

Fonctions logiques combinatoires (COM)

Cours Conception Numérique

Hes·so VALAIS
WALLIS



Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientation : [Énergie et techniques environnementales \(ETE\)](#)

Cours : Conception Numérique (Cnum)

Auteur : [Christophe Bianchi](#), [François Corthay](#), [Pierre Pompili](#), [Silvan Zahno](#)

Date : 25 août 2022

Version : v2.1

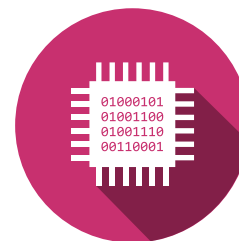


Table des matières

1	Introduction	2
2	Représentation de fonctions combinatoires	3
2.1	Représentation par table de vérité	3
2.2	Représentation par chronogramme	3
2.3	Représentation par diagramme de Venn	4
3	Fonctions logiques élémentaires	5
3.1	Tampon	5
3.2	Inverseur	5
3.3	Fonction ET	5
3.4	Fonction OU	5
3.5	Fonction OU-exclusif	6
3.6	Tableau de consultation LUT	6
3.6.1	Exemple	6
4	Algèbre booléenne	8
4.1	Notation : opérateurs logiques	8
4.2	Commutativité et associativité	8
4.3	Eléments neutres et absorbants	8
4.4	Distributivité	9
4.5	Lois de De Morgan	9
4.6	OU-exclusif et inversions	9
4.7	Définition : forme polynomiale	10
4.8	Définition : monôme	10
4.9	Résumé de l'algèbre de Boole	10
	Références	11
	Acronymes	12



1 Introduction

Les systèmes logiques combinatoires traitent des cas où les sorties dépendent uniquement des entrées, contrairement aux systèmes logiques séquentiels où les sorties dépendent aussi de la mémorisation de certains événements. Ce chapitre présente les fonctions de base des systèmes combinatoires : **inverseurs**, **ET logique**, **OU logique**, etc. Il donne aussi les techniques de base utilisées pour représenter et analyser les fonctions logiques combinatoires.



2 Représentation de fonctions combinatoires

2.1 Représentation par table de vérité

Une table de vérité se construit en dressant la liste de toutes les combinaisons possibles des entrées du système et en spécifiant pour chaque combinaison la valeur de la sortie.

Un système à n entrées indépendantes possède 2^n combinaisons possibles. La liste de ces combinaisons correspond aux nombres binaires compris entre 0 et $2^n - 1$.

Exemple : commande de ventilation

Un système de ventilation est utilisé pour régler la température d'un local. En règle générale, si la température extérieure est supérieure à la température du local, la ventilation est mise en marche pour réchauffer la salle. Toutefois, en été et de jour, la ventilation s'utilise pour refroidir la salle : si la température extérieure est inférieure à la température du local, la ventilation est mise en marche.

La commande de la ventilation dépend ici de 3 variables logiques :

- *winter*, qui vaut '1' l'hiver et '0' l'été,
- *night*, qui vaut '1' la nuit et '0' le jour,
- *ext_hot*, qui vaut '1' quand température extérieure est supérieure à la température du local et '0' sinon.

Ces 3 variables d'entrée permettent de déterminer la valeur du signal *ventilation* qui commande la mise en marche du ventilateur. La table 1 donne la valeur de ce signal de sortie en fonction des 2^3 combinaisons possibles des entrées.

<i>winter</i>	<i>night</i>	<i>ext_hot</i>	<i>ventilation</i>
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

TABLE 1 – Table de vérité de la command de ventilation

2.2 Représentation par chronogramme

Un chronogramme représente le déroulement temporel des signaux du système.

Pour complètement spécifier le fonctionnement du système, il faut veiller à faire apparaître toutes les possibilités.

Exemple : commande de ventilation

Le chronogramme de la figure 1 présente un scénario de fonctionnement du système de commande de ventilation de l'exemple précédent.

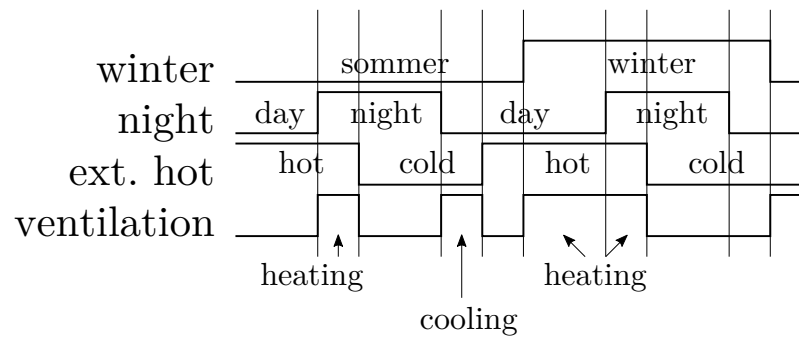
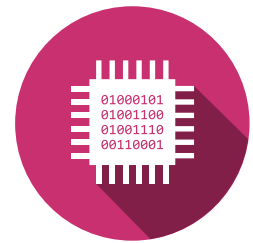


FIGURE 1 – Chronogramme de la commande de ventilation

2.3 Représentation par diagramme de Venn

Le diagramme de Venn présente les entrées sous forme d'ensembles et assigne à chaque région du diagramme une couleur ou une trame qui spécifie la valeur de la sortie.

Dans un tel diagramme, il faut s'assurer de placer les ensembles de manière à faire apparaître toutes leurs intersections possibles.

Exemple : commande de ventilation

Le diagramme de Venn de la figure 2 donne, en fonction de toutes les possibilités d'entrée, la valeur de la commande de ventilation pour l'exemple précédent.

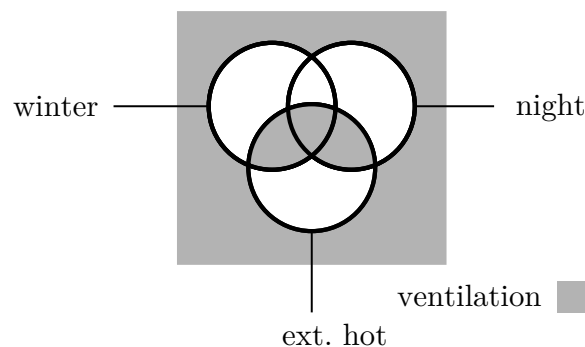
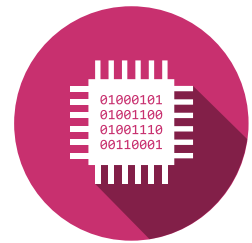


FIGURE 2 – Diagramme de Venn de la commande de ventilation



3 Fonctions logiques élémentaires

3.1 Tampon

La figure 3 donne le symbole de la fonction tampon, sa table de vérité ainsi que la représentation de la fonction dans un diagramme de Venn.

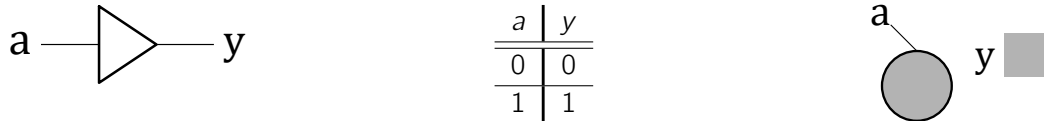


FIGURE 3 – Tampon

3.2 Inverseur

La figure 4 donne deux symboles de l'inverseur, sa table de vérité ainsi que la représentation de la fonction dans un diagramme de Venn.

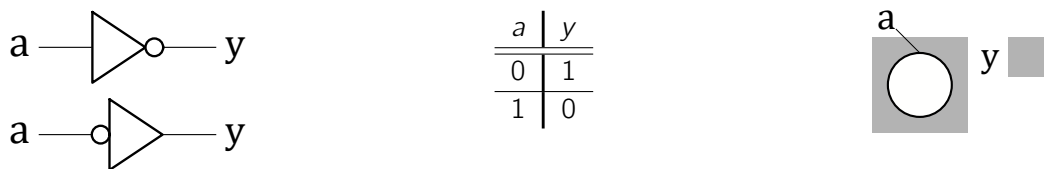


FIGURE 4 – Inverseur

3.3 Fonction ET

La figure 5 donne le symbole de la fonction ET, sa table de vérité ainsi que la représentation de la fonction dans un diagramme de Venn.

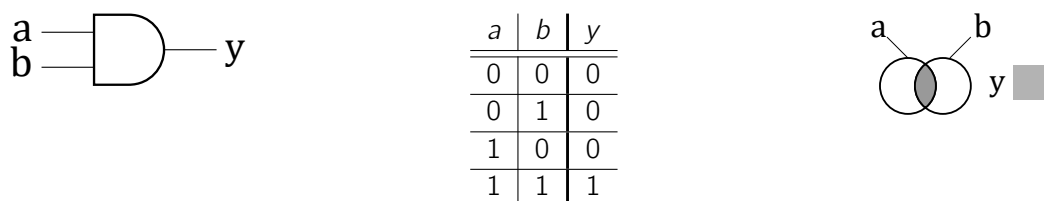
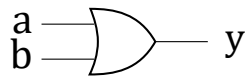
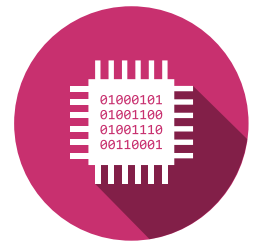


FIGURE 5 – Fonction ET

3.4 Fonction OU

La figure 6 donne le symbole de la fonction OU, sa table de vérité ainsi que la représentation de la fonction dans un diagramme de Venn.



a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

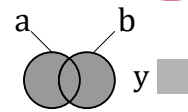
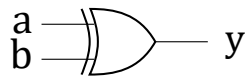


FIGURE 6 – Fonction OU

3.5 Fonction OU-exclusif

La figure 7 donne le symbole de la fonction OU-exclusif, sa table de vérité ainsi que la représentation de la fonction dans un diagramme de Venn.



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

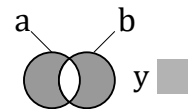


FIGURE 7 – Fonction OU-exclusif

3.6 Tableau de consultation LUT

Look-Up Table (LUT) sont constituées d'un certain nombre de portes OU, précédées de portes ET dont les entrées sont programmables. Il est ainsi possible de réaliser des fonctions à partir de tables de vérité. La figure 10 représente un LUT avec 3 entrées a , b et c et une sortie y .

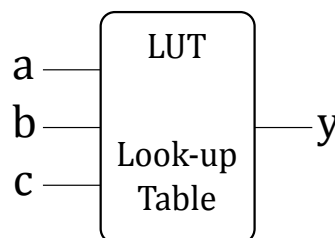


FIGURE 8 – LUT avec 3 entrées

3.6.1 Exemple

Dans la figure 9 est représentée la fonction d'un additionneur. Les entrées a , b et c_{in} sont additionnées et donnent le résultat s .

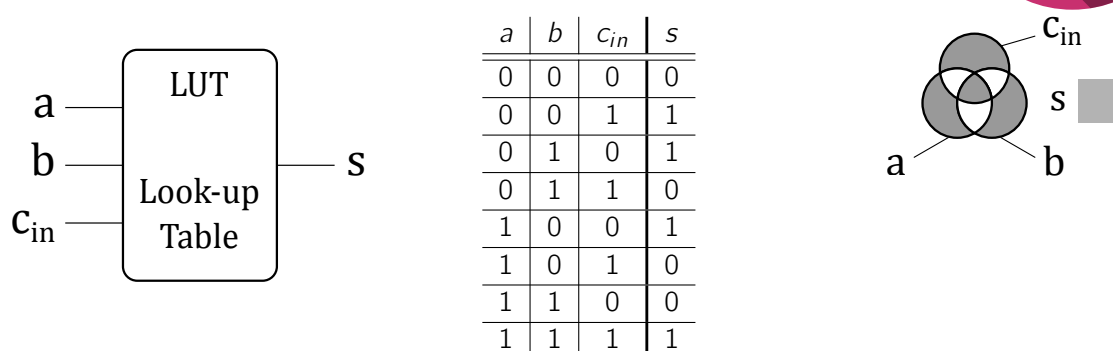
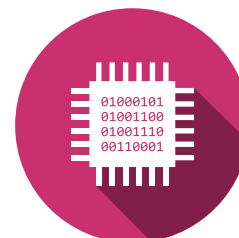


FIGURE 9 – Fonction d'additionneur mise en œuvre dans un élément de LUT

Dans l'illustration 10, on voit la structure interne des LUT qui implémente la fonction d'additionneur.

Dans une puce **Programmable Array Logic (PAL)**, il y a des LUT avec différentes entrées comme le montre la figure 11. Cela permet de réaliser des fonctions à partir de la forme polynomiale de leurs équations, pour autant que le nombre de termes du polynôme soit inférieur ou égal au nombre d'entrées des portes OU disponibles.

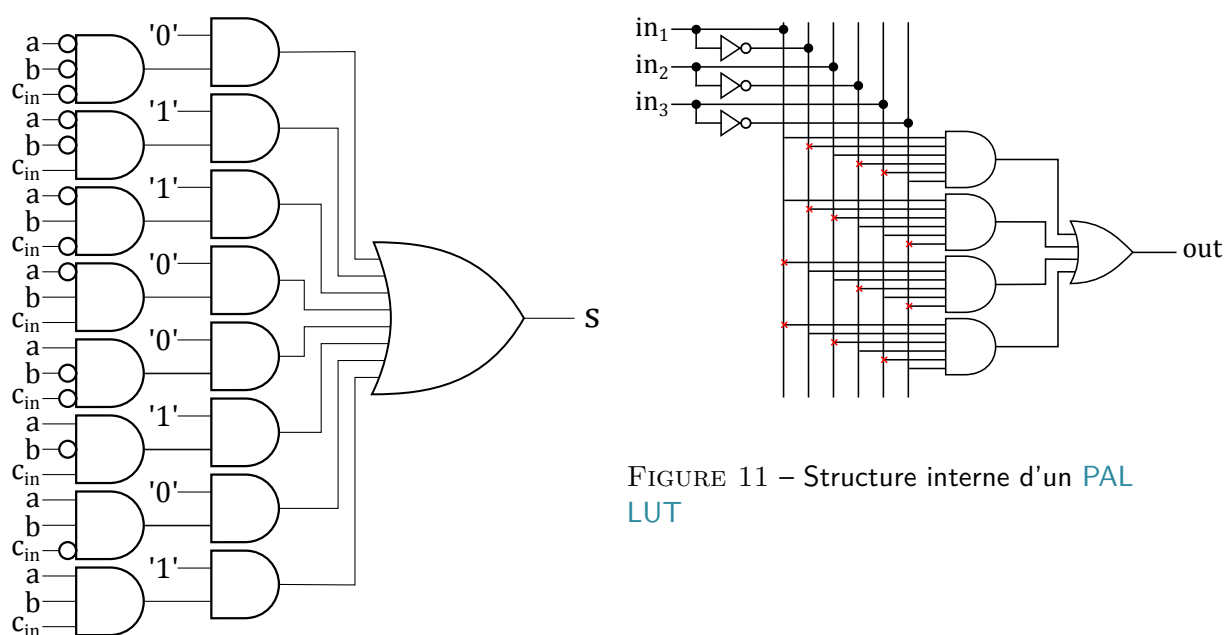
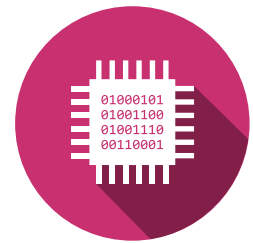


FIGURE 11 – Structure interne d'un PAL LUT

FIGURE 10 – Structure interne du LUT



4 Algèbre booléenne

L'algèbre booléenne, du nom du mathématicien Boole, traite des opérations sur des variables logiques pouvant valoir '0' ou '1'.

4.1 Notation : opérateurs logiques

La table 2 présente l'expression algébrique des fonctions logiques combinatoires de base.

Opérateur	Notation
Inverseur	$y = \bar{a}$
ET	$y = a * b$
OU	$y = a + b$
OU-exclusif	$y = a \oplus b$

TABLE 2 – Opérateurs logiques

La fonction OU-exclusif peut se définir à l'aide des fonctions inversuer, ET et OU comme le montre l'eq 1.

$$a \oplus b = a\bar{b} + \bar{a}b \quad (1)$$

4.2 Commutativité et associativité

Les opérateurs ET, OU et OU-exclusif sont commutatifs.

$$\begin{aligned} a * b &= b * a \\ a + b &= b + a \\ a \oplus b &= b \oplus a \end{aligned} \quad (2)$$

Les opérateurs ET, OU et OU-exclusif sont associatifs.

$$\begin{aligned} (a * b) * c &= a * (b * c) = a * b * c \\ (a + b) + c &= a + (b + c) = a + b + c \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c) = a \oplus b \oplus c \end{aligned} \quad (3)$$

4.3 Eléments neutres et absorbants

Le '1' est l'élément neutre de l'opération ET.

$$a * 1 = a \quad (4)$$

Le '0' est l'élément absorbant de l'opération ET.

$$a * 0 = 0 \quad (5)$$

Le '0' est l'élément neutre de l'opération OU.



$$a + 0 = a \quad (6)$$

Le '1' est l'élément absorbant de l'opération OU.

$$a + 1 = 1 \quad (7)$$

Le '0' est l'élément neutre de l'opération OU-exclusif.

$$a \oplus 0 = a \quad (8)$$



Une porte ET permet de transmettre un signal ou de le forcer à '0'. Une porte OU permet de transmettre un signal ou de le forcer à '1'. Une porte OU-exclusif permet de transmettre un signal ou de l'inverser.

4.4 Distributivité

Les opérateurs ET et OU sont distributifs l'un par rapport à l'autre.

$$\begin{aligned} a * (b + c) &= a * b + a * c \\ a + (b * c) &= (a + b) * (a + c) \end{aligned} \quad (9)$$

4.5 Lois de De Morgan

Les lois de De Morgan permettent de remplacer un opérateur ET par un opérateur OU.

$$\begin{aligned} \overline{a * b * c} &= \bar{a} + \bar{b} + \bar{c} \\ \overline{a + b + c} &= \bar{a} * \bar{b} * \bar{c} \end{aligned} \quad (10)$$

4.6 OU-exclusif et inversions

L'opérateur OU-exclusif donne une indication de parité : si le nombre d'entrées à '1' est impair, la fonction retourne un '1' ; si le nombre d'entrées à '1' est pair, la fonction retourne un '0'.

Le fait d'inverser 2 ou un nombre pair d'entrées d'un opérateur OU-exclusif n'en modifie pas la sortie. Le fait d'inverser 1 ou un nombre impair d'entrées d'un opérateur OU-exclusif inverse la sortie. En résumé, il est possible d'inverser un nombre pair de lignes d'entrée ou de sortie d'un opérateur OU-exclusif sans en modifier la fonction. Ceci est présenté à la figure 12.

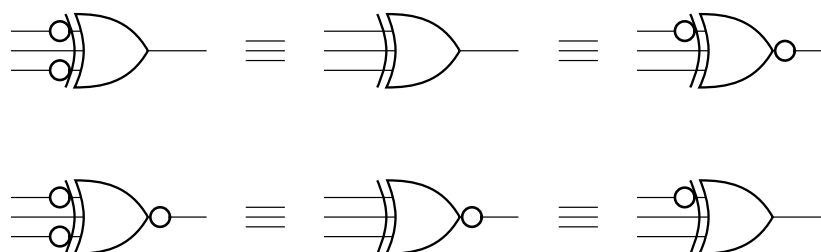


FIGURE 12 – OU-exclusif et inversions



4.7 Définition : forme polynomiale

Une forme polynomiale d'une fonction logique combinatoire exprime celle-ci par une somme de produits.

Le polynôme est constitué de ses termes reliés par une fonction OU. Chaque terme est constitué de variables reliées par une fonction ET. Les variables peuvent être présentes telles quelles ou bien inversées.

Exemple

La forme polynomiale de la fonction $y = a \oplus b \oplus c$ est donnée par $y = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$.

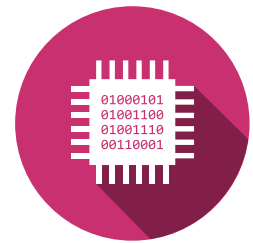
4.8 Définition : monôme

Un monôme est un terme d'un polynôme.

Exemple

Le polynôme $ab + \bar{a}c + b\bar{c}$ est formé des monômes ab , $\bar{a}c$ et $b\bar{c}$.

4.9 Résumé de l'algèbre de Boole

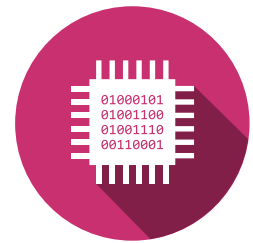


Topic	Rules
Misc	$\overline{0} = 1$ $\overline{1} = 0$ $\overline{(\overline{a})} = a$
AND	$a * 0 = 0$ $a * 1 = a$ $a * a = a$ $a * \overline{a} = 0$ $(a * b) * c = a * (b * c) = a * b * c$
OR	$a + 0 = a$ $a + 1 = 1$ $a + a = a$ $a + \overline{a} = 1$ $(a + b) + c = a + (b + c) = a + b + c$
XOR	$a \oplus b = a\overline{b} + \overline{a}b$
Commutativity	$a * b = b * a$ $a + b = b + a$
Associativity	$(a * b) * c = a * (b * c) = a * b * c$ $(a + b) + c = a + (b + c) = a + b + c$
Distributivity	$a * (b + c) = (a * b) + (a * c)$ $a + (b * c) = (a + b) * (a + c)$
De Morgan	$\overline{(a * b)} = \overline{a} + \overline{b}$ $\overline{(a + b)} = \overline{a} * \overline{b}$
Absorption	$a + (a * b) = a$ $a * (a + b) = a$ $a + (\overline{a} * b) = a + b$ $a * (\overline{a} + b) = a * b$

TABLE 3 – Résumé de l'algèbre de Boole

Références

- [1] Suhail ALMANI. *Electronic Logic Systems*. second edition. New-Jersey : Prentice-Hall, 1989.
- [2] Jean Michel BERNARD et Jean HUGON. *Pratique Des Circuits Logiques*. quatrième édition. Paris : Eyrolles, 1987.
- [3] Klaus Elektronik Bd BEUTH. *Digitaltechnik*. Vogel Buchverlag, 1990. ISBN : 3-8023-1755-6.
- [4] Michael D. CILETTI et M. Morris MANO. *Digital Design*. second edition. New-Jersey : Prentice-Hall, 2007.
- [5] David J. COMER. *Digital Logic and State Machine Design*. Saunders College Publishing, 1995.
- [6] Donald L. DIETMEYER et R. DAVID. "Logic Design of Digital Systems". In : *Journal of Dynamic Systems, Measurement, and Control* 94.2 (1^{er} juin 1972), p. 174-174. ISSN : 0022-0434. DOI : [10.1115/1.3426575](https://doi.org/10.1115/1.3426575). URL : <https://doi.org/10.1115/1.3426575> (visité le 01/06/2021).
- [7] William I. FLETCHER. *Engineering Approach to Digital Design*. New-Jersey : Prentice-Hall, 1980.
- [8] Randy H. KATZ et Gaetano BORRIELLO. *Contemporary Logic Design*. California : The Benjamin/Cummings Publishing Company Inc, 2005.



- [9] Martin V. KÜNZLI et Marcel MELI. *Vom Gatter Zu VHDL : Eine Einführung in Die Digitaltechnik*. vdf Hochschulverlag AG, 2007. ISBN : 3 7281 2472 9.
- [10] David LEWIN et Douglas PROTHEROE. *Design of Logic Systems*. second edition. Hong Kong : Springer, 2013.
- [11] Daniel MANGE. *Analyse et synthèse des systèmes logiques*. Editions Géorgi. T. Traité d'électricité, volume V. St Saphorin : PPUR presses polytechniques, 1995. 362 p. ISBN : 978-2-88074-045-0. Google Books : [5NSdD4GRl3cC](#).
- [12] Clive MAXFIELD. *Bebop to the Boolean Boogie*. Elsevier, 2009. ISBN : 978-1-85617-507-4. DOI : [10.1016/B978-1-85617-507-4.X0001-0](#). URL : <https://linkinghub.elsevier.com/retrieve/pii/B9781856175074X00010> (visité le 27/05/2021).
- [13] Ronald J. TOCCI et André LEBEL. *Circuits Numériques : Théorie et Applications*. deuxième édition. Ottawa : Editions Reynald Goulet inc. / Dunod, 1996.
- [14] John F. WAKERLY. *Digital Design : Principles And Practices*. 3rd edition. Prentice-Hall, 2008. ISBN : 0-13-082599-9.

Acronymes

LUT Look-Up Table. [1](#), [6](#), [7](#)

PAL Programmable Array Logic. [7](#)