CHAP 2 TABLE DE VÉRITÉ - ALGÈBRE DE BOOLE

rappel du chap. 1:

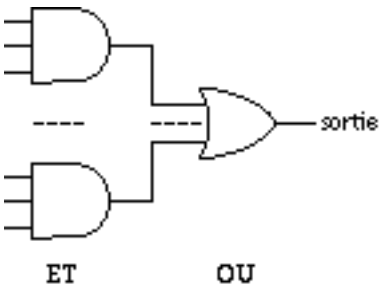
-on emploie un système binaire limité à 2 états 0,1
-problème: étant donné une relation entrée/sortie binaire, faire le design du hardware qui implante cette relation.

- 2 cas 1 - Cas : sorties dépendent seulement des états présents aux entrées
2 - Cas : sorties dépendent de "l'histoire" des entrées

Ce sont les circuits séquentiels e.g diagramme d'état ou *state diagram* décrits par une table d'entrée sortie (chap. 6,7)

2 idées du chap 2 :

1) une fois un problème [entrées présentes – sorties] définit par 1 table on peut avoir une solution en "ANDING ENSEMBLE" une combinaison d'entrée et ORING ensemble toutes les sorties de ces portes et :



C'est ce qu'on appelle une somme de produits.

2) La sortie d'un circuit OU-ET (somme de produits) peut être représentée par une équation en algèbre de Boole. De plus, en appliquant les théorèmes de l'algèbre de Boole on peut réduire le nombre de portes.

2.1 Combinaison d'entrées

Logique combinatoire - car on "combine" des entrées

N jetés d' une pièce de monnaie $\rightarrow 2^N$ *possibilités*

même chose en binaire en notant
que évidemment $(101)_2$ est différent de $(110)_2$

donc N entrées binaires $\Rightarrow 2^N$ états possibles

ex :8 entrées $\Rightarrow 2^8$=256 combinaisons

4 entrées $\Rightarrow 2^4 = 16$ combinaisons

2 entrées $\Rightarrow 2^2 = 4$ combinaisons
(avec e_1 et e_0, voir exemple ci-contre)

2 entrées e_i e_o
0 0
1 0
1 1

2.2 Tables de vérité (tdv)

Une table de vérité définit les relations entrée(s)/sortie(s) en faisant la liste de toutes les possibilités, 1 ligne à la fois dans la table.

Une tdv contient 2^N lignes, N= nombre d'entrées.

On peut avoir plus d'une sortie.

Ex : si N entrées et M sorties $\Rightarrow$ (N+M) colonnes dans la table tdv

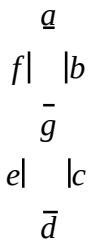
Exemple: de table de vérité (1 entrée $\Rightarrow 2^1$=2 lignes)

entrée	sortie						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0

Entrée = 0 $\Rightarrow$ affiche 0

Entrée = 1 $\Rightarrow$ affiche 1

décodeur 7 segments pour 0 et 1 sur VOM (Volt-O-Meter)



On peut avoir aussi un décodeur pour tous les chiffres 0...9 (ex : circuit TTL 7448 - BCD)

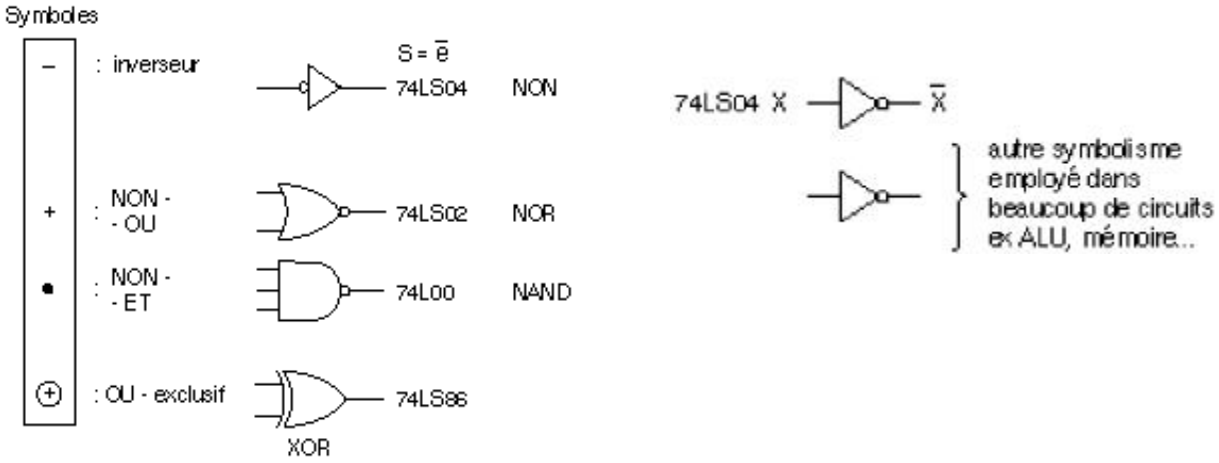
Taille des tables de vérité

La plus petite:
celle de l'inverseur/tampon, (1 entrée $\Rightarrow$ 2'$\Rightarrow$ 2 cas, 2 lignes $\Rightarrow$ Maximum: pas de limite!

Ex : Table de multiplication de 2 chiffres 5 de 8 bits $\Rightarrow 2^8 \times 2^8 = 2^{16} = 65356$
$\Rightarrow$65536 lignes dans la table: $2^{m+n} = 2^{8+8}$

2.3 Primitives Hardware

Première approche, trouver des circuits tous faits ex: TTL, circuits de mémoire (dans lequel on implante la table de vérité. C'est une approche coûteuse, on verra des approches économiques/minimales plus tard.
-Primitives Hardware élémentaires: portes simples eg. 74xx séries,



Type de boîtiers :

Low power - 74LSxx

High speed 74HLSxx

Barre sur le symbole ⇒ inversion

Les portes NAND et NOR incorporent un inverseur ⇒ donc sont des *building blocs* très utiles. Pour les problèmes complexes, on peut combiner plusieurs de ces parties ensemble.

2.4 Méthode pour résoudre une table de vérité

Voici une méthode robuste, sure mais non optimale :

Pour chaque sortie 1 en d'une table de vérité, réaliser le produit des entrées correspondantes (MINTERM)
Faire somme (ou-or) de tous les produits intermédiaires.

Minterm : Un minterm est un produit de variables qui donne un 1 dans la tdv pour les valeurs spécifiées des variables (i.e. pour un rang donné).
Formation d'un Minterm : La variable apparaît non complémentée si elle correspond à un 1 dans la tdv. La variable est complémentée si elle correspond à un 0 dans la tdv.

Il y a 2^N combinaisons possibles pour N entrées (donc 2^N Minterms).

Ex: si on a la table définie par : m(3,5,7)

		A	B	C	SORTIE
$A\bar{B}C$	5	1	0	1	1
$\bar{A}BC$	3	0	1	1	1
ABC	7	1	1	1	1
		toues les autres lignes}			0

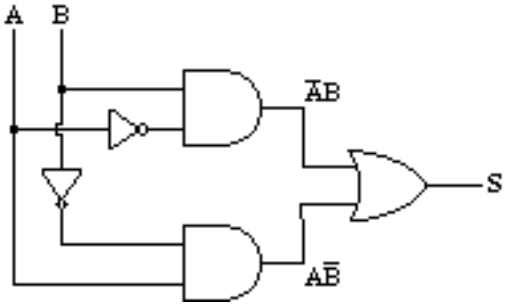
$\begin{cases} alors \\ s = A\bar{B}C + \bar{A}BC = ABC \\ \text{est un circuit de type somme de produit (SOP)} \end{cases}$

Solution correcte, mais non minimale.
Sans réduction subséquente la solution fait référence à une SOP CANONIQUE

Ex: Prenons la tdv d'un ou exclusif :

		A	B	S	liste des minterms
termes à consi-dérer	0	0	0	0	$\bar{A}\bar{B}$
	1	0	1	1	$\bar{A}B$
	2	1	0	1	$A\bar{B}$
	3	1	1	0	AB

$\begin{cases} trouver \textbf{ la SOP canonique} \\ s = \bar{A}B + \bar{B}A \\ \textbf{m(1, 2)} \end{cases}$



Il y a ici 2 étages de portes (délais) après les entrées (directes et complémentées).

Ex : comparateur de deux nombres binaires de 2 bits. On veut la solution de type SOP canonique + schéma complet.
soit la tdv:

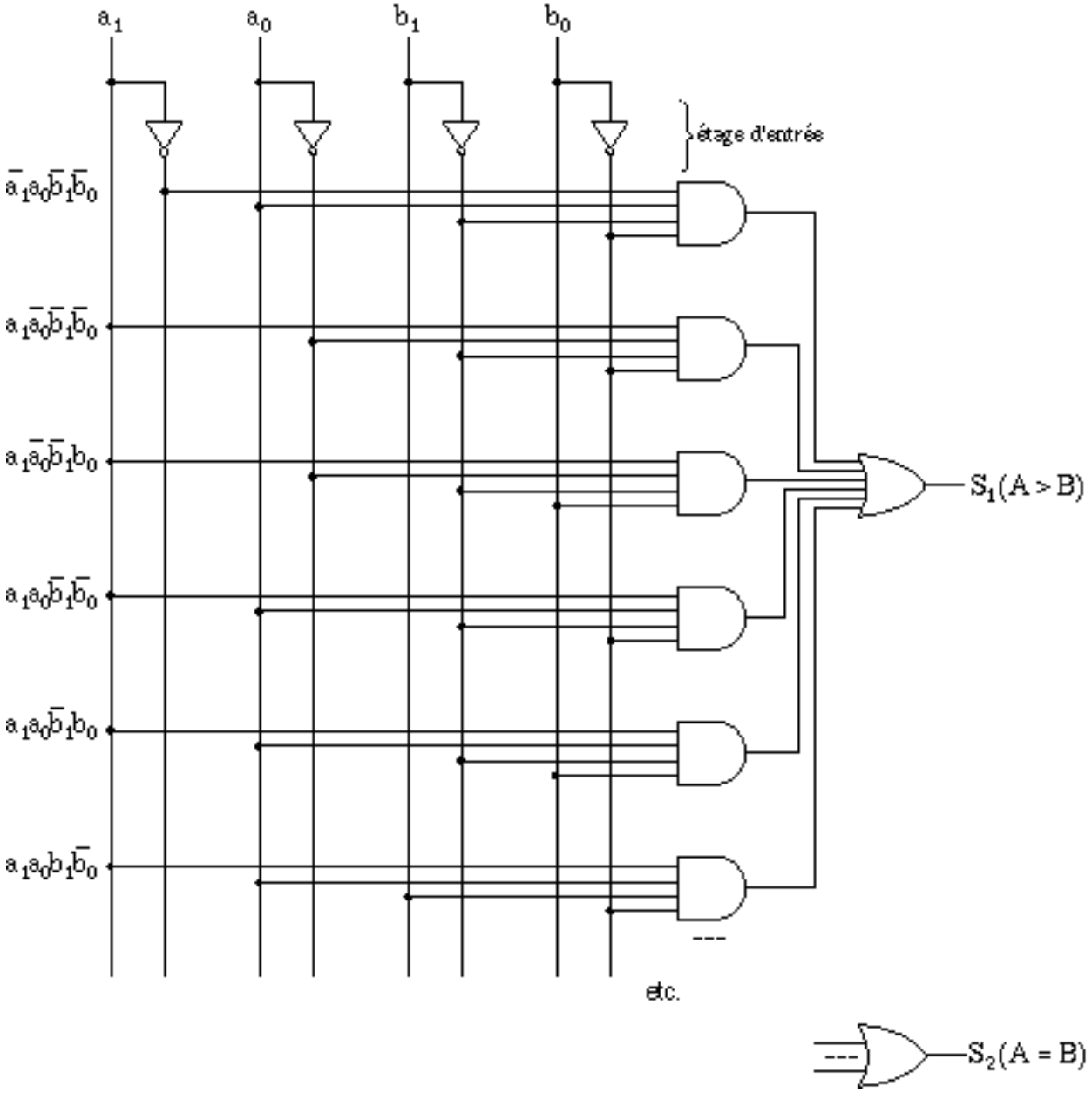
	A		B		A>B sortie	A=B
	a_1	a_o	b_1	b_o	S_1	S_2
0	0	0	0	0	0	1
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	0	1	1	0	0
4	0	1	0	0	1	0
5	0	1	0	1	0	1
6	0	1	1	0	0	0
7	0	1	1	1	0	0
8	1	0	0	0	1	0
9	1	0	0	1	1	0
10	1	0	1	0	0	1
11	1	0	1	1	0	0
12	1	1	0	0	1	0
13	1	1	0	1	1	0
14	1	1	1	0	1	0
15	1	1	1	1	0	1

$S1=\overline{a_1}\, a_o\, \overline{b_1}\, \overline{b_o} + a_1\, \overline{a_o}\, \overline{b_1}\, \overline{b_o} + a_1 \overline{a_o}\, \overline{b_1}\, \overline{b_o} + a_1\, a_o\, \overline{b_1}\, \overline{b_o} + a_1\, a_o\, \overline{b_1}\, b_o + a_1\, a_o\, b_1\, \overline{b_o}$

$S2=\overline{a_1}\, a_o\, \overline{b_1}\, \overline{b_o} + \overline{a_1}\, a_o\, \overline{b_1}\, b_o + a_1 \overline{a_o}\, b_1\, \overline{b_o + a_1\, a_o\, b_1\, b_o}$

Notation de cette tdv, pour A = B
m (0,5,10,15)

A la page suivante on donne le schéma complet pour le cas S_1(A>B)



On voit qu'après l'étage d'entrée, il y a 2 étages de porte pour la sortie.

2.5 Algèbre booléen

-Nous allons maintenant voir d'autres méthodes pour minimiser l'utilisation du hardware et en particulier l'algèbre booléen.

L'algèbre: permet de résoudre des équations pour trouver des inconnues qui sont représentées par des variables.

ex: L'équation d'une parabole :

$y = ax^2 + bx + c$

x : variable indépendante – entrée

y : variable dépendante - sortie

liens avec une table de vérité $\leftrightarrow$ algèbre

graphe $\leftrightarrow$ algèbre nombre réel

En algèbre booléen, on travaillera seulement sur 2 nombres seulement : 0 et 1
Faux Vrai
(False) (True)

et avec les opérations suivants AND, OR, INVERSE

Règles (postulats de Huntington, 1904)

-Fermeture:	A+Be {0,1)	A.Be {0,1) .
-Associativité :	A+(B+C) = (A+B)+C	A.(B.C.)=(A.B).C
-Commutativité	A+ B = B+A	A.B = B.A
-Distributivité	A.(B+C) = A.B+A.C.	A+(B.C.) = (A+B). (A+C)
- Identités pour 0,1		
	0+A=A et 0.A=0	1.A=A et 1+A=1
complément:	$A + \overline{A} = 1$	$A.\overline{A} = 1$

En changeant « + » par « · » on peut reproduire les 2 colonnes, c'est la dualité
Les variables booléennes ont 2 valeurs possibles 0,1
Une expression booléenne est une combinaison de variables booléennes. e.g. SOP

Théorèmes utiles

$\overline{\overline{X}} = X$

$X+\overline{X}=1$	$X.\overline{X}=0$
X+1=1	X.0=0
X+0=X	X.1=X
X+X=X	X.X=X
ou	et

Tous ces théorèmes peuvent se prouver au moyen de règles simples.

La distributivité a 2 formes

$R \bullet (Q + P) = RQ + RP$

$R + (Q.P) = (R + Q).(R + P)$

$2+(3.5)=2+15=17$
et $(2+3).(2+5)=5.7=35$ } Note pas valide pour les nombres réels mais corrects pour les nombres binaires

$0(1.0)=(0+1).(0+0)$
$0 = 0$

Peut se démontrer en testant tous les cas avec une table de vérité c'est ce qu'on appelle une preuve par induction.

Théorèmes Booléens-basés sur la distributivité

$X + X.Y = X$ **ABSORPTION** $X(X+Y)=X$
sortons le X *et on a* : $X(1+Y)$
$X.1 = X$

$X + \overline{X}.Y = X + Y =$ *absorption* $X.(\overline{X}+Y)=X.Y$
$\underbrace{(X+\overline{X})}_{1}.(X+Y) \leftarrow$ *distributivité du* 2 *majuscules OU*
$X.Y + \overline{X}.Y = Y$ **ADJACENCE** $(X+Y).(\overline{X}+Y)=Y$

sortons le $Y(X+\overline{X})Y.1 = Y$*(qfd)*

L'abjacence est un théorème très utile pour réduire les expressions.

Lois de Morgan

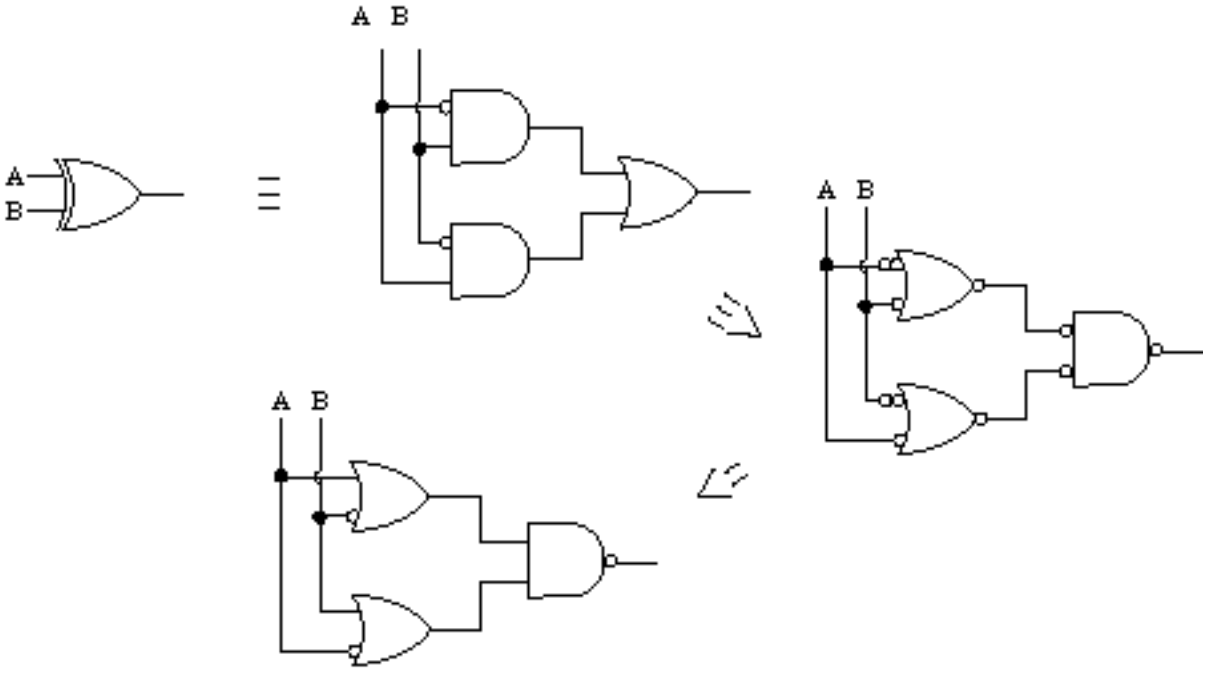
Ces lois permettent de convertir les opérations NAND et NOR , ce qui est très utile pour les simplifications :

1 $\overline{A+B+C+....} = \overline{A}.\overline{B}.\overline{C}...$
2 $\overline{A.B.C.} = \overline{A}+\overline{B}+\overline{C}...$

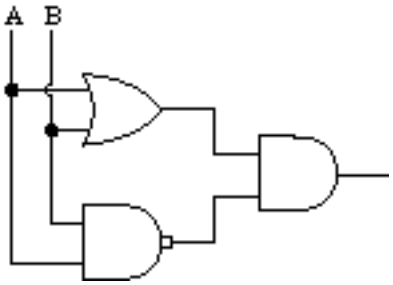
Ou exclusif

$A \oplus B = A.\overline{B}+\overline{A}.B = (A+B).\quad (\overline{A.B})$

et en remplaçant les "et" par des "ou" on obtient :



On peut aussi avoir :



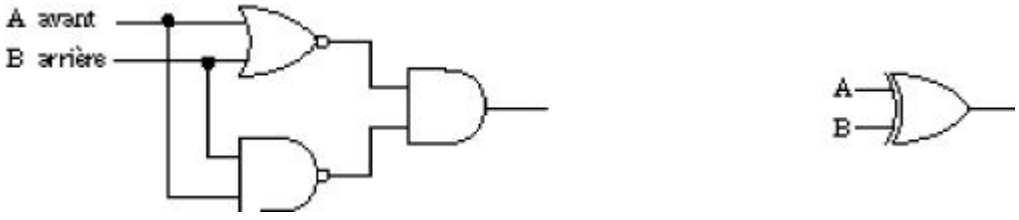
Le ou exclusif est commutatif et associatif et aussi:

$A \oplus \ A = 0$

$A \oplus \ \overline{A} = 1$

La porte ou exclusif est utile pour résoudre certains problèmes logiques, par contre comme elle est une combinaison de primitives, son usage n'est pas très répandu.

Ex : Système d'alarme qui sonne si <u>seulement</u> une des 2 portes est ouverte, mais pas les 2

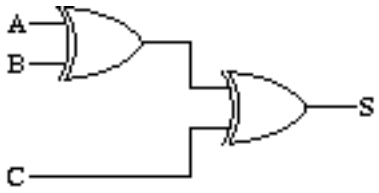


(A+B) $(\overline{AB})$ce qui est l'équivalent de $A \oplus B$ avec porte Ou exclusif. C'est une solution + simple et plus rapide (1 porte vs 3 portes, 1 délai vs 2 délais).

On peut généraliser les portes ou-exclusif à plusieurs entrées.

Ex : à 3 entrées

$A \oplus B \oplus C$

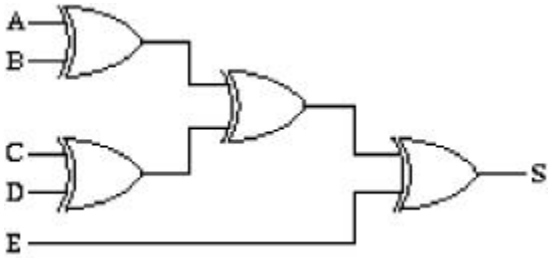


??

$$S = C \oplus (A \oplus B) \begin{cases} \text{est à 1 seulement} \\ \text{si un nombre impair d' entrées est actif} \end{cases}$$

	A	B	C	$(A \oplus B)$	$(A \oplus B) \oplus C$
	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	1
	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	1	0
3	1 1	1 1	0 1	0 0	0 1

Dans l'exemple si contre, on a 1 en sortie avec un nombre impair d'entrées à 1 et notamment avec 1, 3 entrées à 1.

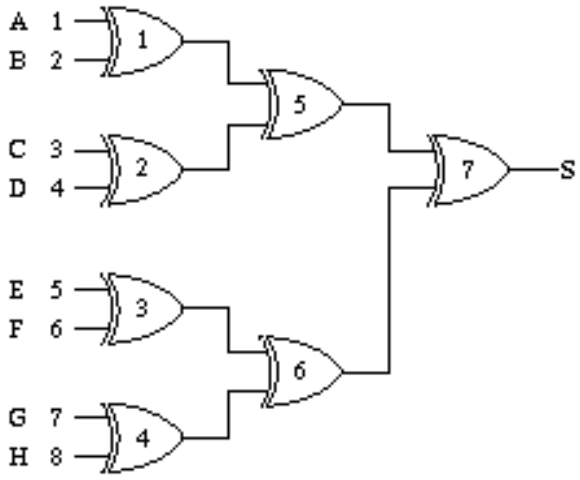


Une porte ou exclusif à 2 entrées est en fait un comparateur.

XOR -> sortie active seulement si les entrées sont différentes
XNOR -> sortie active seulement si les entrées sont égales

On peut s'en servir pour vérifier la parité d'un chaîne de bits (ici pour tester la parité impaire).

Ex : Design d'un circuit de test de parité impaire avec un maximum de 3 délais de porte et dont l'entrée est une chaîne de 8 bits.



sortie=1
si nb impair de bit

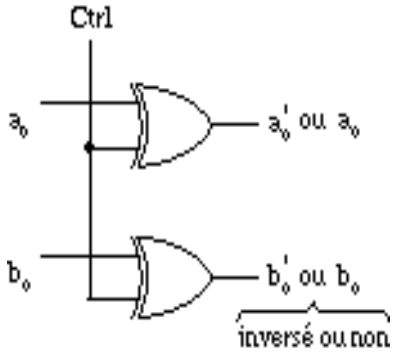
Autre approche possible avec la méthode SOP: 2^8 =128 termes dans la table (et ça exige des porte ET à 8 entrées)

On peut aussi se servir des portes XOR pour modifier la polarité d'un signal.

A	ctrl	
0	0	0
0	1	1
1	0	1
1	1	0

$$A \oplus Ctrl = \begin{cases} A \; si \; \text{ctrl}=0 \\ \overline{A} \; \text{si ctrl}=1 \end{cases}$$

C'est utile dans certaines applications.



2.6 Utilisation de portes NON-ET dans les design SOP (*Sum-of-Products*)

On a déjà vu l'utilisation de l'approche ET-OU pour encadrer les tdv, de $\hat{m}$ avec les XOR pour certaines applications.

On peut aussi employer des $NAND_S$ (NON-ET)

Employons les symboles

$\sum \Leftrightarrow OR \quad \Pi \Leftrightarrow AND,$ alors une forme canonique de

SOP s'expriment OUT $= \Sigma m_i^{\text{termes intermédiaire qui donnent out } =1}$

m_i = mintermes

$$OUT = \Sigma mi = \overline{\overline{\Sigma mi}} = \overline{\Pi mi}$$

Donc la forme de droite représente une double NON-ET (à 2 couches). L'avantage c'est qu'on a plus qu'un seul type de porte NAND au lieu de AND + OR. Ça ne simplifie pas l'expression (on garde la forme canonique si on en avait une au départ).

Ex : On a la table suivante

	A	B	C	S
	0	0	0	0
$\bar{A}\bar{B}C$	0	0	1	1
	0	1	0	0
$\bar{A}BC$	0	1	1	1
$A\bar{B}\bar{C}$	1	0	0	1
	1	0	1	0
$AB\bar{C}$	1	1	0	1
	1	1	1	0

La forme canonique de cette tdv est

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

Simplifions la:

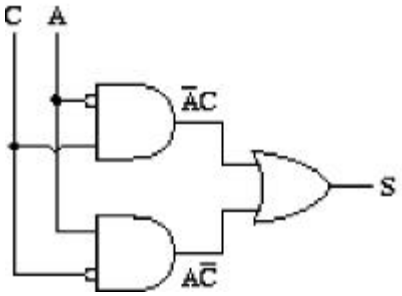
$$S = \underbrace{\bar{A}C}\left(\underbrace{\bar{B} + B}_{1}\right) + \underbrace{\bar{A}C}\left(\underbrace{\bar{B} + B}_{1}\right)$$

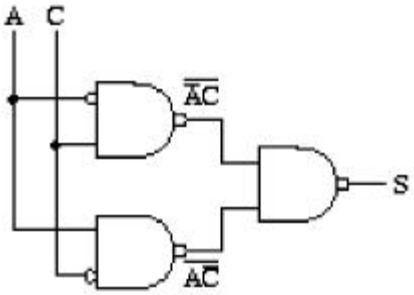
$$S = \bar{A}C + \bar{A}C = (A \oplus C)$$

Ce qui peut se réécrire:

$$S = \left\{\overline{\overline{AC} + \overline{AC}} = \overline{\overline{\overline{AC}} \quad \overline{\overline{AC}}}\right\}$$

Circuit ET-OU





Circuits NAND
Circuit 74LS00

En conclusion: On peut remplacer, sans affecter la sortie, tous les ET et tous les OU d'un design SOP à 2 niveaux par des NANDs.

2.7 Chercher les zéros, design avec des NORS

On va maintenant s'attarder aux "0" dans les tdv. On fera le design de produits de somme (POS = *Product-of-Sum*). On aura des circuits de type : $\overset{\text{circuit}}{\text{OR - ET}}$
POS est la version en logique négative d'un SOP.
C'est utile pour des tables de vérité avec peu de 0, beaucoup de 1.

Maxterm (définition) : C'est une somme qui donne 0 dans la tdv pour les valeurs spécifiées des variables.

Maxterm (réalisation) : C'est une somme de variables.
Les variables apparaissent non complémentées si à 0 dans la tdv.
Les variables apparaissent complémentées si à 1 dans la tdv.

Rappel: Une porte OU à N entrées possède une tdv avec 1 seul zéro.

OU

	A	B	S
0	0	0	$0^{M_o=A+B}_{\text{1 seul zéro}}$
1	0	1	1
2	1	0	1
3	1	1	1

On a toujours des mintermeset des MAXTERMES dans un tdv :
Mi produit et mi somme, 1 = index dans la table

Exemple :

$$m_1 = \overline{A}B \text{ et } M1 = A + \overline{B}$$

minterms en lettre minuscule . maxterme en lettre majuscule {Chaque ligne de la tdv est associée à 1 minterme et à 1 Maxterne

MINTERME: COMBINAISON QUI DONNE 1 (PRODUIT) MAXTERME: COMBINAISON QUI DONNE 0 (SOMME)

ex $m_2 = A\overline{B}$ et $M_2 = \overline{A} + B$

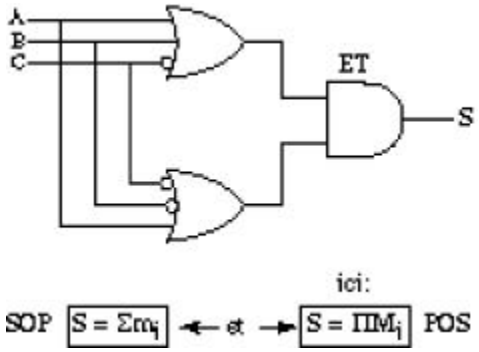
EX: SOIT LA TDV SUIVANTE

A	B	C	S	
0	0	0	1	
0	0	1	0	$M_1 = A + B + \overline{C}$
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	0	$M_6 = +\overline{A} + \overline{B} = C$
1	1	1	1	

Comment combiner ces 2 termes pour la trouver la fonction S? On veut un zéro en S lorsque n'importe lequel de ces termes max sont obtenus et justement, un ET produit un 0 si n'importe quel argument est nul [x.0 = 0] donc :

$$S = M_1 . M_6 = \left(A + B + \overline{C}\right).\left(\overline{A} + \overline{B} + C\right)$$

C'est une équation de type POS = Produit de Somme



2.8 Lois de De Morgan appliquée en circuits POS

On a vu $\boxed{S = \Pi M_i}$ M=Maxterme qui synthétise les 0.

On peut appliquer le raisonnement vu précédemment :

$S = \Pi M_i = \overline{\overline{\Pi M_i}} = \overline{\Sigma \overline{M_i}} \leftarrow NOR$

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

donc :

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

On a donc deux choix selon la tdv :

- peu de 1 design SOP avec NAND $\begin{cases} .\Sigma m_i \\ .(\Pi \overline{m_i}) \end{cases}$

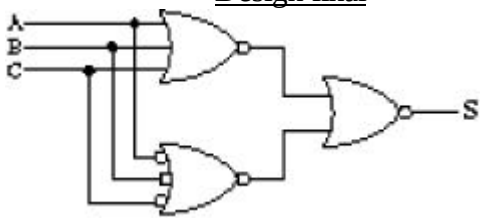
-peu de 0 design POS avec NOR $\begin{cases} \overline{.\Sigma \overline{M_i}} \\ .\Pi M_i \end{cases}$

Ex :

A	B	C	S
0	0	0	$0 \varnothing M_o = A + B + C$
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	1	$0 \varnothing M_7 = \overline{A} + \overline{B} + \overline{C}$
1	1	1	$0 \varnothing M7 = \overline{A} + \overline{B} + \overline{C}$

Ici peu de $0 \Rightarrow$ design avec NOR_S

Design final



donc S = $(M_o . M_7) = (A + B + C) . (\overline{A} + \overline{B} + \overline{C})$

avec NOR_S seulement
Cette expression pourrait être simplifiée éventuellement.

Et avec la relation précédente :

$S = \overline{\overline{M_o . M_7}} = \overline{\overline{M_o} + \overline{M_7}} = \overline{(A + B + C)} + \left(\overline{\overline{A} + \overline{B} + \overline{C}}\right)$ **c'est la solution en NOR_S.**

2.9 Implicants

-Un terme qui ne peut être simplifié davantage sauf en altérant la tdv est appellé un impliquant principal (*prime implicant*).

-Un impliquant: n'importe quel produit d'une expression SOP.

-Un impliqué principal est une somme de compléments qui ne peuvent être simplifiés davantage.

Ex : Soit la relation suivante :

S= AC+BC+AB

-Puisqu'on ne peut simplifier ces termes, ils sont tous des impliquants principaux essentiels (ils doivent être là pour fournir la bonne réponse!)

-La somme d'un ensemble complet d'impliquants principaux est l'expression SOP minimisée d'une tdv.

2.10 Optimisation des design numérique

Méthodes vues :
- SOP canonique (pas simplifié)
- théorèmes booléens pour simplifier
- POS canonique, solution en NANDs et en NORs
- XOR (certains cas)

-On a aucune méthode qui garantisse une réalisation avec un nombre minimum de porte.
-C'est un problème difficile, on verra au chapitre 3 des méthodes pour des résultats surs.
-Il y aura toujours un compromis entre : le nombre de portes total, le nombre de variables dans les termes d'entrées, les délais de propagation.
D'autres critères à considérer:

- modularité
- nombre de portes différentes
- consommation électrique
- bruit (tolérance au)
- synchronisation

2.11 Synchronisation dans les circuits combinatoires

-Les délais de propagation peuvent causer des problèmes, ce sont des considérations importantes, autant que la simplification.

-Les glitches ou aléas sont causés par les délais de propagation différents selon les parcours: certains signaux sont prêts avant d'autres ce qui cause des valeurs de sorties intermédiaires.

⇒ DANGER