# 1.1 - Logique booléenne

# 1.1 - Logique booléenne

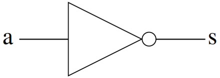
- point de vue mathématique : anneau (en fait corps) à deux valeurs (addition, multiplication, éléments neutres, distributivité, etc.)
- valeurs 0 et 1 : symboles (vrai-faux, haut-bas, marche-arrêt, ouvert-fermé)
- représentation : $0 \simeq 0V$ (masse, VSS, GND), $1 \simeq +5V$ (alim, VDD, VCC)
- logique booléenne/binaire/combinatoire
- opérateurs/portes logiques : $+$(OU) $*$(ET) ˜(NON)

## 1.1 - Logique booléenne

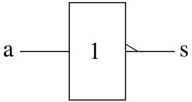
- NON/NOT : complémentaire d'une valeur logique
- table de vérité :

| a | s |
|---|---|
| 0 | 1 |
| 1 | 0 |

- symboles




symbole usuel



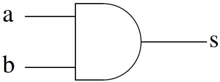

symbole ANSI

- notation : $s = \overline{a}$ ou $s = \neg a$

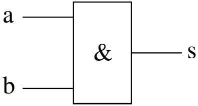
- ET/AND : vrai ssi ses deux entrées sont vraies
- table de vérité :

| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- symboles




symbole usuel



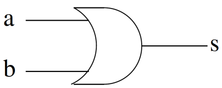

symbole ANSI

- notation : $s = a.b$ ou $s = a \wedge b$

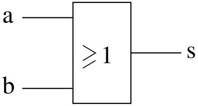
- OU/OR : faux ssi ses deux entrées sont fausses
- table de vérité :

| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

- symboles




symbole usuel



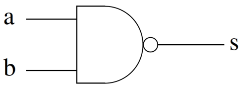

symbole ANSI

- notation : $s = a + b$ ou $s = a \vee b$

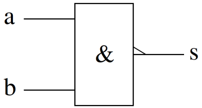
- NON-ET/NAND : faux ssi ses deux entrées sont vraies
- table de vérité :

| a | b | s |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- symboles



symbole usuel



symbole ANSI

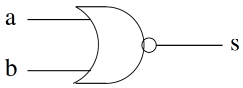
- notation : $s = \overline{a.b}$ ou $s = a \downarrow b$

# 1.1 - Logique booléenne

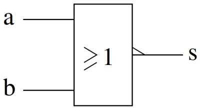
- NON-OU/NOR : vrai ssi ses deux entrées sont fausses
- table de vérité :

| a | b | s |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

- symboles




symbole usuel



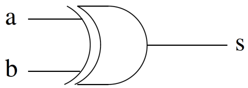

symbole ANSI

- notation : $s = \overline{a + b}$ ou $s = a \uparrow b$

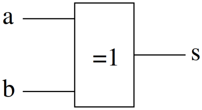
- OU-EXCLUSIF/XOR : vrai ssi ses deux entrées sont différentes
- table de vérité :

| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- symboles




symbole usuel



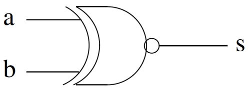

symbole ANSI

- notation : $s = a \oplus b$ ou $s = a \forall b$

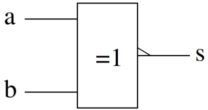
- NI-EXCLUSIF/XNOR : faux ssi ses deux entrées sont différentes
- table de vérité :

| a | b | s |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- symboles




symbole usuel



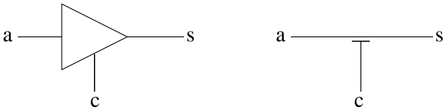

symbole ANSI

- notation : $s = \overline{a \oplus b}$

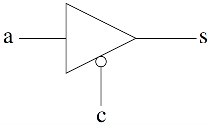
## 1.1 - Logique booléenne

- ET-MULTIPLE, OU-MULTIPLE, XOR-multiple, etc.
- généralisation à des fonctions ayant un nombre quelconque d'entrées
- ET-MULTIPLE : vrai ssi toutes les entrées sont vraies
- OU-MULTIPLE : faux ssi toutes les entrées sont fausses
- XOR-MULTIPLE : vrai ssi le nombre d'entrées vraies est impair

- PORTE TROIS-ETATS/TRI-STATE
- agit en interrupteur
- deux entrées et une sortie
- entrées : signal $a$ + contrôle $c$
- contrôle : si $c = 1$ alors $s = a$, si $c = 0$ alors $s = Z$ (haute impédance, sortie déconnectée de l'entrée)
- symbole




- contrôle inversé



## 1.2 - Logique combinatoire

# 1.2 - Logique combinatoire

*Définitions, propriétés*

- fonction logique ou booléenne : une ou plusieurs entrées, combinaison de portes logiques
- système combinatoire :
  - à base de fonctions logiques
  - boucle ouverte (aucune sortie n'est utilisée comme entree)
  - pas de notion de temps
  - une seule sortie (vecteur) par ensemble d'entrées
  - table de vérité
- combien de fonctions logiques à $n$ entrées :
  - le $n$-uplet des entrées peut avoir $2^n$ valeurs
  - la sortie booléenne peut être arbitrairement fixée pour chacune, d'où $2^{2^n}$ fonctions possibles
  - exemple : avec $n = 4$, il existe 65536 fonctions possibles

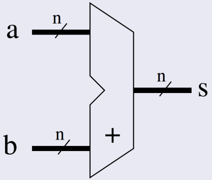
### Exercices

1. On appelle fonction majorité toute fonction booléenne qui à chacune de ses entrées $(x_0, x_1, \ldots, x_{n-1})$ associe 1 (respectivement 0) si ses variables $x - i$ qui sont égales à 1 (respectivement 0) sont majoritaires.
   - cette fonction est-elle complètement ou incomplètement définie ?
   - combien existe-t-il de fonctions majorité ?
2. Montrez que pour toute fonction booléenne $f(x_0, x_1, \ldots, x_{n-1})$ on a :

$$\begin{aligned} f(x_0, x_1, \ldots, x_{n-1}) &= x_i f(x_0, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_{n-1}) \\ &\quad + \overline{x_i} f(x_0, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_{n-1}) \end{aligned}$$

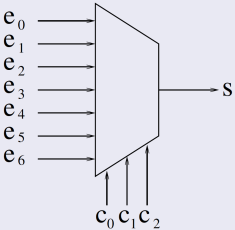
### Additionneur

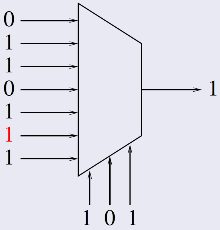
Réalise l'addition de deux nombres entiers sur $n$ bits.



## Multiplexeur

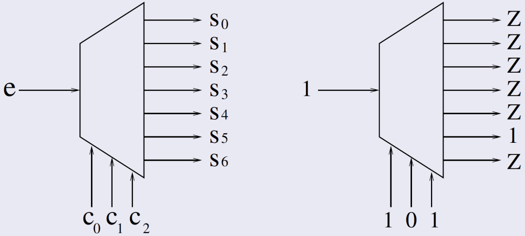
- transmet plusieurs signaux en entrée sur une seule sortie, en sélectionnant une des entrées en fonction du contrôle
- $2^n$ entrées sélectionnables par contrôle sur $n$ bits
- les $n$ bits de contrôle donnent le numéro de l'entrée sélectionnée



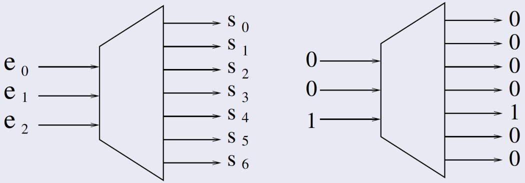
## Démultiplexeur

- rôle inverse du multiplexeur
- choisit la sortie vers laquelle l'entrée est transmise



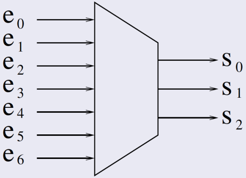
## Décodeur

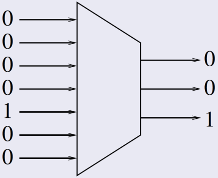
- $n$ entrées et $2^n$ sorties
- entrée : code une valeur de 0 à $2^n - 1$
- la sortie ayant ce numéro passe à 1 (les autres à 0)
- élément essentiel dans la réalisation des mémoires



## Encodeur

- rôle inverse du décodeur
- $2^n$ entrées et $n$ sorties
- une seule entrée à 1
- la sortie code le numéro de cette entrée



## 1.2 - Logique combinatoire

*Analyse d'un système combinatoire*

*Analyse d'un système combinatoire*

- forme canonique : disjonction de conjonctions de littéraux (OU de ET)
- autant de termes que de '1' dans la table de vérité
- exemple

| a | b | c | s |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

donne $\overline{a}.\overline{b}.c + \overline{a}.b.\overline{c} + a.\overline{b}.c + a.b.\overline{c} + a.b.c$

Théorèmes de Boole

- $a.0 = 0$
- $a.1 = a$
- $a.a = a$
- $a.\overline{a} = 0$
- $a + 0 = a$
- $a + 1 = 1$
- $a + a = a$
- $a + \overline{a} = 1$

Formules de De Morgan

- $\overline{a + b} = \overline{a}.\overline{b}$
- $\overline{a.b} = \overline{a} + \overline{b}$

**Exercices**

1. Montrez, dans l'algèbre de Boole $(\{0,1\},+,.,\bar{\ })$ :
   - $a+a.b=a$
   - $a.(a+b)=a$
   - $a.b+\overline{a}.b=b$
   - $(a+b).(\overline{a}+b)=b$
2. Développez les formules suivantes :
   - $\overline{a+b+c.d}$
   - $\overline{a+b}+\overline{c.d}$
3. Réduisez la formule suivante :
   - $\overline{a}.\overline{b}.c+\overline{a.b}.c+\overline{a}.b.c+a.b.c$

## 1.2 - Logique combinatoire

### *Tableaux de Karnaugh*

- permet de trouver une forme compacte en regroupant les cases contenant 1
- obtenu à partir de la table de vérité

Exemples :

- 2 entrées : tableau de Karnaugh de s

| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| a\b | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# 1.2 - Logique combinatoire

## *Tableaux de Karnaugh*

• 4 entrées $a_1a_0b_1b_0$ : tableau de Karnaugh de s

| $a_1$ | $a_0$ | $b_1$ | $b_0$ | s |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

$\Rightarrow$

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 0 | 1 | 1 | 0 |

## 1.2 - Logique combinatoire

*Tableaux de Karnaugh*

- 4 entrées $a_1a_0b_1b_0$ : tableau de Karnaugh de s

Entrées toujours codées dans l'ordre croissant du codage binaire (ici de 0000 à 1111)

| | $a_1$ | $a_0$ | $b_1$ | $b_0$ | s |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 0 |

Entêtes de ligne et de colonnes codées dans l'ordre croissant du codage de **Gray**

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 0 | 1 | 1 | 0 |

# 1.2 - Logique combinatoire

## *Tableaux de Karnaugh*

• 4 entrées $a_1a_0b_1b_0$ : tableau de Karnaugh de s

| | $a_1$ | $a_0$ | $b_1$ | $b_0$ | s |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 0 |

$\Rightarrow$

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 0 | 1 | 1 | 0 |

## *Tableaux de Karnaugh*

• Pourquoi le codage de Gray pour les tableaux de Karnaugh ?

Les codages des entrées de 2 cases voisines ont un seul bit qui les différencie

Important : la première et la dernière case de chaque ligne et de chaque colonne sont également considérées comme voisines (un seul bit différencie leur codage)

• exemples

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 |

$$s = \bar{a}_1 a_0 b_1 \bar{b}_0 + a_1 a_0 b_1 \bar{b}_0 = a_0 b_1 \bar{b}_0 (a_1 + \bar{a}_1) = a_0 b_1 \bar{b}_0$$

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 |

$$s = a_1 a_0 \bar{b}_0$$

# 1.2 - Logique combinatoire

*Tableaux de Karnaugh*

• Plus le regroupement de cases voisines est grand, plus l'équation résultante est simple

• Exemples de groupements de 4 cases voisines :

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

$$s = \overline{b}_1 b_0$$

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

$$s = a_0 b_0$$

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

$$s = \overline{a}_0 \overline{b}_1$$

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 1 |

$$s = \overline{a}_0 \overline{b}_0$$

# 1.2 - Logique combinatoire

*Tableaux de Karnaugh*

• Exemples de groupements de 8 cases voisines :

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

$s = a_0$

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 |

$s = \bar{b}_0$

• Les groupements ne se font que par $2^n$ cases (2, 4, 8, 16, …)

• Exemples de groupements non valides :

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 1 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

- Les cases contenant 1 peuvent être utilisées plusieurs fois par des groupements différents pour reconstituer l'équation de la sortie
- Objectifs :
  - maximiser la taille des groupements
  - minimiser le nombre de groupements
- Exemples complets :

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

# 1.2 - Logique combinatoire

## *Tableaux de Karnaugh*

- Les cases contenant 1 peuvent être utilisées plusieurs fois par des groupements différents pour reconstituer l'équation de la sortie
- Objectifs :
  - maximiser la taille des groupements
  - minimiser le nombre de groupements
- Exemples complets :

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

| $a_1a_0\backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

# 1.2 - Logique combinatoire

## *Tableaux de Karnaugh*

- Les cases contenant 1 peuvent être utilisées plusieurs fois par des groupements différents pour reconstituer l'équation de la sortie
- Objectifs :
  - maximiser la taille des groupements
  - minimiser le nombre de groupements
- Exemples complets :

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

$$s = \boxed{a_1 a_0 \bar{b}_1} + \boxed{a_0 b_0} + \bar{a}_1 \bar{a}_0 b_1 \bar{b}_0$$

| $a_1a_0 \backslash b_1b_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

$$s = \boxed{a_1 a_0} + a_0 b_1 + \boxed{a_0 b_0} = a_0(a_1 + b_1 + \bar{b}_0)$$

### Eléments indéterminés

- une fonction peut ne pas être définie pour certaines combinaisons de variables (impossibles compte tenu du reste de l'architecture)
- on met alors un symbole (ensemble vide) dans le tableau de Karnaugh
- ce symbole peut etre considéré comme un 0 ou un 1
  - 1 si ça permet de maximiser la taille d'une groupement
  - 0 si ça permet de minimiser le nombre de groupements

### Exercices

On note $f_{n,i}$ la fonction booléenne qui à $(x_0, x_1, \ldots, x_{n-1})$ associe la valeur 1 si et seulement si $x_{n-1}x_{n-2}\ldots x_1x_0$ est le codage binaire de $i$.

Donnez les formes canoniques, les tableaux de Karnaugh et les formes simplifiées des fonctions booléennes suivantes :

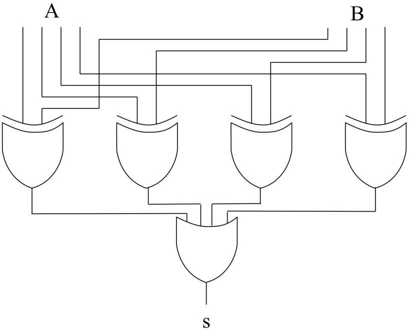
- $f_{3,3} + f_{3,4} + f_{3,6} + f_{3,7}$
- $f_{4,0} + f_{4,1} + f_{4,3} + f_{4,11} + f_{4,12} + f_{4,14} + f_{4,15}$

Dessinez les schémas logiques correspondants.

# 1.2 - Logique combinatoire

*Exemple simple : comparateur $A \neq B$*

- un comparateur sur n bits avec n comparateurs sur 1 bit
- comparateur sur 1 bit : OU-EXCLUSIF ( 1 si les deux entrées sont différentes )
- on compare tous les bits 1 à 1 puis on fait un OU sur les résultats :



# 1.2 - Logique combinatoire

## *Exemple complet : additionneur*

• idée : combinaison de plusieurs circuits travaillant sur 1 bit

• avantages : facile à réaliser, travail avec une taille de n bits (n quelconque), conception claire, schéma logique clair, réalisation matérielle du circuit simple

• circuit primitif : additionneur 1 bit complet avec prise en compte de la retenue précédente (Full Adder)

$$
\begin{array}{r}
\scriptstyle 0\,1\,1\,1\,1\,1\,0\phantom{0} \\
0010\;1011 \\
+\;1011\;0110 \\
\hline
1110\;0001
\end{array}
$$

# 1.2 - Logique combinatoire

*Exemple complet : additionneur*

Table de vérité :

| a | b | c | r | s |
|---|---|---|---|---|
| 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 |

Tableau de Karnaugh pour r :

| a\bc | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |

$$r = ac + bc + ab$$

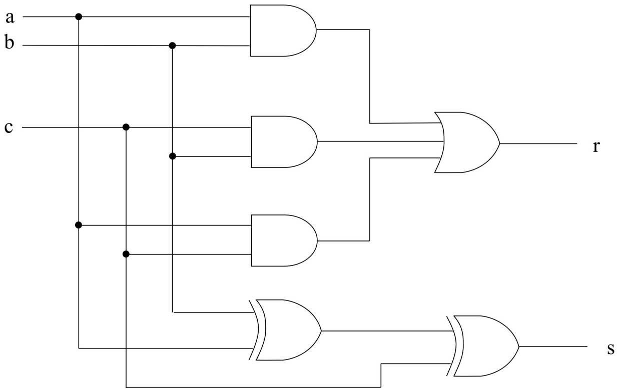
Tableau de Karnaugh pour s :

| a\bc | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

$$s = a \oplus b \oplus c$$

*Exemple complet : additionneur (FA : full adder)*

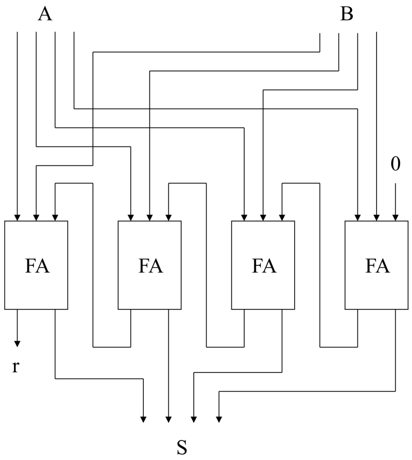
Schéma logique :



# 1.2 - Logique combinatoire

## *Exemple complet : additionneur*

Schéma logique d'un additionneur 4 bits constitué d'additionneurs complets :



**Exercice**

Concevez un comparateur de deux entiers codés sur 2 bits $a = (a_1 a_0)$ et $b = (b_1 b_0)$ dont la sortie $c$ égale à 1 si et seulement si $a > b$. Déterminez la table de vérité, la forme canonique, la table de Karnaugh, une formule booléenne la plus simplifiée possible, et le schéma logique associé.

- calcul au cours du temps
- temps discret ("pas" de temps)
- nécessite données mémorisées (état)

## 1.3 - Logique séquentielle

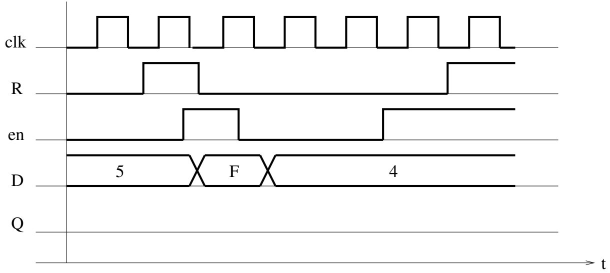
*Schéma général*

- entrées/sorties
- bloc de logique combinatoire
- ressources de mémorisation
- rebouclage : prochain état dépend de l'état précédent

# 1.3 - Logique séquentielle

## *Chronogrammes*

- une ligne par signal : état haut ou bas (ou bus : valeur)
- en général signal d'horloge
- représentation des états binaires au cours du temps
- temps de stabilisation des signaux souvent ignoré



- le système ne mémorise son entrée/état qu'au signal de synchronisation d'une horloge partagée par toute l'archi
- on note alors $X_n$ la valeur du signal $X$ après le $n$-ième front d'horloge

**Machine de Moore**

la sortie ne dépend que de l'état, i.e. est directement issue des éléments mémorisés (via combinatoire éventuelle), qui eux dépendent de l'entrée et d'eux-mêmes

### Machine de Moore simplifiée

la sortie est exactement l'état stocké

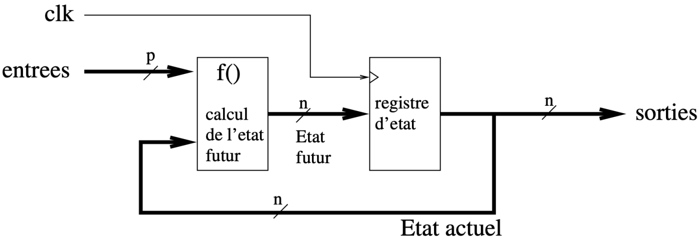
### Machine de Mealy

la sortie dépend de l'état stocké et de l'entrée

En fait, les machines de Moore et de Mealy sont équivalentes.

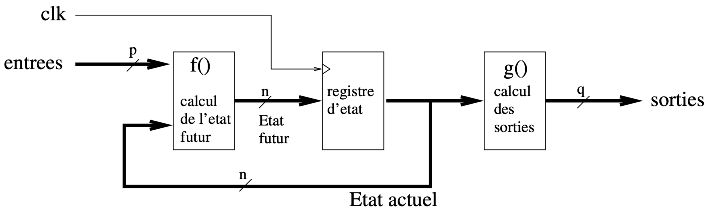
## 1.3 - Logique séquentielle

*Machine de Moore simplifiée*



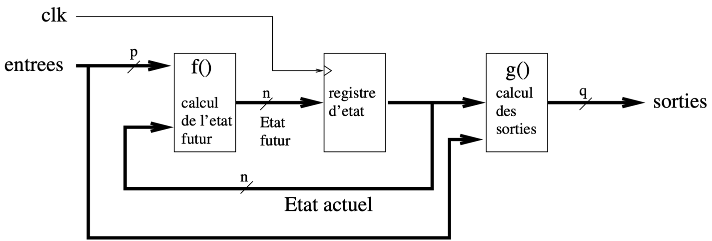
# 1.3 - Logique séquentielle

## *Machine de Moore*



# 1.3 - Logique séquentielle

*Machine de Mealy*



# 1.3 - Logique séquentielle

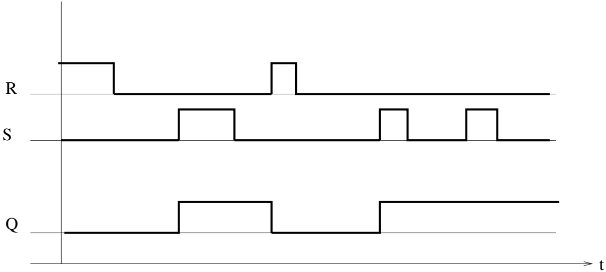
*Mémoires élémentaires : les bascules*

# 1.3 - Logique séquentielle

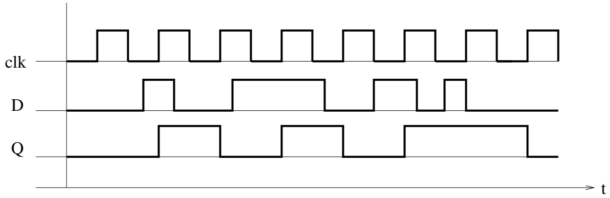
## *Bascules*

- Mémoires élémentaires (1 bit).
- Bascules synchrones (flip-flop) et asynchrones (latch).
- Principales bascules :
  - Bascule RS (asynchrone) : entrée R (reset) et S (set) permettent de changer l'état de la sortie (conservé lorsque R=S=0)
  - Bascule D (synchrone) : changement d'état possible seulement sur front montant de l'horloge, la sortie "stockant" alors l'entrée
  - Bascule D passante (asynchrone) : tant que clk=1, sortie=entrée
  - Bascule JK (synchrone) : deux entrées J et K, si J≠K alors sortie=J, si J=K=1 alors inversion de la sortie

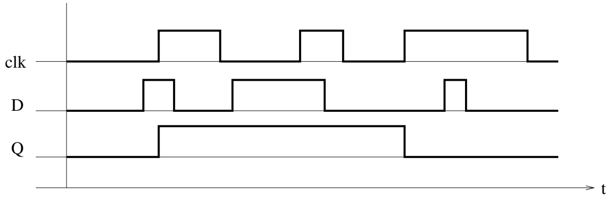
Chronogramme bascule RS :



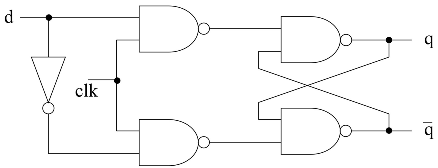
Chronogramme bascule D synchrone :



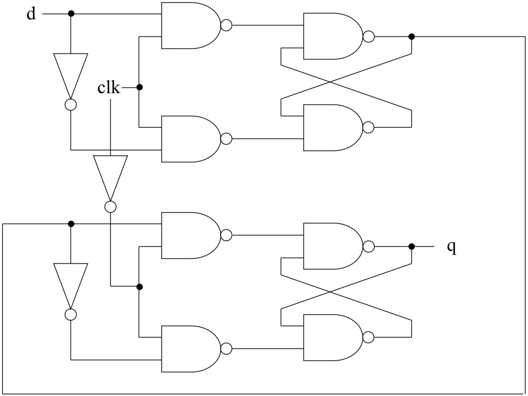
Chronogramme bascule D synchrone (horloge peut être un signal quelconque) :



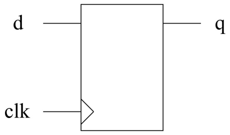
**Réalisation : bascule D asynchrone**



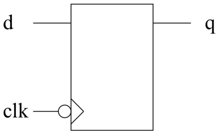
**Réalisation : bascule D synchrone**



Symbole d'une bascule D sur front montant :




Symbole d'une bascule D sur front descendant :



**Registre**

- circuit capable de mémoriser 1 mot (n bits)
- en général situé dans le processeur (local, rapide)
- assembleur accède aux registres

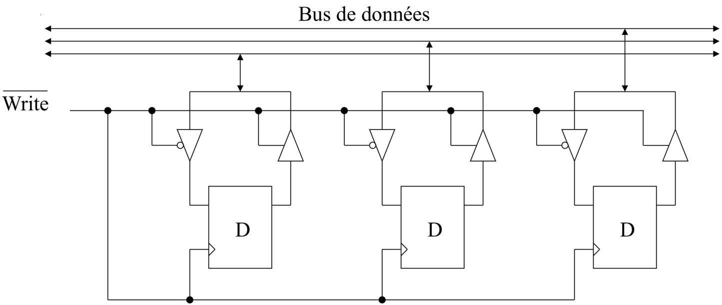
**Mémoire**

- ensemble de m registres de taille n
- décodeur d'adresse
- signal écriture/lecture

# 1.3 - Logique séquentielle

## *Exemples*

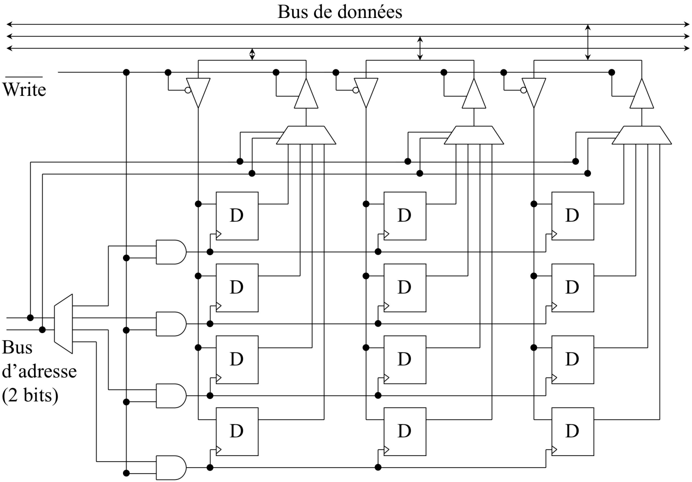
**Registre 3 bits :**



# 1.3 - Logique séquentielle

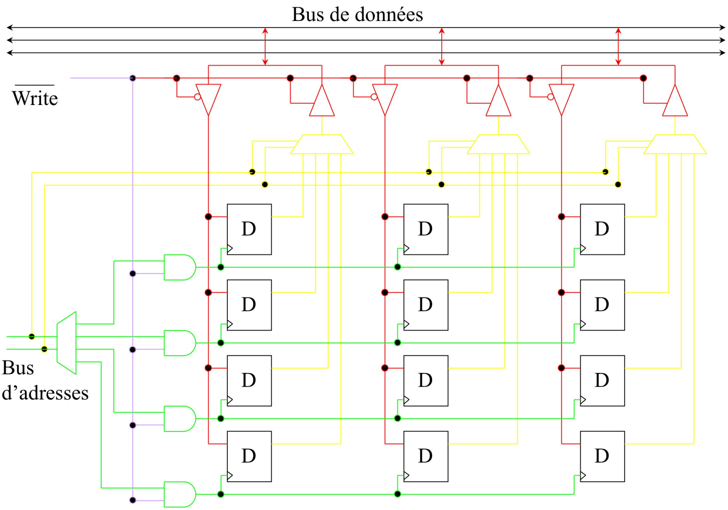
## Exemples

**Mémoire de 4 mots de 3 bits :**



# 1.3 - Logique séquentielle

## *Exemples*



**Compteur**

- circuit logique séquentiel dont l'état parcourt un cycle

- peut servir à déclencher des actions les unes après les autres

- peut servir à enchaîner les instructions d'une séquence d'un programme

- peut contrôler une boucle

- peut parcourir les adresses consécutives d'une mémoire

- … etc

**Compteur de 0 à 7**

8 états donc 3 bascules sont nécessaires ($D_0$, $D_1$ et $D_2$).

Table de transitions :

| t | | | | t+1 | | | |
|---|---|---|---|---|---|---|---|
| Etat | $D_2$ | $D_1$ | $D_0$ | Etat | $D_2$ | $D_1$ | $D_0$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 2 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 3 | 0 | 1 | 1 |
| 3 | 0 | 1 | 1 | 4 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 5 | 1 | 0 | 1 |
| 5 | 1 | 0 | 1 | 6 | 1 | 1 | 0 |
| 6 | 1 | 1 | 0 | 7 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

# 1.3 - Logique séquentielle

## *Exemples*

Pour $D_2^{t+1}$ :

| $D_2 \backslash D_1D_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |

$\Longrightarrow$ $D_2^{t+1} = D_2 \oplus (D_1 . D_0)$

Pour $D_1^{t+1}$ :

| $D_2 \backslash D_1D_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |

$\Longrightarrow$ $D_1^{t+1} = D_1 \oplus D_0$

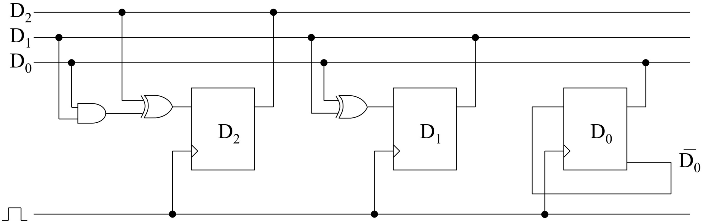
Pour $D_0^{t+1}$ :

| $D_2 \backslash D_1D_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |

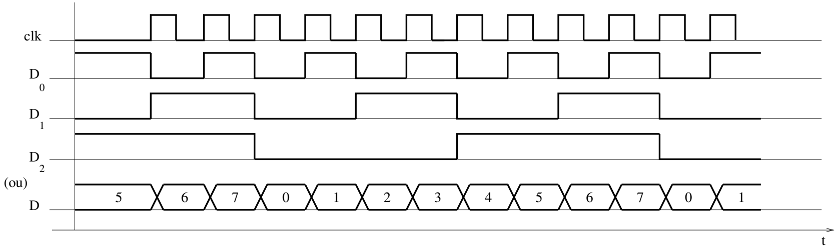
$\Longrightarrow$ $D_0^{t+1} = \overline{D_0}$

## 1.3 - Logique séquentielle

*Exemples*



Chronogramme du compteur :



### Exercice

Concevez un compteur synchrone de 0 à 11, sans rebouclage à zéro, avec un signal de reset synchrone (la remise à zéro se fait au front d'horloge montant lorsque le signal de reset est égal à 1).