Office de la Formation Professionnelle
et de la Promotion du Travail

Technicien Spécialisé

Génie Electrique

Tronc commun

Manuel de cours

Module 11

Logique combinatoire et séquentielle

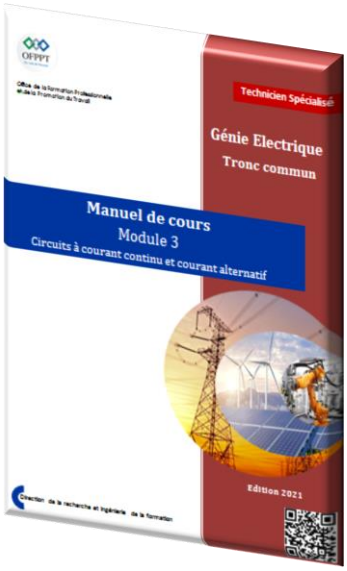
Edition 2021

Direction de la Recherche et Ingénierie de la Formation

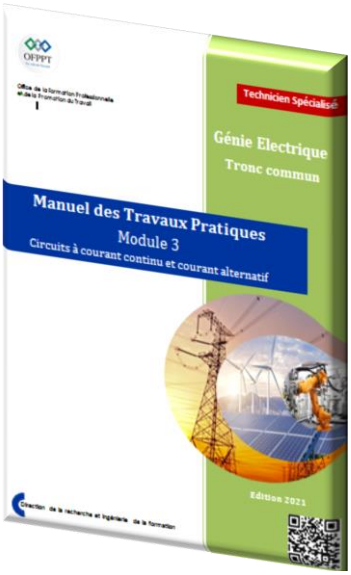
Avant-propos

Les manuels de cours, de travaux pratiques et le guide e-learning sont téléchargeables à partir de la plateforme e-learning moyennant les codes QR suivants :

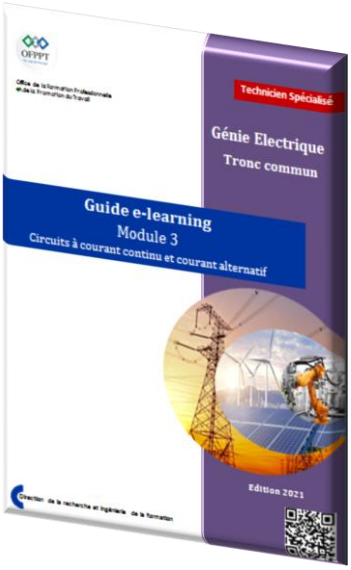
Manuel de cours



Manuel des travaux pratiques



Guide e-learning



SOMMAIRE

COMPETENCES-CIBLES ET OBJECTIFS OPERATIONNELS

Module 11 : Logique combinatoire et séquentielle

Code : GETC – 11 **Durée : 45 heures**

ENONCE DE LA COMPETENCE

Pour démontrer sa compétence, le stagiaire doit
Appliquer les notions de la logique combinatoire et séquentielle
selon les conditions, les critères et les précisions qui suivent.

CONTEXTE DE REALISATION

- Individuellement
- À partir :
  - De directives ;
  - Manuels et Fiches techniques ;
  - De schémas ;
  - D'une équation non simplifiée.
  - D'une représentation graphique d'une séquence ;
- À l'aide :
  - Composants logiques ;
  - Matériaux d'assemblage ;
  - Outils et d'instruments de mesure ;
  - Equipement de protection individuelle.
  - Documents audiovisuels

CRITÈRES GÉNÉRAUX DE PERFORMANCE

- Pertinence de la terminologie utilisée.
- Pertinence de l'utilisation des outils et des instruments.
- Qualité des travaux.
- Respect des règles de santé et de sécurité au travail.

ÉLEMENTS DE LA COMPETENCE	CRITÈRES PARTICULIERS DE PERFORMANCE
A. Réaliser des circuits à logique combinatoire.	• Utilisation pertinente des règles et des méthodes de transformation des propositions logiques. • Utilisation judicieuse des lois de l'algèbre de Boole et des méthodes de simplification afin d'utiliser un minimum de circuits intégrés dans la matérialisation d'une expression logique. • Choix pertinent des technologies pour la réalisation des opérateurs logiques. • Interprétation correcte des fiches techniques des circuits intégrés logiques. • Production de systèmes fonctionnels. • Utilisation correcte d'encodeurs et de décodeurs ainsi que de multiplexeurs et de démultiplexeurs
B. Réaliser des circuits de logique combinatoire et séquentielle utilisés couramment dans des systèmes électroniques.	• Analyse mathématique et modélisation des systèmes numériques. • Branchement fonctionnel des bascules, des registres à décalage et des compteurs. • Réalisation conforme à un cahier de charge de systèmes à circuits logiques et numériques.
C. Réaliser des montages contenant des mémoires.	• Conformité du schéma avec la représentation graphique. • Tracé adéquat du schéma. • Conformité du schéma de montage avec le cahier de charges ; • Sélection judicieuse des composants. • Fonctionnement correct du circuit.

Chapitre I

SYSTEMES DE NUMERATION, CODAGE ET ARITHMETIQUE BINAIRE

1. Quelques définitions :

La numération arabe est universellement adoptée, étant donné sa capacité à faire les calculs. Il s'agit du système de numération avec la base 10. Comme on le sait, dans cette base familière :

- ✓ On utilise les 10 symboles, appelés chiffres, de l'ensemble : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- ✓ Un nombre quelconque peut s'écrire en utilisant les puissances de 10 ;

Exemple : $571 = 5 \times 10^2 + 7 \times 10^1 + 1 \times 10^0$.

Les nombres tels que nous les utilisons sont, en réalité, une convention d'écriture. Tout nombre entier positif peut s'écrire sous la forme d'un polynôme arithmétique.

$$N = an \times \mathbf{B}^n + an\text{-}1 \times \mathbf{B}^{n-1} + ...+ a1 \times \mathbf{B}^1 + ao \times \mathbf{B}^0$$

Où B **est la base**, **a** est le chiffre de rang **n** et **n** représente le poids.
Dans la **base B**, on a besoin de **B symboles** pour écrire tous les nombres.

Mais la représentation des nombres avec le système décimal (base 10) n'est pas la seule utilisée. On peut donc en utiliser d'autres, en particulier le système binaire (**base 2**). Les circuits logiques ne connaissent que deux valeurs 0 et 1 ; alors on peut faire des calculs et des traitements comme on le fait avec le système décimal. Ceci permet de rendre le traitement de l'information automatique et rapide.

- ✓ **DIGIT** : Contraction de "digital unit" unité digitale. Un digit est un élément d'information numérique de base quelconque.
  Exemple : Les nombres 1644 (base 10) et A84F (base 16) sont constitués chacun de 4 digits.
- ✓ **POIDS D'UN DIGIT** : La valeur de chaque digit dépend de sa position. A chaque rang (position), est affecté un poids. Les positions des digits d'un nombre écrit en base B ont pour poids des puissances de B.
- ✓ **BIT** : Contraction de "binary digit" digit binaire. Un bit ne peut prendre que deux états 0 ou 1.
  Exemple : le nombre binaire 10100101 est constitué de 8 bits.
- ✓ **MSD** : C'est le digit le plus significatif, de poids le plus fort (Most Significant Digit).
  Exemple : pour le nombre A4F5, le MSB est A
- ✓ **LSD** : C'est le digit le moins significatif, de poids le plus faible (Least Significant Digit).
  Exemple : pour le nombre A4F5, le LSB est 5
- ✓ **MOT** : Un MOT est l'association (concaténation) de plusieurs digits ou bits (peut être aussi appelé courant un « nombre »)
  - ➢ un mot de 4 bits s'appelle un quartet; ex : 1010
  - ➢ un mot de 8 bits s'appelle un octet; ex : 1011 0110
- ✓ **BASE** : Un nombre est écrit en base B, chacun de ses digits peut être écrit avec B symboles différents :
  - ➢ Symboles en base 10 [10 symboles]: 0 1 2 3 4 5 6 7 8 9
  - ➢ Symboles en base 16 [16 symboles]: 0 1 2 3 4 5 6 7 8 9 A B C D E F

- ✓ **CAPACITE DE COMPTAGE** : Avec N digits écrits en base B, on peut compter de 0 à B^N-1, soit B^N nombres différents.

Exemple 1 : avec un nombre de 3 digits en base 10, on peut compter de 0 à 999 (10^3-1), soit 10^3 nombres différents.

Exemple 2 : avec un nombre de 4 digits en base 2, on peut compter de 0 à 15 (2^4 - 1), soit 2^4 nombres différents.

2. Codage de l'information binaire

Un système électronique traite les informations en binaire de différentes natures. Par exemple, en traitement de texte, on manipule des caractères ; pour qu'un ordinateur traite ces caractères, il faut associer alors à chaque caractère un nombre binaire. Cette association s'appelle "Codage" de l'information binaire et permet d'utiliser plusieurs codes suivant le domaine d'application. L'opération inverse s'appelle "Décodage" ou "Transcodage". On étudie en particulier : Le code binaire pur, le code GRAY, le code BCD, le code ASCII et le code à barre.

2-1- Le code binaire pur

Il est aussi appelé code binaire naturel**.** C'est le code binaire sans aucune codification, c'est-à-dire qui découle directement du principe général de la numération. C'est le code naturel utilisé dans les systèmes numériques (ordinateur, etc.). Le tableau suivant donne le code binaire pur pour un exemple d'un mot de 4 bits (A3A2A1A0) :

Valeur décimale	Code binaire			
	A3	A2	A1	A0
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

Pour remplir rapidement une table de vérité avec toutes les combinaisons possibles des variables d'entrée, on procède comme en décimal :

- On part du poids faible (A_0), qui balaye la plage 0 à 1 ;
- On passe au poids suivant (A_1), qui reste à 0 pour la plage 0 à 1 de A_0 , puis à 1 pour la même plage ;
- Et ainsi de suite.

2-2- Le code GRAY

Dans les systèmes industriels où on a besoin de mesurer un déplacement linéaire ou angulaire, on utilise le "code GRAY". La raison de ce choix est que si le système qui mesure le déplacement (capteur) utilise le code binaire pur, le déplacement d'une position à une autre voisine génère des combinaisons intermédiaires fausses, car plusieurs bits varient en même temps.

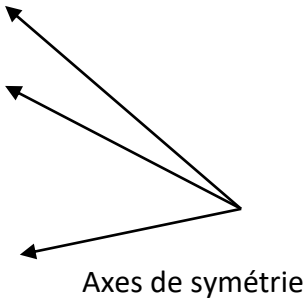
Pour remédier à ce problème, il suffit de coder chaque position de façon que les valeurs de positions successives ne différent que d'un seul bit. C'est pour cela qu'on l'appelle "code à distance unité". On l'appelle aussi "code binaire réfléchi" parce que pour le construire, on procède par réflexion comme l'indique le tableau suivant avec 4 bits :

- ✓ on a 16 combinaisons différentes ;
- ✓ dans le passage d'une combinaison à une autre, il n'y a qu'un bit qui change.

Le code GRAY est aussi utilisé dans l'écriture des tableaux de Karnaugh, lors de la simplification des équations logiques.

Le principe d'obtention du code GRAY est donné comme suit:

Valeur décimale	Code binaire				Code GRAY			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



2-3- Le code BCD :

Le code BCD (Binary Coded Decimal) qui veut dire Binaire Codé en Décimal est la traduction en binaire des 9 premiers chiffres du système décimal.
Si on a un nombre décimal N à m chiffres, il sera codé en BCD sur (m x 4) bits : chaque chiffre décimal est traduit en code BCD sur 4 bits.

Exemple : $(571)_2$ = 1000111011 en binaire pur
= 5 - 7 - 1
= 0101 - 0111 - 0001 en BCD

Le code BCD est utilisé pour les afficheurs lumineux.

2-4- Le code ASCII :

Le code ASCII (American Standard Code for Information Interchange) est un code qui représente les caractères éditables ou non éditables : éditables parce que l'on peut les éditer comme le caractère "A" et non éditables comme le cratère "Escape" ou "Return". Il est codé sur 7 bits (b6 b5 b4 b3 b2 b1 b0), ce qui permet de représenter 128 (2^7) caractères différents. La table suivante montre un tel codage. Par exemple, Le code de la lettre "A" (majuscule) est :

- ✓ en binaire : b6 b5 b4 b3 b2 b1 b0 = 1000001 ;
- ✓ en hexadécimal 41 ;
- ✓ en décimal 65.

b6	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
b5	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
b4	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

b3b2b1b0
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Table ASCII standard (codes de caractères de 0 à 127)

000		(nul)	016	▶	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺	(soh)	017	◀	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☻	(stx)	018	↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥	(etx)	019	‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦	(eot)	020	¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	♣	(enq)	021	§	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠	(ack)	022	▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	•	(bel)	023	↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	◘	(bs)	024	↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009		(tab)	025	↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010		(lf)	026		(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂	(vt)	027	←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	♀	(np)	028	∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013		(cr)	029	↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	♫	(so)	030	▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	☼	(si)	031	▼	(us)	047	/	063	?	079	O	095		111	o	127	⌂

3. Les systèmes de numération

De nombreux systèmes de numération sont utilisés en électronique numérique. Les plus courants sont les systèmes de numération suivants :

- ✓ Binaire (Base 2)
- ✓ Décimal (Base 10)
- ✓ Hexadécimal (Base 16)
- ✓ Octal (Base 8)

3-1- Principe d'une base

La base est le nombre qui sert à définir un système de numération. La base du système décimal est dix alors que celle du système octal est huit. Quel que soit la base numérique employée, elle suit la relation suivante :

$$\sum_{i=0}^{i=n}(b_i a^i)=b_i a^n+.....+b_5 a^5+b_4 a^4+b_3 a^3+b_2 a^2+b_1 a^1+b_0 a^0$$

bi : chiffre de la base de rang i et **ai** : puissance de la base a d'exposant de rang i

3-2- Le système décimal (base 10)

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

Tout nombre écrit dans le système décimal vérifie la relation suivante :

$745 = 7 \times 100 + 4 \times 10 + 5 \times 1$
$745 = 7 \times 10 \times 10 + 4 \times 10 + 5 \times 1$
$745 = 7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$

Chaque chiffre du nombre est à multiplier par une puissance de 10 : c'est ce que l'on nomme le **poids du chiffre**.

L'exposant de cette puissance est nul pour le chiffre situé le plus à droite et s'accroît d'une unité pour chaque passage à un chiffre vers la gauche.

$$\mathbf{12\ 435 = 1 \times 10^4 + 2 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 5 \times 10^0}$$

Cette façon d'écrire les nombres est appelée **système de numération de position**.

Dans notre système conventionnel, nous utilisons les puissances de 10 pour **pondérer** la valeur des chiffres selon leur position, cependant il est possible d'imaginer d'autres systèmes de nombres ayant comme base un nombre entier différent.

3-3- Le système octal (base 8)

Le système octal utilise un système de numération ayant comme **base 8** (octal => latin octo = huit). Il faut noter que dans ce système nous n'aurons plus 10 symboles mais 8 seulement : 0, 1, 2, 3, 4, 5, 6, 7. Ainsi, un nombre exprimé en base 8 pourra se présenter de la manière suivante : $\mathbf{(745)_8}$

Lorsque l'on écrit un nombre, il faudra bien préciser la base dans laquelle on l'exprime pour lever les éventuelles indéterminations (745 existe aussi en base 10). Ainsi le nombre sera mis entre parenthèses (745 dans notre exemple) et indicé d'un nombre représentant sa base (8 est mis en indice).

Cette base obéira aux mêmes règles que la base 10, vue précédemment, ainsi on peut décomposer $(745)_8$ de la façon suivante :

$(745)_8 = 7 \times 8^2 + 4 \times 8^1 + 5 \times 8^0$
$(745)_8 = 7 \times 64 + 4 \times 8 + 5 \times 1$
$(745)_8 = 448 + 32 + 5$
Nous venons de voir que :
$\mathbf{(745)_8 = (485)_{10}}$.

3-4- Le système binaire (base 2)

Dans le système binaire, chaque chiffre peut avoir 2 valeurs différentes : 0, 1.
De ce fait, le système a pour **base 2**.
Tout nombre écrit dans ce système vérifie la relation suivante :
$(10\ 110)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
$(10\ 110)_2 = 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$
donc : $(10110)_2 = (22)_{10}$.

La correspondance entre base 2, base 10 et base 8 est indiquée dans le tableau ci-après :

Système décimale	Système octal	Système binaire
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111
16	20	10000
17	21	10001
-	-	-
-	-	-
63	77	111111
64	100	1000000
65	101	1000001

3-5- Le système hexadécimal (base 16)

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

Un nombre exprimé en base 16 pourra se présenter de la manière suivante :
$(5AF)_{16}$

La correspondance entre base 2, base 10 et base 16 est indiquée dans le tableau ci-après :

Base 10	Base 16	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

4. Conversion entre bases

4-1- Conversion de la base 2, 8 ou 16 vers la base 10

On exploite directement la forme polynomiale de la représentation binaire, octale ou hexadécimale du nombre.

Exemple 1 : $(110101)_2 = 1x2^5+1x2^4+0x2^3+1x2^2+0x2^1+1x2^0 = 32+16+0+4+0+1 = (53)_{10}$

Exemple 2 : $(305)_8 = 3x8^2+0x8^1+5x8^0 = (197)_{10}$

Exemple 3 : $(61F)_{16} = 6x16^2+1x16^1+15x16^0 = (1567)_{10}$

Exemple 4 : $(10110, 01)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

$= 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times 0,5 + 1 \times 0,25$

$= (22, 25)_{10}$

Exemple 5 : $(372, 06)_8 = 3 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2}$

$= 3 \times 64 + 7 \times 8 + 2 \times 1 + 0 \times 0,125 + 6 \times 0,015625$

$= (250, 09375)_{10}$

Exemple 6 : $(FD, 2A)_{16} = F \times 16^1 + D \times 16^0 + 2 \times 16^{-1} + A \times 16^{-2}$

$= 15 \times 16 + 13 \times 1 + 2 \times 0,0625 + 10 \times 0,00390625$

$= (253, 1640625)_{10}$

4-2- Conversion de la base 10 aux base 2, 8 et 16

Cette conversion se fait en deux parties :

- La partie entière.
- La partie fractionnaire.

La conversion de la partie entière consiste à répéter la division par (2, 8, ou 16) du nombre décimal à convertir et au report des restes jusqu'à ce que le quotient soit 0.

La conversion de la partie fractionnaire consiste à multiplier le nombre **fractionnaire** par (2, 8,16) et noter la **partie** entière et la **partie fractionnaire** obtenues. Multiplier la **partie fractionnaire** obtenue sur la ligne précédente par (2, 8,16), et noter le résultat (**partie** entière et **partie fractionnaire**).

Exemple 1 : Conversion du nombre 91,2 en base 2

Conversion de la partie entière				**Conversion de la partie fractionnaire**	
		Position	Reste		
91/2	45	2^0	**1**	0,2 x2	**0**,4
45/2	22	2^1	**1**	0,4x2	**0**,8
22/2	11	2^2	**0**	0,8x2	**1** ,6
11/2	5	2^3	**1**	0,6x2	**1**,2
5/2	2	2^4	**1**	**0,2x2**	
2/2	1	2^5	**0**		
1/2	**0**	2^6	**1**		

$(\mathbf{91,2})_{10} = (\mathbf{1011011,00110011\ldots})_2$

Exemple 2 : Conversion du nombre 459,3 en base 8

Conversion de la partie entière				**Conversion de la partie fractionnaire**	
		Position	Reste		
459/8	57	8^0	**3**	0,3 x8	**2**,4
57/8	7	8^1	**1**	0,4x8	**3**,2
7/8	0	8^2	**7**	0,2x8	**1** ,6
				0,6x8	**4**,8
				0,8x8	**6**,4
				0,4x8	**3**,2

$(\mathbf{459,3})_{10} = (\mathbf{713,231463146\ldots.})_8$

Exemple 3 : Conversion du nombre 751,1 en base 16

Conversion de la partie entière				**Conversion de la partie fractionnaire**	
		Position	Reste		
751/16	46	16^0	**15 (F)**	0,1 x16	**1**,6
46/16	2	16^1	**14 (E)**	0,6x16	**9**,6
2/16	0	16^2	**2**	**0,6x16**	**9** ,6

$(\mathbf{751,1})_{10} = (\mathbf{2EF,1999\ ....})_{16}$

4-3- Conversion de la base binaire à la base octale

On obtient l'équivalent octal du nombre binaire en le partageant en tranche de 3 chiffres de droite à gauche pour la partie entière et de gauche à droite pour la partie fractionnaire, puis en remplaçant chaque tranche par son équivalent octal.

Exemple 1 : $(111000110101)_2$ = 111/000/110/101
= 7 / 0 / 6 / 5
= $(7065)_8$

Exemple 2 : $(011010101110,001011100)_2$ = 011/010/101/110 , 001/011/100
= 3/2/5/6 , 1/3/4
= $(3256,134)_8$

Exemple 3: $(2\ 4\ 3,2\ 1)_8$ = 2/4/3,2/1 = 010/100/011,010/001
= $(010100011,010001)_2$

4-4- Conversion de la base 2 à la base 16

La conversion de la base 2 à la base 16 (et inversement) se fait aisément, la base 16 étant un multiple entier de la base 2. Elle permet de représenter sous une forme réduite un nombre binaire.
$2^4 = 16$ => un groupe binaire de 4 bits est **transcodable** directement en un digit hexadécimal.
Méthode : On divise le nombre binaire en tranches de 4 bits (à partir du LSB). Chacun des quartets est ensuite converti en un digit hexadécimal par simple sommation pondérée.

Exemple 1 : $(111000110101)_2$ = 1110 - 0011 - 0101
= E - 3 - 5
= $(E35)_{16}$

Exemple 2 : $(D4C7)_{16}$ = D - 4 - C - 7
= 1101- 0100 – 1100 - 0111= $(1101\ 0100\ 1100\ 0111)_2$

Exemple 3 : $(0011110100101110\ ,11010100)_2$ =0011/1101/0010/1110,1101/0100
=3/D/2/E,D/4
=$(3D2E,D4)_{16}$

Exemple 4: $(B\ 3\ 0,\ 1\ A)_{16}$ =1011/0011/0000,0001/1010
=$(101100110000,00011010)_2$

5. Conversion entre codes

5-1- Conversion du code binaire (naturel) en code de Gray (binaire réfléchi)

Soit un nombre N en binaire naturel, pour obtenir son équivalent n en binaire réfléchi, il suffit d'effectuer l'opération suivante :

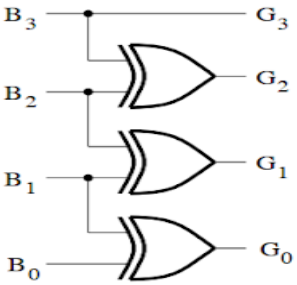
$$n = \frac{N \oplus 2N}{2}$$

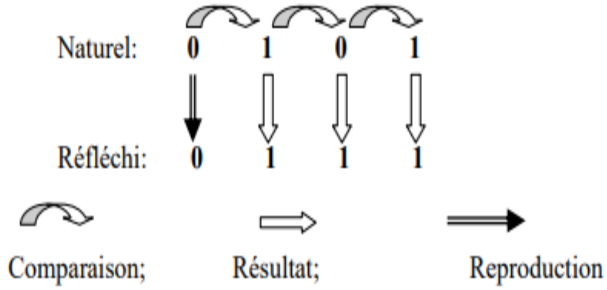
- On effectue un **décalage de la droite vers la gauche** sur le nombre binaire naturel N pour obtenir 2N,
- On effectue l'opération **OU Exclusif** de N et 2N

- Puis on effectue un **décalage de la gauche vers la droite** du résultat de l'OU Exclusif nous (divisons par 2).

Le résultat de cette opération donne le code gray correspondant.

Une autre méthode consiste à appliquer la règle suivante : $G_n=B_n \oplus B_{(n+1)}$.





Exemple :

Nombre Binaire N	**Code Gray**
N = 0101	**0111**
N = 0111	**0100**
N= 001101	**001011**
N= 0100100	**0110110**

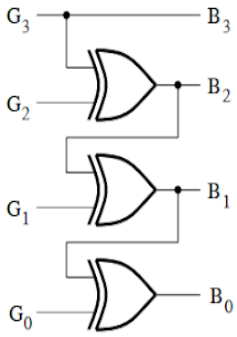
La construction du code Gray pour les nombres de 0 à 15 est représentée par le tableau suivant :

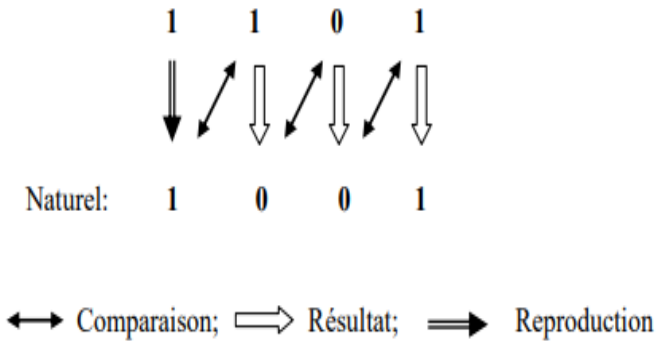
Nombre décimal	Code binaire pur				Code Gray			
	B4	B3	B2	B1	G4	G3	G2	G1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0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

5-2- Conversion du code de Gray au code binaire

Pour la conversion du code Gray en code binaire la relation suivante s'apparente à l'équation vue pour le convertisseur inverse : $G_n=B_n \oplus B_{(n+1)}$.

Là encore les fonctions OU Exclusif sont de mise pour la réalisation du circuit de transcodage.





Exemple 1:

Code Gray	1101	00010110	10010110	01100111
Code Binaire	**1001**	**00011011**	**11100100**	**01000101**

5-3- Exercices d'application

Exemple 1 : Donner l'équivalent en code Gray des nombres binaires suivants :

a/ $(11)_2$ = (.....)$_{Gray}$

b/ $(1011)_2$ = (... .....)$_{Gray}$

c/ $(10111)_2$ = (... .....)$_{Gray}$

d/ $(11111011101)_2$ = (... .....)$_{Gray}$

Exemple 2 : Donner l'équivalent binaire des codes Gray suivants :

a/ $(11)_{Gray}$ = (.....)$_2$

b/ $(1011)_{Gray}$ = (... .....)$_2$

c/ $(10111)_{Gray}$ = (... .....)$_2$

d/ $(110110)_{Gray}$ = (... .....)$_2$

6. Notions d'arithmétique binaire

On peut faire des calculs arithmétiques, puisqu'on a les mêmes règles de calcul qui s'appliquent à toutes les bases. On traite alors de l'arithmétique binaire et on se limite aux cas de l'addition et de la soustraction.

6-1- Représentation des nombres :

Dans les calculs, on manipule des nombres positifs et négatifs ; il faut alors coder le signe algébrique. Plusieurs modes de représentation sont adoptés en fonction des calculs à effectuer et les caractéristiques technologiques des systèmes de traitement.

Représentation par valeur absolue et signe :

Pour le bit de signe, on adopte la convention **0** pour le signe (+) et **1** pour le signe (-).
Par exemple :$(+35)_{10}$ = **0** 100011 et $(-35)_{10}$ = **1** 100011.
Cette solution a comme inconvénient la complexité de la réalisation technologique due à :

- Un traitement spécifique du signe ;
- Une double représentation du 0.

Représentation par complément à 2 :

Soit un nombre binaire A sur n bits et son complément $\overline{A}$ (nommé aussi complément à 1 de A), on a : $A + \overline{A} = 2^n - 1$
On déduit de cette relation que : $-A = \overline{A} + 1 - 2^n$.

Comme le (2^n) ne rentre pas dans le format défini (An-1 An-2… A1 A0), il sera ignoré. On a alors :

$$-A = \overline{A} + 1.$$

Le terme $(\overline{A} + 1)$ est appelé complément à 2.

Exemple : Pour n = 4,on obtient :

(A)10	(A) 2	$(\overline{A})2$	$(\overline{A} + 1)2$	(-A)10
7	0111	1000	1001	-7
6	0110	1001	1010	-6
5	0101	1010	1011	-5
4	0100	1011	1100	-4
3	0011	1100	1101	-3
2	0010	1101	1110	-2
1	0001	1110	1111	-1
0	0000	1111	0000	0

On remarque que :

- ✓ Le MSB représente le signe avec 0 (+) et 1 (-).
- ✓ Le zéro n'a qu'une seule représentation ;
- ✓ Alors pour effectuer une soustraction, il suffit de faire une addition avec le complément à 2. Le résultat se lit directement en complément à 2 :
  - Si le signe est + la lecture est directe;
  - Si le signe est -, on convertit le résultat en recherchant le complément à 2 de celui-ci.

6-2- L'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 + 0 = 1

1 + 1 = 10 (= 0 + report de 1 sur la gauche) , 1 + 1 + 1 = 11 (= 1 + report de 1 sur la gauche)

Exemple :

0111
+
1011
=
10010

6-3- Multiplication binaire :

On multiplie les nombres binaires de la même façon qu'on multiplie les nombres décimaux. En réalité, le processus est plus simple car les chiffres du multiplicateur sont toujours 0 ou 1, de sorte qu'on multiplie toujours par 0 ou par 1. La multiplication binaire se base sur les quatre opérations suivantes: (**1 x 1 = 1**, **1 x 0 = 0**, **0 x 1 = 0**, **0 x 0 = 0**).

Exemple:

	1 0 0 1	**multiplicande = $(9)_{10}$**
	· 1 0 1 1	**multiplicateur = $(11)_{10}$**
=	1 0 0 1	
	1 0 0 1	
	0 0 0 0	
	1 0 0 1	**produits partiels**
=	1 1 0 0 0 1 1	**produit final = $(99)_{10}$**

6-4- Soustraction binaire :

La soustraction repose sur les quatre opérations suivantes :
(**0 – 0 = 0, 1 – 0 = 1, 0 – 1 = 1** on emprunt **1, 1 – 1 = 0**)

Exemple 1 :

11011
-
00111
11
=
10100

Exemple 2 :

1100
-
0111
111
=
0101

6-5- Division binaire :

Les règles de base sont :
0 / 0 = Indéterminé
0 / 1 = 0
1 / 0 = Impossible
1 / 1 = 1

Exemple 1 :

```
 1 0 0 1  |  1 1        9/3 = 3
-  1 1    |  1 1
 0 0 1 1
-0 0 1 1
 0 0 0 0
```

Exemple 2 :

```
 1 0 1 0 , 0  |  1 0 0      10/4 = 2,5
-1 0 0        |  1 0 , 1
 0 0 1 0
-  0 0 0
     1 0 0
   1 0 0
   0 0 0
```

6-6- Exercices d'application:

Effectuer les opérations arithmétiques suivantes :

Exemple 1 :

0011	1101	110	1111/11
+	-	*	
1101	0010	11	

Exemple 2 :

10110110	1010111	1001	100000/110
+	-	*	
1011101	10101	1100	

Exemple 3 :

1110111			1000010/1011
+			
101101	10110101	1001	
+	-	*	
1101	1110101	1100	

Chapitre II

ALGEBRE DE BOOLE ET FONCTIONS LOGIQUES

1. Introduction

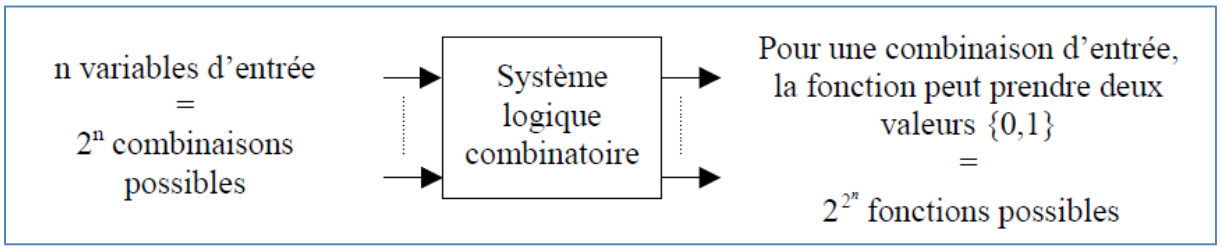
L'algèbre de Boole est l'outil mathématique pour étudier ces dispositifs et les circuits logiques représentent l'outil technologique pour réaliser pratiquement les opérations de cette algèbre. Les circuits qu'on va étudier dans ce chapitre sont dits "**combinatoires**", car l'état de leurs sorties ne dépend que de l'état des entrées.

2. Quelques définitions :

Variable logique : grandeur, représentée par un identificateur (lettre ou nom) qui peut prendre l'une des valeurs **0** ou **1**.
Niveau logique : En électronique, une variable logique est concrétisée par un signal électrique (tension ou courant) qui peut prendre deux **niveaux électriques** (ou niveaux logiques) :

- ✓ le niveau logique **Haut (H)** ou **High**
- ✓ le niveau logique **Bas (L)** ou **Low**

Algèbre de BOOLE : Ensemble de variables à 2 états, de valeur, ou état "**1**" (vrai) ou "**0**" (faux) et muni d'un petit nombre d'opérateurs fondamentaux : **NON**, **ET**, **OU**.
Fonction logique de n variables binaires : groupe de variables reliées par des opérateurs logiques (NON, ET, OU).
Système combinatoire : Un système est **dit "combinatoire**" lorsque qu'à une combinaison des variables binaires d'entrée correspond une (et une seule) combinaison des variables de sorties.



Note : on parle de systèmes combinatoires par opposition aux **systèmes séquentiels**, dans lesquels les variables de sortie dépendent à la fois des variables d'entrée et de l'état antérieur des variables de sortie.

La fonction binaire est une application qui fait correspondre à un mot binaire d'entrée X = Xn-1Xn-2….X1X0 une variable binaire de sortie Y.

$$F : E^n \rightarrow E \quad \text{avec } E = \{0,1\}$$
$$X \rightarrow Y$$

- ✓ Une fonction binaire est dite **fonction combinatoire**, si pour une des combinaisons d'entrées correspond un seul état de la sortie (0 ou 1).
- ✓ Un **système logique combinatoire** est un système qui possède plusieurs fonctions logiques combinatoires. A partir d'un mot binaire d'entrée $X_{n-1}X_{n-2}….X_1X_0$, ce système permet d'élaborer plusieurs variables de sorties $Y_{m-1}Y_{m-2}….Y_1Y_0$ telles que :

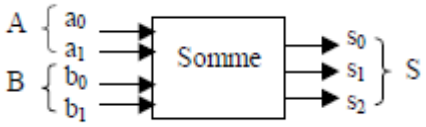
$$Y_i = F_i(X_{n-1}X_{n-2}….X_1X_0) \quad \text{avec } 0 \leq i \leq m-1$$

Exemple :

Soient A et B deux nombres décimaux compris entre 0 et 3. Les nombres A et B sont codés en binaire naturel (chacun sur 2bits). Un système logique combinatoire doit réaliser le calcul du code binaire de la somme A + B (de 0 à 6).

Ce système logique combinatoire comporte :

- 4 variables d'entrées :
  - a1et a0 correspondant au code binaire naturel de A,
  - b1et b0 correspondant au code binaire naturel de B
- 3 variables de sorties :
  - s2, s1et s0 correspondant au code binaire de la somme A+B.



Les variables de sorties sont des fonctions logiques combinatoires des variables d'entrées.

3. Notion de table de vérité :

Une **table de vérité** est une **représentation graphique** (tableau) faisant connaître la réaction du circuit logique, c'est à dire l'état de la sortie S en fonction de toutes les combinaisons de valeurs (0 ou 1) que peuvent prendre les variables binaires d'entrées E1, E2, …, Ei, …,En.

Le tableau suivant donne un exemple d'une table de vérité pour une fonction logique à deux entrées **E1** et **E2** et une sortie **S.**

E1	E2	S
0	0	0
1	0	1
0	1	1
1	1	0

4. Les opérateurs de base de l'algèbre de binaire

4-1- Présentation des opérateurs élémentaires :

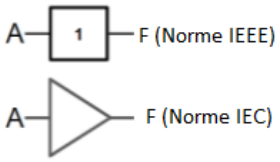
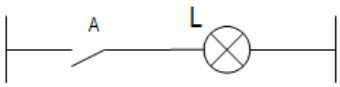
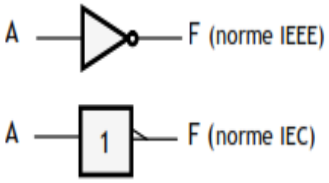
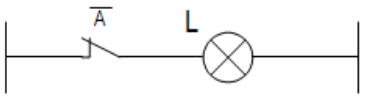
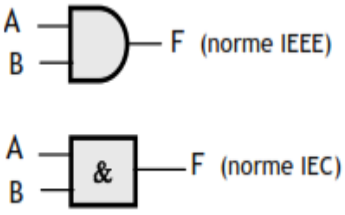
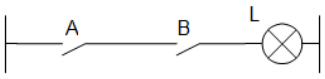
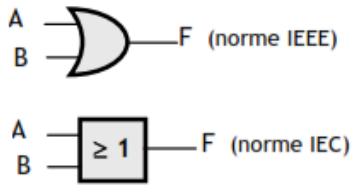
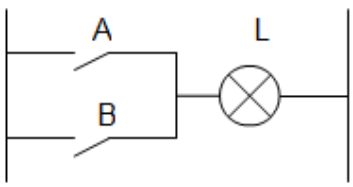
Il existe trois opérations élémentaires pour définir une algèbre de Boole qui sont :

- ✓ L'opérateur logique : Non (Not) représenté par une barre ($\bar{}$);
- ✓ le produit logique : ET (AND) représenté par le signe (**.**) ou ($\wedge$) ;

✓ la somme logique : OU (OR) représenté par le signe (**+**) ou (v) ;

Le tableau suivant donne les propriétés des opérations booléennes de base.

IEEE: Institute of Electrical and Electronics Engineers.
IEC : International Electrotechnical Commission

Fonction	Symbole et propriétés	Equation et table de vérité	Schéma à contact
Identité : OUI		$L = A$ A | L 0 | 0 1 | 1	
Inversion : NON (**NOT**)		$L = \bar{A}$ A | L 0 | 1 1 | 0	
Produit logique : ET (**AND**)	 **Propriétés** La fonction AND est commutative: F = A.B = B.A. La fonction AND est associative: F = A.(B.C) = (A.B).C = A.B.C. Identités remarquables : **X.0 = 0 ; X.1=X ;** **X .X=X ; X.$\bar{X}$ =0**	$L = A.B$ A | B | L 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	
Somme logique : OU (**OR**)	 **Propriétés** La fonction AND est commutative: F = A+B = B+A. La fonction AND est associative:	$L = A + B$ A | B | L 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1	

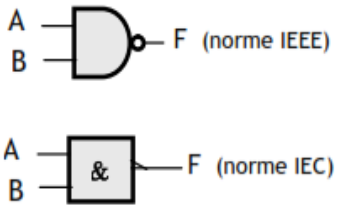
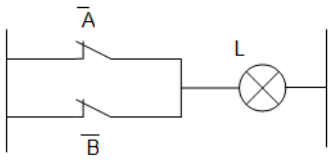
	F = A+(B+C) = (A+B)+C = A+B+C. Identités remarquables : **X+0 = X ; X+1=1 ;** **X +X=X ; X+$\bar{X}$ =1**		

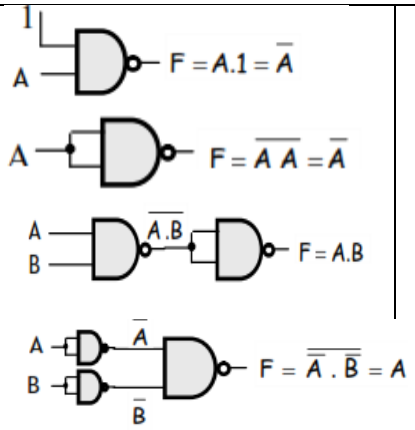
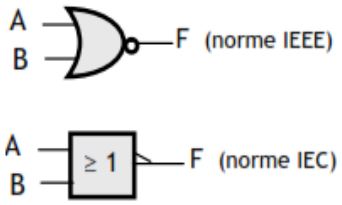
4-2- Présentation des autres fonctions de base:

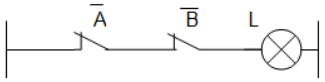
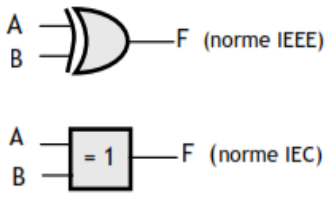
Les autres fonctions logiques de base sont :

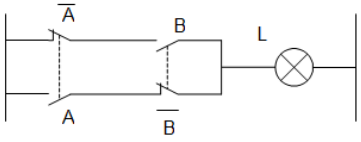
- ✓ La fonction Non ET (NAND) ;
- ✓ La fonction Non OU (NOR) ;
- ✓ La fonction OU exclusif (XOR).

Le tableau suivant donne les propriétés essentielles des fonctions logiques de base citées ci-dessus.

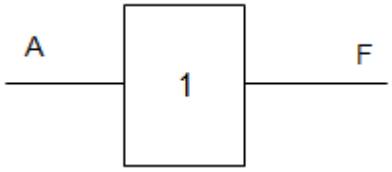
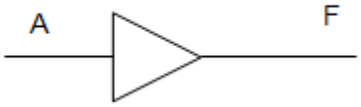
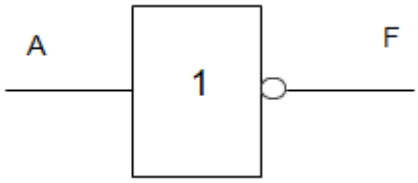
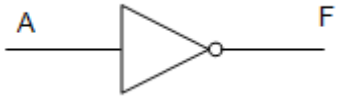
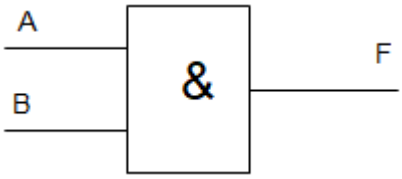
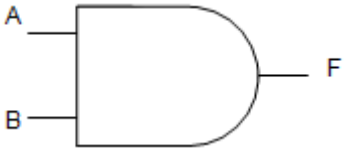
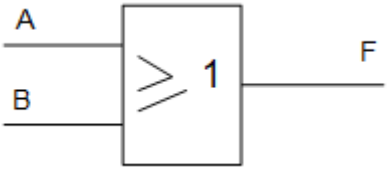
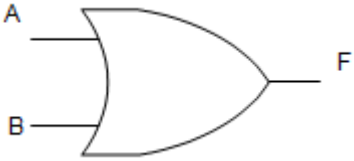
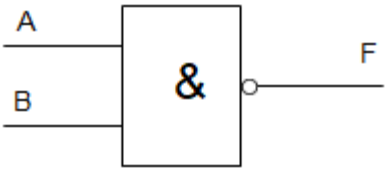
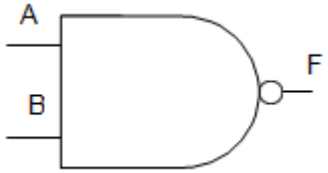
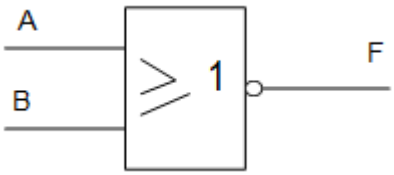
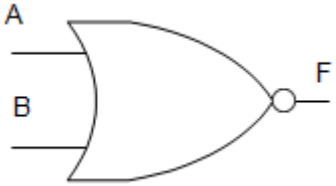
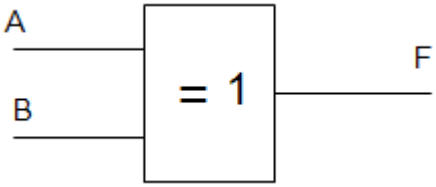
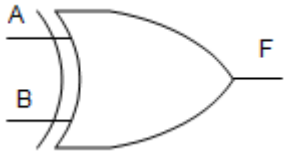
Fonction	**Symbole**	**Equation et table de vérité**	**Schéma à contact**
Fonction (**NAND**)	 La fonction est commutative : $F = \overline{A.B} = \overline{B.A}$ La fonction n'est pas associative : $F = \overline{A.(\overline{B.C})} \neq \overline{(\overline{A.B}).C}$ La fonction est généralisable pour n entrées. Elle permet de réaliser toutes les opérations de base :NON, ET ,OU	$F = \overline{A.B}$ $= \bar{A} + \bar{B}$ A B F 0 0 1 0 1 1 1 0 1 1 1 0	

<table>
<tr>
<td></td>
<td> 
</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fonction (NOR)</td>
<td> 

Comme la fonction NAND, la fonction NOR est commutative, mais non associative ; elle est aussi généralisable pour n entrées.
L'opérateur NOR est un système logique complet, comme le NAND.</td>
<td>$F = \overline{A + B}$
$= \overline{A} . \overline{B}$
<table>
<tr><td>A</td><td>B</td><td>F</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td>
<td> 
</td>
</tr>
<tr>
<td>Fonction (XOR)</td>
<td> 

L’opération XOR est commutative :
F = A⊕B = B⊕A.
L’opération XOR est associative :
F = A⊕ (B⊕C) = (A⊕B) ⊕C = A⊕B⊕C.
L’opération XOR n'est pas généralisable pour n entrées.</td>
<td>$F = A \oplus B$
<table>
<tr><td>A</td><td>B</td><td>F</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td>
<td> 
</td>
</tr>
</table>

En résumé

	Symboles	
Fonction	**Norme IEEE**	**Norme IEC**
OUI : $\mathbf{F = A}$		
NON : $\mathbf{F = \overline{A}}$		
ET (AND) : $\mathbf{F = A \,.\, B}$		
OU (OR): $\mathbf{F = A + B}$		
NON ET (NAND): $\mathbf{F = \overline{A \,.\, B}}$		
NON OU (NOR): $\mathbf{F = \overline{A + B}}$		
OU exclusif : $\mathbf{F = A \oplus B}$		

5. Représentation des fonctions logiques :

Pratiquement, une fonction logique est représentée par :

- ✓ sa table de vérité ou son tableau de Karnaugh;
- ✓ son **équation logique** qui n'est qu'une association de sommes et de produits logiques;
- ✓ Son logigramme qui est une représentation symbolique, sous forme d'un schéma, formé par les différentes liaisons entres les symboles des opérateurs élémentaires.

5-1- Représentation des fonctions logiques par Table de vérité

On appelle table de vérité, un tableau qui indique pour chacune des combinaisons possibles des variables d'entrée la valeur de la variable de sortie.

Règles de construction

La table de vérité est une compilation, sous forme de tableau, de tous les états logiques de la sortie en fonction des états logiques des entrées.

Les étapes à suivre pour construire une table de vérité :

- Écrire, sur une première ligne, le nom des variables d'entrées et celui de variable de sortie;
- Diviser le tableau en un nombre de colonnes égal au total des entrées et de la sortie;
- Déterminer le nombre de combinaisons possibles à l'aide des variables d'entrée : soit $2^{\text{nombre d'entrée}}$
- Tracer des lignes horizontales en un nombre égal au nombre de combinaisons possibles;
- Remplir chaque ligne par une combinaison possible des variables d'entrée : ça revient àcompter en binaire de 0 à $(2^n - 1)$;
- Inscrire, dans la colonne « sortie », la valeur de la fonction pour chaque combinaison

Exemple : Table de vérité d'une fonction logique à 3 variables d'entrée.

Entrées			**Sortie**
A	**B**	**C**	**S**
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

5-2- Écriture d'une équation à partir d'une table de vérité

Il existe 2 méthodes :

Méthode 1 : Produit de sommes :

On considère les lignes de la table de vérité dont la sortie est à **l'état logique « 0 »**sous forme d'une **somme logique « OU ».** Les parties d'équation ainsi obtenues peuvent être réunies par le **produit logique « ET »**.

Variable = 1 ➔ $\overline{\text{Variable}}$
Variable = 0 ➔ Variable

Exemple : Soit la table de vérité suivant à 3 variables

Entrées			Sortie	
A	**B**	**C**	**S**	
0	0	0	**0**	$A + B + C$
0	0	1	**0**	$A + B + \overline{C}$
0	1	0	**0**	$A + \overline{B} + C$
0	1	1	**1**	
1	0	0	**0**	$\overline{A} + B + C$
1	0	1	**0**	$\overline{A} + B + \overline{C}$
1	1	0	**1**	
1	1	1	**1**	

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

Méthode 2 : Somme de produits:

On considère les lignes de la table de vérité dont la sortie est à **l'état logique « 1 »** sous forme d'un **produit logique « ET** ». Les parties d'équation ainsi obtenues peuvent être réunies par la **somme logique « OU »**.

Exemple : Soit la table de vérité suivant à 3 variables

Entrées			Sortie	
A	**B**	**C**	**S**	
0	0	0	**0**	
0	0	1	**0**	
0	1	0	**0**	
0	1	1	**1**	$\overline{A} . B . C$
1	0	0	**0**	
1	0	1	**0**	
1	1	0	**1**	$A . B . \overline{C}$
1	1	1	**1**	$A . B . C$

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

Définitions :

- ✓ On appelle **minterme** de n variables, un **produit logique de ces variables**. Avec n variables, on a 2^n mintermes (autant de combinaisons possibles den éléments binaires).
- ✓ On appelle **maxterme** de n variables, **une somme logique de ces variables**. Avec n variables, on a 2^n maxtermes (autant de combinaisons possibles den éléments binaires).

Exemple : Pour 2 variables a et b, on a :

- ✓ 4 mintermes : $\bar{a}.\bar{b}\ , \bar{a}.b, a.\bar{b}\ , a.b$
- ✓ 4 maxtermes : $\bar{a}+\bar{b}\ , \bar{a}+b, a+\bar{b}\ , a+b$

A partir de la table de vérité d'une fonction logique, il est possible d'obtenir deux expressions algébriques (écrites sous deux formes différentes mais équivalentes) correspondantes à la fonction. Elles représentent les expressions canoniques de la fonction considérée :

- ✓ La 1ère forme canonique (somme de produits algébriques ou **somme de mintermes**) est obtenue en considérant les états d'entrée pour lesquels **la fonction vaut 1**.
- ✓ La 2ème forme canonique (produits de sommes algébriques ou **produit de maxtermes**) est obtenue en considérant les états d'entrée pour lesquels **la fonction vaut 0**.

Exemple :

Soit un système caractérisé par la fonction F(a,b) définit par la table de vérité suivante :

a	b	F(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

F(a,b) est une fonction logique de deux variables a et b. Soient F(0,0), F(1,0), F(0,1) et F(1,1) les 4 valeurs que peut prendre la fonction pour les états d'entrée correspondants. F(a, b) s'écrit alors de la façon suivante :

- ✓ La 1ère forme canonique

  F(a, b) = F(0,0). $\bar{a}.\bar{b} + F(0,1).\bar{a}.b + F(1,0).a.\bar{b} + F(1,1).a.b$

  F(a, b) =1. $\bar{a}.\bar{b} + 0.\bar{a}.b + 0.a.\bar{b} + 1.a.b = \bar{a}.\bar{b} + a.b$

- ✓ La 2ème forme canonique.

  F(a, b) = (F(0,0) + $\bar{a}+\bar{b}$). $(F(0,1) + \bar{a}+b).\left(F(1,0) + a+\bar{b}\right).(F(1,1) + a+b)$

  F(a, b) = (1 + $\bar{a}+\bar{b}$). $(0 + \bar{a}+b).\left(0 + a+\bar{b}\right).(1 + a+b)$

  F(a, b) = ($\bar{a}+b$). $\left(a+\bar{b}\right)$

Représentation décimale d'une fonction logique

Définitions :

On utilise un codage décimal pour représenter les états des variables binaires d'entrée. Par convention ce code est l'équivalent décimal du mot codé en binaire naturel.
$N = a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + ...+ a_1.2^1+a_0.2^0$ où les a_i représentent les variables logiques d'entrées.

Une fonction F peut être définie comme :

- ✓ Une somme des états pour lesquelles elle vaut 1. On note $F = \sum(N1, N2, \dots, Np)$
- ✓ Un produit des états pour lesquelles elle vaut 0. On note $F = \prod(N1, N2, \dots, Nq)$

Exemple :
Soit un système caractérisé par la fonction F(a,b,c) définit par la table de vérité suivante :

N	a	b	c	F
0	0	0	0	**1**
1	0	0	1	**0**
2	0	1	0	**0**
3	0	1	1	**1**
4	1	0	0	**0**
5	1	0	1	**1**
6	1	1	0	**1**
7	1	1	1	**0**

La fonction F peut s'écrire :

- ✓ Une somme des états pour lesquelles elle vaut 1.
  - $F = \sum(0,3,5,6) = \bar{a}.\bar{b}.\bar{c} + \bar{a}.b.c + a.\bar{b}.c + a.b.\bar{c}$
- ✓ Un produit des états pour lesquelles elle vaut 0.
  - $F = \prod(1,2,4,7) = (a+b+\bar{c}).(a+\bar{b}+c).(\bar{a}+b+c).(\bar{a}+\bar{b}+\bar{c})$

Représentation des fonctions incomplètement spécifiées

Il arrive dans certains systèmes logiques combinatoires que la valeur prise par la fonction caractérisant le système ne soit pas spécifiée pour une ou plusieurs des combinaisons des variables d'entrée. C'est-à-dire qu'on ne connaît pas à priori (ou alors ce n'est pas défini) la valeur 0 ou 1 que prend la fonction pour une ou plusieurs combinaisons d'entrée.

Pour une fonction incomplètement spécifiée, il existe :

- ✓ La couverture supérieure d'une fonction incomplète : c'est la fonction logique obtenue en remplaçant toutes les valeurs non spécifiées par des 1.
- ✓ La couverture inférieure d'une fonction incomplète : c'est la fonction logique obtenue en remplaçant toutes les valeurs non spécifiées par des 0.

6. Simplification des fonctions logiques:

Une fonction logique est sous forme normale ou canonique si chacun de ses termes contient toutes les variables, directes ou inverses, dont elle dépend.
Si l'une de ses variables ne figure pas dans un de ses termes, alors elle est sous forme simplifiée. Cette forme est fort bien recherchée pour aboutir à la réalisation pratique avec un minimum de matériel et à moindre coût. Pour cette fin, on utilise, en général, 3 méthodes de simplification :

- ✓ La méthode algébrique ;
- ✓ La méthode graphique à base du diagramme de Karnaugh ;

6-1- Méthode algébrique

Pour chaque variable de sortie figurant sur la table de vérité, on écrit la "somme" logique des lignes ou la variable de sortie prend la valeur 1. Lorsque les états "1" sont plus nombreux que les états "0", il est avantageux d'écrire le complément de la somme logique des lignes où la variable de sortie prend la valeur 0.

Les propriétés essentielles des fonctions combinatoires sont données comme suit :

Distributivité du produit par rapport à la somme :

$$A.(B+C) = A.B + A.C$$

Distributivité de la somme par rapport au produit :

$$(A+B).(A+C) = A.(1+B+C) + B.C = A + B.C$$

Factorisation :

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

Loi d'absorption :

$$A + AB = A(1+B) = A$$
$$AB + \bar{A}B = B(A+\bar{A}) = B$$
$$(A + \bar{A}.B) = (A + A.B) + \bar{A}.B = A + (A.B + \bar{A}.B) = A + B.(A+\bar{A}) = A + B$$

Théorème de De Morgan :

Ce théorème d'une grande utilité, permet de calculer le complément d'une expression logique quelconque (somme de produits ou produit de sommes) :

$$\overline{X+Y} = \bar{X}.\bar{Y}$$
$$\overline{X.Y} = \bar{X} + \bar{Y}$$

Son rôle est de simplifier et optimiser la conception des structures à base d'opérateurs logiques.

D'une façon générale, Le complément d'une expression quelconque s'obtient en complémentant les variables et en permutant les opérateurs "+" et ".".

Exemple :

Soit l'expression logique F définie par : $F = (\bar{a} + b).c + a.c + (\overline{a+b})$
Le complément de F s'écrit alors :

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

$$= (0 + \bar{c}.\bar{a} + a.\bar{b}.\bar{c} + \bar{c}).(a + b)$$
$$= \bar{c}.((\bar{a} + a.\bar{b} + 1).(a + b)$$
$$= \bar{c}.(a + b)$$

Cette méthode peut convenir pour les cas où le nombre de variables d'entrée ne dépasse pas 2 ou 3. Par fois pour simplifier une fonction, on peut ajouter un terme déjà existant.

Dans le cas où le nombre de variables devient trop important, il est plus avantageux d'utiliser une méthode graphique intitulée « Tableau de Karnaugh » permettant de trouver directement une expression simplifiée de l'équation de sortie d'une fonction logique.

6-2- Méthode graphique (Tableau de KARNAUGH)

Le tableau de Karnaugh d'une fonction logique est la transformation de sa table de vérité sous forme d'une table contractée à 2 dimensions.

C'est un diagramme qui reprend les indications de la table de vérité pour les mettre sous une autre forme. Le nombre de cases est égal au nombre de lignes de la table de vérité, ou encore au nombre de combinaisons des variables d'entrée. Le nombre de case du tableau est égal au nombre de combinaisons possibles pour les entrées soit : $C = 2^n$
Avec **C** : nombre de combinaisons et **n** : nombre de variables (ou entrées).

Exemples :

- 1 variable d'entrée A ➔ 2^1 combinaisons = 2 cases,
- 2 variables d'entrée A et B ➔ 2^2 combinaisons = 4 cases
- 3 variables d'entrée A, B et C ➔ 2^3 combinaisons = 8 cases
- n variable d'entrée ➔ 2^n combinaisons = 2^n cases

Pour pouvoir simplifier par suite l'équation à partir du diagramme de Karnaugh, il faut qu'une seule variable change d'état pour deux cases adjacentes. On utilise donc le code Gray au lieu du code binaire.

Disposition des combinaisons à l'intérieur du diagramme de Karnaugh

La méthode consiste principalement à mettre en évidence graphiquement ou visuellement, les groupements de cases, de type : $(AB + A.\bar{B}) = A$

Aussi, dans un tableau de Karnaugh, on peut utiliser une case plusieurs fois selon la relation de la redondance : X + X + … + X = X

Le passage de la table de vérité au tableau de Karnaugh se fait selon la procédure suivante:

- ✓ Chaque ligne de la table de vérité correspond à une case du tableau de Karnaugh ;
- ✓ Les cases sont disposées de telle sorte que le passage d'une case à une case voisine se fasse par changement de l'état d'une seule variable à la fois en utilisant le code GRAY.

Exemples :

Table de vérité | **Tableau de Karnaugh**

A	B	S
0	0	
0	1	
1	0	
1	1	

B\A	0	1
0		
1		

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

C\AB	00	01	11	10
0				
1				

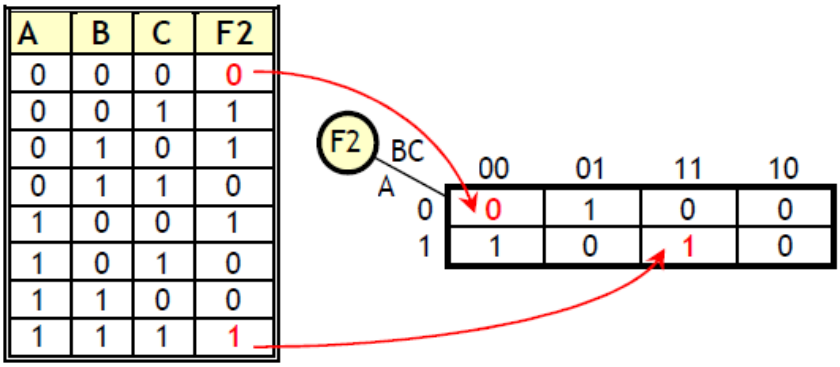
A	B	C	D	S
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

CD\AB	00	01	11	10
00				
01				
11				
10				

Simplification d'une équation par le diagramme de Karnaugh

La mise en œuvre de cette méthode se fait alors en 2 phases :

- ✓ La transcription de la fonction à simplifier dans le tableau de Karnaugh ;
- ✓ La recherche des groupements de cases qui donneront des expressions simplifiées.

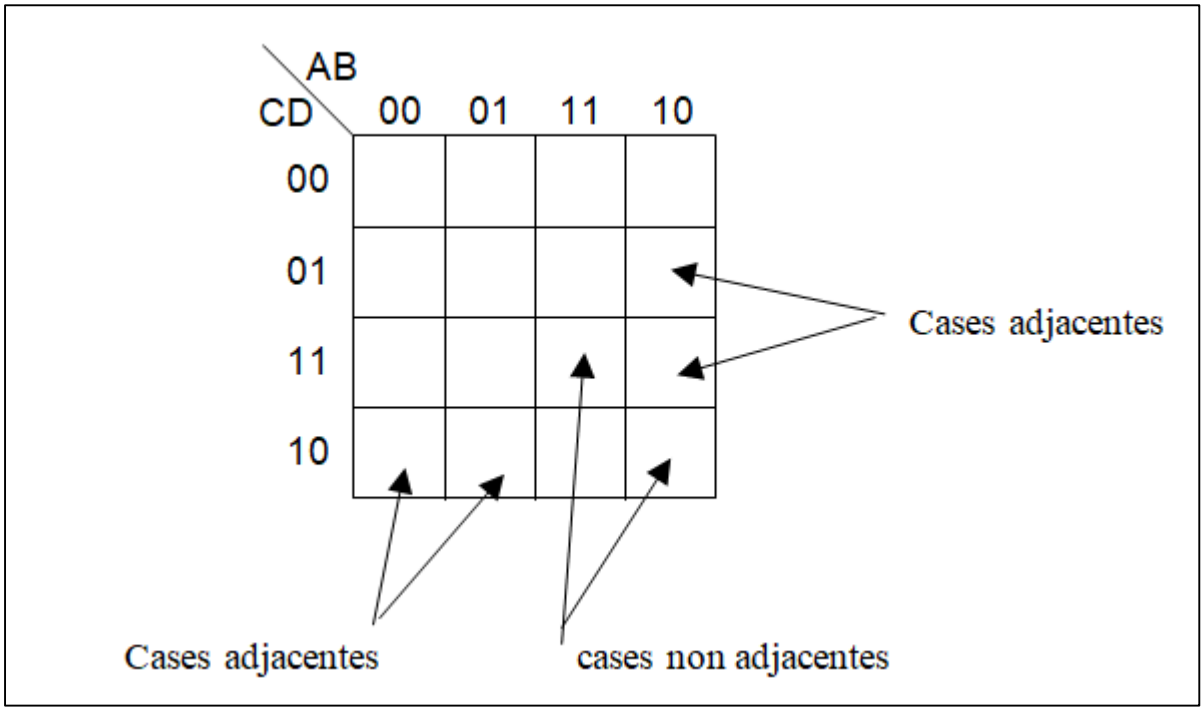


A	B	C	F2
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Le groupement des cases réunie les cases adjacentes, contenant des valeurs 1 à condition que le nombre de cases du groupement soit égal à une puissance de 2 (1, 2, 4, 8, 16 …). Le regroupement de 6 cases est impossible.

- **Cases adjacentes**

Deux cases sont adjacentes lorsqu'elles sont situées côte à côte, que ce soit à l'horizontale ou à la verticale. De plus, une seule variable doit changer d'état pour que deux cases soient considérées comme adjacentes.



- **Règles de regroupement**

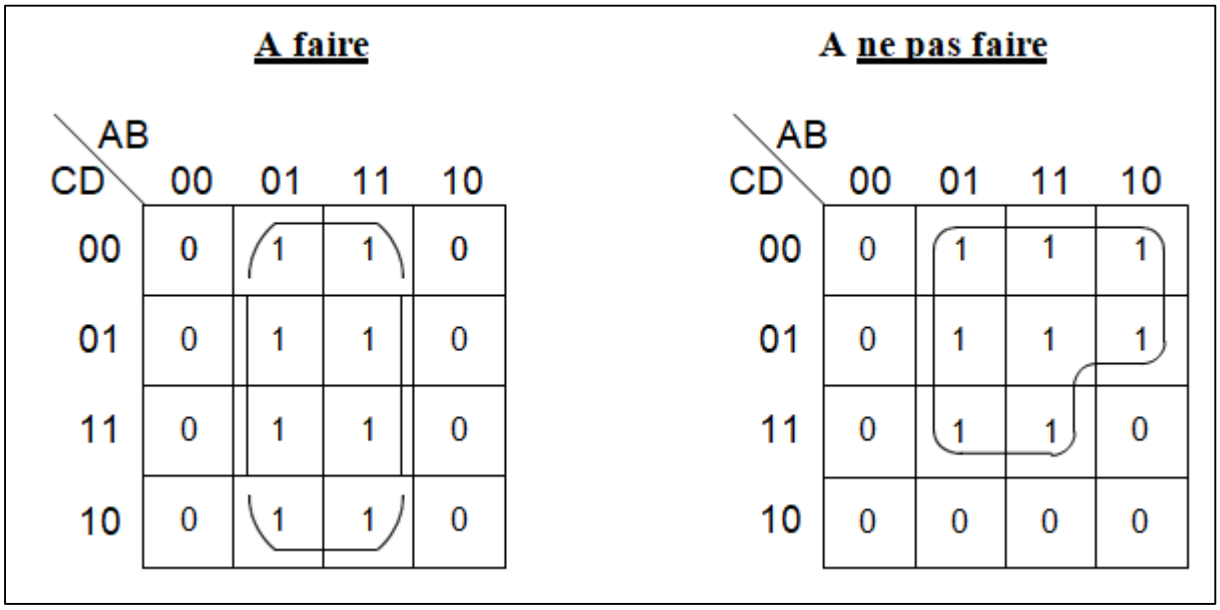
Le regroupement des cases adjacentes permet de réduire une équation logique le plus simplement possible. Pour ce faire, certaines règles doivent être respectées :

Règle 1 : Le regroupement des cases adjacentes doit se faire par puissance de deux : 2^0, 2^1, 2^2, 2^3, ….(1, 2, 4, 8 ….)

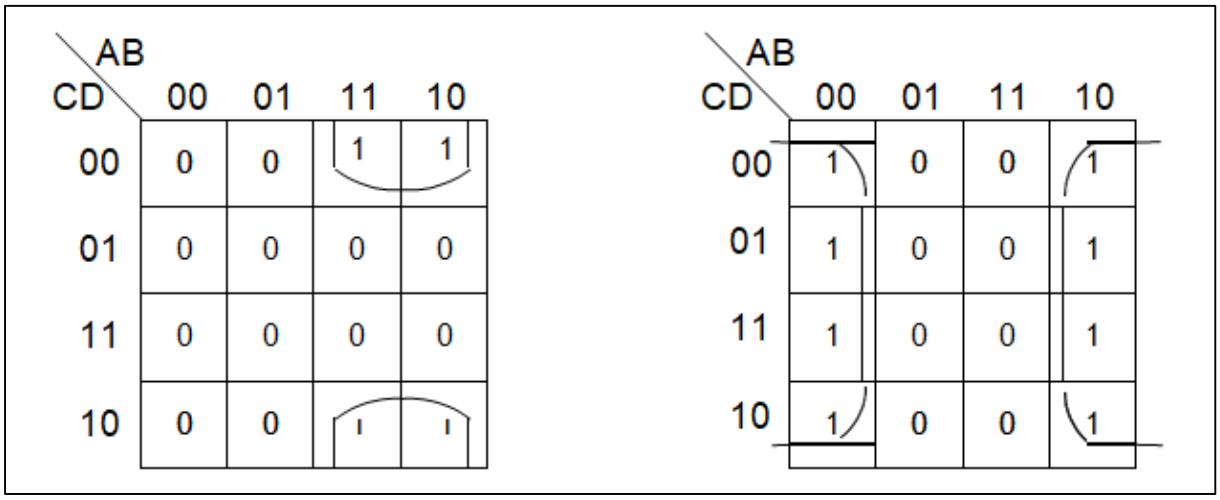
Règle 2 : Les cases appartenant au même groupement doivent avoir la même valeur binaire de la variable de sortie.

Règle 3 : La longue et la hauteur des groupements doivent être des puissances de deux.

Règle 4 : Les regroupements de quatre cases ou plus doivent être disposés symétriquement par rapport à l'un des axes du diagramme.



Règles 5 : Les cases des extrémités de gauche peuvent être regroupées avec celles de droite,avec celles des bords hauts ou encore avec celles du bas.



Règle 6 : Les quatre cases des 4 coins d'un diagramme de Karnaugh peuvent être regroupées

A faire

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

A ne pas faire

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

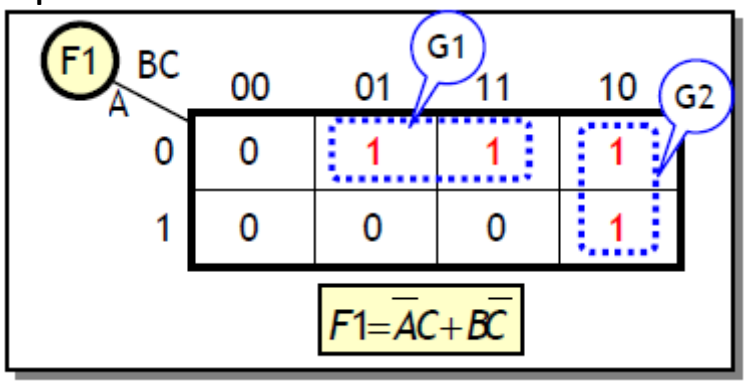
Écriture des équations à partir de regroupement

Somme de produits

Chaque **regroupement de 1** donne le produit logique des variables d'entrée qui n'ont pas changé d'état. L'ensemble de ces regroupements est une **somme logique**.

- Variable **A = 1**, on la représente par **A**
- Variable **A = 0**, on la représente par $\overline{A}$

- **Exemple 1**



Dans le groupement 1, B qui a changé d'état et A=0 et C=1➔ $\overline{A}$.C

Dans le groupement 2, A qui a changé et B =1 et C=0 ➔ $B.\overline{C}$

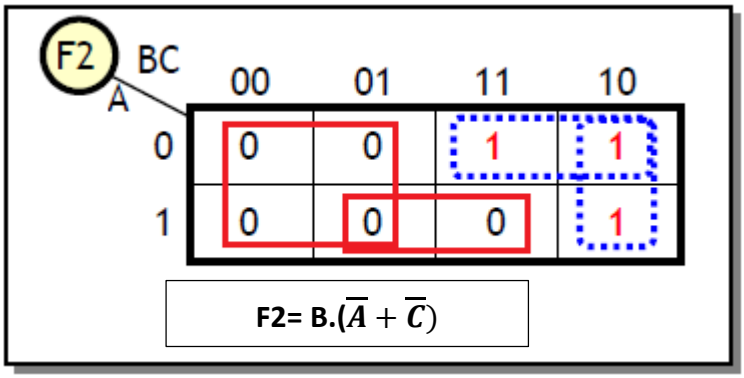
Ce qui donne **F1=** $\overline{A}$**.C +** $B.\overline{C}$

Produits de sommes

Chaque **regroupement de 0** donne la somme logique des variables d'entrée qui n'ont pas changé d'état. L'ensemble de ces regroupements est **un produit logique**.

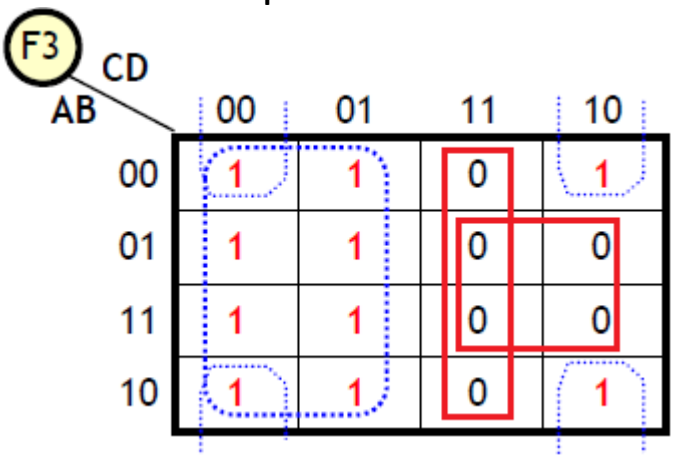
- Variable **A = 1**, on la représente par $\overline{A}$
- Variable **A = 0**, on la représente par **A**

- **Exemple 2**



Dans le groupement 1, A et C qui ont changés et B=0 ➔ B
Dans le groupement 2, B qui a changé et A=1 et C=1 ➔ $\overline{A}+\overline{C}$
Ce qui donne **F2= B.($\overline{A}+\overline{C}$)**

- **Exemple 3**



Regroupement des 1 :
- Dans le groupement 1, A, B et D qui ont changés d'état et C=0 ➔ $\overline{C}$
- Dans le groupement 2, A et C qui ont changés d'état et B=0 , D=0➔ $\overline{B}.\overline{D}$

Ce qui donne **F3= $\overline{C}$ + $\overline{B}.\overline{D}$**

Regroupement des 0 :
- Dans le groupement 1, A et B qui ont changés d'état et C=1 et D=1 ➔ $\overline{C}+\overline{D}$
- Dans le groupement 2, A et D qui ont changés d'état et B=1 , C=1➔ $\overline{B}+\overline{C}$

Ce qui donne **F3 = ($\overline{C}+\overline{D}$).($\overline{C}+\overline{B}$)**

Quelques remarques sur la simplification par tableau de karnaugh :

✓ On ne regroupe pas des cases qui ne sont pas symétriques, car cela ne donne pas de termes vérifiant la forme simplificatrice :

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

✓ Le nombre de variables supprimées dépend de la taille du groupement. Ainsi :
- Un groupement de 2 cases symétriques entraîne la suppression d'une variable ;
- Un groupement de 4 cases symétriques entraîne la suppression de 2 variables ;
- En général, un groupement de 2k cases entraîne la suppression de k variables.

7. Schémas logiques (Logigrammes):

Un schéma logique est la représentation graphique de l'équation d'une ou plusieurs variables de sortie grâce aux opérateurs de base vus précédemment.
On distingue 3 types de schémas logiques :

- Le 1er type comprend des opérateurs **NON**, **ET**, **OU**;
- Le 2éme type ne comprend que des opérateurs **NON ET** (NAND);
- Le 3éme type ne comprend que des opérateurs **NON OU** (NOR).

7-1 Différents types de schémas logiques

Schéma logique comprenant des opérateur NON, ET,OU

Pour traduire une équation en schéma logique avec ces opérateurs, il faut :

- ✓ Déterminer le **nombre d'opérateurs NON** ➔ égal au nombre des variables complimentées.
- ✓ Déterminer le **nombre d'opérateurs ET** ➔ égal au nombre de groupes de produits logiques et déduire le nombre d'entrées nécessaires sur chaque opérateur.
- ✓ Déterminer le **nombre d'opérateur OU** ➔ égal au nombre de groupes de sommes logiques et déduire le nombre d'entrées nécessaires sur chaque opérateur.
- ✓ Relier les différents opérateurs de base.

Exemple 1 :

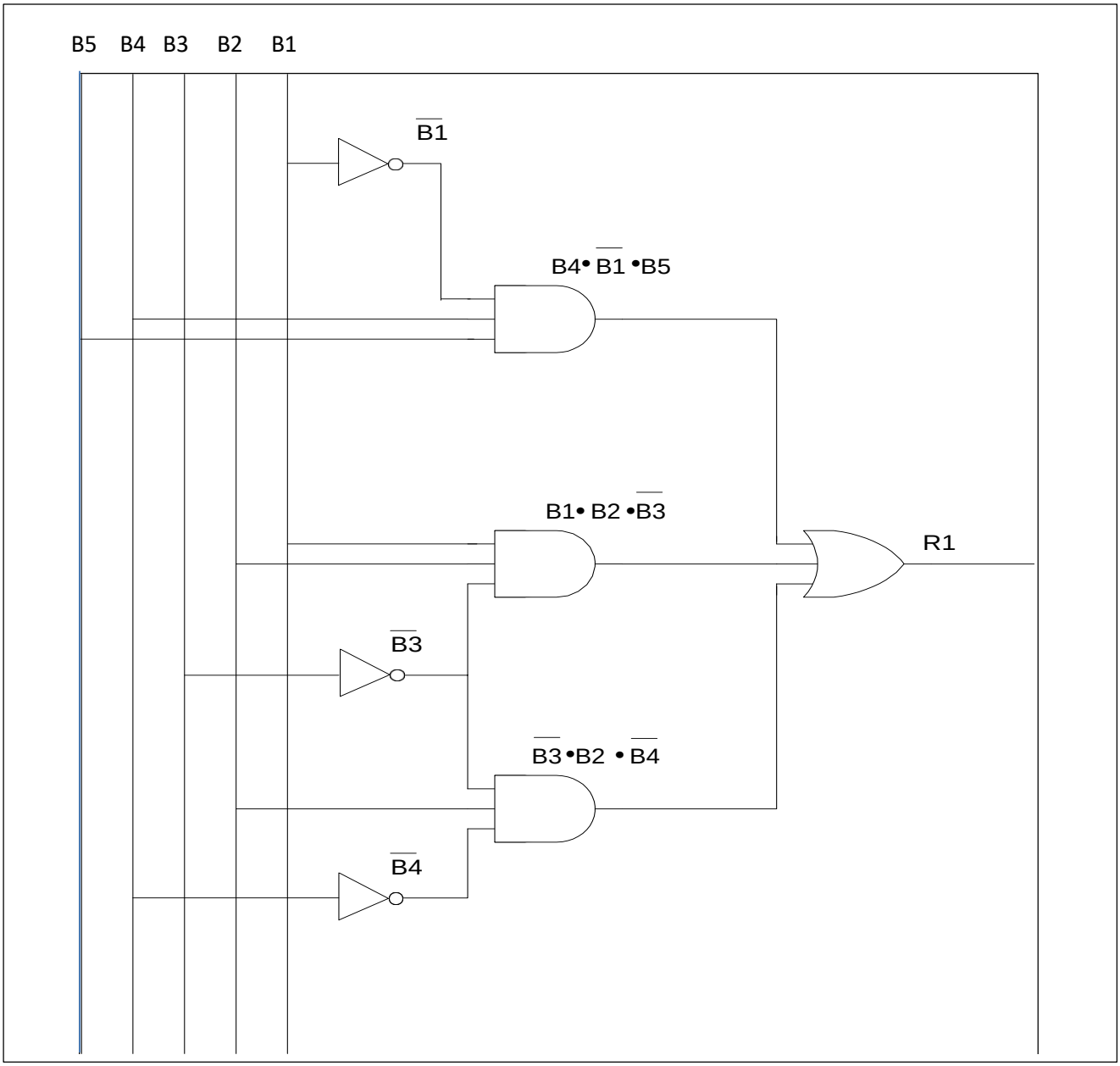
On donne la fonction logique suivante :

$$\boldsymbol{R1 = B1.B2.\overline{B3} + B4.\overline{B1}.B5 + \overline{B4}.B2.\overline{B3}}$$

D'après l'expression de cette fonction logique :

- ✓ Le **nombre d'opérateurs NON** = 4
    - L'équation comporte deux fois $\overline{B3}$; pour les obtenir, il suffit d'utiliser un seul opérateur NON, d'où **3 opérateurs NON** au lieu de 4.
- ✓ Le **nombre d'opérateurs ET** = 3 à 3 entrées
- ✓ Le **nombre d'opérateurs OU** = 1 à 3 entrées

Le schéma logique de la fonction R1 est donné alors par la figure suivante :



Exemple 2 :

On donne la fonction logique suivante :

$$\boldsymbol{R1 = B1.\overline{B2}.(B3 + B4) + B5.(B1 + B3 + B4)}$$

D'après l'expression de cette fonction logique :

- ✓ Le **nombre d'opérateurs NON** = 1
- ✓ Le **nombre d'opérateurs ET** = 2 (1 à 2 entrées et 1 à 3 entrées)
- ✓ Le **nombre d'opérateurs OU** = 3 (2 à 2 entrées et 1 à 3 entrées)

Le schéma logique de la fonction R1 est donné alors par la figure suivante :

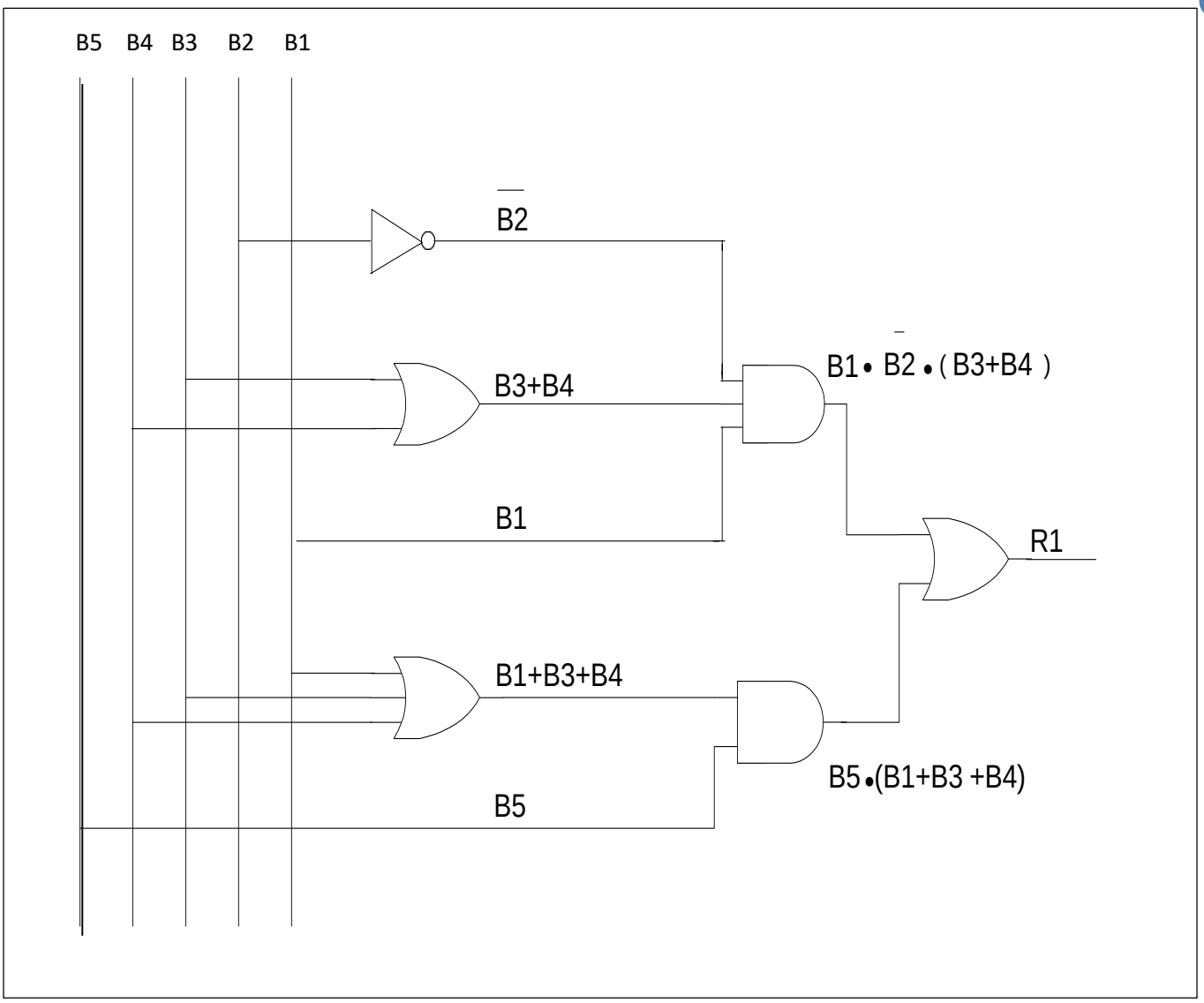


Schéma logique ne comprenant que des opérateurs NAND (NON ET)

Pour réaliser ce schéma, il faut les deux conditions suivantes :

- ✓ l'équation ne doit comporter que des **ET logiques** ➔ transformer l'équation en appliquant **le théorème de De Morgan**.
- ✓ l'équation doit être **entièrement recouverte par une barre** ➔utiliser les propriétés de la négation ($\boldsymbol{S = \overline{\overline{S}}}$)

Exemple 1 :

On donne la fonction logique suivante :

$$\boldsymbol{R1 = B1.B2.\overline{B3} + \overline{B1}.B4.B5 + B2.\overline{B3}.\overline{B4}}$$

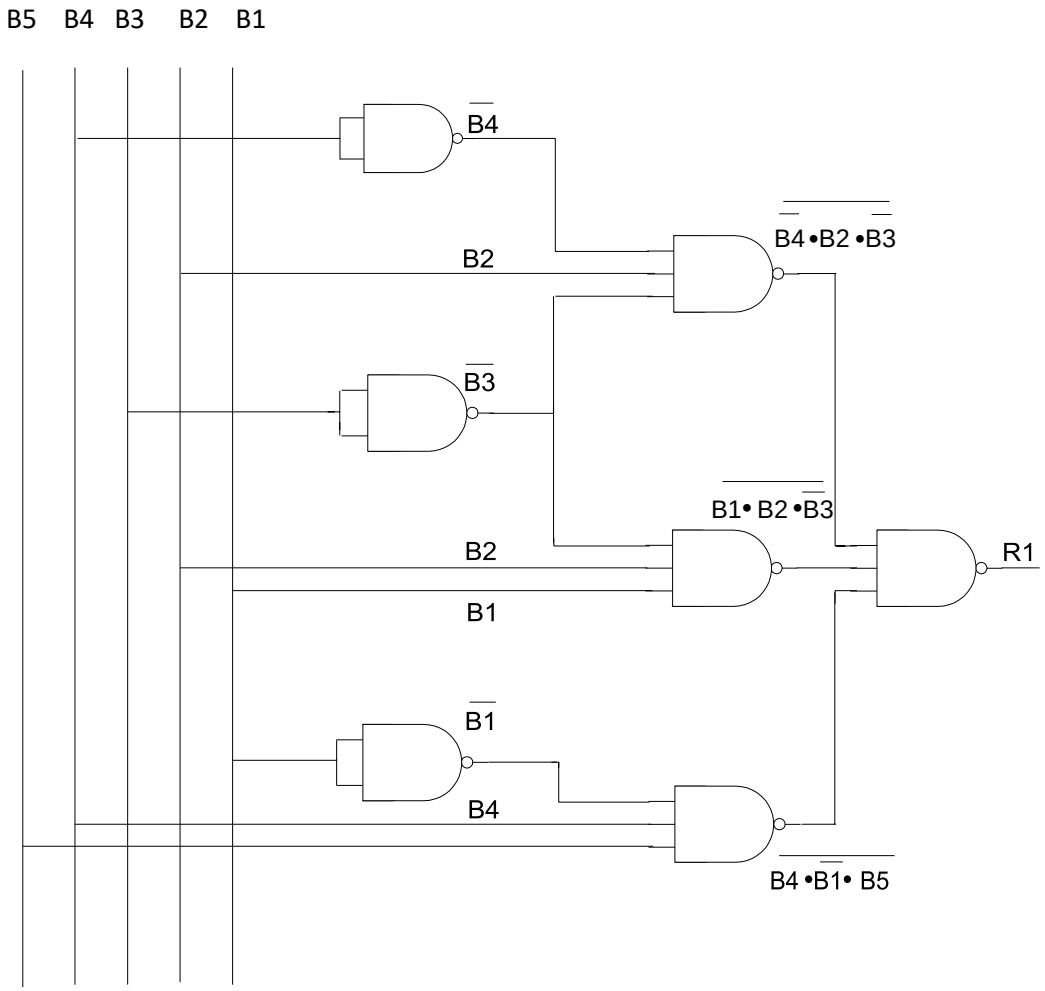
Transformation des OU logiques en ET logique en appliquant le théorème de De Morgan :

$$\boldsymbol{R1 = \overline{\overline{R1}} = \overline{\overline{B1.B2.\overline{B3} + \overline{B1}.B4.B5 + B2.\overline{B3}.\overline{B4}}}}$$

$$\boldsymbol{= \overline{\overline{B1.B2.\overline{B3}}.\overline{\overline{B1}.B4.B5}.\overline{B2.\overline{B3}.\overline{B4}}}}$$

D'après l'expression de cette fonction transformée en ET logiques, on trouve 8 barres ➔
Il faut 8 opérateurs NAND (NON ET) et puisque le terme $\boldsymbol{\overline{B3}}$ se répète 2 fois ➔
Il faut **7 opérateurs NAND (NON ET)**

Le schéma logique de la fonction R1 est donné alors par la figure suivante :



Exemple 2 :

On donne la fonction logique suivante :

$$\boldsymbol{R1 = B1.\overline{B2}.(B3 + B4) + B5.(B1 + B3 + B4)}$$

Transformation des OU logiques en ET logique en appliquant le théorème de De Morgan :

$$\boldsymbol{R1 = B1.\overline{B2}.\left(\overline{\overline{B3 + B4}}\right) + B5.(\overline{\overline{B1 + B3 + B4}})}$$

$$\boldsymbol{R1 = B1.\overline{B2}.\left(\overline{\overline{B3}.\overline{B4}}\right) + B5.(\overline{\overline{B1}.\overline{B3}.\overline{B4}})}$$

$$\boldsymbol{\overline{\overline{R1}} = R1 = B1.\overline{B2}.\left(\overline{\overline{B3}.\overline{B4}}\right) + B5.(\overline{\overline{B1}.\overline{B3}.\overline{B4}})}$$

$$\boldsymbol{= \overline{\overline{B1.\overline{B2}.\left(\overline{\overline{B3}.\overline{B4}}\right)}.\overline{B5.(\overline{\overline{B1}.\overline{B3}.\overline{B4}})}}}$$

D'après l'expression de cette fonction transformée en ET logiques, on trouve 11 barres ➔
Il faut 11 opérateurs NAND (NON ET) et puisque le terme $\boldsymbol{\overline{B3}}$ et $\boldsymbol{\overline{B4}}$ se répètent 2 fois chacun

➔ Il faut **9 opérateurs NAND (NON ET)**

Le schéma logique de la fonction R1 est donné alors par la figure suivante :

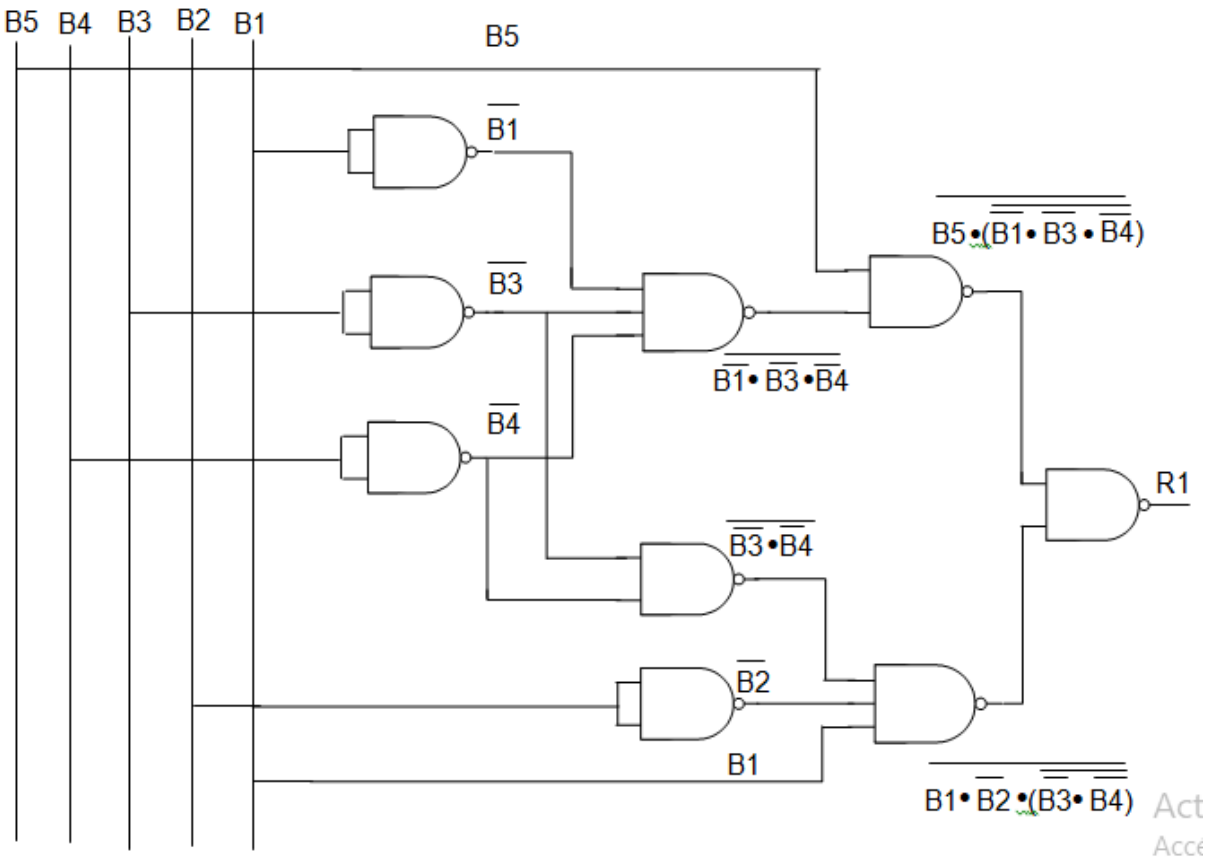


Schéma logique ne comprenant que des opérateurs NOR (NON OU)

Pour réaliser ce schéma, il faut les deux conditions suivantes :

- ✓ l'équation ne doit comporter que des **OU logiques** ➔ transformer l'équation en appliquant **le théorème de De Morgan**.
- ✓ l'équation doit être **entièrement recouverte par une barre** ➔utiliser les propriétés de la négation ($\boldsymbol{S = \overline{\overline{S}}}$)

Exemple 1 :

On donne la fonction logique suivante :

$$\boldsymbol{R1 = B1.B2.\overline{B3} + \overline{B1}.B4.B5 + B2.\overline{B3}.\overline{B5}}$$

Transformation des ET logiques en OU logique en appliquant le théorème de De Morgan :

$$\boldsymbol{R1 = \overline{\overline{B1.B2.\overline{B3}} + \overline{\overline{B1}.B4.B5} + \overline{B2.\overline{B3}.\overline{B5}}}}$$

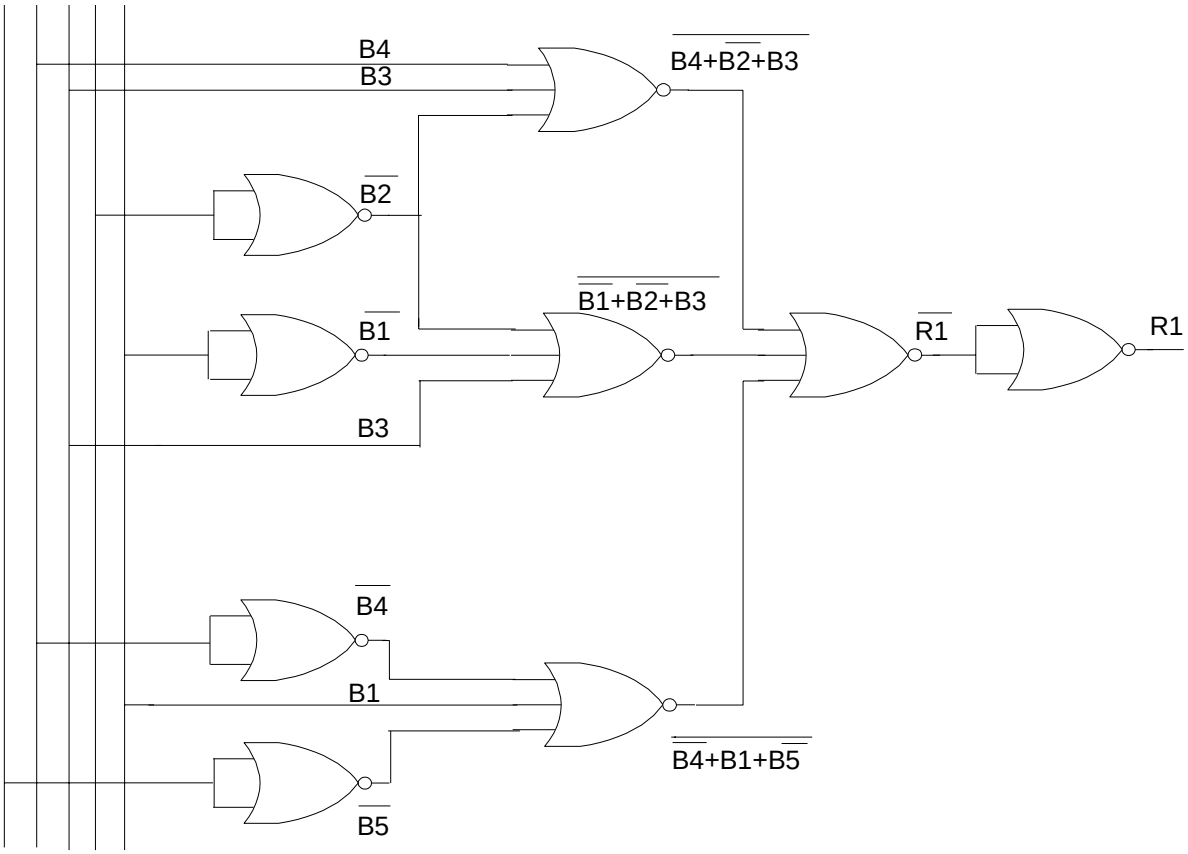
$$\boldsymbol{R1 = \overline{\overline{B1} + \overline{B2}+.B3} + \overline{B1 + \overline{B4} + \overline{B5}} + \overline{\overline{B2} + B3 + B5}}$$

$$\boldsymbol{R1 = \overline{\overline{R1}} = \overline{\overline{\overline{B1} + \overline{B2}+.B3} + \overline{B1 + \overline{B4} + \overline{B5}} + \overline{\overline{B2} + B3 + B5}}}$$

D'après l'expression de cette fonction transformée en OU logiques, on trouve 10 barres ➔ Il faut 10 opérateurs NOR (NON OU) et puisque le terme $\boldsymbol{\overline{B2}}$ se répète 2 fois chacun ➔ Il faut **9 opérateurs NOR (NON OU)**

Le schéma logique de la fonction R1 est donné alors par la figure suivante :

B5 B4 B3 B2B1



Exemple 2 :

On donne la fonction logique suivante :

$$\boldsymbol{R1 = B1.\overline{B2}.(B3 + B4) + B5.(B1 + B3 + B4)}$$

Transformation des ET logiques en OU logique en appliquant le théorème de De Morgan :

$$\boldsymbol{R1 = \overline{\overline{B1.\overline{B2}.(B3 + B4)}} + \overline{\overline{B5.(B1 + B3 + B4)}}}$$

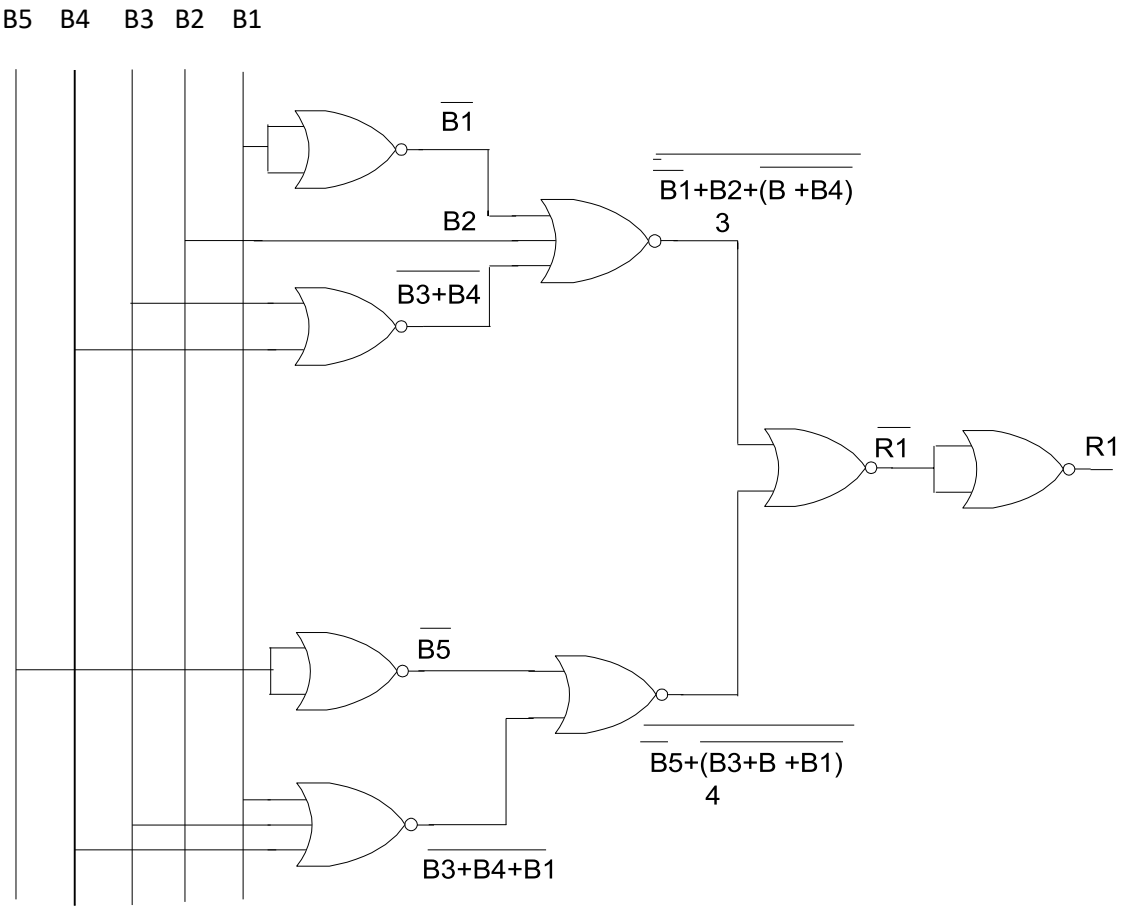
$$\boldsymbol{= \overline{\overline{B1.\overline{B2}} + \overline{(B3 + B4)}} + \overline{\overline{B5} + \overline{(B1 + B3 + B4)}}}$$

$$\boldsymbol{= \overline{\overline{B1} + B2 + \overline{(B3 + B4)}} + \overline{\overline{B5} + \overline{(B1 + B3 + B4)}}}$$

$$\boldsymbol{R1 = \overline{\overline{R1}} = \overline{\overline{\overline{B1} + B2 + \overline{(B3 + B4)}} + \overline{\overline{B5} + \overline{(B1 + B3 + B4)}}}}$$

D'après l'expression de cette fonction transformée en OU logiques, on trouve 8 barres ➔ Il faut 8 opérateurs NOR (NON OU) ➔ Il faut **8 opérateurs NOR (NON OU)**

Le schéma logique de la fonction R1 est donné alors par la figure suivante :



8. Exemples de technologies de réalisation des fonctions logiques

Le passage à la réalisation matérielle d'un système logique à l'aide d'opérateurs intégrés du commerce implique le respect des contraintes technologiques fonctionnelles et d'implantation.

8-1 Les familles logiques et code d'identification

Les circuits intégrés sont classé selon la nature des éléments utilisés.

Circuits utilisant des transistors bipolaires

- ✓ **RTL** (Resistor Transistor Logic): Logique à résistance d'entrée et transistor de sortie,
- ✓ **DTL** (Diode Transistor Logic): Logique à diode d'entrée et transistor de sortie
- ✓ **TTL** (Transistor Transistor Logic): Logique à transistor d'entrée et transistor de sortie (série N : Normale, H (high speed), L (Low power), LS (Low power schottky), S (schottky), ALS (Advanced LS))

Circuits utilisant des transistors à effet de champ à grille isolée

- ✓ **MOS** (Metal Oxide Semiconductor): Circuits composes initialement par des transistors canal P (PMOS) puis transistor à canal N (NMOS).

- ✓ **CMOS** (Complementary MOS): Circuits composes par l'association des 2 types de transistors MOS.

8-2 Les formes de boitier

Les boitiers plats (flat package)

Le circuit avec ses connexions est disposé entre deux plaques planes en céramique.

Le boitier hermétique et de faible épaisseur, soudé sur un circuit imprimé du côté des composants comme le montre la figure 2-1.

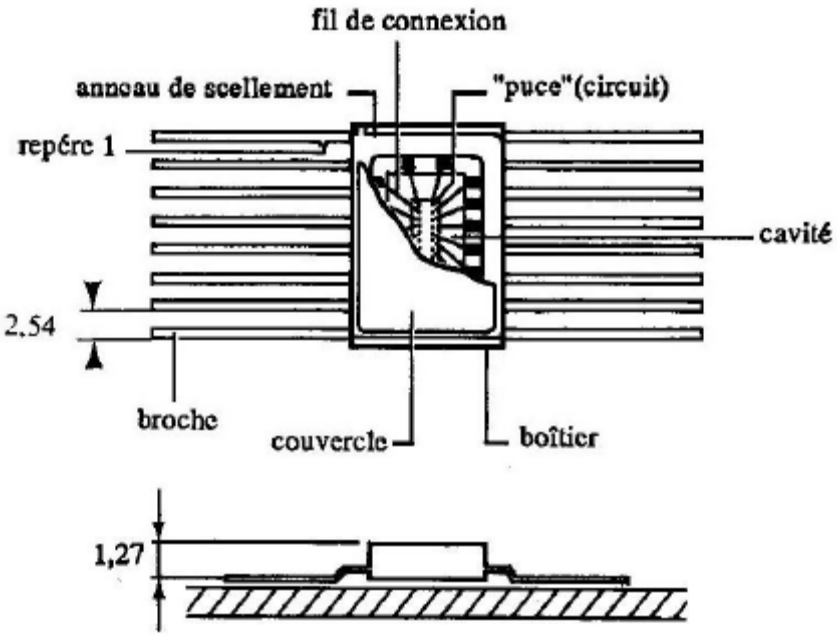


Figure 2-1 : Exemple de boitier plat

Les boitiers DIL (Dual In Line)

Ce sont les boitiers les plus fréquemment rencontrés qui sont composés de 6 à 80 connexions réparties en deux lignes, comme le montre la figure 2-2.

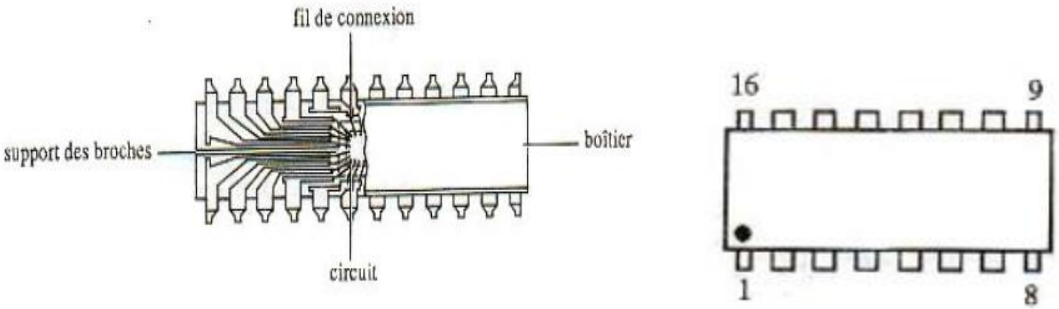


Figure 2-2 : Exemple de boitier DIL

Les connexions sont numérotées dans le sens trigonométrique à partir de l'encoche du boitier ou du repère (point).

Lorsque les connexions sont placés d'un seul côté, le boitier est appelé SIL (Single In Line).

Les boitiers SO (Small Outline)

Ce sont des boitiers semblables au DIL dont l'espace entre deux sorties est divisé par2 et dont les connexions ne sont plus droites mais coudées à fin de permettre une soudure du coté composant (soudure sans trou), comme le montre la figure 4-3.

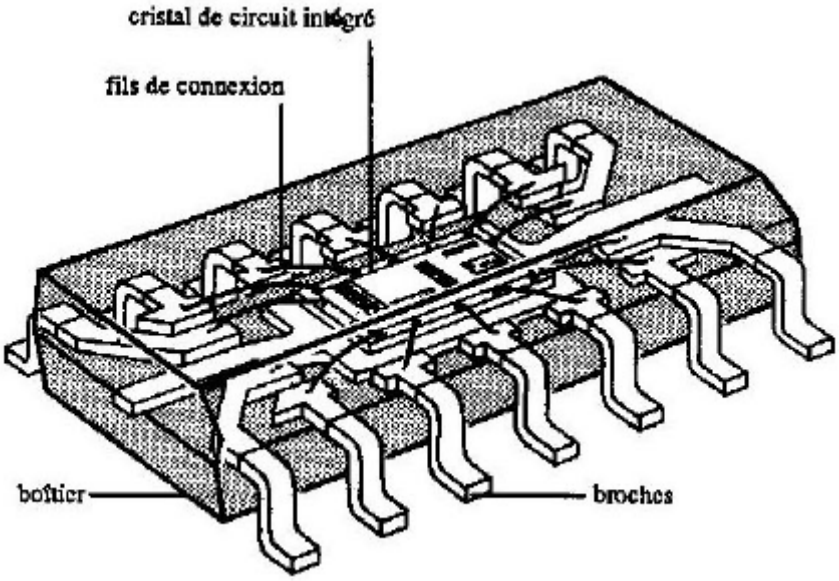


Figure 2-3 : Exemple de boitier SO

Le repérage des bornes est identique au DIL.

Les boitiers Chip carrier

Ce sont des boitiers enfichables dans des supports spéciaux sans broche extérieure.

La figure 4-4 donne un exemple de boitier Chip Carrier

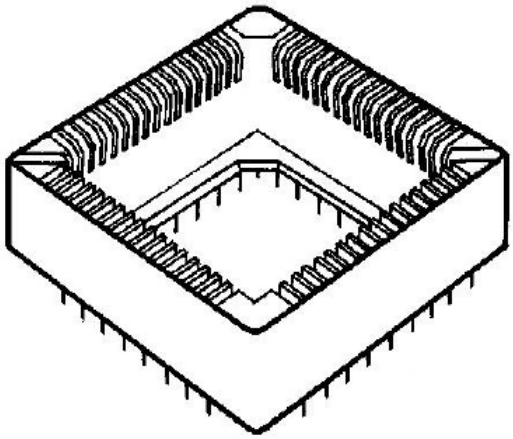


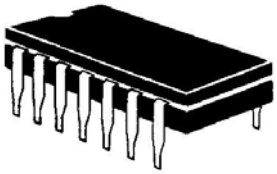
Figure 2-4 : Exemple de boitier Chip carrier

Les composants électroniques peuvent être de deux familles : TTL ou CMOS.

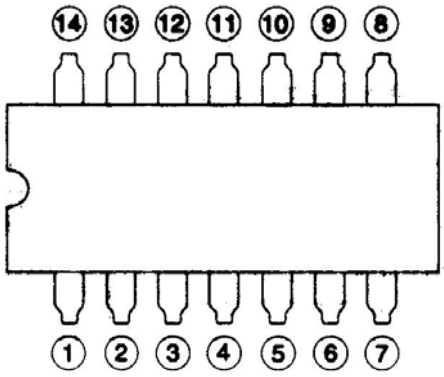
Famille TTL (transistor transistoc logic)

Elle fait principalement usage de combinaisons de transistors bipolaires pour la fabricationdes circuits intégrés CI.

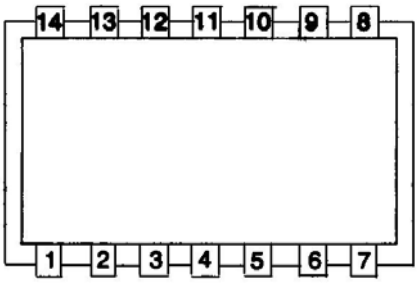
Ces C.I. sont constitués d'un boîtier qui contient la puce, laquelle est reliée à l'extérieur par un certain nombre de pattes (ou broches). Ce nombre varie généralement entre 14 et 28. La tension d'alimentation est +5V.



Vue en trois dimensions

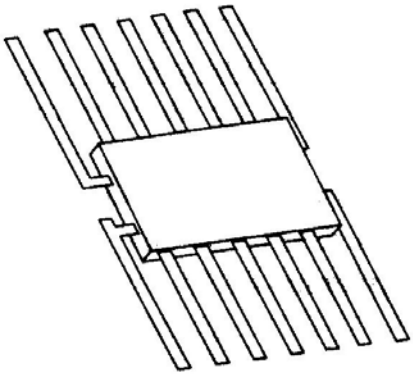


Vue de dessus avec identification des pattes

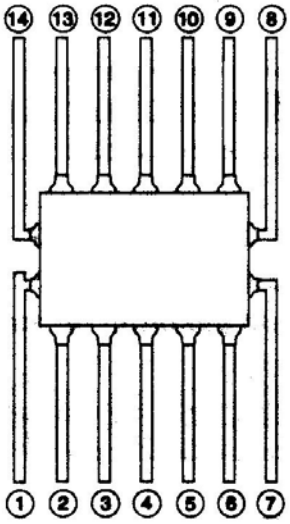


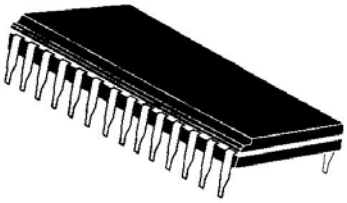
Représentation généralement employée pour l'identification

On peut remarquer qu'il y a encoche sur des côtés du boîtier. Elle permet de localiser rapidement le numéro de chaque patte et d'obtenir un branchement simple et rapide.
Il existe plusieurs formes de boîtiers pour les circuits intégrés (voir figure suivante).

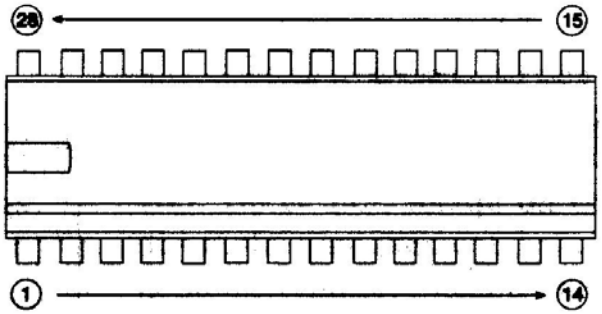


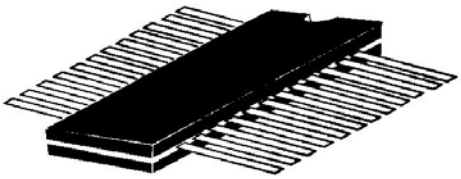
Type « T »



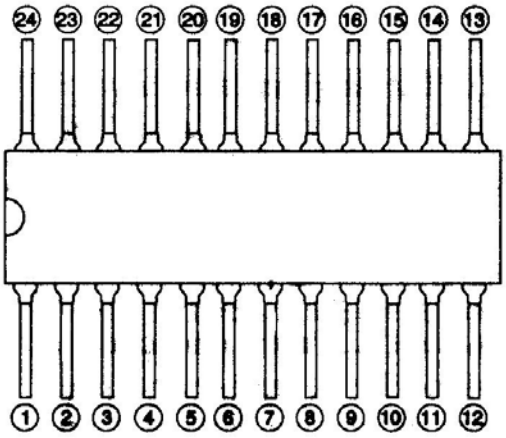


Type « N »





Type « W »



Suivant la gamme de température d'utilisation, on distingue deux séries des C.I TTL :

- La **série 5400** : gamme de température d'utilisation militaire indiquée par 5 (-55°, +125°C);
- La **série 7400** : gamme de température d'utilisation générale indiquée par 7 (0°,+70°C).

La famille TTL se subdivise en cinq sous-groupes, dont chacun possède ses propres caractéristiques de fonctionnement.

Rien = standard
H=rapide (Hi - Speed)
LS=Schottky faible (Low Speed)
L = Faible consommation
S = Schottky (Ultra Hi - Speed) consommation (Lo Pwr Schottky)

Exemples :

a) SN 54L121N

SN	:	préfix standard ;
5	:	gamme militaire ;
4	:	circuit logique ;
L	:	sous-groupe faible consommation ;
121	:	fonction du circuit intégré ;
N	:	boîtier de plastique enfichable 14 ou 16 broches.

b) SN 74LS10J

SN	:	préfix standard ;
7	:	gamme générale ;
4	:	circuit logique ;
LS	:	sous-groupe, Schottky faible consommation ;
10	:	fonction du circuit intégré (NON ET) ;
J	:	type de boîtier : boîtier céramique enfichable DIL 14 ou 16 broches.

c) SN 74LS00

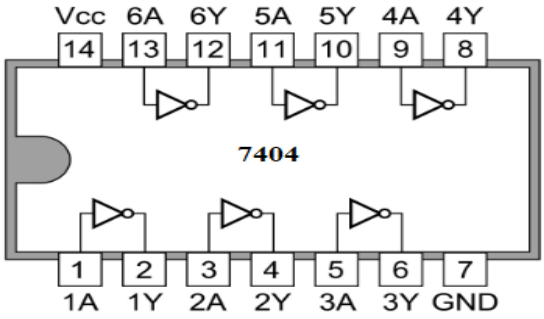
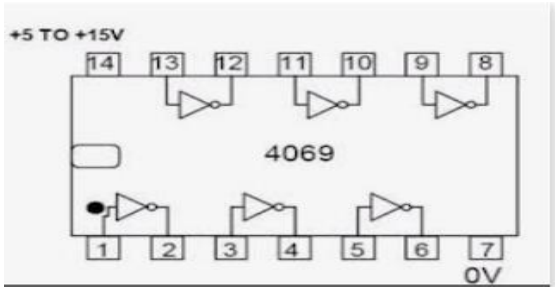
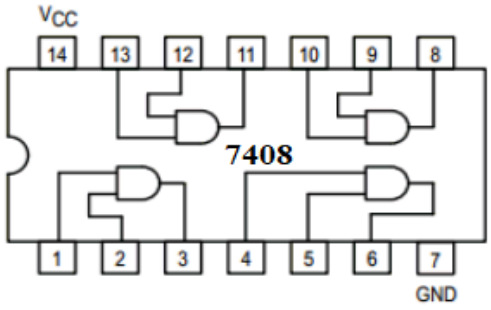
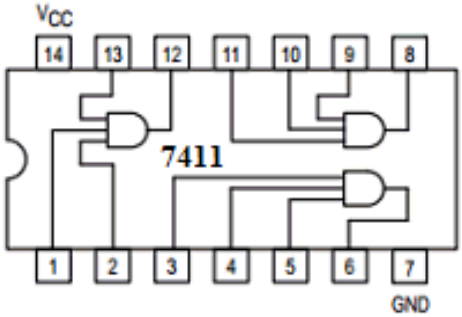
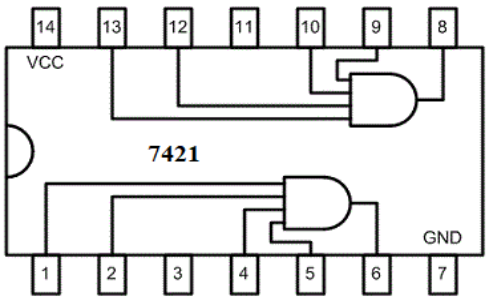
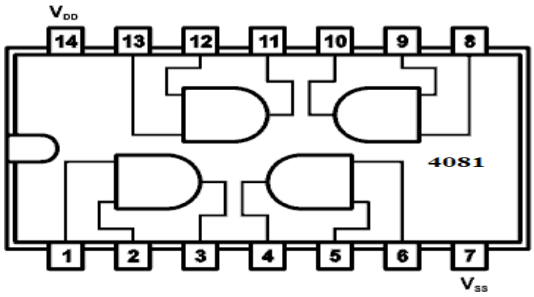
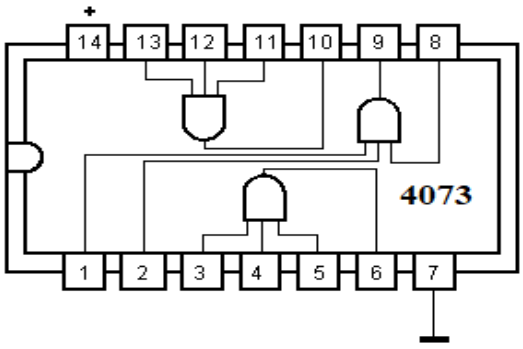
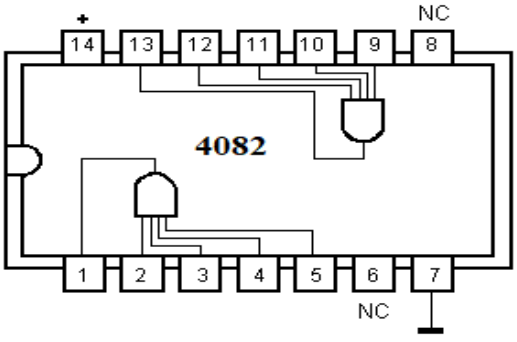
SN	:	préfix standard ;
7	:	gamme générale ;
4	:	circuit logique ;
LS	:	sous-groupe, Schottky faible consommation ;
00	:	fonction du circuit intégré (NON ET) ;

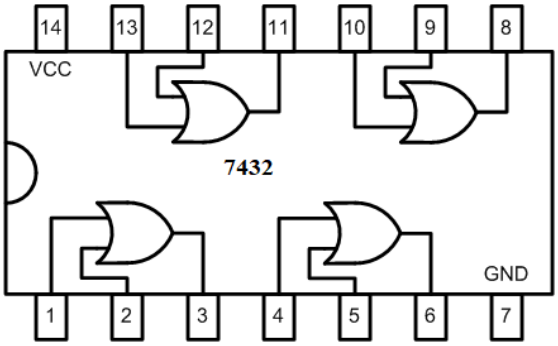
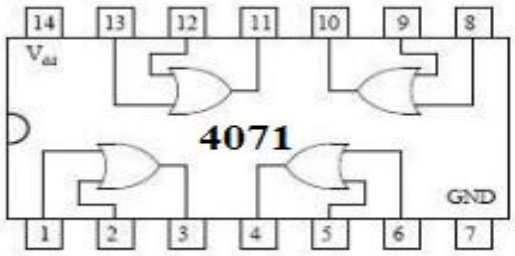
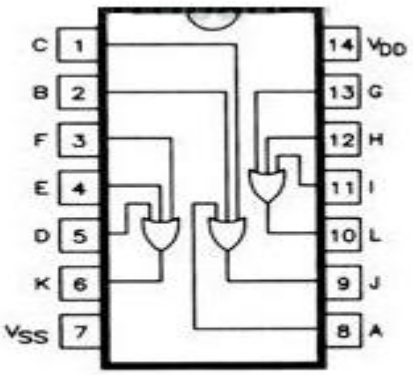
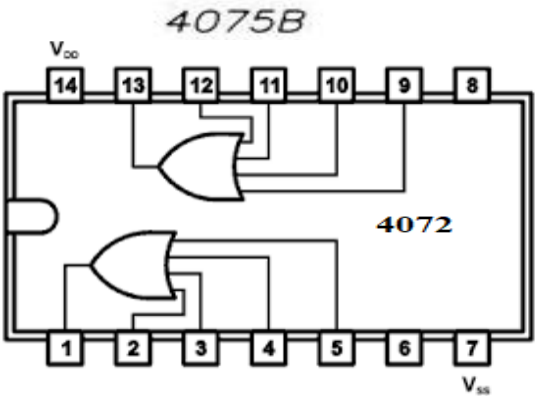
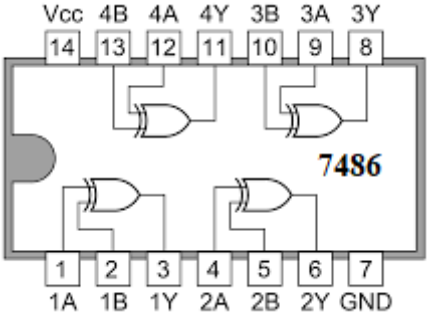
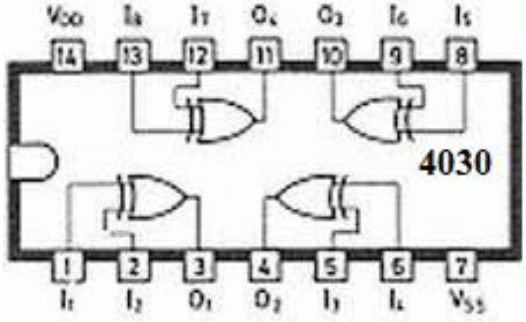
Famille CMOS (Comptementary Métal Oxyde Semiconductor)

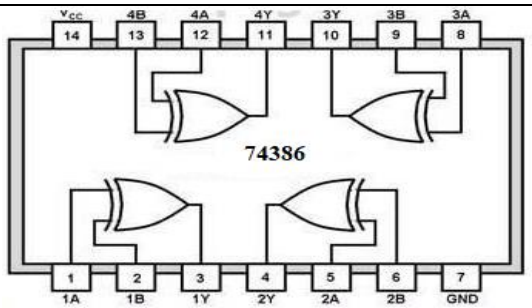
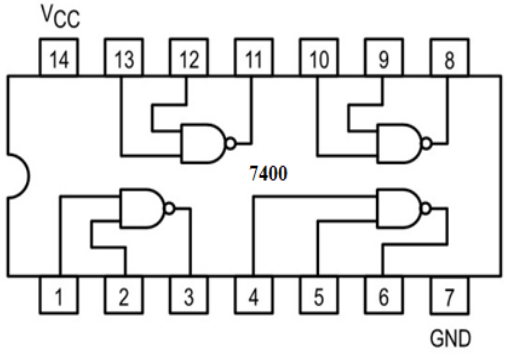
Elle dérive principalement des transistors à effet de champ. Cette famille se divise en deux sous-groupes :

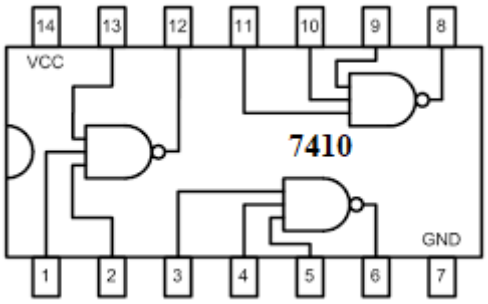
- le **type A**, qui peut fonctionner à des tensions variant de + 3V à +12 V (+15V maximum);
- le **type B**, qui peut fonctionner à des tensions variant de +3V à + 18V (+20V maximum).

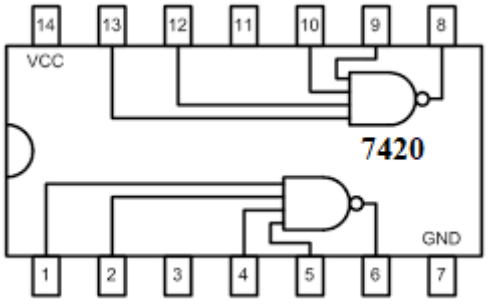
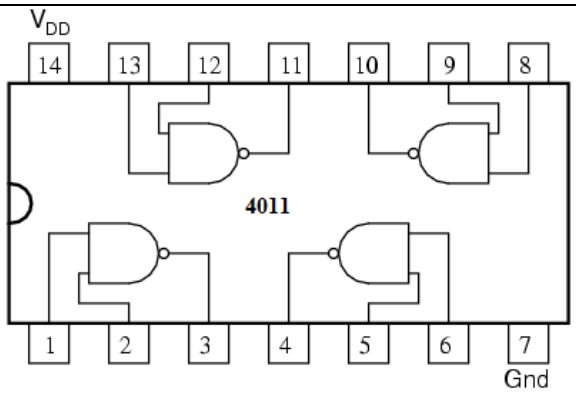
8-3 Brochage des principaux circuits

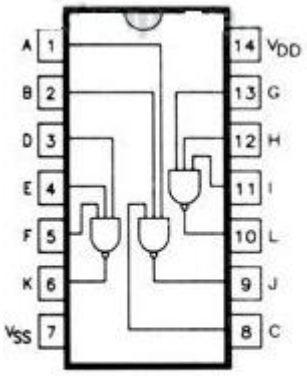
Fonction	Circuit intégré TTL	Circuit intégré CMOS
Fonction NON		
Fonction ET	  	  

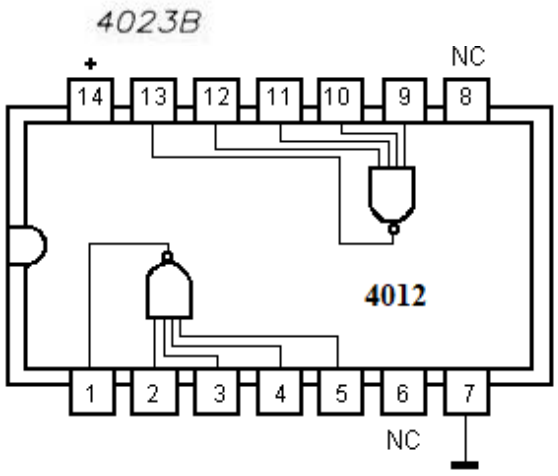
Fonction OU		  
Fonction XOR		

<table>
<tr>
<td></td>
<td> 
</td>
<td></td>
</tr>
<tr>
<td>Fonction NAND</td>
<td> 

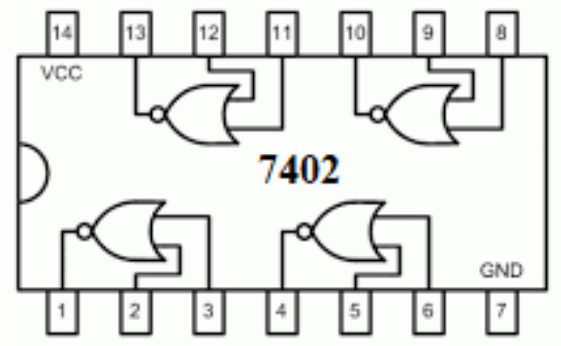


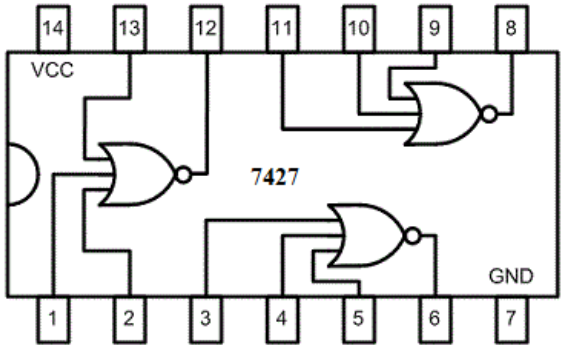

</td>
<td> 

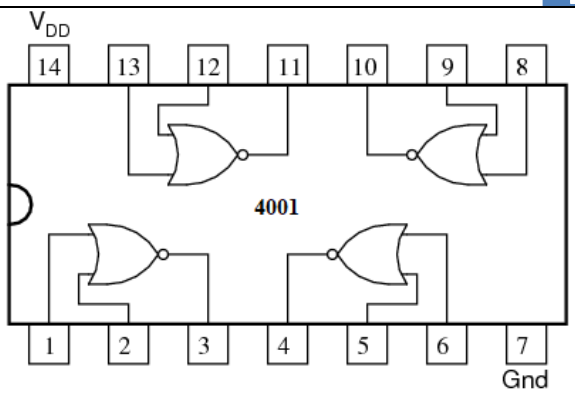


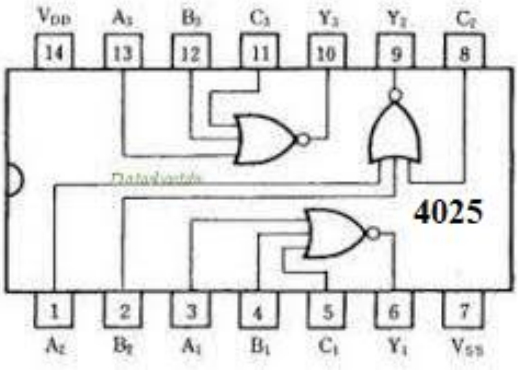

</td>
</tr>
</table>

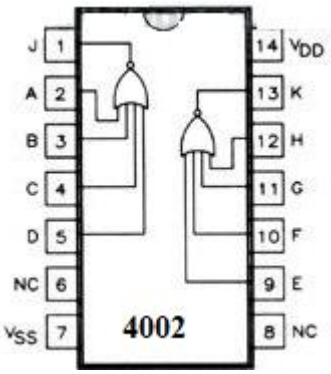
Fonction NOR









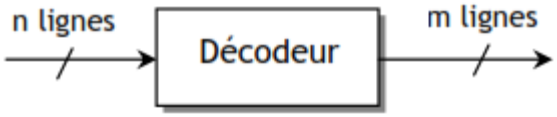


9- Les fonctions combinatoires avancées

Dans les systèmes numériques, on utilise souvent des fonctions qui ont justifié leurs réalisations en circuits intégrés. On note en particulier les décodeurs, les multiplexeurs, les démultiplexeurs et les circuits arithmétiques. Bien qu'ils soient plus ou moins remplacés actuellement par les systèmes programmables (circuits logiques programmables et microprocesseur), ils sont encore utilisés.

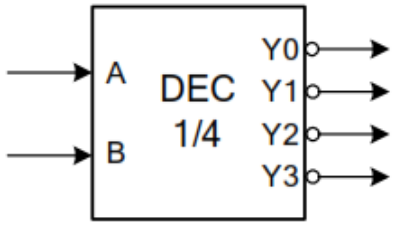
9-1 Les décodeurs

La fonction de décodage consiste à faire correspondre à un code présent en entrée sur **n lignes**, un autre code en sortie sur **m lignes** avec en général m ≠ n :



Décodeur 1 parmi n:

Ce type de décodeur permet de faire correspondre à un code présent en entrée sur n lignes une sortie et une seule active parmi les N = 2^n sorties possibles. On le désigne aussi par décodeur m lignes vers n lignes. Pour comprendre le principe d'un tel décodeur, étudions le décodeur 1 parmi 4 ou 2 vers 4, donné à la figure suivante ; le niveau active des sorties est le 0, car c'est souvent le cas :

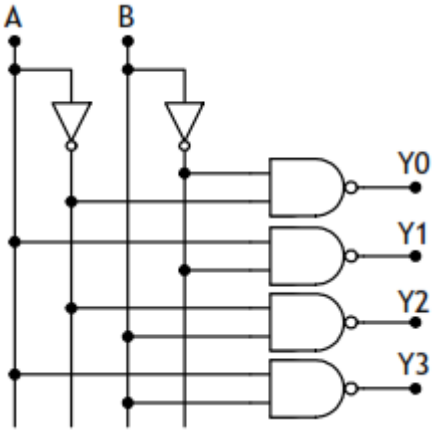


ENTREES		SORTIES			
B	A	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

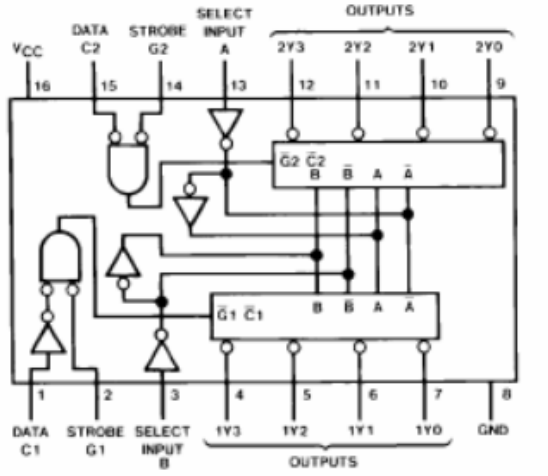
On détermine les équations de sorties à l'aide de la table de Karnaugh :

$\overline{Y0} = \bar{B}.\bar{A}$, $\overline{Y1} = \bar{B}.A$, $\overline{Y2} = B.\overline{A}$, $\overline{Y3} = A.B$

Le schéma d'implémentation du décodeur sera alors celui de la figure ci -dessous :



Les circuits intégrés réalisant cette fonction contiennent des entrées de validation comme G ou E permettant de sélectionner le circuit. On peut citer comme exemple le double décodeur 74LS156 dont le brochage et la table de fonction sont donnés à la figure suivante :

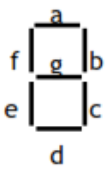


2-Line-to-4-Line Decoder or
1-Line-to-4-Line Demultiplexer

Inputs				Outputs			
Select		Strobe	Data				
B	A	G1	C1	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

Décodeur BCD – 7 segments :

Ce type de décodeur permet de convertir le code BCD 4bits à l'entrée pour obtenir à la sortie un code 7 segments permettant de commander un afficheur 7 segments permettant l'écriture de tous les chiffres et aussi d'autres symboles comme le montre la figure suivante :



Identification des segments

Désignations numériques et résultat de l'affichage

Pour mettre en équation ce type de décodeur, il faut dresser la table de vérité suivante :

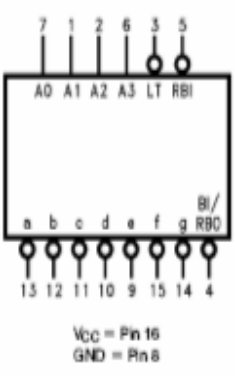
Nombre BCD à décoder	ENTREES				SORTIES						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

La table de karnaugh de chaque segment permet alors d'obtenir les équations de ce décodeur. Les 0 étant les moins nombreux, l'écriture des équations de commande d'extinction des segments sera plus facile :

$$\bar{a} = A.\bar{B}.\bar{C}.\bar{D} + \bar{A}.C \qquad \bar{b} = A.\bar{B}.C + \bar{A}.B.C \qquad \bar{c} = \bar{A}.B.\bar{C} \qquad \bar{d} = \bar{A}.\bar{B}.C + A.B.C + A.\bar{B}.\bar{C}$$

$$\bar{e} = A + \bar{B}.C \qquad \bar{f} = A.\bar{C}.\bar{D} + A.B + B.\bar{C} \qquad \bar{g} = \bar{B}.\bar{C}.\bar{D} + A.B.C$$

Comme exemple de décodeur, on peut citer le circuit intégré 74LS47 dont le schéma de brochage et la table de vérité sont données à la figure suivante :



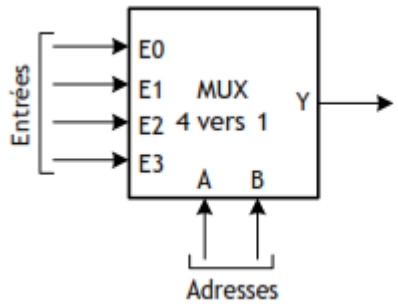
Truth Table

Decimal or Function	Inputs							Outputs						
	$\overline{LT}$	$\overline{RBI}$	A3	A2	A1	A0	$\overline{BI/RBO}$	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H
$\overline{BI}$	X	X	X	X	X	X	L	H	H	H	H	H	H	H
$\overline{RBI}$	H	L	L	L	L	L	L	H	H	H	H	H	H	H
$\overline{LT}$	L	X	X	X	X	X	H	L	L	L	L	L	L	L

9-2 Le Multiplexeur :

Un multiplexeur permet de sélectionner une entrée parmi 2n pour transmettre l'information portée par cette ligne à un seul canal de sortie. La sélection de l'entrée se fait alors à l'aide de n lignes d'adressage.

Pour comprendre le principe, considérons un multiplexeur à quatre entrées, donc deux lignes d'adressage et une ligne de sortie :

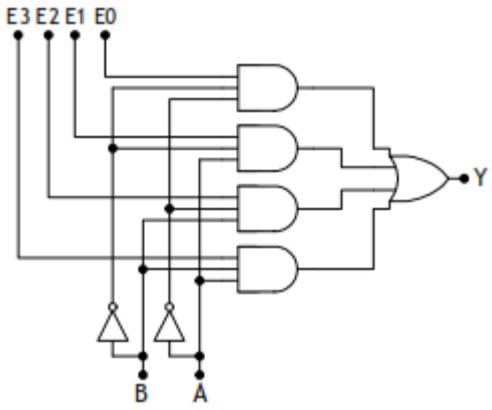


ADRESSES		SORTIE
B	A	Y
0	0	E0
0	1	E1
1	0	E2
1	1	E3

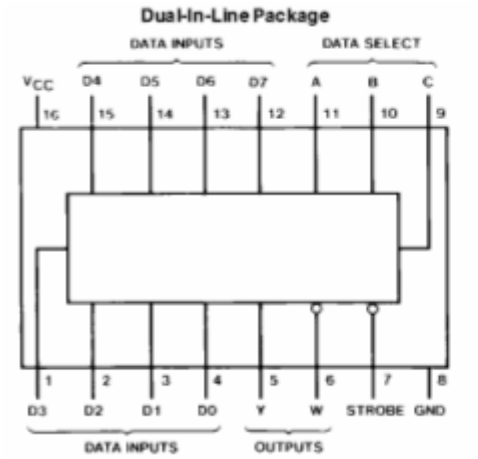
De la table de vérité, on déduit l'expression logique de la sortie Y :

$$Y = \bar{A}.\bar{B}.E0 + A.\bar{B}.E1 + \bar{A}.B.E2 + A.B.E3$$

Le schéma d'implantation du multiplexeur 4 vers 1 sera celui de la figure ci-dessous.



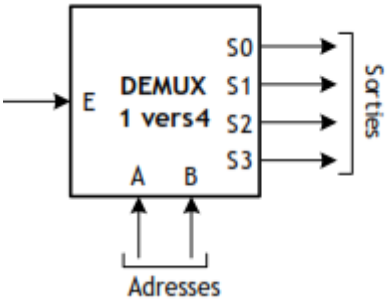
Les circuits intégrés réalisant cette fonction contiennent des entrées de validation (Strobe - Enable) permettant de sélectionner le circuit comme le 74LS151 qui est multiplexeur 8 vers 1:



Inputs				Outputs	
Select			Strobe S	Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

9-3 Le démultiplexeur :

Le démultiplexeur effectue l'opération inverse d'un multiplexeur à savoir il permet de distribuer l'information présente à l'entrée vers l'une des 2n sorties. La sélection de la sortie se fait à l'aide de n lignes d'adressage. Pour comprendre le principe, considérons un démultiplexeur à quatre sorties (voir figure suivante), donc deux lignes d'adressage et une ligne d'entrée :

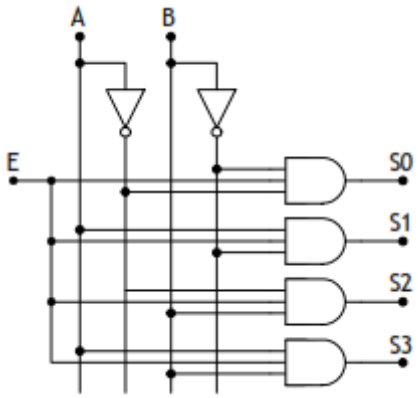


ADRESSES		SORTIES			
B	A	S0	S1	S2	S3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

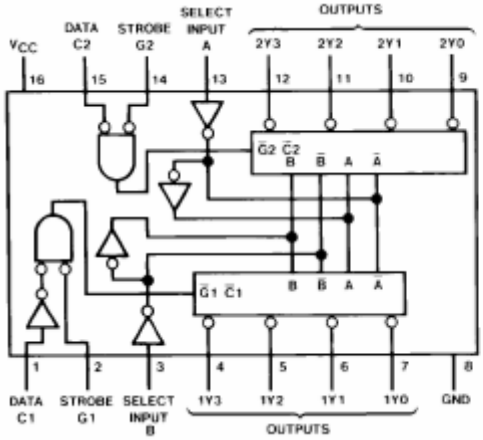
A partir de la table de vérité, on détermine les équations de sortie suivantes :

$$S0 = E.\bar{B}.A\ , S1 = E.\bar{B}.A, \qquad S2 = E.B.\bar{A}\ , S3 = E.A.B$$

Le schéma d'implémentation du démultiplexeur sera alors celui de la figure ci-dessous :



Les circuits intégrés réalisant cette fonction contiennent des entrées de validation (Strobe et Enable) permettant de sélectionner le circuit comme le 74LS155 qui est un double démultiplexeur 1 vers 4 dont le schéma de brochage et la table de vérité sont données à la figure suivante :



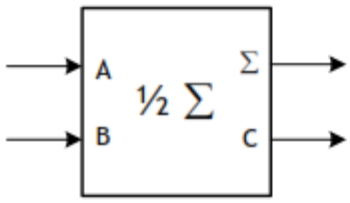
2-Line-to-4-Line Decoder or
1-Line-to-4-Line Demultiplexer

Inputs				Outputs			
Select		Strobe	Data				
B	A	G1	C1	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

9-4 L'additionneur :

Le demi-additionneur :

C'est un circuit permettant d'effectuer l'addition de deux bits A et B pour générer leur somme $\sum$ et leur retenue C (Carry) comme le montre le schéma et la table de vérité de la figure suivante :

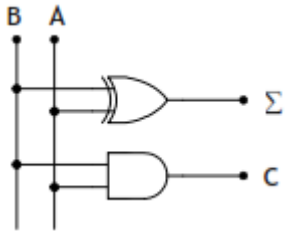


ENTREES		SORTIES	
B	A	Σ	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

A partir de la table de vérité, on peut écrire les deux fonctions sous la forme suivante :

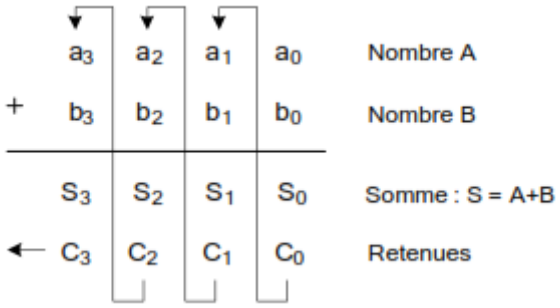
$$\sum = A.\bar{B} + \bar{A}.B = A \oplus B, \quad C = A.B$$

Ce qui peut être réalisé par le circuit schématisé sur le logigramme de la figure ci-dessous.

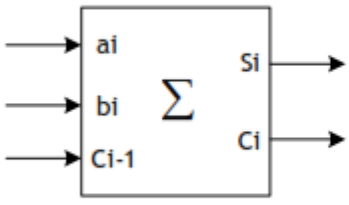


9-5 L'additionneur complet :

Pour effectuer une addition de deux nombres binaires de n bits, on additionne successivement les bits du même poids en tenant compte de la retenue de l'addition précédente comme le montre l'exemple suivant :



Il faut donc concevoir une cellule élémentaire appelée additionneur complet et qui permet de réaliser l'addition des bits ai et bi en plus de la retenue Ci-1 de l'addition précédente. Un tel circuit est définit par le schéma et la table de vérité de la figure suivante :



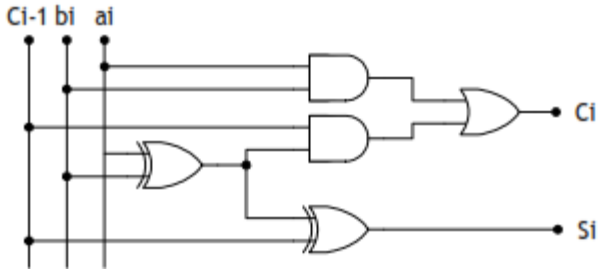
ENTREES			SORTIES	
ai	bi	Ci-1	Si	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A l'aide de la table de Karnaugh, on détermine les équations de sorties suivantes :

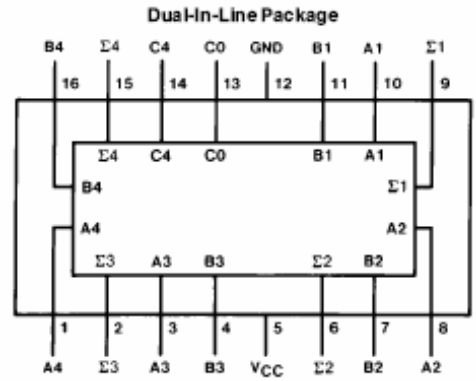
$$S_i = a_i.\bar{b}_i.\bar{C}_{i-1} + \bar{a}_i.b_i.\bar{C}_{i-1} + \bar{a}_i.\bar{b}_i.C_{i-1} + a_i.b_i.C_{i-1} = a_i \oplus b_i \oplus C_{i-1}$$

$$C_i = a_i.b_i + a_i.\bar{b}_i.C_{i-1} + \bar{a}_i.b_i.C_{i-1} = a_i.b_i + (a_i \oplus b_i).C_{i-1}$$

Le schéma d'implantation de l'additionneur complet sera celui de la figure suivante.



Comme exemple d'additionneur complet de mots de 4 bits, on peut citer le circuit intégré 74LS83 dont le schéma de brochage et la table de vérité sont données à la figure suivante :



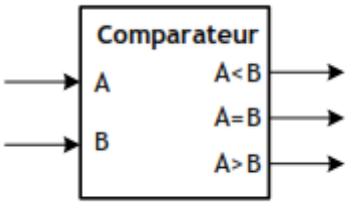
Truth Table

Inputs				Outputs					
				When C0 = L			When C0 = H		
				When C2 = L			When C2 = H		
A1 / A3	B1 / B3	A2 / A4	B2 / B4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

9-6 Le comparateur :

Un comparateur est un circuit permettant de détecter l'égalité de deux nombres et éventuellement d'indiquer le nombre le plus grand ou le plus petit.

Pour comprendre le principe, on va réaliser un comparateur simple permettant de comparer deux mots de 1 bit. La table de vérité d'un tel comparateur est donnée à la figure suivante :

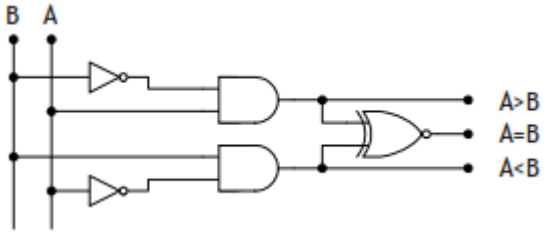


ENTREES		SORTIES		
B	A	S1:A<B	S2:A=B	S3:A>B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

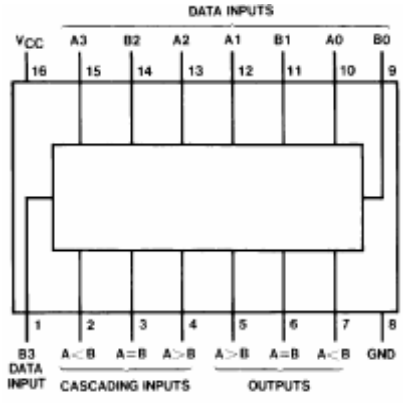
A partir de la table de vérité, on peut écrire les trois fonctions sous la forme suivante :

$$S1 = \bar{A}.B \quad , \quad S3 = A.\bar{B} \, , \quad S2 = S1 \oplus S3$$

Le schéma d'implantation de ce comparateur 2 bits sera celui de la figure suivante :



Comme exemple de comparateur binaire, on peut citer le circuit intégré 74LS85 dont le schéma de brochage et la table de vérité sont données à la figure suivante :



Function Table

Comparing Inputs				Cascading Inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

Chapitre III

LES SYSTEMES SEQUENTIELLES

1. Introduction

Toute machine fonctionne selon un cycle, c-à-d que partant d'un état donné, la machine effectuera différents mouvements, différentes actions et repassera à l'état de départ.

Tout ce qui se passe entre deux passages dans cet état de départ est appelé cycle.

Exemple : Poinçonneuse semi-automatique.

La poinçonneuse représentée très schématiquement ci-dessous se compose d'une table fixe, la tôle à poinçonner et d'un poinçon mobile.

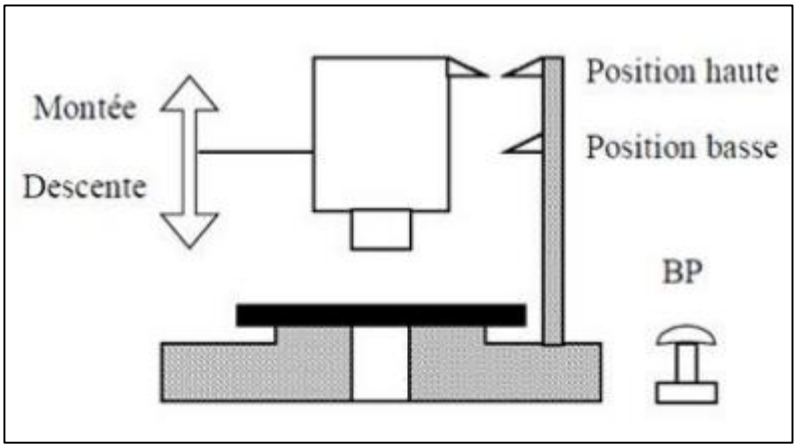


Figure 3-1 : Schéma simplifié d'une poinçonneuse semi-automatique

Considérons la poinçonneuse en sa position origine de repos, poinçon en haut.

L'opérateur en donnant l'information «Marche» en appuyant sur le bouton **BP** provoque automatiquement la descente du poinçon suivie de sa remontée en position de repos.

Nous dirons alors que la poinçonneuse a décrit un cycle.

Une séquence est un ensemble de comportements liés les uns aux autres par des conditions.

2. Règles de construction de diverses représentations graphiques d'une séquence ou d'un cycle :

Pour pouvoir construire les diverses représentations graphiques d'une séquence ou d'un cycle, il faut déterminer :

2-1 Les grandes étapes :

Sur une machine le comportement de l'automatisme se manifeste par des **actions** ou plus exactement par des **ordres** envoyés vers les organes chargés d'exécuter ces actions.
On appelle « **grande Étape**» chacun de comportements d'un système.

Reprenons l'exemple de la poinçonneuse semi-automatique. Une telle machine présentesuccessivement trois comportements différents.

- **ETAPE 1** : Comportement : La poinçonneuse est au repos
- **ETAPE 2** : Comportement : Descendre le poinçon.
- **ETAPE 3** : Comportement : Remonter le poinçon.

Nous pouvons donc, dans un premier temps, définir une étape comme une situation du cycle de fonctionnement pendant laquelle le comportement de l'automatisme de commande demeure constant.

Sous une autre forme, tout changement de comportement provoque obligatoirement lepassage à une autre étape.

Sur la poinçonneuse deux actions sont effectuées :

- La descente du poinçon associée à l'étape 2.
- La remontée du poinçon associée à l'étape 3.

2-2 Les points de prise de décision :

Il s'agit maintenant de déterminer ce qui provoque un changement de comportement de la machine, c'est-à-dire les conditions logiques qui déterminent le passage d'un comportement à un autre.

Nous qualifierons chaque passage d'un comportement à un autre comme étant le franchissement d'un point de prise de décision pour bien montrer son irréversibilité.

Par exemple, le passage de la position de repos (étape 1) à la descente du poinçon (étape 2) ne peut s'effectuer que si l'opérateur fournit l'information «Marche» et que si le poinçon est en position haute («condition initiale»).

Reprenons l'exemple de la poinçonneuse semi-automatique

ETAPE 1 : Étape initiale : Position initiale du poinçon.

Point de décision 1 : Condition de passage de l'étape 1 à l'étape 2 :
Information «marche» et poinçon en position haute.

ETAPE 2 : Descendre le poinçon.

Point de décision 2 : Condition de passage de l'étape 2 à l'étape 3 :
Poinçon en position basse.

ETAPE 3 : Remonter le poinçon.

Point de décision 3 : Condition de passage de l'étape 3 à l'étape 1 :
Poinçon en position haute.

Nous pouvons définir **les points de prise de décision** comme des points où on exploite des conditions variables impliquant le choix d'une voie parmi plusieurs ou le passage d'une étape à une autre. C'est là où on effectue des tests ou alternance.

Ces points de décision sont appelés aussi **transitions** qui sont conditionnées par des **réceptivités** constituées de fonctions logiques des différentes variables nécessaires au passage à l'étape suivante.

2-3 Les points d'entrée ou de sortie des données

Les points d'entrée ou de sortie des données sont des points où il faut mettre à disposition une information d'entrée à traiter ou il faut enregistrer une information de sortie traitée.

Considérons l'exemple du brassage du linge dans une machine à laver.

- ✓ Chaque nouvelle position du programmateur est une information d'entrée qu'on appelle**point d'entrée**.
- ✓ Dans un local, le chauffage ne doit fonctionner que pour des températures inférieures à 18°C . Après le test (θ_L < 18°C), sur la réponse OUI, on doit enregistrer l'information traitée (chauffage en marche), par contre, sur la réponse NON, on doit enregistrer l'information traitée (chauffage arrêté). L'enregistrement de l'une des informations traitées est un **point de sortie** des données.

2-4 Répétition ou arrêt de la séquence

La reprise de séquence ou boucle, permet de reprendre une ou plusieurs fois la même séquence tant qu'une condition fixée n'est pas obtenue (c'est un type d'aiguillage).

Exemple :

Reprenons l'exemple du chauffage d'un local.On a ici deux sortes de reprises de séquence :

- ✓ Après le test (θ_L < 18°C), sur la réponse OUI, c'est une boucle conditionnelle qui permetde faire marcher le chauffage et reprendre l'étape de la mesure de température.

- ✓ Après la dernière information de sortie (chauffage arrêté) c'est une boucle d'initialisation qui autorise le système à continuer sa régulation.

2-5 Saut de séquence

Exemple : Perceuse avec ou sans débourrage

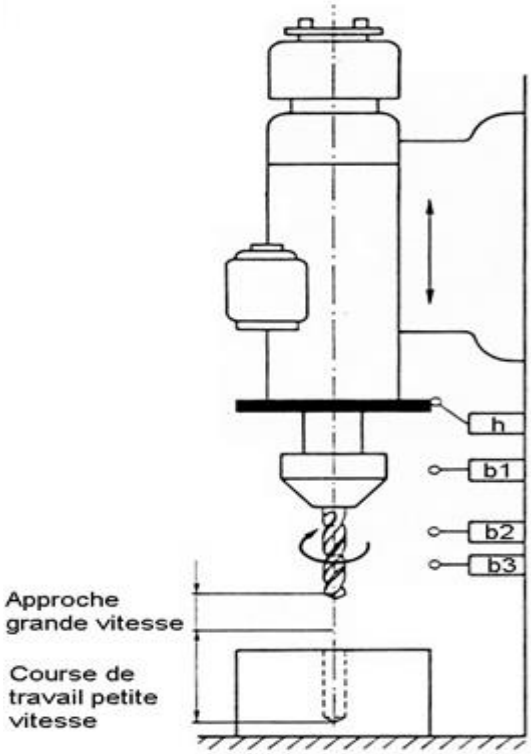
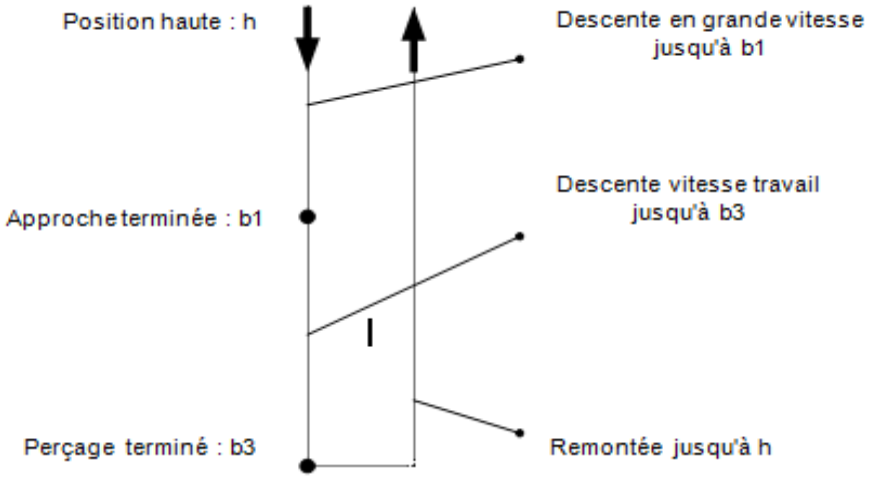


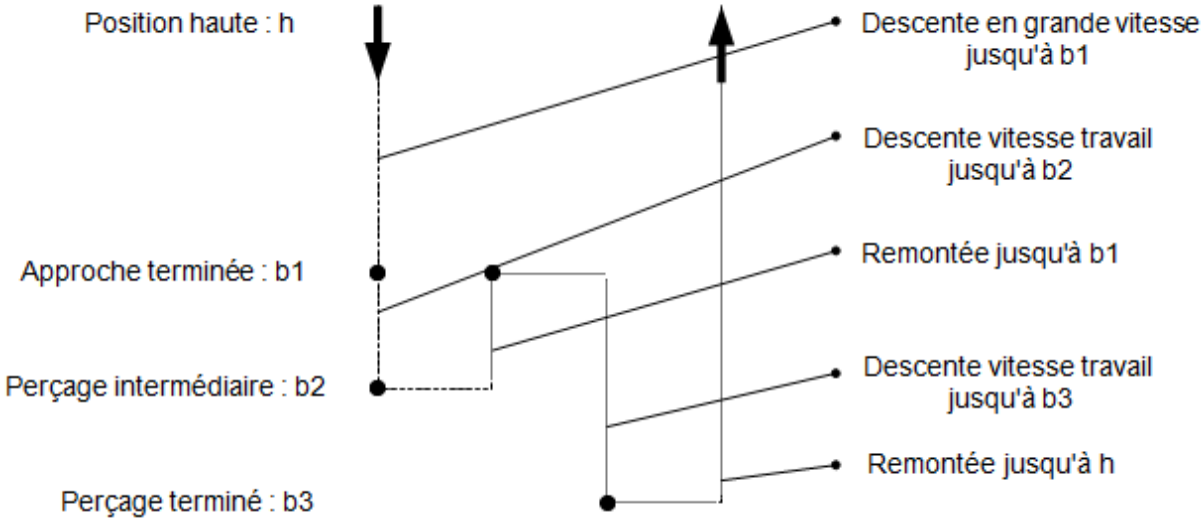
Figure 3-2 : Schéma simplifié d'une poinçonneuse semi-automatique

Suivant l'épaisseur et la nature des pièces à percer l'opérateur peut choisir entre deux cycles possibles :

- ✓ soit le **cycle sans débourrage** : comprenant les mouvements suivants :



- ✓ soit le **cycle avec débourrage** effectuant une remontée de la broche à une position intermédiaire afin de dégager le foret avant de terminer le perçage déjà commencé. Ce cycle est le suivant :



Les étapes du cycle avec débourrage sont :

ETAPE 1 : Étape initiale (ATTENTE)
ETAPE 2 : Descente en grande vitesse (APPROCHE)
ETAPE 3 : Descente en petite vitesse (PERCAGE)
ETAPE 4 : Remontée en grande vitesse (DEGAGEMENT)
ETAPE 5 : Descente en petite vitesse (PERCAGE)
ETAPE 6 : Remontée en grande vitesse (RETOUR)

Les étapes du cycle sans débourrage sont :

ETAPE 1 : Étape initiale (ATTENTE)

ETAPE 2 : Descente en grande vitesse (APPROCHE)
ETAPE 3 : Descente en petite vitesse (PERCAGE)
ETAPE 6 : Remontée en grande vitesse (RETOUR)

Remarquons que le cycle sans débourrage correspond au **saut des étapes 4 et 5** dont les comportements sont inutiles dans ce cycle.

2-6 Choix conditionnel entre plusieurs séquences

Dans le fonctionnement d'un équipement automatisé, il est nécessaire d'effectuer une sélection exclusive d'une séquence parmi plusieurs séquences (aiguillage).

Exemple: station de pompage (voir figure)

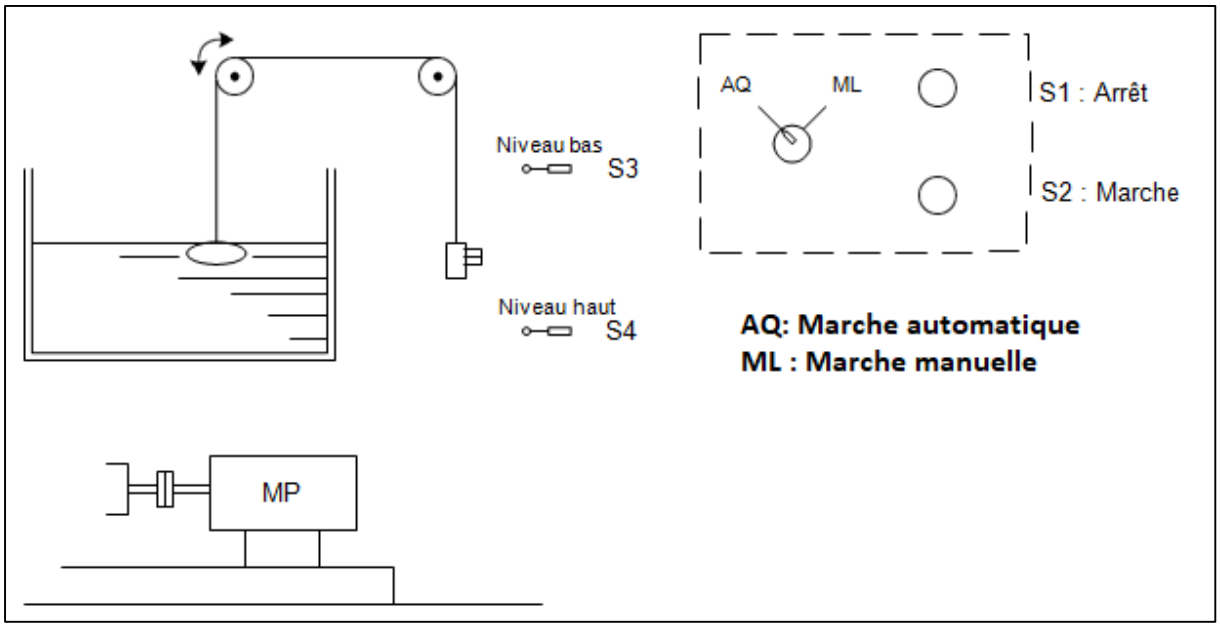


Figure 3-3 : Schéma simplifié d'une station de pompage

Un groupe motopompe alimente en eau, à partir des bassins de reprise, le réservoir d'un château d'eau. Deux modes de fonctionnement sont possibles :

- ✓ **Marche manuelle** : le responsable de l'installation commande à volonté la marche ou l'arrêt du groupe motopompe.
- ✓ **Marche automatique** : (commande automatique) : en fonction de deux niveaux prédéterminés d'eau dans le réservoir, niveau bas et haut, le groupe se met automatiquement en marche ou s'arrête.

On a donc une étape initiale commune aux deux modes de fonctionnement :

ETAPE 1 : étape initiale (ATTENTE)

Équipement sous tension.

Suivant que le commutateur est sur position marche automatique ou sur position marchemanuelle on a le choix entre deux séquences

Séquence 1 : marche manuelle

Point de décision : position du **commutateur sur ML** et **information marche**.

- ✓ **Étape 2** : Mettre le groupe en marche. Point de décision : information d'arrêt.
- ✓ **Étape 3** : Arrêter le groupe

Séquence 2 : Marche automatique
Point de décision : position du **commutateur sur AQ** et information **niveau bas atteint**.

- ✓ **Étape 4** : Mettre le groupe en marche
  Point de décision : information **niveau haut atteint**.
- ✓ **Étape 5** : Arrêter le groupe.

Après la fin de la séquence choisie 1 ou 2, on a un point de décision qui permet de vérifier si on a la position repos du contacteur du moteur de pompe et puis retour à l'étape initiale.

2-7 Séquences simultanées :

Le cycle de fonctionnement d'un équipement automatisé peut comporter plusieurs séquences qui s'exécutent simultanément mais dont les évolutions des étapes actives dans chaque séquence restent indépendantes.

Exemple : Poste de perçage (voir figure suivante)

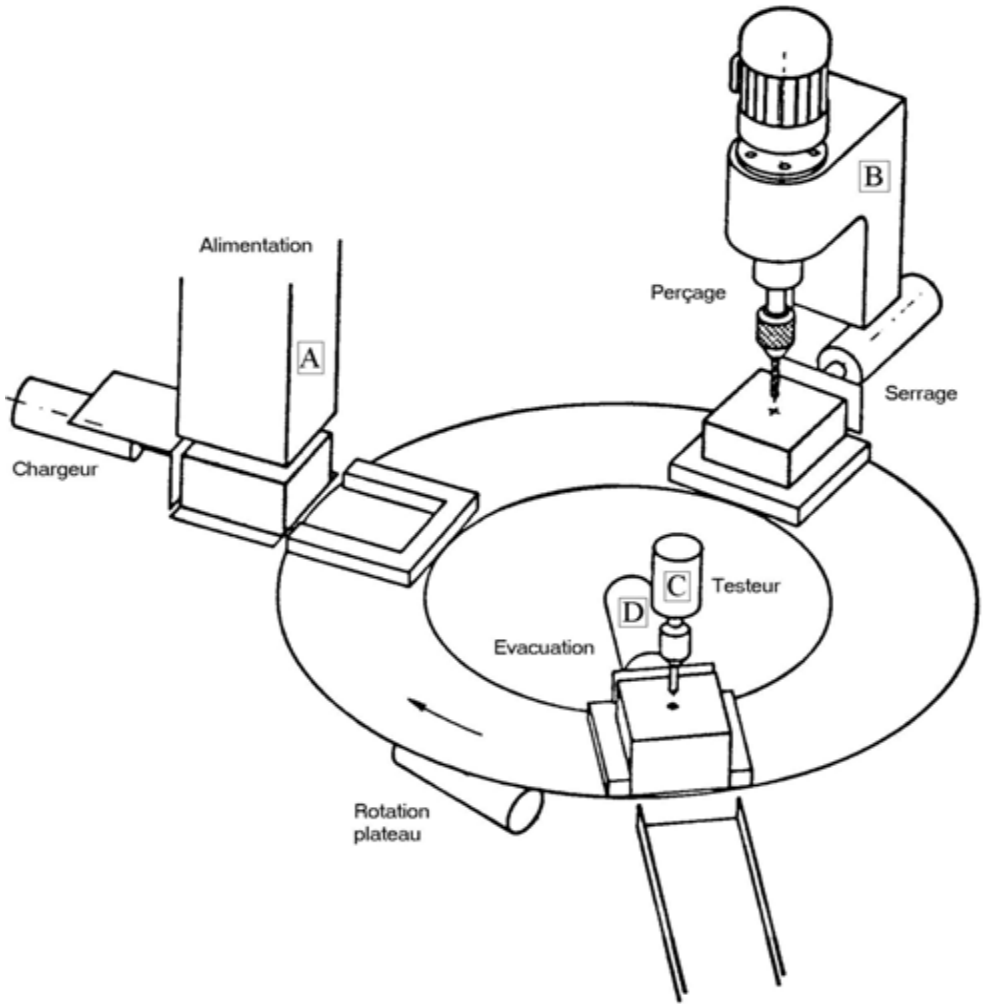


Figure 3-4 : Schéma simplifié d'un poste de perçage

Un plateau tournant dessert 3 postes de travail : le premier de chargement, le deuxième de perçage, et le troisième de contrôle et d'évacuation des pièces percées.

Donc on aura 3 séquences :

- ✓ Séquence 1 : de chargement
- ✓ Séquence 2 : de perçage
- ✓ Séquence 3 : de contrôle et d'évacuation.

Chacune de ces séquences est composée d'un certain nombre d'étapes.
Lorsque l'ordre marche apparaît à condition que la partie opérative soit correctement positionnée, les trois séquences précitées sont simultanément activées. A partir de cette situation les 3 trois évoluent indépendamment les unes des autres mais elles devront être toutes achevées pour aboutir à une évolution commune à l'étape qui provoque la rotation du plateau.

2-8 Aspect sécuritaire de la séquence

La nature de la séquence en elle-même représente un certain nombre de sécurités :

- ✓ Les étapes d'une séquence se déroulent dans un ordre chronologique bien déterminé.
- ✓ Une étape (2) ne peut être activée que si l'étape précédente (1) et la condition de passage de l'étape (1) à l'étape (2) est satisfaite.
- ✓ L'activation d'une étape entraîne immédiatement la désactivation de l'étape précédente
- ✓ Si au cours d'un fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste active.

Parfois, on doit répéter certaines conditions pour des raisons de sécurité. Prenons l'exemple de la perceuse avec ou sans débourrage déjà vue.

Pour reprendre l'étape initiale à partir de l'étape 6 on doit tester sur la position haute de la broche.

Pour passer de l'étape 1 à l'étape 2 on doit avoir l'information départ-cycle et on doit tester si on a la position haute de la broche et si on a la broche en rotation.

La répétition de la condition «position haute » peut paraître redondante car elle est écrite à la fois dans la condition de passage de l'étape 6→l'étape 1 et celle de passage de l'étape 1 → l'étape
2. Mais le fait d'avoir obtenue une fois cette condition vérifiée lors du passage de l'étape 6 à l'étape 1 ne prouve pas qu'elle soit toujours présente au moment de la demande de départ cycle, si des opérations de réglage ont eu lieu entre temps par exemple.

La condition de passage de l'étape 1 à l'étape 2 doit faire intervenir cette information pour des raisons de sécurité.

3. Modes de départ, de marche et d'arrêt d'une séquence.

Les principaux symboles associés à diverses représentations graphiques d'une séquence sont résumés dans le tableau ci-dessous :

3-1 MODE DE MARCHE :

Un mode de marche est un choix de fonctionnement, effectué par l'opérateur, conditionnant la façon dont doit se dérouler le cycle de l'automatisme de commande.

Malgré la grande variété des modes de marche rencontrés sur les automatismes industriels, il est possible de les regrouper en deux grandes catégories :

- ✓ Les marches automatiques ou de production.
- ✓ Les marches d'intervention.

Les marches automatiques :

Les marches automatiques sont considérées comme le fonctionnement normal de l'automatisme.

- ✓ Fonctionnement semi-automatique – Marche cycle par cycle – Cycle unique :

  Chaque cycle, commandé par l'information «départ cycle», se déroule automatiquement mais nécessite à chaque fois une nouvelle intervention de l'opérateur pour pouvoir exécuter le cycle suivant.

- ✓ Fonctionnement automatique – Marche cycle automatique – Cycles continus :

  Après action sur un bouton poussoir «départ cycle», le cycle se répète indéfiniment jusqu'à ce que l'ordre d'arrêt soit donné, cet arrêt ne s'effectuant qu'une fois le cycle terminé.

  Précisons bien que cette demande d'arrêt n'intervient que pour éviter l'exécution d'un nouveau cycle mais ne provoque pas l'arrêt du cycle en cours.

Les marches d'intervention :

Les marches dites d'intervention ou de maintenance, dont les plus connues sont les marches manuelles, nécessitent de la part de celui qui les utilise une connaissance très précise de la machine et de ses possibilités. Ces modes ne seront donc généralement exécutés que sous la responsabilité d'un régleur ou d'un agent de maintenance.

- ✓ Fonctionnement séquence par séquence ou étape par étape :

  Dans ces fonctionnements l'évolution du cycle est fractionnée séquence par séquence ou étape par étape, le passage d'une séquence à une autre ou d'une étape à la suivante s'effectuant sur commande de l'opérateur. De tels fonctionnements ne sont pas toujours possibles suivant la technologie utilisée.

  Ces modes de fonctionnement sont particulièrement utiles à la mise en route d'une installation, lors de la localisation d'un incident ou d'un réglage à effectuer. Ils permettent une analyse fine des différents comportements du cycle et facilitent les réglages de parties bien précises de la machine.

- ✓ Fonctionnement manuel :

L'exécution d'une action est directement liée à un ordre manuel, l'exécution de cet ordre étant généralement asservie à certaines sécurités.

3-2 MODE D'ARRET :

Les arrêts ne constituent pas à proprement parler un mode de marche mais peuvent imposer aussi au cycle des structures particulières.

L'arrêt momentané :

Un **arrêt momentané** interrompt immédiatement les ordres de commande de toute ou partie des actions en cours.

Il est donc possible, sous le contrôle de l'opérateur, de reprendre le fonctionnement du cycle à l'endroit où il a été interrompu.

Les arrêts d'urgence :

Un **arrêt d'urgence** provoque l'annulation de tous les ordres de commande, que ceux-ci soient manuels ou automatiques. Il peut quelques fois laisser certaines actions maintenues ou en enclencher d'autres suivant le sens de la sécurité.

L'arrêt d'urgence peut aussi effectuer la remise à zéro du ou des cycles, c'est à dire la désactivation de toutes les étapes actives, ou réinitialiser le cycle si cette opération ne s'avère pas dangereuse pour la partie opérative.

La machine doit donc dans certains cas être ramenée à sa position initiale ou d'origine, manuellement ou, à partir d'une séquence particulière de dégagement.

3-3 LES CONDITIONS INITIALES ET LE DEPART D'UNE SEQUENCE :

Les **conditions initiales** sont particulièrement importantes car elles correspondent au contrôle des positions que doit avoir la machine au début du cycle automatique.

Ces conditions initiales doivent être vérifiées systématiquement avant le démarrage de chaque cycle, même si elles ont déjà été obtenues à la fin du cycle précédent.

Ces conditions de départ doivent être insérées dans des registres à décalage, des compteurs binaires ou à décade.

4. Représentations graphiques d'une séquence

4-1 INTRODUCTION

Tout système automatisé comprend deux parties essentielles :

- Une **partie opérative (PO)** qui comporte des actionneurs permettant de réaliser les opérations sollicitées par la partie commande.
- Une **partie commande (PC)** permettant de piloter la partie opérative en fonction des informations qu'elle reçoit :

    - Soit des personnes extérieures au système par l'intermédiaire des boutons poussoirs, claviers etc.
    - Soit des capteurs contrôlant la partie opérative.

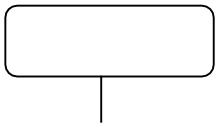
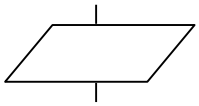
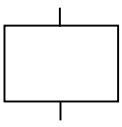
La partie commande de nombreux systèmes techniques automatisés est en logique séquentielle. Il est donc nécessaire de trouver des représentations graphiques qui permettent de représenter le fonctionnement de la partie opérative et de décrire ensuite le fonctionnement de la partie commande.

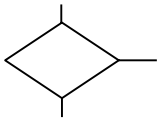
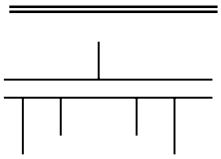
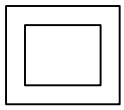
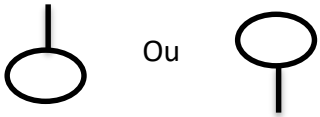
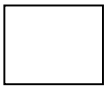
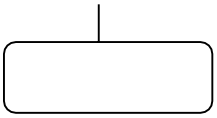
Ainsi la même représentation permet :

- L'analyse du problème à résoudre;
- L'étude de la partie commande;
- Le dépannage de la machine et sa commande.

4-2 DIFFERENTES REPRESENTATIONS GRAPHIQUES D'UNE SEQUENCE

Les principaux symboles associés à diverses représentations graphiques d'une séquence sont résumés dans le tableau ci-dessous :

SYMBOLES	DESIGNATIONS
	Début d'un ordinogramme
	Point d'entrée de données ou de sortie de résultats
	Action c'est-à-dire opération ou groupe d'opérations sur desdonnées. C'est le symbole général «traitement»

	Indication d'un point de décision (test ou alternance) C'est-à-dire exploitation de conditions variables impliquant lechoix d'une voie parmi plusieurs.
	Ce symbole est utilisé lorsqu'une ou plusieurs voies doivent l'avoir atteint avant qu'une ou plusieurs voies qui en sortent soient utilisées en parallèle ou suivant un ordre quelconque.
	Étape initiale
 Ou	Renvoi : donne la possibilité de raccorder des segments de grandes séquences.
	Étape simple
	Transition
	Fin d'un ordinogramme.

Sens conventionnel des liaisons

Le sens général des lignes de liaisons doit être :

- de haut en bas ;
- de gauche à droite ;

Lorsque le sens ainsi défini n'est pas respecté, des pointes de flèches à cheval sur la ligne indiquent le sens utilisé :  ; 

1) **<u>Algorithme – Algorigramme</u>** :

Un algorithme est une règle. Il s'exprime par une suite ordonnée de directives composées d'actions et de décisions qu'il faut exécuter en séquence suivant un enchaînement strict pouraccomplir une tâche quelconque.

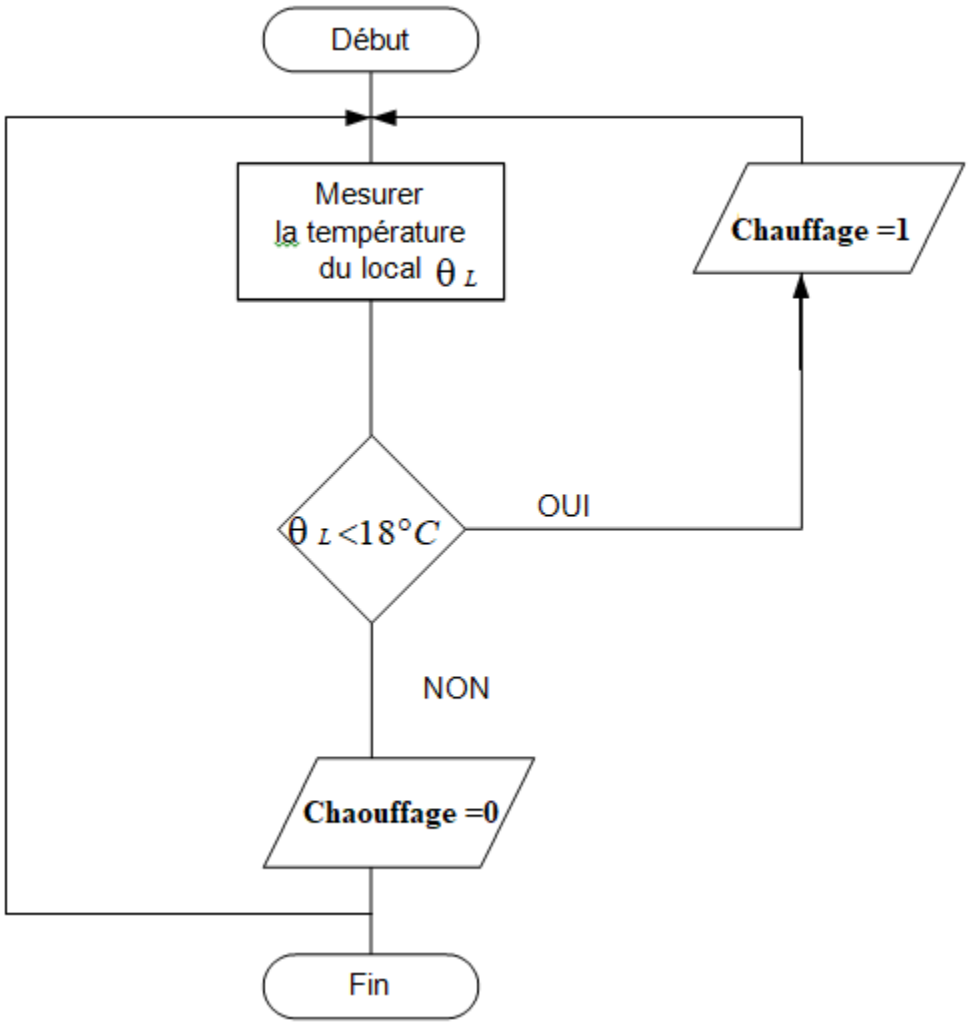
On peut considérer que toute succession de tâches logiques constitue l'algorithme de son résultat.

L'**algorigramme** reproduit dans une représentation graphique normalisée tous les cheminements du raisonnement logique qui détermine la composition de l'algorithme

Exemple :

a)- **Chauffage d'un local**

Dans un local le chauffage ne doit fonctionner que pour des températures inférieures à 18°C.



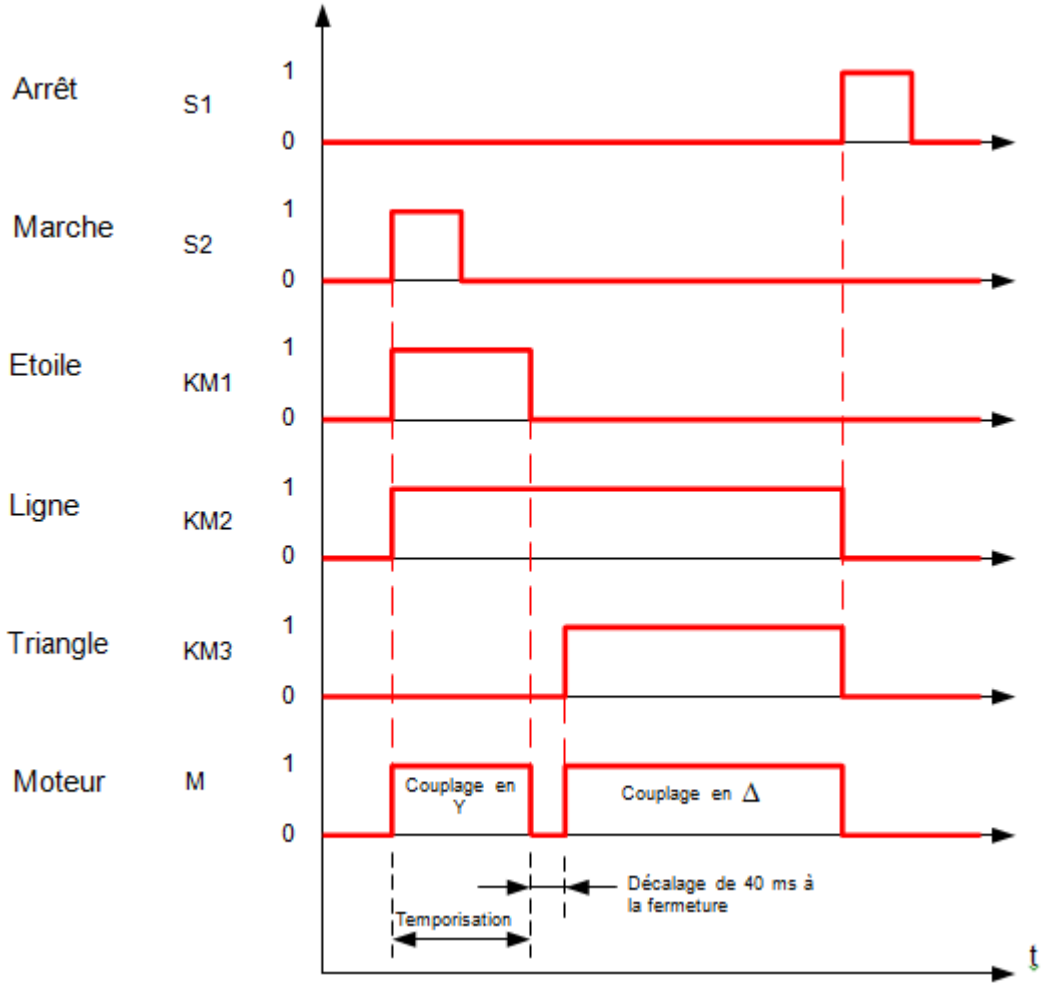
a)

2) Chronogramme

Il permet de visualiser l'interaction des variables binaires d'un circuit. Il représente par un graphique les états 0 et 1 de celles-ci en fonction du temps.

Exemple

Chronogramme d'un démarrage étoile-triangle d'un moteur asynchrone à rotor à cage : commande semi-automatique, un sens de marche.



3) Grafcet

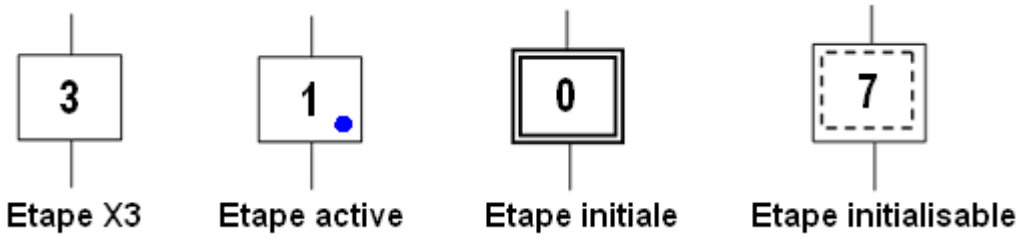
Le **Grafcet** est une représentation graphique du comportement d'un système automatisé.Le tracé de ce graphique est défini par :

- ✓ Des **éléments de base** : Étape, Transition, liaisons orientées permettant de construire lastructure séquentielle de l'automatisme ;
- ✓ Une interprétation : **Actions associées aux étapes**, Réceptivités associées aux transitionspermettant de décrire le fonctionnement de la partie opérative et de la partie commande ;
- ✓ Des **règles d'évolution**, permettant d'obtenir des documents pouvant être interprétés sansambiguïté par les différents intervenants dans l'automatisme.

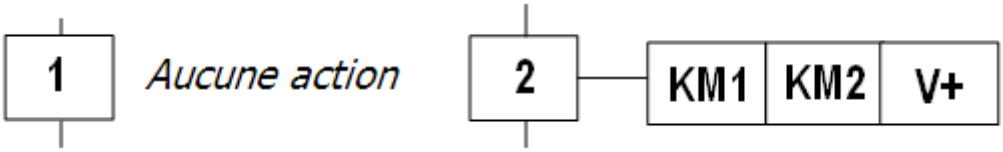
Éléments de base

- ✓ **Étape** : Caractérise un comportement invariant d'une partie ou de la totalité de la partie commande du système.

La situation initiale d'un système automatisé est indiquée par une étape dite étape initiale et représentée par un carré double.

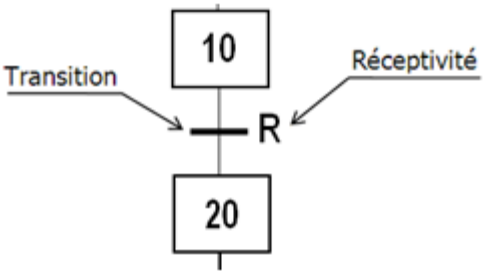


✓ **Actions associées à l'étape** : Elles traduisent ce qui doit être fait chaque fois que l'étape à laquelle elles sont associées est active. A chaque étape est associée une action ou plusieurs, c'est à dire un ordre vers la partie opérative ou vers d'autres grafcets. Mais on peut rencontrer aussi une même action associée à plusieurs étapes ou une étape vide (sans action).

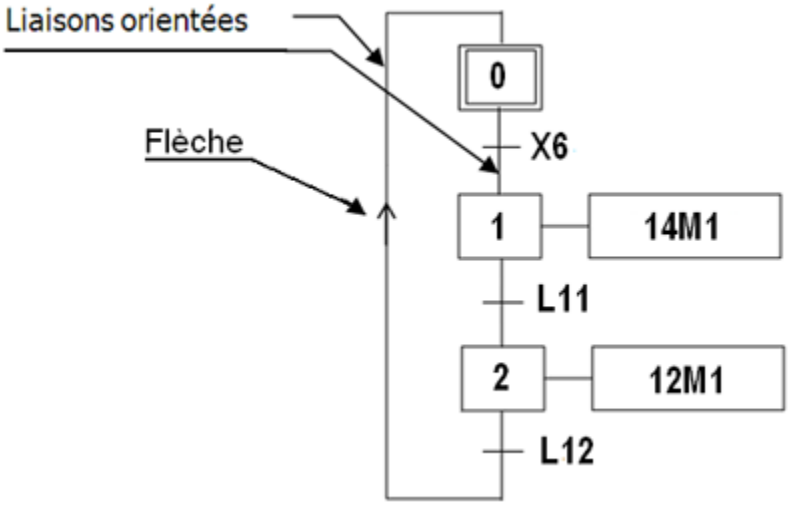


✓ **Transition** : Une transition indique la possibilité d'évolution qui existe entre deux étapes et donc la succession de deux activités dans la partie opérative. Lors de son franchissement, elle va permettre l'évolution du système.

✓ **Réceptivité associée à la transition** : A chaque transition est associée une condition logique appelée réceptivité qui exprime la condition nécessaire pour passer d'une étape à une autre. C'est une condition logique vraie ou fausse des différentes variables nécessaires au franchissement de la transition.



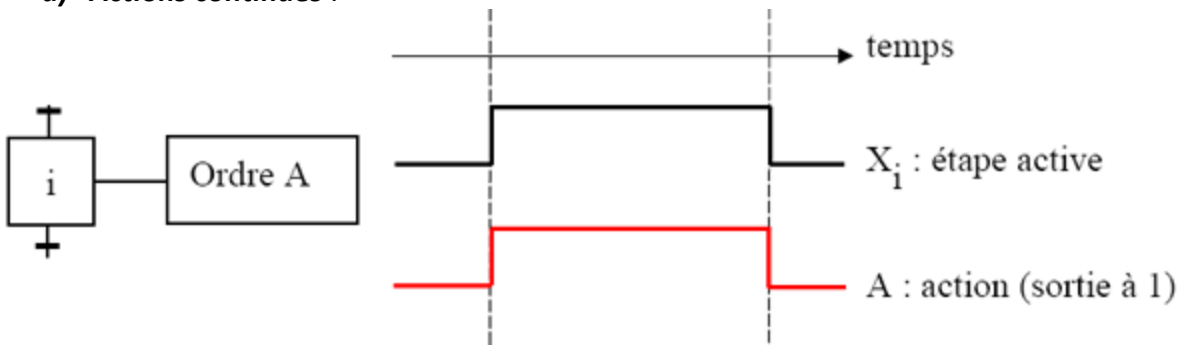
✓ **Liaison orientées** : Elles sont de simples traits verticaux qui relient les étapes aux transitions et les transitions aux étapes. Elles sont normalement orientées de haut vers le bas. Une flèche est nécessaire dans le cas contraire.



Classification des actions associées aux étapes

L'action associée à l'étape peut être de 3 types : **continue**, **conditionnelle** ou **mémorisée**. Les actions peuvent être classées en fonction de leur durée par rapport à celle de l'étape.

a) Actions continues *:*

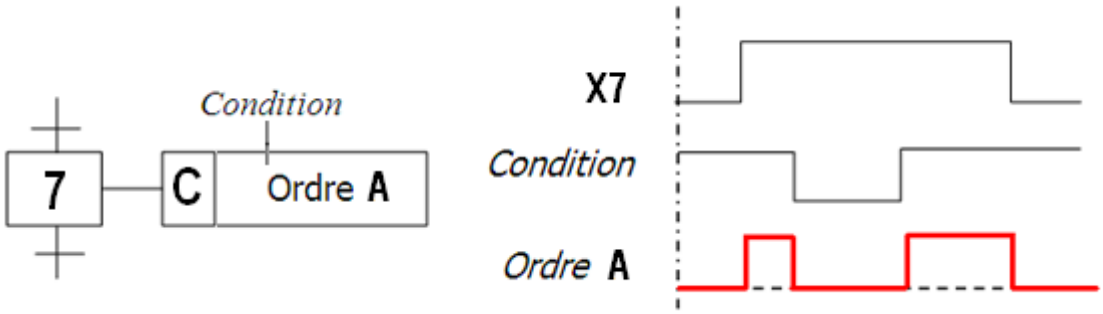


L'ordre est émis, de façon continue, tant que l'étape, à laquelle il est associé, est active.

b) Actions conditionnelles:

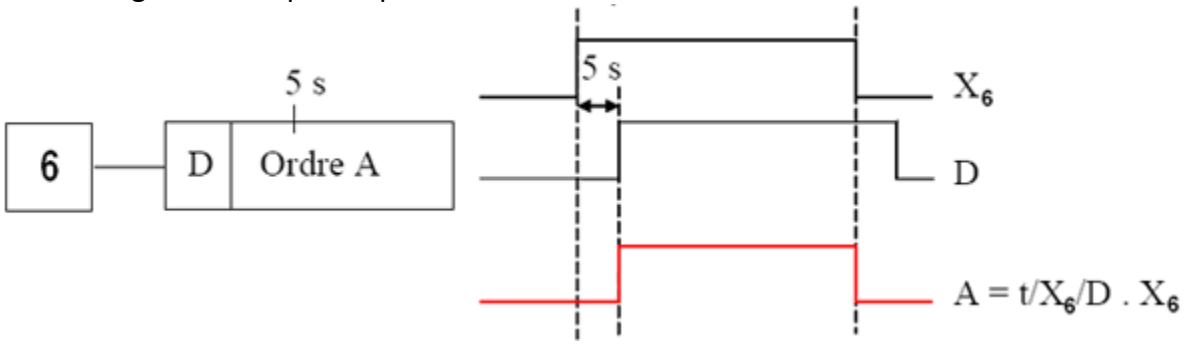
Une action **conditionnelle** n'est exécutée que si l'étape associée est active et si la condition associée est vraie. Elles peuvent être décomposées en 3 cas particuliers:

Action conditionnelle simple : ***Type C***



Action retardée : *Type D (delay)*

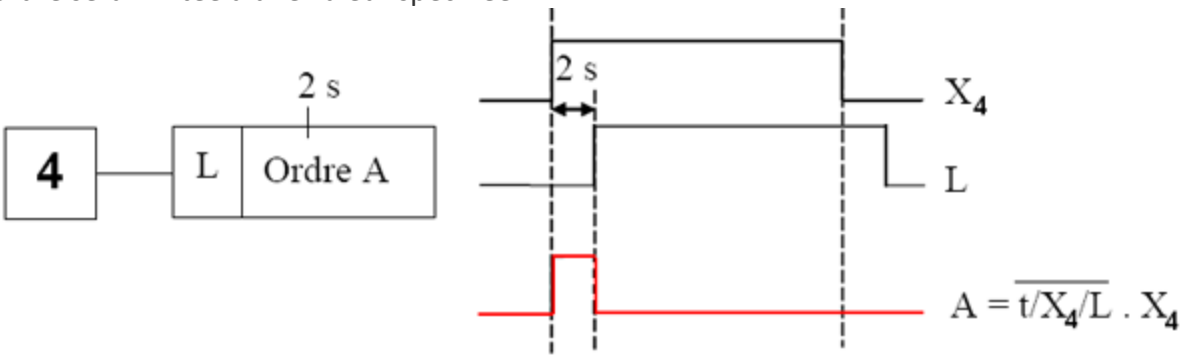
Le temps intervient dans cet ordre conditionnel comme condition logique. L'indication du temps s'effectue par la notation générale " t / xi / q " dans laquelle "xi" indique l'étape prise comme origine du temps et "q" est la durée du retard.



Exemple : "t /x6/ 5s" : prendra la valeur logique 1,5s après la dernière activation de l'étape 6.

Action de durée limitée: *Type L (limited)*

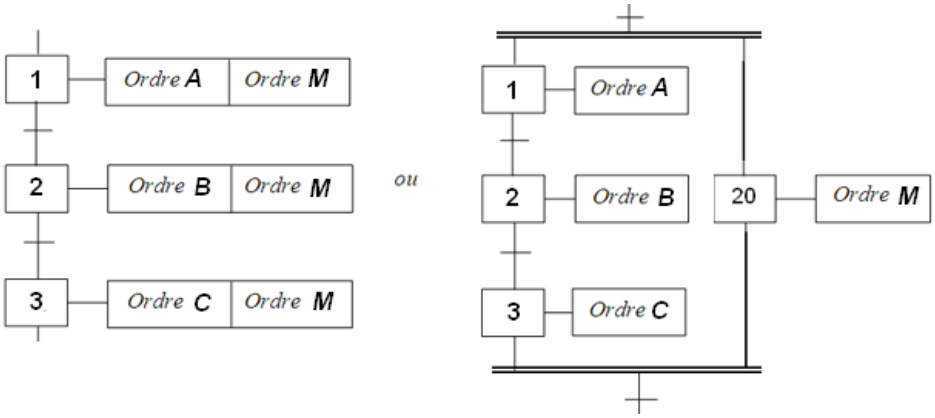
L'ordre est émis dès l'activation de l'étape à laquelle il est associé ; mais la durée de cet ordre sera limitée à une valeur spécifiée.



L'ordre "A" est limité à 2s après l'activation de l'étape 4.

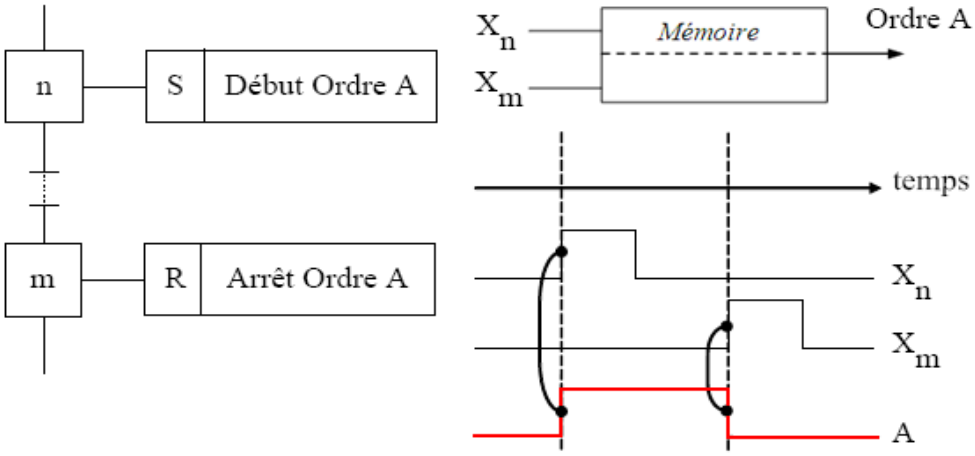
c) Action maintenue sur plusieurs étapes:

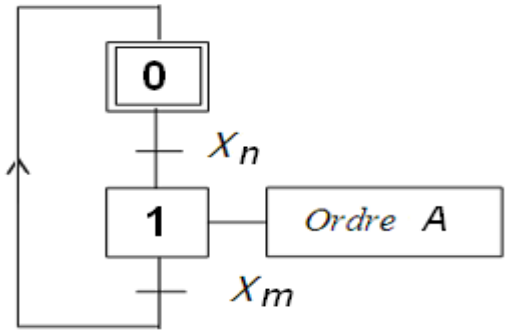
Afin de maintenir la continuité d'une action sur plusieurs étapes, il est possible de répéter l'ordre continu relatif à cette action, dans toutes les étapes concernées ou d'utiliser une description sous forme de séquences simultanées.



d) Action mémorisée :

Le maintien d'un ordre, sur la durée d'activation de plusieurs étapes consécutives, peut également être obtenu par la mémorisation de l'action, obtenue par l'utilisation d'une fonction auxiliaire appelée fonction mémoire.





Cette fonction pourra être décrite par un **GRAFCET**

<u>Règles d'évolution d'un GRAFCET</u>

Règle N°1 : Condition initiale

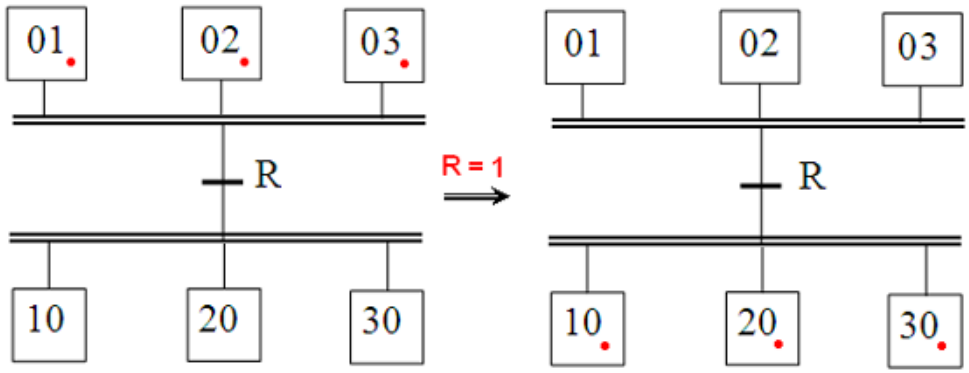
A l'instant initial, seules les étapes initiales sont actives.

Règle N°2 : Franchissement d'une transition.

Pour qu'une transition soit validée, il faut que toutes ses étapes amont (immédiatement précédentes reliées à cette transition) soient actives. Le franchissement d'une transition se produit lorsque la transition est validée, **ET seulement si** la réceptivité associée est **vraie**.

Règle N°3 : Evolution des étapes actives

Le franchissement d'une transition entraîne obligatoirement l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.



Règle N°4 : Franchissement simultané

Toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies.

Règle N°5 : Conflit d'activation

Si une étape doit être simultanément désactivée par le franchissement d'une transition aval, et activée par le franchissement d'une transition amont, alors elle reste active. On évite ainsi des commandes transitoires (néfastes à la partie opérative).

Emploi du diagramme fonctionnel Grafcet

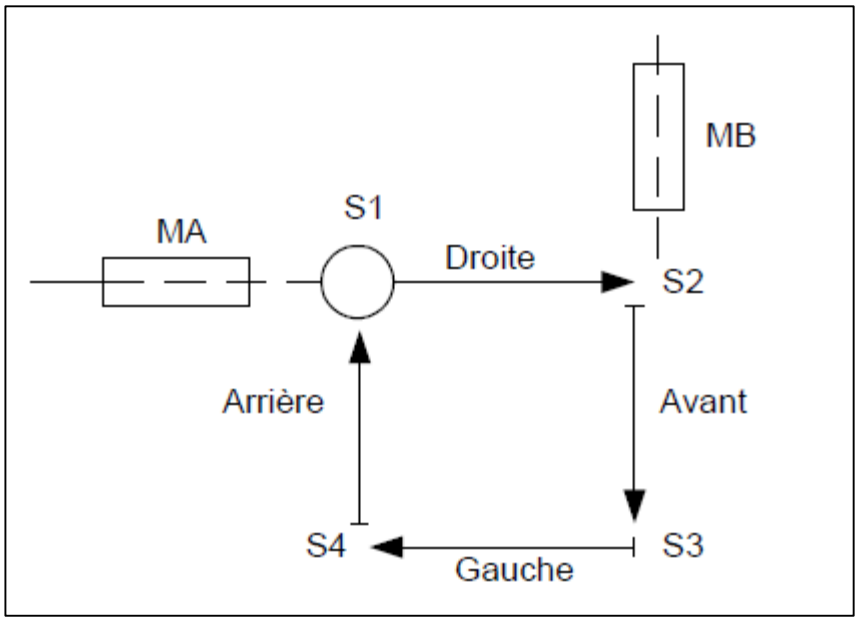
Afin de définir correctement le cahier des charges d'un équipement, le diagramme fonctionnel est utilisé à 2 niveaux :

- ✓ **Niveau 1** : Permet de comprendre ce que l'automatisme doit faire face aux différentes situations pouvant se présenter à lui. C'est un grafcet de point de vue partie opérative.

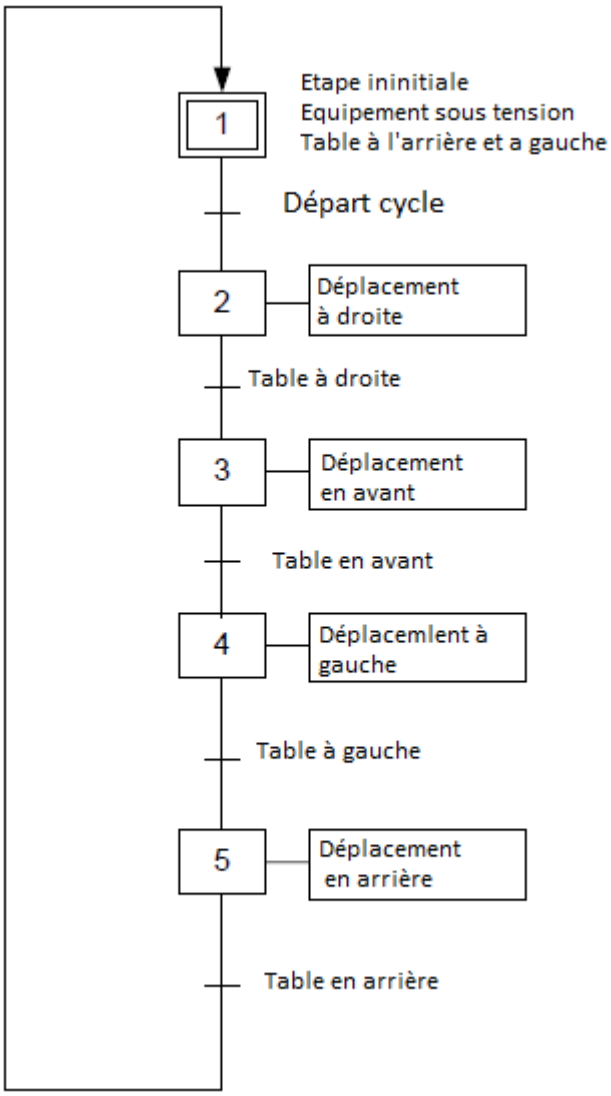
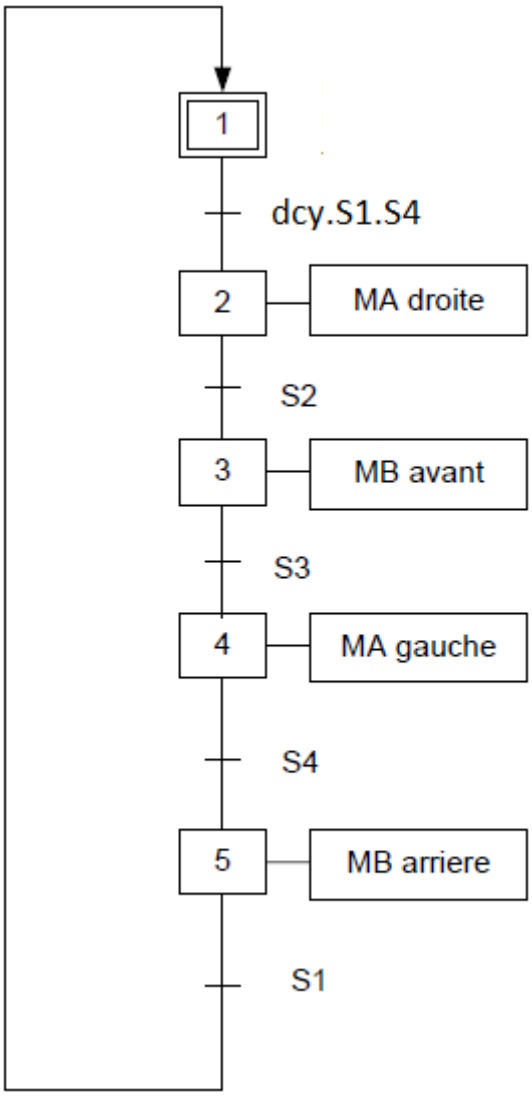
- ✓ **Niveau 2** : Le choix technologique étant fait, la description donne les précisions nécessairesà la réalisation pratique de l'équipement. C'est un grafcet de point de vue partie commande.

Exemple :

La table d'une machine-outil se déplace suivant un cycle carré (voir figure suivante). Deux moteurs MA et MB assurent respectivement les mouvements Droite-Gauche et Avant-Arrière. Les capteurs S1, S2, S3, S4 contrôlent la fin des mouvements.



Le Grafcet de niveau 1 est donné par la figure suivante :

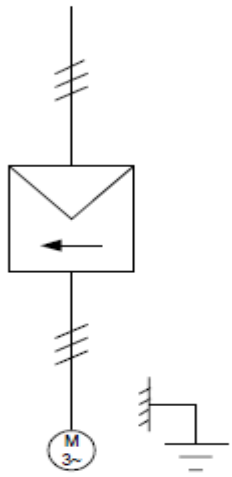
Grafcet Niveau 1	Grafcet Niveau 2
	

4) Schéma fonctionnel d'une machine

Le schéma fonctionnel d'une machine est destiné à faire comprendre son fonctionnement. Il représente par des symboles ou des figures simples une machine, une installation ou une partie d'installation avec ses interdépendances fonctionnelles, mais sans que toutes les liaisons soient représentées.

Exemple 1 :

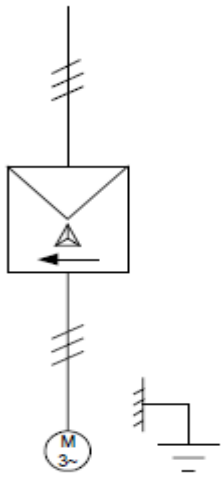
Symbole fonctionnel général d'un démarreur moteur avec un seul sens de marche.



Commande manuelle du démarrage direct d'un moteur triphasé asynchrone à rotor à cage avec un seul sens de marche.

Exemple 2 :

Symbole fonctionnel général d'un démarreur Y-□ de moteur avec un seul sens de marche.



Commande manuelle d'un démarrage étoile-triangle d'un moteur asynchrone triphasé à rotor à cage avec un seul sens de marche

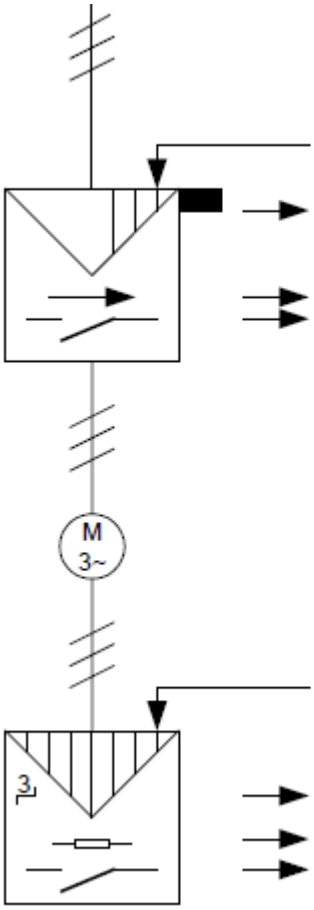
.

Exemple 3 :

Exemple avec toutes les représentations graphiques vues

Démarrage rotorique semi-automatique, trois temps, un seul sens de marche d'un moteur asynchrone triphasé à rotor bobiné.

- ✓ <u>Schéma fonctionnel</u>



Démarreur semi-automatique avec :
Mise à l'arrêt automatique;

Un sens de marche;
Par contacteurs.

Moteur asynchrone triphasé à rotor bobiné

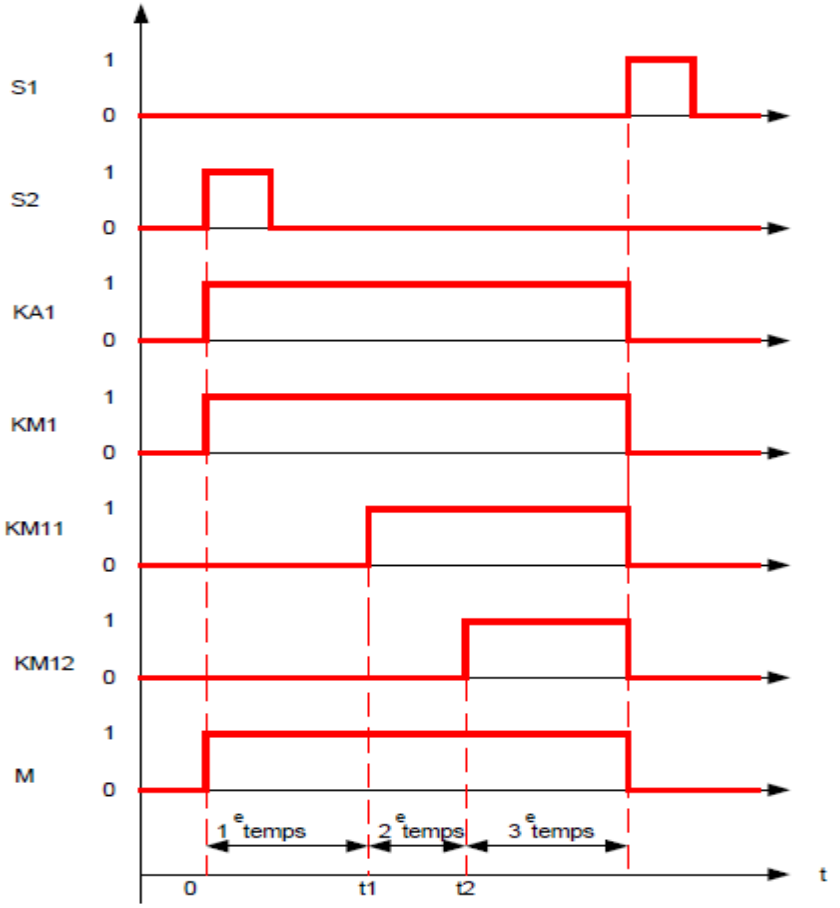
Démarrage rotorique automatique avec :

3 échelons (crans);
Rhéostatique;
Par contacteur

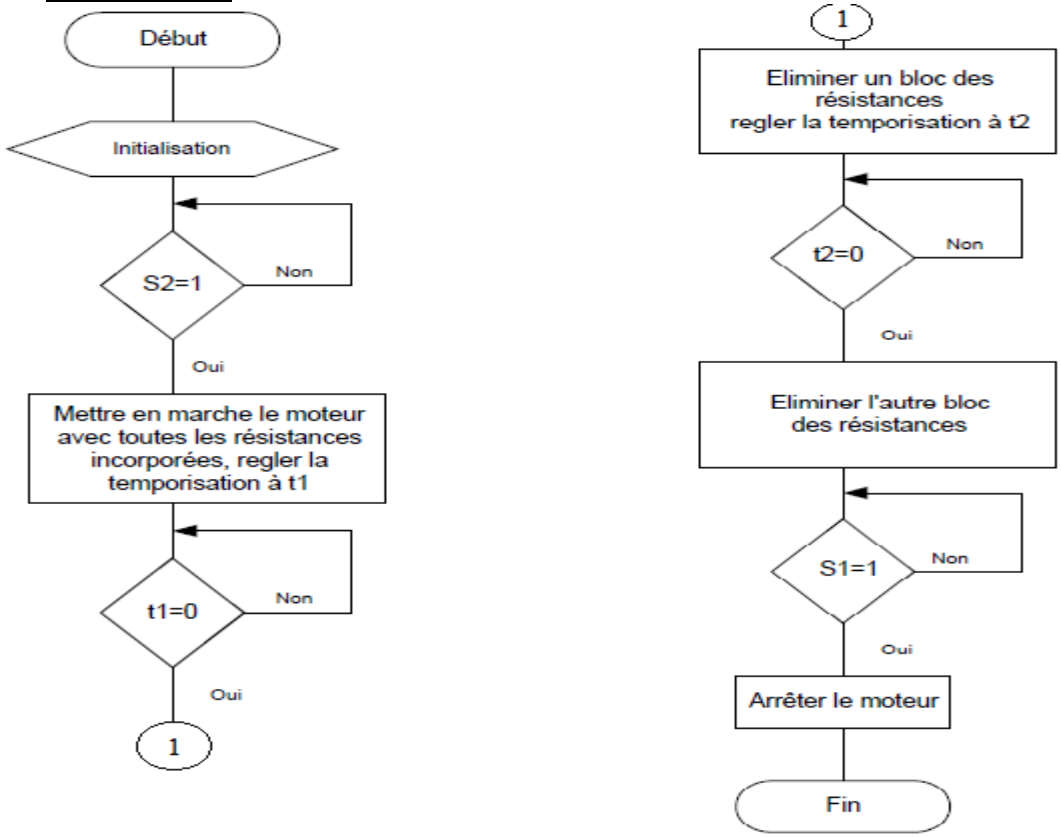
✓ Chronogramme

Nomenclature :

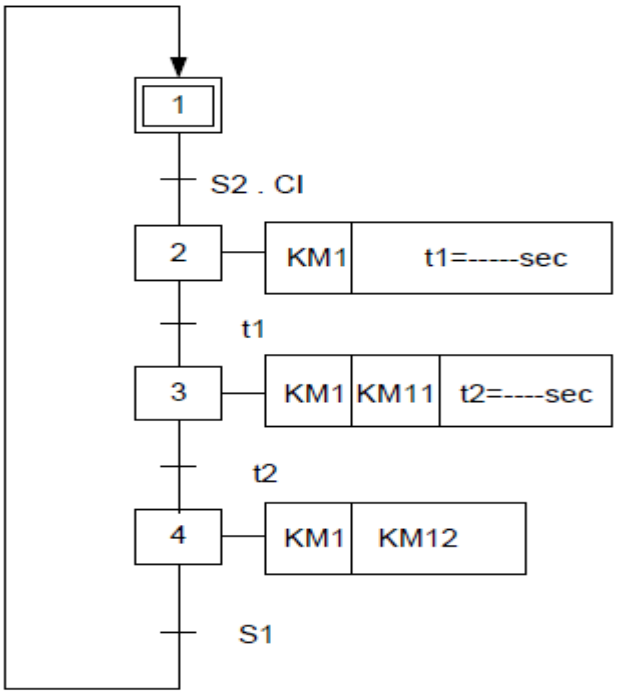
S1, S2 : boutons poussoirs « Arrêt » et « Marche »
KA 1 : contacteur auxiliaire
KM1 : contacteur tripolaire (réseau)
KM11 : contacteur tripolaire ou tétrapolaire (2ème temps)
KM12 : contacteur tripolaire ou tétrapolaire (3ème temps)
M3 ~ : moteur asynchrone triphasé à rotor bobiné.



✓ <u>Algorigramme</u>



✓ Grafcet



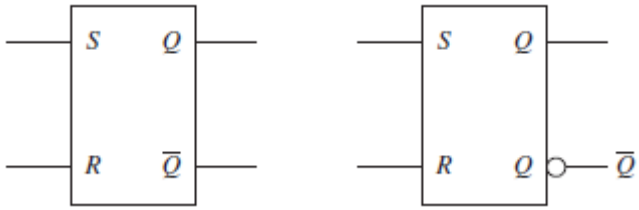
5. Les bascules

Une bascule est un circuit de mémorisation qui pour une combinaison d'états logiques de ses entrées présente sur sa sortie deux états complémentaires stables.

Une bascule est une mémoire élémentaire qui ne peut mémoriser qu'un seul bit.

5.1 Bascule RS

Le symbole générique de cette bascule est donné par le symbole suivant:



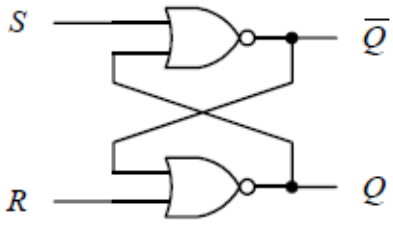
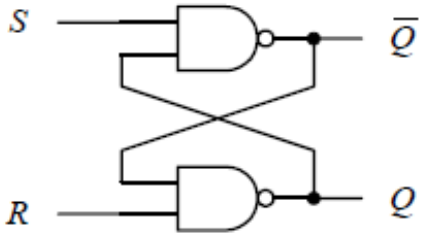
- ✓ **S** : entrée de mise à 1 (SET) de Q ;
- ✓ **R** : entrée de mise à 0 (RESET) de Q ;
- ✓ $\overline{\mathbf{Q}}$ et **Q** : sortie complémentaires.

Les ordres appliqués sur les entrées provoquent immédiatement en sortie le changement d'état correspondant. L'action simultanée sur R et S est interdite.

La table de vérité qui décrit le fonctionnement de la bascule RS est donnée comme suit :

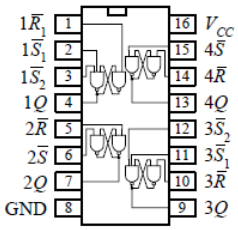
Entrées		Sorties		Mode de fonctionnement de la bascule
S	**R**	**Q**	**$\overline{Q}$**	
0	0	Inchangé		Mémorisation de l'état précédent
1	0	1	0	Mise à 1
0	1	0	1	Mise à 0
1	1	Ambiguïté		Les états de sorties sont indéterminés ne pas utiliser.

Le logigramme de la bascule RS, peut être réalisé par des portes logiques NAND ou NOR.

Logigramme en utilisant des opérateurs NOR :	**Logigramme en utilisant des opérateurs NAND :**
	

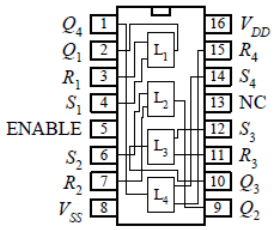
Exemples de circuits intégrés

74LS279A Quad SR latch



Note that two of the four latches have two S inputs and that inputs are active-LOW

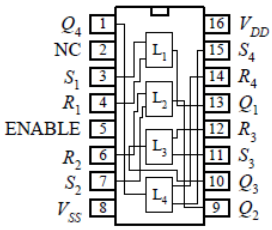
4043 Quad 3-state NOR SR latch



S	R	E	Q
X	X	0	OC
0	0	1	Hold
1	0	1	1
0	1	1	0
1	1	1	Δ

X = Don't care
OC = Open-circuit (high Z state)
Δ = Dominated by S=1 input

4044 Quad 3-state NAND SR latch



S	R	E	Q
X	X	0	OC
1	1	1	Hold
0	1	1	1
1	0	1	0
0	0	1	ΔΔ

X = Don't care
OC = Open-circuit (high Z state)
ΔΔ = Dominated by R=1 input

A LOW enable input effectively disconnects the latch states from the Q outputs, resulting in an open-circuit condition or high-impedance (Z) state at the Q outputs.

5.2 Bascule RSH

C'est une bascule synchrone à entrée d'horloge statique.

Dans la bascule RS, la sortie change d'état, au temps de propagation près, au moment où la combinaison des états des entrées est changée, son mode de fonctionnement est asynchrone.

Dans une bascule synchrone RSH le changement d'état de la sortie qui correspond à une nouvelle combinaison d'états d'entrées ne peut s'effectuer que sur le front actif, montant ou

descendant, d'un signal d'horloge.

La bascule RSH comprend :

- ✓ Trois entrées :
  - **S** : mise à 1 ;
  - **R** : mise à 0 ;
  - **H** : entrée d'horloge, active sur le front montant ou descendant du signal.

- ✓ Deux sorties : $\overline{\mathbf{Q}}$ et $\mathbf{Q}$ dont les états sont complémentaires.

La table de vérité qui décrit le fonctionnement de la bascule RSH est donnée comme suit :

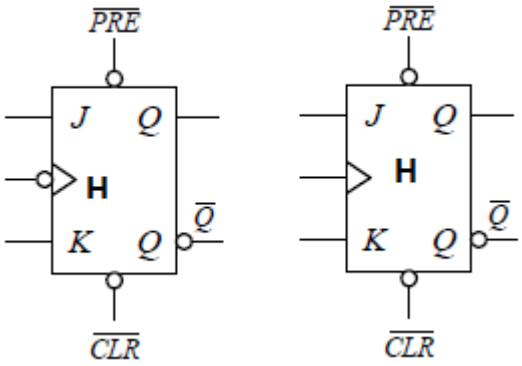
Entrées			**Sorties**		**Mode de fonctionnement de la bascule**
H	**S**	**R**	$\mathbf{Qn+1}$	$\overline{\mathbf{Q}}\boldsymbol{n+1}$	
ou	0	0	$\mathbf{Qn}$	$\overline{\mathbf{Q}}\boldsymbol{n}$	Mémorisation de l'état précédent (inchangé)
ou	1	0	1	0	Mise à 1
ou	0	1	0	1	Mise à 0
ou	1	1	Ambiguïté		Les états de sorties sont indéterminés ne pas utiliser.

5-3 Bascule JK :

La bascule JK permet d'effectuer trois opérations :

- placer la sortie à 0,
- placer la sortie à 1,
- ou faire le complément de la sortie actuelle.

Elle peut fonctionner en mode synchrone (front montant ou descendant) ou en mode asynchrone (en utilisant les entrées de forçage)



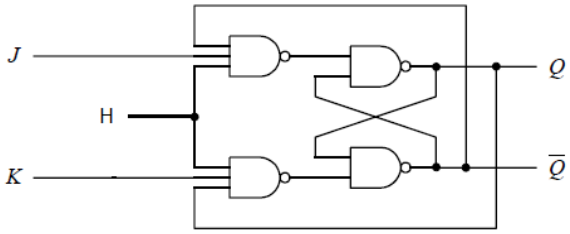
La bascule JK présente :

- ✓ cinq entrées :
  - **J** : mise à 1 ;
  - **K** : mise à 0 ;
  - **H** : entrée d'horloge, active sur le front montant ou descendant du signal.
  - $\overline{\boldsymbol{CLR}}$: Entrée de forçage pour la mise à zéro de la sortie Q (active en niveau bas)
  - $\overline{\boldsymbol{PRE}}$: Entrée de forçage pour la mise à 1 de la sortie Q (active en niveau bas)
- ✓ Deux sorties : $\overline{\mathbf{Q}}$ et **Q** dont les états sont complémentaires.

La table de vérité qui décrit le fonctionnement de la bascule JK avec front maontant est donnée comme suit :

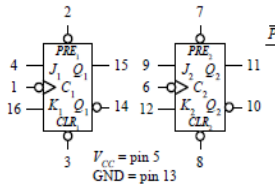
Entrées					**Sorties**		**Mode de fonctionnement de la bascule**
$\overline{PRE}$	$\overline{CLR}$	H	J	K	$Qn+1$	$\overline{Q}n+1$	
0	**1**	**x**	**x**	**x**	**1**	**0**	Forçage de la sortie à 1
1	**0**	**x**	**x**	**x**	**0**	**1**	Forçage de la sortie à 0
0	**0**	**x**	**x**	**x**	**1**	**1**	Les états de sorties sont indéterminés ne pas utiliser.
1	**1**	↑	**0**	**0**	**Qn**	$\overline{Q}n$	Mémorisation de l'état précédent (inchangé)
1	**1**	↑	**1**	**0**	1	0	Mise à 1
1	**1**	↑	**0**	**1**	0	1	Mise à 0
1	**1**	↑	**1**	**1**	$\overline{Q}n$	**Qn**	Changement d'état
1	**1**	0, 1, ou ↓	**1**	**1**	**Qn**	$\overline{Q}n$	Mémorisation de l'état précédent (inchangé)

Le logigramme de la bascule JK, peut être réalisé par des portes logiques NAND comme le montre la figure suivante :



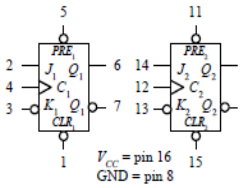
Exemples de circuits intégrés

74LS76 dual negative edge-triggered JK flip-flop with Preset and Clear



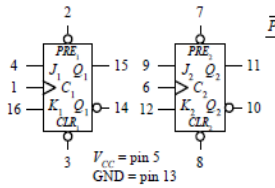
$\overline{PRE}$	$\overline{CLR}$	$\overline{C}$	J	K	Q	$\overline{Q}$	Mode
L	H	X	X	X	H	L	Preset
H	L	X	X	X	L	H	Clear
L	L	X	X	X	H	H	not used (race)
H	H	↓	h	h	$\overline{q}$	q	Toggle
H	H	↓	l	h	L	H	Reset
H	H	↓	h	l	H	L	Set
H	H	↓	l	l	q	$\overline{q}$	Hold

74109 dual JK positive edge-triggered flip-flop with Preset and Clear



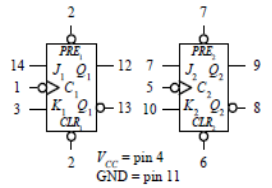
$\overline{PRE}$	$\overline{CLR}$	C_p	J	$\overline{K}$	Q	$\overline{Q}$	Mode
L	H	X	X	X	H	L	Preset
H	L	X	X	X	L	H	Clear
L	L	X	X	X	H	H	not used (race)
H	H	↑	h	l	$\overline{q}$	q	Toggle
H	H	↑	l	h	L	H	Reset
H	H	↑	h	l	H	L	Set
H	H	↑	l	l	q	$\overline{q}$	Hold

7476 dual pulse-triggered JK flip-flop with Preset and Clear



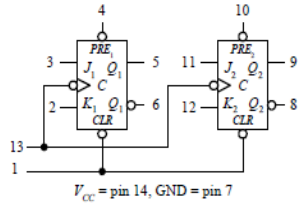
$\overline{PRE}$	$\overline{CLR}$	C	J	K	Q	$\overline{Q}$	Mode
L	H	X	X	X	H	L	Preset
H	L	X	X	X	L	H	Clear
L	L	X	X	X	H	H	not used (race)
H	H	⊓	h	h	$\overline{q}$	q	Toggle
H	H	⊓	l	h	L	H	Reset
H	H	⊓	h	l	H	L	Set
H	H	⊓	l	l	q	$\overline{q}$	Hold

74HC73 dual pulse-triggered JK flip-flop with Clear



CLR	CLK	J	K	Q	$\overline{Q}$	Mode
L	X	X	X	L	H	Clear
H	⊓	h	h	$\overline{q}$	q	Toggle
H	⊓	l	h	L	H	Reset
H	⊓	h	l	H	L	Set
H	⊓	l	l	q	$\overline{q}$	Hold

74114 dual pulse-triggered JK flip-flop with common Clock

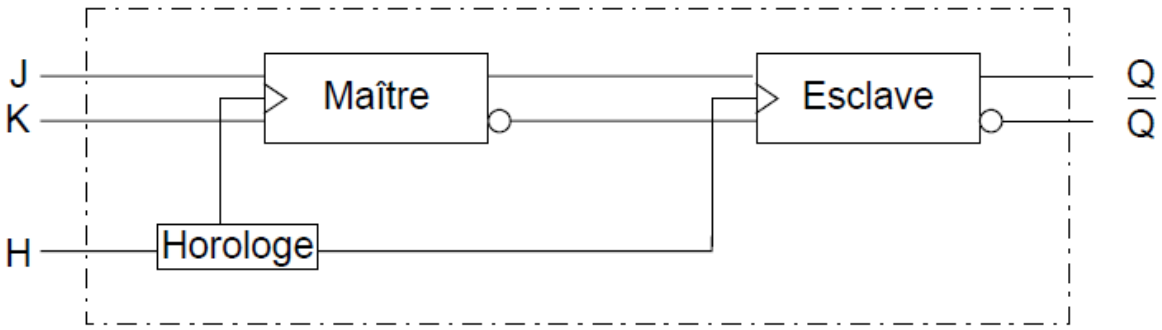


$\overline{PRE}$	$\overline{CLR}$	$\overline{C}$	J	$\overline{K}$	Q	$\overline{Q}$	Mode
L	H	X	X	X	H	L	Preset
H	L	X	X	X	L	H	Clear
L	L	X	X	X	H	H	not used (race)
H	H	↓	h	h	$\overline{q}$	q	Toggle
H	H	↓	l	h	L	H	Reset
H	H	↓	h	l	H	L	Set
H	H	↓	l	l	q	$\overline{q}$	Hold

H = HIGH voltage level steady state
h = HIGH voltage level one setup time prior to the HIGH-to-LOW Clock transition
L = LOW voltage level steady state
l = LOW voltage level one setup time prior to the HIGH-to-LOW Clock transition
q = Lowercase letters indicate the state of the referenced output prior to the HIGH-to-LOW Clock transition
X = Don't care
⊓ = Positive Clock pulse
↓ = Negative Clock edge
↑ = Positive Clock edge

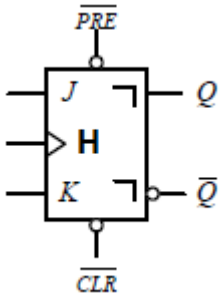
5-4 Bascule JK maître-esclave :

Elle est constituée de deux bascules JK, l'une maître, l'autre esclave, commutant à des niveaux différents du signale d'horloge.



La bascule maître reçoit les informations d'entrée et le front actif du signal d'horloge. La bascule esclave recopie la bascule maître sur le front opposé de l'horloge.

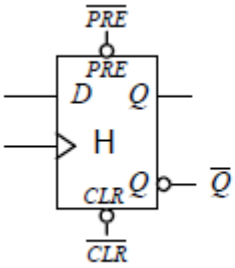
Le symbole de la bascule JK maître-esclave est donné par la figure suivante :

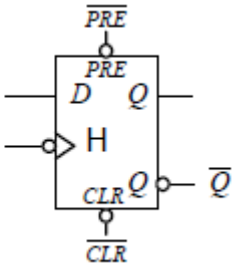


5-5 Bascule D :

La **bascule D** permet de générer un "retard" (delay). Elle est enclenchée par le signal d'horloge, l'unique entrée **D** (DATA) détermine l'état de la bascule. La sortie Q prend la même valeur que celle de l'entrée D quand le signal d'horloge effectue une transition.

Le symbole et la table de vérité d'une bascule D est donné par la figure suivante :





La bascule D présente :

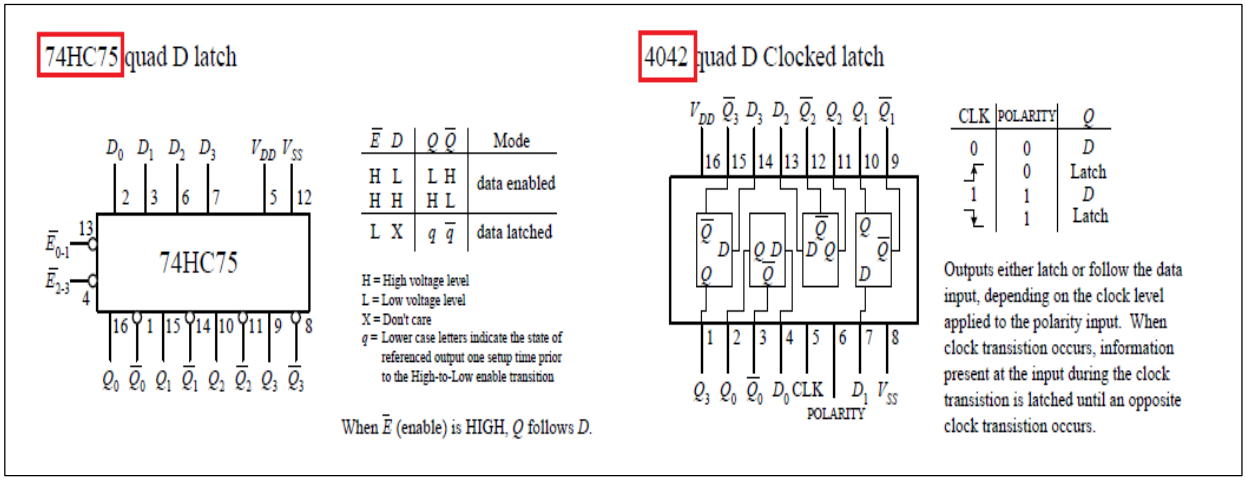
- Une entrée **D** (Data);
- **H** : entrée d'horloge, active sur le front montant ou descendant du signal.

- $\overline{CLR}$: Entrée de forçage pour la mise à zéro de la sortie Q (active en niveau bas)
- $\overline{PRE}$: Entrée de forçage pour la mise à 1 de la sortie Q (active en niveau bas)
- Deux sorties : $\overline{Q}$ et Q dont les états sont complémentaires

La table de vérité qui décrit le fonctionnement de la bascule D avec **front montant** est donnée comme suit :

Entrées				Sorties		Mode de fonctionnement de la bascule
$\overline{PRE}$	$\overline{CLR}$	H	D	$Qn+1$	$\overline{Q}n+1$	
0	1	x	x	1	0	Forçage de la sortie à 1
1	0	x	x	0	1	Forçage de la sortie à 0
0	0	x	x	1	1	Les états de sorties sont indéterminés ne pas utiliser.
1	1	0, 1, ou ↓	x	Qn	$\overline{Q}n$	Mémorisation de l'état précédent (inchangé)
1	1	↑	1	1	0	Mise à 1
1	1	↑ ou	0	0	1	Mise à 0

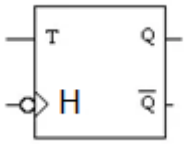
Exemples de circuits intégrés



5-6 Bascule T :

La **bascule T** tire son nom du terme anglais '**toggle**'.

- Si **T = 1**, on bascule à chaque impulsion d'horloge.
- Si **T = 0**, la sortie ne change pas : état mémoire.



La bascule D présente :

- Une entrée **T** (Toggle);
- **H** : entrée d'horloge, active sur le front montant ou descendant du signal.
- Deux sorties : $\overline{\mathbf{Q}}$ et **Q** dont les états sont complémentaires

Elle n'existe pas intégrée sauf dans des PLD, FPGA... mais on peut la fabriquer avec une bascule D en reliant la sortie non Q à l'entrée D (toutefois on ne réalise ainsi qu'une bascule T avec T=1), ou à l'aide d'une bascule JK en reliant J et K pour faire l'entrée T.

La table de vérité qui décrit le fonctionnement de la bascule T avec **front montant** est donnée comme suit :

Entrées		Sorties		Mode de fonctionnement de la bascule
H	**T**	$\mathbf{Qn+1}$	$\overline{\mathbf{Q}}\boldsymbol{n+1}$	
x	**0**	$\mathbf{Qn}$	$\overline{\mathbf{Q}}\boldsymbol{n}$	Mémorisation de l'état précédent (inchangé)
	**1**	$\overline{\mathbf{Q}}\boldsymbol{n}$	$\mathbf{Qn}$	Changement d'état

6. Les compteurs- décompteurs

Un compteur (ou décompteur) est un circuit électronique constitué essentiellement par un ensemble de bascules et le plus souvent d'un réseau combinatoire.

Ce compteur (ou décompteur) permet de comptabiliser le nombre d'événements qui se produisent pendant un temps donné. Chaque événement est traduit en impulsion électrique.

Ces circuits possèdent le plus souvent une entrée (quelquefois deux entrées) sur laquelle parviennent les impulsions à compter ou à décompter.

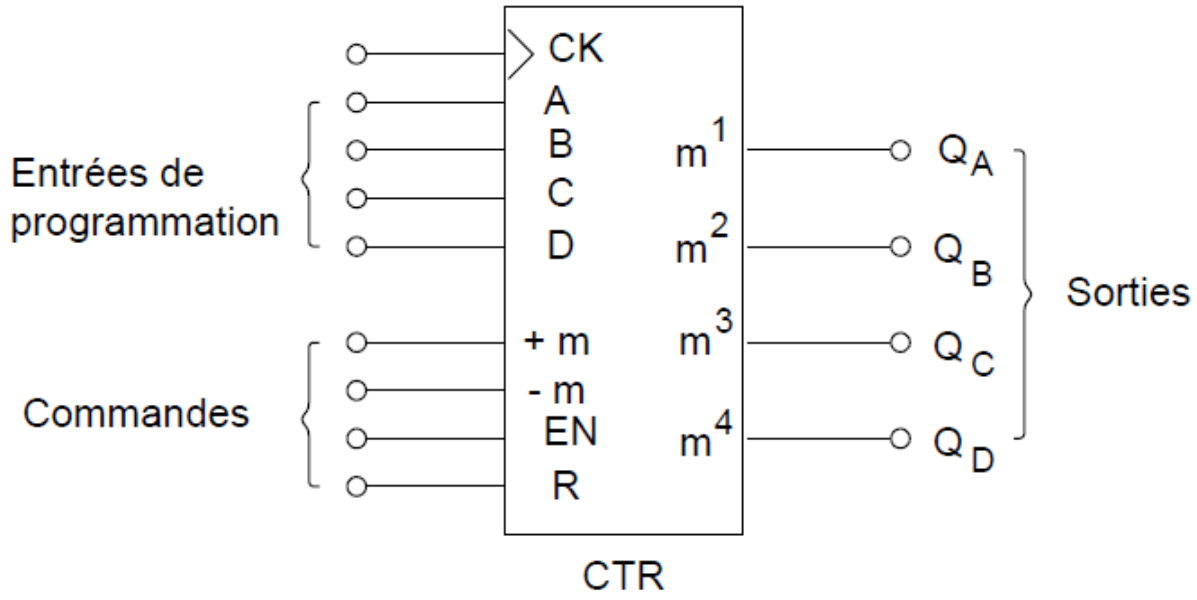
L'information disponible est située sur l'ensemble des sorties des bascules.

Le compteur permet de dénombrer dans la limite des bascules qui le constituent (capacité du compteur) les impulsions appliquées en entrée.

Fonction : Un compteur est un système logique dont les sorties changent d'état chaque foisqu'une information appropriée est appliquée à son entrée.

Le compteur peut être :

- ✓ binaire si le facteur de démultiplication est égale à 2 ou une puissance entrée de 2.
- ✓ Décimal si le facteur de démultiplication est égal à 10 ou une puissance entrée de 10.
- ✓ Modulo n dans les autres cas.



CTR = compteur,

Si le compteur travaille en **décompteur par m** ou modulo m sa définition symbolique devient : **CTR Div m**.

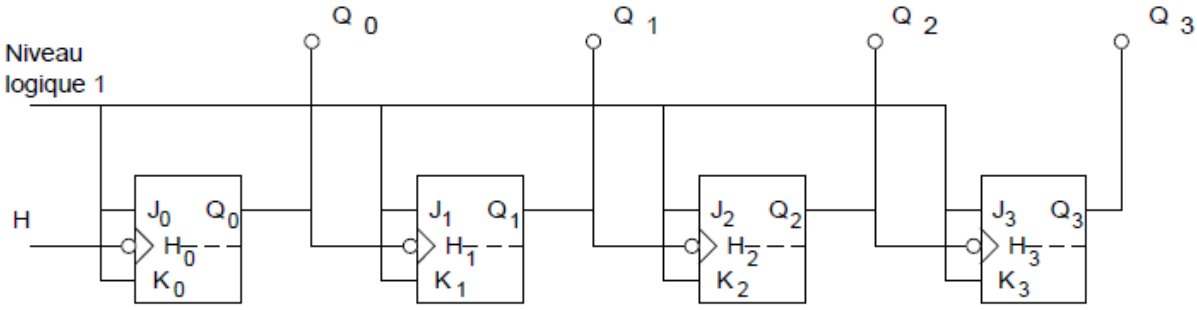
La symbolisation des commandes se définit par :

- ✓ **+m** = entrée de comptage
- ✓ **-m** = entrée de décomptage
- ✓ **EN**= entrée de validation du comptage ou décomptage
- ✓ **R**= entrée de remise à zéro
- ✓ **m1, m2, m3, m4**= sorties du compteur.

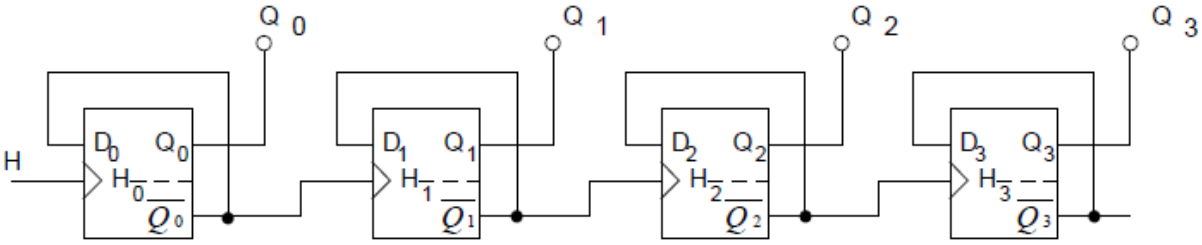
Réalisation d'un compteur asynchrone linéaire à base de bascules :

Le nombre de bascules est égal au nombre de bits, les liaisons entre les bascules restent les mêmes quel que soit le nombre de bits.

- **Compteur asynchrone binaire 4 bits** réalisés à partir de 4 bascules JK



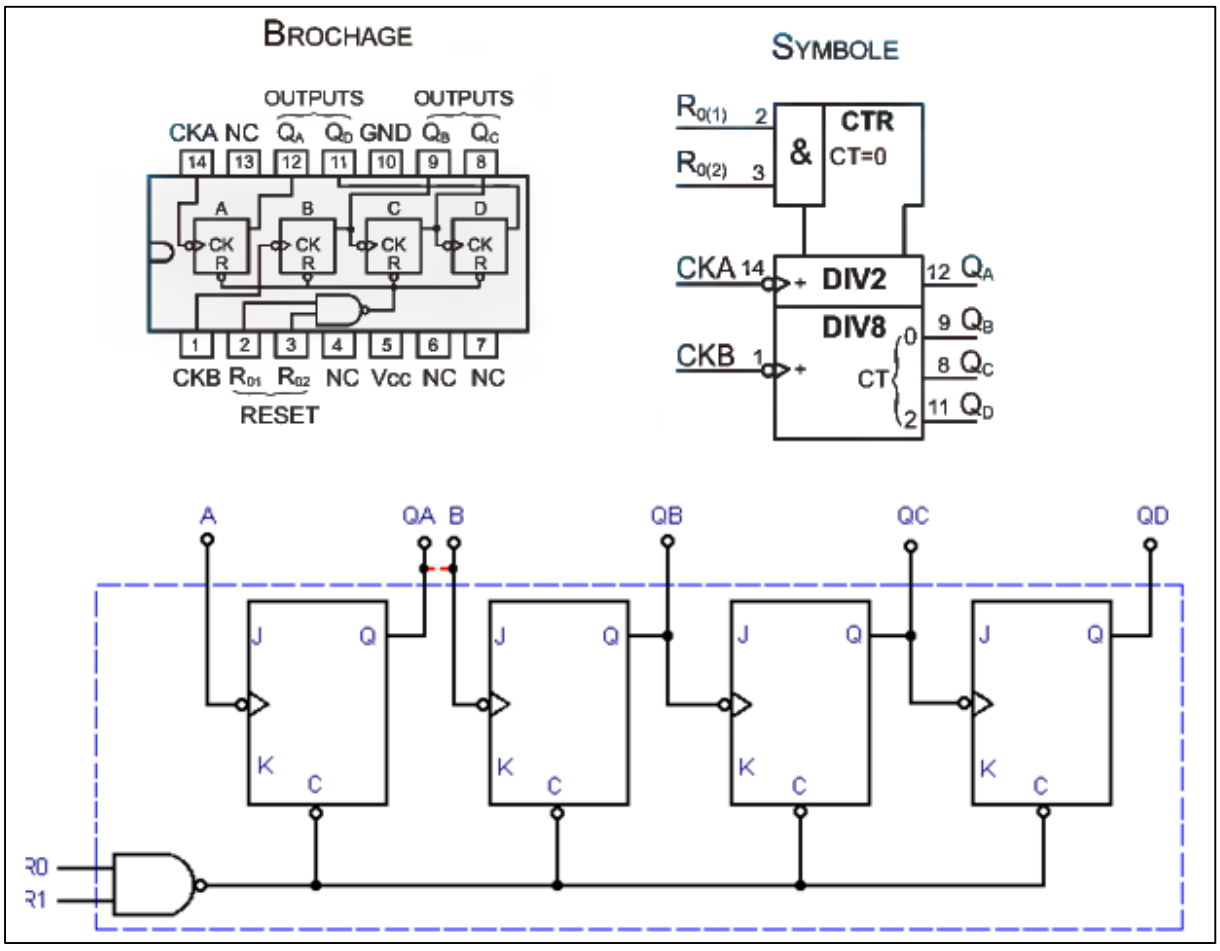
- **Compteur asynchrone binaire 4 bits** réalisés à partir de 4 bascules D



Exemples de circuits intégrés :

- <u>LE COMPTEUR INTÉGRÉ 7493</u>

La figure suivante représente le schéma de principe du compteur intégré 7493 réalisé en technologie TTL ainsi que son brochage.



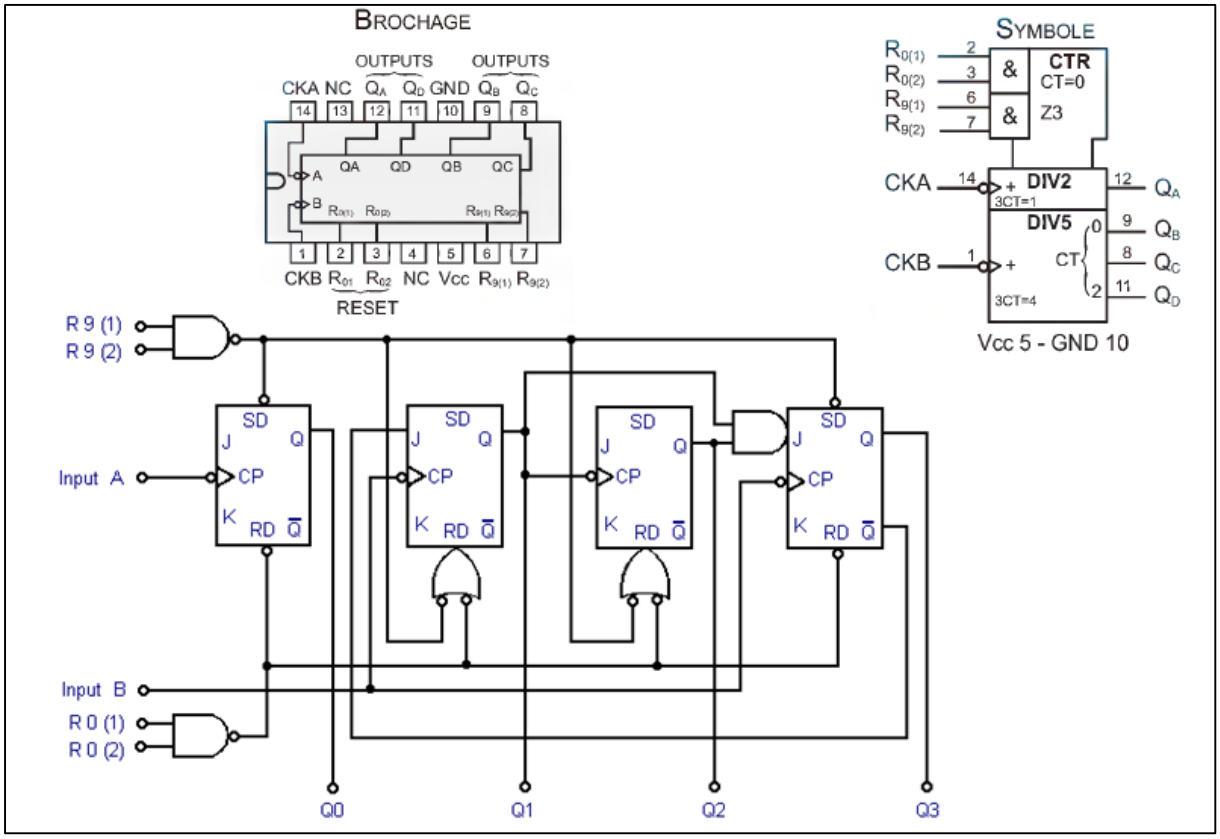
Les entrées J et K des bascules sont câblées intérieurement à «1».

Une remise à zéro générale asynchrone du compteur est possible grâce aux entrées R0 et R1. Pour cela les deux entrées R0 et R1 doivent être simultanément à «1».

Ce compteur peut fonctionner en diviseur par 8 en présentant l'horloge sur l'entrée B ou en diviseur par 16 en présentant l'horloge sur l'entrée A et en reliant la sortie QA à l'entrée B.

- UN COMPTEUR INTÉGRÉ MODULO 10 : LE 7490

C'est un compteur très utilisé, réalisé en technologie TTL. Son schéma est donné à la figure suivante :



Ce circuit comporte deux sections. Une section diviseur par 2 et une section diviseur par 5.

Il est possible soit de les utiliser séparément, soit de les réunir ensemble pour obtenir un compteur BCD modulo 10 ou bien d'un diviseur par 10.

La première section diviseur par 2 est constituée par la première bascule JK dont l'entrée d'horloge est notée «INPUT A» et la sortie «Q0». Les entrées J et K non câblés sur le schéma sont toutes rattachées au niveau logique H.

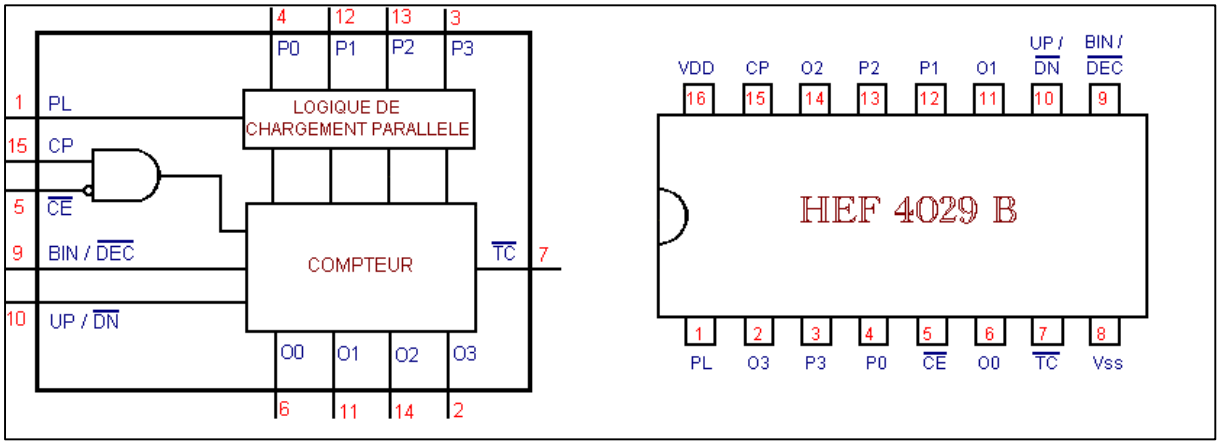
La deuxième section comporte les trois bascules JK suivantes. L'entrée d'horloge est notée «INPUT B» et les trois sorties sont Q1, Q2 et Q3. Cette troisième sortie Q3 délivre un signal divisé par 5 par rapport au signal d'horloge appliqué à l'entrée «INPUT B».

Ce compteur modulo 5 utilise le code binaire.

Pour obtenir un compteur modulo 10 en code BCD, il suffit de relier la sortie Q0 à l'entrée INPUT B. La sortie Q0 qui divise par deux la fréquence d'horloge commande elle-même la section diviseur par 5. Il est donc possible de recueillir un signal en sortie Q3 dont la fréquence est le 1 / 10 ème de celle de l'horloge.

- LE COMPTEUR INTÉGRÉ HEF 4029B

C'est un compteur / décompteur synchrone binaire / décimal 4 bits réalisé en technologie MOS. Son schéma fonctionnel et son brochage sont donnés à la figure suivante :



Le signal d'horloge est appliqué sur l'entrée CP. Ce sont les fronts montants qui sont actifs. $\overline{CE}$ est une entrée de validation. Si elle se trouve au niveau H, le compteur est inhibé ainsi que la retenue. PL est l'entrée de chargement parallèle asynchrone prioritaire. Dès qu'elle passe au niveau H, les quatre données présentes sur P0, P1, P2 et P3 sont transférées sur les sorties O0, O1, O2 et O3.

La commande UP / $\overline{DN}$ permet soit de compter (UP / $\overline{DN}$ au niveau H), soit de décompter (UP / $\overline{DN}$au niveau L).

La commande BIN / $\overline{DEC}$ permet le comptage / décomptage soit en code binaire (BIN / $\overline{DEC}$ au niveau H), soit en code décimal (BIN / $\overline{DEC}$au niveau L).

La sortie $\overline{TC}$ est normalement au niveau H et passe au niveau L lorsque le compteur atteint le compte maximal en mode comptage ou le compte minimal en mode décomptage à condition que $\overline{CE}$ soit au niveau L.

Le tableau suivant présente les différents modes de fonctionnement de ce compteur.

Tableau de fonctionnement du compteur **HEF 4029B.**					
PL	BIN / $\overline{DEC}$	UP / $\overline{DN}$	$\overline{CE}$	CP	MODE
H	X	X	X	X	Chargement parallèle
L	X	X	H	X	Sans changement
L	L	L	L	↑	décomptage décimal
L	L	H	L	↑	Comptage décimal
L	H	L	L	↑	Décomptage binaire
L	H	H	L	↑	Comptage binaire

7. Les registres

7-1 Définition :

Un **registre** est un circuit logique capable de **mémoriser une information (**un mot binaire de n bits) ou **transférer** cette information vers un autre circuit.

Un registre composé de **n bascules** en cascades et possède la capacité de stockage d'un mot binaire de n bits.

Un registre est caractérisé par :

- **La capacité**: nombre de bits du mot binaire qu'il peut mémoriser.
- **Le mode d'écriture ou de chargement**: dépend du nombre d'entrées :
  - *écriture série* : génération bit par bit, avec transmission par un seul fil conducteur.
  - *écriture parallèle* : génération globale du mot de n bits, avec transmission par un bus de n bits (n fils conducteurs).
- **Le mode de lecture**:
  - *lecture série* : exploitation bit par bit du mot (une seule sortie).
  - *lecture parallèle* : exploitation globale du mot (n sorties).

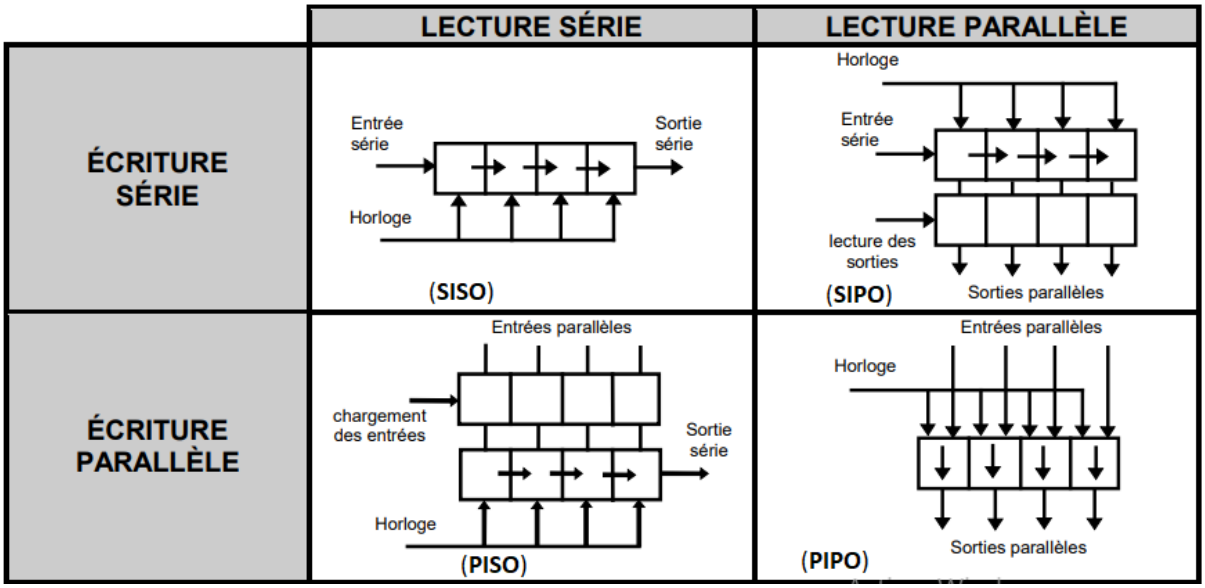
7-2 Types de registres :

Selon le mode d'accès en écriture (entrée) et en lecture (sortie), série ou parallèle, Il existe quatre types de registre

- Registre à mode d'écriture (ou chargement) parallèle et mode de lecture (ou sortie) parallèle (**PIPO**). Ce type de registre est utilisé pour mémoriser une donnée parallèle

à un instant ce qui permettra de l'exploiter ultérieurement.

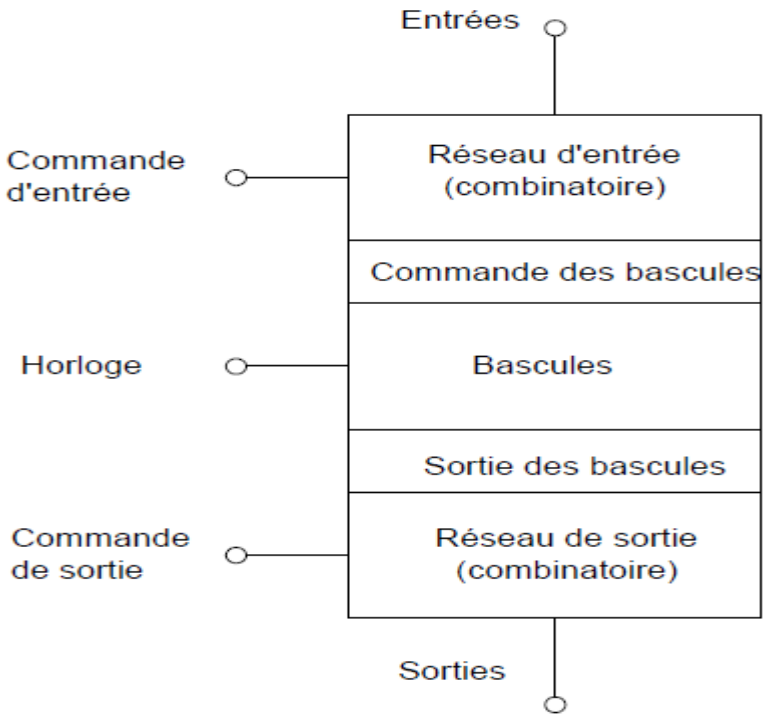
- Registre à mode d'écriture (ou chargement) parallèle et mode de lecture (ou sortie) série (**PISO)**. Ce type de registre permet de réaliser une conversion parallèle/série. Ils sont utilisés lors de la transmission de données séries.
- Registre à mode d'écriture (ou chargement) série et mode de lecture (ou sortie) parallèle (**SIPO)**. Ce type de registre permet de réaliser une conversion série/parallèle. Ils sont utilisés lors de la réception de données séries.
- Registre à mode d'écriture (ou chargement) série et mode de lecture (ou sortie) série (**SISO)**. Ce type de registre est utilisé pour mémoriser une donnée série à un instant ce qui permettra de l'exploiter ultérieurement.



Ces quatre types peuvent être classés en deux catégories :

- les **registres de mémorisation** (*tampon*)
- les **registres à décalage**.

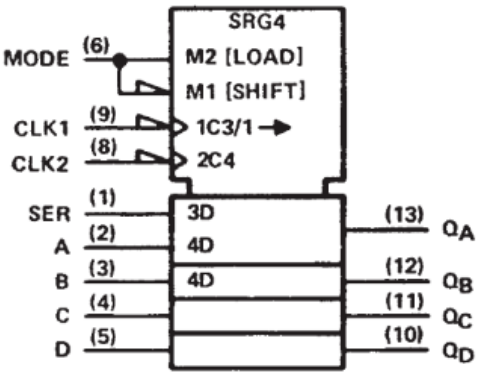
Le schéma fonctionnel d'un registre est donné par la figure suivante:



Un registre n bits comprend :

- n bascules qui peuvent être du type RSH, D, JK.
- Une entrée du signal d'Horloge pour la synchronisation.
- Une commande pour le changement des entrées, ou l'écriture.
- Une commande pour l'activation des sorties, ou la lecture.
- Une commande pour le décalage interne des bits du mot mémorisé à droite ou à gauche.

Exemple : Registre à décalage ayant pour référence 7495



MODE SELECT — TRUTH TABLE

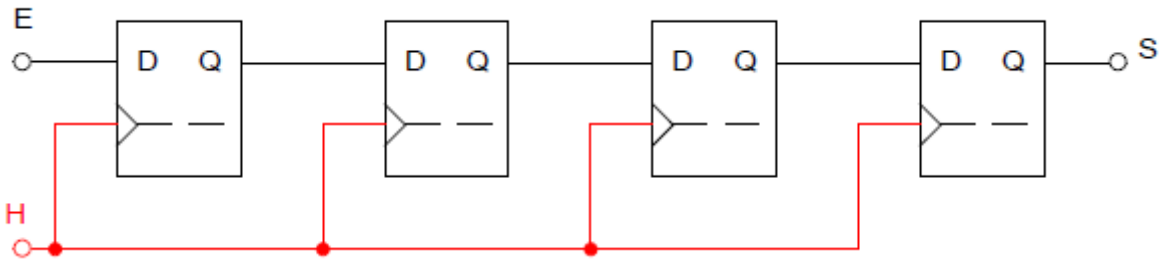
OPERATING MODE	INPUTS					OUTPUTS			
	S	CP_1	$\overline{CP_2}$	D_S	P_n	Q_0	Q_1	Q_2	Q_3
Shift	L	↓	X	l	X	L	q_0	q_1	q_2
	L	↓	X	h	X	H	q_0	q_1	q_2
Parallel Load	H	X	↓	X	P_n	P_0	P_1	P_2	P_3
Mode Change	↓	L	L	X	X	No Change			
	↑	L	L	X	X	No Change			
	↓	H	L	X	X	No Change			
	↑	H	L	X	X	Undetermined			
	↓	L	H	X	X	Undetermined			
	↑	L	H	X	X	No Change			
	↓	H	H	X	X	Undetermined			
	↑	H	H	X	X	No Change			

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
l = LOW Voltage Level one set-up time prior to the HIGH to LOW clock transition.
h = HIGH Voltage Level one set-up time prior to the HIGH to LOW clock transition.
P_n = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

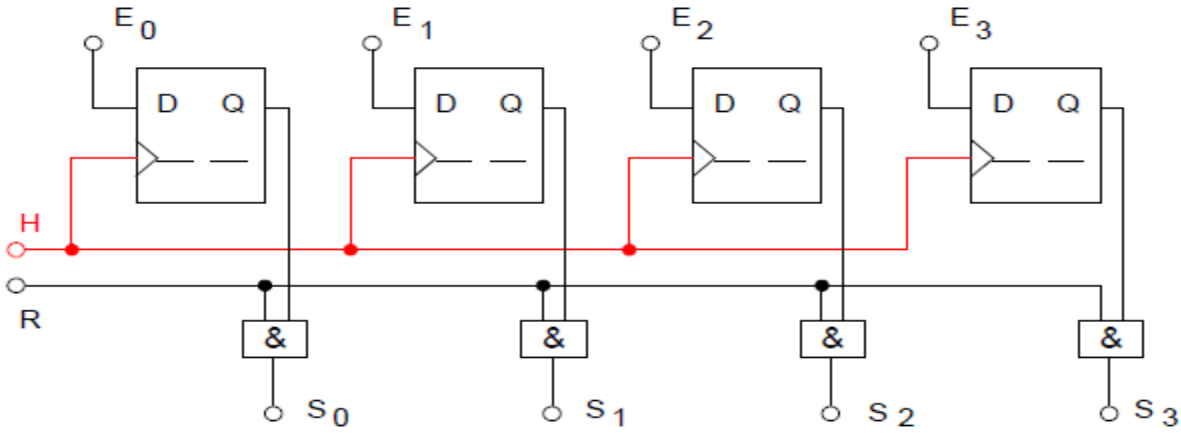
7-3 Réalisation d'un registre à base de bascules :

Le nombre de bascules est égal au nombre de bits.

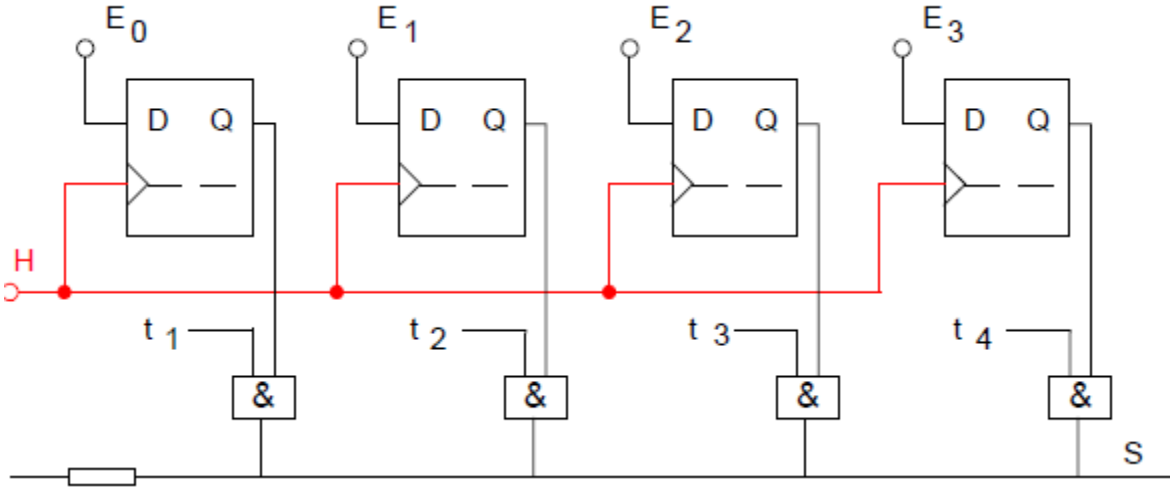
- **Registre SISO 4 bits** réalisés à partir de 4 bascules D



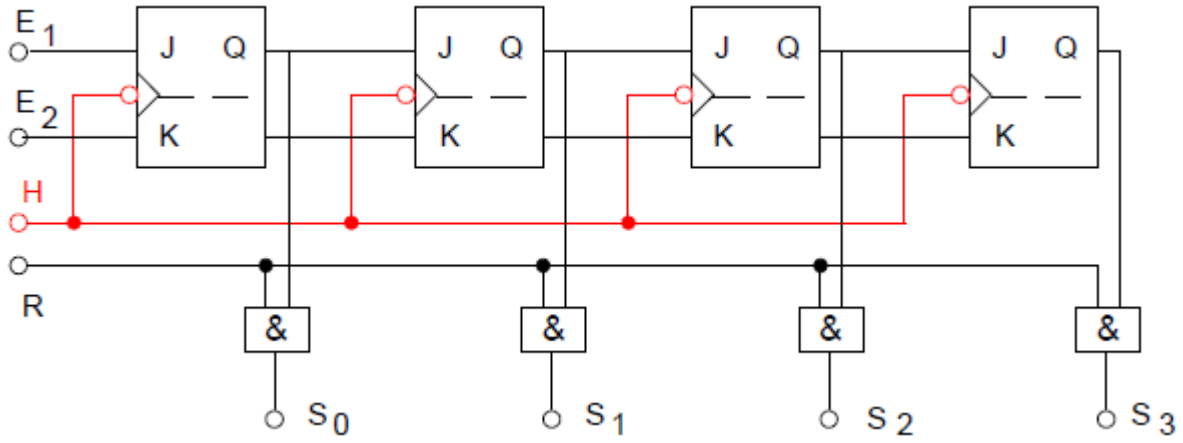
- **Registre PIPO 4 bits** réalisés à partir de 4 bascules D



- **Registre PISO 4 bits** réalisés à partir de 4 bascules D



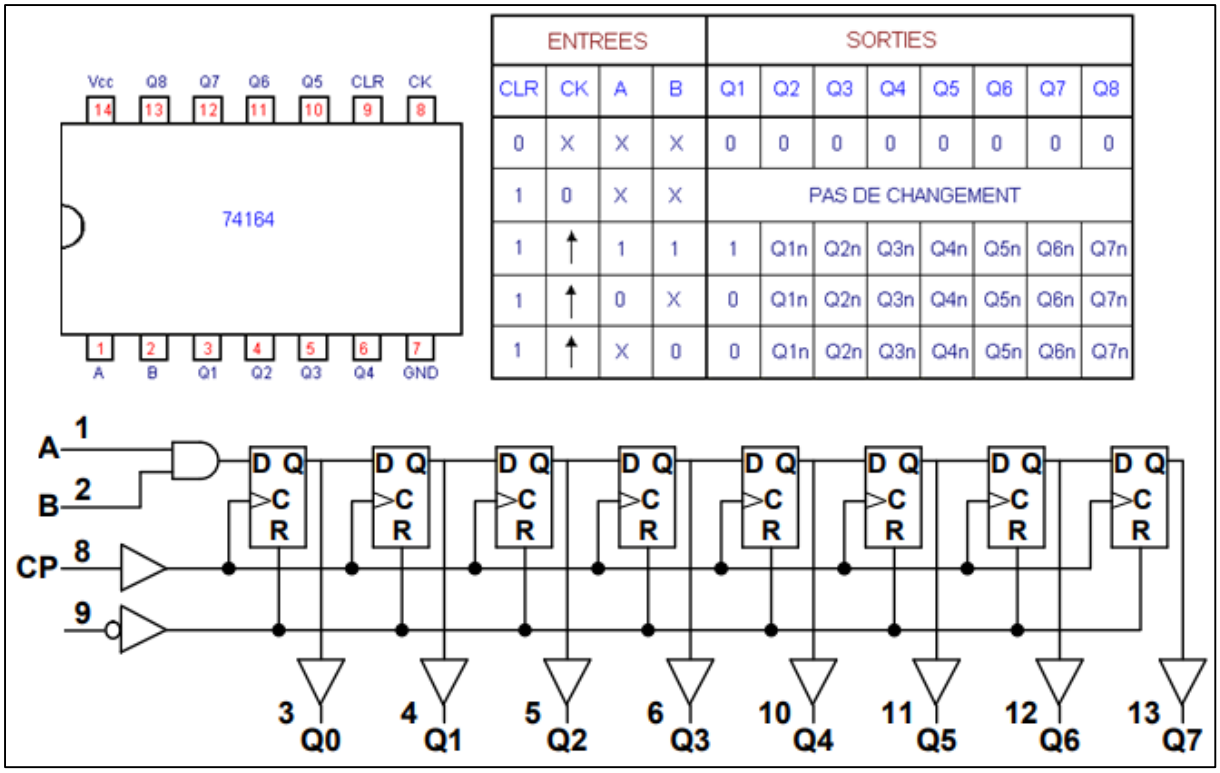
- **Registre SIPO 4 bits** réalisés à partir de 4 bascules JK



7-4 Exemples de circuits intégrés :

- REGISTRE SERIE PARALLELE INTÉGRÉ : LE 74164

Le circuit intégré 74164 est un registre à décalage à deux entrées séries et huit sorties parallèles ayant une entrée d'horloge (CK) et une entrée asynchrone de remise à zéro générale prioritaire (CLR). Le brochage de ce circuit et sa table de vérité est donné à la figure suivante :



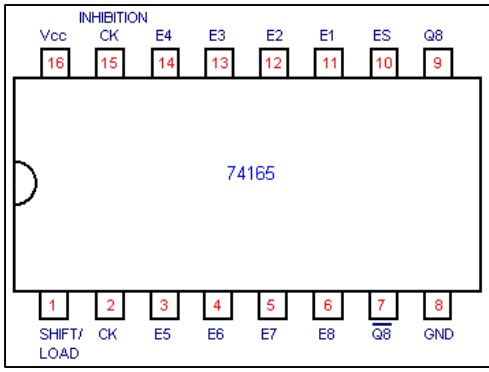
ENTREES				SORTIES							
CLR	CK	A	B	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
0	X	X	X	0	0	0	0	0	0	0	0
1	0	X	X	PAS DE CHANGEMENT							
1	↑	1	1	1	Q1n	Q2n	Q3n	Q4n	Q5n	Q6n	Q7n
1	↑	0	X	0	Q1n	Q2n	Q3n	Q4n	Q5n	Q6n	Q7n
1	↑	X	0	0	Q1n	Q2n	Q3n	Q4n	Q5n	Q6n	Q7n

- REGISTRE PARALLÈLE - SÉRIE ASYNCHRONE INTÉGRÉ : LE 74165

Le circuit intégré 74165 est un registre à décalage 8 bits à une entrée série (ES) et une sortie (Q8) permettait de pré-positionner son contenu de façon asynchrone. Il possède huit

entrées parallèles (E1 à E8), une entrée de commande de décalage et chargement asynchrone (SHIFT / LOAD), une entrée d'horloge (CK) et une entrée d'inhibition (CK INHIBIT). Il est à noter que ces deux entrées CK et CK INHIBIT sont interchangeables.

Le brochage de ce circuit intégré et sa table de vérité sont donnés à la figure suivante :

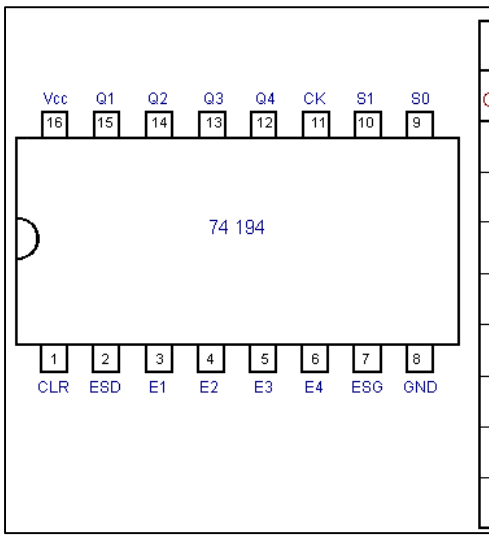


ENTREES												Sorties internes		Sorties Externes	
SHIFT/ LOAD	CK INHIBIT	CK	E8	E1	E2	E3	E4	E5	E6	E7	E8	Q1	Q2	Q8	$\overline{Q8}$
0	X	X	X	E1	E2	E3	E4	E5	E6	E7	E8	E1	E2	E8	$\overline{E8}$
1	0	0	X	X	X	X	X	X	X	X	X	Pas de changement			
1	0	↑	1	X	X	X	X	X	X	X	X	1	Q1n	Q7n	$\overline{Q7n}$
1	0	↑	0	X	X	X	X	X	X	X	X	0	Q1n	Q7n	$\overline{Q7n}$
1	1	X	X	X	X	X	X	X	X	X	X	Pas de changement			

- <u>REGISTRE INTÉGRÉ UNIVERSEL : LE 74 194</u>

Le circuit intégré 74 194 est un registre à décalage bidirectionnel 4 bits ayant deux entrées de commande (S0 et S1), une entrée d'horloge (CK), une entrée de données série pour le décalage à gauche (ESG), une entrée de données série pour le décalage à droite (ESD), quatre entrées parallèles (E1 à E4), une entrée asynchrone de remise à zéro générale prioritaire (CLR) et quatre sorties parallèles (Q1 à Q4).

Le brochage de ce circuit intégré et sa table de vérité sont donnés à la figure suivante :



ENTREES										SORTIES			
CLR	S1	S0	ESG	ESD	E1	E2	E3	E4	CK	Q1	Q2	Q3	Q4
0	X	X	X	X	X	X	X	X	X	0	0	0	0
1	X	X	X	X	X	X	X	X	0	PAS DE CHANGEMENT			
1	1	1	X	X	E1	E2	E3	E4	↑	E1	E2	E3	E4
1	0	1	X	1	X	X	X	X	↑	1	Q1n	Q2n	Q3n
1	0	1	X	0	X	X	X	X	↑	0	Q1n	Q2n	Q3n
1	1	0	1	X	X	X	X	X	↑	Q2n	Q3n	Q4n	1
1	1	0	0	X	X	X	X	X	↑	Q2n	Q3n	Q4n	0
1	0	0	X	X	X	X	X	X	X	PAS DE CHANGEMENT			

8. Les mémoires

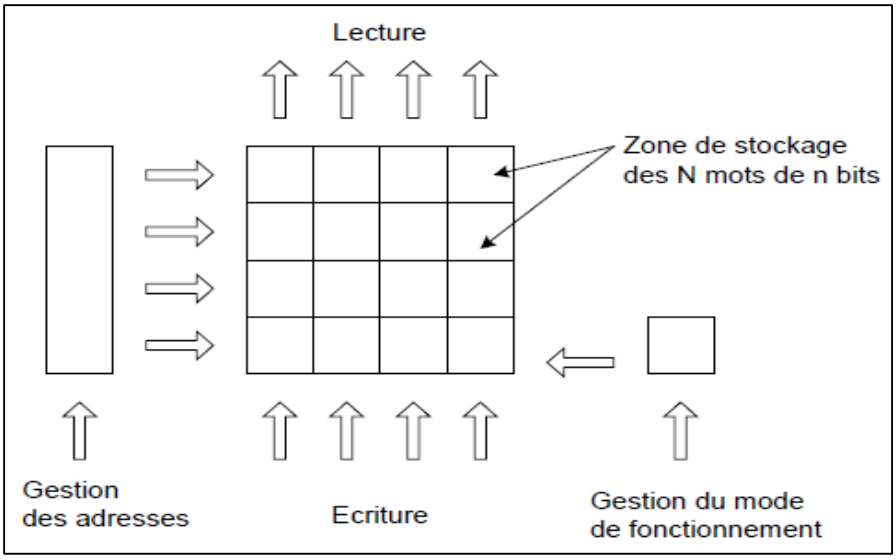
8-1 Définition :

Une mémoire permet de stocker et de restituer une très grande quantité d'informations correspondant à N mots de bits.

Une mémoire intégrée est une association de registres qui ont chacun une adresse bienprécise dans la mémoire.

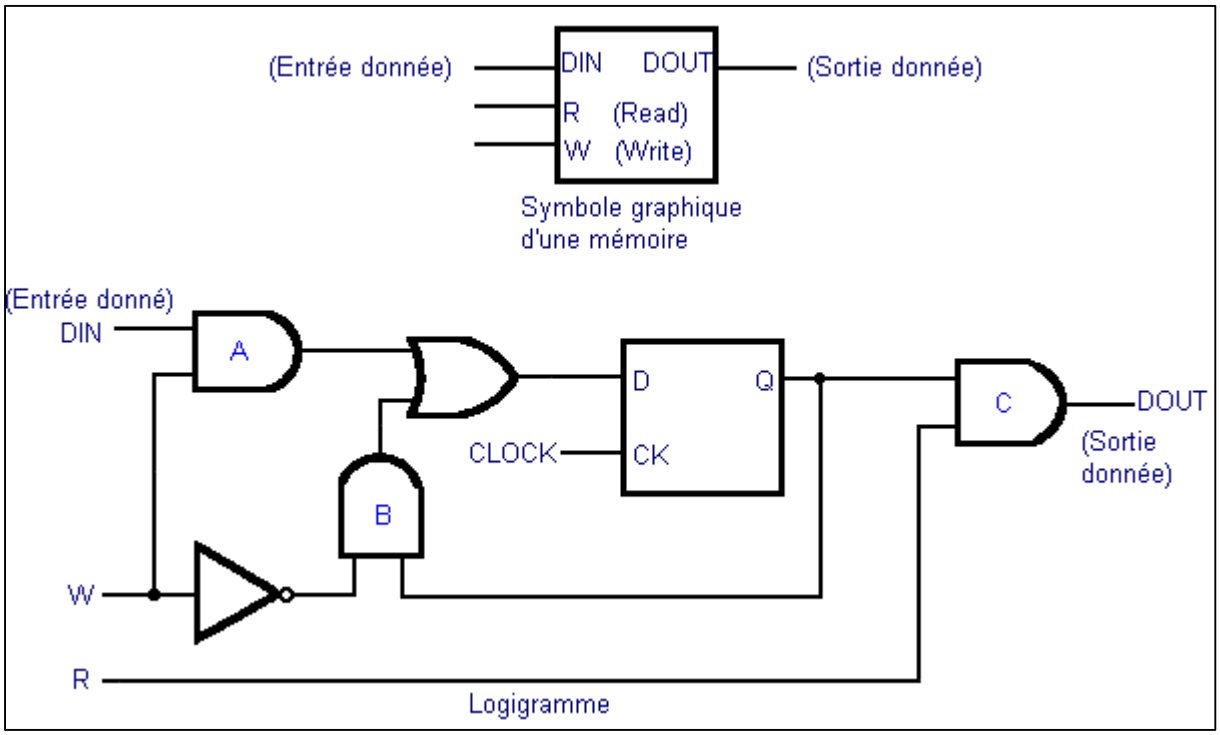
Une mémoire comprend :

- une zone pour le stockage des mots.
- Un circuit pour la gestion des adresses.
- Un ensemble de circuits pour la gestion de fonctionnement
  - écriture (write)
  - lecture (read).



8-3 Mémoire électronique élémentaire :

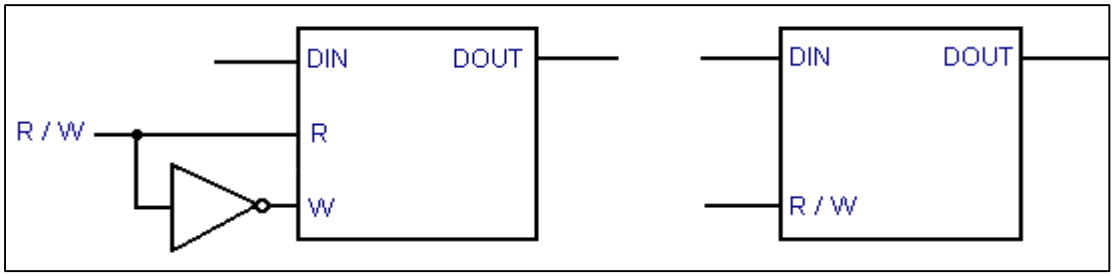
La cellule élémentaire d'une mémoire électronique est essentiellement constituée d'une bascule dotée d'un réseau combinatoire extérieur tel qu'il permette l'enregistrement et la lecture des données.



Une entrée pour les données (**DIN**), une autre pour prédisposer la mémoire à l'écriture (**W**) et une troisième pour la prédisposer à la lecture (**R**) ; la sortie est repérée par le symbole **DOUT**. Les données 0 ou 1 sont écrites dans la bascule lorsque l'entrée W est haute, car ainsi leur passage en mémoire à travers la porte A est validé. Le symbole W est l'initiale de «Write» qui signifie écrire.
Si par contre, l'entrée W est au niveau L, la porte A est bloquée et la porte B passante. De cette façon, la sortie Q est reliée à l'entrée D de la bascule. De ce fait, chaque fois qu'une impulsion d'horloge arrive, le contenu de la bascule ne se perd pas car il est réinscrit à travers l'entrée D.

La donnée présente sur Q est lue lorsque l'entrée R est au niveau haut, ce qui valide la porte C et permet au niveau disponible en Q de s'afficher sur la sortie DOUT. La lettre R est l'initiale de «Read» qui signifie lire.

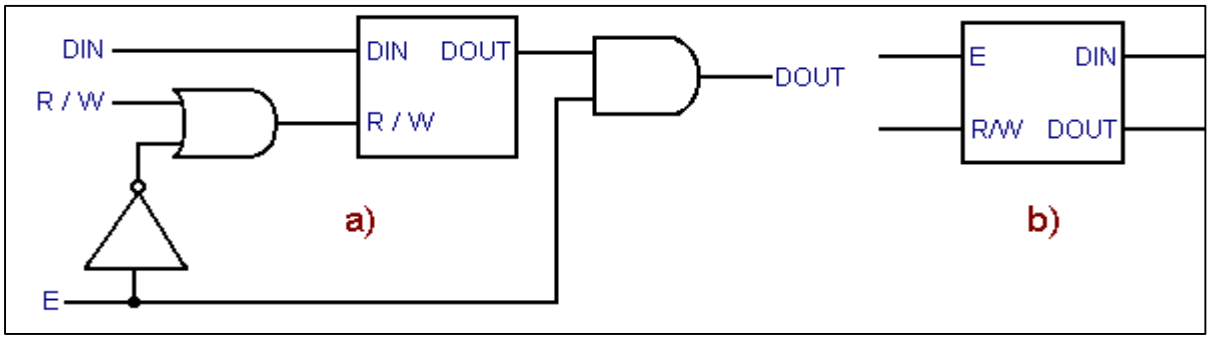
Afin de différencier les deux opérations (W/R), on peut utiliser un inverseur tel que celui représenté figure suivante :



De cette façon, on obtient une borne **R / W (Read / Write)** qui autorisera l'écriture

lorsqu'elle sera à 0 et la lecture lorsqu'elle sera à 1.

Afin de réduire le nombre de pattes du circuit intégré, on modifie la cellule élémentaire de mémoire comme indiqué sur la figure suivante, en faisant apparaître une borne de validation E :

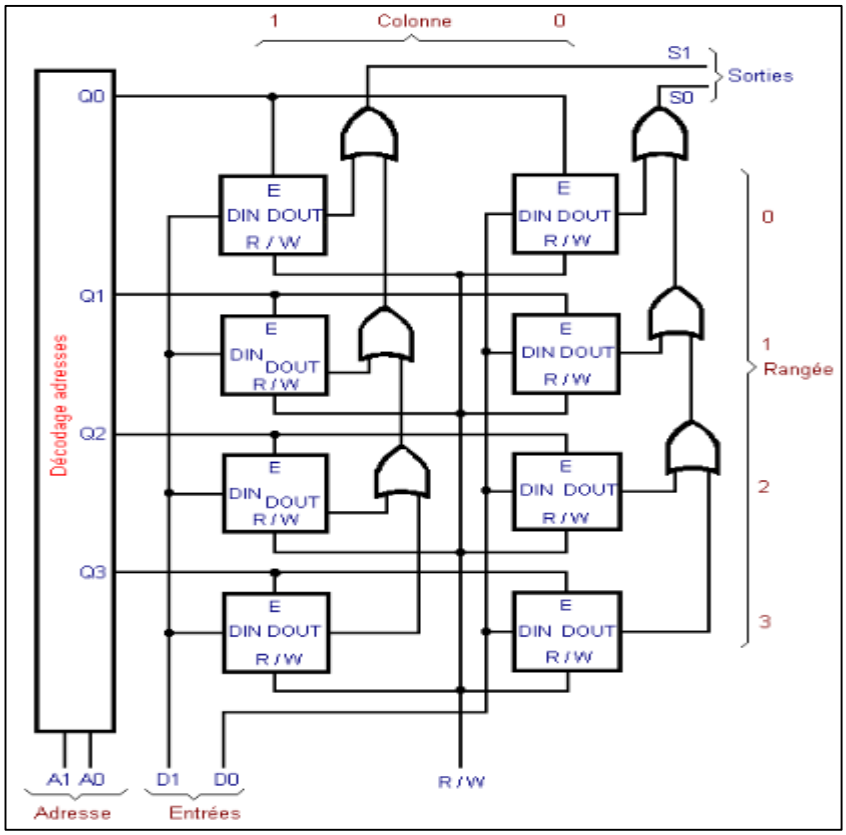


8-3 Mémoire électronique à plusieurs bits :

- **Mémoire électronique quatre mots de deux bits.**

Huit bits regroupés deux à deux en quatre groupes (mots) et qu'il est possible de lire simultanément deux bits.

Les entrées A1 et A0 d'adresse sélectionnent, grâce à un décodeur, la rangée dans laquelle il faut lire ou écrire.



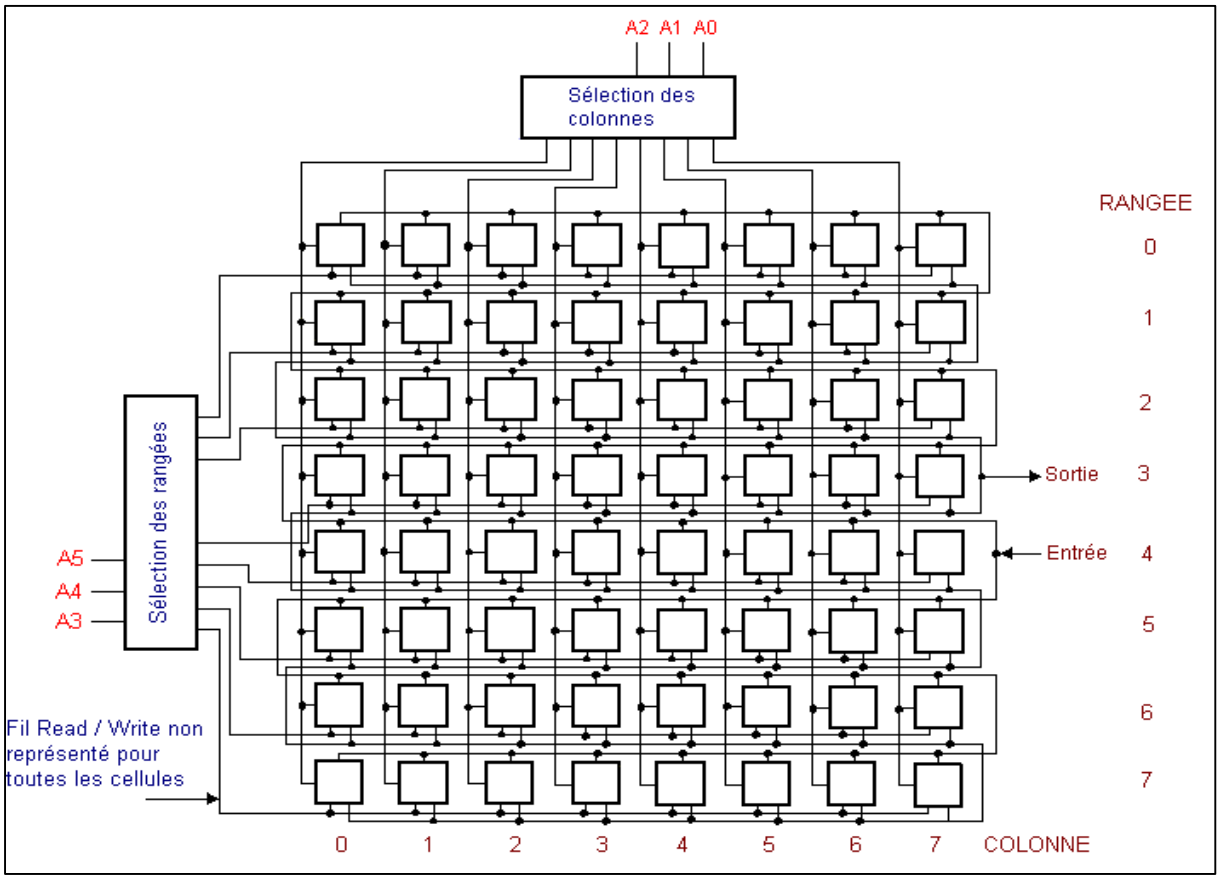
- **<u>Mémoire électronique 64 bits, chacune étant accessible individuellement</u>**

Il existe des mémoires où l'adresse ne définit la position que d'un seul bit ; dans ce cas, il est nécessaire que l'adresse indique également la colonne ; donc, outre le décodeur qui sélectionne les rangées, il est alors nécessaire de disposer d'un décodeur de colonnes. De plus, chaque cellule, nonobstant une entrée de validation pour la rangée, disposera d'une entrée de validation pour la colonne. Lorsque les deux seront actives, on pourra alors lire ou écrire.

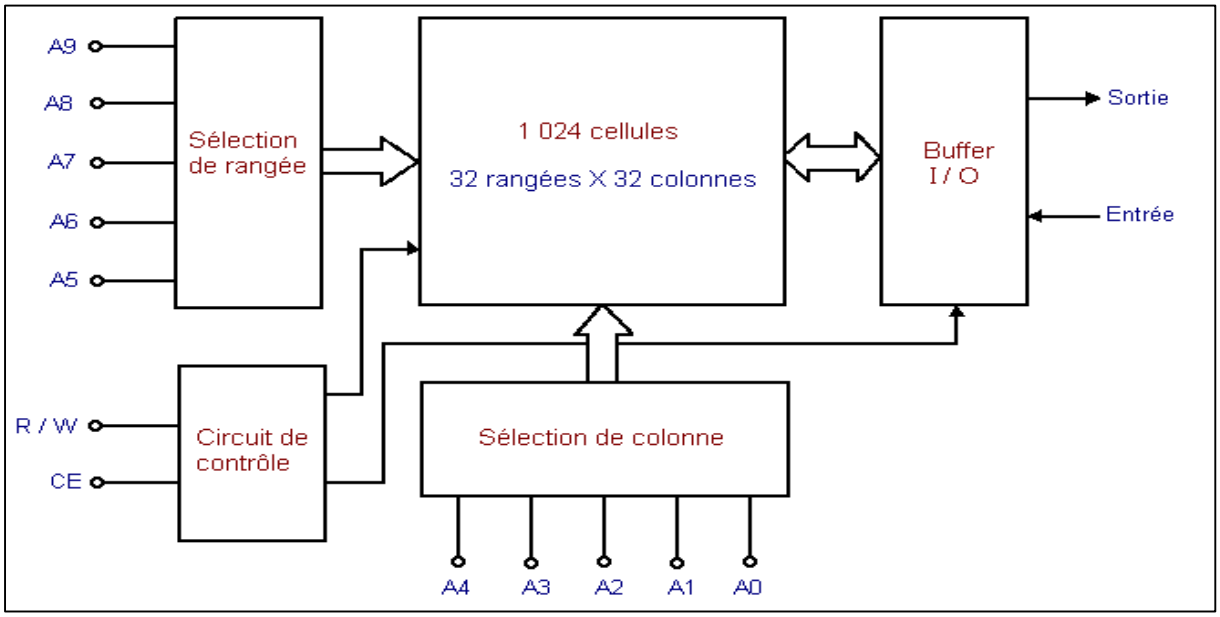
La figure suivante présente un exemple de mémoire à 64 cellules ou bits, chacune étant accessible individuellement.

Ici, on lit ou on écrit un **seul bit à la fois**. Il est donc nécessaire de disposer d'une adresse à six chiffres ; en effet, puisqu'il y a 64 cellules, 64 combinaisons différentes sont nécessaires et il faut six chiffres (**$2^6 = 64$**) pour obtenir ce nombre de combinaisons.

Les trois premiers bits de l'adresse, de A0 à A2, repèrent la colonne ; les trois autres bits de A3 à A5 indiquent la rangée ou ligne horizontale.



Pour représenter une mémoire, on utilise donc habituellement des schémas synoptiques encore plus synthétiques, comme celui de la figure ci-après où toutes les cellules ne sont pas représentées une à une mais remplacées par un rectangle (il s'agit de 32 rangées et de 32 colonnes, soit 1 024 cellules qu'il aurait sinon fallu représenter).



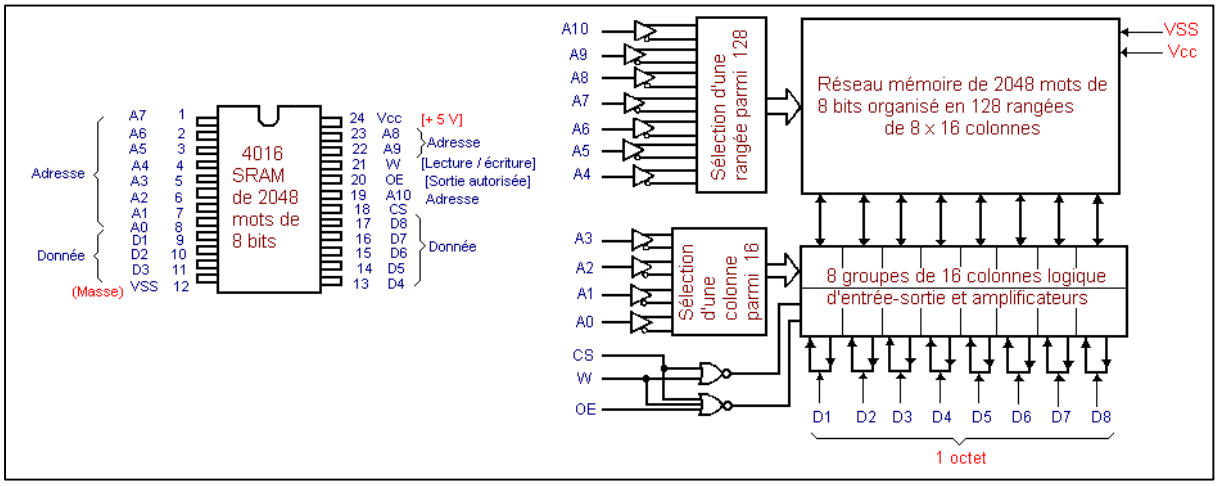
8-4 Familles de mémoires :

Il existe deux familles de mémoires :

- Les mémoires vives (**RAM** : Random Acces Memory) à lecture et écriture possible.
- Les mémoires mortes (**ROM** : Read Only Memory) à lecture seulement.

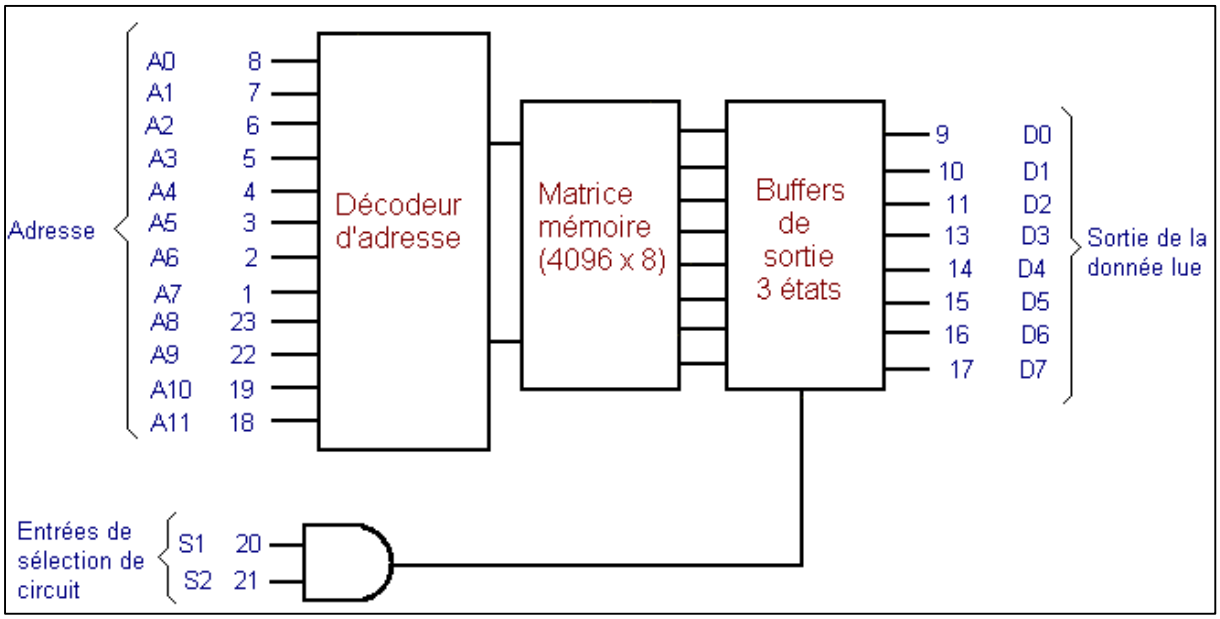
- **EXEMPLE DE MÉMOIRE RAM**

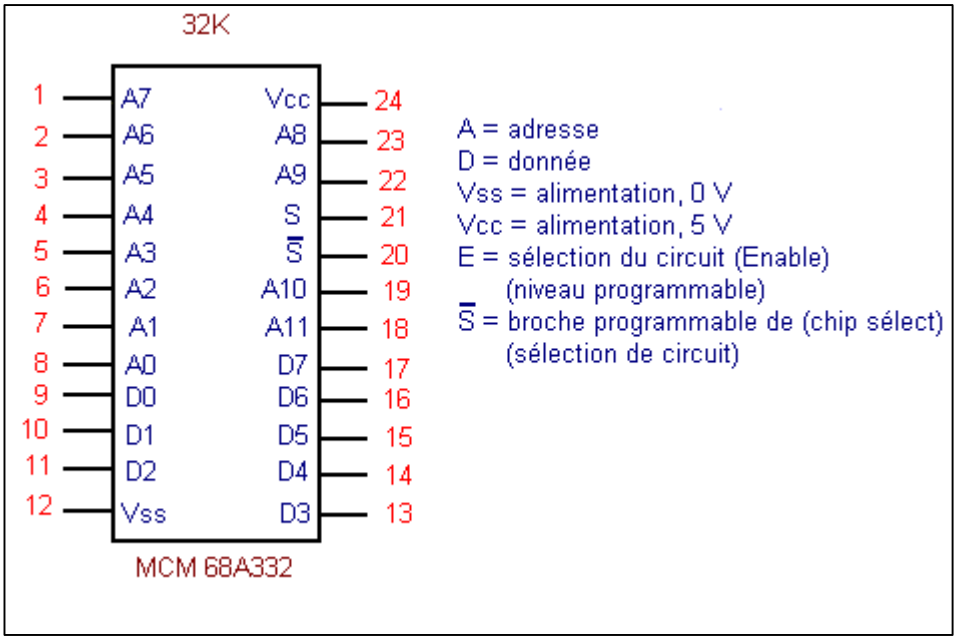
La figure suivante montre le brochage et le schéma synoptique d'une mémoire RAM statique de type 4016 de Texas Instruments de 2K mots de 1 octet (8 bits) :



- **EXEMPLE DE MÉMOIRE ROM**

La figure suivante montre le brochage et le schéma synoptique d'une mémoire ROM de type MCM 68 A332 de 32K mots de 1 octet (8 bits) :





9. Schéma de câblage de quelques applications

9-1 Schéma de câblage des compteurs /décompteurs

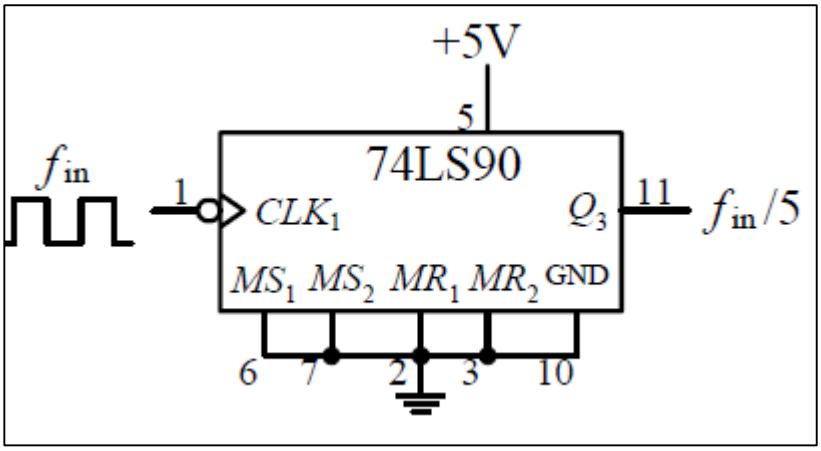
Il existe de nombreuses applications des compteurs.

Nous pouvons citer le comptage d'objets, la mesure du temps, la division du temps pour l'obtention de signaux d'horloge permettant la commande des systèmes synchronisés.

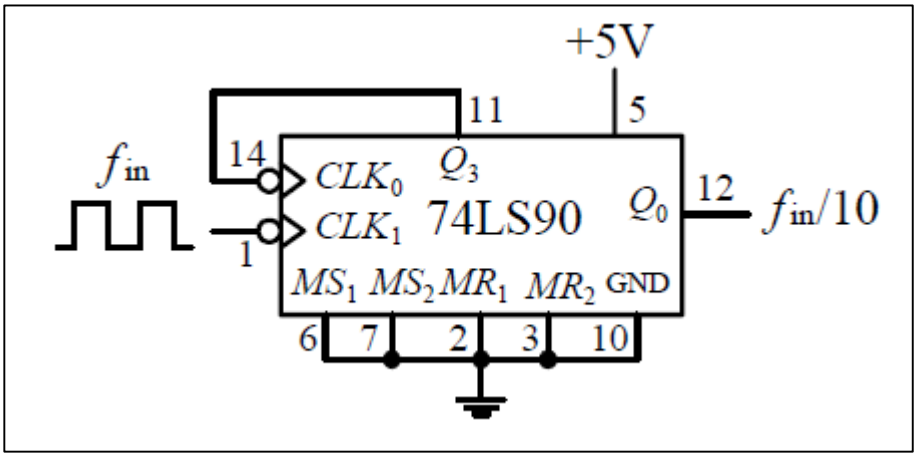
- **Diviseur de fréquence par N (2, 5 et 10)**

Mode CTR DIV 2 : Entrée du signal d'horloge sur CKA, sortie sur QA
Mode CTR DIV 5 : Entrée du signal d'horloge sur CKB, sorties sur QB à QD.
Mode CTR DIV 10 - mode BCD : Entrée du signal d'horloge sur CKA, sortie QA reliée à CKB, sorties QA à QD.

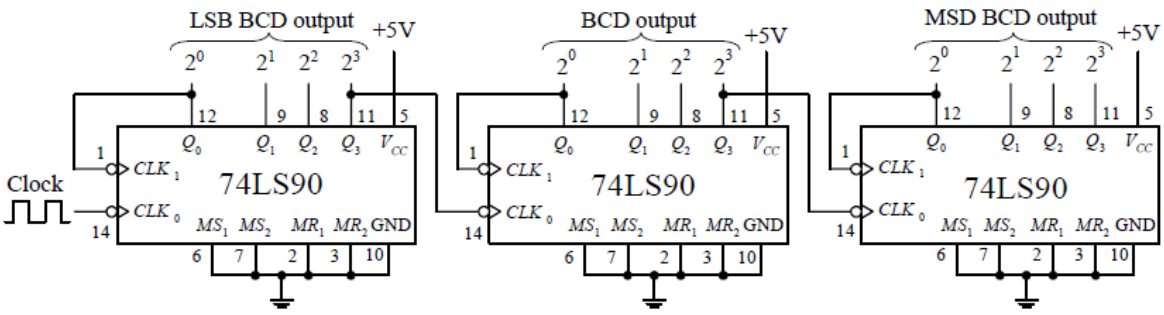
- Diviseur de fréquence par 5 :



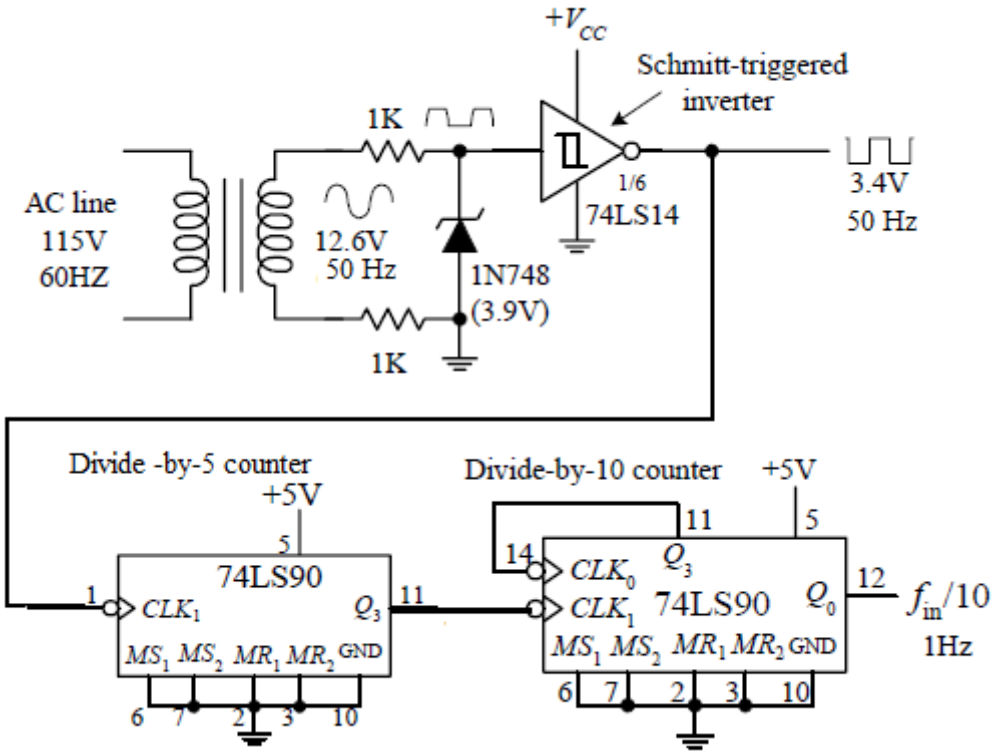
- Diviseur de fréquence par 10 :



- **Compteur BCD (000 …999)**

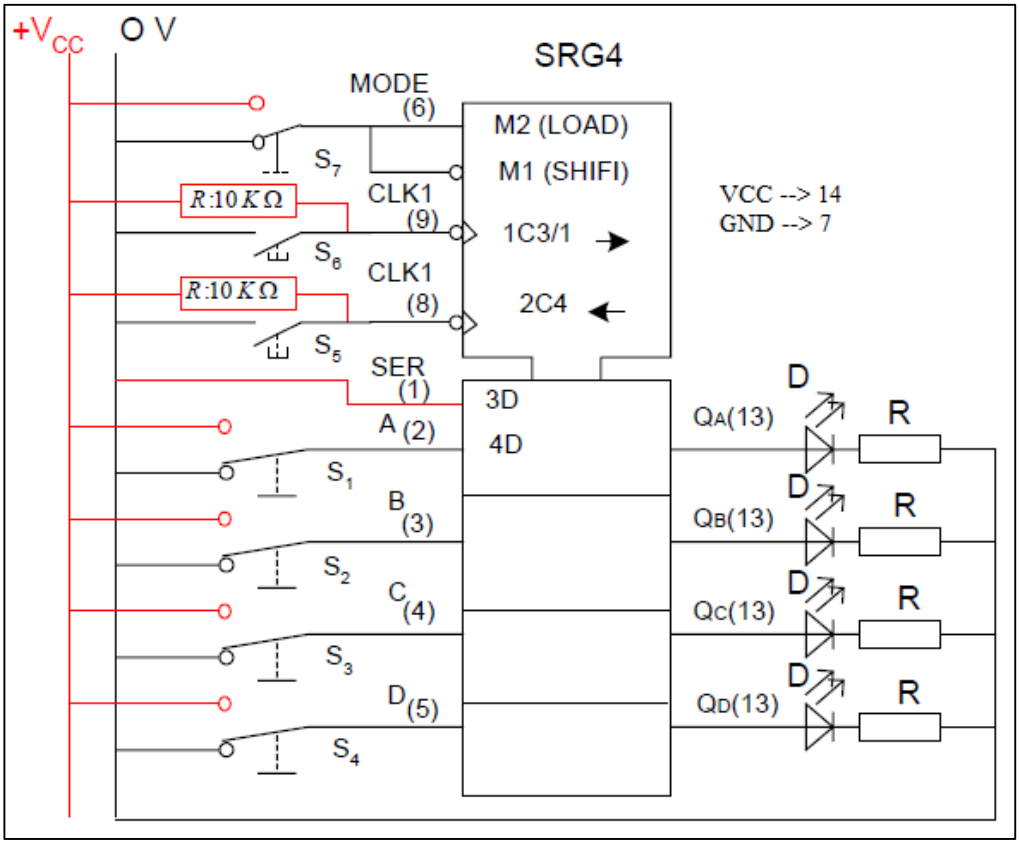


- **Générateur d'horloge (50 Hz, 10Hz, 1 Hz)**



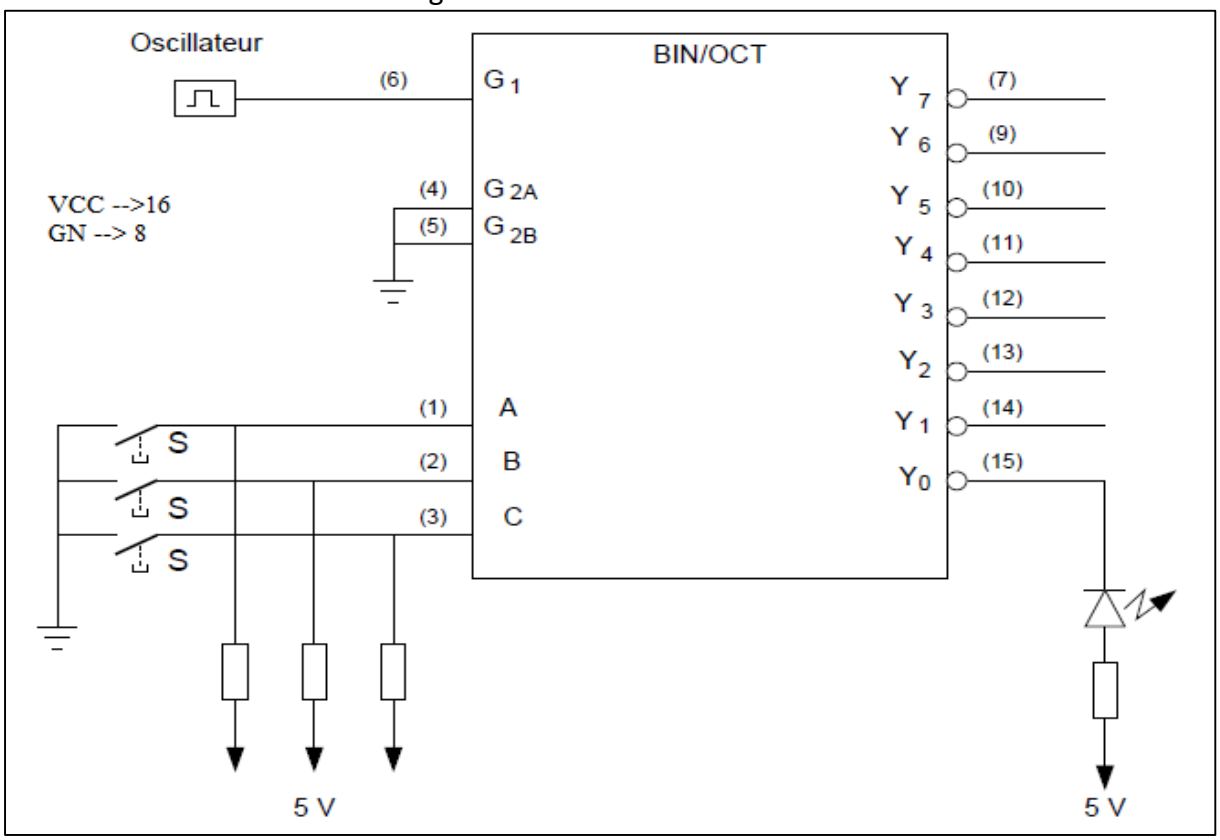
9-2 Schéma de câblage des registres

- Schéma de câblage du C.I 7495



9-3 Schéma de câblage d'un décodeur

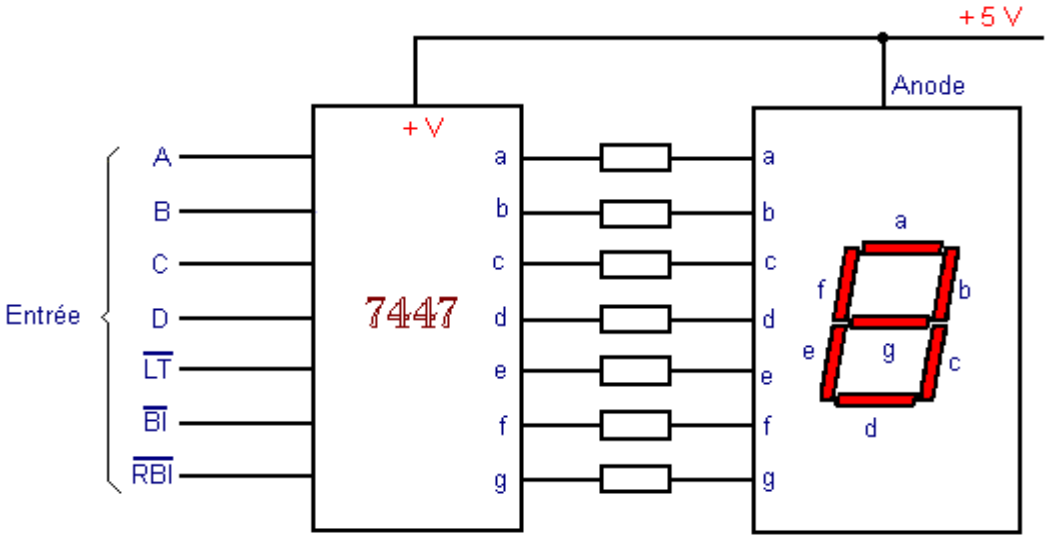
- Schéma de montage du décodeur 74LS138



Le signal d'Horloge (oscillateur) valide le décodeur par G1 et les capteurs S1, S2, S3, (sélection) désignent la sortie qui oscillera au rythme du générateur d'impulsion.

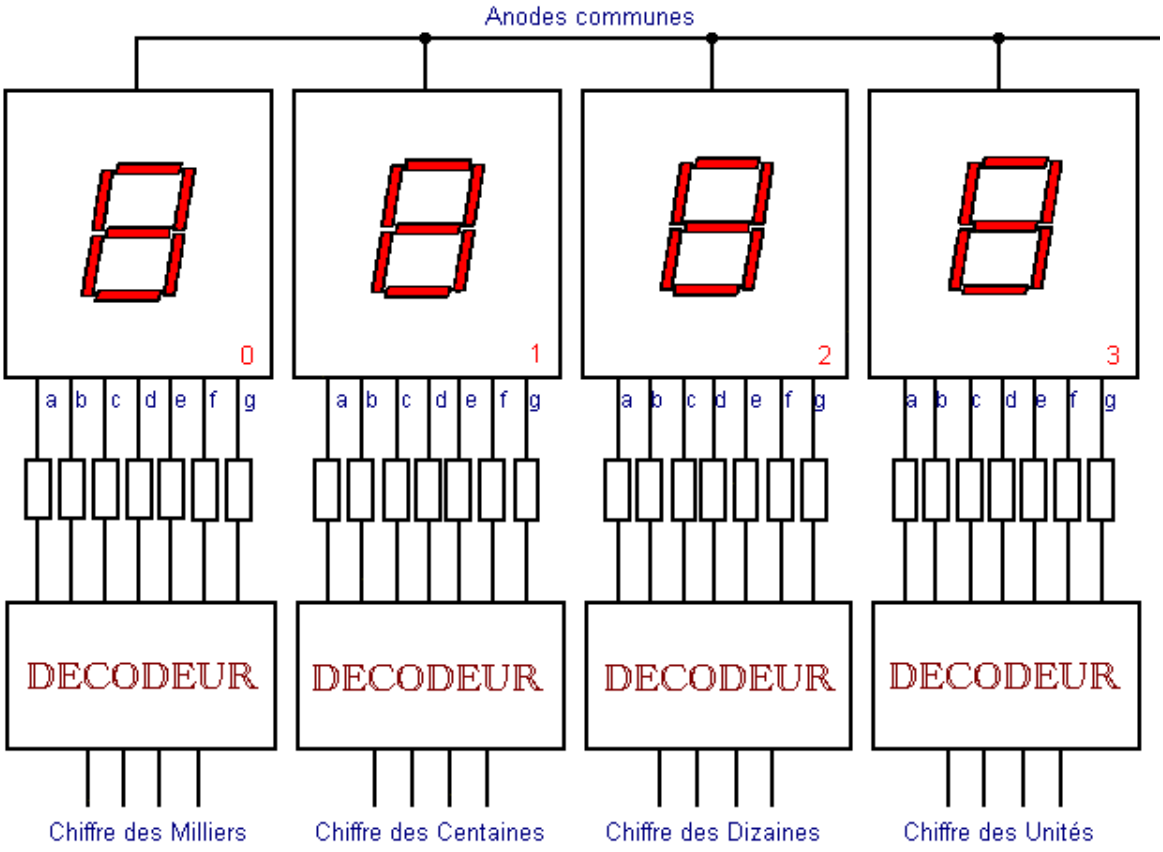
9-4 Schéma de câblage de différents afficheurs

- Schéma de montage d'un décodeur 7447 commandant un afficheur 7 segments



	Entrées							Sorties						
Décimal ou fonction	LT	RBI	D	C	B	A	BI / RBO	a	b	c	d	e	f	g
0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
1	1	X	0	0	0	1	1	1	0	0	1	1	1	1
2	1	X	0	0	1	0	1	0	0	1	0	0	1	0
3	1	X	0	0	1	1	1	0	0	0	0	1	1	0
4	1	X	0	1	0	0	1	1	0	0	1	1	0	0
5	1	X	0	1	0	1	1	0	1	0	0	1	0	0
6	1	X	0	1	1	0	1	1	1	0	0	0	0	0
7	1	X	0	1	1	1	1	0	0	0	1	1	1	1
8	1	X	1	0	0	0	1	0	0	0	0	0	0	0
9	1	X	1	0	0	1	1	0	0	0	1	1	0	0
10	1	X	1	0	1	0	1	1	1	1	0	0	1	0
11	1	X	1	0	1	1	1	1	1	0	0	1	1	0
12	1	X	1	1	0	0	1	1	0	1	1	1	0	0
13	1	X	1	1	0	1	1	0	1	1	0	1	0	0
14	1	X	1	1	1	0	1	1	1	1	0	0	0	0
15	1	X	1	1	1	1	1	1	1	1	1	1	1	1
BI	X	X	X	X	X	X	0	1	1	1	1	1	1	1
RBI	1	0	0	0	0	0	0	1	1	1	1	1	1	1
LT	0	X	X	X	X	X	1	0	0	0	0	0	0	0

- Schéma de montage d'un afficheur 7 segments pour 4 chiffres


Anodes communes
0
1
2
3
a b c d e f g
DECODEUR
DECODEUR
DECODEUR
DECODEUR
Chiffre des Milliers
Chiffre des Centaines
Chiffre des Dizaines
Chiffre des Unités

9-5 Quelques références de circuits intégrés

Fonction du circuit	**Circuits intégrés correspondants**
Décodeur 3 vers 8 ou décodeur binaire – octal	74LS138
Décodeur binaire – 7 segments	74S49
Décodeur BCD – 7 segments	74LS47
Multiplexeur Double sélecteur – multiplexeur 4 vers 4 avec sorties3 états	74LS253
Démultiplexeur Double décodeur – Démultiplexeur 2 vers 4	74155
Compteur – décompteur binaire synchrone programmable 4 bits avec 2 horloges et RAZ	74LS193 OU 40193
Compteurs – décompteur décimaux synchroneprogrammables	74LS190
Compteur à décade	4017
Compteur décimal	74LS90
Compteur décimal synchrone programmable	74LS160
Registre à décalage 4 bit shift registers	7494
Registre à décalage 8bit parallèle out serial shiftregisters	74164, 74L164, 74LS164
Registre à décalage 4 bits PIPPO	7495

Bibliographie

[1] **Sequential and Arithmetic Logic Circuits**, Tertulien Ndjountche, Wiley 2016

[2] **Digital Electronics 1 Combinational Logic Circuits**, Tertulien Ndjountche, Wiley 2016

[3] **Digital Design Principles and Practices**, John F. Wakerly, Prentice Hall (2005)

[4] **Digital Electronics with VHDL (Quartus II Version)**, William Kleitz, Pearson (2013)

https://www.electronique-et-informatique.fr/Electronique-et-Informatique/

https://matheleve.tn/wp-content/uploads/2021/03/

Chapitre IV

TRAVAUX DIRIGES / AUTOEVALUATION

1- Travaux dirigés

Exercice N° 1 : Donner l'équivalent décimal des nombres octaux suivants :

a/ $(72)_8$ = (… .....)$_{10}$
b/ $(1251)_8$ = (… .....)$_{10}$
c/ $(17,3)_8$ = (.....)$_{10}$
d/ $(512,65)_8$ = (… .....)$_{10}$

Exercice N°2 : Donner l'équivalent octal des nombres décimaux suivants :

a/ $(96)_{10}$ = (… .....)$_8$
b/ $(19,25)_{10}$ = (… .....)$_8$
c/ $(728,5)_{10}$ = (… .....)$_8$
d/ $(129)_{10}$ = (… .....)$_8$

Exercice N° 3 : Donner l'équivalent octal des nombres binaires suivants :

a/ $(11)_2$ = (… .....)$_8$
b/ $(10110)_2$ = (.....)$_8$
c/ $(100011011000,1101)_2$ = (… .....)$_8$
d/ $(11111101101)_2$ = (… .....)$_8$

Exercice N°4 : Trouver l'équivalent binaire des nombre octaux suivants :

a/ $(5)_8$ = (.....)$_2$
b/ $(63)_8$ = (… .....)$_2$
c/ $(674)_8$ = (… .....)$_2$
d/ $(152)_8$ = (… .....)$_2$

Exercice N° 5 : Convertir les nombres hexadécimaux suivants en décimale :

a/ $(18)_{16}$ = (.....)$_{10}$
b/ $(A2)_{16}$ = (… .....)$_{10}$
c/ $(FEE)_{16}$ = (… .....)$_{10}$
d/ $(AC,2)_{16}$ = (… .....)$_{10}$

Exercice N° 6 : Trouver l'équivalent hexadécimal des nombres décimaux suivants :

a/ $(72)_{10}$ = (… .....)$_{16}$
b/ $(86,31)_{10}$ = (… .....)$_{16}$
c/ $(122)_{10}$ = (… .....)$_{16}$
d/ $(716,40)_{10}$ = (… .....)$_{16}$

Exercice N° 7 : Donner l'équivalent hexadécimal des nombres binaires suivants :

a/ $(101)_2$ = (.....)$_{16}$
b/ $(11011)_2$ = (... .....)$_{16}$
c/ $(101110111,1100)_2$ = (.....)$_{16}$
d/ $(111101111,1011101)_2$ = (.....)$_{16}$

Exercice N° 8 : Donner l'équivalent binaire des nombres hexadécimaux suivants :

a/ $(18)_{16}$ = (.....)$_2$
b/ $(A2)_{16}$ = (... .....)$_2$
c/ $(CAFE)_{16}$ = (... .....)$_2$
d/ $(A25,5E)_{16}$ = (.....)$_2$

Exercice N° 9 : Donner l'équivalent BCD des nombres décimaux suivants :

a/ $(8)_{10}$ = (.....)$_{BCD}$
b/ $(17)_{10}$ = (.....)$_{BCD}$
c/ $(128)_{10}$ = (.....)$_{BCD}$
d/ $(92)_{10}$ = (.....)$_{BCD}$

Exercice N°10 : Trouver l'équivalent décimal des codes BCD suivants :

a/ $(101)_{BCD}$ = (.....)$_{10}$
b/ $(110100)_{BCD}$ = (.....)$_{10}$
c/ $(10100110111)_{BCD}$ = (.....)$_{10}$
d/ $(1000100111)_{BCD}$ = (.....)$_{10}$

Exercice N° 11 : Faites la conversion binaire – décimal des nombres fractionnaires suivants :

a/ $(1,1)_2$ = (.....)$_{10}$
b/ $(10,1011)_2$ = (.....)$_{10}$

c/ $(111,111101)_2$ = (.....)$_{10}$

d/ $(1011,00101)_2$ = (.....)$_{10}$

Exercice N° 12 : Faites la conversion décimal – binaire des nombres suivants :

a/ $(12,5)_{10}$ = (... .....)$_2$
b/ $(154,75)_{10}$ = (.....)$_2$
c/ $(26)_{10}$ = (.....)$_2$
d/ $(172,125)_{10}$ = (.....)$_2$

Exercice N°13 :

A partir des tables de vérité suivantes, écrire l'équation à l'aide de la méthode de la somme de produits et celle du produit de sommes.

Exemple 1

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

Exemple 2

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

Exemple 3

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

Exercice N°14 :

A partir de la description du circuit, établir la table de vérité correspondante et établir son équation logique en choisissant le type d'écriture.

a) Une lampe éclaire si on agit sur un bouton poussoir A ou si on agit sur un bouton poussoir B. Elle n'éclaire pas s'il n'y a pas d'action ni sur A ni sur B, ou s'il y a action à la fois sur A et sur B.

b) Une perceuse peut fonctionner (c.-à-d. que l'on peut mettre son moteur en marche) dans les seuls cas suivants :

- S'il y a une pièce, dans un étau et si cet étau est serré.
- S'il n'y a pas de pièce, étau serré ou non.

c) Le système de commande de l'ouverture de la porte du garage d'un hôtel :

- Pour l'entrée dans le garage : avec l'autorisation d'entrée délivrée depuis son bureau, par le réceptionniste et la demande de l'accès du client, le système d'ouverture de la porte est actionné.
- Pour la sortie du garage : seule la demande de sortie du client est nécessaire pour ouvrir la porte.

Exercice N°15 :

Effectuer les opérations arithmétiques suivantes :

Exemple 1 :

0011	1101	110	1111/11
+	-	*	
1101	0010	11	

Exemple 2 :

10110110	1010111	1001	100000/110
+	-	*	
1011101	10101	1100	

Exemple 3 :

1110111			1000010/1011
+			
101101	10110101	1001	
+	-	*	
1101	1110101	1100	

Exercice N°16 :

A partir des tables de vérité suivantes, écrire l'équation à l'aide de la méthode de la somme de produits et celle du produit de sommes.

Exemple 1

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

Exemple 2

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

Exemple 3

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

Exercice N°17 :

A partir de la description du circuit, établir la table de vérité correspondante et établir son équation logique en choisissant le type d'écriture.

a) Une lampe éclaire si on agit sur un bouton poussoir A ou si on agit sur un bouton poussoir B. Elle n'éclaire pas s'il n'y a pas d'action ni sur A ni sur B, ou s'il y a action à la fois sur A et sur B.

b) Une perceuse peut fonctionner (c.-à-d. que l'on peut mettre son moteur en marche) dans les seuls cas suivants :

- S'il y a une pièce, dans un étau et si cet étau est serré.
- S'il n'y a pas de pièce, étau serré ou non.

c) Le système de commande de l'ouverture de la porte du garage d'un hôtel :

- Pour l'entrée dans le garage : avec l'autorisation d'entrée délivrée depuis son bureau, par le réceptionniste et la demande de l'accès du client, le système d'ouverture de la porte est actionné.
- Pour la sortie du garage : seule la demande de sortie du client est nécessaire pour ouvrir la porte.

Exercice N°18 :

A partir des tables de vérité, Élaborer les diagrammes de Karnaugh à la droite des tables de vérité qui suivent :

Table de vérité	Tableau de Karnaugh

A	B	S1
0	0	1
0	1	1
1	0	0
1	1	1

B\A	0	1
0		
1		

A	B	C	S2
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

C\AB	00	01	11	10
0				
1				

A	B	C	D	S3
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

CD\AB	00	01	11	10
00				
01				
11				
10				

Exercice N°19 :

Effectuer les regroupements et écrire l'équation logique simplifiée correspondante des diagrammes de Karnaugh précédents. (Utilisez les regroupements des 1 et regroupements des zéro)

Exercice N°20 :

Effectuer les regroupements et écrire l'équation logique simplifiée correspondante des diagrammes de Karnaugh suivants. (Utilisez les regroupements des 1 et regroupements des zéro)

B\A	0	1
0	1	1
1	0	0

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

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

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

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

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

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

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

Exercice N°21 :

Pour chacune des équations logiques suivantes, établir le diagramme de Karnaugh. Effectuer les regroupements et écrire l'équation logique simplifiée correspondante.

$F1 = \overline{A}.\overline{B} + A.\overline{B} + A.B$

$F2 = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.B.C$

$F3 = \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.B.\overline{C}.D + A.B.\overline{C}.D + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.C.\overline{D} + A.\overline{B}.\overline{C}.D + A.\overline{B}.C.D$

Exercice N°22 :

Traduire les équations suivantes en schémas logiques comprenant des opérateurs **NON**, **ET**, **OU** :

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

$F2 = A.\overline{B} + A.\overline{B}.\overline{C} + A.\overline{B}.C$

$F3 = B.\overline{C}.\overline{D} + \overline{A}.\overline{C}.D + \overline{A}.C.\overline{D} + A.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + A.\overline{B}.\overline{D}$

$F4 = \overline{A}.\overline{C} + B.C$

Exercice N°23 :

Traduire les mêmes équations que l'exercice 1 en schémas logiques ne comprenant que des opérateurs **NAND** (NON ET).

Exercice N°24 :

Traduire les mêmes équations que l'exercice 1 en schémas logiques ne comprenant que des opérateurs **N0R** (NON OU).

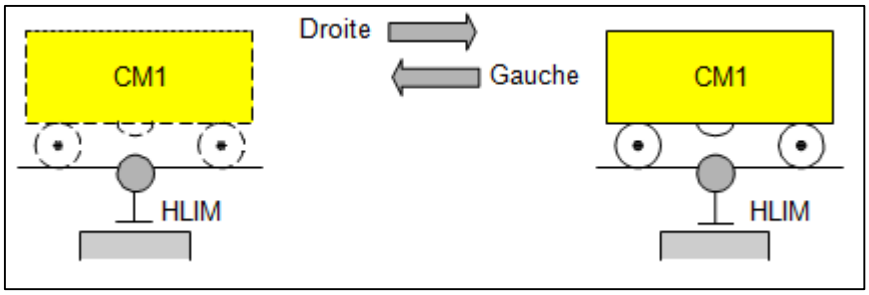
Exercice N°25

A partir d'un cahier de charge d'un équipement automatisé, déterminer les grandes étapes, les points d'entrée ou de sortie de données, les points de prise de décision, s'il y a répétition ou arrêt de la séquence, s'il y a saut de séquence, s'il y a un choix conditionnel entre plusieurs séquences etc.

Un chariot peut se déplacer entre deux fins de course.
Initialement, le chariot se trouve à gauche. En activant un bouton départ cycle (dcy) le chariot effectue le cycle suivant :

- ✓ déplacement vers la droite jusqu'à fin de course HLIM
- ✓ déplacement vers la gauche jusqu'à fin de course HLIM
- ✓ arrêt du chariot.

Décrire les règles de construction de divers représentations graphiques.

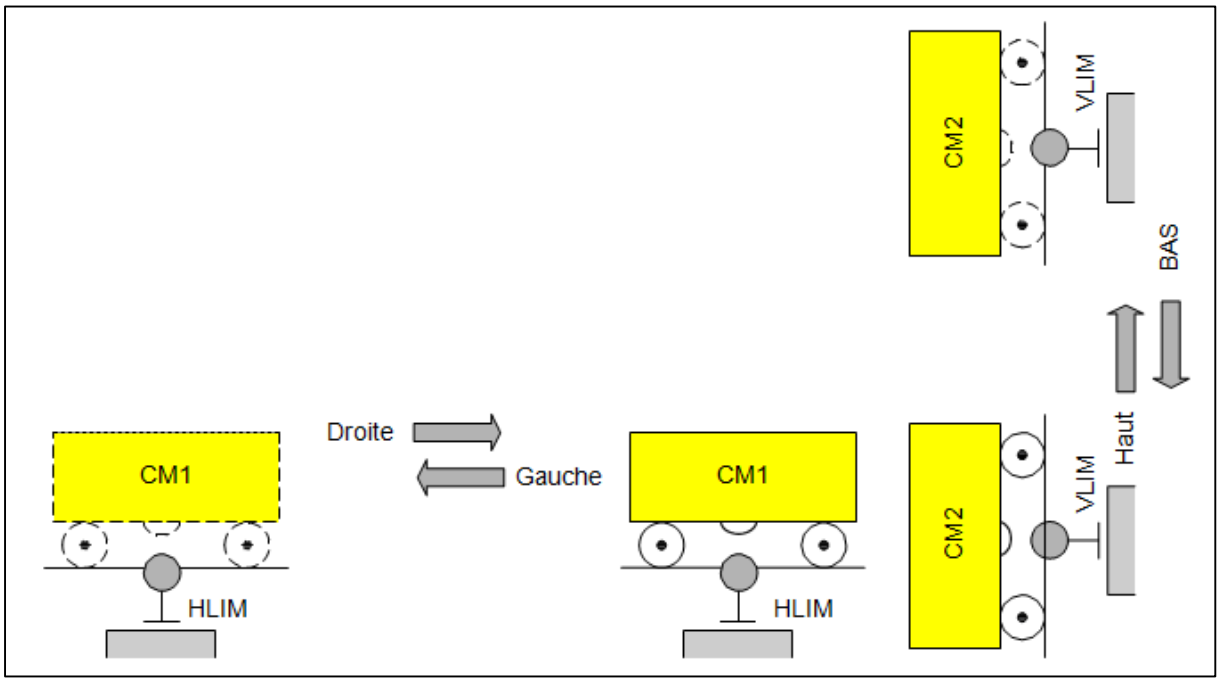


Exercice N°26:

Le chariot 1 est à droite et le chariot 2 en bas. En activant le bouton poussoir dcy les chariots effectuent le cycle suivant :

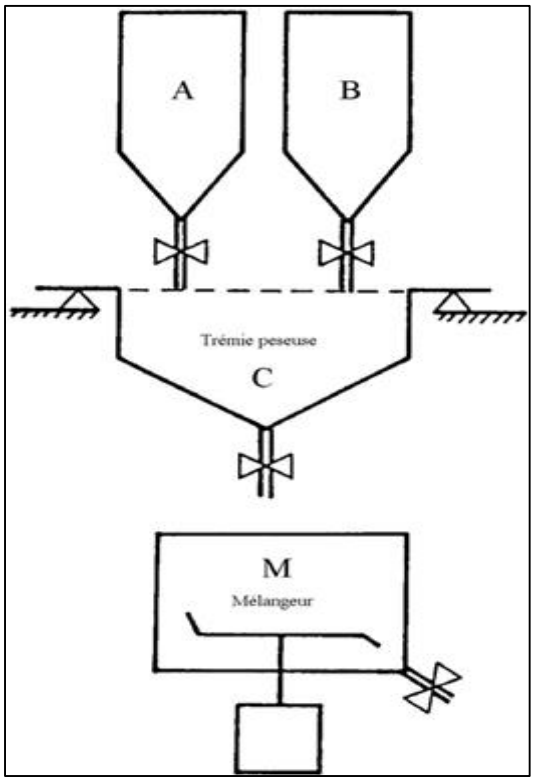
- ✓ CH 1 se déplace vers la gauche et le CH 2 vers le haut
- ✓ CH 1 se déplace vers la droite jusqu'à HLIM et temporisation de 5s
- ✓ A la fin de la temporisation le CH 2 se déplace vers le bas.

Décrire les règles de construction de divers représentations graphiques.



Exercice N°27 :

Une station de mélange se compose de deux réservoirs contenant deux produits A et B pouvant se déverser dans une trémie peseuse C. Un mélangeur M permet d'obtenir l'homogénéisation du mélange formé par ces deux produits grâce à la rotation d'une hélice.



L'ordre de départ du cycle donné par l'opérateur ne peut être pris en compte que si les conditions initiales sont réalisées, c'est à dire si la trémie et le mélangeur sont vides.

La quantité de produit A est d'abord pesée dans la trémie C et celle-ci est immédiatement vidangée dans le mélangeur M.

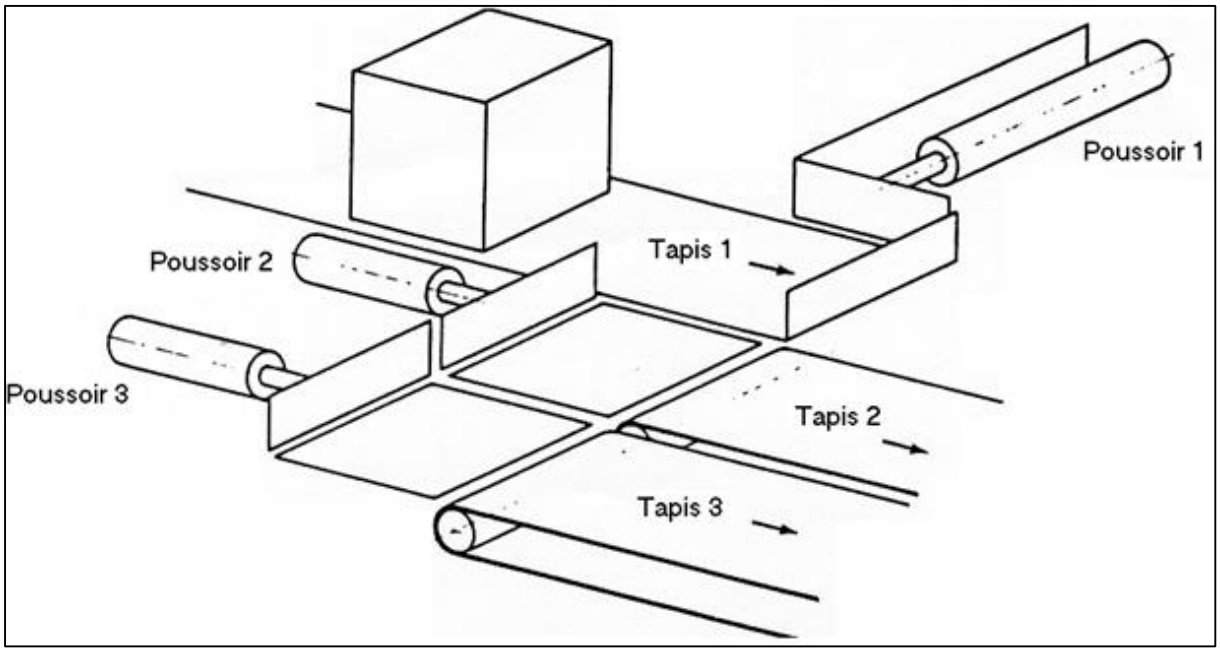
Le produit B est ensuite pesé et mélangé au produit A présent dans le mélangeur.

Ces deux produits sont malaxés pendant 20s, temps au bout duquel le mélangeur est vidangé.

Décrire les règles de construction de divers représentations graphiques.

Exercice N°28 :

Un dispositif automatique destiné à trier des caisses de deux tailles différentes se compose d'un tapis, amenant les caisses, de trois poussoirs et de deux tapis d'évacuation suivant le figure ci- dessous.



Le poussoir 1 pousse les petites caisses devant le poussoir 2 qui à son tour les transfère sur le tapis d'évacuation 2, alors que les grandes caisses sont poussées devant le poussoir 3, ce dernier les évacue sur le tapis 3.

Pour effectuer la sélection des caisses, un dispositif de détection placé devant le poussoir 1 permet de reconnaître sans ambiguïté le type de caisse qui se présente.

Décrire les règles de construction de divers représentations graphiques.

Exercice N°29

Soit une came C entraînée en rotation par un motoréducteur. Cette came doit effectuer un tour et un seul à chaque fois que l'ordre lui est donné.

Déterminer les modes de marche et d'arrêt.

Exercice N° 30 :

A l'étape de départ la lampe est éteinte.

Une première impulsion sur un bouton poussoir «b» allume la lampe. Lorsque le bouton «b» est relâché, la lampe reste allumée.
Une seconde pression sur «b» éteint la lampe, celle-ci s'allumera de nouveau lorsque b sera enfoncé.

Déterminer les modes de marche et d'arrêt.

Exercice N°31 :

Un chariot peut se déplacer entre deux positions caractérisées par deux fins de course. Initialement, le chariot se trouve à gauche. En activant un bouton poussoir départ cycle «dcy

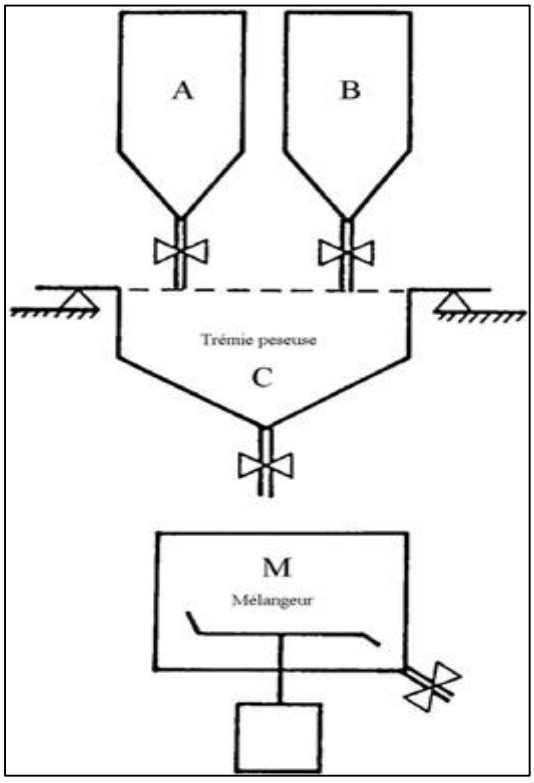
», le chariot effectue le cycle suivant :

- ✓ Déplacement vers la droite jusqu'à fin de course1;
- ✓ Déplacement vers la gauche jusqu'à fin de course 2;
- ✓ Puis arrêt du chariot.

Déterminer les modes de marche - arrêt et les conditions initiales de départ.

Exercice N°32 :

Une station de mélange se compose de deux réservoirs contenant deux produits A et B pouvant se déverser dans une trémie peseuse C. Un mélangeur M permet d'obtenir l'homogénéisation du mélange formé par ces deux produits grâce à la rotation d'une hélice.



L'ordre de départ du cycle donné par l'opérateur ne peut être pris en compte que si les conditions initiales sont réalisées, c'est à dire si la trémie et le mélangeur sont vides.

La quantité du produit A est d'abord pesée dans la trémie C et celle-ci est immédiatement vidangée dans le mélangeur M.

Le produit B est ensuite pesé et mélangé au produit A présent dans le mélangeur.

Ces deux produits sont malaxés pendant 20s, temps au bout duquel le mélangeur est vidangé.

Déterminer les conditions initiales, les modes de marche – arrêt.

2- Autoévaluation

2.1 SYSTEMES DE NUMERATION

1- Le nombre binaire « 1011 » s'écrit en décimal:
   - 10
   - mille onze
   - 11
2- Le nombre « 37B » est écrit en:
   - Octal
   - Hexadécimal
   - Décimal
3- Le nombre binaire « 101011010110 » converti en base 16 donne:
   - 6706
   - 2774
   - AD6
4- En base 2, « 1+1 » égale:
   - 11
   - 10
   - 2
5- L'addition binaire de « 1101110 + 1011101 » a pour résultat:
   - 1001011
   - 11001011
   - 2112211
6- La somme « 84A2+5F7D » a pour résultat:
   - C2FF
   - E41F
   - 1441F
7- Dans le code Gray (binaire réfléchi) le nombre « 10 » :
   - précède « 11 ».
   - succède à « 11 ».
   - n'existe pas.
8- « -9 » s'écrit en binaire:
   - -1001
   - 11001
   - 10111
9- Sur 8 bits, on peut avoir :
   - 256 valeurs
   - 255 valeurs
   - 512 valeurs
10- Sur un octet la valeur maximale décimale est :
   - 256
   - 255
   - 512

2.2 ALGEBRE BOOLEENNE

1- D'après cette table de vérité, « S » à pour équation Booléenne:

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

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

2- L'expression simplifiée de l'équation logique suivante est :

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

- $Y = c$
- $Y = a + c$
- $Y = a.c$

3- L'équation Booléenne « S = (A et B) ou C » peut s'écrire:

- S = (A + B)C
- S = (AB+C)
- S = (A+C) et (B+C)

4- Le complément de l'équation « $W = \overline{A}(B + \overline{C}) + D$ » est:

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

5- L'équation Booléenne « $Y = (A + BC)(\overline{A}\ \overline{B} + C) + C$ » se simplifie en:

- $Y = C$
- $Y = A + B$
- $Y = AB$

6- L'équation $Z = \overline{\overline{A}B + A\overline{B}}$ est égale à:

- $Z = 0$
- $Z = 1$
- $Z = AB + \overline{A}\,\overline{B}$

7- Le circuit CD 4011 est un circuit logique qui contient :

- Quatre portes «ET» à 2 entrées
- Quatre portes «NON ET» à 2 entrées
- Quatre portes «NON OU» à 2 entrées

8- Le circuit SN74LS00 est un circuit logique qui contient :
- Quatre portes «ET» à 2 entrées
- Quatre portes «NON ET» à 2 entrées
- Quatre portes «NON OU» à 2 entrées

9- Une porte NOR (OU-NON) à pour équation:
- $\overline{A} + \overline{B}$
- $\overline{A}\,\overline{B}$
- $A\overline{B} + \overline{A}B$

10- Une porte XOR (OU Exclusif) à pour équation:
- $\overline{A} + \overline{B}$
- $\overline{A}\,\overline{B} + AB$
- $A\overline{B} + \overline{A}B$

2.3 LOGIQUE COMBINATOIRE

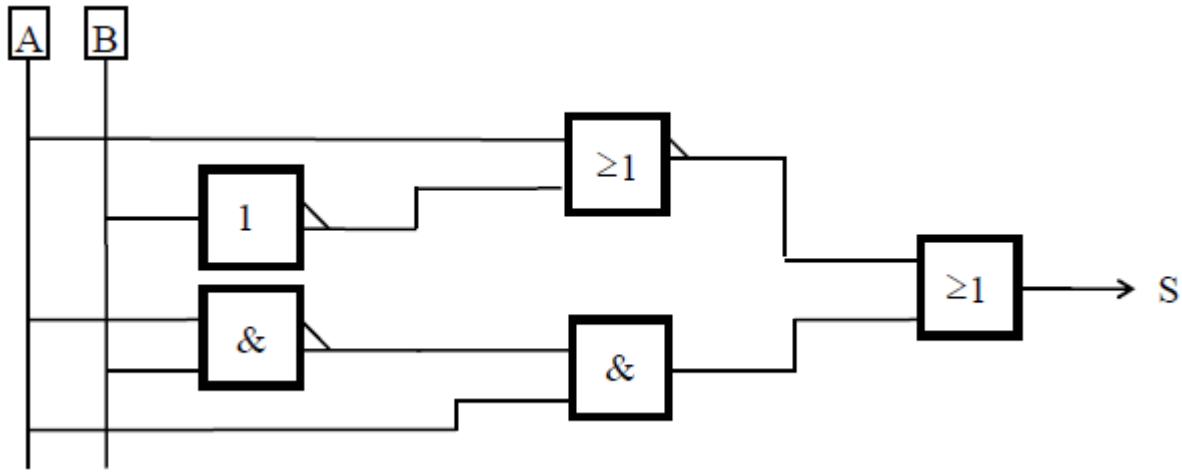
1- On donne le tableau de Karnaugh suivant:

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

L'équation simplifiée est :

- $\overline{C}\,\overline{D} + \overline{B}\,\overline{D} + A$
- $\overline{C}DA + B + \overline{C}D$
- $A + \overline{D} + BDC + \overline{B}$

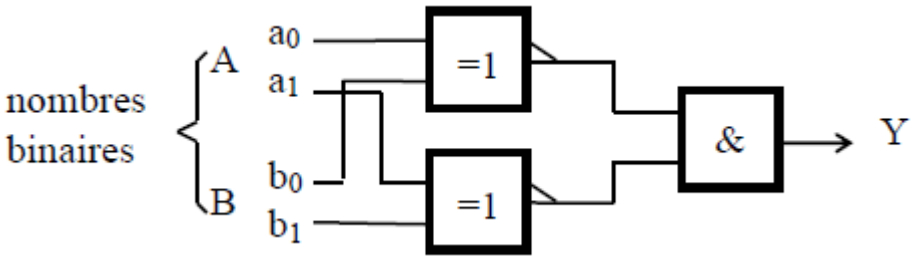
2- On donne le logigramme suivant:



L'équation logique de ce logigramme est :

- $S = AB + \overline{A}\,\overline{B}$
- $S = A\overline{B} + \overline{A}B$
- S = A + B

3- Le logigramme ci-dessous a une fonction:



- détection de différence
- détection d'égalité
- additionneur binaire

4- Un circuit 74LS appartient à la famille :
- CMOS rapide
- TTL rapide
- TTL à faible consommation

5- Un circuit 74HC appartient à la famille :
- CMOS rapide
- TTL rapide
- TTL à faible consommation

6- La référence « 74HCT » est un CI :
- CMOS à transdistortion
- CMOS compatible TTL
- TTL compatible CMOS

2.4 LOGIQUE SEQUENTIELLE

1- Une bascule R-S a pour fonction principale une mise:
- au niveau haut
- au niveau bas
- en mémoire

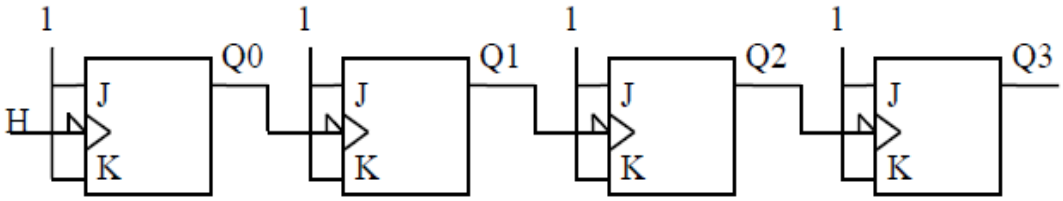
2- La sortie « Q » d'une bascule JK ayant ses entrées J et K à « 1 » :
- est invariable
- bascule à chaque front actif d'horloge
- prend la valeur de $\overline{Q}$ à chaque front actif d'horloge

3- La sortie «$\overline{Q}$ » d'une bascule D:
- prend la même valeur que l'entrée D

- prend la même valeur que l'entrée D au front actif du signal d'horloge
- prend la valeur complémentée de D

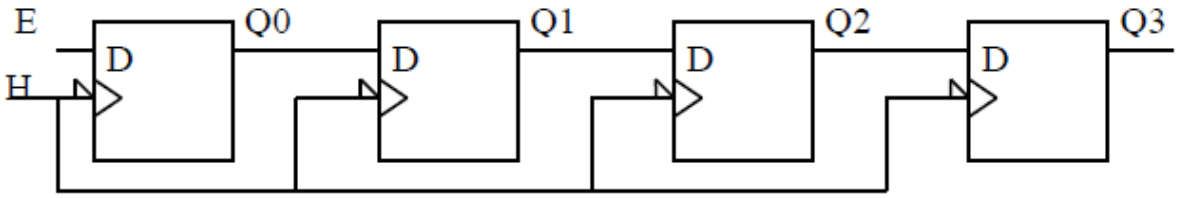
4- Une bascule D dont la sortie Q est reliée à l'entrée D réalise une division de fréquence par:
- 2
- 4
- 8

5- Le circuit suivant est :



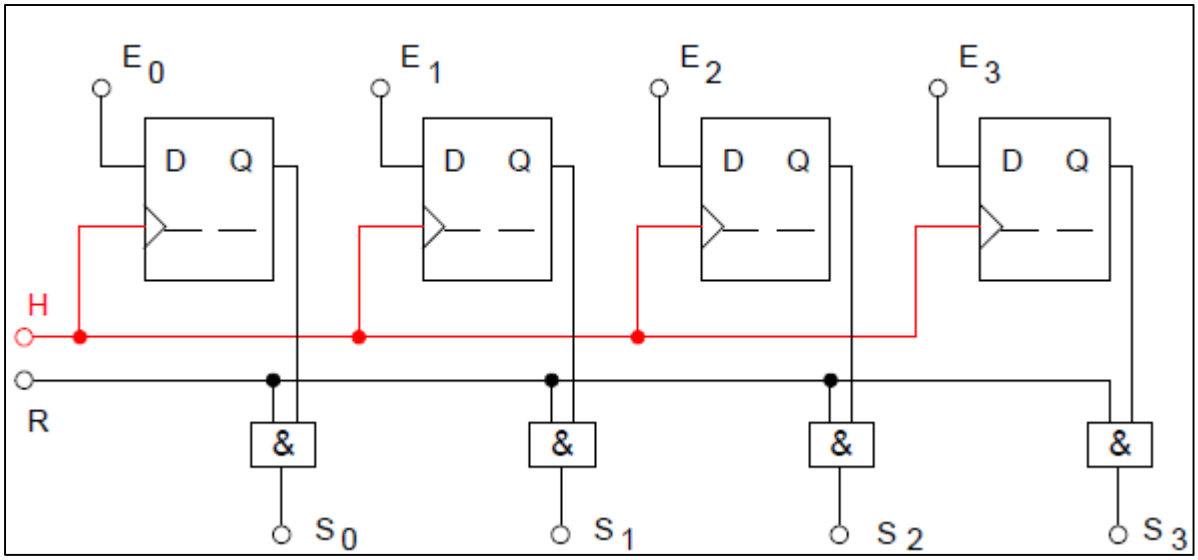
- un registre à décalage
- un compteur asynchrone
- un compteur synchrone

6- Le circuit suivant est :



- un registre à décalage
- un compteur asynchrone
- un compteur synchrone

7- Le circuit suivant est :



- un registre à décalage
- un compteur asynchrone
- un compteur synchrone

8- Un compteur BCD compte:

- de 0 à 1111 en binaire
- de 0 à 1001 en binaire
- de 0 à F en hexadécimal

9- Un registre à décalage 4 bits série-parallèle possède:

- quatre entrées séries et quatre sorties parallèles
- une entrée parallèle et quatre sorties séries
- une entrée série et quatre sorties parallèle

10- Un décodeur DCB-décimal à:

- en entrée un code binaire 8 bits et 16 lignes de sortie
- 10 lignes d'entrées et 4 bits de sortie
- en entrée un code binaire 4 bits et 10 lignes de sortie

11- Un codeur (encodeur) de clavier numérique possède:

- 4 entrées et 10 sorties
- 10 entrées et 4 sorties
- 3 entrées et 4 sorties

12- Le circuit intégrés 74151 possède principalement 8 entrées, 3 bits de sélection et une sortie. Est-ce:
- un multiplexeur
- un démultiplexeur
- un convertisseur

2.5 LES MEMOIRES

1- Une mémoire ayant une capacité de 4K x 8 peut stocker:
- 32000 bits
- 4096 octet
- 4000 x 8 bits

2- Cette même mémoire aura:
- 8 lignes (bits) d'adresse
- 32000 lignes d'adresse
- 12 lignes d'adresse

3- Une EPROM est une mémoire:
- vive
- une mémoire morte électriquement effaçable
- une mémoire morte effaçable par ultraviolet

4- Une EEPROM est une mémoire:
- vive
- une mémoire morte électriquement effaçable
- une mémoire morte effaçable par ultraviolet

5- Une RAM est une mémoire:
- vive
- une mémoire morte électriquement effaçable
- une mémoire morte effaçable par ultraviolet