_A \ Q_B	0	1
0	1	1
1	Φ	Φ

$J_A = 1$

Q_A \ Q_B	0	1
0	Φ	Φ
1	1	1

$J_B = 1$

Les équations de commutations d'un décompteur binaire synchrone modulo 2^n avec n bascules JK sont :

$J_A = K_A = 1$

$J_B = K_B = \overline{Q}_A$

$J_C = K_C = \overline{Q}_B.\overline{Q}_A$

……

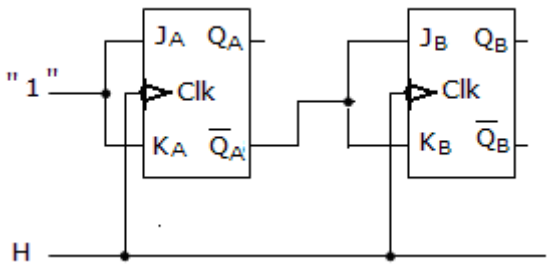
……

……

……

$J_Z = K_Z = \overline{Q}_Y.\overline{Q}_X$…….$\overline{Q}_B.\overline{Q}_A$

c) Le Logigramme:



VI.2.4/ Compteur Synchrone Décimal (DCB):

Les ordinateurs travaillent souvent en code DCB (**d**écimal **c**odé **b**inaire), dans lequel chaque position décimale (valeurs de 0 à 9) est codée en binaire. Les compteurs conçus pour des codes DCB avec poids de 8, 4,2 et1 comptent de 0 à 9 comme les compteurs

binaires alors que la 10éme impulsion de commutation doit commander une mise à 0. De tels compteurs DCB ou décimaux se composent de quatre bascule par position décimale (décade), ce qui correspond à une capacité de 0 à 15. Les valeurs 0 à 9 sont utilisées ici les valeurs restantes de 10 à 15 sont supprimées soit par arrêt du compteur par les entrées **C**lea**r (R**eset**)** et **Pr**eset **(S**et**)** dont le rôle et de mettre la bascule soit à **0** soit à **1** indépendamment de l'horloge, le forçage se fait manuellement. Soit par des circuits appropriés, les modifications sont réalisées par la méthode décrit dans le paragraphe (VI.2.1).

Compteur progressif synchrone modulo 16 ⟹ on a besoin de 4 bascules JK :

$J_A = K_A = 1$

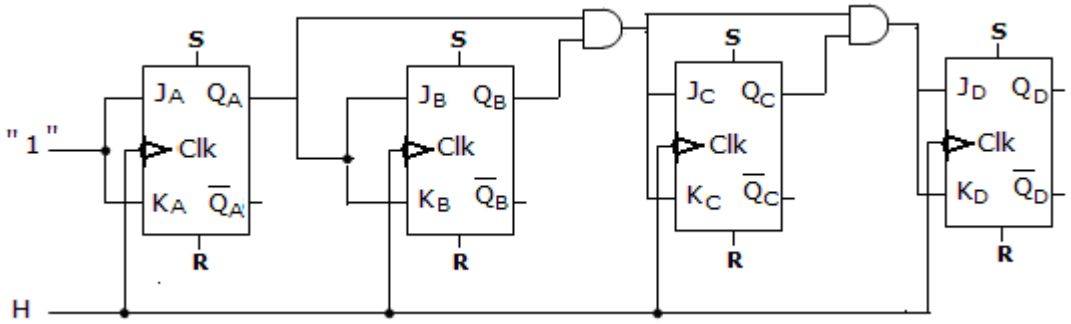
$J_B = K_B = Q_A$

$J_C = K_C = Q_B . Q_A$

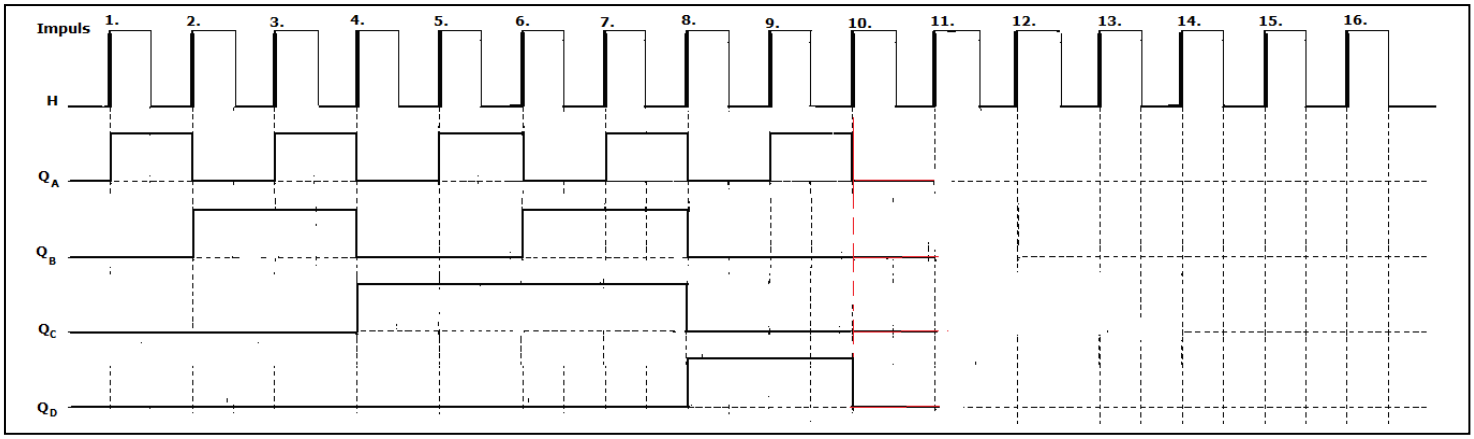
$J_D = K_D = Q_C . Q_B . Q_A$

Pour un compteur synchrone décimal, il va compter de 0 à 9, il faut arrêter le compteur modulo 16 lorsqu'il arrive à 9.

a) Le Logigramme:



b) Chronogramme :

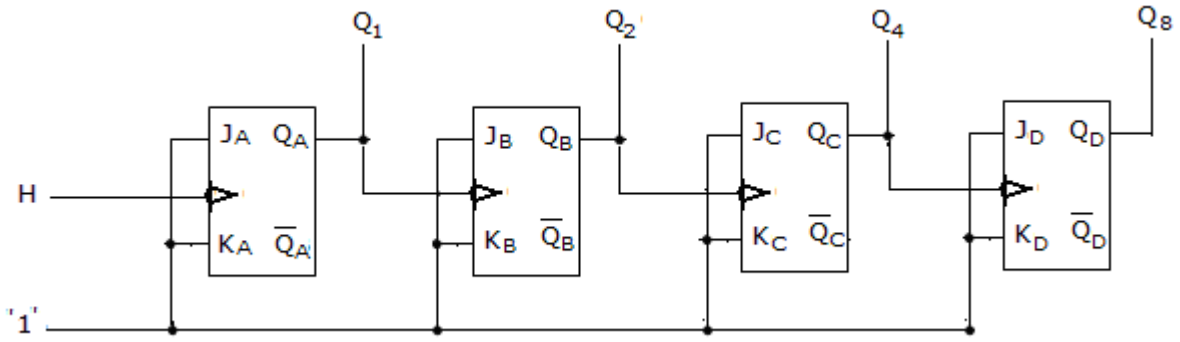


VI.3./ Compteurs Asynchrones :

Un compteur asynchrone est constitué de n bascules J-K fonctionnant en mode T. Le signal d'horloge n'est reçu que par le premier étage (bascule LSB : Least Significant Bit). Pour chacune des autres bascules le signal d'horloge est fourni par une sortie de la bascule de rang immédiatement inférieur.

VI.3.1/ Compteur Binaire à comptage Progressif :

Chaque bascule d'un compteur asynchrone est dans le cas le plus simple, un diviseur binaire (bascule T bistable). Ces circuits sont disposés en cascade de sorte que la sortie d'une bascule soit systématiquement reliée à l'entrée dynamique de la bascule suivante lequel, lorsqu'il bascule en position zéro fournit un signal de déclenchement à l'entrée dynamique. Dans le compteur ci-dessus, toutes les bascules T bistables sont excitées par un front montant (0⟶1).



VI.3.1.1/ Mode de fonctionnement :

On suppose que tous les bascules sont en position zéro. Seuls les fronts d'impulsion de 0 à 1 peuvent engendrer un enclenchement (impulsion de commutation).

1ére impulsion de commutation :

-bascule A : passage de l'état zéro à l'état un.

2éme impulsion de commutation :

-bascule A : **passage de l'état un à l'état zéro**.

-bascule B : passage de l'état zéro à l'état un.

3éme impulsion de commutation :

-bascule A : passage de l'état zéro à l'état un.

4éme impulsion de commutation :

-bascule A : **passage de l'état un à l'état zéro**.

-bascule B : **passage de l'état un à l'état zéro**.

-bascule C : passage de l'état zéro à l'état un.

5éme impulsion de commutation :

-bascule A : passage de l'état zéro à l'état un.

6éme impulsion de commutation :

-bascule A : **passage de l'état un à l'état zéro**.

-bascule B : passage de l'état zéro à l'état un.

7éme impulsion de commutation :

-bascule A : passage de l'état zéro à l'état un.

8éme impulsion de commutation :

-bascule A : **passage de l'état un à l'état zéro**.

-bascule B : **passage de l'état un à l'état zéro**.

-bascule C : **passage de l'état un à l'état zéro**.

-bascule D : passage de l'état zéro à l'état un.

9éme impulsion de commutation :

-bascule A : passage de l'état zéro à l'état un.

Etc.

Remarque : Cette liste montre clairement que, dans le cas d'un compteur asynchrone binaire, une bascule passant à l'état zéro active le circuit suivant.

Bibliographie

[1] J.Letocha, « Introduction aux circuits logiques », Edition Mc-Graw Hill,1982.

[2] G. Almouzni, « Electronique Numérique »; EISTI , 2008.

[3] C. Alxandre. « Circuits Numériques », Polycopie de cours, Janvier 2004.

[4] «Electronique Numérique Théorie », Mentor Sciences, Avril 90, Copyright PTT, Suisses.

[5] C. Brie, « Logique Combinatoire et Séquentielle: Méthodes, outils et réalisations », Edition Ellipses, Collection Technosup, Decembre 2002.

[6] C. Brie, « Logique Combinatoire et Séquentielle: Méthodes, outils et réalisations », Edition Ellipses, Collection Technosup, Decembre 2002.

[7] Y. Darbellay. «Cours Electronique Numérique» Electronique Numérique», Etml , juillet 2006.

[8] T. Floyd, « Systèmes numériques », Edition Reynald Goulet Inc ,2006.

[9] J.P. Ginisti, « La Logique Combinatoire » , Edition Puf, Collection Que sais-je ?, Février 1997.

[10] J. Lagasse , «Logique Combinatoire et séquentielle » Edition Dunod, Paris 1969.

[11] J. Lagasse, J. Erceau , « Logique combinatoire et séquentielle », Edition Dunod, 1971.

[12] L. Museur,. « Electronique Numérique Logique Combinatoire et Séquentielle » Polycopie de Cours, Université Paris13, Institut Gallilé, 2007

[13] R. Strandh, I. Durand, « Architecture de l'ordinateur - Portes logiques, circuits combinatoires, arithmétique binaire, circuits séquentiels et mémoires. Exe » Edition Dunod, 2005.

[14] J. Villemejane , « Du codage des nombres à la logique séquentielle» Cours, Département de Génie Électrique et Informatique Industrielle, Institut Universitaire de Technologie de Creteil-Vitry,2013.□ .

[15] R.Katz, « Contemporary Logic Design» 2nd.Edition Prentice Hall,2005.□ .