# Architecture des ordinateurs

## (X31I050)

Frédéric Goualard

Laboratoire d'Informatique de Nantes-Atlantique, UMR CNRS 6241
Bureau 112, bât. 11
Frederic.Goualard@univ-nantes.fr

# Circuits logiques

*Circuits combinatoires*

# Fonctions logiques

Ordinateur *numerique* binaire ➟ exprimer toutes les opérations sous forme de fonctions logiques :

Fonction $n$-aire $f$ :

$$\begin{aligned} f\colon \{0,1\}^n &\to \{0,1\} \\ (a_1,\ldots,a_n) &\mapsto f(a_1,\ldots,a_n) \end{aligned}$$

complétement ou incomplètement définie

$f(a_1, a_2, a_3)$

| $a_1$ | $a_2$ | $a_3$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Définition complète

$g(a_1, a_2, a_3)$

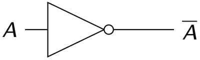
| $a_1$ | $a_2$ | $a_3$ | $g$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | $x$ |
| 1 | 0 | 0 | $x$ |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | $x$ |

Définition incomplète

**Tables de vérité**

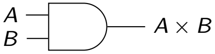
# Fonctions logiques (1/3)

Négation $\neg A$



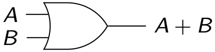

| $A$ | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

ET logique $A \wedge B$




| $A$ | $B$ | $A \times B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OU logique $A \vee B$




| $A$ | $B$ | $A + B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Algèbre de Boole

| | |
|---|---|
| Associativité de $\vee$ | $A+(B+C)=(A+B)+C$ |
| Associativité de $\wedge$ | $A\times(B\times C)=(A\times B)\times C$ |
| Commutativité de $\vee$ | $A+B=B+A$ |
| Commutativité de $\wedge$ | $A\times B=B\times A$ |
| Distributivité de $\wedge$ p.r. $\vee$ | $A\times(B+C)=(A\times B)+(A\times C)$ |
| Distributivité de $\vee$ p.r. $\wedge$ | $A+(B\times C)=(A+B)\times(A+C)$ |
| Identité pour $\vee$ | $A+0=A$ |
| Identité pour $\wedge$ | $A\times 1=A$ |
| Élément absorbant pour $\wedge$ | $A\times 0=0$ |
| Élément absorbant pour $\vee$ | $A+1=1$ |
| Idempotence pour $\vee$ | $A+A=A$ |
| Idempotence pour $\wedge$ | $A\times A=A$ |
| Absorption | $A\times(A+B)=A$ |
| Absorption | $A+(A\times B)=A$ |
| Complémentarité | $A\times\overline{A}=0$ |
| Complémentarité | $A+\overline{A}=1$ |
| Complémentarité | $\overline{\overline{A}}=A$ |
| Simplification | $A+(\overline{A}\times B)=A+B$ |
| Simplification | $A\times(\overline{A}+B)=A\times B$ |
| Simplification | $(A\times B)+(A\times\overline{B})=A$ |
| Simplification | $(A+B)\times(A+\overline{B})=A$ |

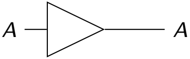
# Dualité : lois de De Morgan

Dualité $\vee$ / $\wedge$ :

$$\overline{A+B} = \overline{A} \times \overline{B}$$
$$\overline{A \times B} = \overline{A} + \overline{B}$$

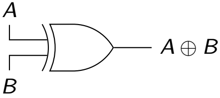
Passage d'un connecteur à l'autre

Identité $A$



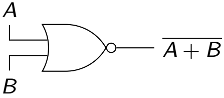

| $A$ | $A$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

OU exclusif $A \oplus B$



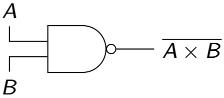

| $A$ | $B$ | $A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

« NON OU » (NOR) $\overline{A + B}$




| $A$ | $B$ | $\overline{A + B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

« NON ET » (NAND) $\overline{A \times B}$




| $A$ | $B$ | $\overline{A \times B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Fonctions logiques (3/3)

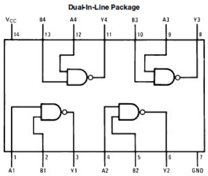
- Nombreuses autres fonctions logiques :
  - « NON OU »exclusif
  - Fonctions ternaires
    - « NOR »à trois entrées (*Apollo Guidance Computer*)
    - Fonction de Toffoli
    - ...
  - ...

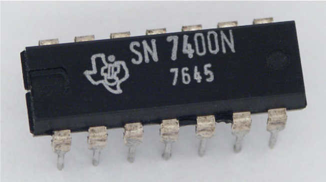
## Définition

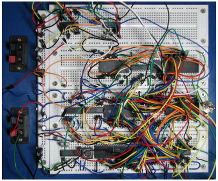
*Un ensemble de connecteurs logiques $\mathcal{S}$ est fonctionnellement complet si toute fonction logique peut s'écrire en utilisant uniquement les connecteurs présents dans $\mathcal{S}$.*

Ensemble fonctionnellement complet minimal : on ne peut retirer de connecteur de $\mathcal{S}$ sans perdre la complétude fonctionnelle

- L'ensemble (ET, OU, NON) est fonctionnellement complet (non minimal)
- (OU, NON) : fonctionnellement complet et minimal
- (NAND) et (NOR) : fonctionnellement complets et minimaux



# Produits de sommes / sommes de produits

- En général, pas de représentation unique d'une fonction logique :

$$A B \overline{C} + C D + C \overline{D} \equiv A B + C$$
$$A \oplus B \equiv A \overline{B} + \overline{A} B$$

- Nombreuses formulations équivalentes utilisant les différents connecteurs logiques

Forme somme. Somme (OU) de termes

$$A \overline{B} C + A \overline{B} C + A \overline{C}$$

Forme produit. Produit (ET) de termes

$$(A + B + C) \times (A + \overline{B} + C) \times (\overline{A} + \overline{B} + C)$$

## Définition (Forme canonique produit de sommes)

Une expression logique est sous forme canonique « produit de sommes » si :

- Toutes les variables apparaissent dans chaque facteur ($A$ ou $\overline{A}$) ;
- Chaque facteur est une disjonction (OU) de variables ;
- Tous les facteurs sont différents ;
- Les facteurs sont connectés par des conjonctions (ET).

Exemple :

$$f(A, B, C) = (\overline{A} + \overline{B} + C) \times (\overline{A} + \overline{B} + C)$$

## Définition (Forme canonique somme de produits)

Une expression logique est sous forme canonique « somme de produits » si :

- Toutes les variables apparaissent dans chaque facteur ($A$ ou $\overline{A}$) ;
- Chaque facteur est une conjonction (ET) de variables ;
- Tous les facteurs sont différents ;
- Les facteurs sont connectés par des disjonctions (OU).

Exemple :

$$f(A, B, C) = \overline{AB}C + \overline{AB}C$$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$f(A,B,C) =$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C}$$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C}$$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$\overline{\overline{A}\,\overline{B}\,\overline{C}}$

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$\overline{\overline{A}\,\overline{B}\,\overline{C}} = A+B+C$ (De Morgan)

$$f(A,B,C) = \overline{A}B\overline{C}+AB\overline{C}+ABC$$

# Formes canoniques et table de vérité

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$\begin{aligned} f(A,B,C) = &(A+B+C)\times \\ &(A+B+\overline{C})\times \\ &(A+\overline{B}+\overline{C})\times \\ &(\overline{A}+B+C)\times \\ &(\overline{A}+B+\overline{C}) \end{aligned}$$

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

# Circuits combinatoires


Boîte noire dont les sorties $S_i$ ($i \in \{1, \ldots, m\}$) ne dépendent que des entrées $E_j$ ($j \in \{1, \ldots, n\}$)

- Objectif : réaliser un circuit à partir d'une fonction booléenne
- Méthode :
  1. Calculer une forme canonique à partir de la table de vérité
  2. Simplifier l'expression
     - Manipulations algébriques
     - Tableau de Karnaugh
     - ...
  3. Associer une porte logique à chaque opérateur

Synthèses particulières :

- Circuit avec uniquement des NOR
- Circuit avec uniquement des NAND

# Tableau de Karnaugh (1/2)

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

| $A$ \ $BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

# Tableau de Karnaugh (1/2)

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

| $A$ \ $BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

Simplifications :

- $ABC + AB\overline{C} \equiv AB$

# Tableau de Karnaugh (1/2)

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

| $A \backslash BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

Simplifications :

- $ABC + AB\overline{C} \equiv AB$
- $\overline{A}B\overline{C} + AB\overline{C} = B\overline{C}$

# Tableau de Karnaugh (1/2)

| $A$ | $B$ | $C$ | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(A,B,C) = \overline{A}B\overline{C} + AB\overline{C} + ABC$$

| $A \backslash BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

Simplifications :

- $ABC + AB\overline{C} \equiv AB$
- $\overline{A}B\overline{C} + AB\overline{C} = B\overline{C}$

D'où : $f(A,B,C) = AB + B\overline{C}$

# Tableau de Karnaugh (2/2)

Réduction de fonction booléenne $\Sigma(\Pi)$ (minimisation du nombre de termes)

| $A \backslash BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

- Couvrir tous les « 1 »
- Faire le minimum de groupes
- Faire des groupes les plus larges possibles (côtés « en $2^x$ »)

# Tableau de Karnaugh (2/2)

Réduction de fonction booléenne $\Sigma(\Pi)$ (minimisation du nombre de termes)

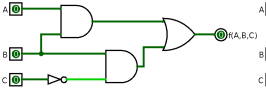
| $A \backslash BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |

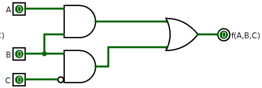
- Couvrir tous les « 1 »
- Faire le minimum de groupes
- Faire des groupes les plus larges possibles (côtés « en $2^x$ »)

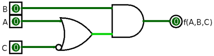
# Exemple de synthèse

| $A \backslash BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |

$$f(A, B, C) = AB + B\overline{C} = B(A + \overline{C})$$





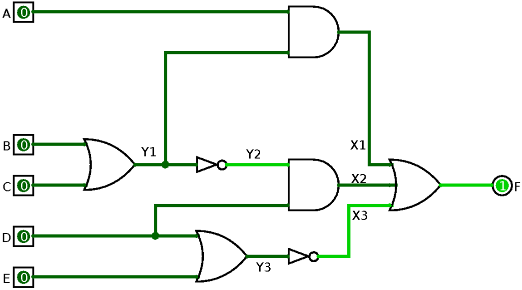


# Analyse de circuit

**Objectif :** obtenir une fonction booléenne à partir d'un schéma logique

**Méthode :**

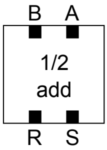
1. Partir de la sortie du circuit ;
2. Remplacer la porte par l'opérateur booléen correspondant ;
3. Simplifier ;
4. Remonter le circuit récursivement jusqu'aux entrées.




$$
\begin{aligned}
F &= X_1 + X_2 + X_3 \\
&= A \times Y_1 + Y_2 \times D + \overline{Y_3} \\
&= A \times (B + C) + \overline{Y_1} \times D + \overline{D + E} \\
&= A \times (B + C) + \overline{B + C} \times D + \overline{D + E}
\end{aligned}
$$

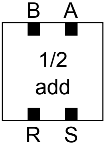
# Demi-additionneur 1 bit

$$
\begin{array}{cccccccccl}
 & & 1 & 0 & 1 & 1 & 0 & \color{red}{\mathbf{1}} & \longleftarrow & \color{red}{A} \\
+ & & 0 & 1 & 1 & 0 & 1 & \color{red}{\mathbf{1}} & \longleftarrow & \color{red}{B} \\
 & & \color{gray}{1} & \color{gray}{1} & \color{gray}{1} & \color{gray}{1} & \color{red}{\mathbf{1}} & & \longleftarrow & \color{red}{R} \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & \color{red}{\mathbf{0}} & \longleftarrow & \color{red}{S}
\end{array}
$$



# Demi-additionneur 1 bit

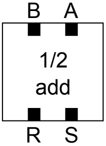
$$
\begin{array}{cccccccccl}
 & & 1 & 0 & 1 & 1 & 0 & \color{red}{\mathbf{1}} & \longleftarrow & \color{red}{A} \\
+ & & 0 & 1 & 1 & 0 & 1 & \color{red}{\mathbf{1}} & \longleftarrow & \color{red}{B} \\
 & & 1 & 1 & 1 & 1 & \color{red}{\mathbf{1}} & & \longleftarrow & \color{red}{R} \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & \color{red}{\mathbf{0}} & \longleftarrow & \color{red}{S}
\end{array}
$$



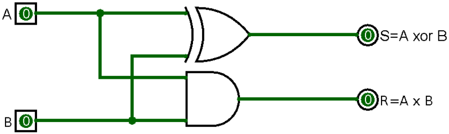

| $A$ | $B$ | $S$ | $R$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# Demi-additionneur 1 bit

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 0 | 1 | 1 | 0 | **1** | ⟵ | $A$ |
| + | | 0 | 1 | 1 | 0 | 1 | **1** | ⟵ | $B$ |
| | | 1 | 1 | 1 | 1 | **1** | | ⟵ | $R$ |
| | 1 | 0 | 0 | 1 | 0 | 0 | **0** | ⟵ | $S$ |

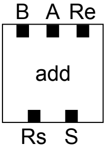



| $A$ | $B$ | $S$ | $R$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |



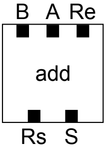
# Additionneur 1 bit

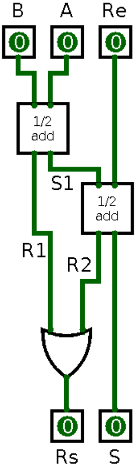
$$
\begin{array}{ccccccccl}
 & & & & & & & 0 & \longleftarrow R_e\\
 & & 1 & 0 & 1 & 1 & 0 & 1 & \longleftarrow A\\
+ & & 0 & 1 & 1 & 0 & 1 & 1 & \longleftarrow B\\
 & & 1 & 1 & 1 & 1 & 1 & & \longleftarrow R_s\\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \longleftarrow S
\end{array}
$$



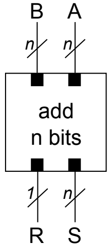
# Additionneur 1 bit

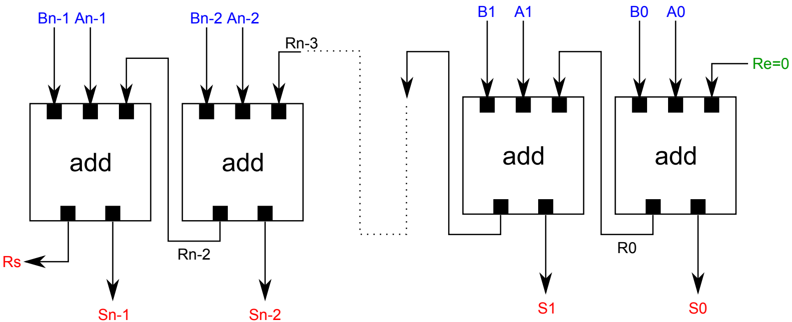
$$
\begin{array}{ccccccccl}
 & & & & & & & \mathbf{0} & \longleftarrow R_e \\
 & & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & \longleftarrow A \\
+ & & 0 & 1 & 1 & 0 & 1 & \mathbf{1} & \longleftarrow B \\
 & & 1 & 1 & 1 & 1 & \mathbf{1} & & \longleftarrow R_s \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} & \longleftarrow S
\end{array}
$$



Chaînage des additionneurs 1 bit :

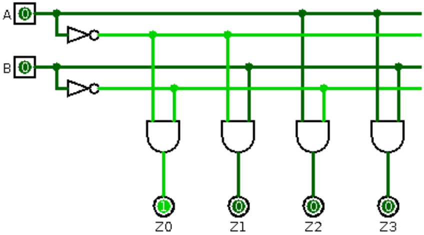


- Une seule sortie à 1 parmi $2^k$ — la $i$-ème — avec $i$ sur $k$ bits

Cas $i = 2$ :

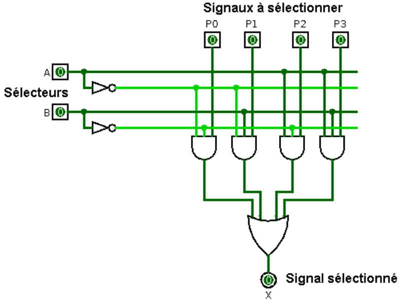
| $A$ | $B$ | $Z_0$ | $Z_1$ | $Z_2$ | $Z_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |



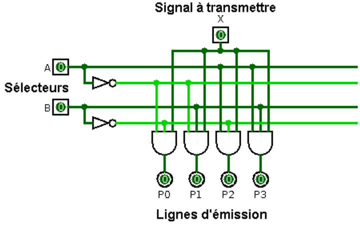
# Multiplexeur/démultiplexeur

**Multiplexage.** Choix du signal à envoyer sur une ligne parmi $n$

**Démultiplexage.** Envoi d'un signal sur une ligne choisie parmi $n$

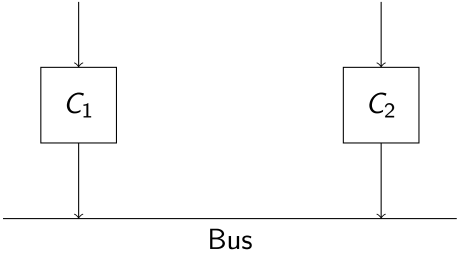



Multiplexeur

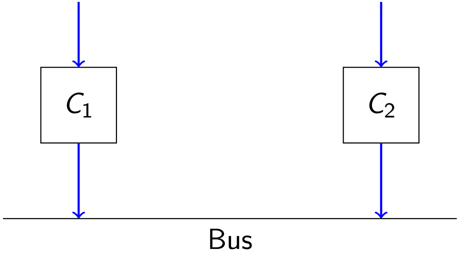



Démultiplexeur

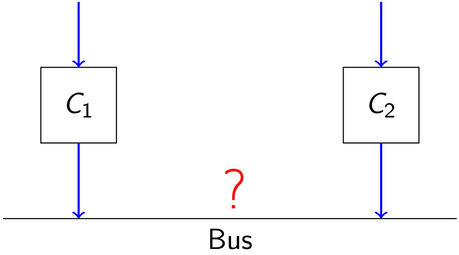
# Logique à trois états



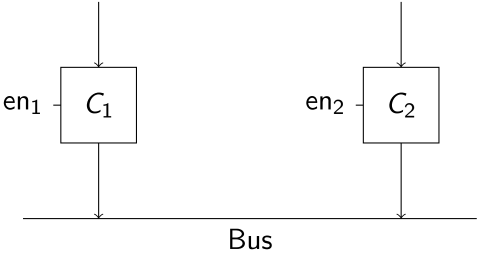
# Logique à trois états



# Logique à trois états



# Logique à trois états




| In | en | Out |
|---|---|---|
| 0 | 0 | $Z$ |
| 1 | 0 | $Z$ |
| $Z$ | 0 | $Z$ |
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| $Z$ | 1 | $Z$ |

- Introduction d'un état à haute impédance $Z$
- Dans l'état $Z$, aucun courant possible en entrée/sortie
- Possibilité de connecter plusieurs circuits à un bus en s'assurant qu'un seul bit *enable* est à 1