Cours : Structure et fonctionnement des ordinateurs

Youssef Lachhab

BTS DSI : Lycée Hassan 2

Variable logique

Variable binaire qui peut prendre deux états associés au caractère vrai ou faux d'un événement.

État Logique

Valeur attribuée à une variable logique. L'état d'une variable peut être vrai ou faux. On représente l'état vrai par "1" et l'état faux par "0". Une variable dans son état vrai est dite "active".

Opérateurs Logiques

Les opérateurs logiques de base sont ET, OU et NON.

Fonction Logique

Ensemble de variables logiques reliées par des opérateurs logiques. Une fonction logique ne peut prendre que deux valeurs : 0 ou 1.

Signal Logique

Quantité physique qui représente une variable logique dans l'un ou l'autre de ses deux états possibles.

Système Logique

Ensemble de composants qui effectuent des fonctions sur des signaux logiques dans le but de stocker, communiquer ou de transformer de l'information.

Ch2 : La logique combinatoire

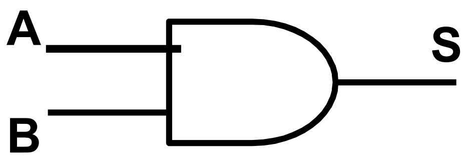
ET

$$S = A \bullet B = AB$$

S est vraie si A est vraie et B est vraie.

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Symbole :



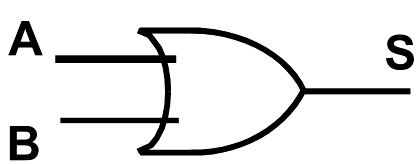
OU

$$S = A + B$$

S est vraie si A est vraie ou B est vraie, ou les deux.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Symbole :



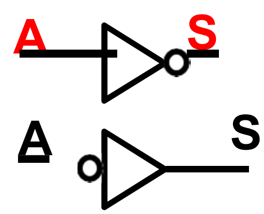
NON

$$S = \overline{A}$$

S est vraie si A est fausse

A	S
0	1
1	0

Symbole :



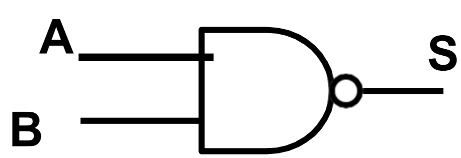
NON-ET

$$S = \overline{A \cdot B} = \overline{AB}$$

S est vraie si (A ou B) est fausse
S est vraie si A est fausse ou B est fausse, ou les deux.

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Symbole :

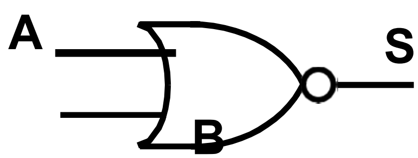


NON-OU

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

S est vraie si (A et B) est fausse.
S est vraie si A est fausse et B est fausse.

Symbole :



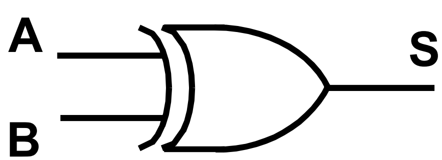
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

XOR (OU EXCLUSIF)

$$S = A \oplus B$$

S est vraie si A est vraie ou B est vraie, mais pas les deux.

Symbole :



A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Remarque : L'expression $S = A \oplus B$ est non-analytique. L'expression analytique du ou exclusif est : $S = \overline{A}B + A\overline{B}$

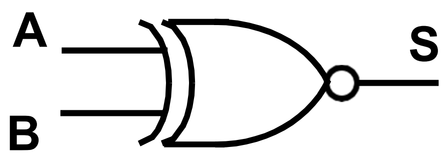
XNOR (NON-OU EXCLUSIF)

$$S = A \otimes B = \overline{A \oplus B}$$

S est vraie si A et B sont vraies ou fausses.

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

Symbole :

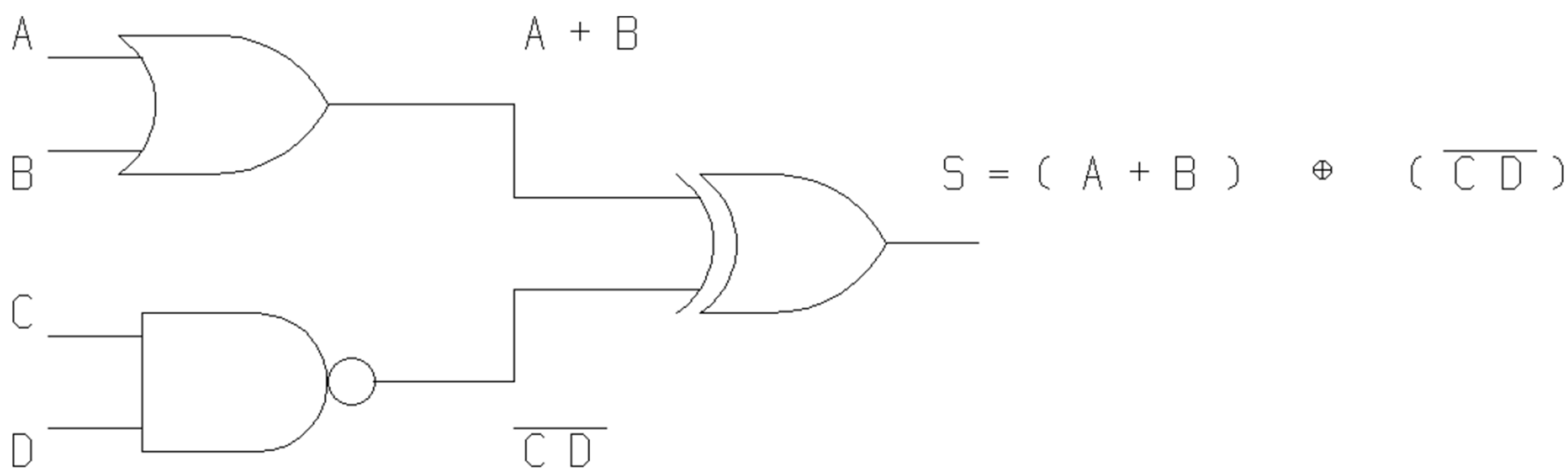


Remarque : L'expression $S = A \otimes B$ est non-analytique. L'expression analytique du non ou exclusif est : $S = AB + \overline{A}\,\overline{B}$

Généralisation

À l'exception des portes XOR et XNOR, ces notions peuvent être généralisées pour des portes à plusieurs entrées.

2.2 Écriture et lecture de schémas



chaque opération est remplacée par son symbole en respectant la hiérarchie des opérations

3. Algèbre de Boole

1. Propriétés de l'algèbre booléenne

$A + 0 = A$	$A \bullet 0 = 0$	$A = A$
$A + 1 = 1$	$A \bullet 1 = A$	
$A + A = A$	$A \bullet A = A$	
$A + \overline{A} = 1$	$A \bullet \overline{A} = 0$	

Hiérarchie des opérations

Dans une expression sans parenthèses, on effectue d'abord les opérations "ET" et, par la suite, les "OU".

Induction parfaite

Dans le domaine linéaire, il n'est pas possible de prouver une équation en la vérifiant pour toutes les valeurs des variables. En logique, puisque les variables sont limitées à deux états, on peut prouver une relation en la vérifiant pour toutes les combinaisons de valeurs pour les variables d'entrée.

Équivalence

Deux fonctions sont équivalentes si on peut leur faire correspondre la même table de vérité.

Si $F = A \cdot B$

Et $G = A + B$

Alors $F = G$

et on dit que F est équivalente à G.

Complémentarité

Deux fonctions sont dites complémentaires si l'une est l'inverse de l'autre pour toutes les combinaisons d'entrées possibles.

Si $F = A \cdot B$

Et $G = A + B$

Alors $F = G$

et on dit que F et G sont complémentaires.

Dualité

Deux expressions se correspondent par dualité si l'on obtient l'une en changeant dans l'autre, les "ET" par des "OU", les "OU" par des "ET", les "1" par des "0" et les "0" par des "1".

Si on sait que $A \cdot B = A + B$ Alors, on saura que $A + B = A \cdot B$

Associativité

$$A + B + C = (A + B) + C = A + (B + C)$$

$$A \bullet B \bullet C = (A \bullet B) \bullet C = A \bullet (B \bullet C)$$

Commutativité

$$A + B = B + A$$

$$A \bullet B = B \bullet A$$

$$(A+B)+C=A+(B+C)=A+B+C$$

$$A \bullet (B+C)=A \bullet B+A \bullet C$$

$$A+(B \bullet C)=(A+B) \bullet (A+C)$$

Distributivité

$$A \bullet (B + C) = A \bullet B + A \bullet C = AB + AC$$

Associativité

$A + B + C = (A + B) + C = A + (B + C)$ $A \bullet B \bullet C = (A \bullet B) \bullet C = A \bullet (B \bullet C)$

Commutativité

$A + B = B + A$

$A \bullet B = B \bullet A$

Distributivité

$A \bullet (B + C) = A \bullet B + A \bullet C = AB + AC$

Théorème de DeMorgan

Première Forme : $\overline{A+B+C+\ldots} = \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\ldots$

Deuxième Forme : $\overline{A\cdot B\cdot C\cdot\ldots} = \bar{A}+\bar{B}+\bar{C}+\ldots$

Exercice :

$a+ab =$

$a+\overline{a}b =$

$a.(a.b) =$

$(a+b).(a+c) =$

$\overline{a}c+ab =$

La décomposition de Shannon :

Fonction Logıque :

$f(a,b,c,d)=ab+\bar{a}cb+d$

$$\mathbf{f(x_1,x_2,x_3,\ldots\ldots\ldots\ldots x_n)=\bar{x}_1 f(0,x_2,x_3,\ldots\ldots\ldots,x_n)+x_1 f(1,x_2,x_3,\ldots\ldots x_n)}$$

<u>Loi de Morgan :</u>

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

$f(a,b)=\overline{a+b}$

$f(a,b)=\bar{a}f(0,b)+af(1,b)$

$\quad=a.\overline{(0+b)}+a\overline{(a+b)}$

$\quad=\bar{a}.\bar{b}+a.\bar{1}$

$\quad=\bar{a}.\bar{b}$

Forme canonique

Une expression est sous sa forme canonique si tous les symboles qui représentent les variables apparaissent dans tous les termes qui la constitue. Lorsqu'une équation est écrite à partir de sa table de vérité, elle est dans sa forme canonique.

- ✓ Si une fonction est une somme de produits, on a une somme canonique.

Exemple :

$$F = \overline{A}BC + ABC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C}$$

- ✓ Si une fonction est un produit de somme, on a un produit canonique.

Exemple :

$$G = (\overline{A} + B + C) \cdot (A + B + C) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C})$$

Écriture par table de vérité :

La fonction vaut 1 si le nombre de 1 est supérieur au nombre de 0.

a	b	c	f	$\overline{f}$
0	0	0	0	1
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	0

Tableau 1 : nombre de 1 > au nombre de 0

Forme canonique :

A. Définition :

c'est l'écriture algébrique de la fonction logique sous la forme de :

- ❖ somme de produit, **première** forme canonique,
- ❖ produit de somme, **deuxième** forme canonique,
- ❖ de portes NAND, **troisième** forme canonique,
- ❖ de portes NOR, **quatrième** forme canonique.

B. Applications :

Si on reprend la fonction du Tableau 1, on peut écrire :

❖ **première forme canonique**, on recherche les combinaisons des variables logiques sous la forme

de somme de produit qui amènent la fonction logique à la valeur 1,

$$f=1 \ \ si \ f=\overline{a}.b.c+a.\overline{b}.c+a.b.\overline{c}+a.b.c$$

deuxième forme canonique, on recherche les combinaisons des variables logiques sous la forme
de produit de somme qui amènent la fonction logique à la valeur 0,

,

$$f=0 \ si \ f=(a+b+c).(a+b+\bar{c}).(a+b+c).(\bar{a}+b+c)$$

a	*b*	*c*	*1ère forme appliquée à f=0*	*2ème forme*
0	0	0	$\bar{a}.\bar{b}.\bar{c}$	a+b+c
0	0	1	$\bar{a}.\bar{b}.c$	$a+b+\bar{c}$
0	1	0	$\bar{a}.b.\bar{c}$	$a+\bar{b}+c$
1	0	0	$a.\bar{b}.\bar{c}$	$\bar{a}+b+c$

Soit (E; +; ×;) une algèbre de Boole.

Une fonction booléenne *f* de *n* variables booléennes de E^n dans E est une application qui à tout *n*-uplet de E fait correspondre un élément de E construit à l'aide des opérations booléennes.

Exemples :

$$f(a, b, c) = a + \bar{a}b + bc$$

$$g(a, b, c, d) = abc + bcd + ad + b$$

Mintermes et maxtermes.

- ❖ Un « minterme » de n variables booléennes est un produit comportant n facteurs, chaque facteur correspondant à une variable donnée ou à son complémentaire.

- ❖ Un « maxterme » de n variables booléennes est une somme comportant n termes, chaque terme correspondant à une variable donnée ou à son complémentaire.

Exemple.

Soit a, b, c et d quatre variables booléennes.

$abcd$, $a\bar{b}c\bar{d}$, $abcd$ et $ab\bar{c}d$ sont quatre mintermes construits à partir des variables a, b, c et d

$$a + b + c + \bar{d} \text{ , } a + \bar{b} + c + \bar{d} \text{ et } a + \bar{b} + c + \bar{d}$$

sont trois maxtermes construits à partir des variables a, b, c et d.

Remarque : à partir de n variables booléennes, on peut élaborer 2^n mintermes et 2^n maxtermes.

Formes canoniques disjonctive et conjonctive d'une fonction booléenne.

Soit f une fonction booléenne de E^n dans E.

Il est possible d'écrire f de façon unique sous la forme d'une somme de mintermes.

Cette somme est appelée « forme disjonctive » de f.

De façon analogue, il est possible d'écrire f sous la forme d'un produit de maxtermes.

Ce produit est appelé « forme conjonctive » de f.

Formes canoniques disjonctive et conjonctive d'une fonction booléenne.

Exemple.

Considérons la fonction booléenne : $f(a,b,c) = a\overline{b}$.

On a :

$$
\begin{aligned}
f(a,b,c) &= a\overline{b}1 && \text{(1 est élément neutre de la multiplication)} \\
&= a\overline{b}(c+\overline{c}) && \text{(principe du tiers exclus)} \\
&= a\overline{b}c + a\overline{b}\overline{c} && \text{(distributivité)}
\end{aligned}
$$

$a\overline{b}c + a\overline{b}\overline{c}$ est la forme canonique disjonctive de la fonction *f*.

Formes canoniques disjonctive et conjonctive d'une fonction booléenne.

Par ailleurs, on a :

$$\begin{aligned}
f(a,b,c) &= a\bar{b} + c\bar{c} \\
&= (a\bar{b} + c)(a\bar{b} + \bar{c}) \\
&= (a+c)(\bar{b}+c)(a+\bar{c})(\bar{b}+\bar{c}) \\
&= ((a+c)+b\bar{b})((\bar{b}+c)+a\bar{a})((a+\bar{c})+b\bar{b})((\bar{b}+\bar{c})+a\bar{a}) \\
&= (a+c+b)(a+c+\bar{b})(\bar{b}+c+a)(\bar{b}+c+\bar{a})(a+\bar{c}+b)(a+\bar{c}+\bar{b})(\bar{b}+\bar{c}+a)(\bar{b}+\bar{c}+\bar{a}) \\
&= (a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+c)(\bar{a}+\bar{b}+\bar{c})
\end{aligned}$$

$(a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+c)(\bar{a}+\bar{b}+\bar{c})$ est la forme canonique conjonctive de la fonction f.

Exemple 2.2

Trouvez la somme et le produit canonique de la fonction suivante:

A	B	C	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$L = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$M = \bar{L} = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C}$$

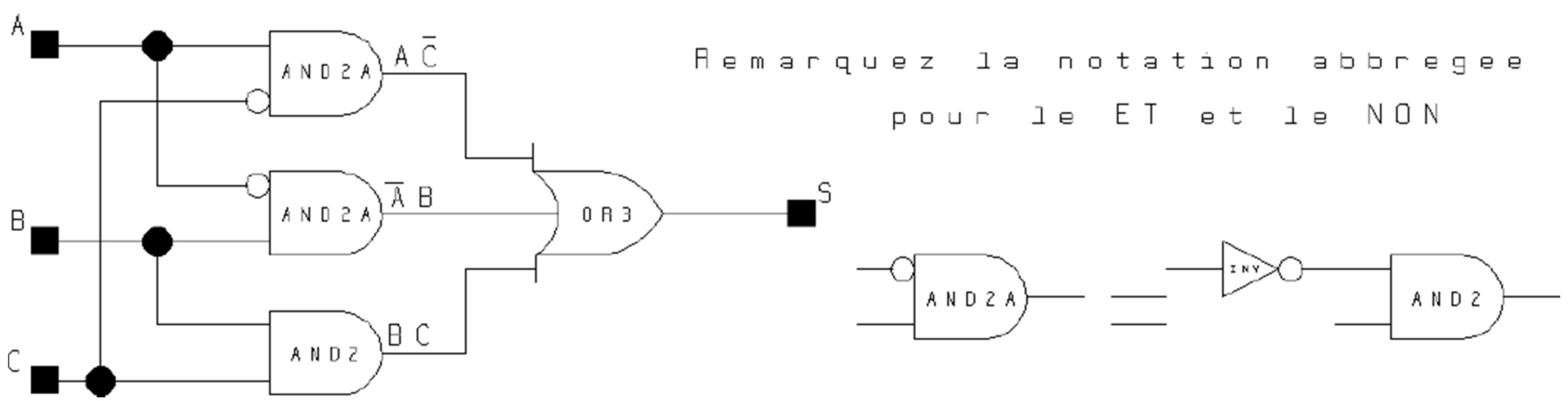
$$L = \bar{M} = \overline{\bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C}}$$

$$L = (\overline{\bar{A}\bar{B}\bar{C}})(\overline{\bar{A}BC})(\overline{A\bar{B}C})(\overline{AB\bar{C}})$$

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

Exemple 2.3

On veut simplifier le circuit suivant :



La transcription directe du schéma en équation donne :

$$S = A\overline{C} + \overline{A}B + BC$$

En reformulant sous la forme canonique on obtient :

$$S = A\overline{C}\,(B + \overline{B}) + \overline{A}B\,(C + \overline{C}) + BC\,(A + \overline{A})$$

$$S = AB\overline{C} + A\overline{B}\,\overline{C} + \overline{A}BC + \overline{A}B\overline{C} + ABC + \overline{A}BC$$

Le terme ABC apparaît deux fois, on peut en éliminer un :

$$S = AB\overline{C} + A\overline{B}\,\overline{C} + \overline{A}BC + \overline{A}B\overline{C} + ABC$$

On regroupe pour mettre AB et $\overline{A}B$ en facteur:

$$S = AB\overline{C} + ABC + \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}\,\overline{C}$$

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

$$S = AB + \overline{A}B + A\overline{BC}$$

Mise en facteur de B:

$$S = B(A + \overline{A}) + A\overline{B}\overline{C}$$

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

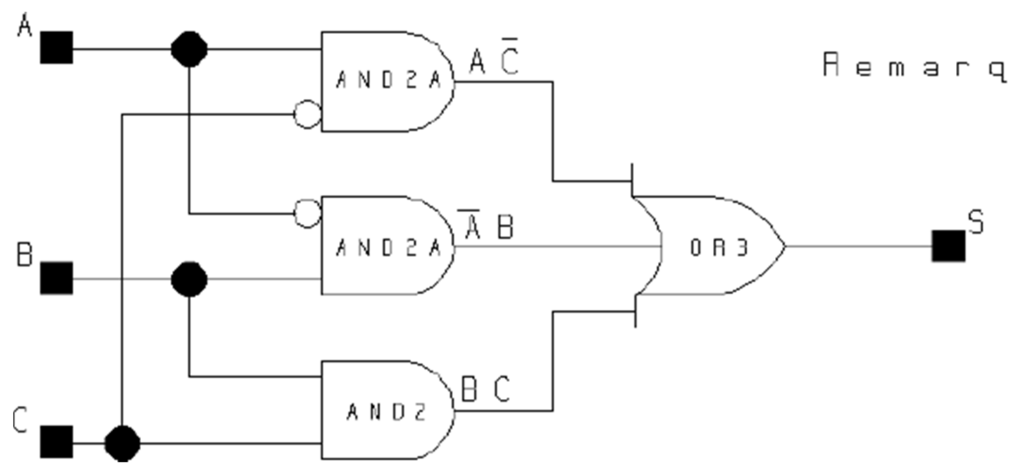
Création d'un nouveau terme pour mettre $A\overline{C}$ en évidence:

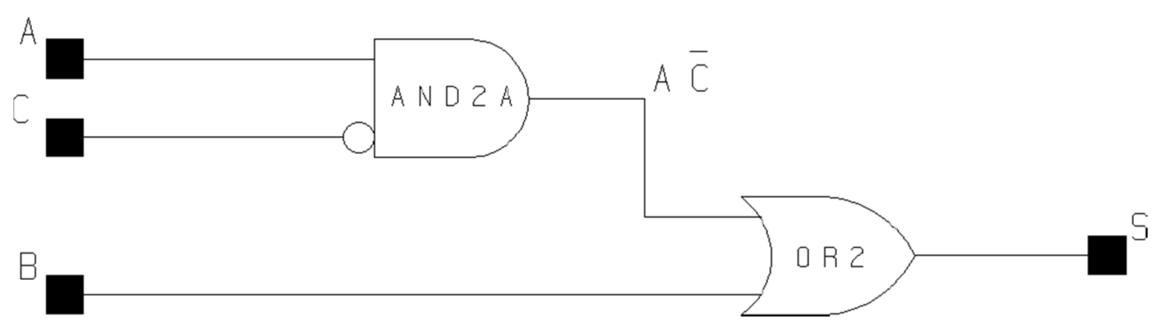
$$S = B(1 + A\overline{C}) + A\overline{B}\overline{C}$$

$$S = B + AB\overline{C} + A\overline{B}\overline{C}$$

$$S = B + A\overline{C}(B + \overline{B})$$

$$\boxed{S = B + A\overline{C}}$$





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

4. Synthèse et simplification

La synthèse des systèmes combinatoires permet de représenter une fonction logique sous une forme telle qu'on puisse la réaliser dans un système avec un nombre minimal de composants, pour la simplicité et la performance.

1. Karnaugh

La méthode de Karnaugh permet de visualiser une fonction et d'en tirer intuitivement une fonction simplifiée. L'élément de base de cette méthode est la table de Karnaugh qui représente, sous forme de tableau, toutes les combinaisons d'états possibles pour un nombre de variables donné. Chaque dimension du tableau pouvant représenter visuellement deux variables, on peut facilement simplifier des fonctions à quatre variables à l'aide d'un tableau à deux dimensions. Bien que plus difficile à visualiser, la simplification de fonctions à cinq ou six variables est possible à l'aide de tableau à trois dimensions.

Code Gray

Le code de Gray, quant à lui, a été élaboré à partir des deux caractéristiques suivantes:

- La transition d'un mot au mot suivant implique qu'un, et seulement un, bit change d'état.
- Le code est cyclique.

La table suivante montre les codes de Gray à 1, 2 et 3 bits.

Gray à 1 bit	Gray à 2 bits	Gray à 3 bits
0	00	000
1	01	001
	11	011
	10	010
		110
		111
		101
		100

Théorème d'adjacence

Le deuxième concept nécessaire à la compréhension des tables de Karnaugh est l'adjacence logique. Deux mots binaires sont dits adjacents s'ils ne diffèrent que par la complémentarité d'une, et seulement une, variable. Si deux mots sont adjacents sont sommés, ils peuvent être fusionnés et la variable qui diffère est éliminée.

Par exemple, les mots ABC et $AB\overline{C}$ sont adjacents puisqu'ils ne diffèrent que par la complémentarité de la variable C. Le théorème stipule donc que :

$$ABC + AB\overline{C} = AB$$

Tables de Karnaugh

La table de Karnaugh a été construite de façon à faire ressortir l'adjacence logique de façon visuelle. La figure suivante représente des tables de Karnaugh à deux, trois, quatre et cinq variables.

B \ A	0	1
0		
1		

C \ BA	00	01	11	10
0				
1				

Ch2 : La logique combinatoire

DC \ BA	00	01	11	10
00				
01				
11				
10				

DC \ BA	00	01	11	10
00				
01				
11				
10				

E= 0

DC \ BA	00	01	11	10
00				
01				
11				
10				

E= 1

La méthode de Karnaugh consiste à indiquer dans la table les cases correspondantes aux états de variable d'entrées produisant une sortie vraie. Cela peut être déterminé à partir de l'équation de la fonction ou sa table de vérité. Il faut toutefois porter une attention particulière lors du transfert d'information de la table de vérité puisque celle-ci utilise un code binaire naturel, qui est sensiblement différent du code de Gray de la table de Karnaugh. Lorsque toute la fonction est représentée dans la table, on procède à des regroupements de "1" qui se situent les uns à côté des autres. Puisque la table de Karnaugh utilise un code de Gray, ces groupements identifient des termes adjacents. La figure suivante identifie certains groupements typiques. Il est important de noter que les groupements sont toujours des rectangles (les carrés sont aussi des rectangles) contenant un nombre de "1" qui est une puissance de deux.

Ch2 : La logique combinatoire

DC \ BA	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

$S=A$

DC \ BA	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

$S=\overline{C}$

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	1
10	0	0	1	1

$S=DB$

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	1	0	0	1
10	0	0	0	0

$S=C\overline{A}$

DC \ BA	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

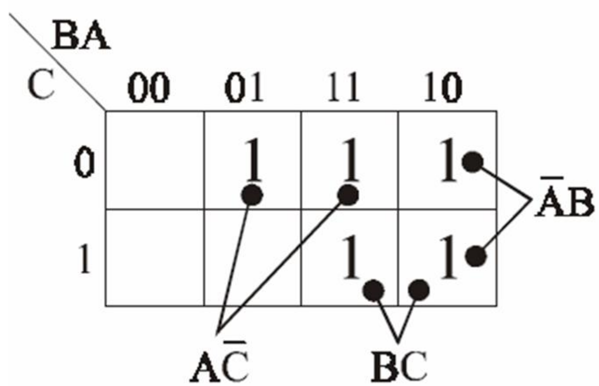
$S=\overline{C}\overline{A}$

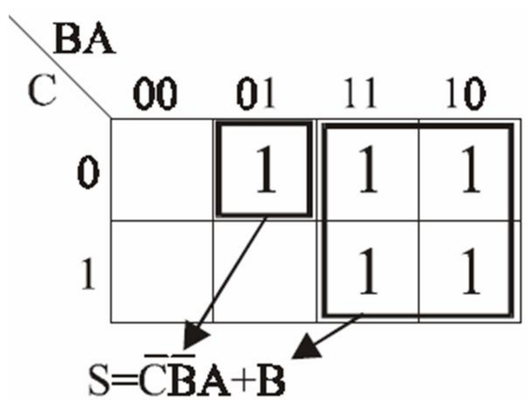
DC \ BA	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

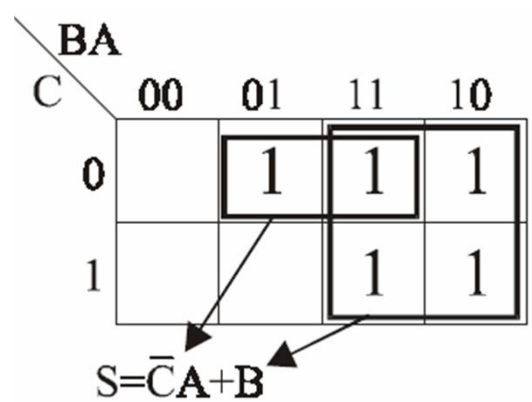
$S=CA$

Exemple 2.5

On veut simplifier $S = A\overline{C} + \overline{A}B + BC$







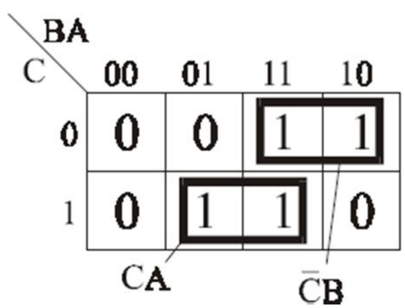
Conditions facultatives

Sous certaines conditions, il se peut qu'un système soit conçu avec des combinaisons d'entrées qui ne se présentent jamais ou qui sont sans intérêts. On parle alors de conditions facultatives (*don't care*).
La représentation dans les tables est symbolisée par "X" ou "∅" (égale 1 ou 0)

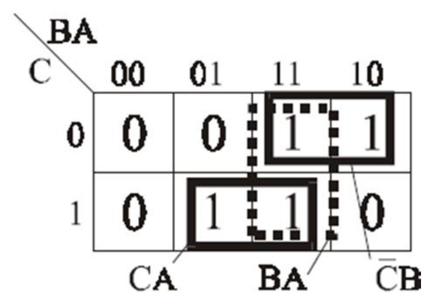
Impliquants redondants

Exemple :

$$F = \bar{C}B\bar{A} + \bar{C}BA + CBA + C\bar{B}A$$



$$F = \bar{C}B + CA$$



$$F = \bar{C}B + CA + BA$$

Ch2 : La logique combinatoire

Symétrie et pliage

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

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

$$F = DA(\overline{B} + \overline{C}) + CB(\overline{D} + \overline{A})$$

$$F = DA\overline{BC} + CB\overline{DA}$$

$$F = DA \oplus BC$$

$$F = DA \oplus BC$$

Symétrie et pliage

C \ BA	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$F = A \oplus B \oplus C$$

DC \ BA	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	0	0	0
10	0	0	0	0

$$F = \overline{D}(A \oplus B \oplus C)$$

Symétrie et pliage

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	1	0	0	0
10	1	1	0	0

$$F = D\bar{B}\bar{A} + D\bar{C}\bar{B} + \bar{D}\bar{C}B + \bar{D}B\bar{A}$$

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

$$F = (\bar{A} + \bar{C})(B \oplus D)$$

$$F = \overline{AC}(B \oplus D)$$

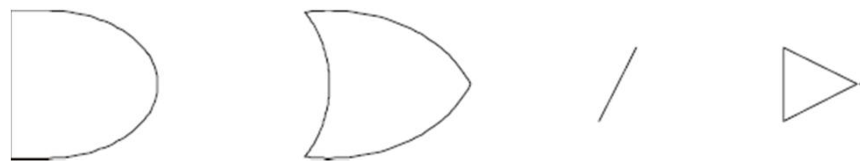
DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	1	0	0	0
10	1	1	0	0

$$F = \overline{AC}(\bar{B}D + B\bar{D})$$

$$F = \overline{AC}(B \oplus D)$$

Symboles et notation

La logique mixte n'utilise que quatre symboles. Ces symboles sont très similaires à ceux utilisés jusqu'ici. Chaque symbole a cependant un certain nombre de variations pour tenir compte de la convention à la sortie et à chacune des entrées. Les quatre symboles de base sont :



Les deux premiers sont les symboles du "ET" et du "OU" respectivement. Le troisième symbole (/ ou *slash*) est utilisé pour signifier un changement de convention, et le dernier symbole, l'inverseur sans la petite boule (o), est un tampon (*buffer*). À ces symboles s'ajoute une notation qui indique la convention du signal. La notation est la petite boule qui vient s'insérer entre le symbole et le fil du signal en logique négative.

Symboles et notation (suite)

Un fil représentant une variable en logique négative aura toujours une petite boule à tous ses nœuds. Une notation spécifique s'applique aussi aux variables pour identifier si elles sont en logique positive ou négative. Les variables en logique positive demeurent inchangées tandis que celles en logique négative doivent être suivies d'un "*".

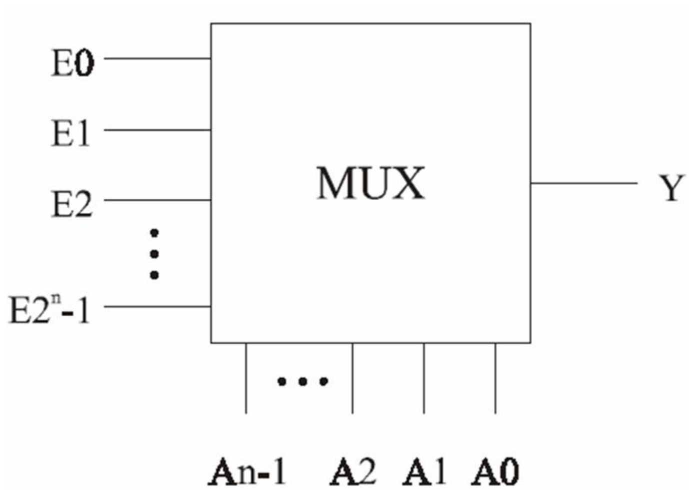
La barre oblique

Le seul nouveau symbole est la barre oblique ("/") qui signifie un changement de convention. Ce symbole se place sur un fil pour séparer deux parties d'un circuit. La convention n'est qu'une interprétation des tensions physique, alors la barre oblique ne correspond à aucun composant physique puisque les tensions demeurent inchangées.
L'exemple suivant illustre divers cas possibles de l'utilisation de la barre oblique. .

2.7 Éléments de base II

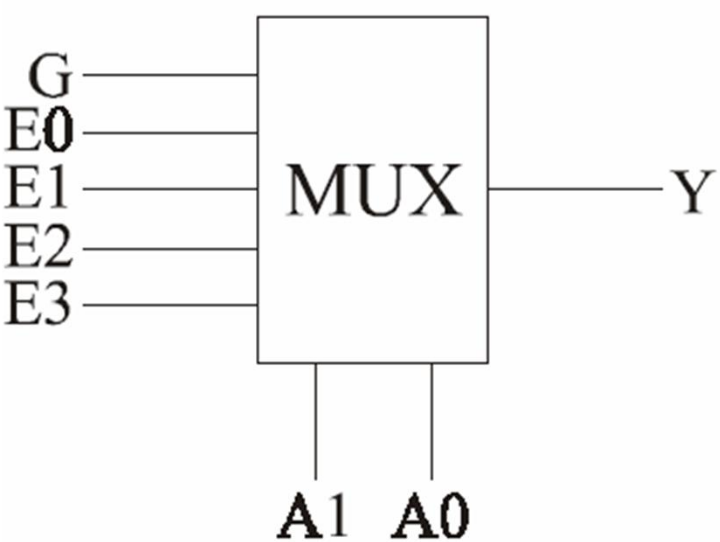
Le multiplexeur (MUX)

Le multiplexeur est un système combinatoire ayant pour fonction de sélectionner une parmi 2^n entrées et de la transmettre à la sortie. La sélection est faite à l'aide de *n* lignes d'adresse. La notation usuelle du MUX est: MUX 2^n à 1.



Exemple :

MUX 4 à 1, 2 lignes d'adresse

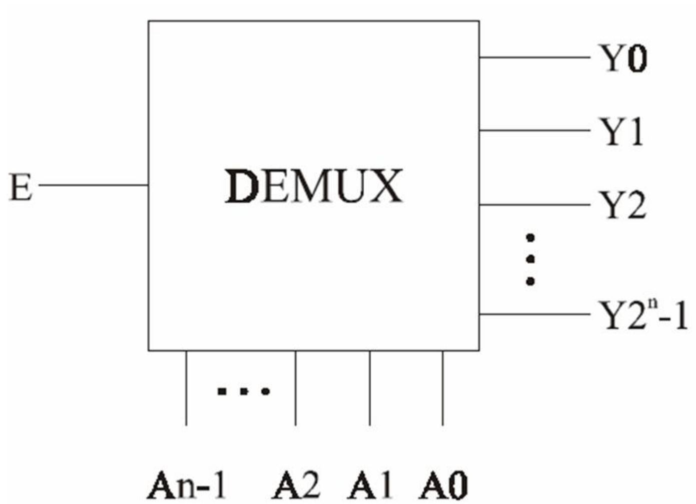


G	A1	A0	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	E0
1	0	1	E1
1	1	0	E2
1	1	1	E3

Le signal supplémentaire *G* (*Strobe*) est un signal d'activation du composant.
Si *G* est inactif, la sortie du MUX sera obligatoirement inactive.

Le démultiplexeur (DEMUX)

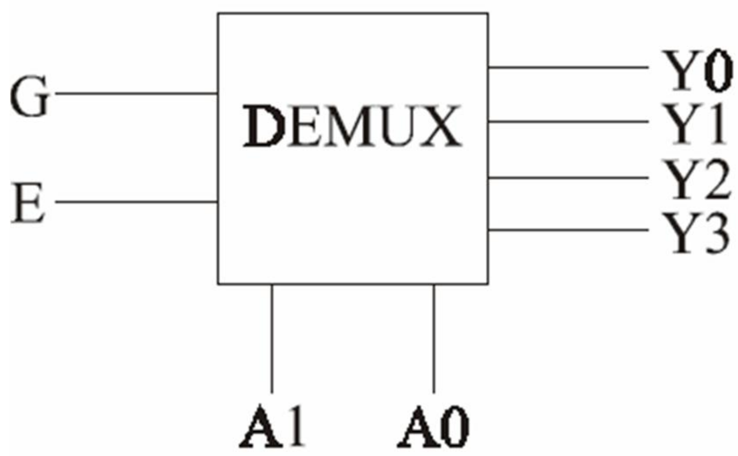
Le démultiplexeur est un système combinatoire ayant pour fonction de transmettre une entrée vers une des 2^n sorties. La sélection est faite à l'aide de *n* lignes d'adresse.
La notation usuelle du DEMUX est : DEMUX 1 à 2^n.



Ch2 : La logique combinatoire

Exemple :

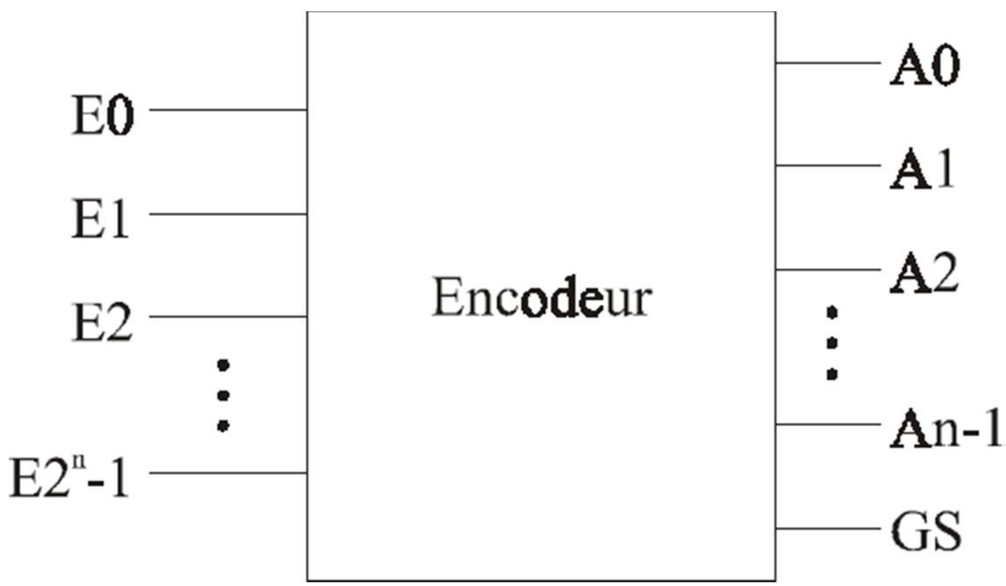
DEMUX 1 à 4, 2 lignes d'adresse



G	A1	A0	Y
0	0	0	Yn inactifs
0	0	1	Yn inactifs
0	1	0	Yn inactifs
0	1	1	Yn inactifs
1	0	0	Y0=E
1	0	1	Y1=E
1	1	0	Y2=E
1	1	1	Y3=E

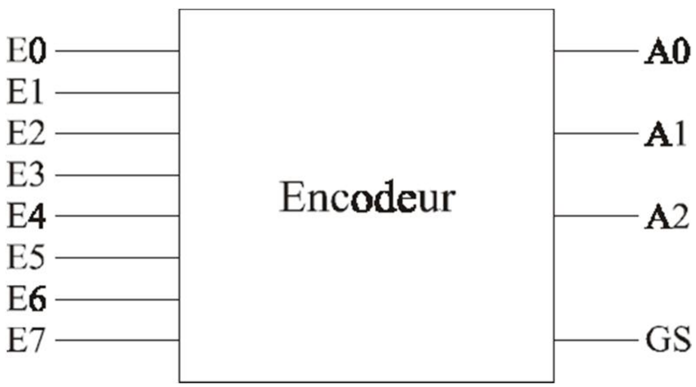
L'encodeur

L'encodeur est un système combinatoire ayant pour fonction de retourner l'index d'activation d'une parmi 2^n entrées. L'index d'activation est donné sur *n* lignes d'adresse. Lorsque plusieurs entrée sont activées, l'encodeur accorde la priorité à l'entrée dont l'index est supérieur.
La notation usuelle de l'encodeur est : encodeur 2^n à n.



Exemple :

encodeur 8 à 3

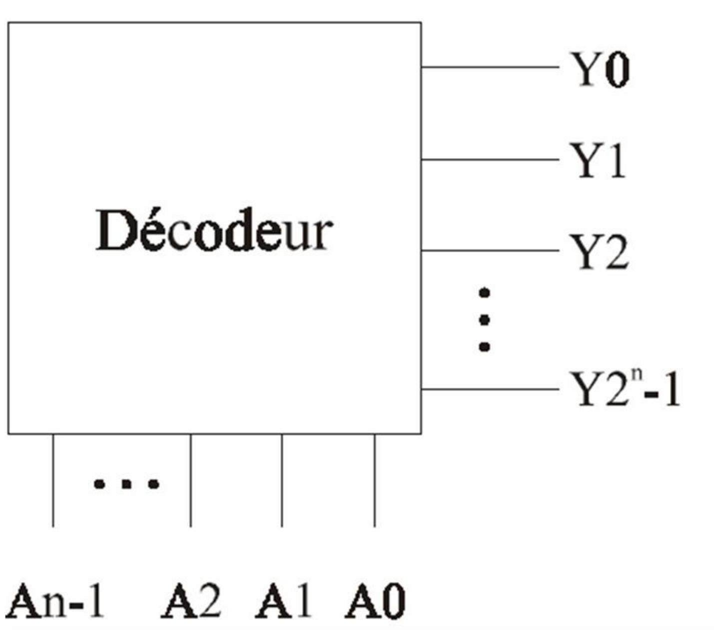


E7	E6	E5	E4	E3	E2	E1	E0	GS	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	X	1	0	0	1
0	0	0	0	0	1	X	X	1	0	1	0
0	0	0	0	1	X	X	X	1	0	1	1
0	0	0	1	X	X	X	X	1	1	0	0
0	0	1	X	X	X	X	X	1	1	0	1
0	1	X	X	X	X	X	X	1	1	1	0
1	X	X	X	X	X	X	X	1	1	1	1

Le signal supplémentaire *GS* (*Got Something*) est un signal qui indique qu'une des entrées est active dans le but de faire la différence entre l'entrée 0 active et lorsqu'aucune entrée n'est activée.

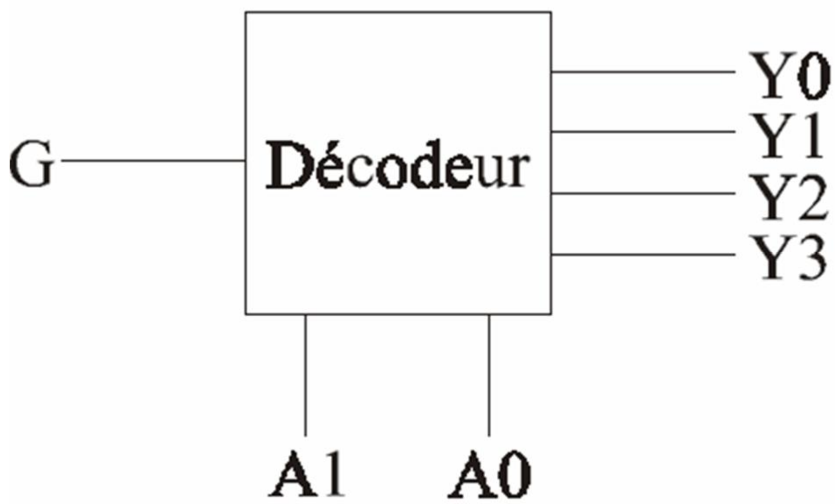
Le décodeur

Le décodeur est un système combinatoire ayant pour fonction d'activer une des 2^n sorties. La sélection est faite à l'aide de *n* lignes d'adresse.
La notation usuelle du décodeur est : décodeur 1 parmi 2^n. Le décodeur se comporte exactement comme un DEMUX avec son entrée toujours à 1.



Exemple :

décodeur 1 parmi 4



G	A1	A0	Y
0	0	0	Yn inactifs
0	0	1	Yn inactifs
0	1	0	Yn inactifs
0	1	1	Yn inactifs
1	0	0	Y0=1
1	0	1	Y1=1
1	1	0	Y2=1
1	1	1	Y3=1

Le signal supplémentaire *G* (*Strobe*) est un signal d'activation du composant. Si *G* est inactif, toutes les sorties du Décodeur seront obligatoirement inactives.