BOULEMA Jean Bernard

E-mail: jbboulema@gmail.com

Tél: 077512298

BP : 13124

- **Ingénieur en informatique Industrielle** de l'école marocaine des sciences de l'ingénieur (**E.M.S.I**)
- **Bachellor en Entrepreneuriat** de l'institut des hautes études économiques et entrepreneuriales (**I.H.E.E**)
- **Master en conception des systèmes d'information** de l'institut africain de l'informatique (**I.A.I**)
- **Info preneur**

COURS D'ELECTRONIQUE NUMERIQUE

Introduction

Le 21e siècle est marqué par la grande présence des appareils et autres circuits électroniques se présentant sous différentes formes dans notre quotidien. Des ordinateurs super performants aux simples puces électroniques installées dans différents gadgets, le monde est, désormais, soumis aux avancées technologiques de ces petits circuits intégrés. En effet, presque tous les domaines sont impactés par les percées de l'électronique. Mais d'où nous viennent les puces électroniques ? Les circuits intégrés nous ont-ils présenté tout ce qu'ils peuvent apporter ?

Mais avant d'arriver aux circuits complexes que nous retrouvons dans les différents objets de notre quotidien ou des mondes professionnels, il a fallu toute une succession de découvertes pour permettre à l'électronique d'avoir cette place prépondérante dans les technologies.

La naissance de l'électronique

C'est avec l'invention de la diode à vide en 1897 par Fleming que l'histoire de l'électronique a réellement commencé. Cette invention a permis plus tard à Lee De Forest de mettre en œuvre une triode à vide permettant d'amplifier les signaux électriques. Par la suite, durant la Seconde Guerre mondiale, les tétrodes et pentodes ont longuement dominé le domaine des technologies jusqu'à l'avènement des transistors. L'ère des transistors a effectivement débuté en 1948 avec le transistor à jonction. Invention ayant obtenu un prix Nobel, elle a été vite délaissée au profit du tube à vide en raison de son caractère encombrant et particulièrement énergivore.

L'essor des circuits intégrés

La période d'après-guerre a connu l'invention des circuits intégrés qui ont tout simplement révolutionné la nature des circuits électroniques. C'est, en effet, avec l'arrivée des circuits intégrés que l'ensemble du circuit électronique a pu se trouver contenir dans une seule puce.
Cela a permis d'obtenir des dispositifs électroniques beaucoup plus petits, plus légers et surtout moins chers. Entre 1958 et 1975, l'introduction des circuits intégrés dans différents appareils a permis de doter ces derniers de capacités élargies. Ainsi, ils permettent de grandement simplifier des processus qui ont longtemps été considérés comme fastidieux.
Le début des années 2000 a connu le début de l'ère du numérique et des circuits intégrés numériques. Ces puces innovantes ont permis de complètement changer l'architecture globale des ordinateurs, les rendant plus performants, mais surtout plus légers et moins encombrants.

L'avenir de la puce électronique

Avec toutes les applications du circuit électronique numérique, tout laisse à penser que la puce électronique est la clé de nouvelles avancées technologiques dans pratiquement tous les domaines. Déjà, elle s'annonce comme la prochaine étape de l'avancée de la civilisation. Un implant, prévu pour être placé sous la peau, servira-t-il de carte bancaire, de billet de train et même de contrôleur des constantes vitales ? Certains le redoutent, d'autres le plébiscitent.
Dans le domaine de la santé, les circuits intégrés permettent de construire des robots grâce auxquels il est possible d'effectuer des opérations à distance. Utilisées à bon escient, les avancées de l'électronique sont porteuses de nombreuses promesses.

L’informatique et ses variantes

1. Le **génie informatique**, ou l'**ingénierie informatique**,
   est une discipline qui traite de la conception, du développement et de la fabrication de systèmes informatiques, aussi bien d'un point de vue matériels que logiciels

2. Le **génie logiciel**, l'**ingénierie logicielle** ou l'**ingénierie du logiciel**
   (en anglais : *software engineering*) est une science de génie industriel qui étudie les méthodes de travail et les bonnes pratiques des ingénieurs qui développent des logiciels. Le génie logiciel s'intéresse en particulier aux procédures systématiques qui permettent d'arriver à ce que des logiciels de grande taille correspondent aux attentes du client, soient fiables, aient un coût d'entretien réduit et de bonnes performances tout en respectant les délais et les coûts de construction[1].

3. **Les réseaux informatiques** sont orientés dans la transmission des données sous forme de fichier informatiques d'où le nom de téléinformatique. Cela se fait surtout en transmission asynchrone des données ou en transmission différée des données.

   Les réseaux des télécommunications sont orientés transmissions des informations sous forme analogique avec pour référence le téléphone. Surtout transmission synchrone des informations (transmission en temps réel)

4. L’**informatique de gestion** est l'ensemble des connaissances, des technologies, et des outils en rapport avec la gestion de données, c'est-à-dire la collecte, la vérification et l'organisation de grandes quantités d'informations. L'informatique de gestion a de nombreuses applications pratiques dans les entreprises : listes de clients, de fournisseurs, de produits, comptabilité, etc.

   En informatique de gestion, les informations sont souvent placées dans des bases de données et traitées par l'intermédiaire de logiciels spécialisés que sont les systèmes de gestion de base de données

5. **Les métiers de l'informatique industrielle**

L'informatique industrielle couvre l'ensemble des techniques de conception d'analyse et de programmation de systèmes à base d'interfaçage de l'informatique avec de l'électronique, électrotechnique, mécanique, robotique, etc., à vocation industrielle. Elle concerne l'utilisation de l'outil informatique pour la fabrication de produits industriels, du bureau d'études (conception assistée par ordinateur) à leur production (fabrication assistée par ordinateur, automatique, robotique) en passant par la logistique, la gestion des stocks, etc.

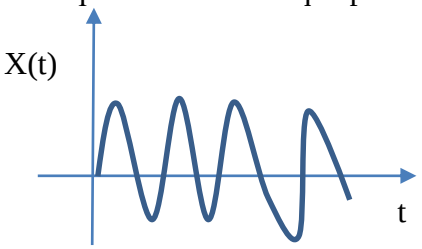
Définition

Électronique: partie de la physique qui étudie et utilise les variations de grandeurs électriques (***champs électromagnétiques, charges électriques, résistance électrique***, ***induction électrique, capacité électrique***..) pour ***capter, transmettre, exploiter et stocker*** de l'information.
Son domaine d'application est très vaste:

- ✓ ***Radio ;***
- ✓ ***Télévision ;***
- ✓ ***ordinateurs,***
- ✓ ***consoles de jeux ;***
- ✓ ***appareils photos ;***
- ✓ ***voiture ;***
- ✓ ***satellite*** ;
- ✓ ***objets connectés ;***
- ✓ ***crypto monnaies ;***
- ✓ ***block chaine ;***

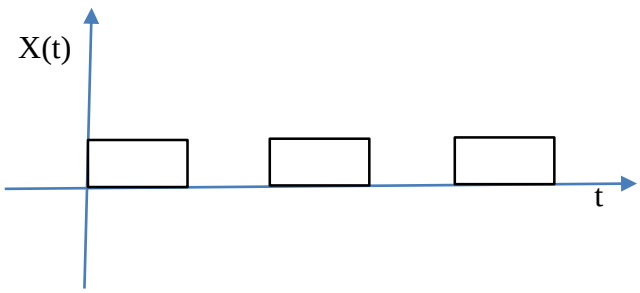
Nous parlons de ***signal analogique*** lorsque l'information produite par la source dispose d'une variation ou d'une gamme continue de ***nuances***. Il peut prendre une infinité de valeurs différentes dans une plage donnée et se transmet continuellement dans l'axe temps

Exemple l'onde électrique produite par un microphone :



Nous parlons de ***signal numérique***, ou digital, lorsque l'information produite par la source est représentée par un système conventionnel de signes distincts, tels que de chiffres dans notre cas : le binaire

Exemple l'onde électrique produite par l'horloge d'un microprocesseur :



Chaque information peut être représentée par un nombre; il suffit d'aligner rapidement les nombres les uns après les autres pour retrouver une image de l'information originale.

I°/ SYSTEME DE NUMERATION

Nous avons pris l'habitude de représenter les nombres en utilisant dix symboles différents: 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 Ce système est appelé le système décimal (déci signifie dix).
Il existe cependant d'autres formes de numération qui fonctionnent en utilisant un nombre de symboles distincts.

Dans un système de numération : le nombre de symboles distincts est appelé la base du système de numération.

Un ***système de numération*** est un ensemble de règles d'utilisation des signes permettant de représenter les nombres, ces derniers sont nés, en même temps que l'écriture

Un système de numération possède n symbole qui vont de 0 à n-1, n est appelé base de ce système de numération
Tout nombre ≥ 2 peut être considérer comme étant la base d'un un système de numération
Dans une base X, on utilise X symboles distincts pour représenter les nombres.
La valeur de chaque symbole doit être strictement inférieure à la base X.
Chaque nombre dans une base X peut être écrit sous sa forme polynomiale.

1. Système de numération décimal ou base 10

Le système de numération à base 10 est un moyen de représenter les nombres avec 10 symboles: N_{10} = {0, 1, 2, 3, 4, 5, 6, 7,8, 9}

Exemple : $(125)_{10}$ et $(-75,5)_{10}$ sont deux nombres écrits dans cette base

2. Système de numération binaire ou base 2

Le système de numération à base 2 est un moyen de représenter les nombres avec 2 symboles: N_{10} = {0, 1}

Exemple : $(1010)_2$ et $(11,0101)_2$ sont deux nombres écrits dans cette base

3. Système de numération octal ou base 8

Le système de numération à base 8 est un moyen de représenter les nombres avec 8 symboles N_8 = {0, 1, 2, 3, 4, 5, 6, 7}

Exemple : $(0733)_8$ et $(73654,7556)_8$ sont deux nombres écrits dans cette base

4. Système de numération hexadécimal ou base 16

Le système de numération à base 16 est un moyen de représenter les nombres avec 16 symboles N_{16} = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Exemple : $(2AF)_{16}$, $(25C, 4BC)_{16}$

5. Transcodage ou Conversion d'un système de numération à un autre

Le transcodage est l'opération qui consiste à convertir un nombre d'une base X vers une autre base Y

1. *Base 10 vers base n*

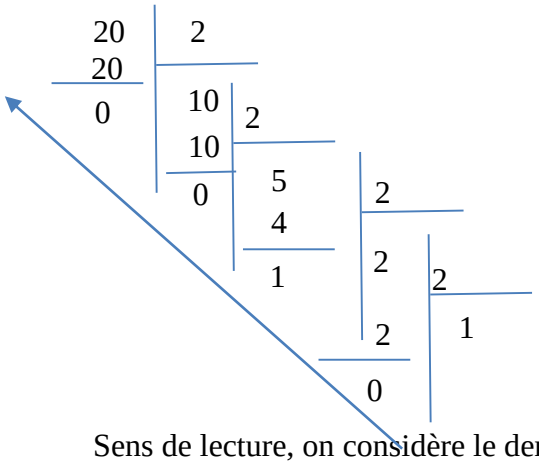
Le principe consiste à faire des divisions successives du nombre par n, et prendre le reste des divisions et le dernier quotient dans l'ordre inverse comme résultat

Pour la partie fractionnaire on multiplie par la base jusqu'à obtenir un résultat égal à 1 ou la précision souhaitée.

Exemple 1 base 10 vers base 2

Convertir le nombre $(20,4)_{10}$ en binaire

Pour la partie entière



Sens de lecture, on considère le dernier quotient et tous les restes

D'où la partie entière vaut : $(10100)_2$

Pour la partie fractionnaire on multiplie succ

Opérations	0,4x2	0,8x2	0,6x2	0,2x2	0,4x2	0,8x2	0,6x2
produits	0,8	1,6	1,2	0,4	0,8	1,6	1,2
Résultat	0	1	1	0	0	1	1

On considère

D'où $(20,4)_{10}=(10100,0110011)_2$

Exemple 2 base 10 vers base 8

Convertir le nombre $(675,75)_{10}$ en octal, on procède de la même manière que pour la base 2 sauf que la base ici c'est 8

D'où $(675,75)_{10}= (1243,8)_{8}$

Exemple 3 base 10 vers base 16

on procède de la même manière que pour la base 2 ou 8 sauf que la base ici c'est 16

$(1235,865)_{10} = X_{16}$ D'où $(1235,865)_{10} = (4D3, DD70A)_{16}$

2. *Base B vers base 10*

Cette conversion est assez simple puisque il suffit de faire le développement en polynôme de ce nombre dans la base B, et de faire la somme par la suite.

Notation polynomiale d'un nombre N en base B

$$\mathbf{N_B = a_{n-1}.B^{n-1}+a_{n-2}.B^{n-2}....a_1.B^1+a_o.B^o + b1.B^{-1}+b_2.B^{-2}+b_3.B^{-3}+....b_n.B^{-n}}$$

Exemple:

$(1101)_2 = 1x2^3+1x2^2+0x2^1+1x2^0 = 8+4+0+1 = \mathbf{(13)_{10}}$

$(1236)_8 = 1x8^3+2x8^2+3x8^1+6x8^0 = 512+128+24+6 = (670)_{10}$

$(1A7)_{16} = 1(16)^2+Ax(16)^1+7x(16)^0 = 256+160+7 = \mathbf{(423)_{10}}$

$(1101, 101)_2 = 1x2^3+1x2^2+0x2^1+1x2^0+1x2^{-1}+0x2^{-2}+1x2^{-3} = \mathbf{(13,625)_{10}}$

$(43,24)_5 = 4x(5)^1+3x(5)^0+2x(5)^{-1}+4x(5)^{-2} = \mathbf{(23,56)_{10}}$

3. Base 2^n vers base 2 (avec n=2, 3, 4,5…)

Chaque symbole de la base $B = 2^n$ peut être représenté par n éléments binaires.

Exemple

$(3A\ 9)_{16} = (0011\ 1010\ 1001)_2$

$(742,5)_8 = (111\ 100\ 010,101)_2$

$(1332)_4 = (01\ 11\ 11\ 10)_2$

4. Base 2 vers base 2^n

Il suffit de regrouper les éléments binaires par paquets de n.

Exemple

$(1\ 01\ 1\ 011)_2 = (001\ 011\ 011) = (133)_8$

$= (0101\ 1011) = (5B)_{16}$

$(1\ \ 01\ 10\ 11)_2 = (1123)_4$

5- Base i vers base j

– si i et j sont des puissances de 2, on utilise la base 2 comme relais ;

Exemple

base 8→base 2→base 16

– sinon, on utilise la base 10 comme relais.

Exemple

base 5→base 10→base 2

6. Travaux dirigés 1

Convertir le nombre décimal suivant $(79,85)_{10}$ dans les bases suivantes : X_2, X_8, X_{16}, X_6, X_4

Convertir les nombres suivants en décimal $(11101,101)_2$, $(2233,1123)_4$, $(7765,75)_8$, $(5FF, 2A)_{16}$

II°/ CODAGE DE L'INFORMATION

Le codage est l'opération qui établit une correspondance entre un ensemble source (nombre, caractère, symbole) vers un ensemble destination contenant des combinaisons de 0 et de 1.

Code binaire pur ou binaire naturel

Le code binaire pur est un code pondéré par des puissances de 2, utilisé en arithmétique binaire. Ses dérivées sont le code octal et le code hexadécimal.
Un code pondéré est un code qui peut être écrit sous sa forme polynomiale, c'est-à-dire qui possède une base et des symboles

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

2. Le Code DCB (décimal codé binaire)

Dans le code DCB, chaque chiffre décimal (0,1, . . . ,9) est codé en binaire avec 4 éléments binaires.

Remarque

Il ne faut pas confondre le code **DCB** et le code binaire pur : quand on code selon le code binaire pur on prend le nombre dans son intégralité et on le convertit ; par contre, quand on code en DCB on code chaque chiffre indépendamment les uns des autres.

Décimal	DCB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001
12	0001 0010
13	0001 0011
14	0001 0100
15	0001 0101
16	0001 0110

Exemple : $(7895)_{10}$

$= (0111\ 1000\ 1001\ 0101)_{DCB}$

3. Le Code Gray

La propriété principale du code gray est que le passage d'un mot-code au suivant entraînera toujours le changement d'un bit et d'un seul.
Ainsi, les transitions s'effectuent sans ambiguïté, éliminant les risques d'aléas. En revanche le code gray n'est pas pondéré, il n'est donc pas adapté pour le calcul numérique. Tout comme le binaire naturel, le code Gray peut coder n'importe quel nombre entier naturel

Le lien entre un nombre n codé en Gray un nombre N codé en binaire naturel est le suivant : $\mathbf{n = (N \oplus 2N) /2}$

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

Pour convertir un nombre binaire en code Gray, il existe plusieurs méthode nous allons plutôt utiliser celle de la formule précédente soit : n= (N⊕ 2N)/2N

Exemple : $(11)_{10}=(1011)_2= X_G$

Calculons X_G : N=01011 ; 2N= 10110 (car multiplier un nombre par sa base revient à ajouter un zéro à la droite du nombre) d'où

N	0	1	0	1	1
2N	1	0	1	1	0
⊕	1	1	1	0	1
n	1	1	1	0	1

D'où $n=X_G=(1110)_G$

4. Le code ASCII

La norme ASCII est largement utilisée en informatique pour coder les caractères. Ce nom provient de l'acronyme anglais '' ***American standard code for information inter change*** '' qui signifie en français '' ***code américain normalisé pour l'échange d'information*** l'ASCII est une norme de codage la plus répandu et compatible avec plus de support. Il contient l'ensemble des caractères alphanumériques utilisé en anglais. Initialement codé sur 7 bits (plus un code de parité). L'ASCII étendu est néanmoins codé sur 8 bits dans le but d'ajouter des caractères, tel que des caractères spéciaux et les lettres accentuées utilisée en français

Voir tableau de caractères ASCII

Décimal	Octal	Hex	Binaire	Caractère	
000	000	00	00000000	NUL	(Null char.)
001	001	01	00000001	SOH	(Start of Header)
002	002	02	00000010	STX	(Start of Text)
003	003	03	00000011	ETX	(End of Text)
004	004	04	00000100	EOT	(End of Transmission)
005	005	05	00000101	ENQ	(Enquiry)
006	006	06	00000110	ACK	(Acknowledgment)
007	007	07	00000111	BEL	(Bell)
008	010	08	00001000	BS	(Backspace)
009	011	09	00001001	HT	(Horizontal Tab)
010	012	0A	00001010	LF	(Line Feed)
011	013	0B	00001011	VT	(Vertical Tab)
012	014	0C	00001100	FF	(Form Feed)
013	015	0D	00001101	CR	(Carriage Return)
014	016	0E	00001110	SO	(Shift Out)
015	017	0F	00001111	SI	(Shift In)
016	020	10	00010000	DLE	(Data Link Escape)
017	021	11	00010001	DC1	(XON) (Device Control 1)
018	022	12	00010010	DC2	(Device Control 2)
019	023	13	00010011	DC3	(XOFF) (Device Control 3)
020	024	14	00010100	DC4	(Device Control 4)
021	025	15	00010101	NAK	(Negative

042	052	2A	00101010	*	(asterisk)
043	053	2B	00101011	+	(plus)
044	054	2C	00101100	,	(comma)
045	055	2D	00101101	-	(minus or dash)
046	056	2E	00101110	.	(dot)
047	057	2F	00101111	/	(forward slash)
048	060	30	00110000	0	
049	061	31	00110001	1	
050	062	32	00110010	2	
051	063	33	00110011	3	
052	064	34	00110100	4	
053	065	35	00110101	5	
054	066	36	00110110	6	
055	067	37	00110111	7	
056	070	38	00111000	8	
057	071	39	00111001	9	
058	072	3A	00111010	:	(colon)
059	073	3B	00111011	;	(semi-colon)
060	074	3C	00111100	<	(less than sign)
061	075	3D	00111101	=	(equal sign)
062	076	3E	00111110	>	(greater than sign)
063	077	3F	00111111	?	(question mark)
064	100	40	01000000	@	(AT symbol)
065	101	41	01000001	A	
066	102	42	01000010	B	
067	103	43	01000011	C	

5. Représentation des nombres à l'intérieur des machines numériques

1. Nombres entiers

Un nombre entier est représenté à l'intérieur de la mémoire d'un ordinateur par son code binaire s'il est positif par son complément à 2 s'il est négatif le tout codé sur 2^n bit

Exemple les nombres 9 et -9 sur 8 bits

$(9)_{10}$

0	0	0	0	1	0	0	1

1.1 Complément à 1 et à 2 d'un nombre binaire

Le complément à 1 d'un nombre binaire est l'opération qui consister à complémenter tous ses bit exemple $C_1(10101100)= (01010011)$

Ainsi si X=1 son complémentaire $\overline{X}=0$; de même si $\overline{X}=1$ son complémentaire X=0

ici X est appelé bit

Un bit (binary digit) est un chiffre d'un nombre binaire il ne peut prendre que 2 valeurs (0 ou 1), à ne pas confondre avec le **BYTE** qui vaut un octet c'est-à-dire un ensemble de 8bit

Rappel : 1 octet=8bit c'est l'unité de mesure de la capacité mémoire en informatique

Ainsi :

1KO (kilo octet)=1KB=2^{10}

1MO= 1MB=2^{20}

1GO=1MB=2^{30}

1TO=1TB=2^{40}

Le complément à 2 d'un nombre binaire est le complément à 1 de ce nombre plus 1

$(-9) \rightarrow c_2(9) = C_1(9) +1= \overline{00001001}+1$

1	1	1	1	0	1	1	1

2. Représentation des nombres relatifs

Pour représenter un nombre relatif (ayant une partie fractionnaire) on une utilise la représentation des nombres en norme **IEEE**

NB : en informatique un nombre entier peut avoir un signe et un nombre relatif est un nombre fractionnaire

On convertit le nombre décimal en binaire et on l'écrit sous la forme A=1,……2^E

Exemple $(-12,75)_{10} = (1100,11)_2= 1,10011x2^3$

On représente la mantisse sur 24 bits (la mantisse est la partie fractionnaire du nombre convertis sous la forme 1,……2^E)

L’exposant sur 7 bits

Le signe sur 1 bit

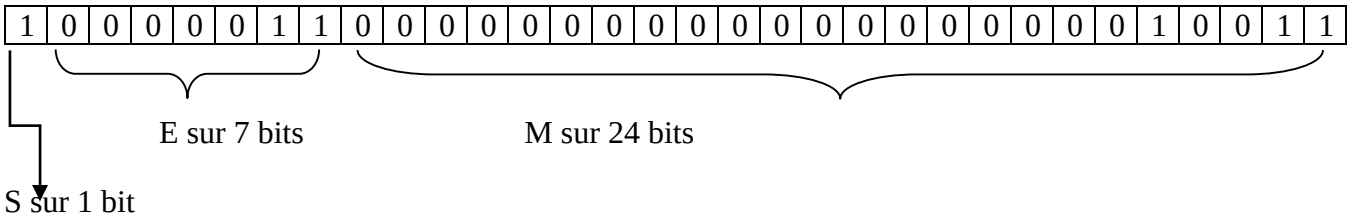
Le tout sur 32 bits

Dans notre cas nous aurons :

S=1 sur 1 bit

M=1001 sur 24 bits

E=$(3)_{10}$=$(11)_2$ sur 7 bits



III° /ALGEBRE DE BOOLE ET FONCTIONS LOGIQUES BINAIRES

Définition:

Algèbre booléenne : est une algèbre binaire mise en œuvre par ***le mathématicien anglais Georges BOOLE (1815-1864)*** pour étudier la logique, les variables booléennes ne peuvent prendre que deux valeurs **vrai ou faux** bien encore **1 ou 0** on peut alors définir des opérateurs (+ et**.**) sur ces variables et consigner le résultat dans une table de vérité ces opérateurs peuvent être réalisés par des circuits électroniques appelés portes logiques

1- Regle de l'algèbre de BOOLE

Somme Logique

a+0 = a
a+1= 1
a+a =a
a+$\bar{a}$=1

Produit logique

a.0=0
a.1=a
a.a=a
a$\bar{a}$= 0

commutativité

a+b+c = a+c+b
a.b.c = a.c. b
Associative
a+b+c = a+(b+c)
abc = a (bc)

Règle du multiple du complément

a+$\bar{a}$ b= a+b
$\bar{a}$ + ab = $\bar{a}$+b
$\bar{a}$ +a$\bar{b}$= $\bar{a}$ + $\bar{b}$
Démonstration : (a (b+1) +$\bar{a}$ b)= (ab+a+$\bar{a}$ b)=ab+a+b=a (b+1) +b= a+b

Règle d'absorption

a+ab=a
a(a+1)=a

2 - Théorème de l'algèbre de BOOLE

Théoreme de **MORGAN**
$(\overline{a + b}) = (\bar{a}.\bar{b})$
$(\overline{ab}) = (\bar{a}+\bar{b})$

3. Simplification des fonctions logiques

1-Simplification par les méthodes algébriques

Pour simplifier algébriquement une fonction logique on utilise les règles et le théorème de l'algèbre de **BOOLE**

2-Travaux dirigés 2

$f1 = \overline{a}\, b + ab + a\overline{b}\,\overline{c}d + a\overline{b} + \overline{b}cd + \overline{c}d$

$f2 = (\overline{a}\, b + \overline{a}\, bc + a\overline{b})\,(\overline{b}c + \overline{c}d)$

$f3 = \overline{a}\, b + abc + (\overline{b}c + \overline{c}db)\, a\overline{b}$

$f4 = \overline{a}\,\overline{b}\,\overline{c} + \overline{a}\, b\overline{c} + ab\overline{c} + \overline{a}\, bc + abc + a\,\overline{b}c$

$f5 = \overline{a}\,\overline{b}\,\overline{c}\,\overline{d} + \overline{a}\,\overline{b}\,\overline{c}\, d + \overline{a}\,\overline{b}cd + \overline{a}bcd + abcd + a\,\overline{b}cd + a\,\overline{b}c\,\overline{d}$

$f1 = b(\overline{a} + a) + a\overline{b}(\overline{c}d + 1) + d(\overline{b}c + \overline{c})$

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

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

$= a + b + d + d\,\overline{b} + d\overline{c} = a + b + d(1 + \overline{b}) + d\overline{c}$

$= a + b + d + d\overline{c} = a + b + d(1 + \overline{c}) = a + b + d$

$f1 = a + b + d$

$f2 = (\overline{a}\, b + \overline{a}\, bc + a\overline{b})\,(\overline{b}c + \overline{c}d)$

$f2 = (\overline{a}\, b\,(1 + c) + a\overline{b})\,(\overline{b}c + \overline{c}d)$

$f2 = (\overline{a}\, b + a\overline{b})\,(\overline{b}c + \overline{c}d)$

$f2 = (\overline{a}\, b + a\overline{b})\,(\overline{b}c + \overline{c}d)$

$f2 = (0 + \overline{a}\, b\overline{c}d + a\overline{b}c + a\overline{b}\,\overline{c}d)$

$f2 = (\overline{a}\, b\overline{c}d + a\overline{b}c + a\overline{b}\,\overline{c}d)$

$f2 = (\overline{a}\, b\overline{c}d + a\overline{b}(c + \overline{c}d))$

$f2 = (\overline{a}\, b\overline{c}d + a\overline{b}(c + d))$

$f2 = \overline{a}\, b\overline{c}d + a\overline{b}c + a\overline{b}d$

$f3 = \overline{a}\, b + abc + (\overline{b}c + \overline{c}db)\, a\overline{b}$

$f3 = \overline{a}\, b + abc + a\overline{b}\,\overline{b}c + \overline{c}dba\overline{b}$

$f3 = \overline{a}\, b + abc + a\,\overline{b}c + 0$

$f3 = \overline{a}\, b + ac(b + \overline{b})$

$f3 = \overline{a}\, b + ac$

2-Simplification par la méthode graphique de KARNAUGH

La simplification des fonctions logiques par la méthode de KARNAUGH a quand même certaines contraintes :

Si la fonction possède n variables, dans chaque terme de la fonction les n variables doivent figurer dans le cas échéant, la simplification risque de donner un résultat moins compact que la fonction du départ

Exemple d'application *f4 et f5*

$f4=\overline{a}\,\overline{b}\overline{c}+\overline{a}\,b\overline{c}+ab\overline{c}+\overline{a}\,bc+abc+a\,\overline{b}c$

$f5=\overline{a}\,\overline{b}\overline{c}\,\overline{d}+\overline{a}\,\overline{b}\overline{c}\,d+\overline{a}\,\overline{b}cd+\overline{a}bcd+abcd+a\,\overline{b}cd+a\,\overline{b}c\,\overline{d}$

Resolution algébrique

$f\,4=\overline{a}\,\overline{c}(\overline{b}+b)+bc(a+\overline{a})+a(\overline{b}c+b\overline{c})$

$f\,4=\overline{a}\,\overline{c}+bc+a\,\overline{b}c+ab\overline{c}$

$f\,4=\overline{a}\,\overline{c}+a\,\overline{b}c+bc+ab=c(a\,\overline{b}+b)+ab+\overline{a}\,\overline{c}=$

$ac+ab+\overline{a}\,\overline{c}$

$f5=\overline{a}\,\overline{b}\overline{c}\,\overline{d}+\overline{a}\,\overline{b}\overline{c}\,d+\overline{a}\,\overline{b}cd+\overline{a}bcd+abcd+a\,\overline{b}cd+a\,\overline{b}c\,\overline{d}$

Exemple d'application *f4 fonction à 3 variables :*

$f4=\overline{a}\,\overline{b}\overline{c}+\overline{a}\,b\overline{c}+ab\overline{c}+\overline{a}\,bc+abc+a\,\overline{b}c$

c \ ab	00	01	11	10
0	1	1	1	
1		1	1	1

$f_4=\overline{a}\,\overline{c}+b+ac$

$f'4=a\,\bar{b}\bar{c}+\bar{a}\,b\bar{c}+\bar{a}\,\bar{b}c+abc+\;ab\bar{c}$

c \ ab	00	01	11	10
0		1	1	1
1	1		1	

$f'4=ab+b\bar{c}+a\bar{c}+\bar{a}\,\bar{b}c$

$f5=\bar{a}\,\bar{b}\bar{c}\,\bar{d}+\bar{a}\,\bar{b}\bar{c}\,d+\bar{a}\,\bar{b}cd+\bar{a}bcd+abcd+a\,\bar{b}cd+a\,\bar{b}c\,\bar{d}$

cd \ ab	00	01	11	10
00	1			
01	1			
11	1	1	1	1
10				1

f'5=cd+$a\,\bar{b}c+\bar{a}\,\bar{b}d+\bar{a}\,\bar{b}c$

f'5= $\bar{a}\,\bar{b}cd+\bar{a}bcd+abcd+a\,\bar{b}cd+a\,\bar{b}c\,\bar{d}$

f'5=ab+$a\,\bar{b}c$

cd \ ab	00	01	11	10
00				
01				
11	1	1	1	1
10				1

4. Représentation des fonctions Logiques à l'aide des les portes Logiques

Une fonction logique est l'une des notions de l'algèbre de **BOOLE** c'est-à-dire qu'elle ne peut prendre que deux valeurs o ou 1
Une porte logique est un opérateur qui permet de réaliser les opérations logiques (. et +) ces portes se trouvent dans des circuits intégrés de technologie CMOS ou TTL

1-La fonction logique NOT (non ou pas)

C'est une fonction qui inverse toute valeur présente à son entrée

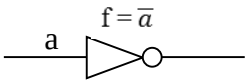
Tableau de vérité

a	f
0	1
1	0

Equation logique

$f = \overline{a}$

représentation graphique à l'aide d'une Porte NOT



2-La fonction logique AND (ET)

C'est une fonction qui fait le produit logique de toutes variables présentes à ses entrées

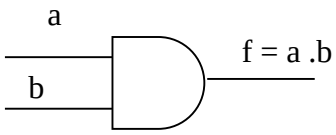
Tableau de vérité

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

Equation logique

f = a .b

représentation graphique à l'aide d'une **Porte AND**



3-La fonction logique OR (OU)

C'est une fonction qui fait la somme logique de toutes variables présentes à ses entrées

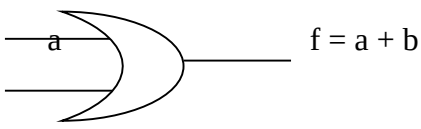
Tableau de vérité

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

Equation logique

f = a +b

représentation graphique à l'aide d'une Porte OR



4-La fonction logique NAND (NON ET)

C'est une fonction qui fait le produit logique complémenté de toutes variables présentes à ses entrées

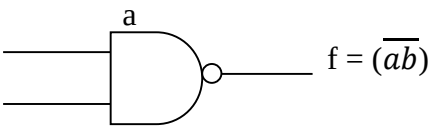
Tableau de vérité

a	b	f
0	0	1
0	1	1
1	0	1
1	1	0

Equation logique

$f = (\overline{ab})$

représentation graphique à l'aide d'une Porte NAND



3- La fonction logique NOR (NON OU)

C'est une fonction qui fait la somme logique complémenté de toutes variables présentes à ses entrées

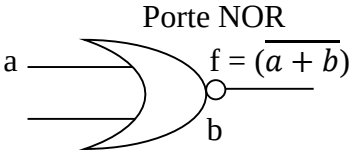
Tableau de vérité

a	b	f
0	0	1
0	1	0
1	0	0
1	1	0

Equation logique

$f = (\overline{a+b})$

représentation graphique à l'aide d'une Porte NOR



5-La fonction logique XOR (ou exclusif)

C'est une fonction qui vaut 1 lorsque les deux variables d'entrées sont complémentaires

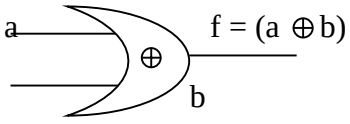
Tableau de vérité

a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

Equation logique

$f = \overline{a}\,b + a\overline{b} = (a \oplus b)$

représentation graphique à l'aide d'une Porte XOR



6-La fonction logique $\overline{XOR}$ (non ou exclusif)

C'est une fonction qui vaut 1 lorsque les deux variables d'entrées sont égales

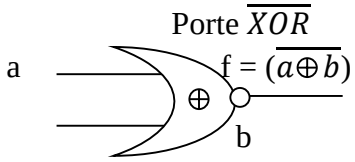
Tableau de vérité

a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

Equation logique

$f = (\overline{a \oplus b})$

représentation graphique à l'aide d'une Porte $\overline{XOR}$



7-Travaux dirigés 3

1. Représenter graphiquement les fonctions logiques suivantes

$X_1=\overline{x}\ \overline{y}\ \overline{z}+\overline{x}y+xz+z\ \overline{x}$

$X_2=\overline{x}\ \overline{y}$ +xyzw+$\overline{xyzw}$+x$\overline{yz}$+yz

2. Représenter graphiquement les fonctions logiques simplifiées f1, f2, f3, f4 et f5

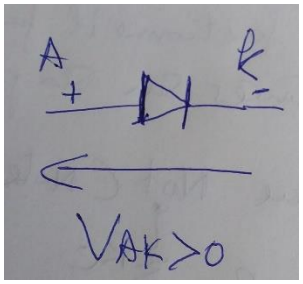
8- réalisation des portes logiques (logic gate) à l'aide des transistors

Définition :

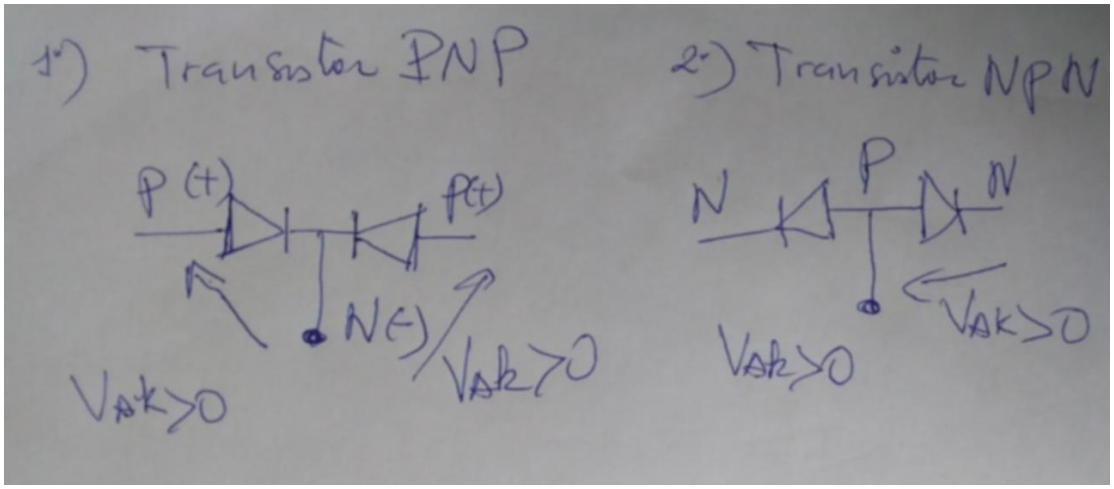
Un transistor est un semi-conducteur dopé (**NPN** ou **PNP**), un semi-conducteur est un conducteur qui conduit dans un seul sens donnée en fonction de sa polarisation

Un autre semi-conducteur très connue est la diode, une diode est un semi-conducteur Dopé **PN**

8.1. Polarisation de la diode



8.1. Polarisation du transistor



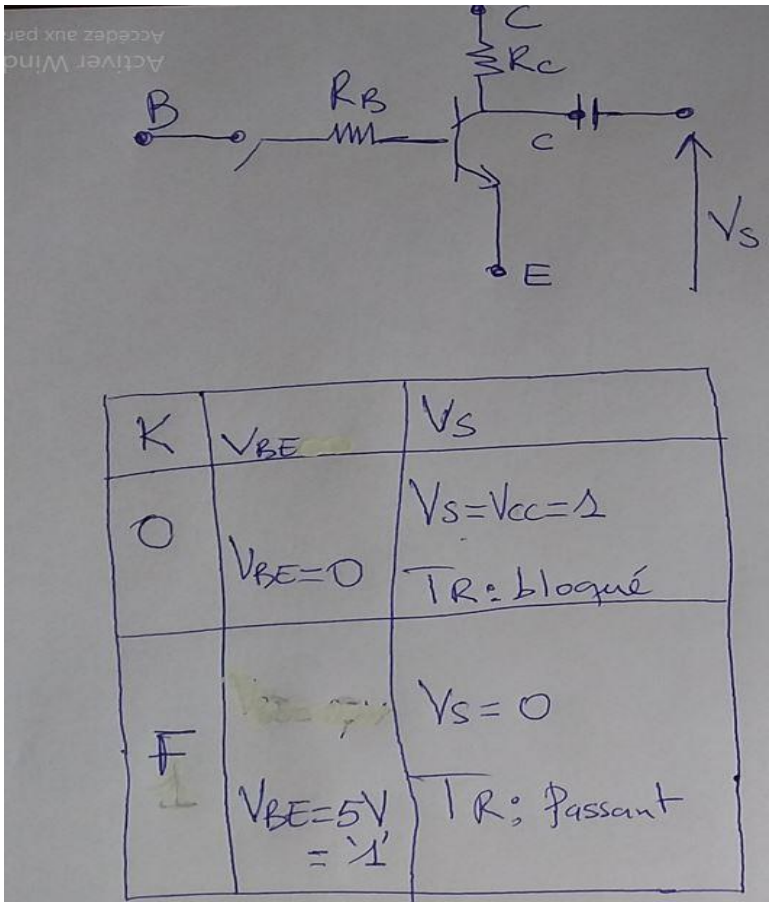
La plus part des semi-conducteur sont fait à base de silicium (sable) ou de germanium la tension de seuil ou dite de coude est de 0,7V pour le silicium et 0,3V pour le germanium :

0,7 V ou 0,3V est la tension à partir de laquelle le semi-conducteur commence à conduire le courant

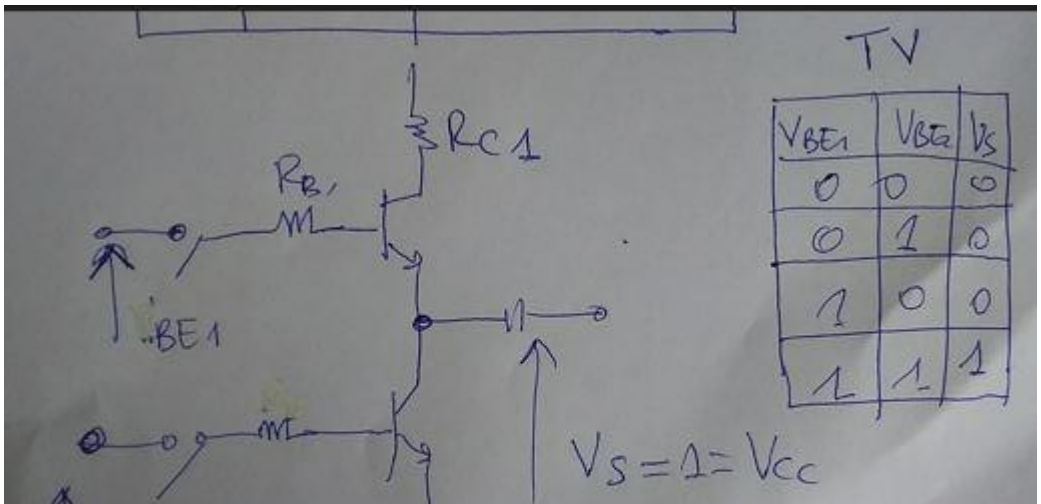
Pour réalise une porte logique à l'aide des transistors on utilise le transistor en mode *commutation :*

Il existe deux de fonctionnement pour un transistor :
Le mode commutation (interrupteur commandé)
Le mode amplification (augmentation de la tension ou du courant ou des deux donc de la puissance)

8.3 Réalisation de la porte logique NOT (NOT Gate)



8.4 Réalisation de la porte logique **AND** (AND Gate)



8.5 Réalisation de la porte logique **OR** à l'aide des transistors bipolaires (OR Gate)

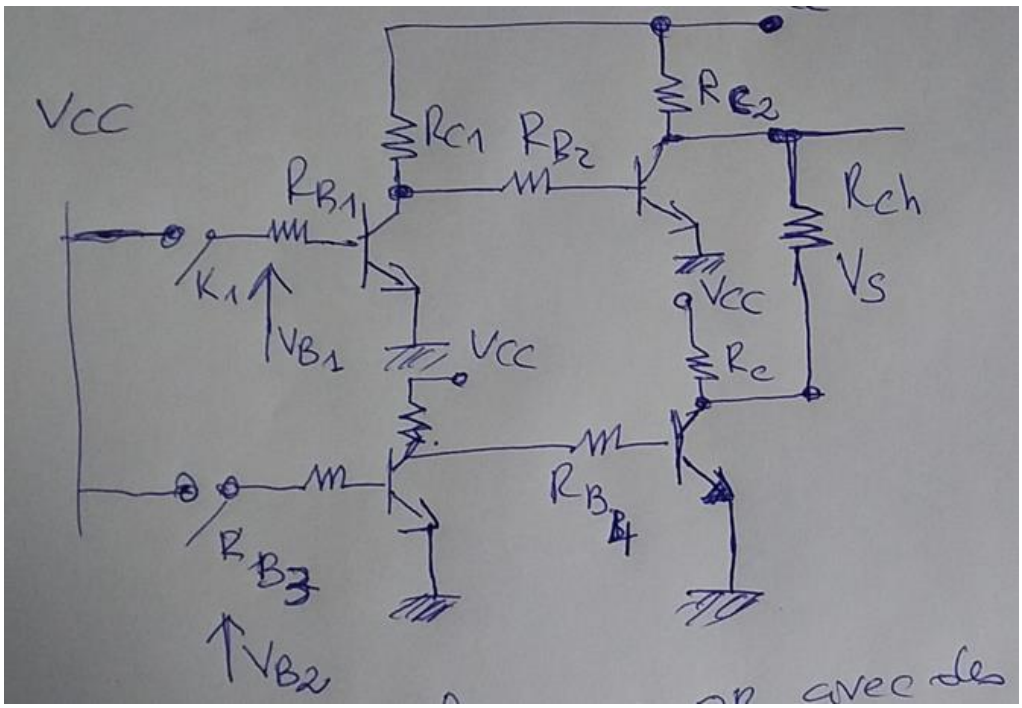


Tableau de vérité

Vb1(v)	Vb2(v)	Vs
0	0	0
0	5	Vcc
5	0	Vcc
5	5	Vcc

8.4 Réalisation de la porte logique AND (AND Gate)

IV°/ PROBLEMES COMBINATOIRES

Dans la pratique l'électronicien sera toujours amené à résoudre des problèmes réels qui n'ont pas de solution en algèbre classique, d'où l'intérêt de faire intervenir l'algèbre de **BOOLE** et les circuits logiques (détecteur de présence, compteur d'objet, détecteur de parité, clavier numérique, vote électronique, objets connectés ….)

1- Concevoir un circuit logique détecteur de nombres premiers, inférieur ou égal à 8

- ✓ Donner la table de vérité,
- ✓ Tracer le tableau de Karnaugh,
- ✓ Après groupement, donner l'équation minimum,
- ✓ Dessiner le schéma logique,
- ✓ Réaliser le même montage en utilisant que des fonctions **NAND.**

2- Concevoir un circuit logique permettant de contrôler une lampe en trois endroits différents

3-Trois équipements : a, b, c, fonctionnent ensemble. Si, au moins deux de ces équipements tombent en panne, on désire alimenter automatiquement un élément de secours X.

Le niveau logique **"1"** indiquera le bon fonctionnement,
Le niveau logique **"0"** indiquera le mauvais fonctionnement.

- ✓ Donner la table de vérité,
- ✓ Tracer le tableau de Karnaugh,
- ✓ Après groupements, donner l'équation minimum,
- ✓ Dessiner le schéma logique,
- ✓ Réaliser le même montage en utilisant que des fonctions **NAND.**

Comment remplir le tableau de vérité ?

En fonction du cahier de charge on remplit le tableau de vérité :

- ✓ La 1ère ligne contient le nom des variables d'entrées et de sortie
- ✓ Les autres lignes contiennent les valeurs (1 ou 0) que peuvent prendre les variables d'entrés et de sortie c'est-à-dire 2^n combinaisons et la sortie évoluera en fonction du cahier de charge en générale ce tableau est rempli de 0 à n-1 en binaire pur
- ✓ L'équation s'obtient en considérant les lignes ou la fonction de sortie vaut 1 et chaque ligne comportera un terme qui va former l'équation finale qui devra être simplifié à la fin soit par la méthode algébrique ou par la méthode graphique de **KARNAUGHT**

V°/ LES CIRCUITS LOGIQUES COMBINATOIRES

Les fonctions logiques « ***combinatoires*** », bases du calcul booléen, qui résultent de l'analyse combinatoire des variations des grandeurs d'entrées uniquement

les fonctions logiques « ***séquentielles*** » ou bascules, qui résultent de l'association de plusieurs fonctions logiques « combinatoires » synchronisées grâce à une « horloge » qui donne le tempo ; les valeurs de sorties dépendent non seulement des valeurs d'entrée, mais aussi de l'instant où elles sont mesurées (avant ou après la synchronisation par l'horloge).

Dans la pratique ce sont des circuits intégrés numériques qui permettent de réaliser des opérations arithmétiques et logiques (l'additionneur, le comparateur, le multiplexeur…)

1- Les opérateurs de transcodage

1-1 Le CODEUR

Un codeur est un circuit qui fonctionne tel que lorsqu'une entrée (sur les N) est activée, les sorties affichent le numéro de l'entrée active dans le code binaire choisi (sur n bits)

Schéma bloc

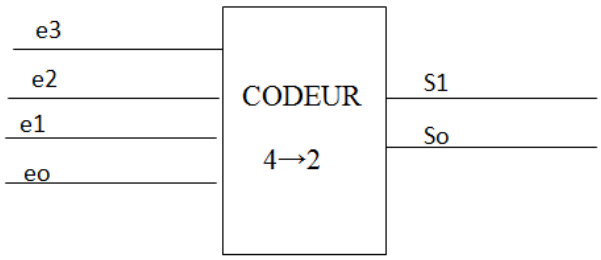


Table de vérité

e3	e2	e1	e0	So	S1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Equation logique

So = $\overline{e1}\ \overline{e0}$ (e2 $\oplus$ e3)

S1 = $\overline{e0}\ \overline{e2}$ (y1 $\oplus$ y3)

Circuit logique du codeur

1-2 le DECODEUR

Un décodeur (n entrées de données et N sorties) est un circuit qui fonctionne tel qu'une seule sortie est active à la fois quand un nombre est codé en binaire pur à l'entrée, c'est la sortie correspondante qui est activée

Schéma bloc

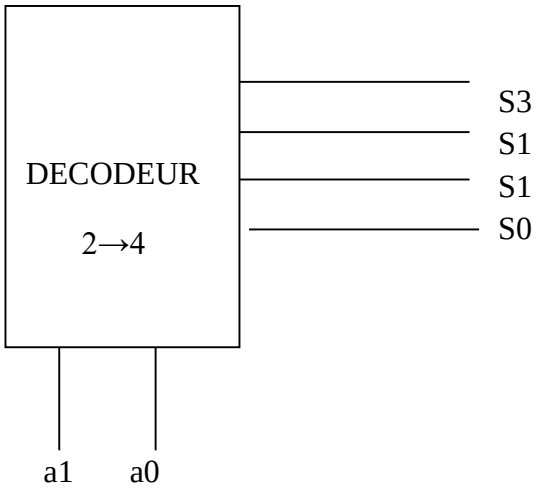


Table de vérité

a_1	a_0	S_3	S_2	S_1	S_o
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Equation logique

$s_o = \overline{a1}\,\overline{ao}$

$s_1 = \overline{a1}\,ao$

$s_2 = a1\,\overline{ao}$

$s_3 = a1\,ao$

2- Le multiplexeur

Récupérer une des 2^n entrées, l'aiguiller vers sur la sortie S à l'aide de n bites d'adresse

Schéma Bloc Tableau de vérité

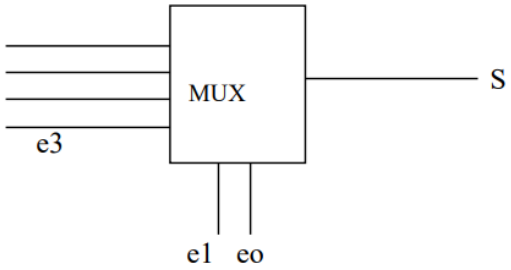


Tableau de vérité

a1	a0	ei	S
0	0	eo	1
0	1	e1	1
1	0	e2	1
1	1	e3	1

Equations logiques :

S= eo$\overline{ao}$ $\overline{a1}$+ e1$\overline{ao}$ a1+ e2ao$\overline{a1}$+ e3aoa1

3- Le démultiplexeur : c'est l'inverse du multiplexeur : Aiguiller un signal d'entrée E vers l'une des 2^n sortie à l'aide de n bits d'adresse

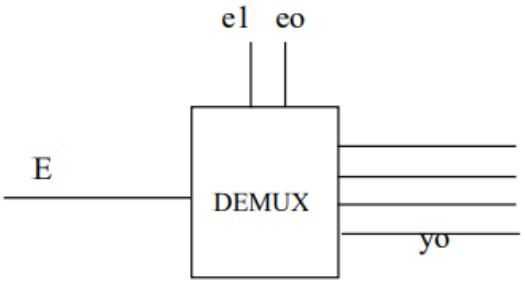


Tableau de vérité

E	e1	eo	**y3**	**y2**	**y1**	**yo**
x	0	0	0	0	0	1
x	0	1	0	0	1	0
x	1	0	0	1	0	0
x	1	1	1	0	0	0

Equations logiques :

yo = $\overline{eo}$ $\overline{e1}$ .E
y1 =$\overline{e1}$ eo.E
y2 =e1$\overline{eo}$ .E
y3 =e1eo.E

4- Additionneur binaire

Circuit électronique capable de réaliser la somme des nombres présents à ses entrées selon les règles de l'addition binaire :

Cas de deux nombres binaires de 1bit chacun

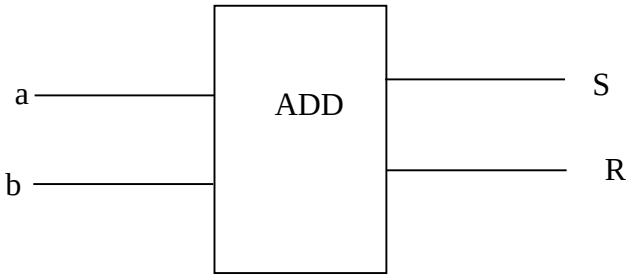


Tableau de vérité

b	a	**S**	**R**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Equations logiques :

Circuit logique:

$S = \overline{a}b + a\overline{b} = a \oplus b$

$R = ab$

5- Comparateur binaire

Circuit électronique capable de comparer deux nombres binaires présents à ses entrées selon les critères ci-après

(S=1 si a>b ; E=1 si a=b ; I=1 si a < b)

Cas de deux nombres binaires de 1bit chacun

Schéma bloc

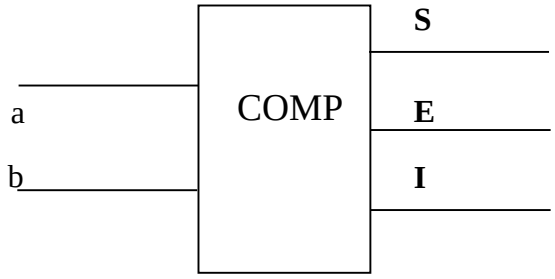


Tableau de vérité

b	a	**S**	**E**	**I**
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Equations logiques :

S = $a\overline{b}$; E= ab+ $\overline{b}\ \overline{a}$

I= $\overline{a}b$