Chapitre 3
Logique combinatoire

Version du 29/08/2024

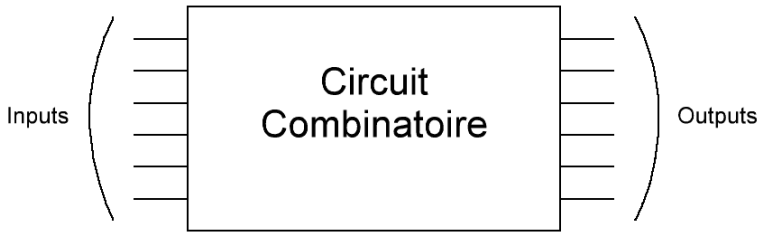
Table des matières

I. Introduction....3
II. Les portes logiques....3
1. La porte NON (NOT)....3
2. La porte ET (AND)....4
3. La porte NON-ET (NAND)....4
4. La porte OU (OR)....4
5. La porte NON-OU (NOR)....5
6. La porte OU EXCLUSIF (XOR)....5
7. La porte NON-OU EXCLUSIF (NXOR)....5
8. Les portes à plus de deux entrées....6
9. Les bulles et les portes....6
10. Les logiques positive et négative....6
11. Associations de portes logiques....10
III. L'algèbre booléenne....11
1. Introduction....11
2. Principaux axiomes et théorèmes....11
3. Principe de dualité....12
4. Simplification algébrique....12
5. Les formes canoniques d'expressions booléennes....14
5.1. Somme de produits et produit de sommes....14
5.2. Somme canonique de produits (première forme canonique)....14
5.2.1. Définition....14
5.2.2. Passer d'une somme de produits à une somme canonique de produits....15
5.2.3. Relation entre une somme canonique de produits et une table de vérité....15
5.3. Produit canonique de sommes (seconde forme canonique)....16
5.3.1. Définition....16
5.3.2. Passer d'un produit de sommes à un produit canonique de sommes....16
5.3.3. Relation entre un produit canonique de sommes et une table de vérité....17
5.4. Passer d'une forme canonique à une autre....18
6. Les diagrammes de Karnaugh....18
6.1. Définition....18
6.2. Simplification d'expressions logiques....21
IV. Résolution de problèmes et synthèse de circuits combinatoires....24
V. Les principaux circuits combinatoires....26
1. L'additionneur....26
2. Le comparateur....26

3. Le décodeur....27
4. Le multiplexeur....28
5. Le démultiplexeur....28

I. Introduction

La logique combinatoire est la discipline qui permet de concevoir les circuits combinatoires. Ces derniers sont des circuits numériques dont l'état des sorties se déduit uniquement à partir de l'état des entrées à un instant donné.



L'état d'une entrée ou d'une sortie est soit l'état logique 0, soit l'état logique 1.

Un circuit combinatoire peut se définir par une table de vérité. Une table de vérité est un tableau qui exprime la valeur des sorties en fonction de la valeur des entrées.

Un circuit combinatoire est constitué uniquement de portes logiques.

II. Les portes logiques

Les portes logiques sont les éléments de base des circuits combinatoires.

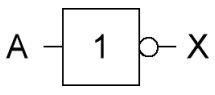
Il existe deux symboles pour chaque porte. Les symboles à forme rectangulaire et les symboles à forme distincte. Dans ce cours, **nous utiliserons les symboles à forme distincte**. Néanmoins, il est nécessaire de bien connaître les symboles à forme rectangulaire, car ils sont présents dans un grand nombre d'ouvrages et de documentations techniques.

1. La porte NON (*NOT*)

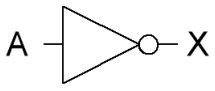
$\mathbf{X = \overline{A}}$

A	X
0	1
1	0

Table de vérité



Symbole à forme rectangulaire



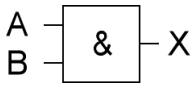
Symbole à forme distincte

2. La porte ET (*AND*)

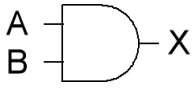
X = A.B

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité



Symbole à forme rectangulaire



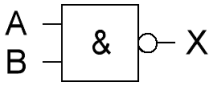
Symbole à forme distincte

3. La porte NON-ET (*NAND*)

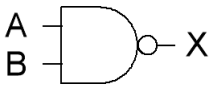
X = $\overline{\text{A.B}}$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Table de vérité



Symbole à forme rectangulaire



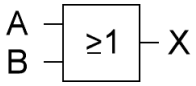
Symbole à forme distincte

4. La porte OU (*OR*)

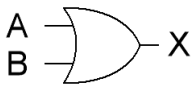
X = A + B

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Table de vérité



Symbole à forme rectangulaire



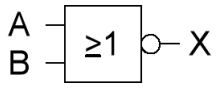
Symbole à forme distincte

5. La porte NON-OU (*NOR*)

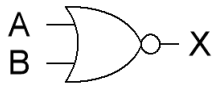
$$X = \overline{A + B}$$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Table de vérité



Symbole à forme rectangulaire



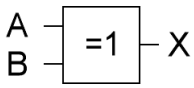
Symbole à forme distincte

6. La porte OU EXCLUSIF (*XOR*)

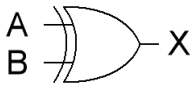
$$X = A \oplus B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité



Symbole à forme rectangulaire



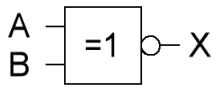
Symbole à forme distincte

7. La porte NON-OU EXCLUSIF (*NXOR*)

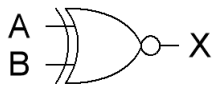
$$X = \overline{A \oplus B}$$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Table de vérité



Symbole à forme rectangulaire

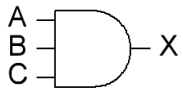


Symbole à forme distincte

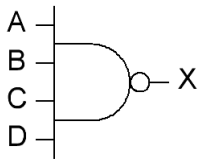
8. Les portes à plus de deux entrées

Les portes logiques peuvent accepter plus de deux entrées.

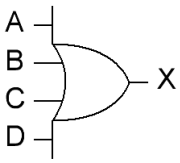
Exemples :



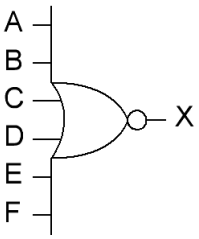
$X = A.B.C$



$X = \overline{A.B.C.D}$



$X = A + B + C + D$

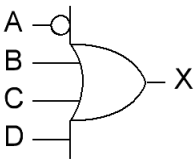


$X = \overline{A + B + C + D + E + F}$

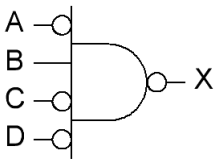
9. Les bulles et les portes

Les petites bulles sur les symboles de portes sont des indicateurs de négation. Ils indiquent une inversion logique sur les entrées ou les sorties. Toute combinaison est possible.

Exemples :



$X = \overline{A} + B + C + D$



$X = \overline{\overline{A}.B.\overline{C}.\overline{D}}$

10. Les logiques positive et négative

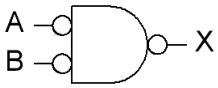
En logique positive, la valeur 0 est associée à l'état inactif (ou état faux) et la valeur 1 à l'état actif (ou état vrai).

Par exemple, nous pouvons dire que :
- La sortie d'une porte OU est vraie (1) quand au moins une de ses entrées est vraie (1) ;
- La sortie d'une porte ET est vraie (1) quand ses deux entrées sont vraies (1).

Ou bien :
- La sortie d'une porte OU est active (1) quand au moins une de ses entrées est active (1) ;
- La sortie d'une porte ET est active (1) quand ses deux entrées sont actives (1).

En logique négative, la valeur 0 est associée à l'état actif (ou état vrai) et la valeur 1 à l'état inactif (ou état faux).

Par exemple, considérons la porte ci-dessous :



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

Ainsi que sa table de vérité :

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Manifestement, nous reconnaissons la table de vérité d'une porte OU. Cela signifie donc qu'il n'y a aucune différence entre une porte OU et la porte ci-dessus. Bien que leurs symboles soient différents, ces deux portes sont physiquement les mêmes :



Porte OU Porte ET négative

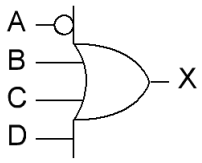
Le premier symbole est une porte OU (positive), car en logique positive, sa sortie est vraie (1) quand au moins une de ses entrées est vraie (1).

Le second symbole est une porte ET négative, car en logique négative, sa sortie est vraie (0) quand toutes ses entrées sont vraies (0).

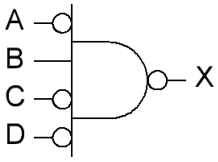
En fait, il est courant de mélanger les notations positive et négative au sein d'un même circuit :

- Quand les entrées ou les sorties n'ont pas de bulles, elles sont actives à l'état haut. C'est-à-dire que la valeur 0 représente un état inactif (faux) et la valeur 1 un état actif (vrai).
- Quand les entrées ou les sorties ont des bulles, elles sont actives à l'état bas. C'est-à-dire que la valeur 0 représente un état actif (vrai) et la valeur 1 un état inactif (faux).

Par exemple :



X est vrai (1) quand *A* est vrai (0) ou *B* est vrai (1) ou *C* est vrai (1) ou *D* est vrai (1).



X est vrai (0) quand *A* est vrai (0) et *B* est vrai (1) et *C* est vrai (0) et *D* est vrai (0).

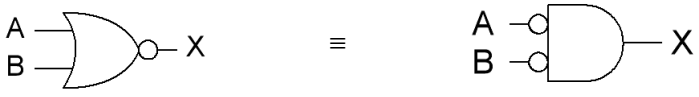
De la même façon, nous pouvons obtenir les équivalents négatifs des autres portes :



Porte NON — Porte NON négative



Porte ET — Porte OU négative



Porte NON-OU — Porte NON-ET négative

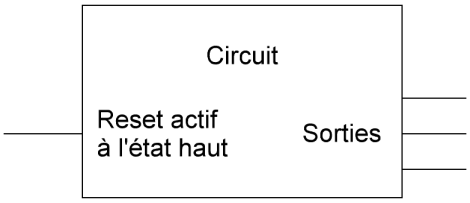


Porte NON-ET — Porte NON-OU négative

L'existence de deux logiques (positive et négative) fait apparaître une difficulté : comment savoir quel symbole de porte utiliser ? Quand doit-on utiliser les symboles de portes positives ? Quand doit-on utiliser les symboles de portes négatives ?

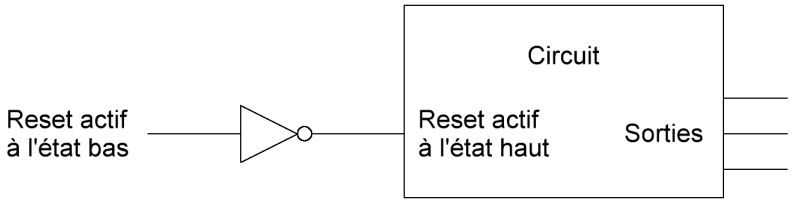
En fait, le principe fondamental reste simple : **quand les entrées ou les sorties sont actives à l'état haut, aucune bulle ne doit être représentée. On placera des bulles dans le cas contraire.**

Prenons comme exemple le circuit suivant :



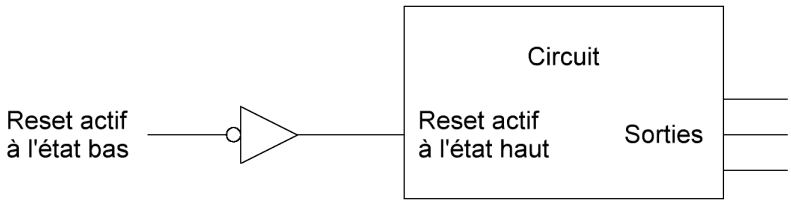
Ce circuit possède une entrée *reset* active à l'état haut et trois sorties :
- Quand l'entrée *reset* est à 0, elle est inactive ;
- Quand l'entrée *reset* est à 1, elle est active et toutes les sorties sont forcées à 0.

Maintenant, supposons que pour une raison quelconque, nous souhaitions que l'entrée *reset* soit active à l'état bas. Pour cela, une simple porte NON est suffisante :



- Quand l'entrée *reset* est à 0, elle est active et toutes les sorties sont forcées à 0 ;
- Quand l'entrée *reset* est à 1, elle est inactive.

Rien n'est théoriquement faux dans le circuit ci-dessus, mais quelque chose n'est pas clair. Un simple coup d'œil à la porte NON n'est pas suffisant pour comprendre immédiatement que l'entrée est active à l'état bas. Pour clarifier les choses, il est préférable d'utiliser le symbole négatif d'une porte NON.

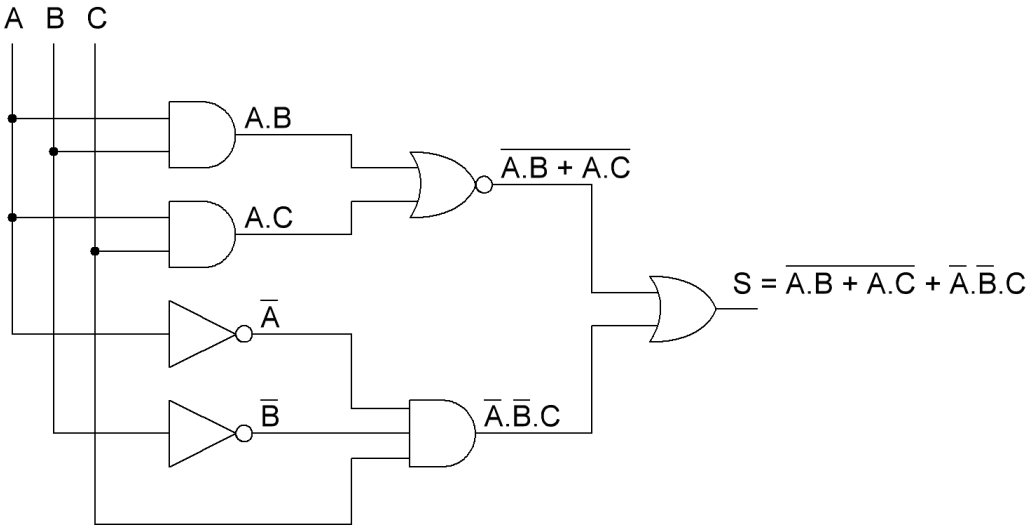


Avec ce symbole, un simple coup d'œil à la porte non est maintenant suffisant.

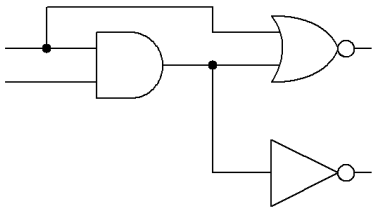
11. Associations de portes logiques

Tout type de circuit combinatoire peut être obtenu en connectant les portes logiques entre elles.

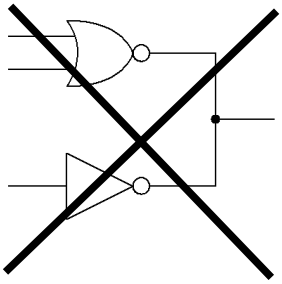
Par exemple :



Toute entrée ou sortie peut se connecter à n'importe quelle entrée.



Mais une sortie ne peut pas être connectée à une autre sortie.



III. L'algèbre booléenne

1. Introduction

L'algèbre booléenne (ou algèbre de Boole) manipule des expressions constituées de variable logique. Elle nous permettra de décrire et de simplifier les circuits combinatoires.

Les trois opérateurs fondamentaux de l'algèbre booléenne sont le NON, le ET et le OU.

2. Principaux axiomes et théorèmes

Priorité	$A + (B.C) = A + B.C$
NON	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A}.A = 0$
ET	$A.(B.C) = (A.B).C = A.B.C$ $A.B = B.A$ $A.A = A$ $A.1 = A$ $A.0 = 0$
OU	$A + (B + C) = (A + B) + C = A + B + C$ $A + B = B + A$ $A + A = A$ $A + 0 = A$ $A + 1 = 1$
OU EXCLUSIF	$A \oplus B = \overline{A}.B + A.\overline{B} = \overline{A} \oplus \overline{B}$ $\overline{A \oplus B} = A.B + \overline{A}.\overline{B} = \overline{A} \oplus B = A \oplus \overline{B}$ $A \oplus (B \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$ $A \oplus B = B \oplus A$ $A \oplus 0 = A$ $A \oplus 1 = \overline{A}$ $A \oplus A = 0$ $A \oplus \overline{A} = 1$
Distributivité	$A.(B + C) = A.B + A.C$ $A + B.C = (A + B).(A + C)$
Théorèmes de De Morgan	$\overline{A.B} = \overline{A} + \overline{B}$ $\overline{A + B} = \overline{A}.\overline{B}$
Autres théorèmes	Théorème 1 : $A + A.B = A$ Théorème 2 : $A + \overline{A}.B = A + B$

Démonstration	
Théorème 1	**Théorème 2**
A + A.B = A.1 + A.B = A.(1 + B) = A.1 = A	A + $\overline{A}$.B = (A + $\overline{A}$).(A + B) → *distributivité* = 1.(A + B) = A + B

3. Principe de dualité

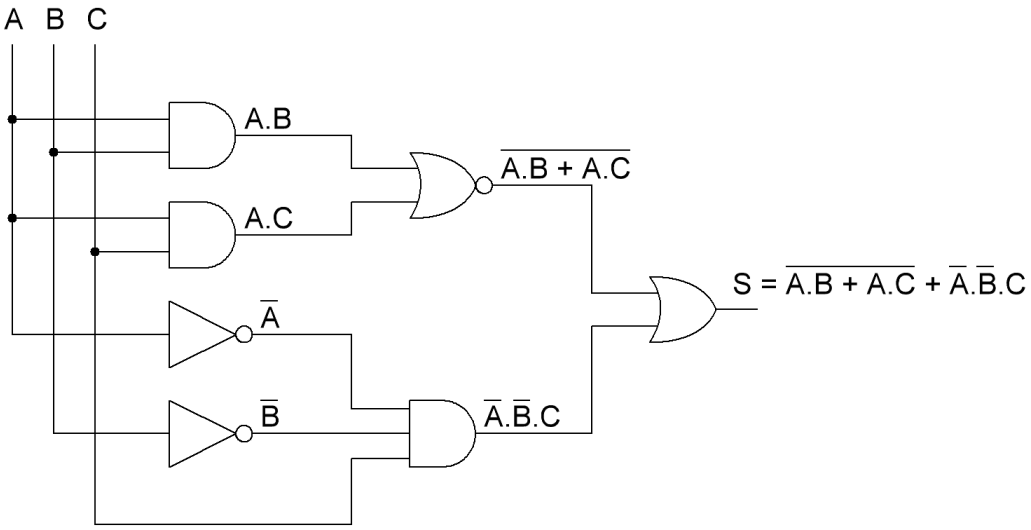
Les égalités booléennes restent vraies si l'on inverse les 0 et les 1 ainsi que les ET et les OU.

Exemples		
1 + 0 = 1	↔	0.1 = 0
X + 1 = 1	↔	X.0 = 0
X.$\overline{X}$ = 0	↔	X + $\overline{X}$ = 1
X + $\overline{X}$.Y = X + Y	↔	X.($\overline{X}$ + Y) = X.Y

4. Simplification algébrique

Les axiomes et les théorèmes de l'algèbre de Boole servent principalement à simplifier les expressions logiques. Ceci permet de réduire le nombre de portes logiques des circuits combinatoires.

Par exemple, essayons de simplifier le circuit suivant :



$S = \overline{A.B + A.C} + \overline{A}.\overline{B}.C$ → *Théorème de De Morgan*
$S = \overline{A.B}.\overline{A.C} + \overline{A}.\overline{B}.C$ → *Théorème de De Morgan*
$S = (\overline{A} + \overline{B}).(\overline{A} + \overline{C}) + \overline{A}.\overline{B}.C$
$S = \overline{A}.\overline{A} + \overline{A}.\overline{C} + \overline{A}.\overline{B} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C$
$S = \overline{A} + \overline{A}.\overline{C} + \overline{A}.\overline{B} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C$ → *Théorème 1 : $\overline{A} + \overline{A}.\overline{C} = \overline{A}$*
$S = \overline{A} + \overline{A}.\overline{B} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C$ → *Théorème 1 : $\overline{A} + \overline{A}.\overline{B} = \overline{A}$*
$S = \overline{A} + \overline{B}.\overline{C} + \overline{A}.\overline{B}.C$ → *Théorème 1 : $\overline{A} + \overline{A}.\overline{B}.C = \overline{A}$*
$S = \overline{A} + \overline{B}.\overline{C}$

Il y a plusieurs façons de symboliser ce circuit. En voici deux exemples :



Remarque : $\overline{A} + \overline{B}.\overline{C} = \overline{A} + \overline{B + C} = \overline{A.(B + C)}$

Autres exemples de simplifications algébriques :

$X1 = (A + B + C.D).(\overline{A} + B).(\overline{B} + C)$
$X1 = (A.\overline{A} + A.B + \overline{A}.B + B.B + \overline{A}.C.D + B.C.D).(\overline{B} + C)$
$X1 = (A.B + \overline{A}.B + B + \overline{A}.C.D + B.C.D).(\overline{B} + C)$ → *Théorème 1 : $B + A.B + \overline{A}.B + B.C.D = B$*
$X1 = (B + \overline{A}.C.D).(\overline{B} + C)$
$X1 = B.\overline{B} + B.C + \overline{A}.\overline{B}.C.D + \overline{A}.C.C.D$
$X1 = B.C + \overline{A}.\overline{B}.C.D + \overline{A}.C.D$ → *Théorème 1 : $\overline{\boldsymbol{A}}.\overline{B}.\boldsymbol{C.D} + \overline{\boldsymbol{A}}.\boldsymbol{C.D} = \overline{A}.C.D$*
$X1 = B.C + \overline{A}.C.D$

$X2 = \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$
$X2 = \overline{A}.B.(\overline{C} + C) + A.\overline{B}.C + A.B.(\overline{C} + C)$
$X2 = \overline{A}.B + A.\overline{B}.C + A.B$
$X2 = B.(\overline{A} + A) + A.\overline{B}.C$
$X2 = B + A.\overline{B}.C$ → *Théorème 2 : $\boldsymbol{B} + A.\overline{\boldsymbol{B}}.C = B + A.C$*
$X2 = B + A.C$

$X3 = A.B.\overline{D} + \overline{B}.C.D + A.B.\overline{C} + B.C.D + \overline{A}.C.D$
$X3 = A.B.\overline{D} + C.D.(\overline{B} + B + \overline{A}) + A.B.\overline{C}$
$X3 = A.B.\overline{D} + C.D + A.B.\overline{C}$
$X3 = A.B.(\overline{C} + \overline{D}) + C.D$ → *Théorème de De Morgan*
$X3 = A.B.\overline{C.D} + C.D$ → *Théorème 2 : $A.B.\overline{\boldsymbol{C.D}} + \boldsymbol{C.D} = A.B + C.D$*
$X3 = A.B + C.D$

5. Les formes canoniques d'expressions booléennes

5.1. Somme de produits et produit de sommes

En algèbre booléenne, le résultat d'un OU peut être appelé une somme et le résultat d'un ET un produit.

Par exemple, les expressions suivantes sont des sommes de produits :
- $A.B.C + B.C.D + A.\overline{B}.C.\overline{D}.E + A.B.C.\overline{D}.\overline{E}$
- $A + A.B + B.\overline{A} + B + \overline{A}.C.D + B.C.D$
- $\overline{B} + A.C + \overline{A}.B + B.\overline{C} + \overline{A}.\overline{B}.C$

Et celles-ci des produits de sommes :
- $(A + \overline{B}).(B + C + D)$
- $(A + \overline{B} + C).(\overline{A} + D + \overline{E}).(C + D + \overline{E}).(A + \overline{B} + C + \overline{D} + E)$
- $A.(B + C + \overline{D}).(A + \overline{C}).(A + D)$

5.2. Somme canonique de produits (première forme canonique)

5.2.1. Définition

Une expression booléenne peut s'exprimer de plusieurs façons différentes, mais elle ne possède qu'une seule somme de *mintermes*, appelée *somme canonique de produits* ou *première forme canonique*.

Un *minterme* est un produit constitué de toutes les variables d'une expression ; chaque variable peut être complémentée ou non.

Prenons comme exemple l'expression suivante :
$A.B.C + B.C.D + A.\overline{B}.C.\overline{D}.E + A.B.C.\overline{D}.\overline{E}$

Elle est constituée de cinq variables : *A*, *B*, *C*, *D* et *E*. Il s'agit également d'une somme de produits.
- Le premier produit *A.B.C* n'est pas un minterme, car *D* et *E* n'apparaissent pas.
- Le deuxième produit *B.C.D* n'est pas un minterme, car *A* et *E* n'apparaissent pas.
- Le troisième produit est un minterme, car il contient toutes les variables de l'expression.
- Le quatrième produit est un minterme, car il contient toutes les variables de l'expression.

Considérons maintenant l'expression suivante :
$A.B.C.D + \overline{A}.B.\overline{C}.D + \overline{A}.B.C.\overline{D} + \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.\overline{B}.\overline{C}.\overline{D}$

Elle est constituée de quatre variables : *A*, *B*, *C* et *D*. Il s'agit d'une somme de produits et tous les produits contiennent les quatre variables. Cette expression est donc une somme de mintermes. Nous appellerons cette forme de l'expression : **somme canonique de produits** ou **première forme canonique**.

5.2.2. Passer d'une somme de produits à une somme canonique de produits

Une somme canonique de produits peut s'obtenir à partir de n'importe quelle somme de produits.

Par exemple, déterminons la somme canonique de l'expression suivante :
$X = \overline{A}.B.C + A.\overline{B} + A.B.\overline{C}.\overline{D}$

Pour ce faire, nous devons convertir chaque terme en minterme.

Cette expression est composée de quatre variables : *A*, *B*, *C* et *D*.

Le premier terme ne contient pas la variable *D* (complémentée ou non), mais nous pouvons facilement l'ajouter :
$X = \overline{A}.B.C.\mathbf{(D + \overline{D})} + A.\overline{B} + A.B.\overline{C}.\overline{D}$
$X = \overline{A}.B.C.\mathbf{D} + \overline{A}.B.C.\mathbf{\overline{D}} + A.\overline{B} + A.B.\overline{C}.\overline{D}$

Le terme suivant ne contient pas les variables *C* et *D*. Il suffit également de les ajouter.
$X = \overline{A}.B.C.D + \overline{A}.B.C.\overline{D} + A.\overline{B}.\mathbf{(C + \overline{C})}.\mathbf{(D + \overline{D})} + A.B.\overline{C}.\overline{D}$
$X = \overline{A}.B.C.D + \overline{A}.B.C.\overline{D} + (A.\overline{B}.\mathbf{C} + A.\overline{B}.\mathbf{\overline{C}}).(\mathbf{D} + \mathbf{\overline{D}}) + A.B.\overline{C}.\overline{D}$
$X = \overline{A}.B.C.D + \overline{A}.B.C.\overline{D} + A.\overline{B}.\mathbf{C.D} + A.\overline{B}.\mathbf{C.\overline{D}} + A.\overline{B}.\mathbf{\overline{C}.D} + A.\overline{B}.\mathbf{\overline{C}.\overline{D}} + A.B.\overline{C}.\overline{D}$

Le dernier terme est déjà un minterme.

Par conséquent, la première forme canonique de *X* est :
$X = \overline{A}.B.C.D + \overline{A}.B.C.\overline{D} + A.\overline{B}.C.D + A.\overline{B}.C.\overline{D} + A.\overline{B}.\overline{C}.D + A.\overline{B}.\overline{C}.\overline{D} + A.B.\overline{C}.\overline{D}$

5.2.3. Relation entre une somme canonique de produits et une table de vérité

Une somme canonique de produits se détermine facilement à partir d'une table de vérité (et inversement).

Prenons comme exemple la table de vérité suivante :

A	B	C	X	
0	0	0	0	
0	0	1	1	← $\overline{A}.\overline{B}.C$
0	1	0	0	
0	1	1	1	← $\overline{A}.B.C$
1	0	0	0	
1	0	1	0	
1	1	0	1	← $A.B.\overline{C}$
1	1	1	1	← $A.B.C$

À partir de la table de vérité, chaque combinaison où **X vaut 1** peut être convertie en minterme. Il faut simplement remplacer les 1 par les variables correspondantes et les 0 par les variables complémentées.

Voici donc la somme canonique de produits pour la variable *X* :
$X = \overline{A}.\overline{B}.C + \overline{A}.B.C + A.B.\overline{C} + A.B.C$

5.3. Produit canonique de sommes (seconde forme canonique)

5.3.1. Définition

Une expression booléenne peut s'exprimer de plusieurs façons différentes, mais elle ne possède qu'un seul produit de *maxtermes*, appelée *produit canonique de sommes* ou *seconde forme canonique*.

Un *maxterme* est une somme constituée de toutes les variables d'une expression ; chaque variable peut être complémentée ou non.

Prenons comme exemple l'expression suivante :
$(A + B + C).(B + C + D).(A + \overline{B} + C + \overline{D} + E).(A + B + C + \overline{D} + \overline{E})$

Elle est constituée de cinq variables : *A*, *B*, *C*, *D* et *E*. Il s'agit également d'un produit de sommes.
- La première somme $A + B + C$ n'est pas un maxterme, car *D* et *E* n'apparaissent pas.
- La deuxième somme $B + C + D$ n'est pas un maxterme, car *A* et *E* n'apparaissent pas.
- La troisième somme est un maxterme, car elle contient toutes les variables de l'expression.
- La quatrième somme est un maxterme, car elle contient toutes les variables de l'expression.

Considérons maintenant l'expression suivante :
$(A + B + C + D).(\overline{A} + B + \overline{C} + D).(\overline{A} + B + C + \overline{D}).(\overline{A} + B + \overline{C} + \overline{D}).(\overline{A} + \overline{B} + \overline{C} + \overline{D})$

Elle est constituée de quatre variables : *A*, *B*, *C* et *D*. Il s'agit d'un produit de sommes et toutes les sommes contiennent les quatre variables. Cette expression est donc un produit de maxtermes. Nous appellerons cette forme de l'expression : **produit canonique de sommes** ou **seconde forme canonique**.

5.3.2. Passer d'un produit de sommes à un produit canonique de sommes

Un produit canonique de sommes peut s'obtenir à partir de n'importe quel produit de sommes.

Par exemple, déterminons le produit canonique de l'expression suivante :
$X = (\overline{A} + C).B$

Pour ce faire, nous devons convertir chaque terme en maxterme.

Cette expression est composée de trois variables : *A*, *B* et *C*.

Le premier terme ne contient pas la variable *B* (complémentée ou non), mais nous pouvons facilement l'ajouter :
$X = (\overline{A} + \mathbf{B.\overline{B}} + C).B$

Appliquons ensuite la distributivité :
$X = (\overline{A} + \mathbf{B} + C).(\overline{A} + \mathbf{\overline{B}} + C).B$

Le terme suivant ne contient pas les variables *A* et *C*. Il suffit également de les ajouter.
$X = (\overline{A} + B + C).(\overline{A} + \overline{B} + C).(\mathbf{A.\overline{A}} + B + \mathbf{C.\overline{C}})$
$X = (\overline{A} + B + C).(\overline{A} + \overline{B} + C).(\mathbf{A} + B + \mathbf{C.\overline{C}}).(\mathbf{\overline{A}} + B + \mathbf{C.\overline{C}})$
$X = (\overline{A} + B + C).(\overline{A} + \overline{B} + C).(\mathbf{A} + B + \mathbf{C}).(\mathbf{A} + B + \mathbf{\overline{C}}).(\mathbf{\overline{A}} + B + \mathbf{C}).(\mathbf{\overline{A}} + B + \mathbf{\overline{C}})$

Par conséquent, la seconde forme canonique de *X* est :
$X = (\overline{A} + B + C).(\overline{A} + \overline{B} + C).(A + B + C).(A + B + \overline{C}).(\overline{A} + B + \overline{C})$

5.3.3. Relation entre un produit canonique de sommes et une table de vérité

Un produit canonique de sommes se détermine facilement à partir d'une table de vérité (et inversement).

Prenons comme exemple la table de vérité suivante :

A	**B**	**C**	**X**	
0	0	0	0	← $A + B + C$
0	0	1	1	
0	1	0	0	← $A + \overline{B} + C$
0	1	1	1	
1	0	0	0	← $\overline{A} + B + C$
1	0	1	0	← $\overline{A} + B + \overline{C}$
1	1	0	1	
1	1	1	1	

À partir de la table de vérité, chaque combinaison où **X vaut 0** peut être convertie en maxterme. Il faut simplement remplacer les 0 par les variables correspondantes et les 1 par les variables complémentées.

Voici donc le produit canonique de sommes pour la variable *X* :
$X = (A + B + C).(A + \overline{B} + C).(\overline{A} + B + C).(\overline{A} + B + \overline{C})$

5.4. Passer d'une forme canonique à une autre

Pour passer d'une forme canonique à une autre, il faut recenser les combinaisons binaires n'appartenant par à une forme. Ceci permettra d'en déduire rapidement l'autre forme.

Par exemple, convertissons cette première forme canonique en une seconde forme canonique :
$\overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C}$

Les combinaisons binaires associées sont :
001, 010, 011, 101, 110

Les combinaisons manquantes sont donc :
000, 100, 111

Par conséquent, la seconde forme canonique est :
$(A + B + C).(\overline{A} + B + C).(\overline{A} + \overline{B} + \overline{C})$

6. Les diagrammes de Karnaugh

6.1. Définition

Un digramme de Karnaugh (ou tableau de Karnaugh) est une méthode graphique de simplification d'expressions booléennes.

Un diagramme de Karnaugh est en quelque sorte une façon différente de représenter une table de vérité puisqu'il exprime les sorties en fonctions des entrées.

Par exemple, observons la table de vérité ci-dessous et son tableau de Karnaugh associé :

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

BC

A \ **X**	**00**	**01**	**11**	**10**
0	1	1	1	0
1	0	1	1	1

← Code Gray

↑
Code Gray

En supposant que *n* est le nombre d'entrées de la table de vérité, le nombre de cases d'un tableau de Karnaugh est de 2^n. La table de vérité ci-dessus à trois entrées (*A*, *B*, *C*), son tableau de Karnaugh associé a donc huit cases.

Les cases se répartissent sur deux lignes et quatre colonnes. Toutefois, le contraire est aussi possible (quatre lignes et deux colonnes). **Les lignes et les colonnes sont numérotées en Code Gray.**

Chaque case contient la valeur de la sortie (*X*) en fonction de la valeur des entrées (*A*, *B*, *C*) :

- Pour toutes les cases de la première ligne, *A* vaut 0.
- Pour toutes les cases de la seconde ligne, *A* vaut 1.
- Pour toutes les cases de la première colonne, *B* vaut 0 et *C* vaut 0.
- Pour toutes les cases de la deuxième colonne, *B* vaut 0 et *C* vaut 1.
- Pour toutes les cases de la troisième colonne, *B* vaut 1 et *C* vaut 1.
- Pour toutes les cases de la quatrième colonne, *B* vaut 1 et *C* vaut 0.

Par exemple :

- La case en haut à gauche, donne la valeur de *X* quand *A = B = C = 0*.
- La case en bas à droite, donne la valeur de *X* quand *A = B = 1* et *C = 0*.

Supposons maintenant que les trois entrées représentent un entier non signé *N*. Les cases du diagramme de Karnaugh peuvent être numérotées en représentation décimale de la façon suivante :

N	**A**	**B**	**C**	**X**
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

		BC			
	N	**00**	**01**	**11**	**10**
A	**0**	*0*	*1*	*3*	*2*
	1	*4*	*5*	*7*	*6*

Comme cela a été dit précédemment, il est possible de représenter le tableau de Karnaugh avec quatre lignes et deux colonnes :

		C	
	N	**0**	**1**
AB	**00**	*0*	*1*
	01	*2*	*3*
	11	*6*	*7*
	10	*4*	*5*

		C	
	X	**0**	**1**
AB	**00**	1	1
	01	0	1
	11	1	1
	10	0	1

Dans un tableau de Karnaugh, il est préférable d'inscrire les variables d'entrée dans un ordre identique à celui de la table de vérité ; l'ordre de lecture de gauche à droite doit être conservé (ici *A*, *B*, *C*). De cette façon, la correspondance entre un tableau de Karnaugh et sa table de vérité sera plus évidente. Par exemple, le tableau ci-dessous est équivalent aux deux autres ci-dessus, mais il est plus long à remplir :

AC \ B	**0**	**1**
00	1	0
01	1	1
11	1	1
10	0	1

Les tableaux de Karnaugh permettent de simplifier les expressions à deux, trois, quatre et cinq variables. Dans ce chapitre, nous nous limiterons à quatre variables maximum. Voici quelques exemples de diagrammes. Chaque case contient sa représentation décimale associée.

A \ B	**0**	**1**
0	*0*	*1*
1	*2*	*3*

Tableau de Karnaugh à deux variables

AB \ C	**0**	**1**
00	*0*	*1*
01	*2*	*3*
11	*6*	*7*
10	*4*	*5*

Tableau de Karnaugh à trois variables

A \ BC	**00**	**01**	**11**	**10**
0	*0*	*1*	*3*	*2*
1	*4*	*5*	*7*	*6*

Tableau de Karnaugh à trois variables

AB \ CD	**00**	**01**	**11**	**10**
00	*0*	*1*	*3*	*2*
01	*4*	*5*	*7*	*6*
11	*12*	*13*	*15*	*14*
10	*8*	*9*	*11*	*10*

Tableau de Karnaugh à quatre variables

6.2. Simplification d'expressions logiques

Trouvons la plus simple expression de *X* à partir de la table de vérité suivante :

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Comme démontré précédemment, voici l'un de ses tableaux Karnaugh :

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

Maintenant, nous devons encercler les groupes de 1 adjacents en respectant les règles suivantes :

- Le nombre de cases d'un groupe doit être une puissance de deux. Par exemple, nous pouvons regrouper, un, deux, quatre, huit 1, etc.
- Tous les 1 (et seulement les 1) doivent être encerclés.
- Le nombre de cercles doit être minimum.
- La taille d'un cercle doit être maximum.

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

Le nombre de cercles correspond au nombre de termes de l'expression booléenne (un terme est un produit). Dans notre exemple, nous avons trois cercles, l'expression contiendra donc trois termes. Autrement dit, l'expression la plus simplifiée de *X* sera une somme de trois produits.

Maintenant, chaque groupe doit être converti en produit de la façon suivante :

- Si une variable varie dans un groupe, cette variable sera ignorée et n'apparaîtra pas dans le produit ;
- Si une variable est constante dans un groupe et vaut 0, la variable apparaîtra complémentée ;
- Si une variable est constante dans un groupe et vaut 1, la variable apparaîtra non complémentée.

Par exemple, commençons par convertir le cercle le plus à gauche :

		BC			
	X	**00**	**01**	**11**	**10**
A	**0**	1	1	1	0
	1	0	1	1	1

- La variable *A* vaut toujours 0. Par conséquent, $\overline{A}$ apparaîtra dans le produit.
- La variable *B* vaut toujours 0. Par conséquent, $\overline{B}$ apparaîtra dans le produit.
- La variable *C* varie. Par conséquent, elle n'apparaîtra pas dans le produit.

Le produit associé à ce cercle est donc : $\mathbf{\overline{A}.\overline{B}}$

Convertissons maintenant le cercle le plus à droite :

		BC			
	X	**00**	**01**	**11**	**10**
A	**0**	1	1	1	0
	1	0	1	1	1

- La variable *A* vaut toujours 1. Par conséquent, *A* apparaîtra dans le produit.
- La variable *B* vaut toujours 1. Par conséquent, *B* apparaîtra dans le produit.
- La variable *C* varie. Par conséquent, elle n'apparaîtra pas dans le produit.

Le produit associé à ce cercle est donc : **A.B**

Pour finir, considérons le dernier cercle :

		BC			
	X	**00**	**01**	**11**	**10**
A	**0**	1	1	1	0
	1	0	1	1	1

- La variable *A* varie. Par conséquent, elle n'apparaîtra pas dans le produit.
- La variable *B* varie. Par conséquent, elle n'apparaîtra pas dans le produit.
- La variable *C* vaut toujours 1. Par conséquent, *C* apparaîtra dans le produit.

Le produit associé à ce cercle est donc : *C*.

Pour conclure, la plus simple expression de *X* est :
$X = \overline{A}.\overline{B} + A.B + C$

La plus simple expression obtenue à l'aide d'un diagramme de Karnaugh contient uniquement les opérateurs élémentaires. Une fois cette expression obtenue, il est donc possible d'obtenir un degré de simplification supplémentaire à l'aide de l'opérateur OU EXCLUSIF.

Nous pouvons donc écrire que :
$X = \overline{A \oplus B} + C$

Remarque :
Il est important de signaler qu'un tableau de Karnaugh doit être considéré comme enveloppé dans un tore. C'est-à-dire que les cases de la colonne de gauche sont considérées comme étant adjacentes aux cases de la colonne de droite et que les cases de la ligne du haut sont considérées comme étant adjacentes aux cases de la ligne du bas. Il est donc possible de regrouper dans une même bulle certaines cases de gauche avec certaines cases de droite et certaines cases du haut avec certaines cases du bas.

Exemple :

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

$X = A.B.\overline{D} + \overline{B}.\overline{C}.\overline{D}$

En fait, deux cases sont adjacentes quand seulement une seule variable change entre les deux cases.

IV. Résolution de problèmes et synthèse de circuits combinatoires

Afin d'illustrer les principales techniques de résolution de problèmes, nous allons étudier un cas concret.

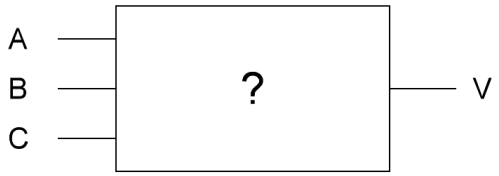
Remplissage de soutes d'un pétrolier

Sur un pétrolier, la cale comprend trois soutes à pétrole (*a*, *b*, *c*). Elles sont remplies de façon indépendante les unes des autres. Quand le remplissage est terminé, un voyant doit s'allumer si l'assiette du bateau est correcte ; c'est-à-dire si le pétrole est correctement réparti dans les soutes. Il y a un capteur par soute (*A*, *B*, *C*) qui indique si oui ou non la soute est pleine.

L'assiette est correcte si les trois soutes sont pleines, ou si les trois soutes sont vides, ou si la soute *b* est pleine et les autres vides, ou si la soute *b* est vide et les autres pleines.

Entrées du système	Sortie du système
A = 1, si la soute *a* est pleine	*V* = 1, si l'assiette est correcte
B = 1, si la soute *b* est pleine	
C = 1, si la soute *c* est pleine	

Nous devons déterminer la plus simple expression de *V* afin de concevoir le circuit combinatoire qui satisfait le système :



La première chose à faire est de déterminer la table de vérité de *V* :

A	B	C	V	
0	0	0	**1**	← Les trois soutes sont vides
0	0	1	0	
0	1	0	**1**	← La soute *b* est pleine et les autres sont vides
0	1	1	0	
1	0	0	0	
1	0	1	**1**	← La soute *b* est vide et les autres sont pleines
1	1	0	0	
1	1	1	**1**	← Les trois soutes sont pleines

Représentons maintenant le diagramme de Karnaugh de la sortie *V* :

		BC			
	V	**00**	**01**	**11**	**10**
A	**0**	1	0	0	1
	1	0	1	1	0

À partir de ce diagramme, nous pouvons déterminer la plus simple expression de *V* (uniquement à l'aide des opérateurs élémentaires).

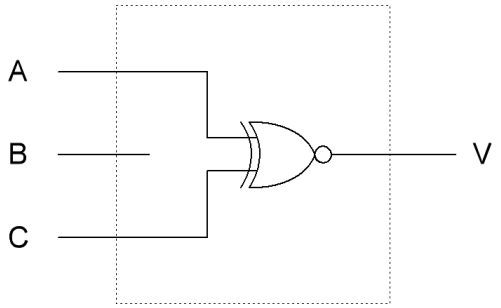
		BC			
	V	**00**	**01**	**11**	**10**
A	**0**	1	0	0	1
	1	0	1	1	0

$$\mathbf{V = \overline{A}.\overline{C} + A.C}$$

Cette expression peut être encore simplifiée à l'aide de l'opérateur OU EXCLUSIF.

$$\mathbf{V = \overline{A \oplus C}}$$

Pour terminer, voici le circuit combinatoire du système :



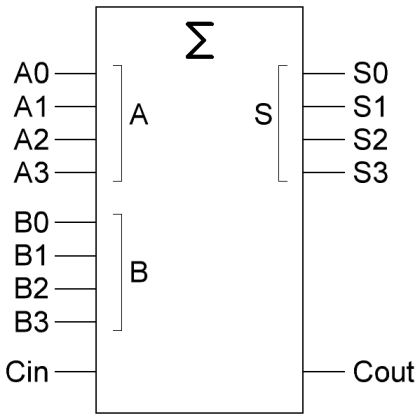
V. Les principaux circuits combinatoires

1. L'additionneur

Un additionneur permet d'additionner deux nombres binaires. Il possède généralement les entrées-sorties suivantes :

- Deux entrées parallèles sur *n* bits contenant les nombres binaires à additionner ;
- Une retenue d'entrée ;
- Une sortie parallèle sur *n* bits contenant le résultat de l'addition ;
- Une retenue de sortie.

Exemple d'un additionneur sur 4 bits :



Par exemple, si *A* = **1010**, *B* = **1100** et *Cin* = **1**:

```
    1 ← Cin
 1010 ← A
+1100 ← B
10111 ← S
↑
Cout
```

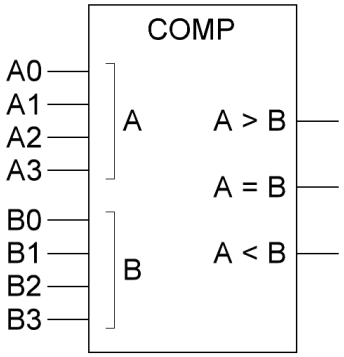
Remarque :
A, *B* et *S* sont des nombres binaires codés sur 4 bits. *A0*, *B0* et *S0* sont respectivement leurs LSB.

2. Le comparateur

Un comparateur effectue la comparaison de deux nombres binaires afin de déterminer la relation existant entre ces deux nombres. Il possède généralement les entrées-sorties suivantes :

- Deux entrées parallèles sur *n* bits contenant les nombres binaires à comparer ;
- Une sortie indiquant si le premier nombre est strictement supérieur au second ;
- Une sortie indiquant si les deux nombres sont égaux ;
- Une sortie indiquant si le premier nombre est strictement inférieur au second.

Exemple d'un comparateur sur 4 bits :



Exemples :

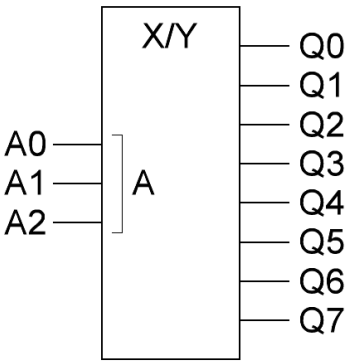
- Si ***A* = 0110** et *B* = **0011**, alors *'A > B'* = 1, *'A = B'* = 0 et *'A < B'* = 0.
- Si ***A* = 0110** et *B* = **0110**, alors *'A > B'* = 0, *'A = B'* = 1 et *'A < B'* = 0.
- Si ***A* = 0110** et *B* = **1110**, alors *'A > B'* = 0, *'A = B'* = 0 et *'A < B'* = 1.

3. Le décodeur

Le décodeur permet de détecter une combinaison spécifique de bits présente sur ses entrées et de l'indiquer par un niveau spécifique de sortie. Il possède généralement les entrées-sorties suivantes :

- Une entrée parallèle sur *n* bits contenant le nombre à décoder ;
- Une sortie parallèle sur **2^n** bits indiquant quel nombre est présent sur l'entrée.

Exemple d'un décodeur *1 parmi 8* (3 lignes d'entrée, 8 lignes de sortie) :



A	**A2**	**A1**	**A0**	**Q0**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**
0	0	0	0	**1**	0	0	0	0	0	0	0
1	0	0	1	0	**1**	0	0	0	0	0	0
2	0	1	0	0	0	**1**	0	0	0	0	0
3	0	1	1	0	0	0	**1**	0	0	0	0
4	1	0	0	0	0	0	0	**1**	0	0	0
5	1	0	1	0	0	0	0	0	**1**	0	0
6	1	1	0	0	0	0	0	0	0	**1**	0
7	1	1	1	0	0	0	0	0	0	0	**1**

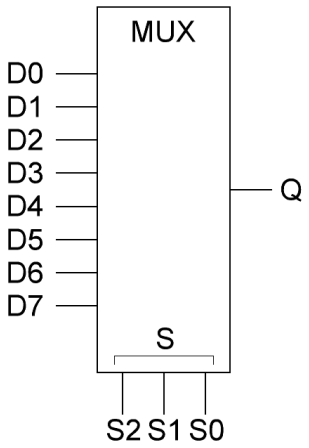
4. Le multiplexeur

Un multiplexeur permet d'envoyer les informations numériques de plusieurs sources vers une seule ligne de sortie. Il possède généralement les entrées-sorties suivantes :

- Une entrée parallèle de sélection sur *n* bits ;
- Une entrée parallèle de donnée sur 2^n bits ;
- Une ligne de sortie.

Les entrées de sélection permettent de sélectionner le bit de donnée qui sera copié sur la sortie.

Exemple d'un multiplexeur *8 vers 1* (3 lignes de sélection, 8 lignes de donnée, 1 ligne de sortie) :



S	S2	S1	S0	Q
0	0	0	0	D0
1	0	0	1	D1
2	0	1	0	D2
3	0	1	1	D3
4	1	0	0	D4
5	1	0	1	D5
6	1	1	0	D6
7	1	1	1	D7

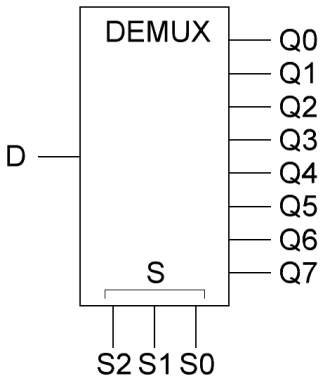
Les multiplexeurs sont souvent utilisés pour transformer une liaison parallèle en une liaison série. À l'intérieur d'un ordinateur, les données sont véhiculées par paquets de bits (par exemple 8, 16, 32 ou encore 64 bits). Lorsque l'on souhaite échanger des données avec des périphériques externes (imprimantes, appareils photo, etc.), il ne serait pas très pratique d'utiliser un câble ou un cordon comportant 64 fils. Il est donc préférable d'utiliser une liaison série qui permet de faire transiter les données sur un seul fil. Par exemple, la liaison USB (*Universal Serial Bus*) est une liaison série.

5. Le démultiplexeur

Un démultiplexeur effectue la fonction inverse d'un multiplexeur. Il permet de distribuer les informations numériques présentes sur une seule ligne vers un nombre donné de lignes de sortie. Il possède généralement les entrées-sorties suivantes :

- Une ligne d'entrée ;
- Une entrée parallèle de sélection sur *n* bits ;
- Une sortie parallèle sur 2^n bits recopiant la ligne d'entrée sur le bit sélectionné.

Exemple d'un démultiplexeur *1 parmi 8* (3 lignes de sélection, 1 ligne de donnée, 8 lignes de sortie) :



S	S2	S1	S0	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	0	0	**D**	0	0	0	0	0	0	0
1	0	0	1	0	**D**	0	0	0	0	0	0
2	0	1	0	0	0	**D**	0	0	0	0	0
3	0	1	1	0	0	0	**D**	0	0	0	0
4	1	0	0	0	0	0	0	**D**	0	0	0
5	1	0	1	0	0	0	0	0	**D**	0	0
6	1	1	0	0	0	0	0	0	0	**D**	0
7	1	1	1	0	0	0	0	0	0	0	**D**

Remarque :
Si son entrée *D* est toujours à 1, alors un démultiplexeur se comporte comme un décodeur.