ECOLE SUPERIEURE POLYTECHNIQUE

Technologie des ordinateurs

Ecole Supérieure Polytechnique (ESP)

Département Génie Informatique (DGI)

Diplôme Supérieur de Technicien en Informatique (DSTI)

Licence 1

Année académique 2019-2020

Dr Birahime DIOUF , dioufbira11@gmail.com

Chapitre 2 : Représentation et traitement de l'information :

Logique combinatoire

Comment sont traitées les informations dans les ordinateurs ?

OBJECTIFS DU CHAPITRE

- Ce cours a pour objectifs de permettre à l'étudiant de pouvoir :
- Comprendre le fonctionnement des composants internes des ordinateurs à travers la maîtrise
  - de l'algèbre de Boole,
  - des portes logiques et des circuits logiques,
  - de la logique combinatoire,
    - Apprendre la structure de quelques circuits combinatoires souvent utilisés (additionneur, soustracteur, décodeur, multiplexeur, afficheur 7 segments, …),
    - Apprendre comment utiliser des circuits combinatoires pour concevoir d'autres circuits plus complexes.

PLAN

- Algèbre de Boole
  - Variables booléennes
  - Opérations logiques élémentaires
- Fonctions (ou portes) logiques
  - fonctions logiques élémentaires (AND, OR et NOT)
  - fonctions logiques induites (NAND, NOR et XOR)
- Théorèmes fondamentaux
  - Théorèmes de base
  - Théorèmes de De Morgan
- Universalité des portes NAND et NOR
- Simplification des fonctions logiques
  - méthodes algébriques et
  - Tables de Karnaugh
- Quelques applications des circuits (additionneur, soustracteur, comparateur, UAL, multiplexeur, démultiplexeur, codeur, décodeur, transcodeur)

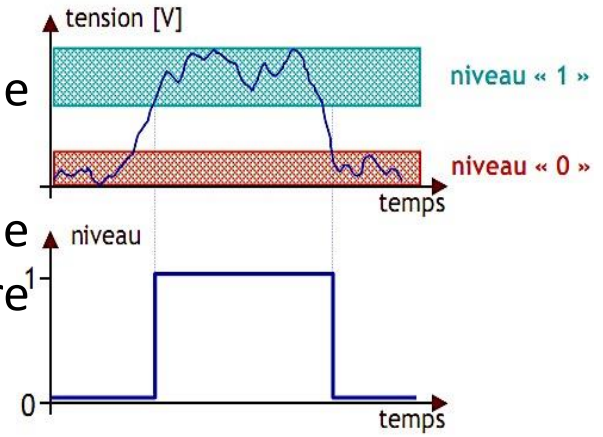
INTRODUCTION

- Un système numérique complexe est réalisé à partir
  - d'un assemblage hiérarchique d'opérateurs logiques élémentaires
  - réalisant des opérations simples sur des variables logiques.

- Les systèmes (circuits) logiques fonctionnent en mode binaire ⟹ les variables d'entrée et de sortie sont booléennes.
- Une variable logique (dite booléenne) est une grandeur binaire ;
  - peut prendre 2 valeurs (états) 0 (faux ou niveau bas) ou 1 (vrai ou niveau haut).
  - utilisée pour représenter une proposition ou l'état d'un objet.

haut	bas
vrai	faux
oui	non
1	0

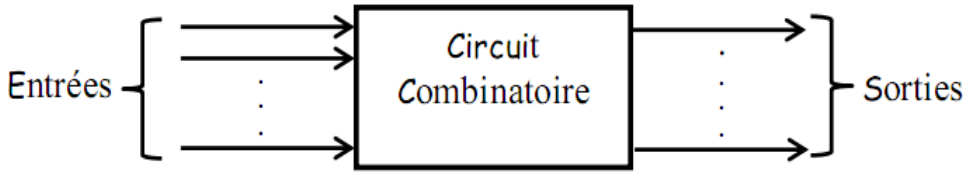
- Dans la pratique il ne s'agira pas de niveaux discrets mais plutôt de plages de tension.
- En électronique numérique, toute tension est interprétée comme une suite de symboles logiques (0/1). La manipulation de ces symboles est basée sur l'algèbre de Boole ou algèbre booléenne.



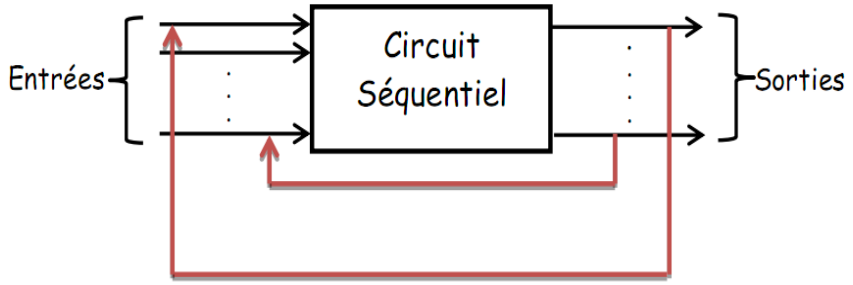
- La manipulation de ces symboles est basée sur l'algèbre de Boole .

INTRODUCTION

- Deux grands types de circuits logiques :
  - circuits logiques combinatoires :
    - Circuits numériques dont la sortie ne dépend que de l'état présent des entrées (sans mémoire des états passés); pour chaque état d'une combinaison d'entrée correspond un et unique état de sortie.



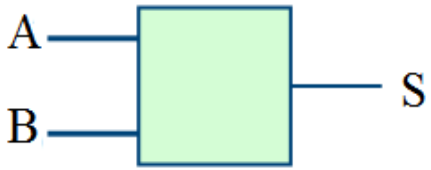
  - circuits logiques séquentielles (avec mémoire) :
    - valeurs de sorties dépendent non seulement des valeurs d'entrée, mais aussi de l'état précédent des sorties du circuit.
    - la logique séquentielle est donc une logique combinatoire avec une mémorisation des sorties.



Portes ou opérations ou fonctions logiques

Opérations logiques élémentaires

- L'algèbre de Boole ne possède que trois opérations.
  - Addition : + OU OR
  - Multiplication : • ET AND
  - Complément ou inversion : $\bar{A}$ NON NOT
- **Tables de vérité**
- Beaucoup de circuits possèdent plusieurs entrées pour une sortie.



- La table de vérité permet de décrire l'état de la sortie en fonction des combinaisons des entrées.

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

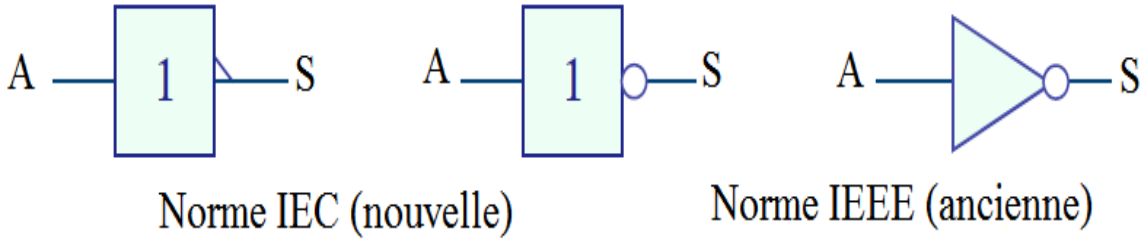
Portes ou opérations ou fonctions logiques

Opérations de base

- **Fonction NON (NOT)**
- Elle ne concerne qu'une variable à la fois ; son résultat est la complémentation ou l'inversion.
- Elle se note $S = \overline{A}$.
- Sa table de vérité est:

A	S
0	1
1	0

- Les symboles correspondants sont :



Norme IEC (nouvelle) Norme IEEE (ancienne)

- IEC (International Electrotechnical Commission),
- IEEE (Institute of Electrical and Electronics Engineers)

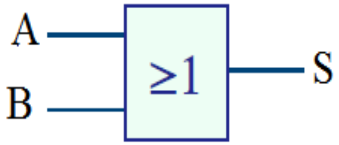
Portes ou opérations ou fonctions logiques

Opérations de base

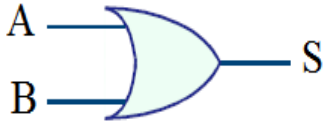
- **Fonction OU (OR)**
- Elle s'exprime par l'addition $S = A + B$. Sa table de vérité est :
- Sa table de vérité est :

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

- Le symbole correspondant est :



Norme IEC (nouvelle)



Norme IEEE (ancienne)

- **NB :** Une opération OU peut avoir N entrées. Si une entrée est à l'état 1, la sortie $S = 1$. 1 est alors prioritaire.

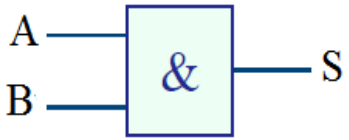
Portes ou opérations ou fonctions logiques

Opérations de base

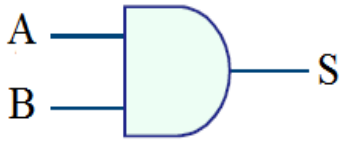
- **Fonction ET (AND)**
- Elle s'exprime par la multiplication : $S = A\,B$ ou $S = A.B.$
- Sa table de vérité est :

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

- Le symbole correspondant est :



Norme IEC (nouvelle)



Norme IEE (ancienne)

- **NB :** Une opération ET peut avoir N entrées. Si une entrée est à l'état 0, la sortie S=0. 0 est alors prioritaire.

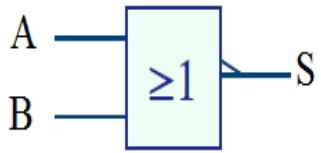
Portes ou opérations ou fonctions logiques

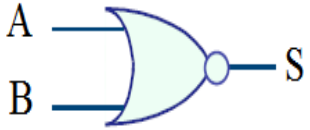
Opérations induites

- **Opération NON-OU (NOR)**
- Elle s'exprime par l'addition complémentée $S = \overline{A + B}$.
- Sa table de vérité est :

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

- Les symboles correspondants sont :



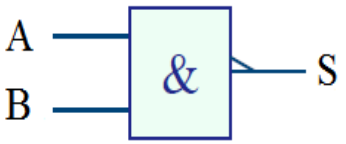


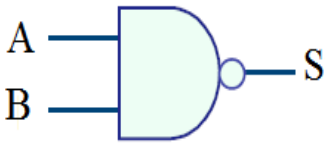
- **NB** : $\overline{A + B} \neq \overline{A} + \overline{B}$

- **Opération NON-ET (NAND)**
- Elle s'exprime par la multiplication complémentée $S = (\overline{AB})$ ou $S = (\overline{A \cdot B})$.
- Sa table de vérité est :

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

- Les symboles correspondants sont :





- **NB** : $\overline{AB} \neq \overline{A}\ \overline{B}$

Portes ou opérations ou fonctions logiques

Opérations induites

- **Opération OU-EXCLUSIF (XOR)**
- Elle s'exprime par $S = A \oplus B$.
- Sa table de vérité est :

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

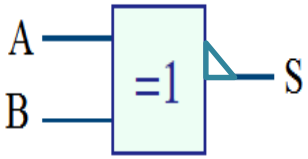
- Les symboles correspondants sont :

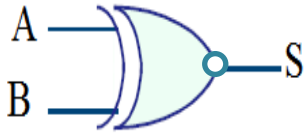
- **NB** : $A \oplus B = \overline{A}B + A\overline{B}$

- **Opération NON-OU-EXCLUSIF (XNOR)**
- Elle s'exprime par $S = \overline{A \oplus B}$.
- Sa table de vérité est :

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

- Les symboles correspondants sont :





- **NB** : $\overline{A \oplus B} = AB + \overline{A}\,\overline{B}$

$\overline{A \oplus B} = \overline{A} \oplus B = A \oplus \overline{B}$

Sommes de produits, produits de sommes et forme canonique

Notions de min-termes et max-termes :

- Un min-terme est donc représenté par un produit logique comportant tous les termes de base sans exception sous leur forme vraie ou leur forme complémentée.
- Un max-terme est donc représenté par la somme logique comportant tous les termes de base sans exception sous leur forme vraie ou leur forme complémentée.
- Exemple : 3 variables

A	B	C	**S**	Min – termes	Max – termes
0	0	0	**1**	$\overline{A}\,\overline{B}\,\overline{C}$	
0	0	1	**1**	$\overline{A}\,\overline{B}\,C$	
0	1	0	**0**		$A+\overline{B}+C$
0	1	1	**0**		$A+\overline{B}+\overline{C}$
1	0	0	**0**		$\overline{A}+B+C$
1	0	1	**1**	$A\,\overline{B}\,C$	
1	1	0	**0**		$\overline{A}+\overline{B}+C$
1	1	1	**1**	$A\,B\,C$	

Expression logique :

Somme de produits

$$S = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + A\,\overline{B}\,C + ABC$$

Produit de sommes

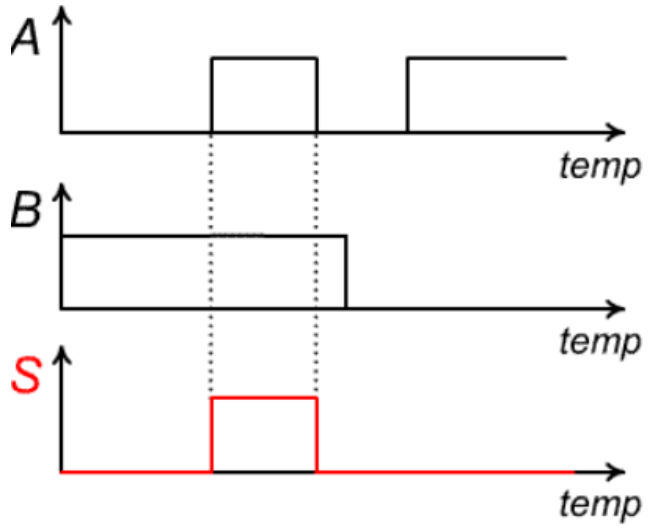
$$S = (A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$$

Chronogramme de circuits logiques

Chronogramme (diagramme des temps):

- Plusieurs modèles pour la description du fonctionnement désiré d'un système et de la conception d'un système.
- Chronogramme (diagramme des temps): C'est le graphe de l'évolution temporelle des variables et des fonctions logiques.
  - C'est un modèle graphique qui représente l'évolution au cours du temps de toutes les entrées et sorties du système.
  - Cette représentation permet de définir un certain nombre d'états du système.
  - L'état initial est choisi arbitrairement.

Exemple : Chronogramme d'une fonction ET



Conception et représentation d'un circuit numérique

- Dans beaucoup d'applications, on cherche à réaliser un circuit à partir d'une description théorique (cahier des charges).
- On établit
  1. d'abord la ***table de vérité***,
  2. puis on détermine l'***expression algébrique***
  3. et enfin on représente le ***schéma du circuit***.

- La conception d'un circuit numérique peut être délicate, car il peut y avoir de multiples contraintes :
  - obligation ou non d'utiliser des puces du même type (matériel)
  - minimiser ou non le nombre de portes (consommation)
  - optimiser ou non la rapidité de l'opération (caractéristiques)
  - permettre ou non des modifications ultérieures (souplesse)
  - ...

Conception et représentation d'un circuit numérique

- **Exercice d'application 1 :**
- Concevoir un circuit à trois entrées, dont la sortie est à l'état haut si le nombre d'entrées à l'état haut vaut 2 ou 3.
- Sa table de vérité est :

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	1
1	1	0	1
1	1	1	1

$s = \bar{A}BC$

$s = A\bar{B}C$

$s = AB\bar{C}$

$s = ABC$

- $S = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$
- Résultat final $S = AB + AC + BC$ Comment trouver ceci ?

Conception et représentation d'un circuit numérique

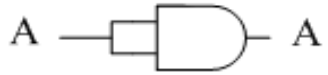
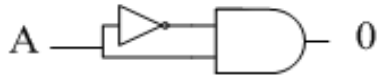
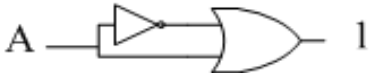
Simplification des fonctions logiques

- Nécessité :
  - Utiliser le moins de composants possibles
  - Simplifier au maximum le schéma de câblage
  - Il faut donc trouver la forme minimale de l'expression logique considérée.
- Deux méthodes :
  - Mathématique ou Algébrique (en utilisant des propriétés et des théorèmes)
  - Graphique (tableaux de Karnaugh; ...)

Simplification des expressions logiques

Simplification algébrique

- Théorèmes fondamentaux
- ***Théorèmes de base***

Multiplication		Addition	
$A \cdot 0 = 0$		$A + 0 = A$	
$A \cdot 1 = A$		$A + 1 = 1$	
$A \cdot A = A$		$A + A = A$	
$A \cdot \overline{A} = 0$		$A + \overline{A} = 1$	

- Commutativité : $A + B = B + A$ et $A \cdot B = B \cdot A$
- Associativité : $A + (B + C) = (A + B) + C$ et $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- Distributivité : $A \cdot (B + C) = A \cdot B + A \cdot C$
- Trois résultats utiles : $\overline{\overline{A}} = A$ $A + AB = A$ et $A + \overline{A}B = A + B$

Exercice d'application 2 : Démontrer ces trois dernières relations.

Simplification des expressions logiques

Simplification algébrique

- Théorèmes fondamentaux
- ***Théorèmes de De Morgan***
- Utiles pour convertir des sommes en des produits et vice-versa.
- Théorèmes pour 2 variables :

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

- Théorèmes pour N variables :

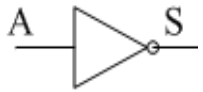
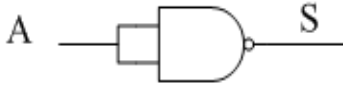
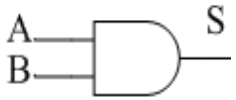
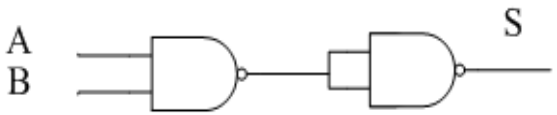
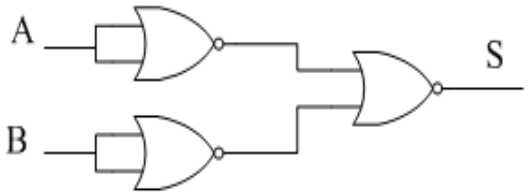
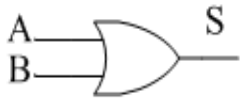
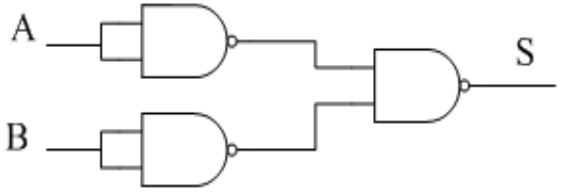
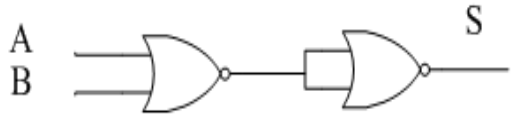
$$\overline{\sum X_i} = \prod \overline{X}_i \qquad \overline{\prod X_i} = \sum \overline{X}_i$$

- **<u>Exercice d'application 3</u> :**
- Simplifier les expressions suivantes :
  1. $F = (A + B)(\overline{A} + \overline{B})$
  2. $F = \overline{\overline{A}B + \overline{\overline{A} + B}}$
  3. $F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}B\overline{C} + A\overline{B}C$

Simplification des expressions logiques

Simplification algébrique

- Universalité des portes NAND et NOR
- Il est possible de réaliser toutes les opérations booléennes au moyen d'une seule sorte d'opérateur : opérateurs NAND ou opérateurs NOR. Elle permettent de réaliser n'importe quelle autre porte.

		Pour NAND	Pour NOR
NOT			
AND			
OR			

- **Exercice 4 :**
- Réaliser à l'aide d'opérateurs NAND puis NOR l'opérateur XOR : $S = A \oplus B = \bar{A}B + A\bar{B}$

Simplification des expressions logiques

Simplification graphique : table de Karnaugh

- Permet d'écrire une équation booléenne, de la simplifier et de déduire une implémentation des composants pour le montage correspondant.
  - Avec N=3 : A, B et C sont les entrées.

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0				
A 1				

  - Avec N=4 : A, B, C, D sont les entrées.

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00				
$\bar{A}B$ 01				
AB 11				
$A\bar{B}$ 10				

Règles pratiques

- A partir de la table, on simplifie en regroupant les 1 adjacents.
- La taille d'un groupe est un multiple de 2 c.-à-d. 2^k (2^0=1, 2^1=2, 4, 8, ...).
- Un groupe de 2^k 1 permet de réduire k variables.
- Le groupe est soit rectangulaire ou carré.
- Former les plus gros groupes possibles.
- Un 1 peut faire partie de plusieurs groupes.
- D'une colonne à l'autre, on ne complémente qu'une seule variable (Gray).
- Les bords sont périodiques (le tableau est cyclique).
- La variable qui apparait à la fois complémentée et non complémentée (c.-à-d. la variable qui change de valeur) est à éliminer.

Simplification des expressions logiques

Simplification graphique : table de Karnaugh

- **Exemples**
- Reprenons l'exemple de l'exercice d'application 1.
- La table de vérité étant la suivante :

Entrées			Sortie
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0	0	1	0
A 1	0	1	1	1

$$Y = AC + AB + BC$$

Simplification des expressions logiques

Simplification graphique : table de Karnaugh

- **Exemples**
- *Soit un circuit à 3 entrées, la sortie S est :*

$$S = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$$

- Nous obtenons les tables suivantes :

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

BC / A	00	01	11	10
0	1	0	1	1
1	1	0	0	0

$$S = \bar{B}\bar{C} + \bar{A}B$$

Simplification des expressions logiques

Simplification graphique : table de Karnaugh

- **Exemples**
- *Soit un circuit à 4 entrées, la sortie S est :*

$$S = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}D$$

- Nous obtenons la table de Karnaugh :

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

$$S = \bar{C}D + \bar{A}D + \bar{B}C\bar{D}$$

Simplification des expressions logiques

Simplification graphique : table de Karnaugh

- **Exemples**
- Si un état n'est pas spécifié, on laisse un « X » ou « ϕ » (état indéterminé) et on lui attribue la valeur qui convient le mieux.
- Prenons les 2 exemples suivants :

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

$$S = \bar{C}D + \bar{A}D$$

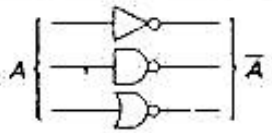
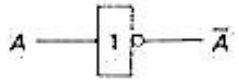
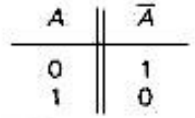
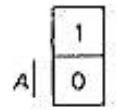
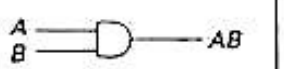
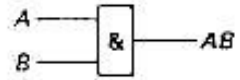
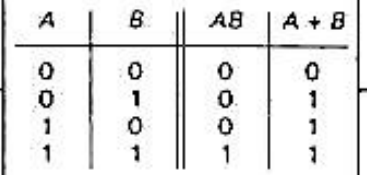
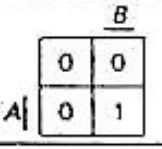
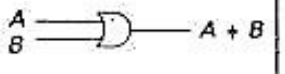
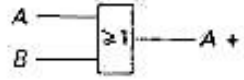
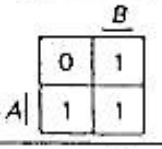
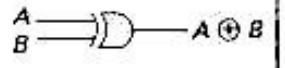
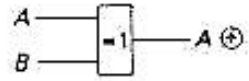
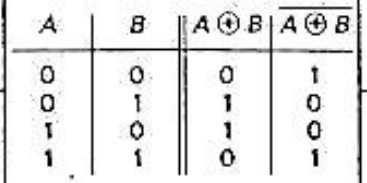
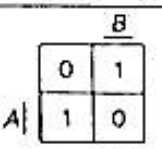
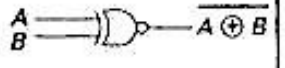
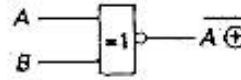
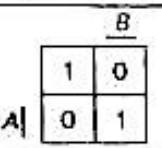
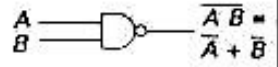
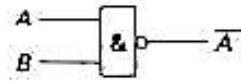
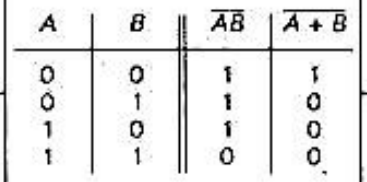
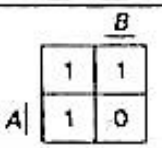
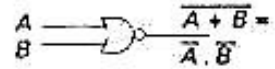
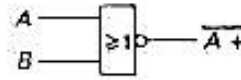
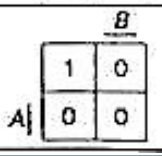
CD / AB	00	01	11	10
00	ϕ	1	1	ϕ
01	1	1	ϕ	0
11	0	1	0	ϕ
10	0	1	0	0

$$S = \bar{C}D + \bar{A}D + \bar{A}\,\bar{C}$$

Tableau récapitulatif des portes logiques

- ***Récapitulatif***

Tableau I. – Opérations logiques élémentaires.

Opération	Symbole usuel	Symbole normalisé	Table de vérité	Tableau de Karnaugh
NOT - INV				
AND - ET				
OR - OU				
XOR - OU Exclusif				
XNOR - NON OU Exclusif				
NAND - NON ET				
NOR - NON OU				

Application des circuits logiques combinatoires

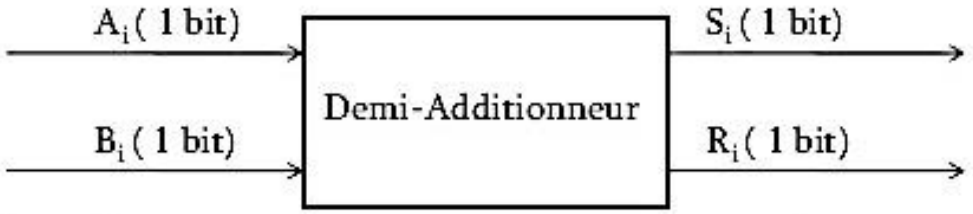
Quelques applications des circuits logiques

- Circuits logiques qui jouent un rôle important dans le hardware (circuits combinatoires standards)
- Les fonctions les plus couramment utilisées :
  - Additionneur
  - Soustracteur
  - Comparateur
  - Codeurs
  - Décodeurs
  - Multiplexeurs
  - Démultiplexeur
  - Afficheur 7 segments
  - …

Application des circuits logiques combinatoires

Demi-additionneur (half adder)

- Un additionneur est un circuit combinatoire qui permet de réaliser la somme arithmétique de 2 nombres A et B.
- **Demi-additionneur** : somme de 2 bits A_i et B_i en entrée, avec en sortie la somme S_i et la retenue R_i.



Rôle : Additionner A_i et B_i ($S_i = A_i + B_i$) en conservant la retenue R_i

Table de vérité

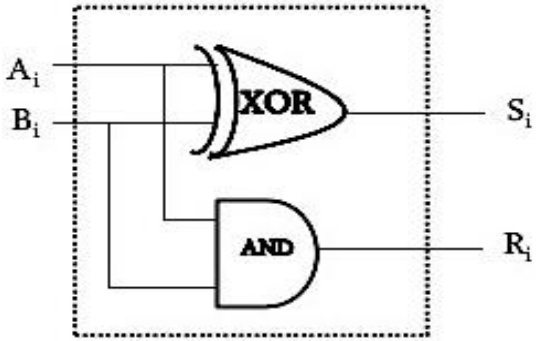
A_i	B_i		S_i	R_i
0	0		0	0
0	1		1	0
1	0		1	0
1	1		0	1

Equations

$$S_i = A_i \oplus B_i$$

$$R_i = A_i B_i$$

Schéma du circuit

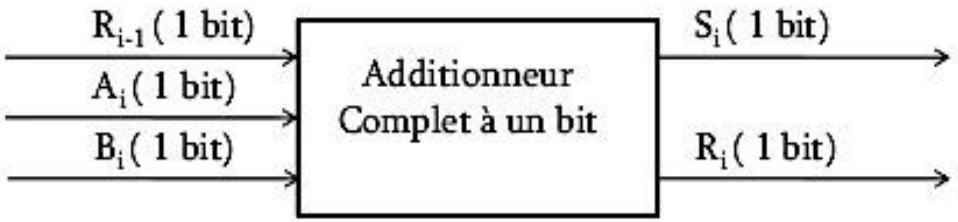


- **Exercice**
- Faire le circuit du demi-soustracteur

Application des circuits logiques combinatoires

Additionneur complet (full adder)

- L'additionneur complet un bit réalise la somme de 2 bits A_i et B_i en tenant compte de la retenue d'entrée R_{i-1}.

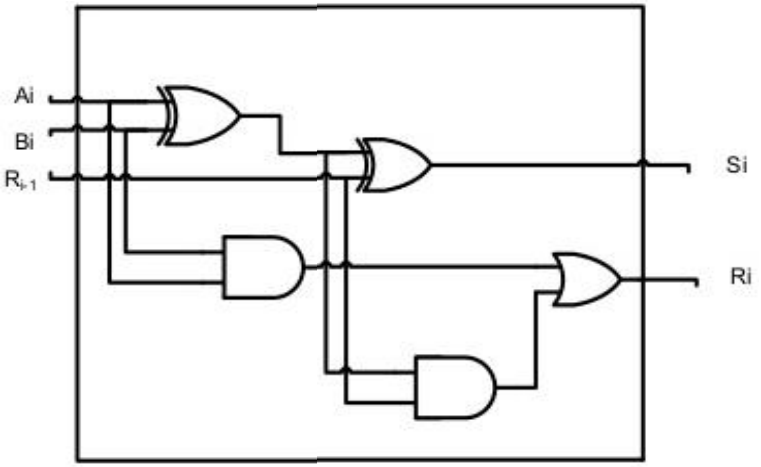


Rôle : Additionner A_i et B_i en prenant en compte la retenue d'entrée R_{i-1} et en conservant la retenue de sortie R_i

$$\begin{array}{rr} & R_{i-1} \\ & A_i \\ + & B_i \\ \hline & R_i \;\; S_i \end{array}$$

Table de vérité

Entrée			Sortie	
R_{i-1}	A_i	B_i	R_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Equations

$$S_i = \overline{A_i}.\overline{B_i}.R_{i-1} + \overline{A_i}.B_i.\overline{R_{i-1}} + A_i.\overline{B_i}.\overline{R_{i-1}} + A_i.B_i.R_{i-1}$$
$$R_i = \overline{A_i}B_iR_{i-1} + A_i\overline{B_i}R_{i-1} + A_iB_i\overline{R_{i-1}} + A_iB_iR_{i-1}$$

- **Exercice :**
- Faire le circuit de l'additionneur complet à 1 bit en utilisant 2 demi-additionneurs.

Application des circuits logiques combinatoires

Additionneur complet (full adder)

- Solution :

$$R_i = A_i.B_i + R_{i-1}.(B_i \oplus A_i)$$
$$S_i = A_i \oplus B_i \oplus R_{i-1}$$

Si on pose $X = A_i \oplus B_i$ et $Y = A_i B_i$

On obtient :

$$R_i = Y + R_{i-1}.X$$
$$S_i = X \oplus R_{i-1}$$

et si on pose $Z = X \oplus R_{i-1}$ et $T = R_{i-1}.X$

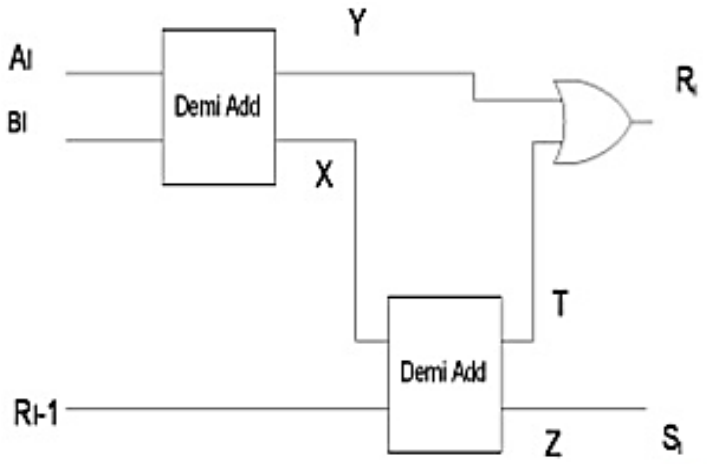
On obtient :

$$R_i = Y + T$$
$$S_i = Z$$

X et Y sont les sorties du premier demi-additionneur ayant comme entrées A_i et B_i.

Z et T sont les sorties du deuxième demi-additionneur ayant comme entrées X et R_{i-1}.

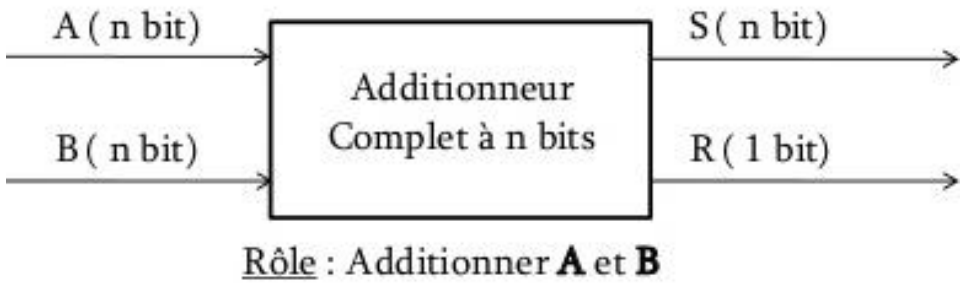
$$X = A_i \oplus B_i$$
$$Y = A_i B_i$$
$$Z = X \oplus R_{i-1}$$
$$T = R_{i-1}.X$$
$$R_i = Y + T$$
$$S_i = Z$$



Application des circuits logiques combinatoires

Additionneur complet à n bits par propagation de retenue

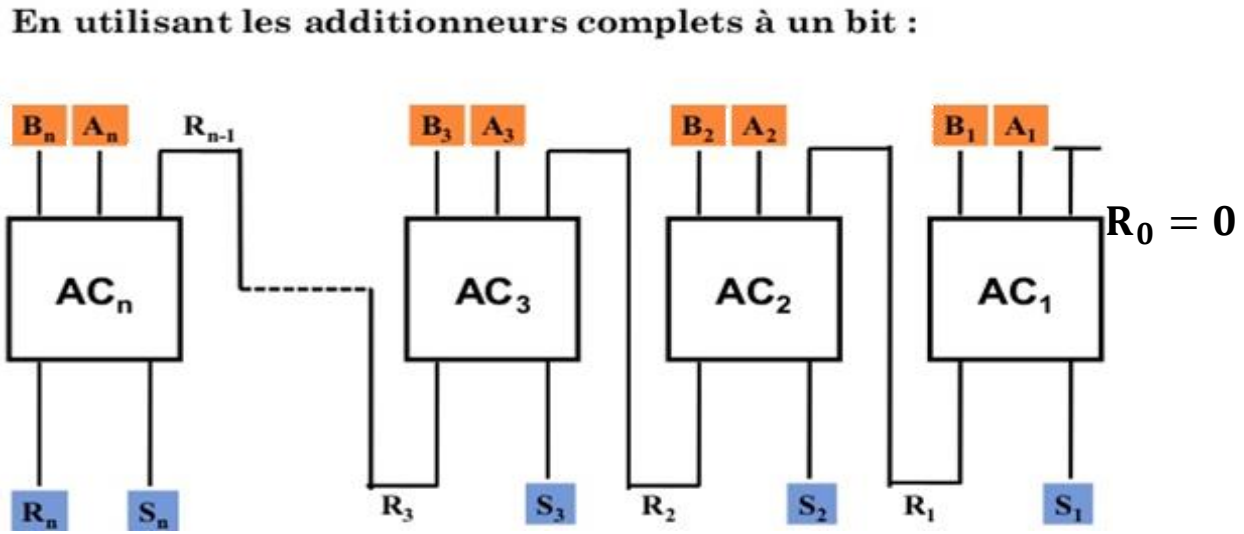
- Circuit combinatoire qui réalise la somme arithmétique de deux nombres de n bits :



Rôle : Additionner **A** et **B**

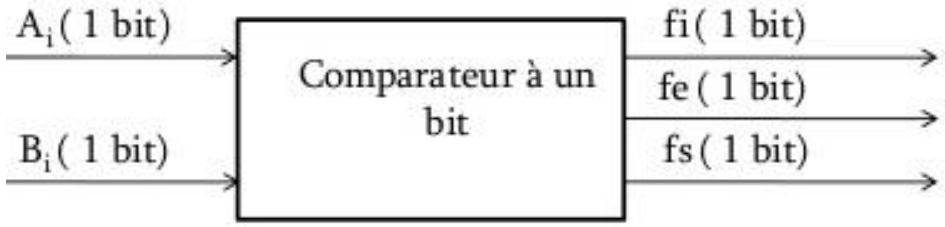
R_n	R_{n-1}	.....	R_3	R_2	R_1	$R_0=0$	
	A_n	.....	A_4	A_3	A_2	A_1	
+	B_n	.....	B_4	B_3	B_2	B_1	
R_n	S_n	.....	S_4	S_3	S_2	S_1	

- En utilisant les additionneurs complets à 1 bit :



Comparateur à 1 bit

- C'est un circuit combinatoire qui permet de comparer deux nombres binaires A et B.



Rôle : Comparer entre deux bits (A et B):
fe : égalité (A=B)
fi : inférieur (A < B)
fs : supérieur (A > B)

Table de vérité

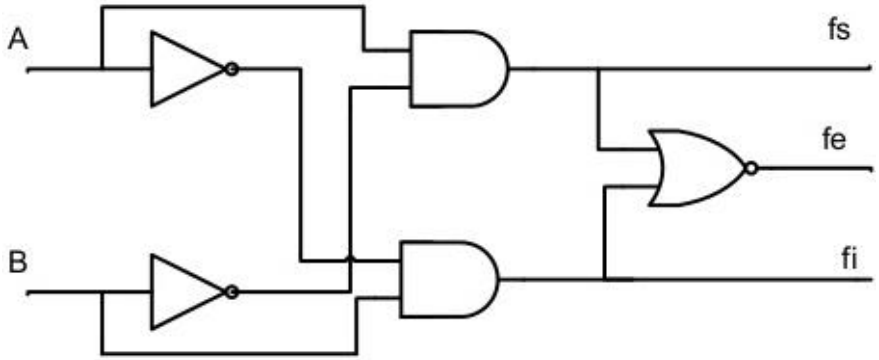
A	B		fs	fe	fi
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

Equations

$$fs = A.\overline{B}$$

$$fi = \overline{A}B$$

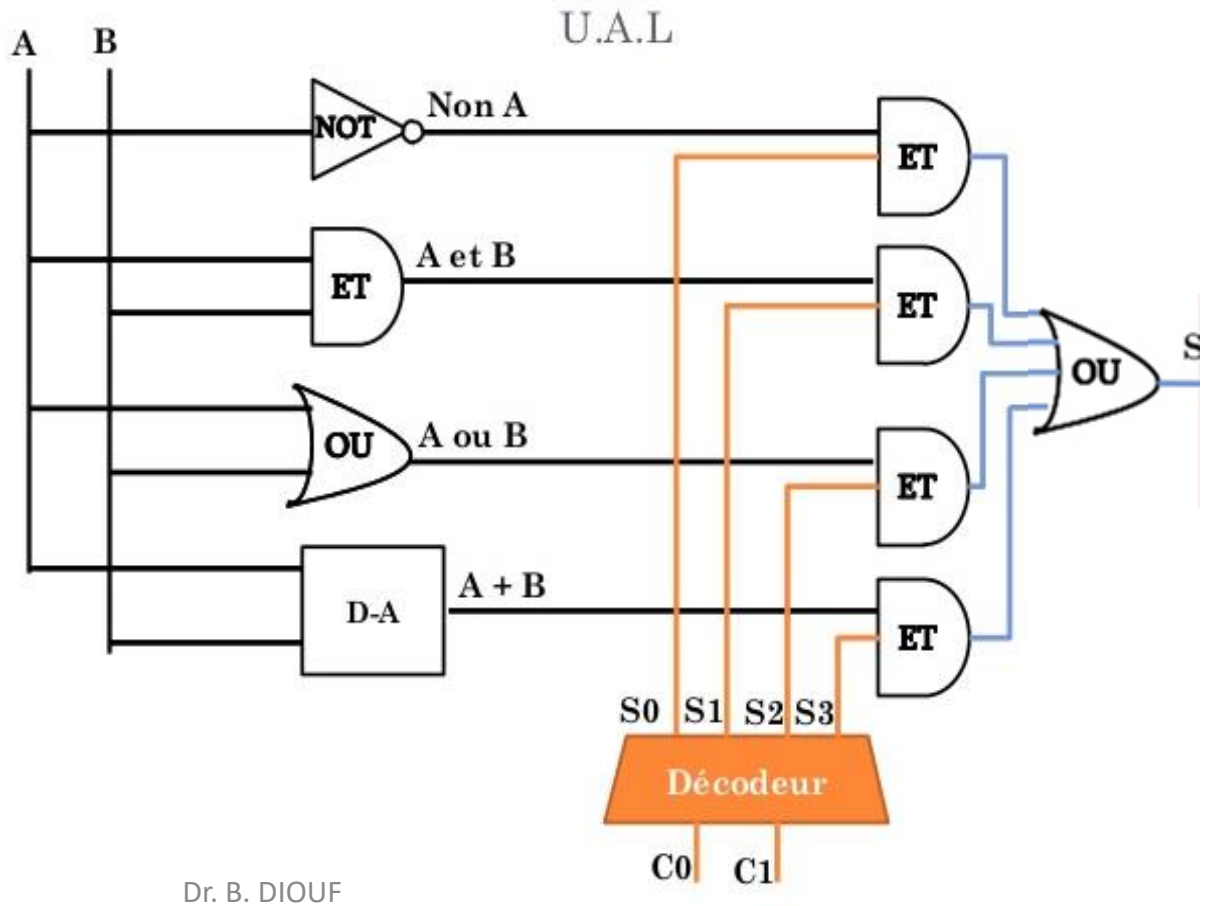
$$fe = \overline{A}\,\overline{B} + AB = \overline{A \oplus B} = \overline{fs + fi}$$



Circuits de transcodage

Applications des décodeurs : UAL (Unité Arithmétique et Logique)

- Un décodeur est un dispositif essentiel à l'entrée de l'UAL du processeur.
- *Exemple :* conversion simplifiée d'UAL à 1 bit :
- Cette UAL possède 2 entrées (A et B) à un bit sur lesquelles 4 opérations sont faites :
  - NOT A
  - A AND B
  - A OR B
  - A + B (addition arithmétique).



Transcodeur BCD/XS3 :

Exercice : Réaliser un transcodage du code BCD vers le code à excès de 3 (XS3(N) = BCD(N) + 3). Les nombres d'entrée et de sortie sont exprimés sur 4 bits, et ce transcodeur pourra convertir tous les chiffres de 0 à 9.

Chiffre converti	Entrées (BCD)				Sorties [XS 3]			
	E_3	E_2	E_1	E_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
-	1	0	1	0	x	x	x	x
-	1	0	1	1	x	x	x	x
-	1	1	0	0	x	x	x	x
-	1	1	0	1	x	x	x	x
-	1	1	1	0	x	x	x	x
-	1	1	1	1	x	x	x	x

$E_3 E_2$ \ $E_1 E_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$$S_3 = E_3 + E_2 E_0 + E_2 E_1$$

$E_3 E_2$ \ $E_1 E_0$	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$$S_2 = E_2 \overline{E_1}\,\overline{E_0} + \overline{E_2} E_0 + \overline{E_2} E_1$$

$E_3 E_2$ \ $E_1 E_0$	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

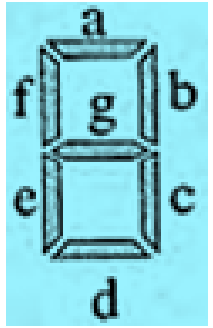
$$S_1 = \overline{E_1}\,\overline{E_0} + E_1 E_0 = \overline{E_1 \oplus E_0}$$

$E_3 E_2$ \ $E_1 E_0$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

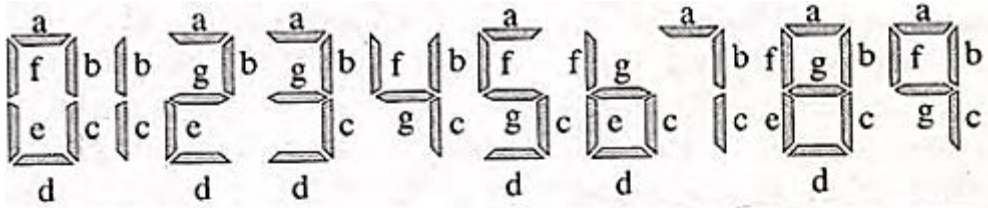
$$S_0 = \overline{E_0}$$

Circuits de transcodage

Transcodeur HEXA/7 SEGMENTS :

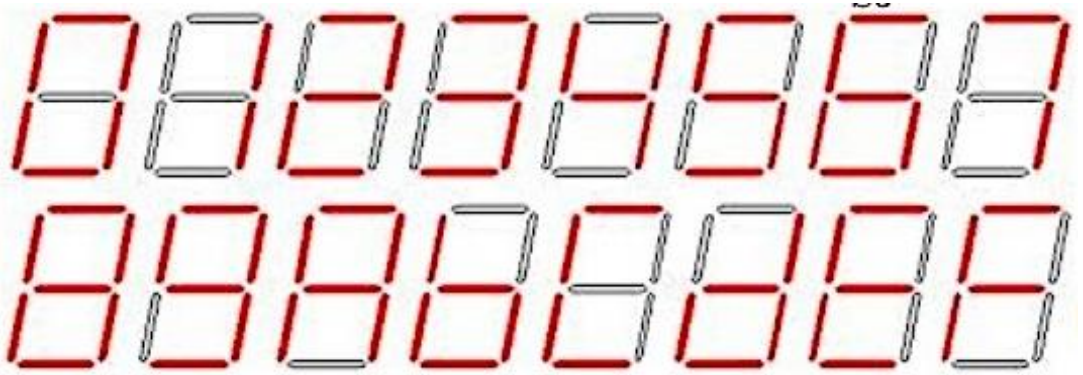


- Les 16 chiffres 0, …, 9 et A, …, F sont affichés au moyen d'un dispositif appelé afficheur à segments. Cet afficheur est un ensemble de diodes électroluminescentes (DEL) :



Chiffre décimal	Entrées D C B A	Sorties a b c d e f g	Afficheur
0	0 0 0 0	1 1 1 1 1 1 0	0
1	0 0 0 1	0 1 1 0 0 0 0	1
2	0 0 1 0	1 1 0 1 1 0 1	2
3	0 0 1 1	1 1 1 1 0 0 1	3
4	0 1 0 0	0 1 1 0 0 1 1	4
5	0 1 0 1	1 0 1 1 0 1 1	5
6	0 1 1 0	0 0 1 1 1 1 1	6
7	0 1 1 1	1 1 1 0 0 0 0	7
8	1 0 0 0	1 1 1 1 1 1 1	8
9	1 0 0 1	1 1 1 0 0 1 1	9

Affichage hexadécimal



- **Exercice :** A l'aide du tableau de Karnaugh déterminer les expressions simplifiées et le schéma logique des 7 sorties