OPÉRATEURS LOGIQUES DE BASE

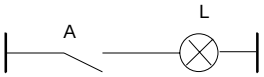
I/ QUELQUES DÉFINITIONS

Définition : On nomme **VARIABLE BINAIRE** tout phénomène qui ne peut prendre que deux états :

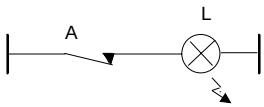
- **L'état logique 0** peut être associé à une affirmation fausse : absence de tension, actionneur non commandé, etc...
- **L'état logique 1** peut être associé à une affirmation vraie : tension présente, actionneur commandé, présence d'un phénomène, etc...

Trouvez l'état logique des variables binaires A et L.

A est un interrupteur ouvert au repos et L est une lampe.



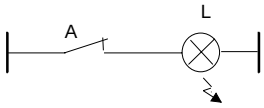
A= ___ L=_____



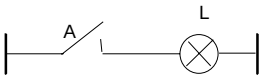
A=____ L=_____

Pour les schémas ci-contre, ***A*** est un interrupteur fermé au repos.

A= ___ L=___



A=____ L=_____



Définition : On appelle **NIVEAU LOGIQUE** en électronique une tension correspondant à un état logique.

- Ainsi la tension la plus élevée d'un circuit logique est généralement associée à l'état logique 1 : On dira qu'il s'agit du niveau logique 1 (NL1).
- Par opposition la tension la plus faible (le 0v souvent) est appelée niveau logique 0 (NL0).

Définition : On appelle **OPERATEUR LOGIQUE** un opérateur mathématique (*mis à jour par le mathématicien Georges BOOLE 1815-1864*) qui permet de lier des variables binaires en vue de décrire avec plus de précision un problème. En principe il n'existe que 3 opérateurs de base :

- ET
- OU
- NON

Grâce à ces trois opérateurs il est possible de décrire un problème simple sous forme d'équation.

L'opérateur *ET* (*AND* en anglais) est représenté dans une équation par le caractère *point* "." : **A *ET* B s'écrit A . B** et se lit **A ET B**
L'opérateur *OU* (*OR* en anglais) est formalisé par le caractère plus "+" : **A *OU* B s'écrit A + B** et se lit **A OU B**
L'opérateur *NON* se représente en surlignant la variable binaire ainsi **NON A s'écrit $\overline{A}$** et se lit **A barre.** Quelques fois **A barre** s'écrit également ***/*A**

Par extension le terme opérateur logique a été associé à des composants électroniques capables de réaliser ces opérations logiques. Ces circuits sont également appelés **PORTES LOGIQUES.**

Exemple d'équation logique :

La sonnerie (***SONNERIE***) du lycée retentit lorsqu'elle n'est pas défectueuse (***FONCTIONNE***) _____ lorsqu'elle doit signaler les débuts des cours (**DEBUT**) _____ la fin des cours (FIN) ____ une alarme incendie (ALARME).

Cette phrase peut se simplifier par une équation :

SONNERIE = FONCTIONNE __ (DEBUT _ FIN _ ALARME)

Définition : Une **TABLE DE VERITE** est un tableau décrivant le résultat d'une fonction logique en fonction de l'état de ses entrées

Compléter la table de vérité de la SONNERIE en vous servant de l'équation précédentes:

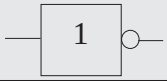
FONCTIONNE	DEBUT	FIN	ALARME	SONNERIE
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

II/ FONCTIONS LOGIQUES

II.1/ Fonction NON

Fonction NOT (NON) : La table de vérité ci-contre est celle d'une fonction **NOT.** L'équation de sortie d'une telle fonction est (si l'entrée est E) **S= /E**. Cette fonction est également appelée **INVERSEUSE** car elle complémente l'état logique d'entrée. Elle peut être réalisée avec une porte NOR ou une porte NAND dont les entrées sont toutes reliées entre elles. Son symbole est représenté ci-contre (***le petit cercle symbolise la complémentation***)

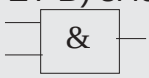
E	S
0	1
1	0



VHDL S<=not E

II.2/ Fonction ET

Fonction ET : La sortie d'une telle fonction est au niveau logique 1 lorsque toutes ses entrées sont au niveau logique 1.
Son équation est **S = A . B** (*lire S est égale à A ET B*) si les deux entrées s'appellent ***A*** et ***B***



VHDL S<=A and B;

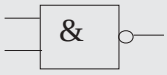
Donner la table de vérité d'une telle porte :

A	B	S
0	0	
0	1	
1	0	
1	1	

Donner le schéma électrique de cette fonction

II.3/ Fonction NAND (NON-ET)

Fonction NAND : Il s'agit de l'association d'un fonction ***ET*** et d'une fonction ***NON.*** Elle est dite universelle car elle permet de réaliser toutes les autres fonctions logiques de base. Sa représentation symbolique est la suivante :



VHDL S<=A nand B;

A partir de la table de vérité ci contre, donner les deux équations possibles pour cette fonction:

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

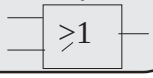
•S =
•S=

Pour que la sortie d'une porte NAND soit à 1 il suffit
_____

Schéma électrique:

II.4/ Fonction OU

Fonction OU : Son équation logique est **S = A + B** (***lire S est égale à A OU B***). Sa table de vérité est donnée ci-contre et sa représentation symbolique est la suivante :



VHDL S<=A or B;

Compléter la table de vérité :

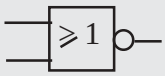
A	B	S
0	0	
0	1	
1	0	
1	1	

Donner le schéma électrique de cette fonction:

II.5/ Fonction NOR

Fonction NOR : La table de vérité ci-dessous est celle d'une fonction NOR. Cette fonction a pour équation de sortie (si A et B sont les deux entrées) **S=A + B** = $\overline{\mathbf{A}}$ **.** $\overline{\mathbf{B}}$ et elle est équivalente à une fonction OU suivie d'une fonction NON.
Cette fonction est dite universelle car elle permet de réaliser toutes les autres fonctions logiques de base.

Son symbole est celui d'une fonction OU complémentée ::



VHDL S<=A nor B;

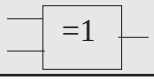
Donner le schéma électrique, équivalent à une porte NOR, comportant deux interrupteurs et un voyant :

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

Pour que la sortie d'une fonction NOR soit à 0, il suffit que

II.6/ Fonction XOR : OU exclusif

Fonction XOR : La fonction OU exclusif délivre un niveau logique seulement si une de ses entrée est au niveau logique 1.
Son équation logique est **S = A./B + /A.B**, on adopte également quelques fois la notation **S** = **A**$\oplus$**B**. La complémentation d'une fonction XOR permet d'obtenir une fonction égalité.
Représentation normalisée :



VHDL S<=A xor B;

Donner la table de vérité de cette fonction :

En déduire le schéma électrique équivalent :

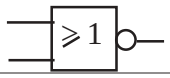
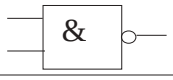
Remarque : On utilise quelques fois les portes XOR complémentées (XNOR) ce qui correspond à une fonction égalité d'équation :

S= /A./B + A.B

La sortie est au NL1 si les entrées sont identiques.

II.7/ Utilisation des portes universelles

Donner la structures à base de portes NAND et NOR des portes logiques de base

		
NON		
ET		
OU		
NOR		
NAND		
XOR		

III/ RÉSOLUTION DE PROBLÈMES DE LOGIQUE COMBINATOIRE

Une structure en **logique combinatoire** est une structure dont l'état logique de sortie ne dépend que d'une combinaison des états logiques d'entrée.

III.1/ Propriété des opérateurs de base de l'algèbre de Boole

Commutativité	A + B = B+ A	A . B = B . A	
Distributivité	A . (B + C) = A.B + A.C		
Elément neutre	A . 1 = A	A + 0 = A	
Antisymétrie	A . 0 = 0	A + 1 = 1	
Identités remarquables	A.A= A	A + A=A	/ (/A) = A
	A . /A = 0	A + /A = 1	
	A + A.B = A. (1 + B) =A		
	A + /A.B = A + B		

Théorèmes de DE MORGAN

$\overline{A \cdot B} = \overline{A} + \overline{B}$ $\qquad$ $\overline{A + B} = \overline{A} \cdot \overline{B}$

III.2/ Simplification d'une expression logique à l'aide des propriétés de l'algèbre de BOOLE

Lorsque l'on souhaite résoudre un problème complexe, on élabore dans un premier temps une table de vérité qui nous donne les combinaisons d'entrée-sortie de la fonction à réaliser. A partir de cette table, il est possible d'extraire une équation logique qui peut être assez longue si le nombre de variables d'entrées est important. Il faut alors simplifier cette équation pour arriver à la structure électronique la plus simple possible. Pour cela, il faudra utiliser les propriétés énoncées précédemment.

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

Exemple : Soit la table de vérité suivante :

L'équation de S est donc :

S=/A.B./C + /A.B.C + A./B.C+A.B./C + A.B.C

Simplifions :

S=B./C.(/A+A) + B.C.(A+/A) + A./B.C

=> S=B./C + B.C + A./B.C

=>S=B.(C+/C) + A./B.C

=>S=B + A./B.C

=>**S=B + AC**

Dans la pratique cette méthode est assez hasardeuse et difficile à utiliser au-delà de 3 variables d'entrées.

III.3/ Simplification par tableau de KARNAUGH

Un tableau de Karnaugh compte 2^n cases, si *n* est le nombre de variables binaires présentent dans l'équation ou la table de vérité. A chaque case correspond un état logique des *n* variables.

Ainsi pour un problème à 3 variables, l'équation des variables dans chaque case se présente de la manière suivante :

C \ AB	0 0	0 1	1 1	1 0
0	/A/B/C	/A B /C	A B /C	A /B /C
1	/A /B C	/A B C	A B C	A /B C

On remarque l'utilisation d'un code binaire réfléchi (le changement de colonne n'implique que le changement d'une seule variable à la fois)

Pour une même case, l'opérateur logique entre chaque variable est le ET.

D'une case à l'autre l'opérateur logique est le OU.

Pour notre exemple précédent, le tableau serait complété de la manière suivante :

C \ AB	0 0	0 1	1 1	1 0
0	0	1	1	0
1	0	1	1	1

C'est grâce au regroupement de cases que l'on arrive à simplifier le problème.

Règles de regroupement :

•Un regroupement ne peut être constitué que d'un nombre de cases de puissance de 2 (2, 4, 8, 16...)

•Les cases regroupées doivent être adjacentes et contenir la valeur 1 (ceci dans le cas ou l'on souhaite connaître l'équation donnant la valeur de sortie à 1). Il est à remarquer que les cases extrêmes du tableau sont adjacentes entre elles (colonne 00 adjacent avec colonne 10, etc...)

•Les regroupements en diagonal ne sont pas admis

•L'équation d'un regroupement correspond à l'équation logique liant les variables ne changeant pas d'état dans tous le regroupement.

•Le regroupement peut se faire également par les bords du tableau

	00	01	11	10
	1	0	0	1

Observons les regroupement du problème précédent:

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

•Les deux cases regroupées entre elles sont adjacentes horizontalement. Leur équation est :

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

On remarque que l'équation finale de ce regroupement ne comporte que les variables A et C qui ne changent pas d'état dans le regroupement.

•Les quatre cases regroupées sont adjacentes horizontalement et verticalement. Pour ces 4 cases la seule variable qui ne change pas est B. L'équation de ce regroupement est donc B et l'équation finale du problème est : B + A.C

III.4/ De l'équation au schéma

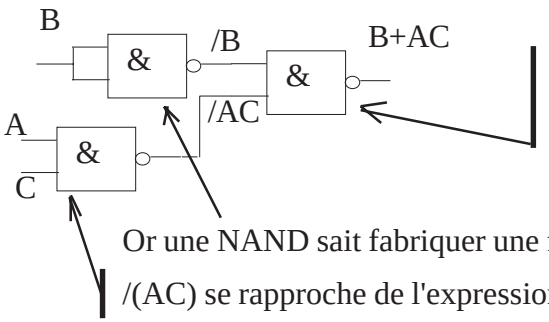
La finalité étant de réaliser une structure logique à partir d'un problème, il faut pour terminer donner le schéma de la structure logique.

Pour cela, on élabore toujours le schéma structurel en commençant par la fin, c'est-à-dire l'équation finale. Mais avant cela, il faut connaître le type de porte utilisées.

Reprenons l'exemple précédent et supposons que l'on souhaite réaliser la structure en NAND uniquement.

L'équation de sortie d'une porte NAND s'écrit $S=\overline{X.Y}$ ①

ou $S=\overline{X}+\overline{Y}$ ②



La structure à obtenir à pour équation S = B + AC elle se rapproche donc par sa forme de la relation 2. Une porte NAND pourra donc isoler B et AC. En entrée de cette porte on retrouvera alors les deux expressions complémentées.

Or une NAND sait fabriquer une fonction NON.

/(AC) se rapproche de l'expression 1.

Si l'on se trouve face à un ET non complémenté ou un OU complémenté, on utilisera en premier une fonction NON ce qui permet de se rapprocher des expressions 1 ou 2.