Informatique *Cours*
Logique Ipesup

Objectifs

À l'issue de cette leçon, l'étudiant doit être capable de :
- comprendre et manipuler les expressions de l'algèbre de Boole ;
- représenter une fonction booléenne sous différentes formes : SOP, POS, formes canoniques et table de vérité ;
- utiliser les cartes de Karnaugh pour la simplification systématique des fonctions logiques ;
- comprendre le fonctionnement des portes logiques de base (NOT, AND, OR) et leur modélisation logique ;
- analyser le rôle des entrées d'activation et de désactivation dans les circuits logiques ;

L'algèbre utilisée pour représenter symboliquement les fonctions logiques est appelée *algèbre de Boole*. Il s'agit d'une algèbre à deux états inventée par George Boole en 1854.

Ainsi, l'algèbre de Boole est un système de logique mathématique destiné à l'analyse et à la conception des systèmes numériques.

Une variable ou une fonction de variables en algèbre de Boole ne peut prendre que deux valeurs, soit 0, soit 1. Par conséquent, il n'existe ni fractions, ni nombres négatifs, ni racines carrées, ni racines cubiques, ni logarithmes, etc.

1 Opérations logiques

En algèbre de Boole, toutes les fonctions algébriques réalisées sont de nature logique. Elles représentent des opérations logiques.
Les opérations de base sont AND, OR et NOT.
En plus de ces opérations fondamentales, il existe des opérations dérivées telles que NAND, NOR, EX-OR et EX-NOR, également utilisées en algèbre de Boole.

NOT

L'opération NOT en algèbre de Boole est similaire à la complémentation ou à l'inversion en algèbre ordinaire. Elle est indiquée par une barre ($^{-}$) ou par une apostrophe ($'$) au-dessus de la variable.

$$A \xrightarrow{\text{NOT}} \bar{A} \text{ ou } A' \quad \text{(loi de complémentation)}$$
$$\bar{\bar{A}} = A \quad \text{(loi de double complémentation)}$$

AND

L'opération AND en algèbre de Boole est similaire à la multiplication en algèbre ordinaire. Il s'agit d'une opération logique réalisée par une porte AND.

$$
\begin{aligned}
&A \cdot A = A \\
&A \cdot 0 = 0 \quad \text{(loi de l'élément nul)} \\
&A \cdot 1 = A \quad \text{(loi de l'élément neutre)} \\
&A \cdot \bar{A} = 0
\end{aligned}
$$

OR

L'opération OR en algèbre de Boole est similaire à l'addiction en algèbre ordinaire et elle est réalisée par une porte OR.

$$
\begin{aligned}
&A + A = A \\
&A + 0 = A \quad \text{(loi de l'élément nul)} \\
&A + 1 = 1 \quad \text{(loi de l'élément neutre)} \\
&A + \bar{A} = 1
\end{aligned}
$$

NAND

L'opération NAND en algèbre de Boole est obtenue en appliquant une opération NOT à une opération AND, c'est-à-dire que la négation de la porte AND est réalisée par la porte NAND.

NOR

L'opération NOR en algèbre de Boole est obtenue en appliquant une opération NOTà une opération OR, c'est-à-dire que la négation de la porte OR est réalisée par la porte NOR.

EX-OR

Contrairement aux opérations logiques de base, cette opération est utilisée à des fins spécifiques et est représentée par le symbole $\oplus$, où :

$$A \oplus B = A\bar{B} + \bar{A}B$$

1.1 Lois de l'algèbre de Boole

L'algèbre de Boole est régie par un ensemble de règles et de lois bien définies.

- Lois commutatives
  1. $A + B = B + A$
  2. $A \cdot B = B \cdot A$
- Lois associatives
  1. $(A + B) + C = A + (B + C)$
  2. $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- Lois distributives
  1. $(B + C) = AB + AC$
  2. $A + BC = (A + B)(A + C)$
- Lois d'idempotence
  1. $A \cdot A = A$

2. $A + A = A$

— Lois d'absorption
   1. $A + AB = A(1 + B) = A$
   2. $A(A + B) = A$

— Loi d'involution
   Cette loi énonce que pour toute variable A :

$$\bar{\bar{A}} = (A')' = A$$

2 Théorèmes algébriques de Boole

Théorème de De Morgan

Le théorème de De Morgan représente les règles plus importantes de l'algèbre de Boole :

1. $\overline{A \cdot B} = \bar{A} + \bar{B}$
   Le complément du produit de variables est égal à la somme de leurs compléments.
2. $\overline{A + B} = \bar{A} \cdot \bar{B}$
   Le complément de la somme de variables est égal au produit de leurs compléments.

Ces lois peuvent être étendues à n variables :

$$\overline{A_1 \cdot A_2 \cdots A_n} = \bar{A}_1 + \bar{A}_2 + \cdots + \bar{A}_n$$

$$\overline{A_1 + A_2 + \cdots + A_n} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdots \bar{A}_n$$

Théorème de transposition

Le théorème de transposition énonce que :

$$(AB + \bar{A}C) = (A + C)(\bar{A} + B)$$

Théorème du consensus / théorème de redondance

Soit une expression booléenne contenant trois variables A, B, C et de la forme telle que :

— une variable apparaît une fois sous forme complémentée et une fois sous forme non complémentée ;
— les deux autres variables apparaissent chacune deux fois.

Alors, le terme contenant les deux variables non communes (appelé *terme de consensus*) est redondant et peut être supprimé.

Exemple

$$AB + \bar{A}C + BC = AB + AC$$

Le théorème du consensus peut être étendu à un nombre quelconque de variables.

Théorème de dualité

Si une expression booléenne est vraie, alors son *expression duale* est également vraie.

Pour obtenir l'expression duale il faut :

1. remplacer chaque symbole OR par un symbole AND et inversement ;
2. complémenter chaque 0 ou 1 apparaissant dans l'expression ;
3. conserver les littéraux / variables inchangés.

Attention

Pour toute expression logique, si l'on applique deux fois l'opération de dualité, on retrouve l'expression initiale.
Si l'application une seule fois de la dualité donne la même fonction ou expression, celle-ci est appelée *expression auto-duale*.

Théorème du complément

Si une expression booléenne est vraie, alors son expression complémentaire est fausse, et inversement.

Pour obtenir l'expression complémentaire, il faut :

1. remplacer chaque symbole OR par un symbole AND et inversement ;
2. complémenter chaque 0 ou 1 apparaissant dans l'expression ;
3. complémenter chaque littéral / variable individuellement.

3 Représentation des fonctions booléennes

Une fonction de n variables booléennes, notée $f(A_1, A_2, \ldots, A_n)$, est une variable de l'algèbre qui ne peut prendre que deux valeurs possibles : 0 ou 1.

Les différentes méthodes de représentation d'une fonction sont :

— **Forme canonique** : tous les termes contiennent toutes les variables, soit sous forme complémentée, soit sous forme non complémentée.

Exemple

$F(A, B, C) = \bar{A}BC + ABC + A\bar{B}\bar{C}$

— **Forme minimale** : nombre minimal de littéraux.

Exemple

$F(A, B, C) = A + ABC + \bar{A}BC = A$

3.1 Minterms et maxterms

On sait que n variables binaires admettent 2^n combinaisons possibles.

Definition

Un *minterme* est un terme produit contenant toutes les variables, sous forme complémentée ou non complémentée, pour lequel la sortie de la fonction vaut 1.
Dans les minterms, on associe la valeur 1 à chaque variable non complémentée et la valeur 0 à chaque variable complémentée.

Definition

Un *maxterme* est un terme somme contenant toutes les variables, sous forme complémentée ou non complémentée, pour lequel la sortie de la fonction vaut 0.
Dans les maxterms, on associe la valeur 0 à chaque variable non complémentée et la valeur 1 à chaque variable complémentée.

3.2 Forme somme de produits (SOP)

Une expression SOP prend généralement la forme de deux ou plusieurs variables reliées par des opérations AND où chaque terme produit est un minterme.

Cette forme est également appelée *forme normale disjonctive* et elle est très utilisée car elle se prête naturellement à l'élaboration de tables de vérité et de diagrammes temporels.

Les formes SOP sont utilisées pour écrire des expressions logiques dont la sortie vaut 1.

Exemple

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

La notation de l'expression SOP est :

$$f_{SOP}(A,B,C) = \sum_m (3,5,6,7) =$$
$$= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

3.3 Forme produit de sommes (POS)

Une expression POS prend généralement la forme de deux ou plusieurs variables à l'intérieur de parenthèses, reliées par des opérations AND entre plusieurs termes où chaque terme individuel est un maxterme.

Cette forme est également appelée *forme normale conjonctive* et elle est utilisée pour écrire des expressions logiques dont la sortie vaut la logique 0.

Exemple

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

La notation de l'expression POS est :

$$f_{POS}A, B, C) = \prod_{M}(0, 1, 2, 4) =$$
$$= (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)(\bar{A}$$

On en conclut que, à partir de la table de vérité et des équations précédentes :

$$f_{SOP} = \sum_{m}(3, 5, 6, 7) \Rightarrow f_{POS} = \prod_{M}(0, 1, 2, 4)$$

3.4 Forme standard somme de produits

Dans cette forme, la fonction est la somme de plusieurs termes produits, chaque terme contenant toutes les variables de la fonction, soit sous forme complémentée, soit sous forme non complémentée.

Elle est également appelée *forme SOP canonique* ou *forme SOP développée*.

Exemple

La fonction

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

peut être représentée sous forme canonique comme :

$$\begin{aligned} Y &= A + B\bar{C} \\ &= A(B + \bar{B})(C + \bar{C}) + B\bar{C}(A + \bar{A}) \\ &= ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + AB\bar{C} + \bar{A}B\bar{C} \\ &= ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} \end{aligned}$$

3.5 Forme standard produit de sommes

Dans cette forme, la fonction est le produit de plusieurs termes sommes, chaque terme contenant toutes les variables de la fonction, soit sous forme complémentée, soit sous forme non complémentée.

Elle est également appelée *forme POS canonique* ou *forme POS développée*.

Exemple

La fonction

$$Y = (B + \bar{C})(A + \bar{B})$$

La forme canonique de la fonction donnée est alors :

$$\begin{aligned} Y &= (B + C + A\bar{A})(A + \bar{B} + C\bar{C}) \\ &= (B + \bar{C} + A)(B + \bar{C} + \bar{A})(A + \bar{B} + C)(A + \bar{B} + \bar{C}) \end{aligned}$$

3.6 Forme table de vérité

Une table de vérité est une représentation tabulaire de toutes les combinaisons possibles d'une fonction donnée.

Exemple

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

	A	B	C	Y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

3.7 Forme duale

La forme duale est utilisée pour convertir la logique positive en logique négative et inversement.

Attention

Dans un système à logique positive, une tension plus élevée correspond à la logique 1, tandis que dans un système à logique négative, une tension plus élevée correspond à la logique 0.

1. Pour la logique positive :

$$\text{Logique } 1 = 0\,\text{V}, \qquad \text{Logique } 0 = -5\,\text{V}$$

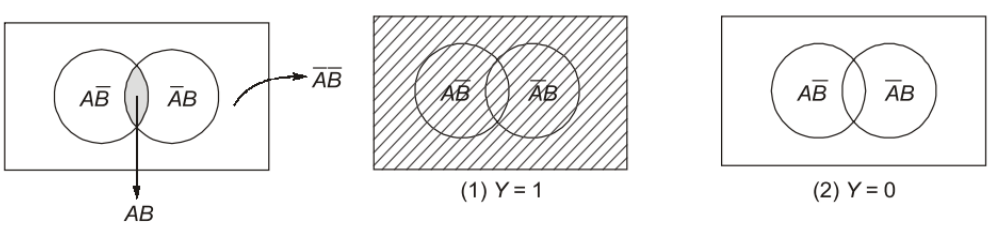
2. Pour la logique négative :

$$\text{Logique } 1 = -0{,}8\,\text{V}, \qquad \text{Logique } 0 = -1{,}7\,\text{V}$$

3.8 Forme diagramme de Venn

Une algèbre de Boole peut être représentée par un diagramme de Venn dans lequel

— chaque variable est considérée comme un ensemble
— l'opération AND est représentée par une intersection
— l'opération OR par une union



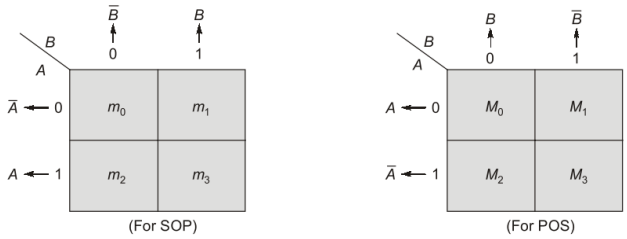
3.9 Carte de Karnaugh

La *carte de Karnaugh* est une méthode graphique qui fournit une procédure systématique pour simplifier et manipuler des expressions booléennes, ou pour convertir une table de vérité en un circuit logique correspondant de manière simple et ordonnée. Dans cette technique, les informations contenues dans une table de vérité ou disponibles sous forme SOP ou POS sont représentées sur la carte de Karnaugh (K-map).

Bien que cette technique puisse être utilisée pour n'importe quel nombre de variables, elle est généralement limitée à 6 variables, au-delà desquelles elle devient très lourde à manipuler, comme pour une K-map à n variables, il y a 2^n cellules.

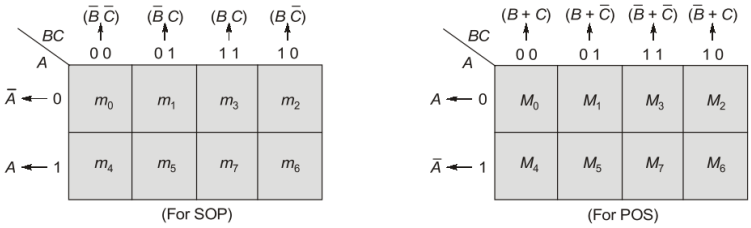
K-map à deux variables

Quatre cellules, quatre mintermes (ou maxtermes).



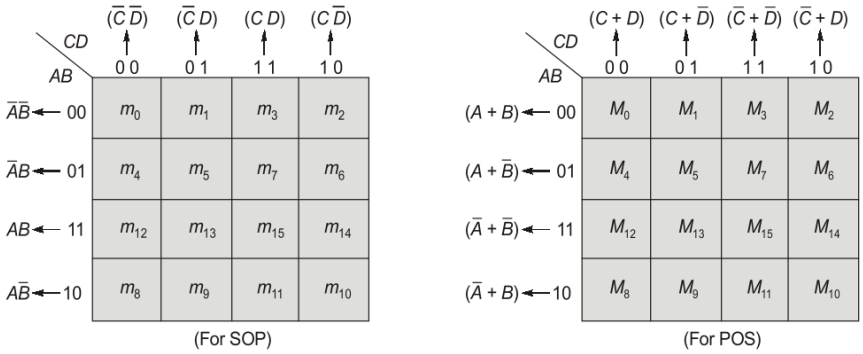
K-map à trois variables

Huit cellules, huit mintermes (ou maxtermes).



K-map à quatre variables

Seize cellules, seize mintermes (ou maxtermes).



Règles complètes de simplification

— Construire la K-map et placer des 1 dans les cellules correspondant aux 1 de la table de vérité. Placer des 0 dans les autres cellules.
— Examiner la carte pour repérer les 1 adjacents et entourer ceux qui ne sont adjacents à aucun autre 1. Ceux-ci sont appelés 1 *isolés*.
— Ensuite, rechercher les 1 adjacents à un seul autre 1 et entourer chaque paire contenant un tel 1.
— Entourer tout groupe de huit cellules (octet), même s'il contient des 1 déjà entourés.
— Entourer tout groupe de quatre cellules (quadruplet) contenant un ou plusieurs 1 non encore entourés, en utilisant le nombre minimal de groupes.
— Entourer toute paire nécessaire pour inclure les 1 restants non encore entourés, en minimisant le nombre de groupes.
— Former la somme logique (OR) de tous les termes générés par chaque groupe.
— Certains circuits logiques peuvent être conçus de sorte que certaines combinaisons d'entrées ne correspondent à aucun niveau de sortie spécifié, généralement parce que ces combinaisons ne se produiront jamais.
— Ainsi, le concepteur peut librement attribuer à une condition *don't care* la valeur 0 ou 1 afin d'obtenir l'expression logique la plus simple.

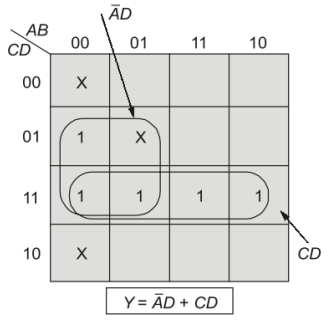
Exemple

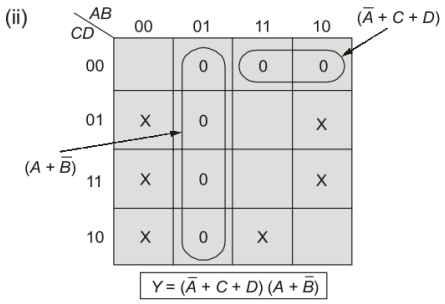
1. En termes de SOP avec conditions *don't care* :

$$f_{SOP}(A,B,C,D) = \sum_{m}(1,3,7,11,15) + d(0,2,5)$$

2. En termes de POS avec conditions *don't care* :

$$f_{POS}(A,B,C,D) = \prod_{M}(4,5,6,7,8,12) \cdot d(1,2,3,9,11,14)$$

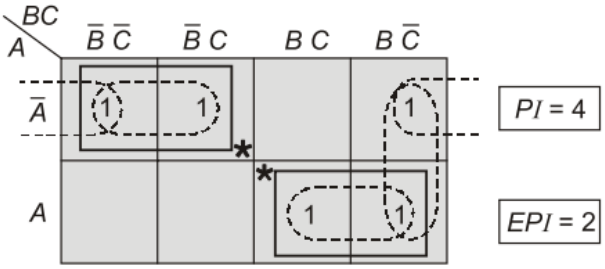




Implicants

Un **implicant** est un terme produit de la fonction donnée pour lequel la sortie de la fonction vaut 1.
Un **implicant premier** est un terme produit minimal de la fonction donnée, tel qu'il est impossible de supprimer un littéral sans perdre la validité de l'implicant.
Un **implicant premier essentiel** est un implicant premier qui couvre au moins un minterme non couvert par aucun autre implicant premier.

Pour la K-map, les implicants, implicants premiers et implicants premiers essentiels sont :



Implicants = $(\bar{A}\bar{B}\bar{C}), (\bar{A}\bar{B}C), (ABC), (\bar{A}BC), (AB\bar{C})$

Implicants premiers (PI) = $\bar{A}\bar{B}, \bar{A}C, AB, B\bar{C}$

Implicants premiers essentiels (EPI) = $\bar{A}\bar{B}, AB$

4 Portes logiques

Les portes logiques sont les éléments fondamentaux de tout système numérique. Elles sont généralement réalisées dans des circuits LSI et VLSI, avec d'autres composants.
Les circuits LSI regroupent beaucoup de composants électroniques sur un même support, tandis que les circuits VLSI en regroupent énormément, au point de former un système complet comme un processeur.

Le terme *porte logique* provient de la capacité de ces dispositifs à prendre des décisions, au sens où ils produisent des sorties différentes pour différentes combinaisons d'entrées.

La fonction de chaque porte logique est représentée par une expression booléenne et donc les entrées et sorties des portes logiques ne peuvent exister qu'à deux niveaux : HAUT et BAS, MARQUE et ESPACE, VRAI et FAUX, MARCHE et ARRÊT, ou simplement 1 et 0.

Les portes logiques sont classées comme :

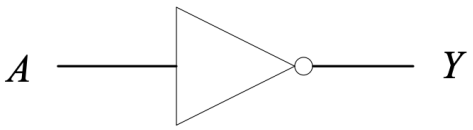
— Portes de base : NOT, AND, OR
— Portes universelles : NAND, NOR
— Portes à usage spécial : EX-OR, EX-NOR

4.1 Portes logiques de base

La porte NOT

L'opération NOT est également appelée *inversion* ou *complémentation*.
La porte NOT possède une seule entrée et une seule sortie : le niveau logique de la sortie est toujours l'opposé du niveau logique de l'entrée.
La sortie est donnée par :

$$Y = \bar{A} = A'$$



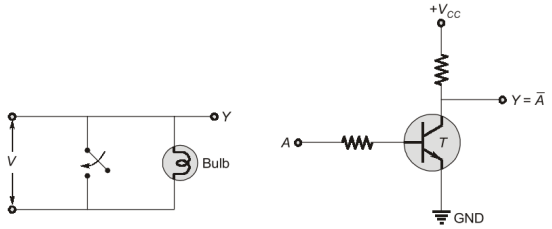
Le petit cercle (bulle) sur le symbole logique indique toujours une inversion.

A	Y
0	1
1	0

Exemple

Les circuits à interrupteur correspondant à une porte NOT sont :
- Lorsque l'interrupteur A est ouvert (logique 0), l'ampoule s'allume (logique 1).
- Lorsque l'interrupteur est fermé (logique 1), l'ampoule est éteinte (logique 0).

Les circuits à interrupteur à transistor correspondant à une porte NOT sont :
- $A = 0$: $T = \text{OFF}$, $Y = +V_{CC}$
- $A = 1$: $T = \text{ON}$, $Y = \text{GND}$

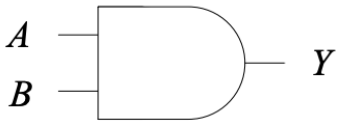


La porte AND

La porte AND peut avoir deux entrées ou plus, mais une seule sortie :
- Si au moins une des entrées est à l'état bas (logique 0), la sortie est 0.
- La sortie vaut 1 uniquement lorsque toutes les entrées sont à 1.

L'expression logique, pour deux entrées, est :

$$Y = AB$$

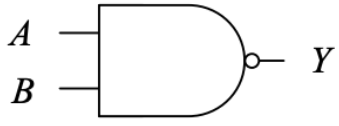


La porte NAND

La porte NAND peut avoir deux entrées ou plus, mais une seule sortie :
- Si toutes les entrées sont à l'état haut (logique 1), la sortie est 0.
- La sortie vaut 1 dès qu'au moins une des entrées est à 0.

L'expression logique est :

$$Y = \overline{AB}$$

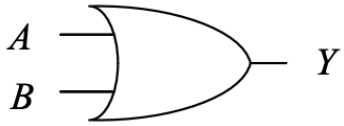


La porte OR

La porte OR peut avoir deux entrées ou plus, mais une seule sortie :
- Si au moins une des entrées est à l'état haut, la sortie est à l'état haut.
- Si toutes les entrées sont à l'état bas, la sortie est à l'état bas.

L'expression logique est :

$$Y = A + B$$

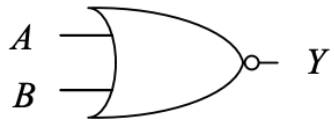


La porte NOR

La porte NOR peut avoir deux entrées ou plus, mais une seule sortie :
- Si au moins une des entrées est à l'état haut (logique 1), la sortie est 0.
- La sortie vaut 1 uniquement lorsque toutes les entrées sont à 0.

L'expression logique est :

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

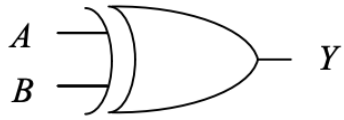


La porte XOR

La porte XOR (eXclusive OR) peut avoir deux entrées ou plus, mais une seule sortie :
- La sortie vaut 1 lorsque les entrées sont différentes.
- La sortie vaut 0 lorsque les entrées sont identiques.

L'expression logique est :

$$Y = A \oplus B = \bar{A}B + A\bar{B}$$



La porte NXOR

La porte XNOR (eXclusive NOR) est la négation de la porte XOR.
Elle peut avoir deux entrées ou plus, mais une seule sortie :
- La sortie vaut 1 lorsque les entrées sont identiques.
- La sortie vaut 0 lorsque les entrées sont différentes.

L'expression logique est :

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