

OFPPT

ROYAUME DU MAROC

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

DIRECTION RECHERCHE ET INGENIERIE DE FORMATION

**RESUME THEORIQUE
&
GUIDE DE TRAVAUX PRATIQUES**

MODULE N° 21 LOGIQUE COMBINATOIRE

SECTEUR : ELECTROTECHNIQUE

SPECIALITE : EMI

NIVEAU : TECHNICIEN

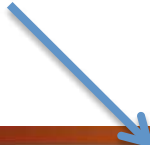
ANNEE 2007

PORTAIL DE LA FORMATION PROFESSIONNELLE AU MAROC

Télécharger tous les modules de toutes les filières de l'OFPPT sur le site dédié à la formation professionnelle au Maroc : www.marocetude.com

Pour cela visiter notre site www.marocetude.com et choisissez la rubrique :

MODULES ISTA



HOME LIVRES **MODULES ISTA** ANNUAIRE ECOLES DOCTORAT LETTRE DE MOTIVATION NOUS CONTACTER SE CONNECTER

Maroc Etude.Com Connaissance - Métier - Technique

[Annonces Google](#) [Emploi Maroc](#) [Messagerie](#) [Telecharger Un Jeu](#) [Maroc Annonces](#)

recherche...

Nous avons 14 invités en ligne

Annonces Google

[Annonces Emploi Maroc](#)
[Jeux Telecharger Gratuit](#)
[Jeux PC En Ligne](#)

Connexion

Identifiant
sniper

Mot de passe
.....

Se souvenir de moi

Connexion

[Mot de passe oublié ?](#)
[Identifiant oublié ?](#)

Notre Bibliothèque que ...Livres à Télé charger Gratuitement

MacKeeper

-20%

Complete your Purchase Now and save 20% Guaranteed with this Coupon Code

Apply Discount Automatically

"On ne jouit bien que de ce qu'on partage" [Madame de Genlis]

Annonces Google

[Jeu De Jeux](#)
[Jeux Sur Internet](#)
[Ecole Ingénieur](#)

Dépanner et configurer votre réseau à domicile

(Outil de Diagnostic)
Wi-Fi / Ethernet
Console de jeu
Imprimante
Messagerie

Document élaboré par :

Nom et prénom

EFP

DR

Mme ELKORNO NAIMA

CDC - GE

Révision linguistique

-
-
-

Validation

-
-
-

SOMMAIRE

RESUME THEORIQUE	7
I. Logique booléenne :	8
I.1 Définition :	8
I.2 Les lois de l'algèbre de Boole	8
I.3 Les Théorèmes de l'algèbre de Boole :	9
I.4 Postulats de l'algèbre de Boole :	9
I.5 Simplification algébrique des équations booléennes :	9
II. Les systèmes de numération :	10
II.1 Base d'un système de numération:	10
II.2 Changement de base :	12
II.3 Opérations arithmétiques avec la base binaire:	15
II.4 Les codes :	17
III. Les fonctions logiques de base	19
III.1 Les trois fonctions logiques de base	19
III.2 Les autres fonctions logiques :	19
III.3 Les fonctions logiques matérialisées avec des interrupteurs:	21
III.4 Symbolisation	24
IV. Table de vérité	25
IV.1 Tableau des combinaisons	25
IV.2 Règles de construction d'une table de vérité :	25
IV.3 Ecriture d'une équation à partir d'une table de vérité	26
IV.4 Elaboration d'une table de vérité à partir d'une équation	28
V. Simplification des fonctions logiques par la méthode de Karnaugh :	29
V.1 Transposition d'une équation logique dans un diagramme de Karnaugh	29
V.2 Simplification d'une équation par le diagramme de Karnaugh	32
V.3 Écriture des équations à partir de regroupement	35
VI. Circuits intégrés logiques	37
VI.1 Famille TTL (transistor transistor logic)	37
VI.2 Famille CMOS (Complementary Metal Oxide Semiconductor)	38
VI.3 Configuration des broches pour les différents modèles des C.I.	38
VII. Schémas logiques :	42
VII.1 Généralités :	42
VII.2 Différents types de schémas logiques:	43
VIII. Utiliser une sonde logique	49
IX. Montage des circuits	50
IX.1 Plaque de montage	50
IX.2 Interrupteurs logiques	51
IX.3 Choix de logique positive ou négative, visualisation des sorties	52
IX.4 Technique de travail	53
IX.5 Caractéristiques des circuits intégrés :	55
X. Circuits de base	57
X.1 Additionneur	57
X.2 Soustracteurs logiques	60
X.3 Multiplicateur	62

GUIDE DES EXERCICES ET TRAVAUX PRATIQUES	65
Exercices :	66
Tp 1 - portes logiques fondamentales : et (and), ou (or), Non (not)	68
Tp 2 - portes logiques fondamentales : non-et (nand), non-ou (nor), ou exclusif (xor)	70
Tp 3 - applications de l'algèbre de Boole	72
Tp 4 : applications de l'algèbre de Boole	74
Tp 5 : établir la table de vérité d'un circuit.....	75
Tp 6: monter des circuits de base	76
Evaluation de fin de module.....	79
Liste des références bibliographiques.....	80

MODULE 21:

LOGIQUE COMBINATOIRE

Code :

Durée : 45h

OBJECTIF OPERATIONNEL

COMPORTEMENT ATTENDU

*Pour démontrer sa compétence le stagiaire doit
appliquer des notions de logique combinatoire
selon les conditions, les critères et les précisions qui suivent*

CONDITIONS D'EVALUATION

- A partir :
 - De directives ;
 - De schémas;
 - D'une équation non simplifiée.
- A l'aide :
 - De manuels techniques ;
 - De fiches techniques ;
 - De composants logiques ;
 - D'outils et d'instruments de mesure ;

CRITERES GENERAUX DE PERFORMANCE

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

OBJECTIF OPERATIONNEL

**PRECISIONS SUR LE
COMPOTEMENT ATTENDU**

**CRITERES PARTICULIERS DE
PERFORMANCE**

- | | |
|---|--|
| <p>A) Appliquer des notions d'algèbre booléenne.</p> <p>B) Effectuer des conversions entre des bases numériques et des codes.</p> <p>C) Établir les tables de vérité d'un circuit.</p> <p>D) Simplifier des équations par la méthode de Karnaugh.</p> <p>E) Traduire des équations en schémas.</p> <p>F) Monter des circuits de base.</p> | <ul style="list-style-type: none">- Respect des règles de l'algèbre de Boole.
- Exactitude des conversions.
- Respect des règles prescrites.
- Exactitude des résultats.
- Regroupement optimal des variables.
- Résultats exacts.
- Conformité du schéma avec l'équation.
- Tracé adéquat du schéma.
- Sélection judicieuse des composants.
- Conformité du montage avec le schéma.
- Fonctionnement correct du circuit. |
|---|--|

Présentation du Module :

Ce module de logique combinatoire permet au stagiaire de :

- Appliquer les notions d'algèbre de Boole;
- Effectuer les conversions entre des bases numériques et des codes;
- Établir la table de vérité d'une fonction logique;
- Transposer avec justesse les variables dans le tableau de Karnaugh et réduire les équations des sorties;
- Traduire les équations en schémas clairs, propres et conformes aux équations de départ;
- Choisir les composants correspondants aux fonctions logiques attendues ;
- Réaliser le montage du circuit choisi avec vérification du fonctionnement qui doit être conforme aux données de départ.

La durée de ce module est de 45 h dont 25 h de théorie, 17 h de pratique et 3 h d'évaluation.

MODULE N° 21: LOGIQUE COMBINATOIRE

RESUME THEORIQUE

I. Logique booléenne :

I.1 Définition :

Beaucoup de systèmes automatisés fonctionnent en utilisant des organes et des fonctions binaires. Ces organes et fonctions binaires ne peuvent être que dans deux états possibles. Par exemple, un détecteur de niveau peut être immergé ou submergé. Un voyant peut être allumé ou éteint.

Par convention, on représente par la valeur logique « 0 » l'un de ces états et par la valeur logique « 1 » l'autre état. La valeur logique « 0 » correspond à un organe binaire (ou une fonction binaire) dans un état dit « **non-activé** », « **non-actionné** » ou « **inactif** » (exemple : un voyant inactif est éteint). La valeur logique « 1 » correspond à un organe binaire (ou une fonction binaire) dans un état dit « **activé** », « **actionné** » ou « **actif** » (exemple : un voyant actif est allumé).

La mathématique des fonctions binaires est appelée **l'algèbre booléenne**, elle définit trois opérateurs de base ainsi qu'une foule de règles et de postulats. Ainsi, toutes les fonctions binaires (dites aussi logiques) sont des relations entre des entrées et des sorties logiques composées d'opérateurs de base et sur lesquelles on peut appliquer diverses règles d'algèbre de Boole.

I.2 Les lois de l'algèbre de Boole

	Lois	
Commutativité	L_1	$A \bullet B = B \bullet A$
	L_2	$A + B = B + A$
Associativité	L_3	$(A \bullet B) \bullet C = A \bullet (B \bullet C)$
	L_4	$(A + B) + C = A + (B + C)$
Distributivité	L_5	$A \bullet (B + C) = A \bullet B + A \bullet C$
	L_6	$(A + B) \bullet (A + C) = A + B \bullet C$
Absorption	L_7	$A + (A \bullet B) = A$
	L_8	$A \bullet (A + B) = A$
Expansion	L_9	$(A \bullet B) + (A \bullet \overline{B}) = A$
	L_{10}	$(A + B) \bullet (A + \overline{B}) = A$
De Morgan	L_{11}	$\overline{A \bullet B} = \overline{A} + \overline{B}$
	L_{12}	$\overline{A + B} = \overline{A} \bullet \overline{B}$
	L_{13}	$\overline{\overline{A + B}} = A + B$
	L_{14}	$\overline{\overline{A \bullet B}} = A \bullet B$
Similitude	L_{15}	$A + \overline{A} \bullet B = A + B$
	L_{16}	$A \bullet (\overline{A} + B) = A \bullet B$

1.3 Les Théorèmes de l'algèbre de Boole :

Théorèmes		
Invariance	T_1	$A \bullet 0 = 0$
	T_2	$A + 1 = 1$
Élément neutre	T_3	$A \bullet 1 = A$
	T_4	$A + 0 = A$
Idempotence	T_5	$A \bullet A = A$
	T_6	$A + A = A$
Complémentarité	T_7	$A \bullet \bar{A} = 0$
	T_8	$A + \bar{A} = 1$
Involution	T_9	$\overline{\bar{A}} = A$

1.4 Postulats de l'algèbre de Boole :

Postulats	
P_1	$0 \bullet 0 = 0$
P_2	$0 \bullet 1 = 1 \bullet 0 = 0$
P_3	$1 \bullet 1 = 1$
P_4	$0 + 0 = 0$
P_5	$1 + 0 = 0 + 1 = 1$
P_6	$1 + 1 = 1$
P_7	$\bar{0} = 1$
P_8	$\bar{1} = 0$

1.5 Simplification algébrique des équations booléennes :

La simplification d'une équation revient à appliquer les règles :

- Des opérations booléennes ;
- Des relations fondamentales ;
- Du théorème de MORGAN ;

Et à utiliser, éventuellement, comme moyens de simplification le tableau ou diagramme de Karnaugh.

Exemple :

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

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

Or, $b\bar{b} = 0$

$$z = (a\bar{b} + ac + bc)(\bar{a} + c) \text{ développons, on aura :}$$

$$z = a\bar{b}\bar{a} + a\bar{b}c + a\bar{c}\bar{a} + acc + bc\bar{a} + bcc$$

Or, $a\bar{a} = 0$ $a\bar{b}\bar{a} = 0$ $\bar{a}ac = 0$ et $ccc = c$

$$z = \bar{a}bc + a\bar{b}c + ac + bc$$

$$z = c[a(\bar{b}+1) + b(\bar{a}+1)]$$

Or, $\bar{a} + 1 = \bar{b} + 1 = 1$

$$z = (a + b)c$$

II. Les systèmes de numération :

II.1 Base d'un système de numération:

La base d'un système de numération est le nombre de chiffres différents qu'utilise ce système de numération. En électronique numérique, les systèmes les plus couramment utilisés sont : le système binaire, le système octal, le système décimal et le système hexadécimal.

Se rappeler que : $a^0 = 1$.

a) Système décimal :

C'est le système de numération décimal que nous utilisons tous les jours. C'est le système de **base 10** qui utilise donc 10 symboles différents : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Un nombre N (entier positif) exprimé dans le système de numération décimal est défini par la relation ci-dessous :

$$N = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \dots + a_0 \times 10^0$$

(où a_n est un chiffre de rang n)

Exemple : $N = (1975)_{10}$
 $N = 1 \times 10^3 + 9 \times 10^2 + 7 \times 10^1 + 5 \times 10^0$

Les puissances de 10 sont appelées les **poids** ou les **valeurs de position**. Le poids est égal à la base élevée à la puissance de son rang.

	Unité	Dizaine	Centaine	Milliers	10*Milliers	100*Millier s
Chiffre	a_0	a_1	a_2	a_3	a_4	a_5
Rang	0	1	2	3	4	5
Poids	10^0	10^1	10^2	10^3	10^4	10^5

b) Système binaire

Le système binaire est le système de base 2, c'est à dire qui utilise deux symboles différents : le 0 et le 1. Chacun d'eux est appelé bit ou élément binaire.

Dans ce système, le poids est une puissance de 2.

Exemple : $N = (10110)_2$
 $N = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
 $N = (22)_{10}$

- Puissance de 2 :

n	0	1	2	3	4	5	6	7	8	9	10	11	12
2^n	1	2	4	8	16	32	64	128	256	512	1024	2048	4096

n	13	14	15
2^n	8192	16384	32768

- Définitions :

Triplet : nombre binaire formé de 3 éléments binaires.

Quartet : nombre binaire formé de 4 éléments binaires.

Octet (byte) : nombre binaire formé de 8 éléments binaires.

Mot (word) : nombre binaire formé de 16 éléments binaires.

L.S.B. : bit le moins significatif ou bit de poids faible (élément le plus à droite d'un nombre binaire).

M.S.B. : bit le plus significatif ou bit de poids fort (élément binaire le plus à gauche d'un nombre binaire).

c) Système octal

Le système de numération octal est de **base 8**, ainsi il utilise 8 symboles différents : 0, 1, 2, 3, 4, 5, 6 et 7.

Dans ce système, le poids est une puissance de 8.

Exemple : $N = (6543)_8$
 $N = 6 \times 8^3 + 5 \times 8^2 + 4 \times 8^1 + 3 \times 8^0$
 $N = (3427)_{10}$

La succession des nombres par ordre croissant est le suivant :

- 1 chiffre : 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2.....etc.

- 2 chiffres : 10, 11, 12, 13, 14, 15, 16, 17, 20, 21....., 27, 30, 31....etc.

- Puissance de 8 :

n	0	1	2	3	4	5
8^n	1	8	64	512	4096	32768

d) Système hexadécimal

Le système hexadécimal est de **base 16** et utilise 16 symboles différents : les dix premiers chiffres décimaux : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 et les 6 premières lettres de l'alphabet : A, B, C, D, E, F.

La succession des nombres hexadécimaux par ordre croissant est la suivante :

- 1 chiffre : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 0, 1, 2, 3.....etc.
- 2 chiffres : 00, 01, 02 , 09, 0A, 0B,....., 0F, 10, 11, 12,....., 19, 1A, 1B.....etc.

Les lettres de A à F correspondent respectivement aux nombres décimaux 10 à 15.
Dans ce système, le poids est une puissance de 16.

Exemple : $N = (AC53)_{16}$
 $N = A \times 16^3 + C \times 16^2 + 5 \times 16^1 + 3 \times 16^0$
 $N = 10 \times 16^3 + 12 \times 16^2 + 5 \times 16^1 + 3 \times 16^0$
 $N = (44115)_{10}$

- Puissance de 16 :

<i>n</i>	0	1	2	3	4	5
16^n	1	16	256	4096	65536	1048576

II.2 Changement de base :

a) Conversion des bases 2, 8 ou 16 en base 10

Pour convertir un nombre de la base 2, 8 ou 16 en nombre de base 10, il suffit de décomposer le nombre en ses quantités et d'en faire la somme.

Exemples :

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

$$= 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times 0,5 + 1 \times 0,25 = (22,25)_{10}$$

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

$$= 3 \times 64 + 7 \times 8 + 2 \times 1 + 0 \times 0,125 + 6 \times 0,015625 = (250,09375)_{10}$$

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

$$= 15 \times 16 + 13 \times 1 + 2 \times 0,0625 + 10 \times 0,00390625 = (253,1640625)_{10}$$

Tableau de correspondance entre nombre de différentes bases

Décimal (base 10)	Binaire (base 2)	Octal (base 8)	Hexadécimal (base 16)
	$2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$		
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8

9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

b) Conversion d'un nombre décimal en un nombre d'un système d'une autre base

Cette conversion se fait en deux parties :

- 1- La partie entière.
- 2- La partie fractionnaire.

Problème : Un nombre N étant donné en base 10, cherchons à l'écrire dans un système de base b .

- On traite d'abord la partie entière :

Pour la partie entière nous la divisons par la base b et nous conservons le reste. Le quotient obtenu est divisé par b et nous conservons le reste. Il faut répéter l'opération sur chaque quotient obtenu. Les restes successifs sont écrits, en commençant par le dernier, de la gauche vers la droite pour former l'expression de N dans le système de base b .

- On traite après la partie fractionnaire :

Exemple 1 : Conversion du nombre $(3786, 4)_{10}$ en base 2 :

- On traite d'abord la partie entière : 3786

$$\begin{array}{r|l} 3786 & 2 \\ \hline 0 & 1893 \end{array}$$

$$\begin{array}{r|l} 1893 & 2 \\ \hline 0 & 946 \end{array}$$

$$\begin{array}{r|l} 946 & 2 \\ \hline 0 & 473 \end{array}$$

$$\begin{array}{r|l} 473 & 2 \\ \hline 0 & 236 \end{array}$$

$$\begin{array}{r|l} 236 & 2 \\ \hline 0 & 118 \end{array}$$

$$\begin{array}{r|l} 118 & 2 \\ \hline 0 & 59 \end{array}$$

$$\begin{array}{r|l} 59 & 2 \\ \hline 1 & 29 \end{array}$$

$$\begin{array}{r|l} 1893 & 2 \\ \hline 1 & 946 \end{array}$$

$$\begin{array}{r|l} 946 & 2 \\ \hline 1 & 473 \end{array}$$

$$\begin{array}{r|l} 473 & 2 \\ \hline 1 & 236 \end{array}$$

$$\begin{array}{r|l} 236 & 2 \\ \hline 0 & 118 \end{array}$$

$$\begin{array}{r|l} 118 & 2 \\ \hline 0 & 59 \end{array}$$

$$\begin{array}{r|l} 59 & 2 \\ \hline 1 & 29 \end{array}$$

$$\begin{array}{r|l} 14 & 2 \\ \hline 0 & 7 \end{array}$$

$$\begin{array}{r|l} 7 & 2 \\ \hline 1 & 3 \end{array}$$

$$\begin{array}{r|l} 3 & 2 \\ \hline 1 & 1 \end{array}$$

$$\begin{array}{r|l} 1 & 2 \\ \hline 1 & 0 \end{array}$$

Le nombre binaire ainsi obtenu est :

$$3786 = (111011001010)_2$$

- On traite après la partie fractionnaire :

$$\begin{array}{r|l} 0,4 \times 2 = & \underline{0},8 \\ 0,8 \times 2 = & \underline{1},6 \\ 0,6 \times 2 = & \underline{1},2 \\ 0,2 \times 2 = & \underline{0},4 \\ 0,4 \times 2 = & \underline{0},8 \end{array}$$

D'où $(0,4)_{10} = (0,01100)_2$

Résultat : $(3786,4)_{10} = (111011001010,01100)_2$

Exemple 2 : $(459,3)_{10}$ le convertir en base 8.

- Partie entière :

Position	↑	Reste	459/8
8^0		3	57/8
8^1		1	7/8
8^2		7	0

$$(459)_{10} = (713)_8$$

- Partie fractionnaire :

$$\begin{array}{r|l} 0,3 \times 8 = & \underline{2},4 \\ 0,4 \times 8 = & \underline{3},2 \\ 0,2 \times 8 = & \underline{1},6 \\ 0,6 \times 8 = & \underline{4},8 \end{array}$$

$$(0,3)_{10} = (0,2314)_8$$

Résultat : $(459,3)_{10} = (713,2314)_8$

c) Autres conversions• **Conversion d'un nombre octal en un nombre binaire :**

Chaque symbole du nombre écrit dans le système octal est remplacé par son équivalent écrit dans le système binaire à trois bits (voir tableau de correspondance).

$$\text{Exemple : } N = (257)_8 = \begin{array}{ccc} (010 & 101 & 111)_2 \\ 2 & 5 & 7 \end{array}$$

• **Conversion d'un nombre binaire en un nombre octal :**

C'est l'opération inverse de la précédente. Il faut regrouper les 1 et 0 du nombre trois par trois en commençant par la droite, puis chaque groupe est remplacé par le chiffre octal correspondant.

$$\text{Exemple : } N = (11001101111)_2 = \begin{array}{ccc} 11 & 001 & 101 & 111 \\ & 3 & 1 & 5 & 7 \end{array}$$

$$N = (3157)_8$$

• **Conversion d'un nombre hexadécimal en un nombre binaire :**

Chaque symbole du nombre hexadécimal est remplacé par son équivalent écrit sur quatre bits dans le système binaire.

$$\text{Exemple : } N = (BF8)_{16}$$

$$N = \begin{array}{ccc} (1011 & 1111 & 1000)_2 \\ B & F & 8 \end{array}$$

• **Conversion d'un nombre binaire en un nombre hexadécimal :**

C'est l'inverse de la précédente. Il faut donc regrouper les 1 et 0 du nombre par quartet en commençant par la droite, puis chaque groupe est remplacé par le symbole hexadécimal correspondant.

$$\text{Exemple : } N = (100001101111)_2$$

$$N = \begin{array}{ccc} 1000 & 0110 & 1111 \\ 8 & 6 & F \end{array}$$

$$N = (86F)_{16}$$

II.3 Opérations arithmétiques avec la base binaire:

a) **Addition Binaire** : Les règles de base sont :

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 0 \text{ reporte } 1 \end{aligned}$$

Exemple :

1 1 1	Reports
1101	0
+	
1011	1
11000	1

b) **Soustraction Binaire** : Les règles de base sont

$0 - 0 = 0$
 $0 - 1 = 1$ Emprunte 1
 $1 - 0 = 1$
 $1 - 1 = 0$

Exemple :

0	Emprunt
11011	1
-	
110	1
10101	0

c) **Multiplication Binaire** : Les règles de base sont :

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

Exemple :

101	
*	
110	
000	
101	
101	
11110	

d) **Division Binaire** : Les règles de base sont :

$0 / 0 = \text{Indéterminé}$
 $0 / 1 = 0$
 $1 / 0 = \text{Impossible}$
 $1 / 1 = 1$

Exemple :

1010	/	10
-10		101
001		
-00		
10		
-10		
00		

II.4 Les codes :**a) Code binaire :**

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

b) Code B C D :*(Binary coded decimal) en français (Décimal codé Binaire)*

Décimal	B C D
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Exemple : $(115)_{10}$

$$= (0001\ 0001\ 0101)_{BCD}$$

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

Ou code américain pour l'échange d'information : c'est un code alphanumérique qui permet de représenter des chiffres, des lettres ainsi que divers caractères spéciaux. Il traduit ces caractères en langage machine.

Colonne C: caractère ASCII ou fonction de contrôle particulière.

Colonne D: décimal.

Colonne O: octal.

Colonne H: hexadécimal.

D	O	H	C	D	O	H	C	D	O	H	C	D	O	H	C
0	000	00	nul	32	040	20	sp	64	100	40	@	96	140	60	‘
1	001	01	soh	33	041	21	!	65	101	41	A	97	141	61	a
2	002	02	stx	34	042	22	“	66	102	42	B	98	142	62	b
3	003	03	etx	35	043	23	#	67	103	43	C	99	413	63	c
4	004	04	eot	36	044	24	\$	68	104	44	D	10	144	64	d
5	005	05	enq	37	045	25	%	69	105	45	E	101	145	65	e
6	006	06	acq	38	046	26	&	70	106	46	F	102	146	66	f
7	007	07	bel	39	047	27	`	71	107	47	G	103	147	67	g
8	010	08	BS	40	050	28	(72	110	48	H	104	150	68	h
9	011	09	HT	41	051	29)	73	111	49	I	105	151	69	i
10	012	0A	LF	42	052	2A	*	74	112	4A	J	106	152	6A	j
11	013	0B	VT	43	053	2B	+	75	113	4B	K	107	153	6B	k
12	014	0C	FF	44	054	2C	,	76	114	4C	L	108	154	6C	l
13	015	0D	CR	45	055	2D	-	77	115	4D	M	109	155	6D	m
14	016	0E	SO	46	056	2E	.	78	116	4E	N	110	156	6E	n
15	017	0F	SI	47	057	2F	/	79	117	4F	O	111	157	6F	o
16	020	10	dle	48	060	30	0	80	120	50	P	112	160	70	p
17	021	11	dc1	49	061	31	1	81	121	51	Q	113	161	71	q
18	022	12	dc2	50	062	32	2	82	122	52	R	114	162	72	r
19	023	13	dc3	51	063	33	3	83	123	53	S	115	163	73	s
20	024	14	dc4	52	064	34	4	84	124	54	T	116	164	74	t
21	025	15	nak	53	065	35	5	85	125	55	U	117	165	75	u
22	026	16	syn	54	066	36	6	86	126	56	V	118	166	76	v
23	027	17	etb	55	067	37	7	87	127	57	W	119	167	77	w
24	030	18	can	56	070	38	8	88	130	58	X	120	170	78	x
25	031	19	em	57	071	39	9	89	131	59	Y	121	171	79	y
26	032	1A	sub	58	072	3A	:	90	132	5A	Z	122	172	7A	z
27	033	1B	esc	59	073	3B	;	91	133	5B	[123	173	7B	{
28	034	1C	fs	60	074	3C	<	92	134	5C	\	124	174	7C	
29	035	1D	gs	61	075	3D	=	93	135	5D]	125	175	7D	}
30	036	1E	rs	62	076	3E	>	94	136	5E	^	126	176	7E	~
31	037	1F	us	63	077	3F	?	95	137	5F	-	127	177	7F	del

d) Code Gray :

(Code binaire réfléchi, ne peut être utilisé pour les opérations arithmétiques).

Nombre Décimal	Binaire	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

C'est une autre forme de la base binaire.

Un seul bit à la fois change d'état lorsqu'on passe d'un nombre au suivant.

III. Les fonctions logiques de base

III.1 Les trois fonctions logiques de base

Les fonctions logiques reposent sur trois opérateurs de base. Ce sont les fonctions logiques : « NON » (en anglais « NOT »), « ET » (en anglais « AND ») et « OU » (en anglais « OR »). Nous les présentons avec leurs équations et leurs tables de vérité.

a) La fonction logique « NON » :

Soit une variable booléenne nommée A. La fonction logique NON (A), appelé complément de A, sera notée \bar{A} (lire A barre). Le résultat de NON (A) sera également une variable booléenne. La table ci-contre est **la table de vérité** de cette fonction.

Fonction F=NON(A)	
A	F
0	1
1	0

Au niveau algébrique, l'équation correspondant à cette table de vérité est : $F = \bar{A}$

b) La fonction logique « ET » :

Soit deux variables booléennes nommées A et B. Le résultat de la fonction logique A ET B sera également une variable booléenne. Le tableau de droite montre **la table de vérité** de cette fonction. la fonction logique ET n'active la sortie que lorsque toutes les entrées sont actives.

Au niveau algébrique, l'équation correspondant à cette table de vérité est $F = A \bullet B$. Le symbole du ET est semblable à celui du produit.

Fonction F = A ET B		
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

c) La fonction logique « OU » :

Soit deux variables booléennes nommées A et B. Le résultat de la fonction logique A OU B sera également une variable booléenne. **La table de vérité** de cette fonction est montrée à droite. La fonction logique OU donne une valeur de sortie égale à 1 dès qu'une des entrées est à 1.

Au niveau algébrique, l'équation correspondant à cette table de vérité est $F = A + B$. Le symbole du OU est semblable à celui de la somme.

Fonction F = A OU B		
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

III.2 Les autres fonctions logiques :

En plus des opérateurs de base, il existe d'autres opérateurs de deux variables et nous présentons ici les plus importants.

a) Opérateur « OUI » ou opérateur égalité

La sortie est à l'état 1 si, et seulement si, l'entrée est à l'état 1.

b) La fonction logique « NON-ET » :

Soit deux variables booléennes nommées A et B. Le résultat de la fonction logique NON(A ET B) sera également une variable booléenne. La figure de droite montre **la table de vérité** de cette fonction. La sortie de la fonction logique NON-ET (en anglais NAND) a un comportement inverse à celle de la fonction logique ET.

Fonction "NON-ET"		
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Au niveau algébrique, l'équation correspondant à cette table de vérité est $F = \overline{A \bullet B}$. L'ajout de la barre au-dessus de la fonction traduit ainsi l'inversion du résultat de l'ET.

c) La fonction logique « NON-OU »

Soit deux variables booléennes nommées A et B. Le résultat de la fonction logique NON(A OU B) sera également une variable booléenne. La table ci-contre montre **la table de vérité** de cette fonction. La sortie de la fonction logique NON-OU (en anglais NOR) a un comportement inverse à celle de la fonction logique OU.

Fonction "NON-OU"		
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

Au niveau algébrique, l'équation correspondant à cette table de vérité est $F = \overline{A + B}$. L'ajout de la barre au-dessus de la fonction traduit ainsi l'inversion du résultat de l'OU.

Remarque :

Ces deux fonctions (NON-ET et NON-OU) sont très utilisées en électronique, car elles représentent des éléments de connexions universels. Toute fonction logique peut en effet être écrite exclusivement à partir de l'une ou l'autre de ces fonctions. Un des avantages de ces éléments de connexion universel est de permettre l'implantation de n'importe quelle fonction logique à l'aide d'un seul type de circuit électronique. Il y a donc **standardisation** sur un seul type de circuit électronique. Ainsi, il est possible d'en acheter une grande quantité pour bénéficier d'un prix de vente avantageux. On ne stocke qu'un seul type de circuit en prévision d'éventuelles pannes.

d) La fonction logique « OU-EXCLUSIF »

Soit deux variables booléennes nommées A et B. Le résultat de la fonction logique A OU-EXCLUSIF B sera également une variable booléenne. **La table de vérité** de droite montre le comportement de cette fonction.

Fonction "OU-EXCL"		
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

la sortie de la fonction logique OUEXCLUSIF (en anglais EXOR) ne donne un niveau logique 1 que si, et seulement si, une seule entrée est à l'état 1.

Au niveau algébrique, l'équation correspondant à cette table de vérité est $F = A \oplus B$.

Cette expression peut aussi être écrite sous une autre forme : $F = A \oplus B = \overline{A}B + A\overline{B}$

e) La fonction logique « NON-OU-EXCLUSIF »

Soit deux variables booléennes nommées A et B. Le résultat de la fonction logique NON (A OUEXCLUSIF B) sera également une variable booléenne. La figure de droite montre la table de vérité de cette fonction. La sortie de la fonction logique NON-OUEXCLUSIF est l'inverse de celle obtenue avec la fonction logique OU-EXCLUSIF.

Funct "NON-OU-EXC"		
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Au niveau algébrique, l'équation correspondant à cette table de vérité est $F = \overline{A \oplus B}$.

III.3 Les fonctions logiques matérialisées avec des interrupteurs:

Comme il était mentionné précédemment un interrupteur est un organe binaire pouvant être actionné (lorsque l'opérateur appui dessus), ou non-actionné (lorsque l'opérateur n'y touche pas). Selon le type d'interrupteur, il peut être actionné par un opérateur (bouton de commande), par un mécanisme (interrupteur de fin de course), par la détection de la présence d'une grandeur physique (détecteur de niveau, de température, ...).

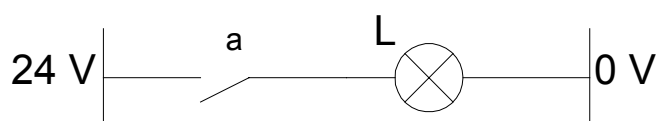
Dans sa position de repos, un interrupteur peut laisser passer ou non le courant électrique. Si l'interrupteur doit être actionné pour qu'il laisse passer le courant, nous avons un interrupteur dit « normalement ouvert ». Par contre, si en actionnant l'interrupteur le courant est coupé (ou ne passe plus), nous avons un interrupteur dit « normalement fermé »

Une ampoule (ou un voyant) est un organe binaire qui peut être actionné (lorsque le courant passe et que l'ampoule est allumée), ou non-actionné (lorsque le courant ne passe plus et que l'ampoule est éteinte).

Dans la suite nous verrons comment, en utilisant les interrupteurs comme entrées logiques et une ampoule comme sortie obtenir des fonctions logiques.

a) Fonction logique « OUI »

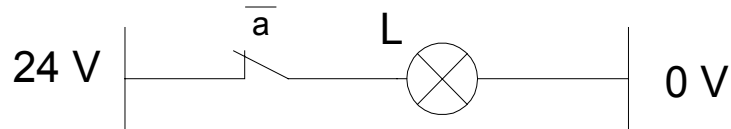
La figure ci-dessous montre la fonction logique OUI. Cette fonction utilise simplement un interrupteur normalement ouvert. Le voyant s'allumera si l'interrupteur est actionné.



La fonction logique de ce montage est : $L = a$

b) Fonction logique « NON »

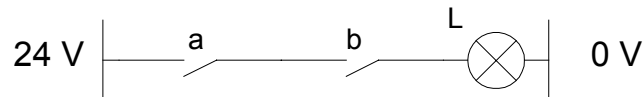
La figure ci-dessous montre la fonction logique NON. Pour obtenir cette fonction, il suffit de brancher un interrupteur normalement fermé. Le voyant s'allume tant que l'interrupteur n'est pas actionné.



La fonction logique de ce montage est : $L = \bar{a}$

c) Fonction logique « ET »

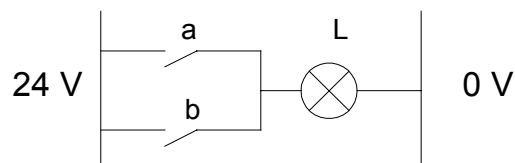
La figure ci-dessous montre la fonction logique ET. Pour obtenir cette fonction, il suffit de brancher deux interrupteurs normalement ouverts en série. Pour que le courant puisse traverser le voyant et l'allume, il faut actionner simultanément les deux interrupteurs.



La fonction logique de ce montage est : $L = a \bullet b$

d) Fonction logique « OU »

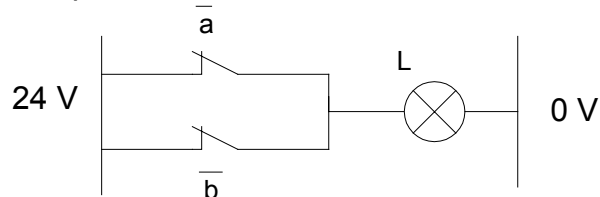
La figure ci-dessous montre la fonction logique OU. Pour obtenir cette fonction, il suffit de brancher deux interrupteurs normalement ouverts en parallèle. Le voyant s'allume dès que l'un des interrupteurs est actionné.



La fonction logique de ce montage est : $L = a + b$

e) Fonction logique « NON-ET »

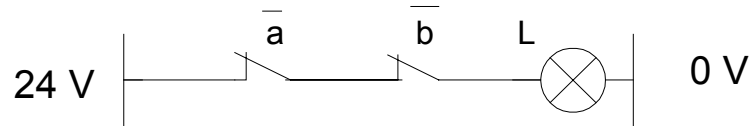
La figure ci-dessous montre la fonction logique NON-ET. Pour obtenir cette fonction, il suffit de brancher deux interrupteurs normalement fermés en parallèle. Le voyant s'éteint seulement si les deux interrupteurs sont actionnés simultanément.



La fonction logique de ce montage est : $L = \overline{a \bullet b} = \bar{a} + \bar{b}$

f) Fonction logique « NON-OU »

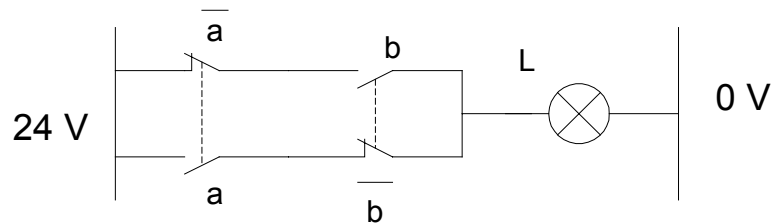
La figure ci-dessous montre la fonction logique NON-OU. Pour obtenir cette fonction, il suffit de brancher deux interrupteurs normalement fermés en série. Le voyant s'éteint dès qu'un des interrupteurs est actionné.



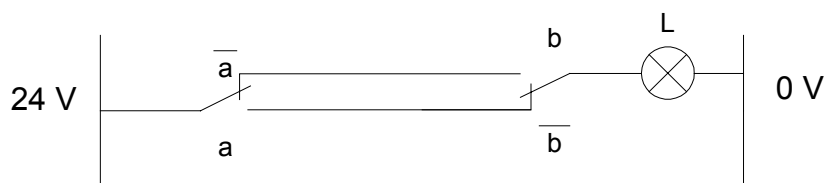
La fonction logique de ce montage est : $L = \overline{a+b} = \bar{a} \times \bar{b}$

g) Fonction logique « OU-EXCLUSIF »

La figure ci-dessous montre la fonction logique OU-EXCLUSIF. Pour obtenir cette fonction, il suffit de brancher, tel que montré, deux interrupteurs ayant deux contacts chacun, l'un normalement ouvert, l'autre normalement fermé. Le voyant s'allume si et seulement si un seul interrupteur est actionné.

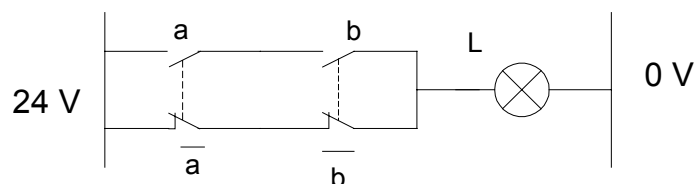


La fonction logique de ce montage est : $L = a \oplus b = \bar{a} b + \bar{b} a$
ou encore :



h) Fonction logique « NON-OU-EXCLUSIF »

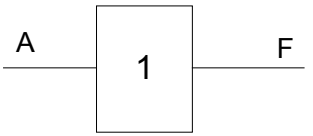
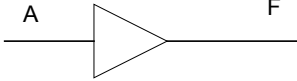
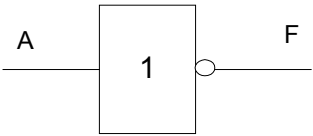
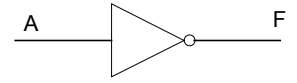
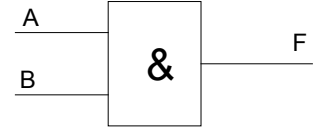
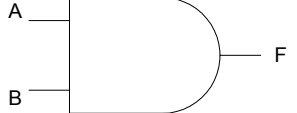
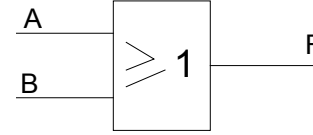
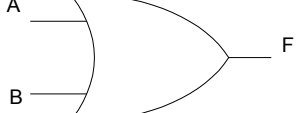
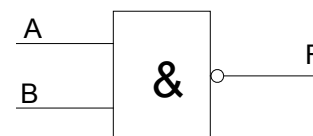
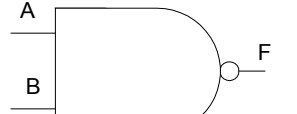
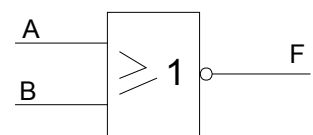

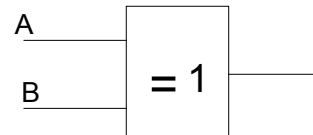
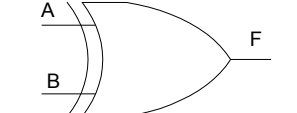
La figure ci-dessous montre la fonction logique NON-OU-EXCLUSIF. Pour obtenir cette fonction, il suffit de brancher, tel que montré, deux interrupteurs ayant deux contacts chacun, l'un normalement ouvert, l'autre normalement fermé. Le voyant s'éteint si et seulement si un seul interrupteur est actionné.



La fonction logique de ce montage est :

$$L = \overline{a \oplus b} = \overline{\bar{a} b + \bar{b} a} = \overline{\bar{a} b} \times \overline{\bar{b} a} = (\bar{a} + b) \times (\bar{b} + a) = \bar{a} \times \bar{b} + ab =$$

III.4 Symbolisation

Fonction	Symboles	
	NFC03-212	Américain
OUI : $F = A$		
NON : $F = \overline{A}$		
ET : $F = A \bullet B$ (AND)		
OU : $F = A + B$ (OR)		
NON ET : $F = \overline{A \bullet B}$ (NAND)		
NON OU : $F = \overline{A + B}$ (NOR)		
OU exclusif : $F = A \oplus B$		

IV. Table de vérité

IV.1 Tableau des combinaisons

De nombreux circuits logiques possèdent plusieurs entrées mais seulement une sortie. Une table de vérité nous fait connaître la réaction d'un circuit logique (sa valeur de sortie) aux diverses combinaisons de niveaux logiques appliqués aux entrées.

Pour une table de N entrées il y a 2^N lignes.

On peut construire un tableau de ces combinaisons comportant autant de colonnes que de variables d'entrées et autant de lignes que de combinaisons.

Pour le remplir, il suffit d'écrire pour chaque ligne l'équivalent binaire des nombres décimaux à compter de 0 à 2^{n-1} .

Exemples :

- a) 2 variables A et B on a $2^2 = 4$ combinaisons à compter de 0 à 3.

A	B	
0	0	→ L'équivalent binaire de 0
0	1	→ L'équivalent binaire de 1
1	0	→ L'équivalent binaire de 2
1	1	→ L'équivalent binaire de 3

- b) 3 variables A, B et C on a $2^3 = 8$ combinaisons à compter de 0 à 7.

	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

IV.2 Règles de construction d'une table de vérité :

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 à compter en binaire de 0 à $(2^n - 1)$;
- Inscire, dans la colonne « sortie », la valeur de la fonction pour chaque combinaison.

Exemple : Soit $S = A \bullet B + B \bullet C$. La table de vérité à 3 variables d'entrée.

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

IV.3 Ecriture d'une équation à partir d'une table de vérité

Il existe 2 méthodes :

a) **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 ».

Exemples :

1) Soit la table de vérité suivante à 2 variables :

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

→ 1^È ligne : $(A + B)$

→ 4^È ligne : $(\overline{A} + \overline{B})$

}

⇒ équation :

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

2) Soit la table de vérité suivant à 3 variables :

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

→ 2^e ligne : $(A+B+\bar{C})$
 → 4^e ligne : $(A+\bar{B}+\bar{C})$
 → 7^e ligne : $(\bar{A}+\bar{B}+C)$

⇒ équation :

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

Remarque :

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

b) 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 ».

Exemples :

1)

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

→ 2^e ligne : $(\bar{A} \cdot B)$
 → 3^e ligne : $(A \cdot \bar{B})$ } \Rightarrow L'équation : $S = (\bar{A} \cdot B) + (A \cdot \bar{B})$

2)

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

→ 1^e ligne : $(\bar{A} \cdot \bar{B} \cdot \bar{C})$
 → 3^e ligne : $(\bar{A} \cdot B \cdot \bar{C})$
 → 5^e ligne : $(A \cdot \bar{B} \cdot \bar{C})$
 → 6^e ligne : $(A \cdot \bar{B} \cdot C)$
 → 8^e ligne : $(A \cdot B \cdot C)$ } \Rightarrow l'équation :
 $S = (\bar{A} \cdot \bar{B} \cdot \bar{C}) + (\bar{A} \cdot B \cdot \bar{C}) + (A \cdot \bar{B} \cdot \bar{C}) + (A \cdot \bar{B} \cdot C) + (A \cdot B \cdot C)$

c) Équivalence entre le résultat d'un produit de sommes et celui d'une somme de produits

Le résultat d'un produit de sommes est égal à celui d'une somme de produits.

Exemple :

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

Somme de produits $\implies S = \bar{A} \cdot B + A \cdot B \quad (1)$

Produit de sommes $\implies S = (A + B) \cdot (\bar{A} + B) \quad (2)$

Preuve de l'égalité de ces deux équations (1) et (2) :

$$\begin{aligned}
 (1) \implies S &= \bar{A} \cdot B + A \cdot B \\
 &= (\bar{A} + A) \cdot B && \text{(Distributivité } L_5) \\
 &= 1 \cdot B && \text{(} T_8 : \text{ Complémentarité)} \\
 &= B && \text{(} T_3 : \text{ Élément neutre)}
 \end{aligned}$$

$$\begin{aligned}
 (2) \implies S &= (A + B) \cdot (\bar{A} + B) \\
 &= A \cdot \bar{A} + A \cdot B + \bar{A} \cdot B + B \cdot B && \text{(Distributivité } L_6) \\
 &= 0 + A \cdot B + \bar{A} \cdot B + B && \text{(} T_7, T_5) \\
 &= B + B && \text{(} L_9 : \text{ Expansion)} \\
 &= B && \text{(} T_6 : \text{ Idempotence)}
 \end{aligned}$$

IV.4 Elaboration d'une table de vérité à partir d'une équation

Nous pouvons passer d'une équation logique à une table de vérité. La technique est fort simple, il suffit de tester l'équation logique avec toutes les combinaisons d'entrée et de trouver pour chaque combinaison la valeur de la sortie. Un exemple simple montrera ce qu'il en est.

Soit $F = (A \cdot B) + \bar{C}$. Pour trouver la table de vérité de F, il faut premièrement connaître le nombre d'entrées. Ici, nous en avons trois, soit A, B et C. Donc la table de vérité devrait avoir normalement quatre (3+1) colonnes et huit (2³) lignes. Ainsi la table résultante sera :

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

V. Simplification des fonctions logiques par la méthode de Karnaugh :

Les tables de vérité permettent de trouver les équations logiques d'un système logique. Malheureusement, ces équations logiques ne sont pas simplifiées donc il faut les simplifier avant de réaliser les fonctions logiques correspondantes. Bien sûr, les règles, postulats et théorèmes de l'algèbre booléenne appliqués à une équation logique mènent à la simplification, mais au prix d'un certain effort. Plus le nombre d'entrées est élevé, plus la simplification est fastidieuse.

Pour simplifier les équations logiques de façon plus efficace. Une technique appelée « **Diagramme de Karnaugh** » est utilisée. Cette méthode présente la table de vérité sous une forme matricielle mettant en évidence les simplifications logiques.

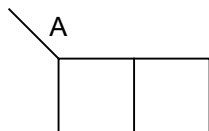
V.1 Transposition d'une équation logique dans un diagramme de Karnaugh

a) Diagramme de Karnaugh

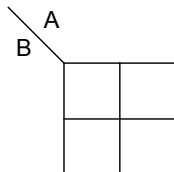
C'est un diagramme qui reprend les indications de la table de vérité pour les mettre sous une autre forme. Le nombre de cases est égal au nombre de lignes de la table de vérité, ou encore au nombre de combinaisons des variables d'entrée.

Exemples :

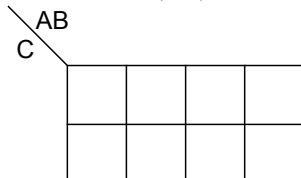
a) 1 variable d'entrée A \Rightarrow 2^1 combinaisons = 2 cases.



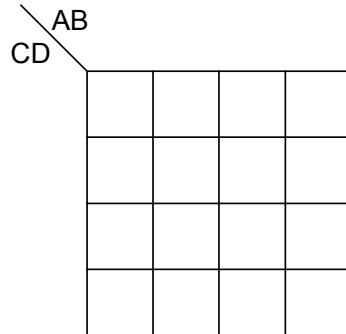
b) 2 variables d'entrée A et B \Rightarrow 2^2 combinaisons = 4 cases.



c) 3 variables d'entrée A, B, et C \Rightarrow 2^3 combinaisons = 8 cases.



d) 4 variables d'entrée A, B, C, et D \Rightarrow 2^4 combinaisons = 16 cases.



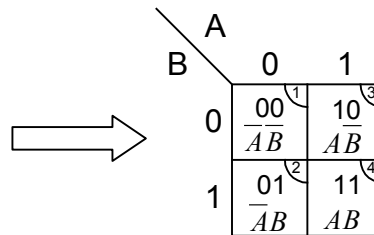
b) Disposition des combinaisons à l'intérieur du diagramme de Karnaugh

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.

Exemples :

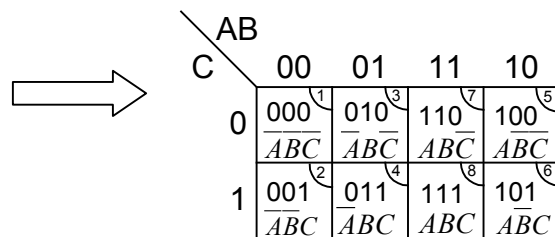
1.

	A	B	S
1-	0	0	
2-	0	1	
3-	1	0	
4-	1	1	



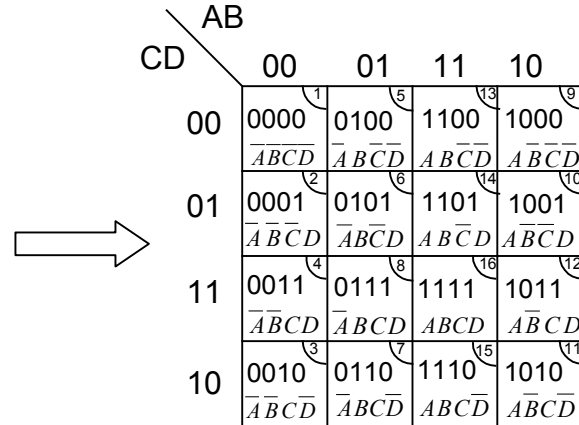
2.

	A	B	C	S
1-	0	0	0	
2-	0	0	1	
3-	0	1	0	
4-	0	1	1	
5-	1	0	0	
6-	1	0	1	
7-	1	1	0	
8-	1	1	1	



3.

	A	B	C	D	S
1-	0	0	0	0	
2-	0	0	0	1	
3-	0	0	1	0	
4-	0	0	1	1	
5-	0	1	0	0	
6-	0	1	0	1	
7-	0	1	1	0	
8-	0	1	1	1	
9-	1	0	0	0	
10-	1	0	0	1	
11-	1	0	1	0	
12-	1	0	1	1	
13-	1	1	0	0	
14-	1	1	0	1	
15-	1	1	1	0	
16-	1	1	1	1	



c) Transposition d'une équation logique dans un diagramme de Karnaugh

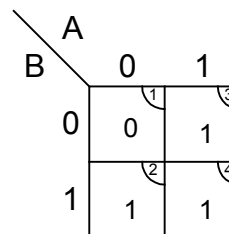
Exemples :

a) Soit l'équation : $S=A+B$

- Table de vérité

	A	B	S
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

- Diagramme de Karnaugh

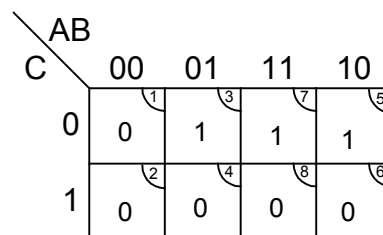


b) Soit l'équation : $S=A \cdot \bar{C} + B \cdot \bar{C}$

- Table de vérité

	A	B	C	S
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	0
5	1	0	0	1
6	1	0	1	0
7	1	1	0	1
8	1	1	1	0

- Diagramme de Karnaugh

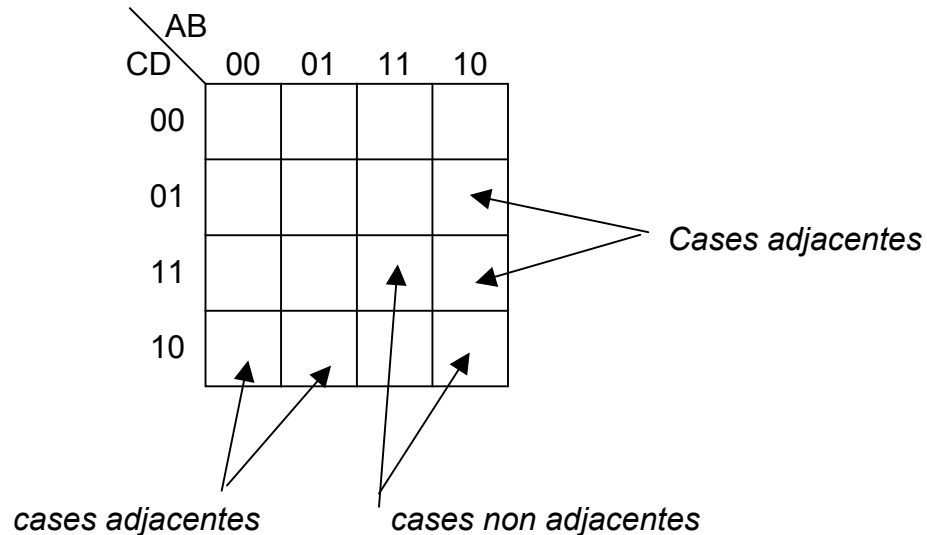


V.2 Simplification d'une équation par le diagramme de Karnaugh

a) 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.

Exemples :



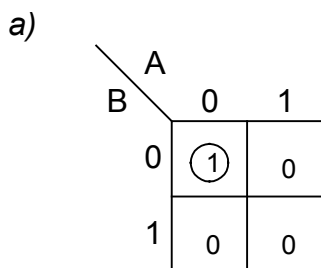
b) Règles de regroupement :

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

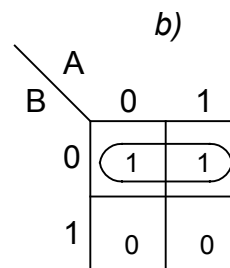
Règle 1 :

Le regroupement des cases adjacentes doit se faire par puissance de deux : $2^0, 2^1, 2^2, 2^3, \dots (1, 2, 4, 8 \dots)$

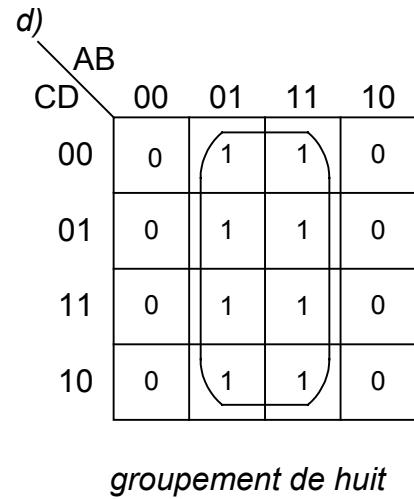
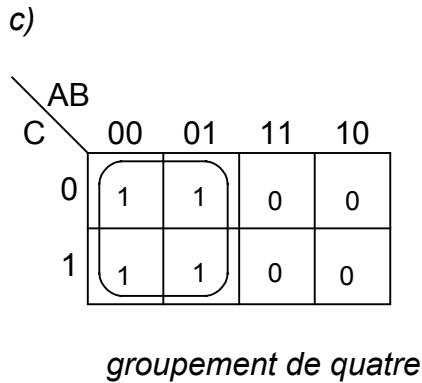
Exemples :



case unique



groupement de deux



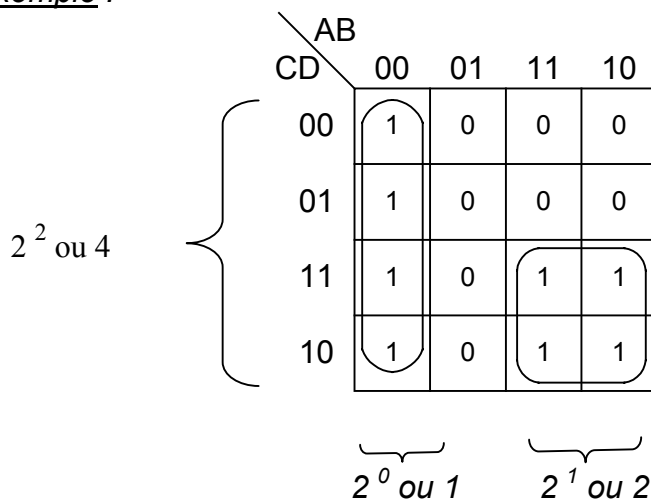
Règle 2 :

Les cases appartenant au même groupement doivent avoir la même valeur binaire de la variable de sortie. (voir les exemples précédents).

Règle 3 :

La longueur et la hauteur des groupements doivent être des puissances de deux.

Exemple :



Règles 4 :

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

Exemples :

A faire

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

A ne pas faire

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

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.

Exemples :

a)

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

b)

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

Règle 6 :

Les quatre cases des 4 coins d'un diagramme de Karnaugh peuvent être regroupées.

Exemples :

A faire

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

A ne pas faire

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

V.3 Écriture des équations à partir de regroupement

a) 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.

Règle : $B = 1 \implies$ On la représente par B ;
 $B = 0 \implies$ On la représente par \bar{B} .

Exemples :

a)

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

groupement 2 (points to the 1s in row 00, columns 01 and 11)
 groupement 1 (points to the 1s in row 01, columns 00, 01, 11, and 10)
 groupement 3 (points to the 1 in row 10, column 00)

Groupement 1 : A et B changent d'état

} \implies L'équation du groupement :
 $\bar{C} \cdot D$

$C = 0$ et $D = 1$ ne changent pas d'état

Groupement 2 : A et D changent d'état

} \implies L'équation du groupement :
 $B \cdot \bar{C}$

$B = 1$ et $C = 0$ ne changent pas d'état

Groupement 3 : $A = 0, B = 0, C = 1$ et $D = 0$
ne change pas d'état

} \implies L'équation du groupement :
 $\bar{A} \cdot \bar{B} \cdot C \cdot \bar{D}$

D'où l'équation finale : $S = \bar{C} \cdot D + B \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C \cdot \bar{D}$

b)

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

groupement 1 (points to the 1s in row 01, columns 11 and 10)
 groupement 2 (points to the 1s in row 11, columns 00, 01, 11, and 10)
 groupement 3 (points to the 1 in row 10, column 11)

$$\begin{array}{l}
 \text{Groupement 1 : } A \cdot D \\
 \text{Groupement 2 : } C \cdot D \\
 \text{Groupement 3 : } A \cdot B
 \end{array}
 \left. \vphantom{\begin{array}{l} \text{Groupement 1 : } A \cdot D \\ \text{Groupement 2 : } C \cdot D \\ \text{Groupement 3 : } A \cdot B \end{array}} \right\} \Rightarrow S = A \cdot D + C \cdot D + A \cdot B$$

b) Produit 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.

Règle : $B = 1 \Rightarrow$ On la représente par \bar{B} ;
 $B = 0 \Rightarrow$ On le représente par B .

Exemples :

a)

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

groupement 1 (circles around 0s in column 00) groupement 2 (circles around 0s in column 11)

Groupement 1 : A, B ne changent pas d'état ($A = 0, B = 0$)
 C et D changent d'état } \Rightarrow L'équation du groupement : $A+B$

Groupement 2 : A, B ne changent pas l'état ($A = 1, B = 1$)
 C et D changent } \Rightarrow L'équation du groupement : $\bar{A}+\bar{B}$

D'où l'équation finale : $S = (A+B) \cdot (\bar{A}+\bar{B})$

b)

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

Un seul groupement : A change d'état

B, C et D ne changent pas d'état
(B = 1, C = 1, D = 0)



L'équation du
groupement :
 $\overline{B+C+D}$

D'où l'équation finale : $S = \overline{B+C+D}$

VI. Circuits intégrés logiques

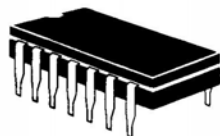
La matérialisation des fonctions logiques a été d'abord réalisée avec des composants discrets puis elle s'est transformée en intégrant plusieurs composants sur un seul circuit.

Un circuit intégré (C.I.) désigne un bloc constitué par un monocristal de silicium de quelques millimètres carrés en forme parallélépipède rectangle aplati, à l'intérieur duquel se trouve inscrit en nombre variable des composants électroniques (transistors, diodes, résistances et, plus rarement des condensateurs).

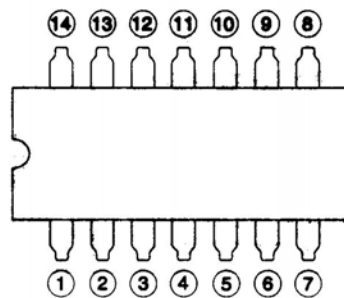
VI.1 Famille TTL (transistor transistor logic)

Elle fait principalement usage de combinaisons de transistors bipolaires pour la fabrication des circuits intégrés.

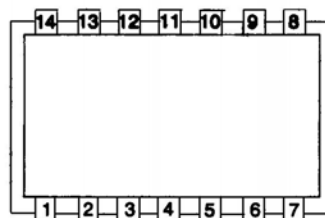
Ces C.I. sont constitués d'un boîtier qui contient la puce, laquelle est reliée à l'extérieur par un certain nombre de pattes (ou broches). Ce nombre varie généralement entre 14 et 28. La tension d'alimentation est +5V.



Vue en trois dimensions



Vue de dessus avec identification des pattes



Représentation généralement employée pour l'identification

Circuit intégré

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.

TTL standard	74XX
TTL low power (faible consommation)	74LXX
TTL schottky	74SXX
TTL fast	74FXX
TTL low power schottky	74LSXX
TTL advanced schottky	74ASXX
TTL advanced low power schottky	74ALSXX

VI.2 Famille CMOS (Complementary Métal Oxyde Semiconductor)

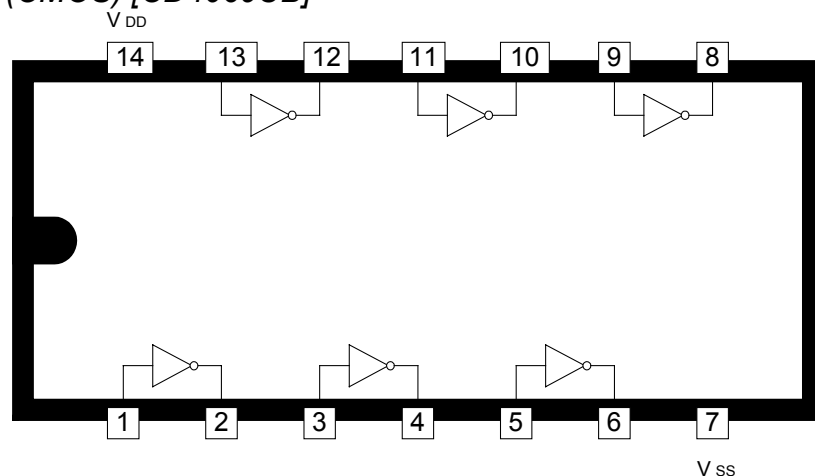
Elle dérive principalement des transistors à effet de champ. Cette famille se divise en deux sous groupes :

- 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)

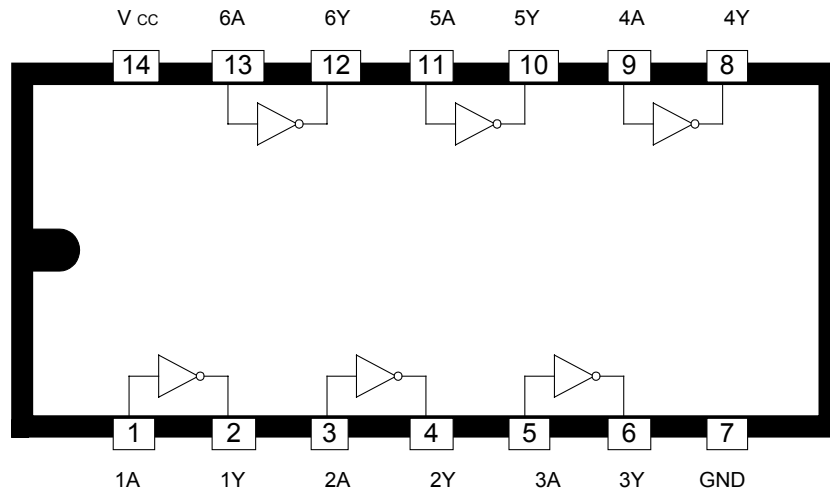
VI.3 Configuration des broches pour les différents modèles des C.I

a) Circuit intégré portes «NON»

- Six portes NON : 4069
(CMOS) [CD4069UB]

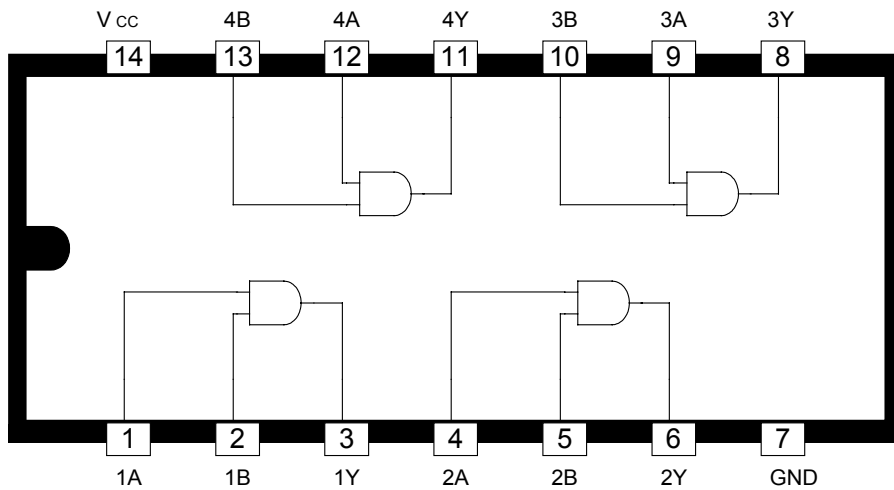


- Six portes NON : 7404
(TTL) [SN74LSD4]



b) Circuit intégré Portes «ET»

➤ Quatre portes «ET» à deux entrées : 7408 (TTL) [SN74LSD8]



➤ Trois portes «ET» à trois entrées : 7411 (TTL) [SN74LS11]

