



# Logique combinatoire



Renaud Costadoat  
Lycée Dorian



## Introduction

### Savoir

Vous êtes capables :

- de modéliser la chaîne d'information d'un système.

### Problématique

Vous devez être capables :

- de représenter l'information dans une partie commande,
- de concevoir des systèmes de traitement de l'information à l'aide de portes logiques.

## Les codes binaires

- Symboles: 0 et 1 appelés bits (binary digit), base: 2,
- La succession de ces nombres est 0, 1, 10, 11, 100, 101, 110, 111,
- Sous forme polynomiale, un nombre binaire quelconque est exprimé par:

$$N = \sum_0^n \alpha_i . 2^i \text{ avec } \alpha_i = 0 \text{ ou } 1$$

- ex:  $10110 \rightarrow 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22$  décimal.
- Définitions:
  - ▶ Un nombre binaire de n bits permet d'obtenir  $2^n$  nombres différents dont le plus grand a pour valeur décimale  $2^n - 1$ ,
  - ▶ On appelle 'octet' (byte en anglais) un nombre de 8 bits (domaine 0..255),
  - ▶ On appelle 'mot' (word en anglais) un nombre de 16 bits (domaine 0..65535), les bits 0..7 constituant l'octet de 'poids faible', les bits 8..15 constituant l'octet de 'poids fort'.

## Les codes binaires réfléchis

- L'utilisation du code binaire vu précédemment (appelé aussi code binaire naturel) dans le traitement numérique d'un signal peut poser de problèmes,
- En effet, supposons un capteur enregistrant les valeurs successives dans un comptage 0000, 0001, 0010, 0011, 0100, 0101,... On voit que le passage de 1 à 2 nécessite la modification des bits 0 et 1, ce qui peut introduire des aléas (effets transitoires néfastes). On risque d'obtenir: 0001 0000 0010 ou 0001 0011 0010
- Pour éviter ces erreurs, il suffit de coder chaque nombre de sorte que 2 nombres successifs ne diffèrent que d'un élément binaire : code à distance unité. (on appelle distance entre 2 mots-code le nombre d'éléments binaires qui diffèrent ),
- Le code Gray est le plus utilisé.

## Les codes binaires réfléchis

- Avec ce code, le passage d'un nombre au suivant ne nécessite que la modification d'un seul bit,
- La relation qui lie un nombre binaire pur avec le nombre binaire codé Gray s'écrit: ( $\oplus$  = OU exclusif),

$$N_g = \frac{N \oplus 2N}{2}$$

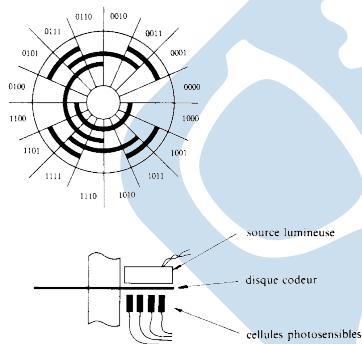
- ex:  $N = 54$  décimal
  - ▶ 110110 binaire pur  $\rightarrow 2N = 1101100$  (obtenu en décalant tous les bits d'une position à gauche),
  - ▶ Le OU exclusif donne 0 si les 2 bits sont identiques,

0	1	1	0	1	1	0
1	1	0	1	1	0	0
<hr/>						
1	0	1	1	0	1	0

- ▶ La division par 2 revient à décaler tous les bits d'une position vers la droite (le bit 0 initial est perdu), cela donne 101101.

## Les codes binaires réfléchis

- Ce code est utilisé par exemple dans un asservissement où la position angulaire d'un axe peut être codée par un dispositif optique composé d'un disque sur lequel on a gravé un motif,
- Des capteurs peuvent alors 'lire' la combinaison désirée.

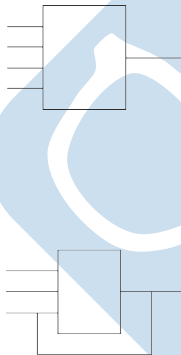


## Définition pour les opérations

- Les **opérations logiques** sont réalisées en associant des **tensions** à des variables logiques,
- Les **états et les valeurs logiques** sont représentés par les nombres 0 et 1,
- La **valeur de la variable** est une **tension électrique** appliquée entre la borne considérée et la masse du montage,
- Une **fonction logique** est représentée par des groupes de variables reliées par des **opérateurs logiques**, elle ne peut prendre que les valeurs 0 et 1,
- La **table de vérité** est une table qui permet de connaître la valeur de S en fonction des diverses **combinaisons** possibles des variables d'entrée  $E_i$ ,
- Le **chronogramme** est le graphe de l'évolution en fonction du temps des variables et des fonctions logiques.

## Combinatoire ou séquentiel ?

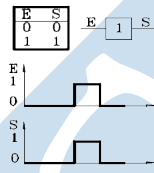
- Une fonction est dite **combinatoire** si ses sorties **ne dépendent que** des combinaisons d'entrée et pas de l'histoire de celles-ci,
- A une combinaison de variables d'entrée correspond une seule combinaison des sorties,
- Aucune mémoire des états précédents n'est conservée.
- Une fonction est dite séquentielle si ses sorties **dépendent** des combinaisons d'entrée et de l'histoire de celles-ci,
- A une combinaison de variables d'entrée correspond plusieurs combinaisons des sorties,
- Tout ou partie des combinaisons d'entrée et de sortie qui peuvent influencer les sorties de nouvelles combinaisons est conservé.



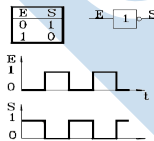


## Fonction à une variable logique

- **OUI:**  $S = E$ ,
- S est **identique** à E.



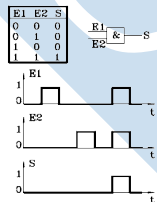
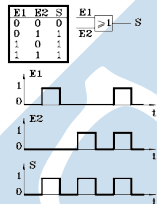
- **NON:**  $S = \bar{E}$ ,
- S est le **complément** de E.



## Fonction à deux variables logiques

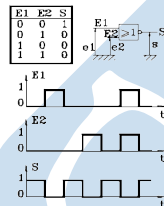
- **OU**:  $S = E_1 + E_2$ ,
- S est vrai si  $E_1$  **ou**  $E_2$  est vrai.

- **ET**:  $S = E_1 . E_2$ ,
- S est vrai si  $E_1$  **et**  $E_2$  sont vrai.

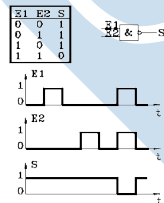


## Fonction à deux variables logiques

- **NON OU:**  $S = \overline{E_1 + E_2}$ ,
- S est le **complément** de  $(E_1 + E_2)$ .

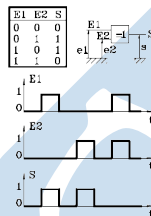


- **NON ET:**  $S = \overline{E_1 . E_2}$ ,
- S est le **complément** de  $(E_1 . E_2)$ .

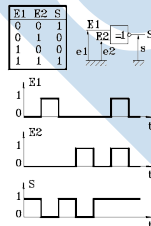


## Fonction à deux variables logiques

- **OU exclusif:**  $S = E_1 \oplus E_2$ ,
- $S = E_1 \cdot \overline{E_2} + \overline{E_1} \cdot E_2$ ,
- S ne vaut 1 que si les 2 variables d'entrée ont des valeurs différentes: **anticoïncidence**.



- **NON OU exclusif:**  $S = \overline{E_1 \oplus E_2}$ ,
- $S = \overline{E_1} \cdot \overline{E_2} + E_1 \cdot E_2$ ,
- S ne vaut 1 que si les 2 variables d'entrée ont des valeurs identiques: **coïncidence**.



## Théorèmes fondamentaux

- identités remarquables:

- $1.E = E, E + 1 = 1, 0.E = 0, E + 0 = E,$

- commutativité:

- $E_1.E_2 = E_2.E_1, E_1 + E_2 = E_2 + E_1,$

- associativité:

- $E_1.(E_2.E_3) = (E_1.E_2).E_3, E_1 + (E_2 + E_3) = (E_1 + E_2) + E_3,$

- distributivité:

- $E_1.(E_2 + E_3) = (E_1.E_2) + (E_1.E_3), E_1 + (E_2.E_3) = (E_1 + E_2).(E_1 + E_3),$

- idempotence:

- $E + E = E, E.E = E$

- complémentarité

- $E + \overline{E} = 1, E.\overline{E} = 0$

- absorption

- $E + E.A = E.$

## Théorèmes fondamentaux

### Principe de dualité:

Toute expression logique demeure vraie si l'on remplace '+' par '.' et 0 par 1 et réciproquement (facilement vérifiable pour les expressions précédentes).

### Théorèmes de De Morgan

- Théorème 1:

Le produit logique complémenté de 2 variables booléennes est égal à la somme logique des compléments de ces variables:

$$\overline{E_1 \cdot E_2} = \overline{E_1} + \overline{E_2},$$

- Théorème 2:

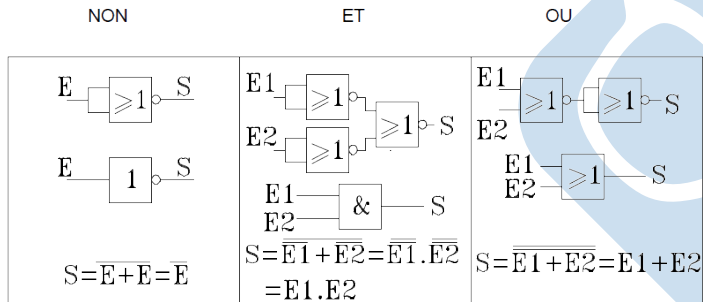
La somme logique complémentée de 2 variables booléennes est égale au produit logique des compléments de ces variables:

$$\overline{E_1 + E_2} = \overline{E_1} \cdot \overline{E_2},$$

- Remarque: Ces relations sont généralisables à n variables booléennes.

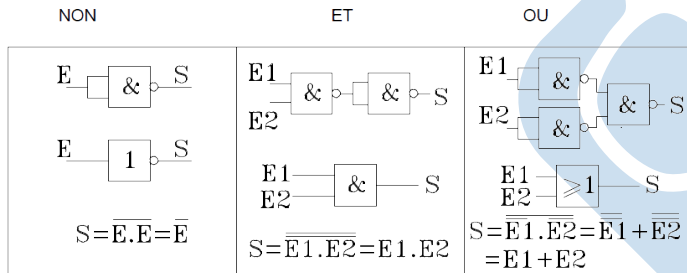
## Représentation des opérateurs logiques

### Emploi d'opérateurs NOR



## Représentation des opérateurs logiques

### Emploi d'opérateurs NAND





## Représentation des opérateurs logiques

### Exemple

- Soit la forme canonique  $S = \overline{E_1} \cdot E_2 + E_1 \cdot \overline{E_2}$ ,

$E_1$	$E_2$	$S$
0	0	0
0	1	1
1	0	1
1	1	0

**Question:** Établir les montages avec des opérateurs NAND et NOR.

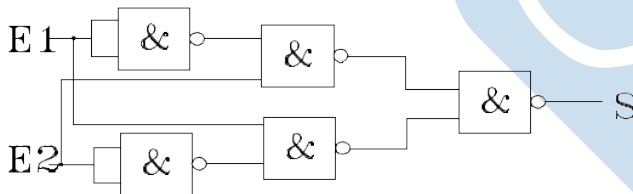
## Représentation des opérateurs logiques

### 1<sup>ère</sup> solution avec opérateurs NAND

- D'après le théorème de De Morgan:

$$\overline{S} = \overline{E_1 \cdot \overline{E_2} + \overline{E_1} \cdot E_2} = \overline{E_1 \cdot \overline{E_2}} \cdot \overline{\overline{E_1} \cdot E_2}$$
$$\text{Donc: } S = \overline{E_1 \cdot \overline{E_2}} \cdot \overline{\overline{E_1} \cdot E_2}$$

- Ce qui donne la solution suivante avec 5 opérateurs NAND:



## Représentation des opérateurs logiques

### 2<sup>ème</sup> solution avec opérateurs NOR

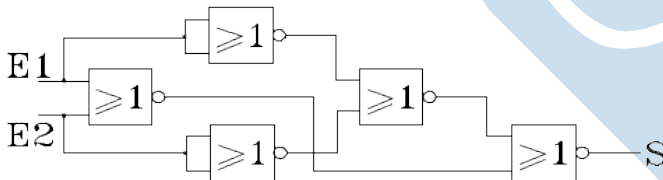
- D'après le théorème de De Morgan:

$$S = E_1 \cdot \overline{E_2} + \overline{E_1} \cdot E_2 = (E_1 + E_2) \cdot (\overline{E_1} + \overline{E_2})$$

$$\text{D'après le théorème de De Morgan : } \overline{S} = \overline{(E_1 + E_2) \cdot (\overline{E_1} + \overline{E_2})} = \overline{(E_1 + E_2)} + \overline{(\overline{E_1} + \overline{E_2})}$$

$$\text{Donc : } S = \overline{\overline{(E_1 + E_2)} + \overline{(\overline{E_1} + \overline{E_2})}}$$

- Ce qui donne la solution suivante avec 5 opérateurs NOR:



## Méthode de Karnaugh

- La représentation d'une forme canonique sous la forme d'une table de vérité devient **lourde** dès que le nombre de variables d'entrée est important. Par exemple, 3 variables nécessitent 8 lignes dans la table,
- La méthode de **Karnaugh** permet de simplifier une expression booléenne et de déduire le montage adéquat,
- Elle consiste à mettre en évidence le regroupement de termes tel que  $A + A.B = A$ ,
- Le codage des états des lignes et des colonnes est binaire réfléchi pour qu'une et une seule variable change d'état d'une case à une case adjacente.

## Méthode de Karnaugh

- Soit la forme canonique  $S = \overline{E_1} \cdot \overline{E_3} + E_2 \cdot \overline{E_4}$ .

$$S = \overline{E_1} \cdot \overline{E_3} \cdot \overline{E_2} \cdot \overline{E_4} + (\overline{E_1} + \overline{E_3}) \cdot (\overline{E_2} + \overline{E_4}) = \overline{E_1} \cdot \overline{E_2} + \overline{E_1} \cdot \overline{E_4} + \overline{E_3} \cdot \overline{E_2} + \overline{E_3} \cdot \overline{E_4}$$

- D'un point de vue théorique, on a avec 4 variables:

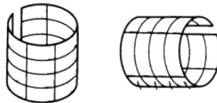
$E_1.E_2/E_3.E_4$	0 0	0 1	1 1	1 0
0 0	$\overline{E_1} \cdot \overline{E_2} \cdot \overline{E_3} \cdot \overline{E_4}$	$\overline{E_1} \cdot \overline{E_2} \cdot \overline{E_3} \cdot E_4$	$\overline{E_1} \cdot \overline{E_2} \cdot E_3 \cdot E_4$	$\overline{E_1} \cdot \overline{E_2} \cdot E_3 \cdot \overline{E_4}$
0 1	$\overline{E_1} \cdot \overline{E_2} \cdot \overline{E_3} \cdot \overline{E_4}$	$\overline{E_1} \cdot \overline{E_2} \cdot \overline{E_3} \cdot E_4$	$\overline{E_1} \cdot \overline{E_2} \cdot E_3 \cdot E_4$	$\overline{E_1} \cdot \overline{E_2} \cdot E_3 \cdot \overline{E_4}$
1 1	$E_1 \cdot \overline{E_2} \cdot \overline{E_3} \cdot \overline{E_4}$	$E_1 \cdot \overline{E_2} \cdot \overline{E_3} \cdot E_4$	$E_1 \cdot \overline{E_2} \cdot E_3 \cdot E_4$	$E_1 \cdot \overline{E_2} \cdot E_3 \cdot \overline{E_4}$
1 0	$E_1 \cdot \overline{E_2} \cdot \overline{E_3} \cdot \overline{E_4}$	$E_1 \cdot \overline{E_2} \cdot \overline{E_3} \cdot E_4$	$E_1 \cdot \overline{E_2} \cdot E_3 \cdot E_4$	$E_1 \cdot \overline{E_2} \cdot E_3 \cdot \overline{E_4}$

- D'un point de vue théorique, on a avec 4 variables:

$E_1.E_2/E_3.E_4$	0 0	0 1	1 1	1 0
0 0	1	1	1	1
0 1	1	0	0	1
1 1	1	0	0	0
1 0	1	1	0	0

## Détermination de fonction logique

- La méthode de recherche de l'**expression minimale** d'une fonction logique  $S$  à partir d'un tableau de Karnaugh consiste à rechercher des cases **adjacentes** comportant des 1 de sorte qu'un regroupement puisse être opéré dans le but de simplifier  $S$ ,
- Les cases extrêmes peuvent être considérées comme adjacentes puisqu'une seule variable change d'état, donc le tableau peut être considéré comme un cylindre vertical ou horizontal,



- *Remarque:* Lorsque certains états de la fonction  $S$  ne sont **pas définis**, l'état de la sortie dans le tableau de Karnaugh est symbolisé par une croix (état indifférent). Les croix peuvent être remplacées par des 0 ou des 1 pour faciliter au mieux les regroupements.

## Détermination de fonction logique

### Recherche d'octets

Huit 1 voisins peuvent être regroupés car 3 variables changent d'état, celles-ci disparaissent dans le terme qui résulte.

	1	1	
	1	1	
	1	1	
	1	1	

1			1
1			1
1			1
1			1

Exemple:  $S = \overline{E_3}$

$E_1.E_2/E_3.E_4$	0 0	0 1	1 1	1 0
0 0	1	1	0	0
0 1	1	1	0	0
1 1	1	1	0	0
1 0	1	1	0	0

## Détermination de fonction logique

### Recherche de quartets

Quatre 1 voisins peuvent être regroupés car deux variables changent d'état, celles-ci disparaissent dans le terme qui résulte.

		1						1											
		1																	
		1																	
		1																	

Exemple:  $S = E_1 \cdot E_2 + \overline{E_2} \cdot \overline{E_4}$

$E_1 \cdot E_2 / E_3 \cdot E_4$	0 0	0 1	1 1	1 0
0 0	1	0	0	1
0 1	0	0	0	0
1 1	1	1	1	1
1 0	1	0	0	1



## Détermination de fonction logique

### Recherche de doublets

Deux 1 voisins peuvent être regroupés car une seule variable change d'état, celle-ci disparaît dans le terme qui résulte.

		1	
		1	

1			1

Exemple:  $S = \overline{E_1} \cdot \overline{E_2} \cdot \overline{E_3} + \overline{E_2} \overline{E_3} \cdot E_4 + E_2 \cdot E_3 \cdot E_4 + E_1 \cdot E_3 \cdot \overline{E_4}$

$E_1 \cdot E_2 / E_3 \cdot E_4$	0 0	0 1	1 1	1 0
0 0	1	1	0	0
0 1	0	0	1	0
1 1	0	0	1	1
1 0	0	1	0	1

## Conclusion

### Savoir

Vous devez être capables :

- de manipuler des équations logiques,
- de manipuler des fonctions combinatoires,
- de concevoir un câblage de portes logiques.

### Problématique

Il est nécessaire d'utiliser d'autres formes de représentation d'un mécanisme.

- *Problème: Comment concevoir une commande séquentielle*
- **Perspectives:** Réaliser des algorithmes séquentiels capables de piloter le comportement d'un système.