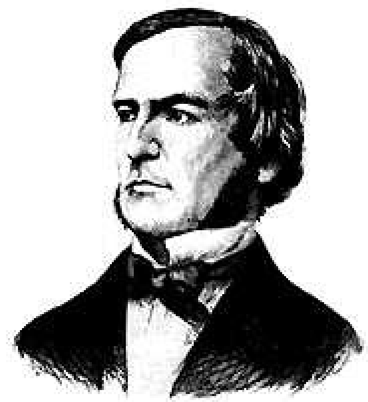


Algèbre de Boole
et
Logique combinatoire

Algèbre de Boole

- $B = \{0, 1\}$: l'espace de Boole
- Une **variable booléenne simple** est définie sur B
  - Deux valeurs possibles
- Une **variable booléenne générale** est définie sur B^n : $(x_{n-1},...x_0)$
  - Juxtaposition de plusieurs bits
  - 2^n valeurs possibles

Algèbre de Boole

Les opérations de l'algèbre de Boole sont les suivantes :

- La **Conjonction** (ou produit logique) de deux variables (bits) : on parlera plus volontiers du **ET** logique, dénoté par un signe croix ($\times$).
- La **Disjonction** (or somme logique) de deux variables (bits) : on parlera plus volontiers du **OU** logique, dénoté par un signe plus ($+$).
- La **Négation** (or complementation or inversion) d'un seul bit : on parlera plus volontiers du **NON**, dénoté par une barre au dessus de la variable ($\bar{a}$).

Ces opérations peuvent être représentées par des **tables de vérité**, qui énumèrent explicitement les correspondances entre les entrées et les sorties :

a	b	$a \times b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

a	$\bar{x}$
0	1
1	0

$x \times y$ vaut 1 si x vaut 1 **et** x vaut 1, 0 sinon.
$x + y$ vaut 1 si x vaut 1 **ou** x vaut 1, 0 sinon.
$\bar{x}$ vaut 1 si x vaut 0, 0 sinon.

Ou inclusif/exclusif ?

Algèbre de Boole

Soit x,y,z des variables booléennes. Les axiomes de l'Algèbre de Boole sont les suivants :

1. $+$ **est associatif** : $x + (y + z) = (x + y) + z$
2. $\times$ **est associatif** : $x \times (y \times z) = (x \times y) \times z$
3. $+$ **est commutatif** : $x + y = y + x$
4. $\times$ **est commutatif** : $x \times y = y \times x$
5. **existence d'un élément neutre pour** $+$: $\exists 0 / x + 0 = x$
6. **existence d'un élément neutre pour** $\times$: $\exists 1 / x \times 1 = x$
7. $\times$ **est distributif sur** $+$: $x.(y + z) = x.y + x.z$
8. $+$ **est distributif sur** $\times$: $x + (y.z) = (x + y).(x + z)$. Ce résultat est plus surprenant !
9. **existence du complément**

Théorèmes

1. $+$ **est idempotent** : $x + x = x$
2. $\times$ **est idempotent** : $x \times x = x$
3. 1 **est absorbant pour** $+$: $x + 1 = 1$
4. 0 **est absorbant pour** $\times$: $x \times 0 = 0$
5. **unicité du complément**
6. **complément du complément** : $\overline{(\overline{x})} = x$
7. $x + x.y = x$
8. $x \times (x + y) = x$
9. $x + \bar{x} \times y = x + y$

Théorème de De Morgan

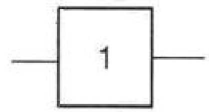
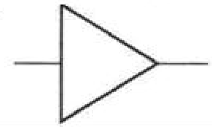
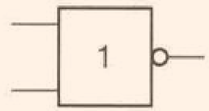
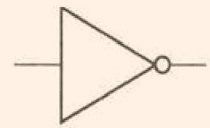
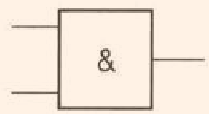
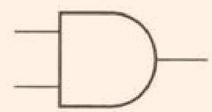
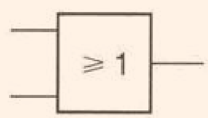
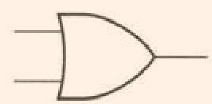
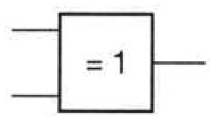
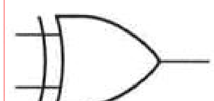
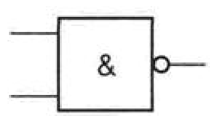
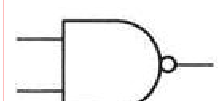
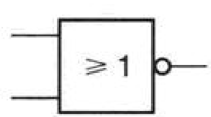
Petit calcul ...

a	b	$a + b$	$Y_1 = \overline{a + b}$	$\overline{a}$	$\overline{b}$	$Y_2 = \overline{a} . \overline{b}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

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

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

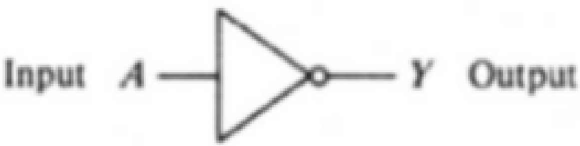
Principales portes logiques

<table>
<tr><td>Porte OUI (YES)</td><td>
</td><td></td><td>entrée sortie
0 0
1 1</td></tr>
<tr><td>Porte NON (NO)</td><td>
</td><td></td><td>entrée sortie
0 1
1 0</td></tr>
<tr><td>Porte ET (AND)</td><td>
</td><td></td><td>entrées sortie
0 0 0
0 1 0
1 0 0
1 1 1</td></tr>
<tr><td>Porte OU (OR)</td><td>
</td><td></td><td>entrées sortie
0 0 0
0 1 1
1 0 1
1 1 1</td></tr>
<tr><td>Porte OU exclusif (XOR)</td><td>
</td><td></td><td>entrées sortie
0 0 0
0 1 1
1 0 1
1 1 0</td></tr>
<tr><td>Porte NON-ET (NAND)</td><td>
</td><td></td><td>entrées sortie
0 0 1
0 1 1
1 0 1
1 1 0</td></tr>
<tr><td>Porte NON-OU (NOR)</td><td>
</td><td></td><td>entrées sortie
0 0 1
0 1 0
1 0 0
1 1 0</td></tr>
</table>

désuets

Symboles les + courants

Inverseur (not)

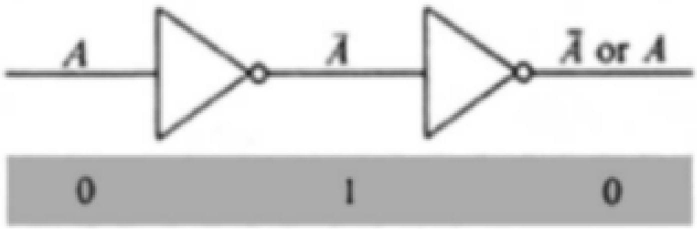


(a) NOT-gate symbol

Input	Output
A	Y
0	1
1	0

Y = /A

(c) NOT Boolean expression

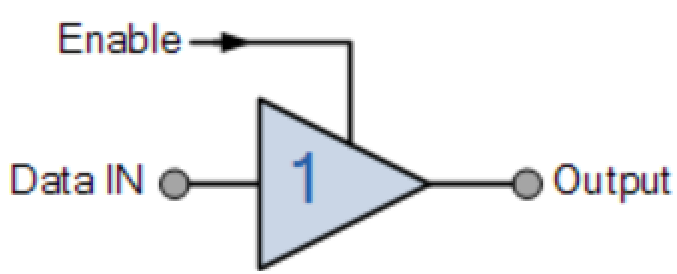


(d) Double inversion

Buffer trois états

une porte à part

Symbol

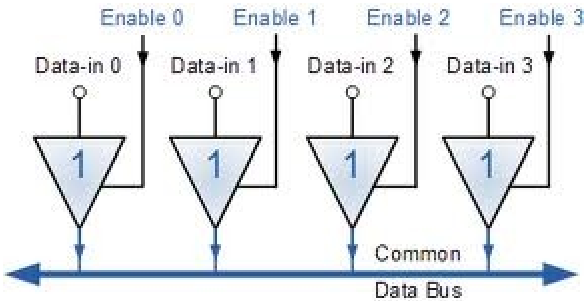


Tri-state Buffer

Truth Table		
Enable	A	Q
1	0	0
1	1	1
0	0	Hi-Z
0	1	Hi-Z

Hi-Z = haute impédance (aucune courant ne circule) = déconnexion
= ni 0 ni 1 =

Indispensable pour éviter les courts circuits lorsque plusieurs circuits tentent d'accéder à un même fil

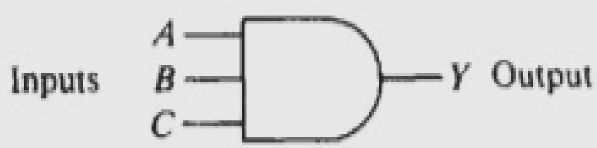


Equation, porte et table de vérité

Représentations « équivalentes »

$$A \cdot B \cdot C = Y$$

(a) Three-variable Boolean expression



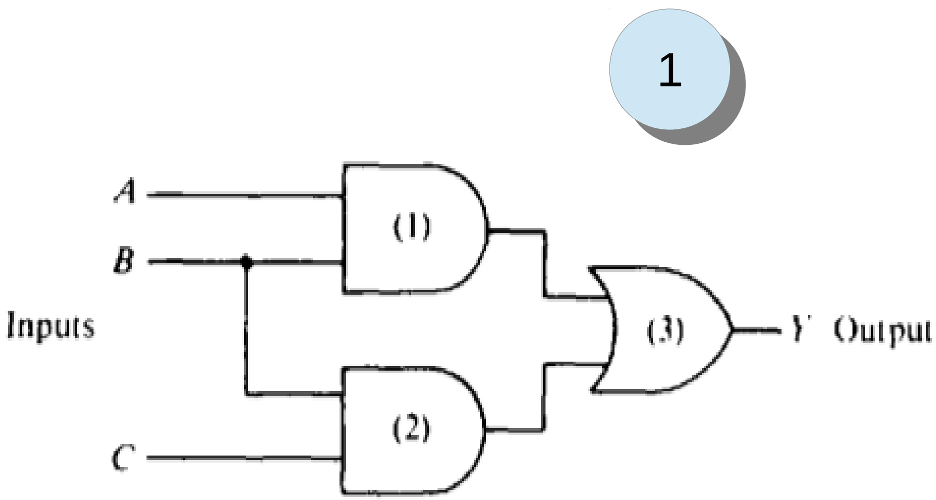
(b) 3-input AND gate symbol

Inputs			Output
C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

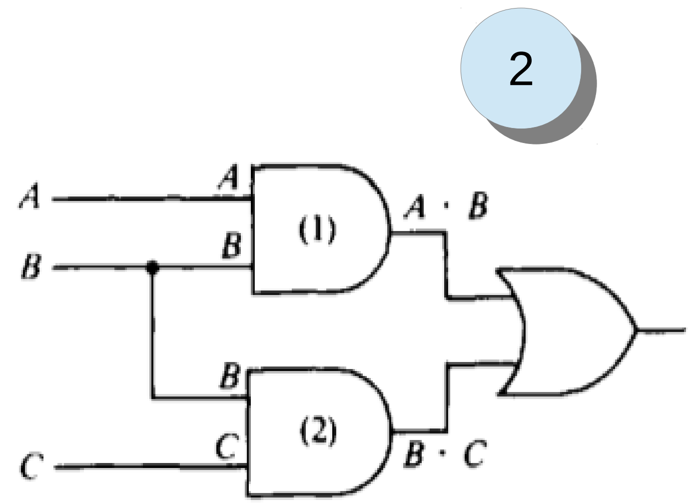
(c) Truth table with three variables

Combinaison de portes logiques

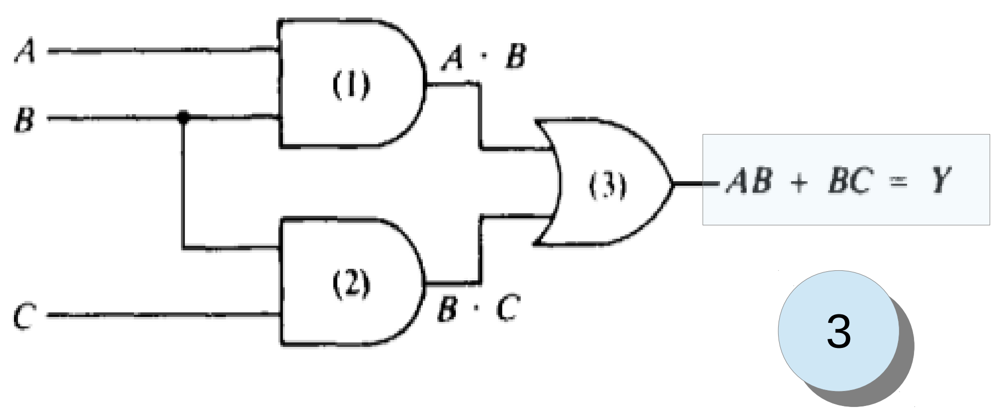
des portes à l'équation



(a) AND-OR logic circuit

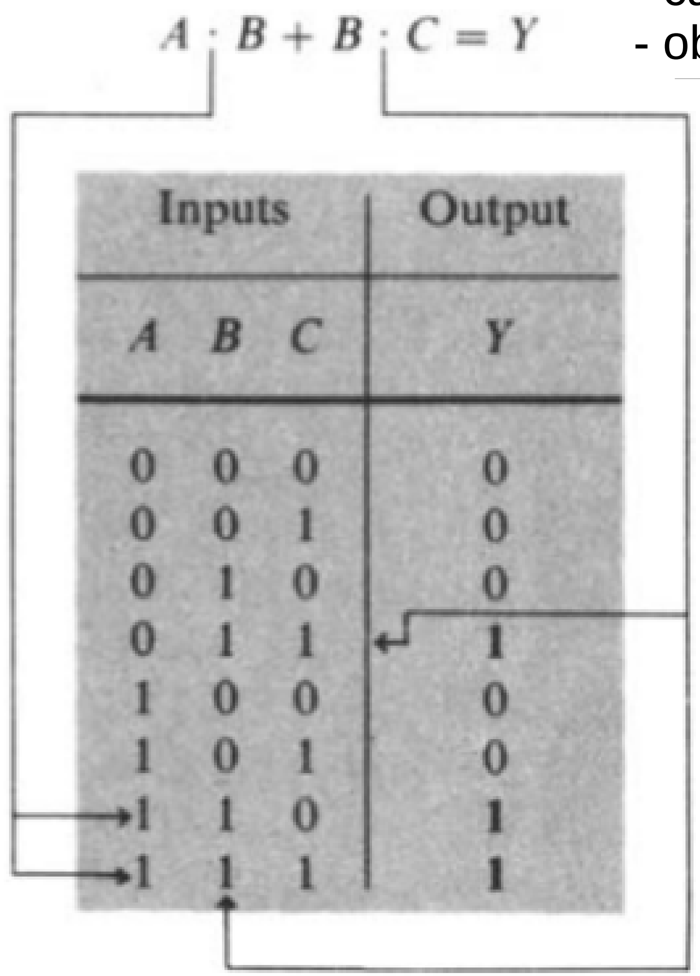


(b) Boolean expressions at the outputs of the AND gates



De l'équation à la table de vérité

Deux méthodes :
- calcul systématique de la fonction pour toutes ses entrées
- observation des valeurs à vrai (ou faux) de la fonction



$$A \cdot B + B \cdot C = Y$$

Inputs			Output
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	0
1	1	0	1
1	1	1	1

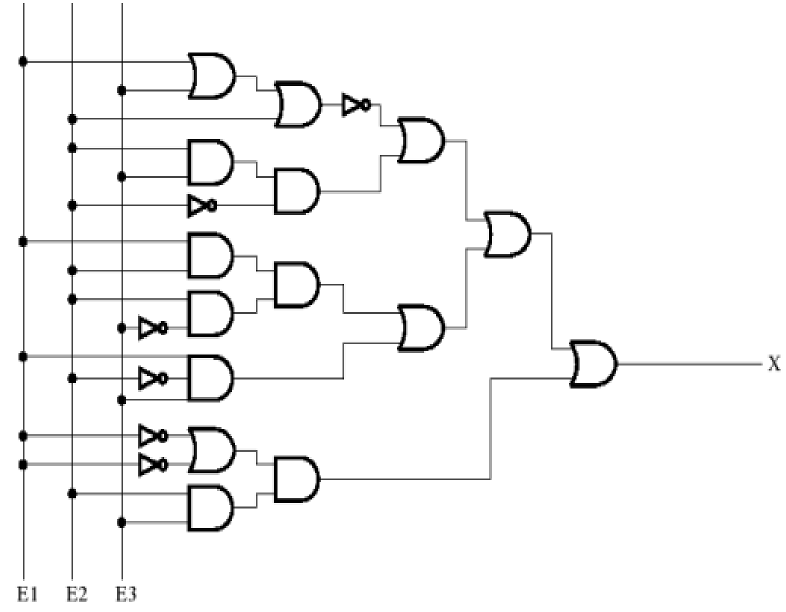
exemple

Je cherche Y à **vrai**
Y est vrai quand :
- soit A et B sont vrais
- soit B et C sont vrais

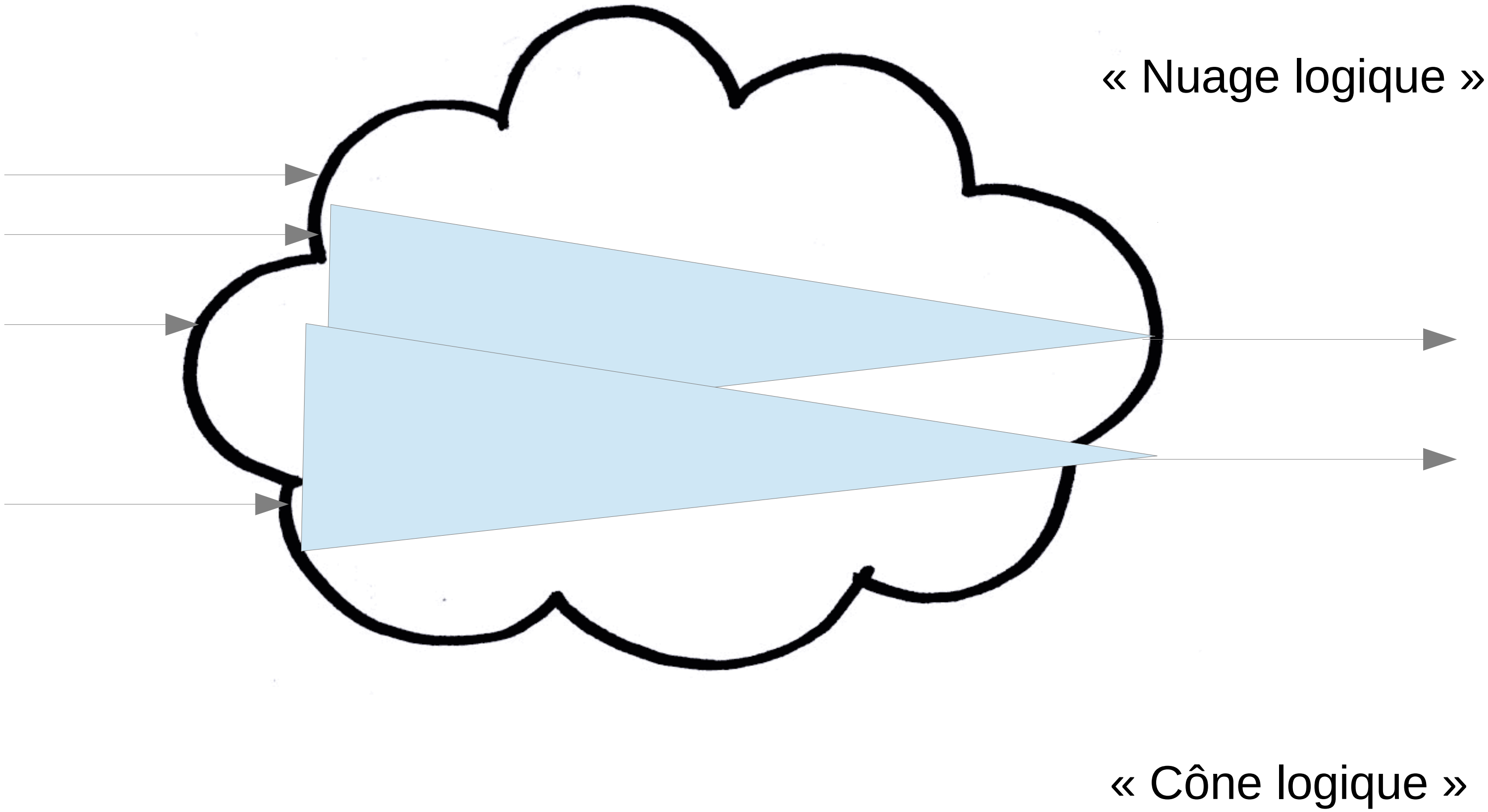
Dans tous les autres cas, Y est **faux**

Circuit combinatoire

- Implémente une fonction combinatoire
  - Au sens mathématique
  - A une variable booléenne générale d'entrée n'est associée qu'une valeur de sortie
- Pas de rebouclage des sorties sur les entrées
- Pas de mémorisation

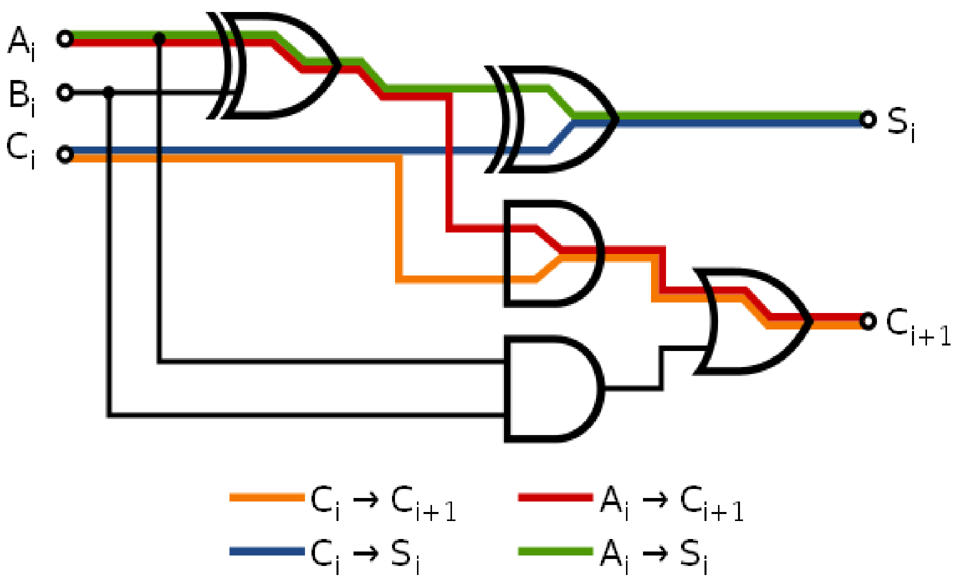


Circuit combinatoire



Chemin critique
d'un circuit combinatoire

Chaque circuit possède un temps de propagation caractéristique entre
Ses entrées et la sortie

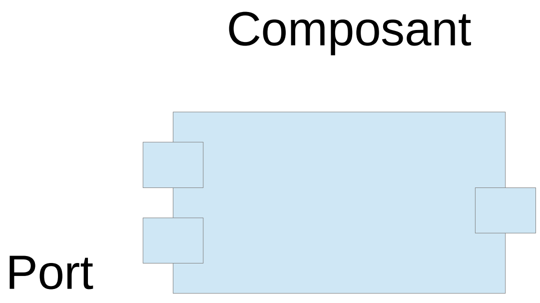


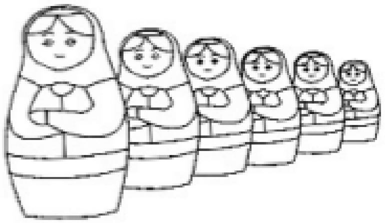
Le chemin critique correspond à la traversée la plus longue parmi tous les chemins (acycliques) possibles

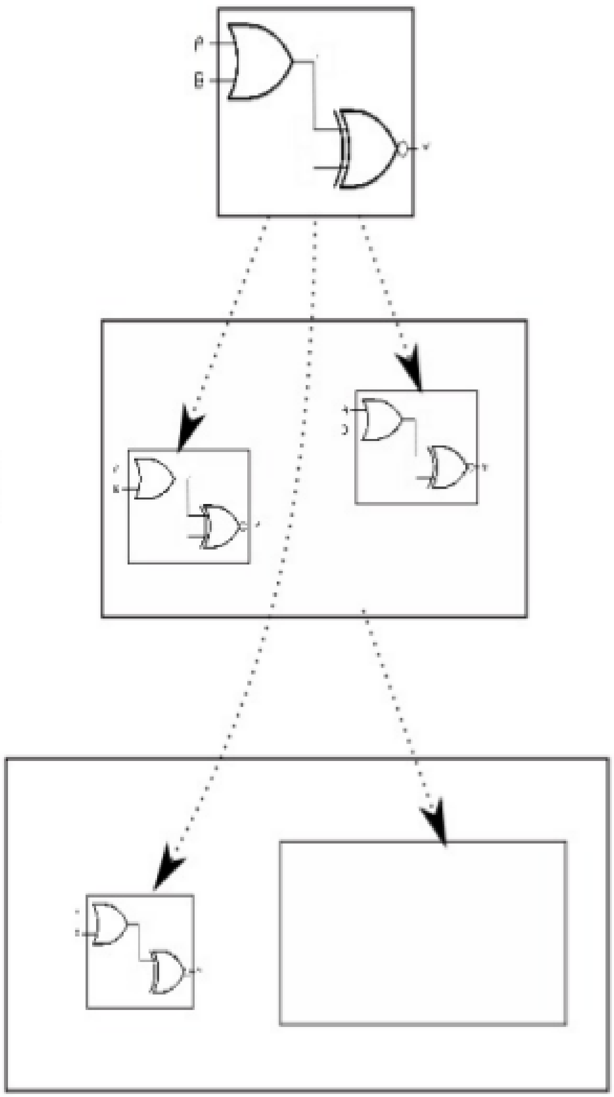
Ici : chemin en rouge pour des délais tous unitaires

(site du Zero)

Elaboration de circuits complexes
Notion de hiérarchie



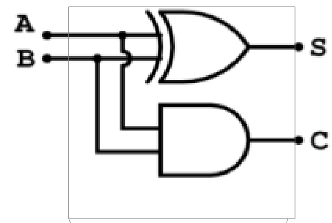




Elaboration de circuits complexes
Notion de hiérarchie

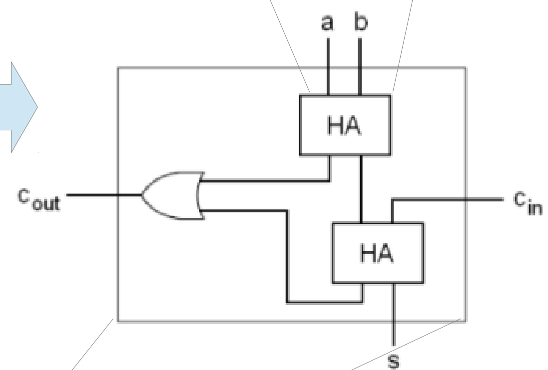
On établit la table de vérité.

a	b	f	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

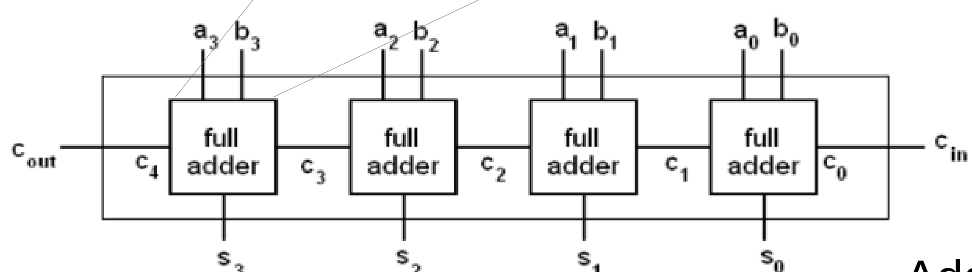


Demi-additionneur

a	b	cin	S	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Additionneur 1 bit

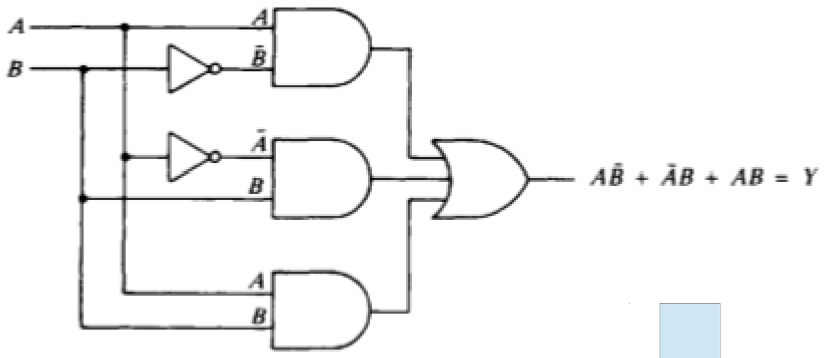


Additionneur 4 bit
(ripple carry)

Simplification des fonctions logiques

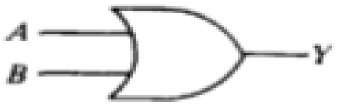
Méthode algébrique

L'expression d'un problème
peut mener à des
Formules non optimisées



Applications des axiomes
et théorèmes de Boole

Expression simplifée
Circuit plus petit, moins gourmand,...



Simplification des fonctions logiques

Méthode algébrique

- $F = (X + Y)(\overline{X} + Y)$
- $G = XY + \overline{Z} + Z (\overline{X} + \overline{Y})$
- $H = (X + Y + Z)(\overline{X} + Y + Z) + XY + YZ$

Simplification des fonctions logiques
Tableaux de Karnaugh

Table de vérité

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Somme de produits

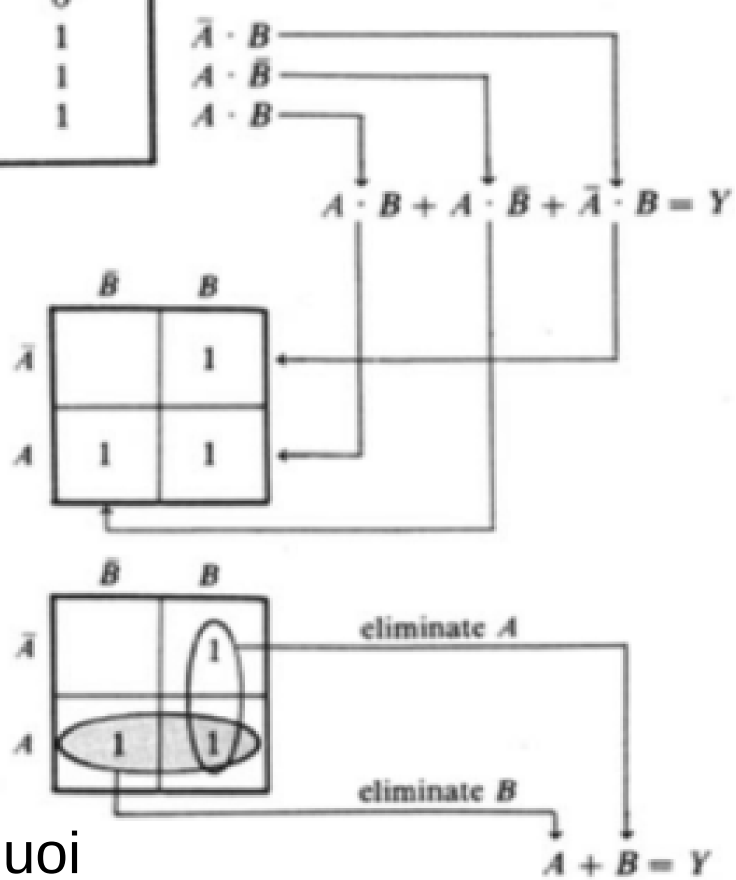


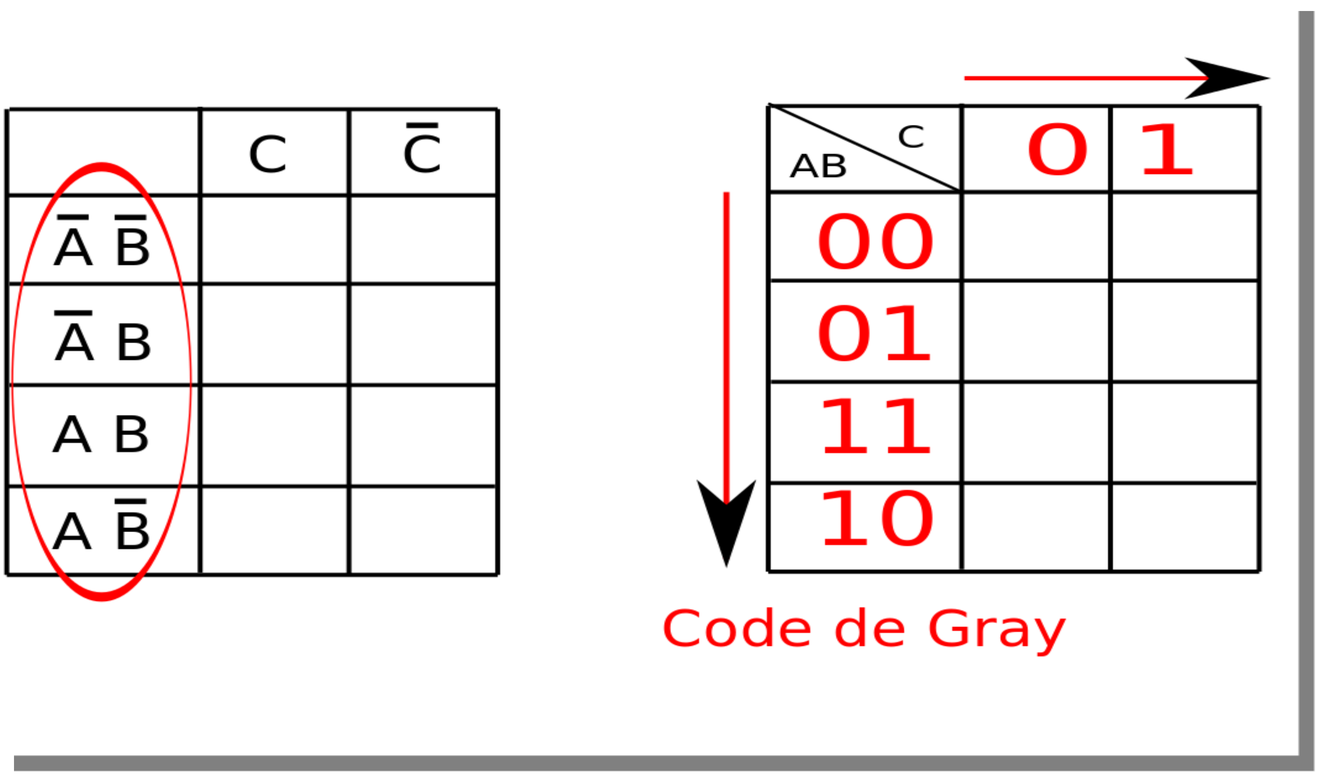
Tableau de Karnaugh
À 2 entrées

Cas simple, mais l'on voit pourquoi
la simplification marche : $B\overline{A}+BA=B(\overline{A}+A)=B$

Simplification des fonctions logiques

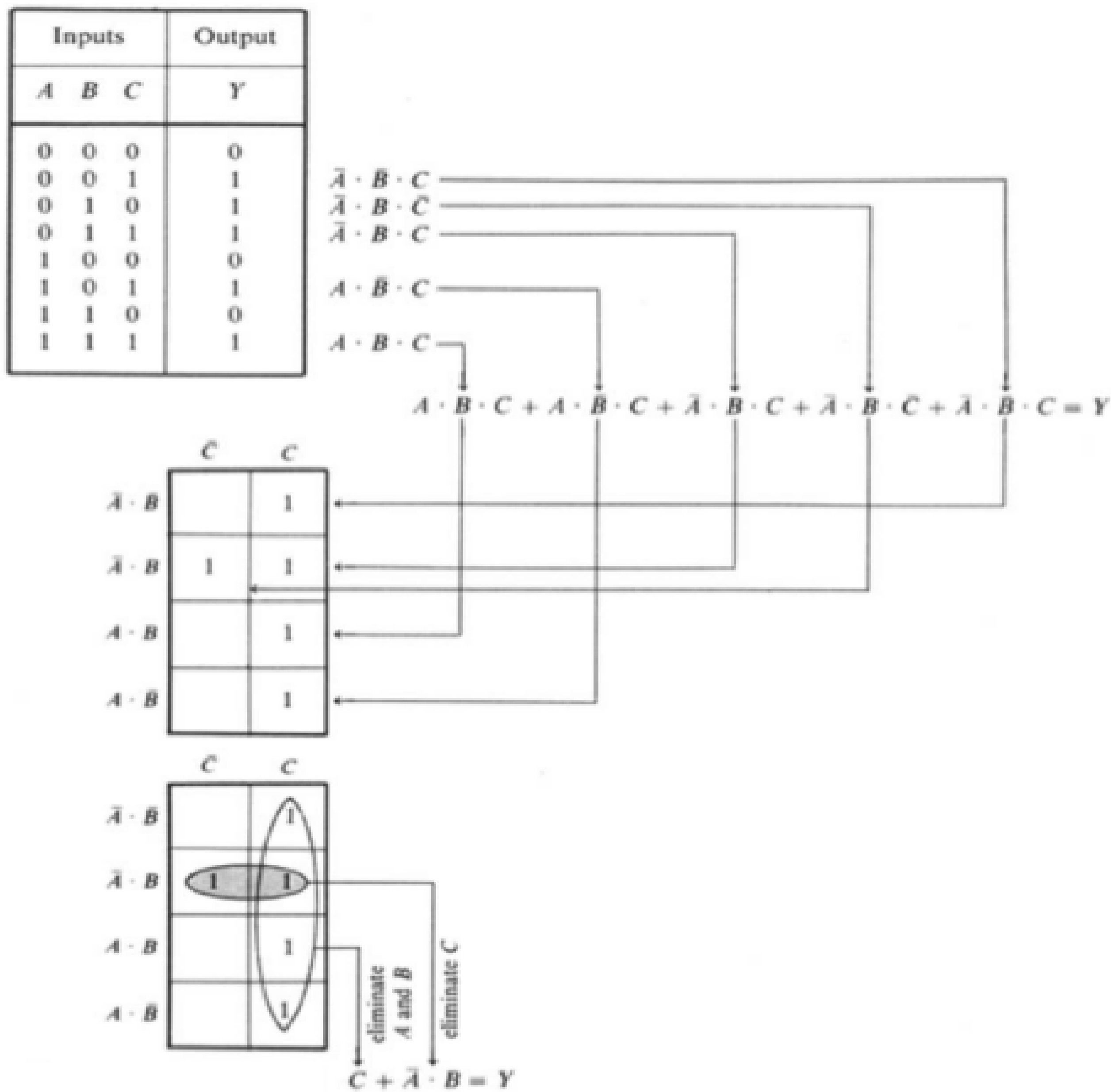
Tableaux de Karnaugh (K-map)

- Fonctionne pour n variables
  - En pratique, **n <=4** seulement
- Pour n>2 :
  - basé sur l' **adjacence des codes de Gray**
  - Ex :



Minimisation des fonctions logiques

Tableaux de Karnaugh



Tableaux de Karnaugh

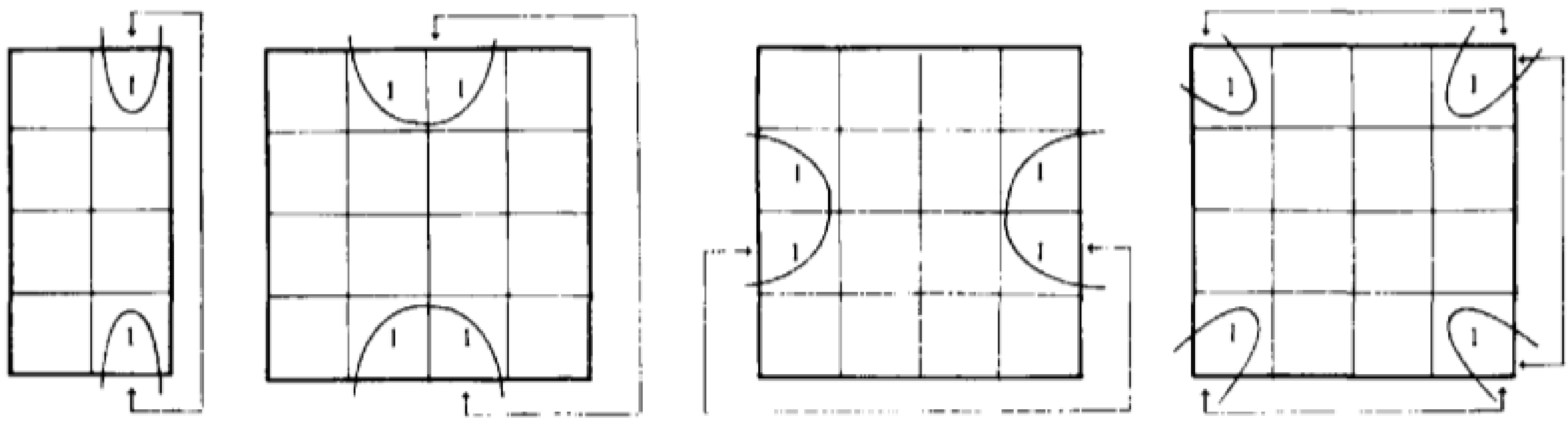
Méthode générale

- A partir de la table de vérité, établir F comme somme de produits
- Pour chacun des produits présents, placer un '1' correspondant dans une K-map
- Effectuer les regroupements de 2,4 ou 8 '1'
  - Avec recouvrements possibles
- Dans ces groupes, supprimer la ou les variables qui apparaissent avec leurs compléments, et conserver les autres
- Restituer le résultat sous forme d'une nouvelle somme

Tableaux de Karnaugh

exotiques

- Motifs de regroupements « exotiques » possibles



Cases adjacentes bien qu'éloignées

Tableaux de Karnaugh

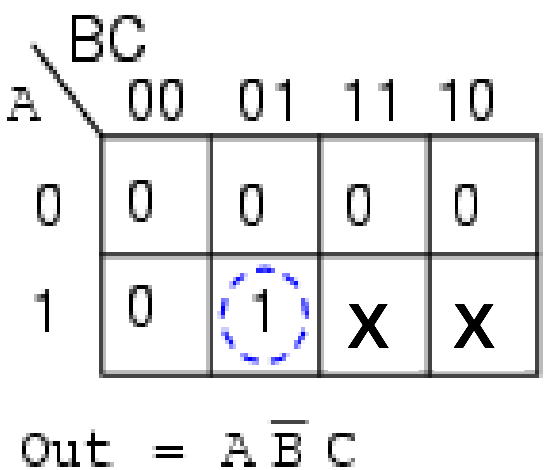
Notion de « don't care »

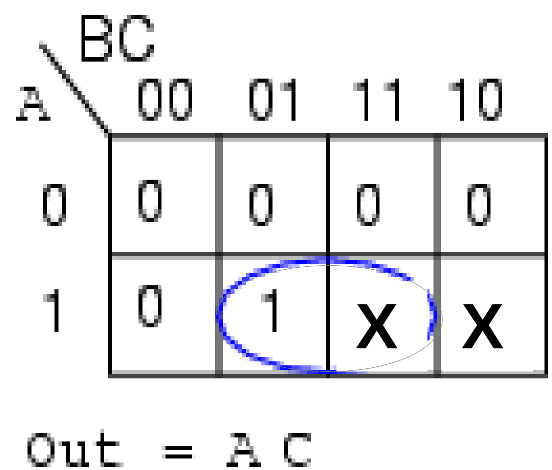
- Parfois l'énoncé d'un problème conduit à une table de vérité incomplète :
  - Pour certaines valeurs des variables en présence, le problème ne spécifie rien sur la fonction, en ce point
  - C'est la notion de « don't care »
    - Une valeur indifféremment 1 ou 0
    - Sans répercussion sur l'ensemble
- Les don't care constituent une opportunité de regroupements plus grands
  - Donc de meilleures simplifications

Tableaux de Karnaugh

Notion de « don't care »

- Exemple :

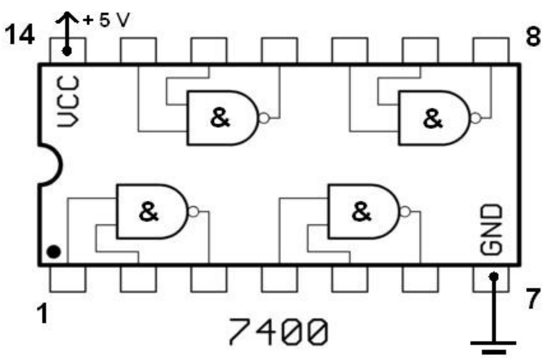


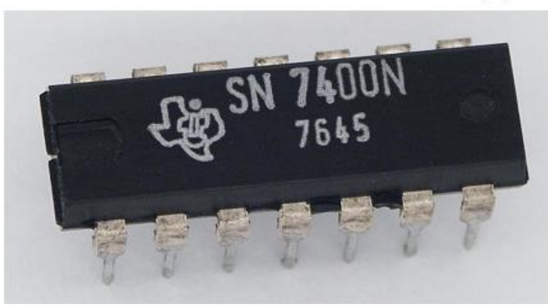


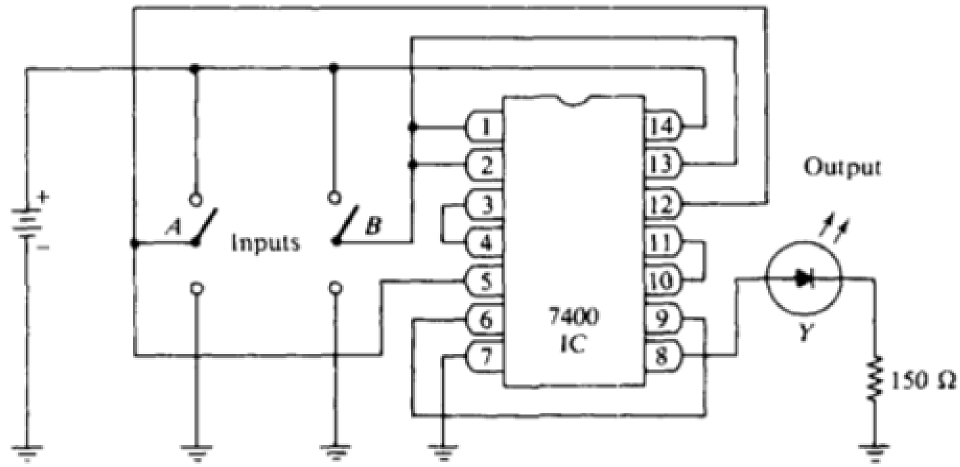
Regroupement possible

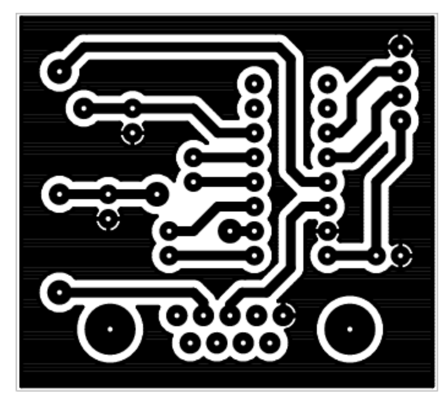
Mapping technologique
circuits discrets

Série 74XX







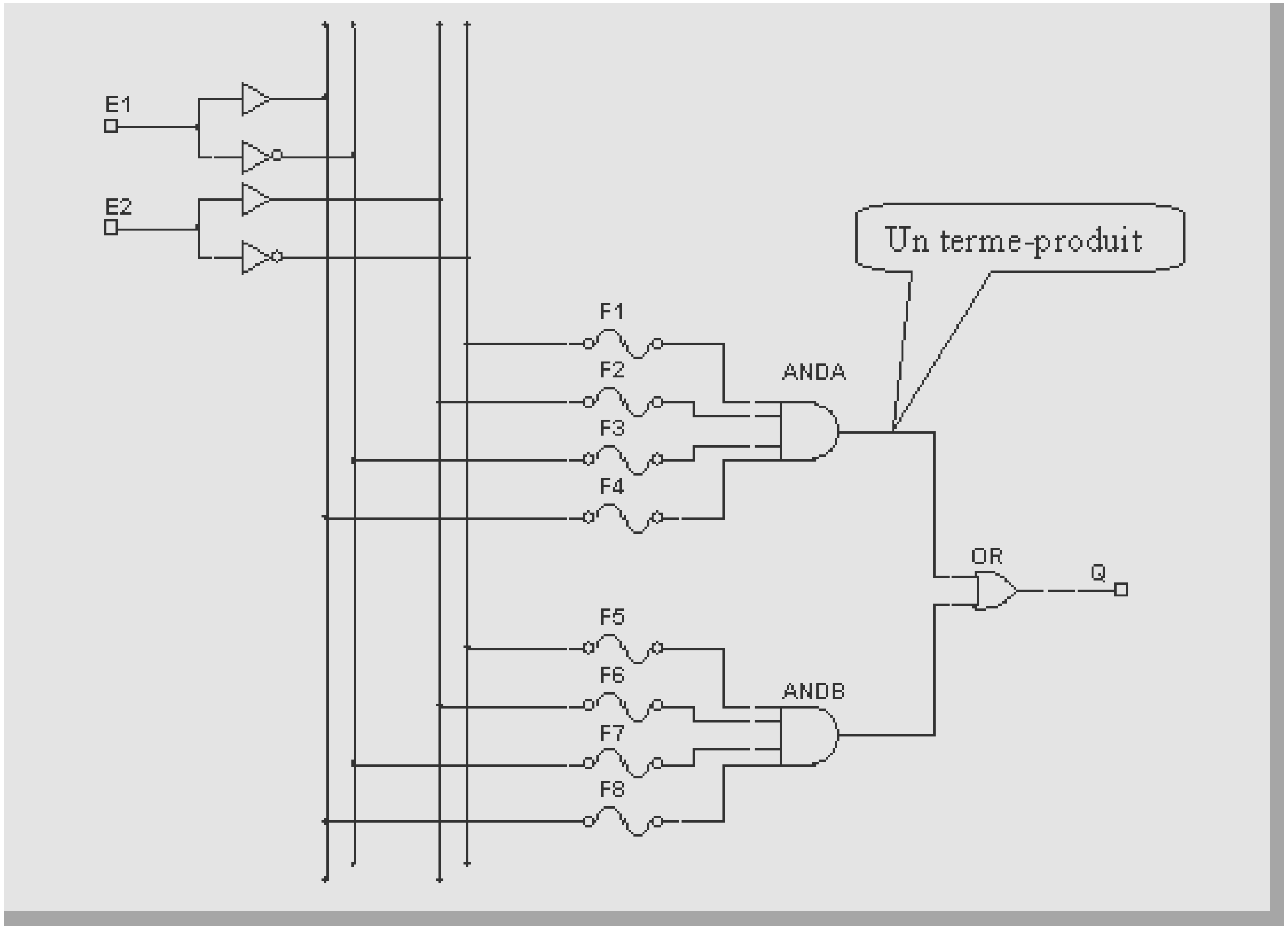


Faible capacité en #portes !!!

Mapping technologique
librairies fondeur

- Les gros circuits ne sont pas réalisés selon ces techniques programmables
  - Les circuits sont réalisés dans le silicium de manière dédiée et figée.
  - Chaque fondeur possède ses bibliothèques de circuits élémentaires (portes) qu'il sait synthétiser sur ce silicium
  - Certaines de vos portes complexes (ex : AND à 5 entrées) peuvent ne pas exister dans cette librairie
  - Il faut donc transformer votre formule dans technologie du vendeur

Mapping technologique
SPLD : simple programmable logic device



Mapping technologique
SPLD

Equation de la PAL 2 entrées 1 sortie vierge

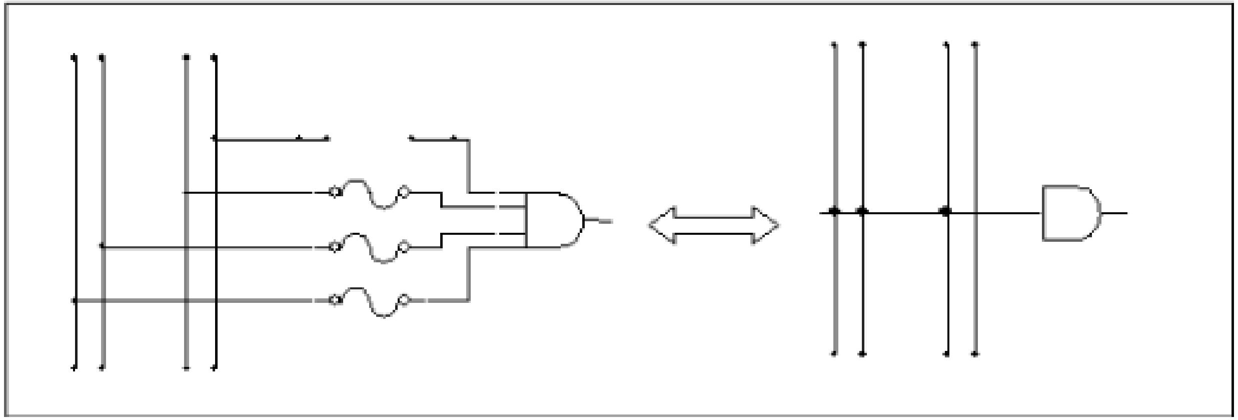
$$Q = E1.\overline{E1}.E2.\overline{E2}. + E1.\overline{E1}.E2.\overline{E2}$$

Un fusible « grillé » ramène un niveau logique haut.

Exemple
pour faire un un « ou exclusif »

$$Q = \overline{E2}.E1 + E2.\overline{E1}$$

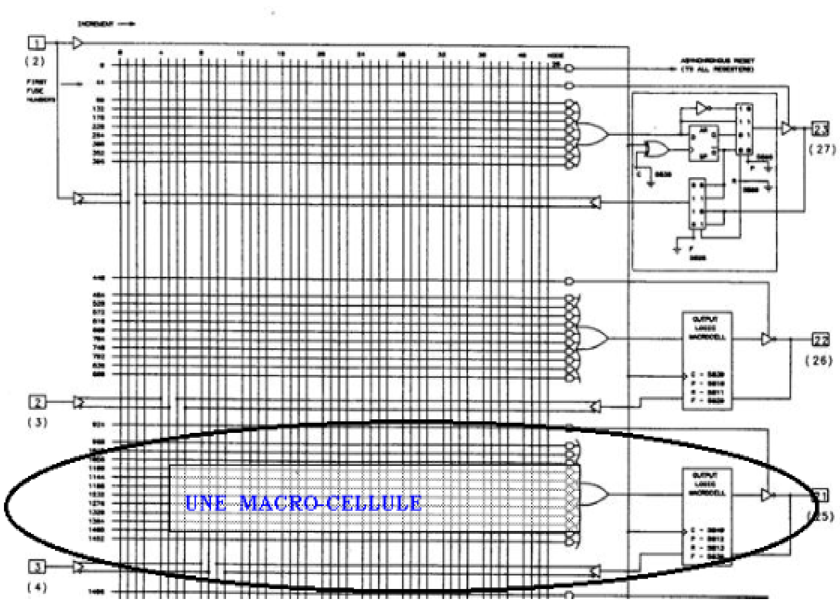
on grille F2,F3,F5,F8
représentation simplifiée



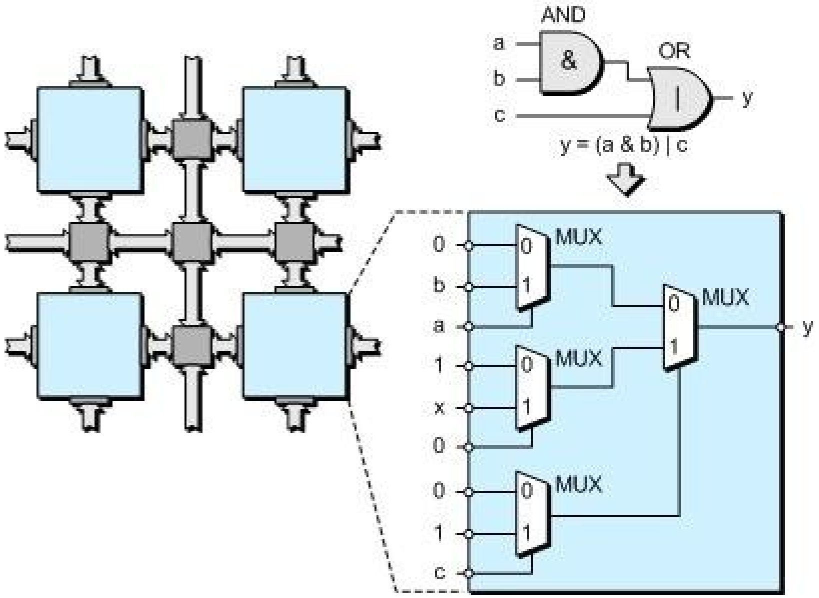
Mapping technologique

Vers des composants programmables de + en + complexes

Extrait du schéma interne d'un SPLD 22V10

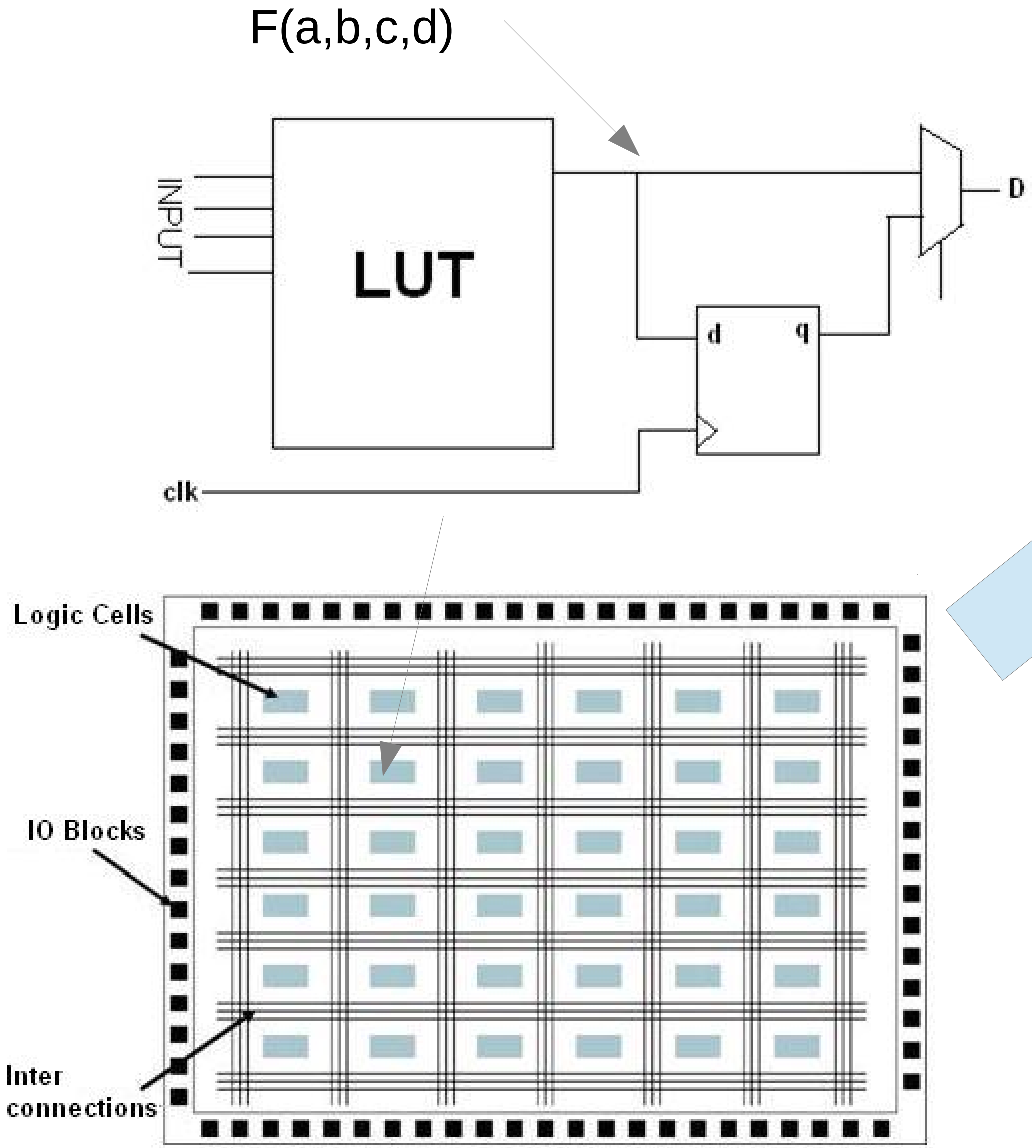


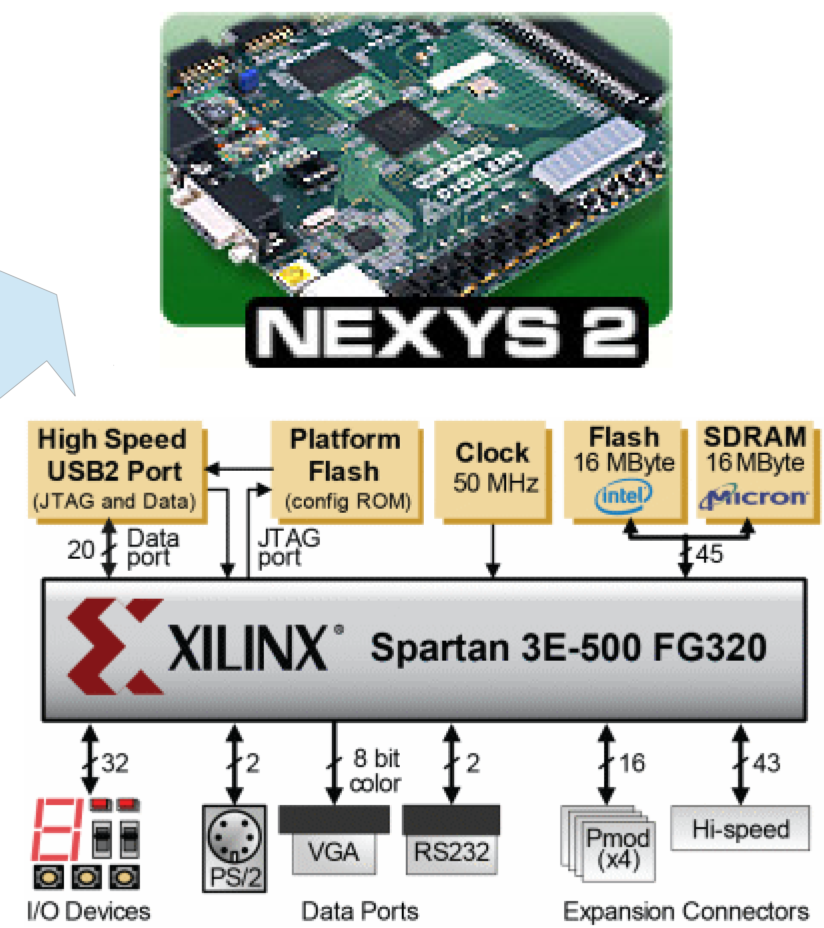
Exemple du P22V10



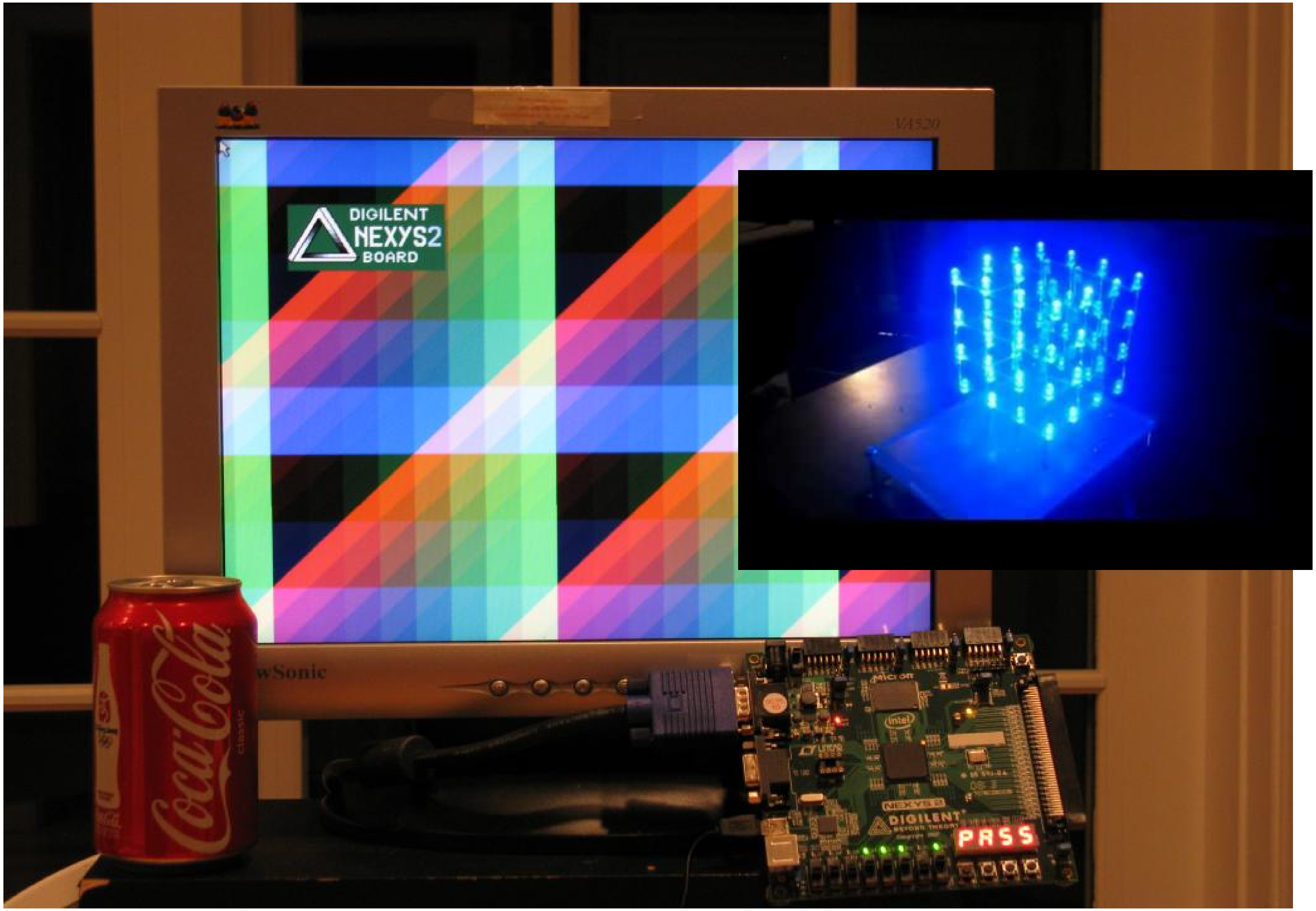
Mapping technologique

FPGA





Have Fun !



Ce qu'il faut retenir

- Table de vérité, équation, circuits combinatoires
- Minimisation
  - Règles algébriques
  - Tables de Karnaugh
- Chemin critique