ITI 1500 **Hiver 2013**

Systèmes Numériques I

Cours

Lundi	11:30 - 13:00	SCS E217
Jeudi	13:00 - 14:30	SCS E217

TUTORIAL Mardi 8 :30-10:00 salle: LMX-221

Professeur : Dr A. Karmouch, Bureau: CBY-A508

Chapitre 2

L’algèbre de Boole

et

les Portes Logiques

Logique Binaire

• La logique binaire comporte

1 – des **Variables** qui peuvent prendre **deux valeurs**

→*les valeurs peut être appelée **Vrai**, **Faux**, **Oui**, **non**, etc.*

2 – des **Opérations** qui ont une signification logique

→ *La logique binaire est l'**algèbre** de **Boole***

Algèbre de Boole

• Bases mathématiques nécessaire pour la description des circuits numériques

• Utilisée pour la description des différentes interconnections de circuits numériques

• Les variables utilisées dans l'algèbre de Boole sont dites **variables booléennes**

→ Nous allons étudier de manière succincte l'algèbre de Boole et son utilisation dans la simplification des fonctions logiques

Algèbre de Boole

- **Composée de**

1- Variables Booléenne

- représentées par des lettres de l'alphabet comme A, B, C, x, y, z etc.

- chaque variable peut avoir deux valeurs distinctes

1 et 0 (Vrai, Faux)

- peut être une fonction de certains variables booléenne (F=ABC)

2- Opération Booléenne

-Il y a trois opérations logiques de base: ET, OU, et NON

Opération logiques - ET

- Représentée par un point (.) ou par l'absence de l'opérateur

Exemple: $x.y = or \quad xy=z$

lire: *x ET y est égale a z*

Interprétation: $Z = 1$ *si et seulement si* $x= 1$ *ET* $y= 1$

Sinon $z = 0$

table de vérité :

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Ne pas confondre l'opération de multiplication binaire

Table vérité donne les valeurs de z pour toutes les valeurs possibles de x et de y

Opération- OU

- *Représentée par le signe plus (+)*

Exemple: $x + y = z$

lire: *x OU y est égale à z*

Interprétation: *z = 1 si x= 1 ou si y= 1 ou si x =1 et y =1.*
z = 0 si x = 0 et y=0

Table vérité :

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Ne pas confondre avec l'opération d'addition binaire

Table vérité donne les valeurs de z pour toutes les valeurs possibles de x et de y

Opération- *NON*

- *Représentée par un prime (') ou une barre (complément)*

Exemple: $x' = z$ *(or* $\bar{x} = z$*)*

lire: NON x est égale a z

Interprétation: $z =$ *"l'inverse de x "*

$x = 1$ *alors* $z=0$; $x = 0$ *alors* $z=1$

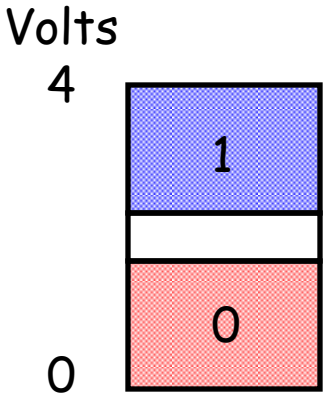
table Vérité:

x	x'
0	1
1	0

Table vérité donne les valeurs de z pour toutes les valeurs possibles de x

Les Signaux Binaires

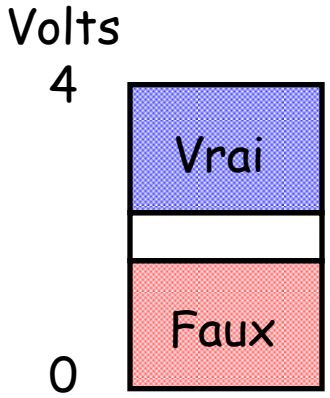
- Les ordinateurs utilisent des tension électriques pour représenter l' information.
- Deux niveaux de tenson électrique sont utilisés pour représenter les valeurs binaires "1" and "0"
- Certain systèmes par exemple peuvent définir que:
  - Binaire '0" est égale 0 Volt
  - Binaire "1" est égale 4 Volt



Binaire logique et signaux binaires

• il est possible de penser au voltage comme représentant deux valeurs logiques ***Vrai*** et ***Faux***.

→ Ces valeurs logiques sont dites valeurs booléennes.



Binaire logique et signaux binaires

- Pour des raisons de simplification on continue a utiliser:
  - 1 pour Vrai
  - 0 pour Faux
- Nous utiliserons cette interprétation combinée avec des opérations spéciales pour concevoir les fonctions logiques et les circuits associés

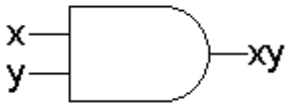
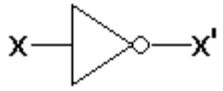
Portes Logiques Fondamentales

• *Les portes logiques sont des* circuits électroniques qui fonctionnent sur l'arrivée d'un ou plusieurs signaux d'entrées pour produire un signale en sortie

• les opérations fondamentales peuvent être implémentées en utilisant les portes logiques

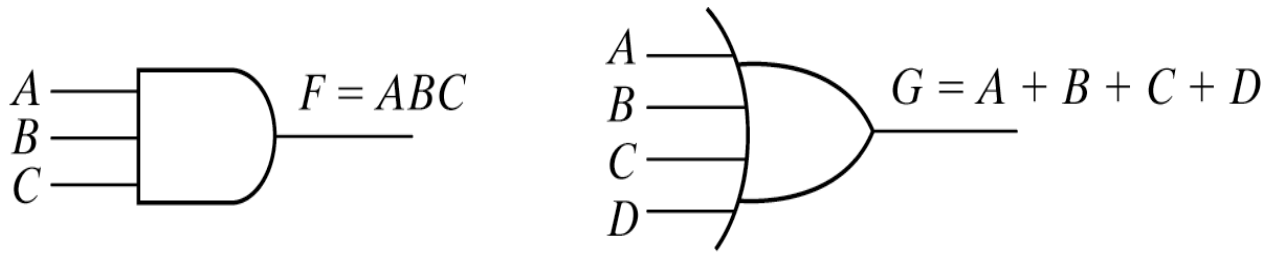
– Symbole pour chaque porte logique est donné ci-dessous

– ces portes donne en sortie le produit, la somme ou le complément de leurs entrées.

Opération logique:	Et (produit) De deux entrées	OU (somme) de deux entrées	NON (complément) avec une entrée
Représentation:	x.y, ou xy	x + y	x'
Porte logique:			

Portes logique avec plusieurs entrée

- les Portes ET et OU peuvent avoir plusieurs entrées



(a) Three-input AND gate (b) Four-input OR gate

Fig. 1-6 Gates with multiple inputs

Portes Logique - Signaux

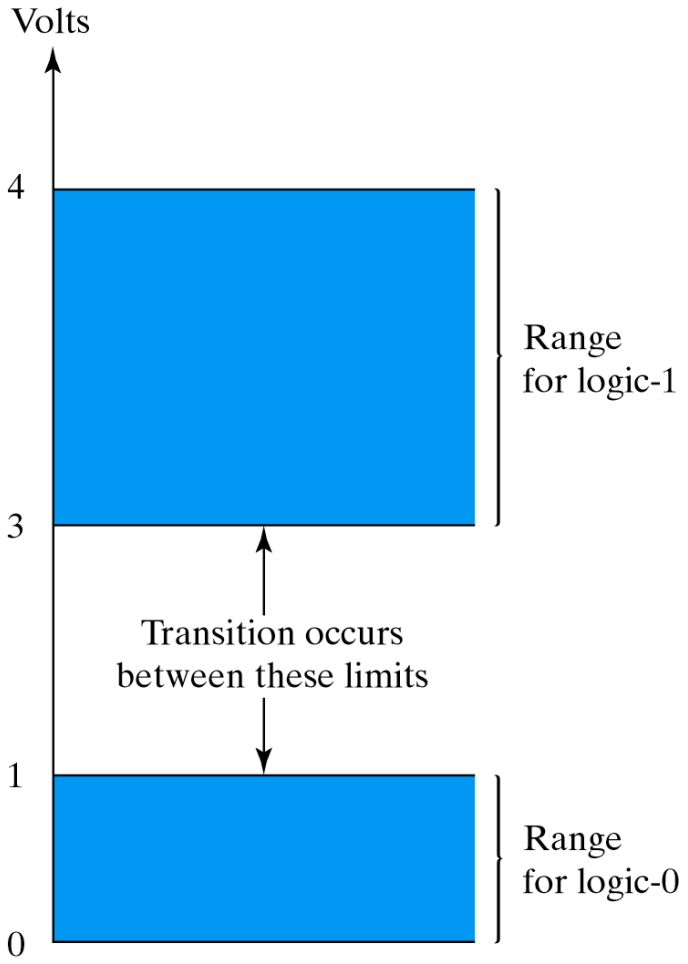
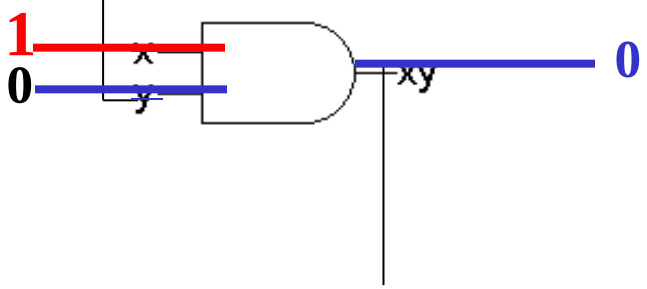


Fig. 1-3 Example of binary signals

Exemple

2 signaux d'entrée

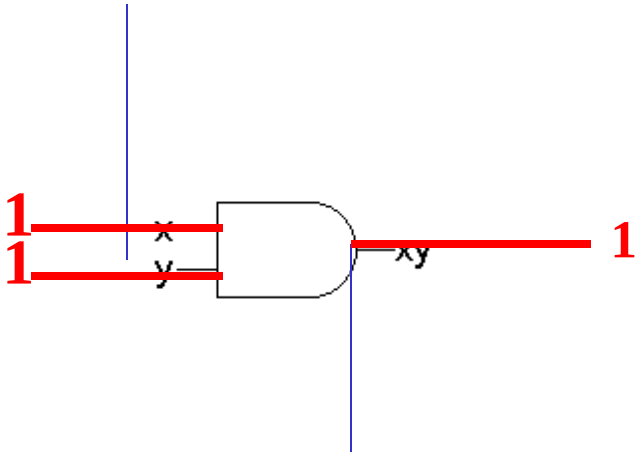


1 signal en sortie

Portes Logique - Signaux

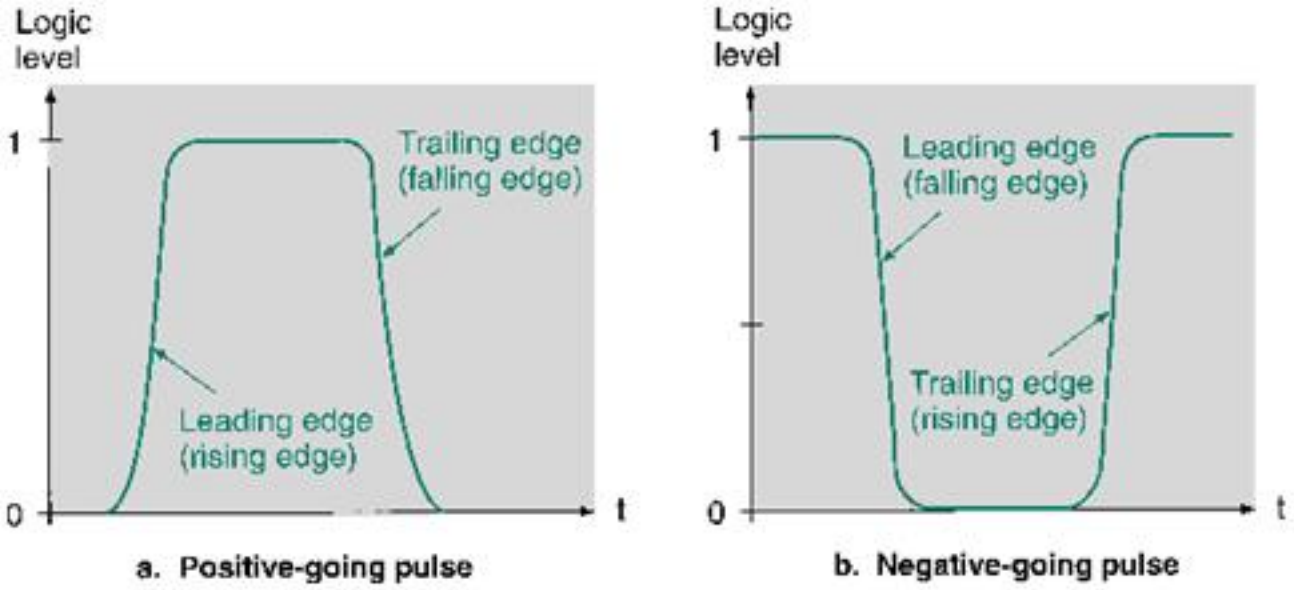
Exemple

2 signaux d'entrée



1 signal en sortie

Timing Diagram –Signaux d'entrée et de sortie



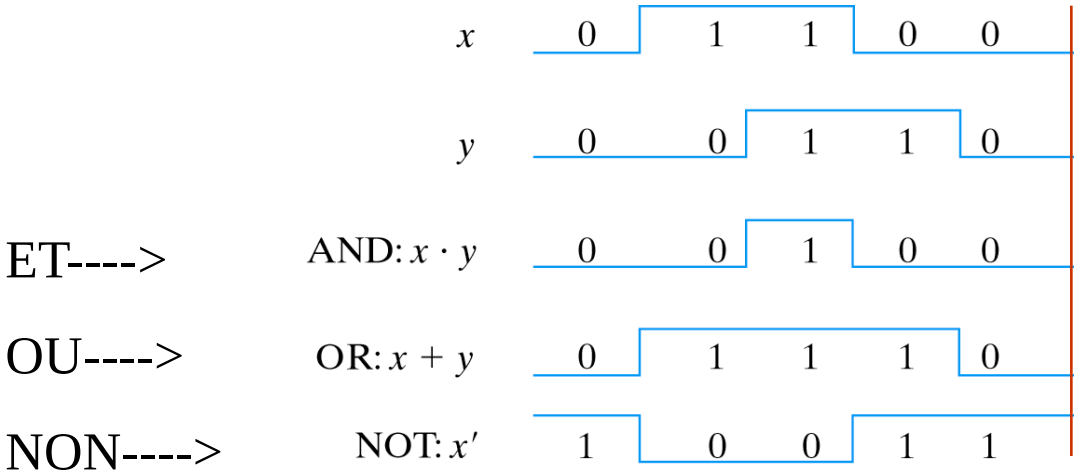


Fig. 1-5 Input-output signals for gates

Expressions Booléennes (fonctions)

- On peut utiliser les opérations de base pour former des expressions plus complexes:

$$f(x,y,z) = x\,y' + z\,x'$$

- **Quelques terminologie et notation:**
  - **f** est le nom de la fonction.
  - ***Terme*** est une implémentation avec une porte logique: dans cette exemple f possède deux termes $x\,y'$ et $z\,x$
  - ***(x,y,z)*** sont ***les variables d'entrées***, représentent chacune un 1 ou un 0.
  - *un* ***littéral*** une occurrence d'une variable d'entrée ou son complément. La fonction f possède 4 littéraux : x, y', z, et x'.

Évaluation des opérateurs logiques

- La priorité des opérateurs est la suivante.

  – Parenthèses d'abord, ensuite

  NON , suivi de ET, et enfin OU.

  → f(x,y,z) = (x + y')z + x'

  –Une expression complètement avec parenthèses n'est pas commode:

  f(x,y,z) = (((x +(y'))z) + x')

Table de Vérité

- La table de vérité représente toutes les valeurs des entrées et des sorties d'une fonction .
- Chaque entrée représente soit 1 soit 0.
  - Une fonction à n variables possède 2^n combinaisons possibles des entrées
- Les entrées sont listées dans l'ordre binaire.-Exemple, de 000 a 111.

f(x,y,z) = (x + y')z + x'

f(0,0,0) = (0 + 1)0 + 1 = 1
f(0,0,1) = (0 + 1)1 + 1 = 1
f(0,1,0) = (0 + 0)0 + 1 = 1
f(0,1,1) = (0 + 0)1 + 1 = 1
f(1,0,0) = (1 + 1)0 + 0 = 0
f(1,0,1) = (1 + 1)1 + 0 = 1
f(1,1,0) = (1 + 0)0 + 0 = 0
f(1,1,1) = (1 + 0)1 + 0 = 1

x	y	z	f(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

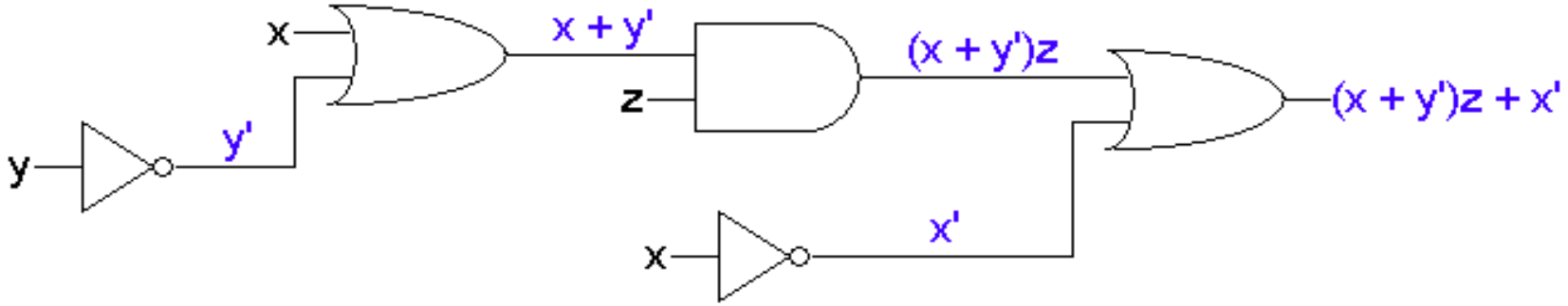
Expression Booléenne et Circuits Logique

- *Une expression* Booléen (fonction) peut être traduite sous forme de circuits logiques en utilisant les portes logiques fondamentales.
- Exemple:

  - Le diagramme ci-dessous montre les entrées et les sorties pour chaque porte

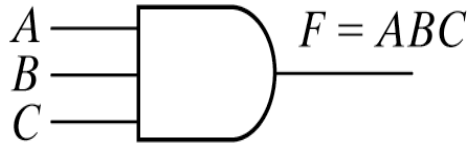
  - La priorités sont explicites dans le circuit.

f(x,y,z) = (x + y')z + x'

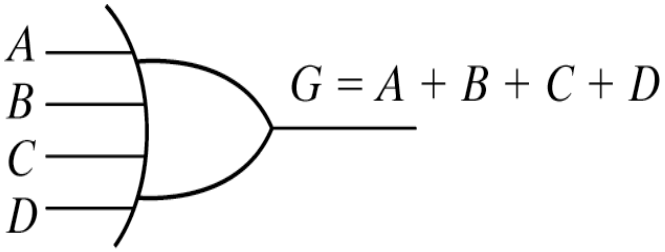


Portes avec plusieurs entrées

- les Portes ET et OU peuvent avoir plus de 2 entrées



(a) Three-input AND gate



(b) Four-input OR gate

Obtention de l'expression Booléenne

• **Une expression booléenne peut être obtenu a partir de: :**

- D'un texte descriptif
- Table de vérité;
- Circuit logique.

Obtenir l'expression booléenne à partir de la table de vérité

- Expression booléenne (NON simplifiée) peut être obtenue a partir de la table vérité.

A	B	C	F_1	
0	0	0	0	
0	0	1	0	
0	1	0	1	*A'BC'*
0	1	1	1	*A'BC*
1	0	0	1	*AB'C'*
1	0	1	1	*AB'C*
1	1	0	1	*ABC'*
1	1	1	1	*ABC*

on peut aussi écrire la fonction comme:

F_1 (A,B,C) = A'BC'+ A'BC+ AB'C'+ AB'C+ ABC'+ ABC

Obtenir L’expression Booléenne

Utilisation des termes qui donnent une sortie fausse

• Il est parfois plus facile de travailler avec les termes qui représente le cas ou la fonction est fausse

• Par exemple, si la fonction possède quatre variables et que parmi les seize possibilités, il y a treize qui qui donne la sortie vrai et trois qui donne la sortie fausse alors il est plus facile de travailler avec la fonction qui a moins de termes.

• Dans notre exemple nous avons deux états sur huit pour lesquelles la fonction est fausse.

Obtenir L'expression Booléenne

- L' expression booléenne (non simplifiée) peut être obtenue de la table de vérité en utilisant les termes pour lesquelles la fonction est fausse

A	B	C	F	
0	0	0	0	*A'B'C'*
0	0	1	0	*A'B'C*
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	

on peut ainsi écrire la fonction:

F' (A,B,C) = A'B'C' + A'B'C

Obtenir L'expression Booléenne

Exemple: concevoir un circuit logique pour être utilisé pour contrôler la sonnerie d'une alarme qui va être installée dans une salle afin de la protéger des entrées non autorisées. Les détecteurs installées dans la salle fournissent les signaux logiques suivant

C = 1 → le système de contrôle est actif

D = 1 → la porte de la salle est fermée

M = 1 → il y a un mouvement dans la salle

Q = 1 → la salle est ouverte aux publique

1- construire la table de vérité

2- obtenir la fonction logique

3- dessiner le circuit logique correspondant

Table de table

C	D	M	Q	Alarme
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Alarme= CD'M'Q' + CD'MQ' + CDMQ'

Les identités de Boole

• L'algèbre de Boole est utilisée dans la conception des circuits logiques pour réduire toute fonction (expression) logique à sa forme la plus simple.

→ minimiser le nombre des littéraux et le nombre de termes

→ un circuit avec moins d' équipement

• C'est un problème difficile compte tenu du fait qu'il n' y a pas de règles systématique à suivre pour réduire une fonction logique

Identités de Boole

1.	$x + 0 = x$	
2.	$x \cdot 1 = x$	
3.	$x + x' = 1$	
4.	$x \cdot x' = 0$	
5.	$x + x = x$	identités de base de l'algèbre de Boole
6.	$x \cdot x = x$	
7.	$x + 1 = 1$	
8.	$x \cdot 0 = 0$	
9.	$(x')' = x$	
10.	$x + y = y + x$	Commutative
11.	$xy = yx$	Commutative
12.	$x + (y + z) = (x + y) + z$	Associative
13.	$x(yz) = (xy)z$	Associative
14.	$x(y + z) = xy + xz$	Distributive
15.	$x + yz = (x+y)(x+z)$	Associative
16.	$(x + y)' = x' y'$	DeMorgan
17.	$(xy)' = x' + y'$	DeMorgan
18.	$x + xy = x$	Absorption
19.	$x(x + y) = x$	Absorption

Vérification Identités de Boole-Exemples

Théorème : $x+x = x$

$$x+x = (x+x)\,1$$
$$= (x+x)\,(x+x')$$
$$= x+xx'$$
$$= x+0$$
$$= x$$

x.1=x

x+x’=1

x + yz = (x+y)(x+z)

x.x’=0

x+0=x

Théorème : x x = x

$$xx = x\,x + 0$$
$$= xx + xx'$$
$$= x\,(x + x')$$
$$= x\,1$$
$$= x$$

Vérification des Identités de Boole-Exemples

- Théorème DeMorgan
  - -- $(x+y)' = x' y'$
  - -- $(x\, y)' = x' + y'$
- Méthode de la table de vérité

x	y	x+y	(x+y)‘	x‘	y‘	x‘y’
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

Vérification des Identités de Boole-Exemples

- Théorème $x + xy = x$

- Par la méthode de la table de vérité

x	y	xy	x + xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Simplification des Expressions Booléennes

→ **En utilisant les identités de Boole**

on peut simplifier la fonction comme suit:

$F_1(A,B,C) = A'BC' + A'BC + AB'C' + AB'C + ABC' + ABC$

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

$= A'B + AB' + AB$

$= A'B + A(B'+B)$

$= A + A'B$

Simplification des Expressions booléennes

Fonction à quatre variables

- Étant donne la fonction suivante:

$$F_{2a}(A,B,C,D) = (AB'(C + BD) + A'B')C$$
$$= (AB'C + \mathbf{AB'BD} + A'B')C$$
$$= (AB'C + \mathbf{A0D} + A'B')C$$
$$= (AB'C + \mathbf{0} + A'B')C$$
$$= (AB'C + A'B')C$$
$$= AB'\mathbf{CC} + A'B'C$$
$$= AB'C + A'B'C$$
$$= (\mathbf{A+A'})B'C$$
$$= B'C$$

$$\boldsymbol{F_{2b}(A,B,C,D)} = B'C$$

→ Les deux expressions sont équivalente!

→ **F_{2a} Nécessite plus de portes logiques que F_{2b}**

Les autres portes logiques

- **Portes logiques fondamentales**

  - ET
  - OU
  - NON

  Sont appelée fondamentales car elles peuvent être utilisées pour réaliser les « autres portes logiques » ainsi que tout Circuit numérique

- **Autres portes logiques**

  - NON-ET
  - NON-OU

  Elle sont adites universelle car tout circuit numérique peut être réalise uniquement avec ces portes

  - OU Exclusif
  - NON-OU Exclusif

Portes NON-ET , NON-OU

• Peuvent être utilisées pour réaliser toutes les opérations logiques de base: ET, OU & NON

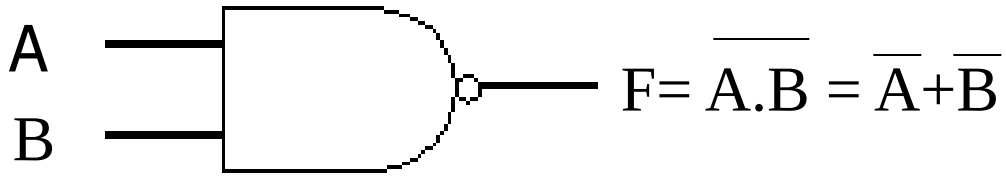
→ Elle sont dites être *fonctionnellement complètes*

• Il est plus simple de construire des circuits avec des portes NON-ET et NON-OU au lieu de combiner les portes ET, OU, NON

• Typiquement les NON-ET, NON-OU sont plus rapides et plus économiques à produire

La porte NON-ET

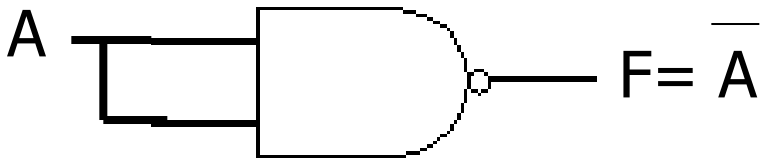
- La porte NON-ET est une combinaison de la porte ET et l'inverseur (porte NON)
- Peuvent être utilisées pour réaliser toutes les opérations logiques de base: ET, OU & NON
- Cette porte est *fonctionnellement complète*.



A	**B**	**F**
0	0	1
0	1	1
1	0	1
1	1	0

La porte NON-ET

→ Équivalent de la porte NON avec NON-ET

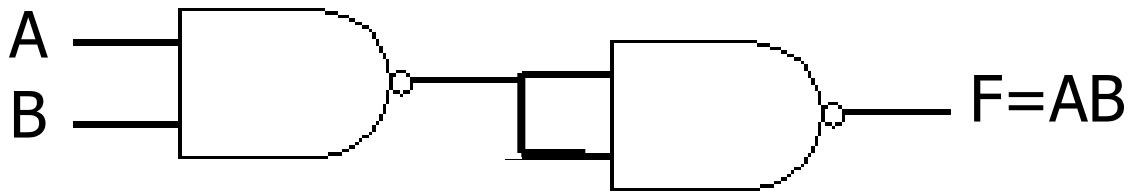


Porte NON

→ Équivalent de la porte ET avec NON-ET

((AB)'(AB)')'= (AB)''+(AB)''=
AB +AB = AB

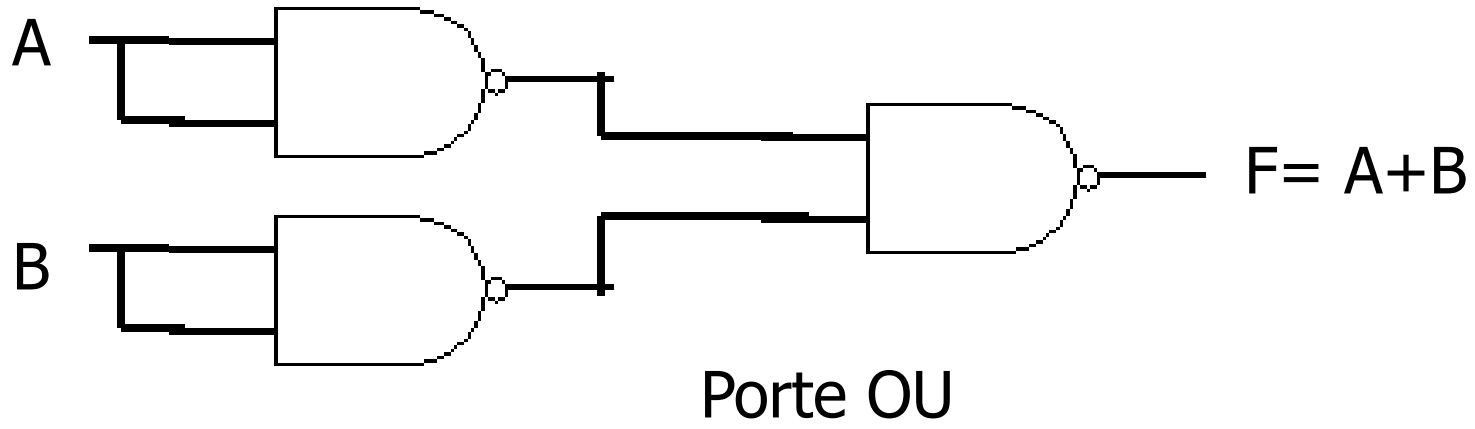
(AB)'



Porte ET

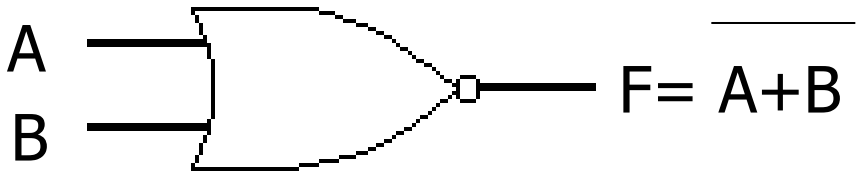
La porte NON-ET

- Équivalent de la porte OU avec NON-ET



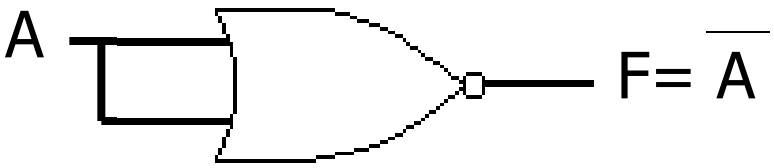
La porte NON-OU

- La porte NON-OU est une combinaison de la porte OU et l'inverseur (porte NON)
- Peuvent être utilisées pour réaliser toutes les opérations logiques de base: ET, OU & NON
- Cette porte est *fonctionnellement complète*.

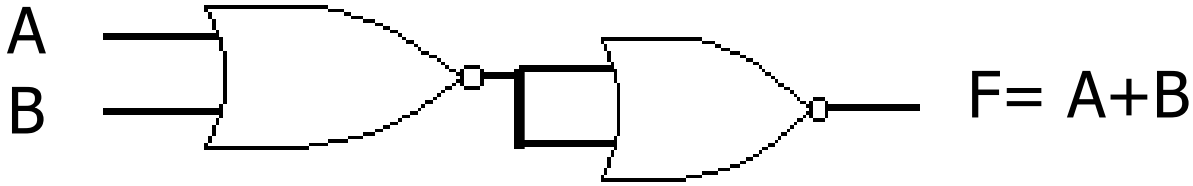


A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

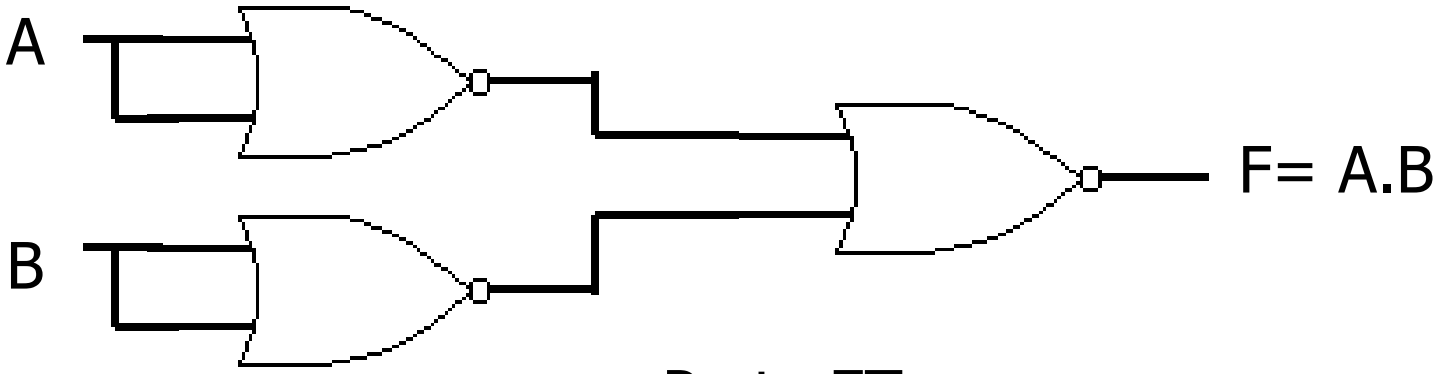
La porte NON-OU est Fonctionnellement Complète



Porte NON

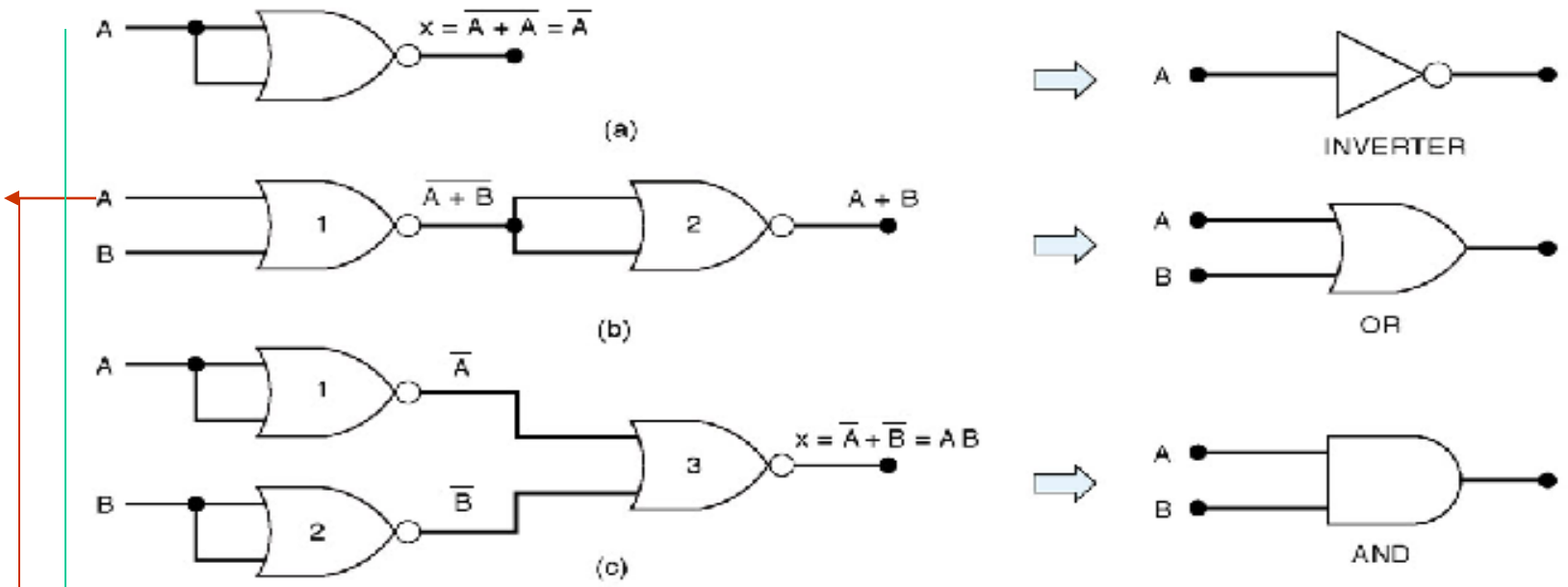


Porte OU



Porte ET

La porte NON-OU est Fonctionnellement Complète (2)



- Equivalent representations des portes ET, OU, et NON

(A+A)' = A'A' = A'

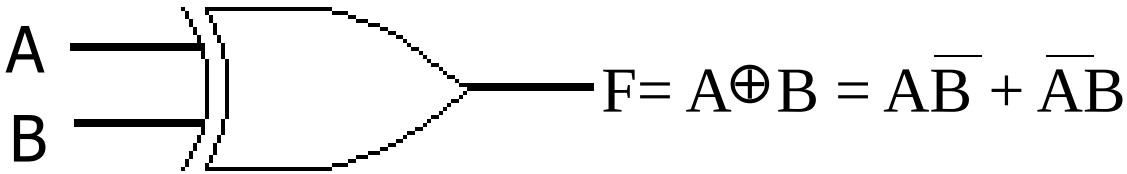
((A+B)' + (A + B)')' = (A+B)"(A+B)" = (A'B')' (A'B')'

= (A"+B") (A"+B") = (A+B) (A+B) = (A+B)

La porte OU-Exclusif

- Ceci est une porte OU-Exclusif.
- La sortie est 1 uniquement quand l'une des entrées est a 1
- Le symbole de l'opération logique est $\oplus$

$1 \oplus 1 = 0$ and $1 \oplus 0 = 1$.

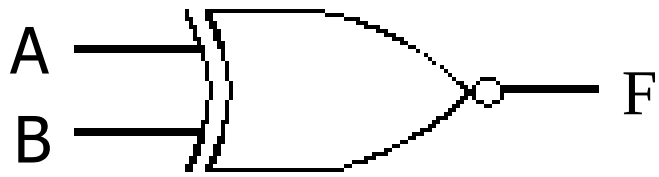


A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

La porte NON-OU exclusif

- C est le complément de la porte NON-OU exclusif
- Le symbole de l'opération logique est ⊙

  1 ⊙ 1 = 1 ET 1 ⊙ 0 = 0.



$$F = \overline{A \oplus B} = (AB) + (\overline{A} \cdot \overline{B}) = AB + A'B'$$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Les formes Standards

- Nous avons vu comment interpréter la table de vérité, obtenir la fonction Booléenne et construire le circuit logique.

- Nous avons simplifié la fonction Booléenne en utilisant les identités de Boole..

→Il y a une manière standard d'écrire la fonction (ou expression) Booléenne:

- **La forme Somme -de -Produits**
- **La forme Produit-de-Sommes**

Forme standard Somme-de-Produits- **Minterms**

• Un Minterm est le produit logique ET de variables dans lequel une variable apparaît une seule fois.

• chaque Minterm représente exactement une combinaison (ligne de la table de vérité.

• n variables donne 2^n Minterms.

Table de vérité

Decimal value	A	B	C	F	Minterm	
0	0	0	0	0	m_0	A'B'C'
1	0	0	1	0	m_1	A'B'C
2	0	1	0	1	m_2	A'BC'
3	0	1	1	1	m_3	A'BC
4	1	0	0	1	m_4	AB'C'
5	1	0	1	1	m_5	AB'C
6	1	1	0	1	m_6	ABC'
7	1	1	1	1	m_7	ABC

Forme standard Somme-de-Produits- **Fonction**

- l'expression Somme-de-produits est de la forme

F (A,B,C, …) = (…) + (…) + (…) + …

- Les parenthèses peuvent contenir une ou plusieurs variables
- Somme de produits peut être réaliser avec des portes logiques comme suit:

F (A,B,C, …) = (ET) OU (ET) OR (ET) OU ...

Forme standard Somme-de-Produits- **Fonction**

• La Forme somme-de-produits n'est pas unique, et ne contient pas nécessairement toutes les variables par exemple:

F (A,B,C) = A'B'C' + A'BC + C'A'B + C'AB' + BAC + BAC

et F (A,B,C) = B + B'C'

sont deux expressions somme-de-produits .

Forme standard Somme-de-Produits- **Fonction**

- A partir de la table de vérité ci-dessous on obtient F comme suit:

F(A,B,C) = A'BC' + A'BC + AB'C' + AB'C + ABC' + ABC

Un notation plus simple nous donne:

$$F(A,B,C) = m_2 + m_3 + m_4 + m_5 + m_6 + m_7$$

$$= \sum m(2, 3, 4, 5, 6, 7)$$

Decimal value	A	B	C	F	Minterm	
0	0	0	0	0	m_0	A'B'C'
1	0	0	1	0	m_1	A'B'C
2	0	1	0	1	m_2	A'BC'
3	0	1	1	1	m_3	A'BC
4	1	0	0	1	m_4	AB'C'
5	1	0	1	1	m_5	AB'C
6	1	1	0	1	m_6	ABC'
7	1	1	1	1	m_7	ABC

Produit-de-sommes: Maxterms

- De la table de vérité nous avons $\overline{F}(A,B,C) = (A'B'C' + A'B'C)$
- Ainsi on obtient F de $\overline{F}$:

$$F(A,B,C) = [\overline{\overline{F}}(A,B,C)] = \overline{(A'B'C' + A'B'C)}$$

$$= (A'' + B'' + C'') \cdot (A'' + B'' + C') = (A + B + C) \cdot (A + B + C')$$

Forme Compact $F = M_0 \cdot M_1 = \prod M\,(0, 1)$

Table de vérité

Decimal	A	B	C	F	Minterm	Maxterm	$M_i = \overline{m_i}$
0	0	0	0	0	m_0	M_0	A+B+C ← (A'B'C')'
1	0	0	1	0	m_1	M_1	A+B+C'
2	0	1	0	1	m_2	M_2	A+B'+C
3	0	1	1	1	m_3	M_3	A+B'+C'
4	1	0	0	1	m_4	M_4	A'+B+C
5	1	0	1	1	m_5	M_5	A'+B+C'
6	1	1	0	1	m_6	M_6	A'+B'+C
7	1	1	1	1	m_7	M_7	A'+B'+C'

Obtenir les formes standards a partir d'une expression Booléenne

- Étant donnée une expression booléenne arbitraire
- Trouver le nombre (2^n) pour les n entrées
- Déterminer la table de vérité et identifier les termes pour lesquelles la fonction est vrai- Les Minterms
- Écrire la fonction comme suit:

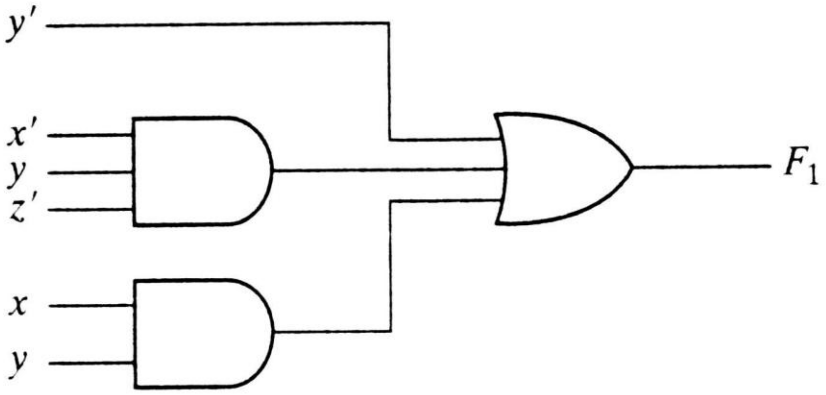
$$F = \sum_{i=0}^{n-1} m_i$$

- Alternativement, identifier les termes pour lesquelles la fonction est Faux -Maxterm
- Écrire la fonction :

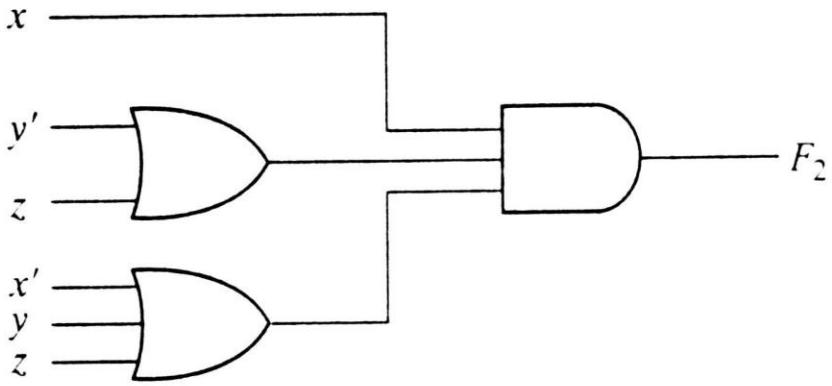
$$F = \prod_{j=0}^{n-1} M_j \quad j \neq i$$

Implémentation des formes standards avec ET et OU

- Implémentation a deux-niveaux

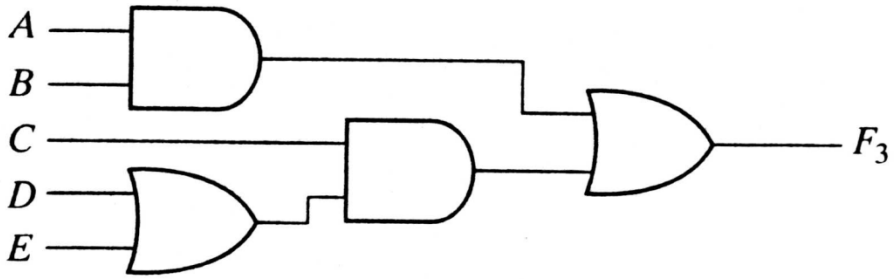


(a) Sum of Products



(b) Product of Sums

- Implémentation multi-niveaux



(a) $AB + C(D + E)$

Exemples

1- Demi-Additionneur

2- Additionneur Complet

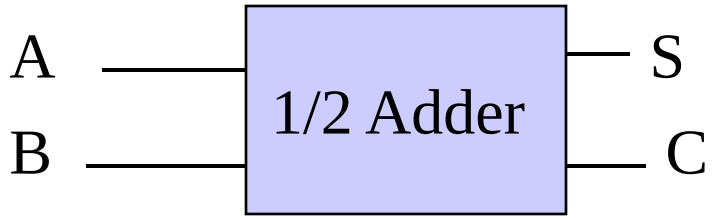
3- Additionneur-Soustracteur

Demi additionneur

→Le demi additionneur accepte deux chiffres binaires en entrée et produit deux chiffres binaires en sortie: le bit somme et le bit retenue.

Table de Vérité

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



$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 1\ \ 0 \end{array}$$

C (retenue)

S(somme)

Demi-Additionneur

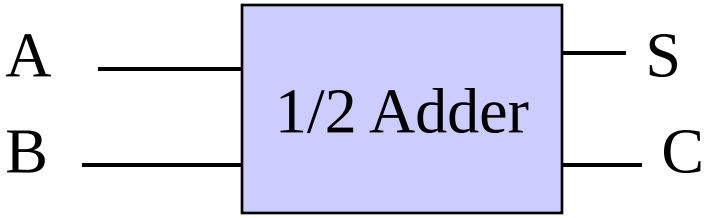


Table de vérité

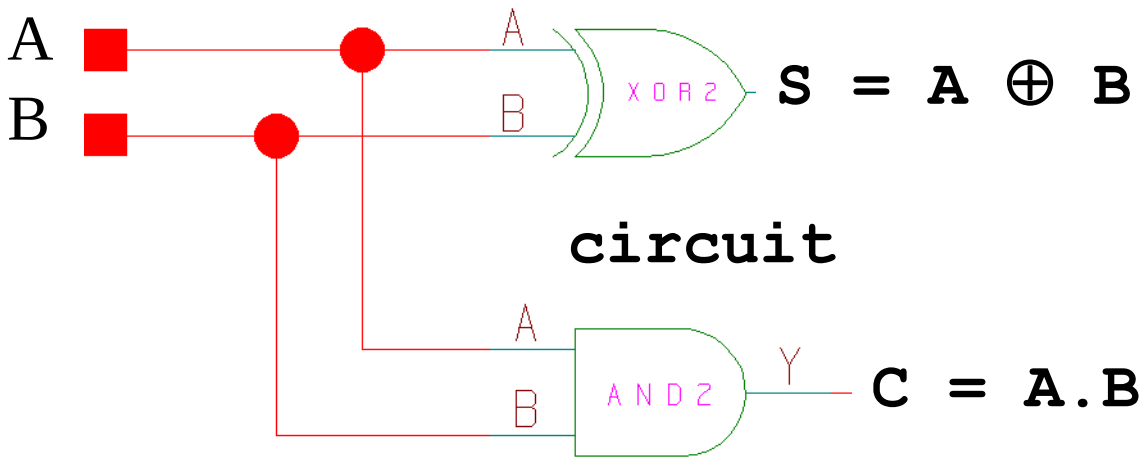
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Fonction logique

S = A'B + AB'

S = A ⊕ B

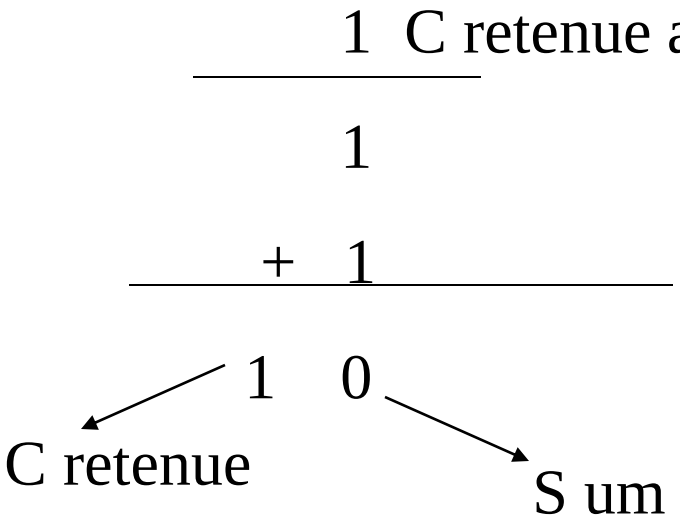
C = A.B

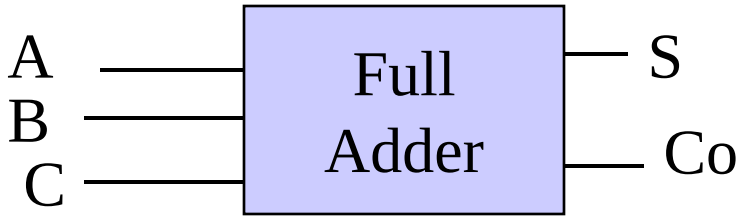


Additionneur Complet

→ L'additionneur complet accepte en entrée deux bits et un bit de retenu, il génère en sortie une somme et une retenue

→ La différence de base entre l'additionneur complet et le demi-additionneur est que l'additionneur complet accepte une retenue en entrée.





Additionneur Complet : Fonction

Fonction Logique

Table de vérité

A	B	C	C_o	S
0	0	0	0	0
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	1

$$C_o = A'BC + AB'C + ABC' + ABC$$

$$= C[A'B + AB'] + AB[C' + C]$$

$$= C[A \oplus B] + AB.1$$

$$= C(A \oplus B) + AB$$

$$S = A'B'C + A'BC' + AB'C' + ABC$$

$$= A'[B'C + BC'] + A[B'C' + BC]$$

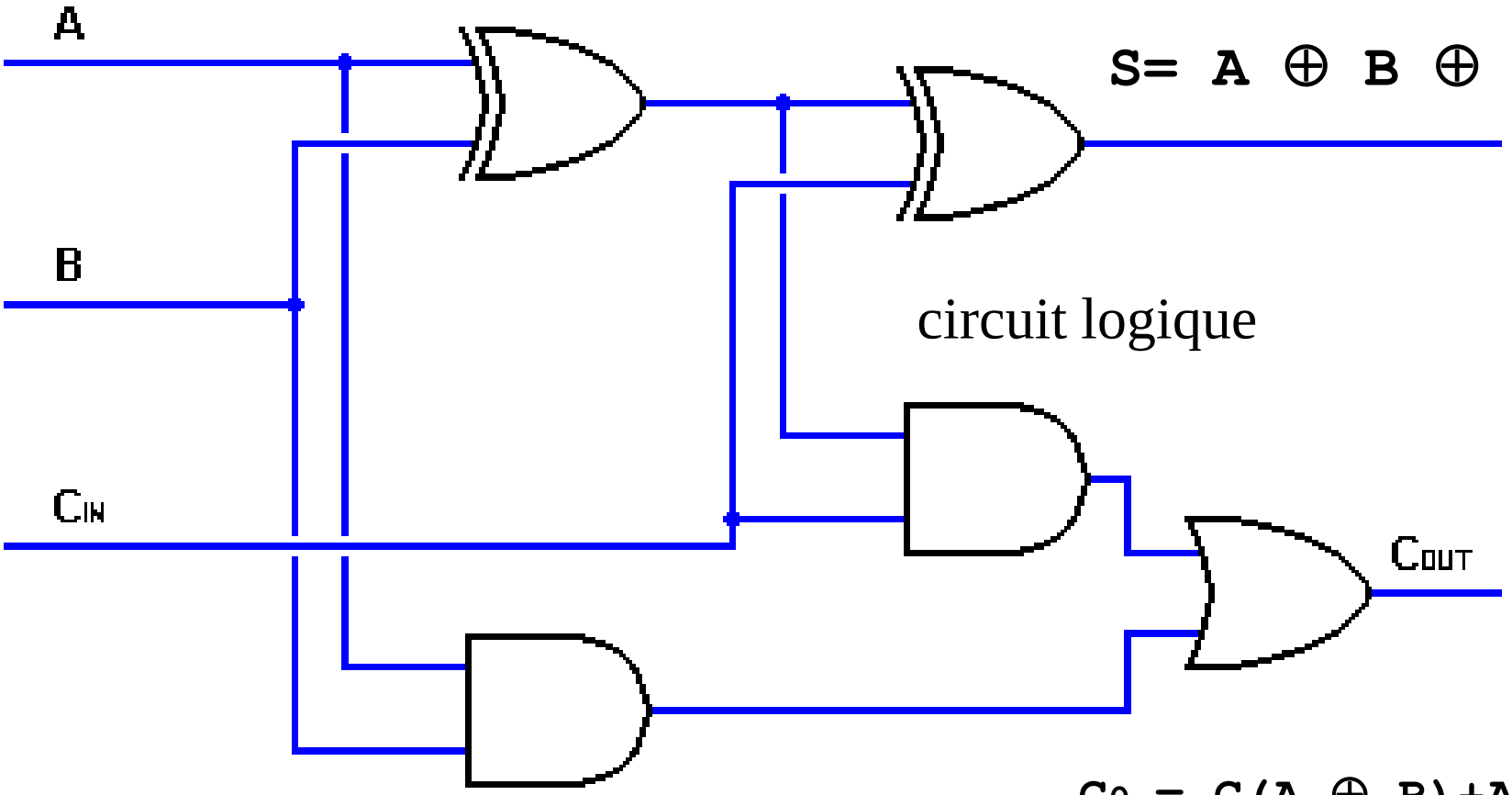
$$= A'[B \oplus C] + A[B \oplus C]'$$

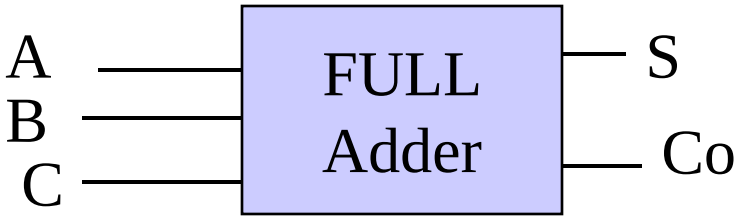
$$= A' \underbrace{X}_{[X]} + A \underbrace{x'}_{[X]'}$$

$$= A \oplus X$$

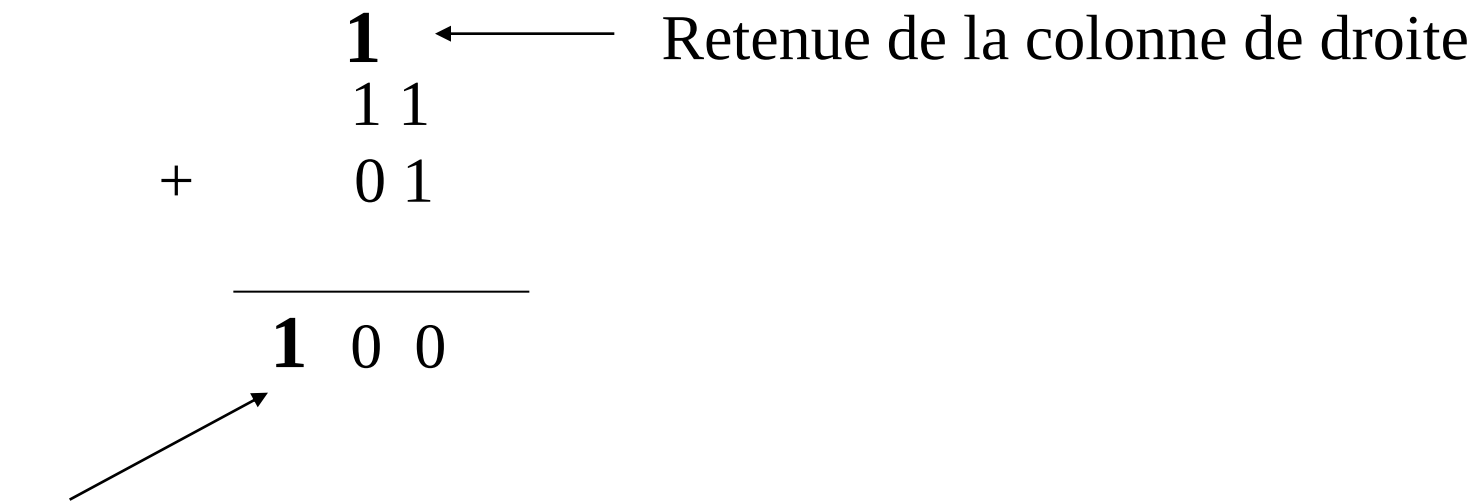
$$= A \oplus B \oplus C$$

Additionneur Complet : réalisation



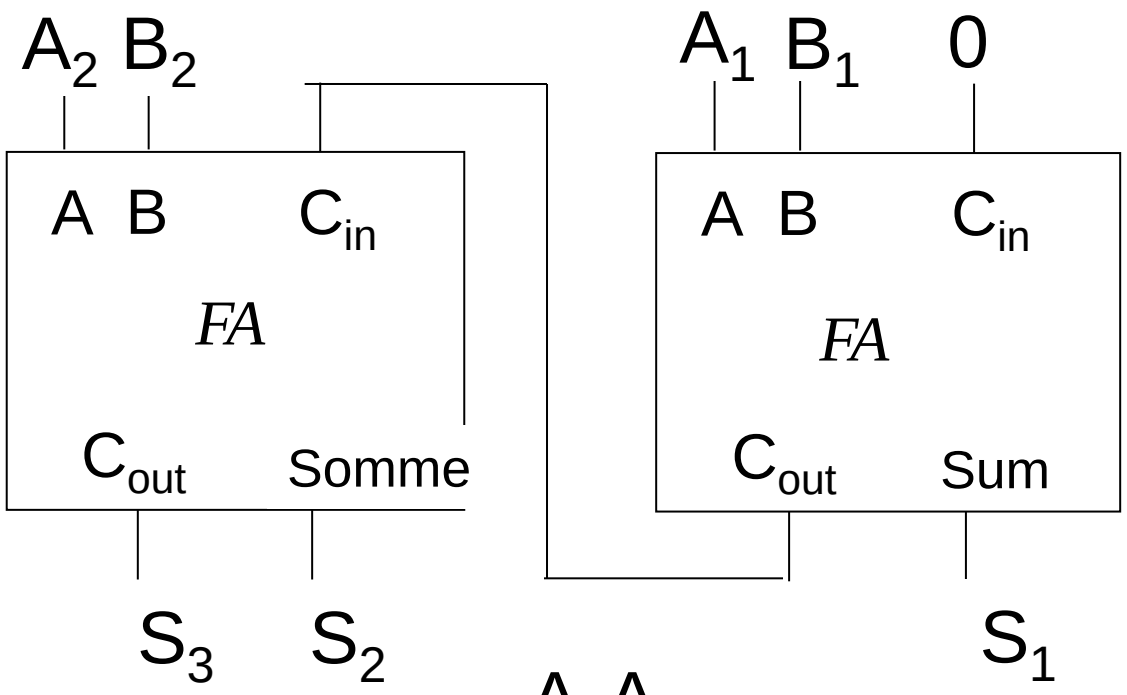


Additionneur de deux bits parallèle



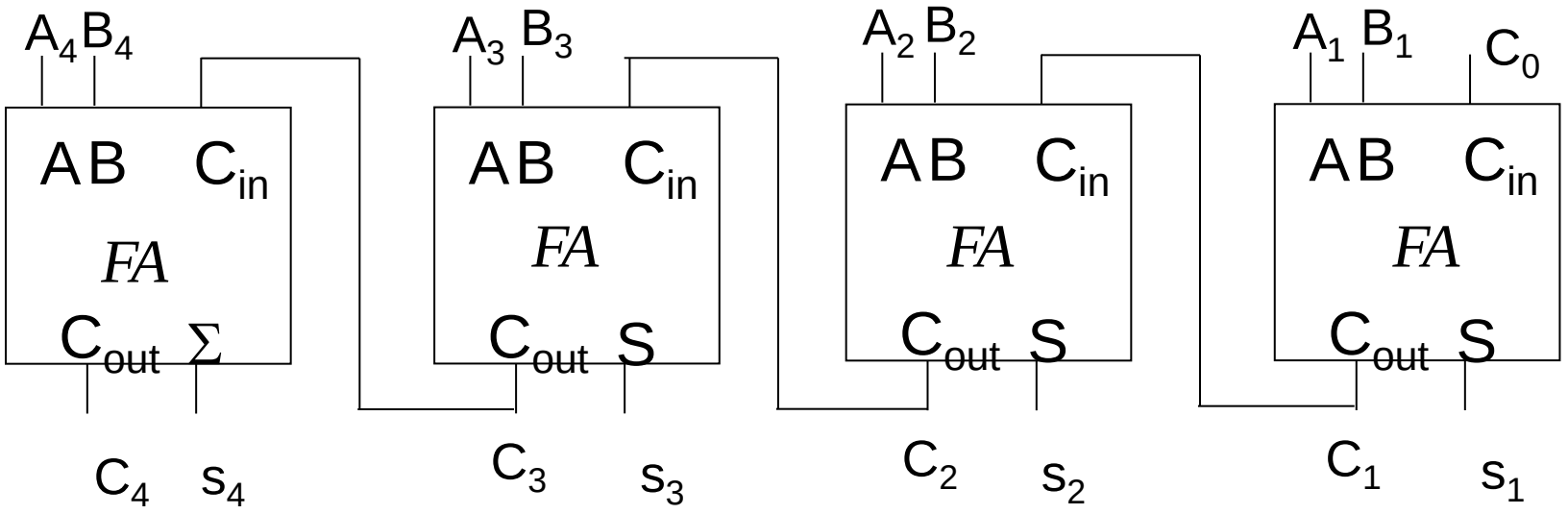
Retenue issue de la deuxième colonne
qui devient le bit de somme

Additionneur parallèle de deux bits



$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline S_3\,S_2\,S_1 \end{array}$$

Additionneur parallèle de quatre bits



Exemples de débordement (rappel)

- registre à 6-bits

  + 17 = 010001

  + 16 = +010000

  =100001 **→débordement**

- **100001 = - (11111) = -(31)$_{10}$** au lieu de **+ (33)$_{10}$**

- Même chose avec registre 7-bits

  + 17 = 0 010001

  + 16 = +0 010000

  = 0100001

  0100001 = + 33 Pas de débordement

Additionneur soustracteur complément à deux

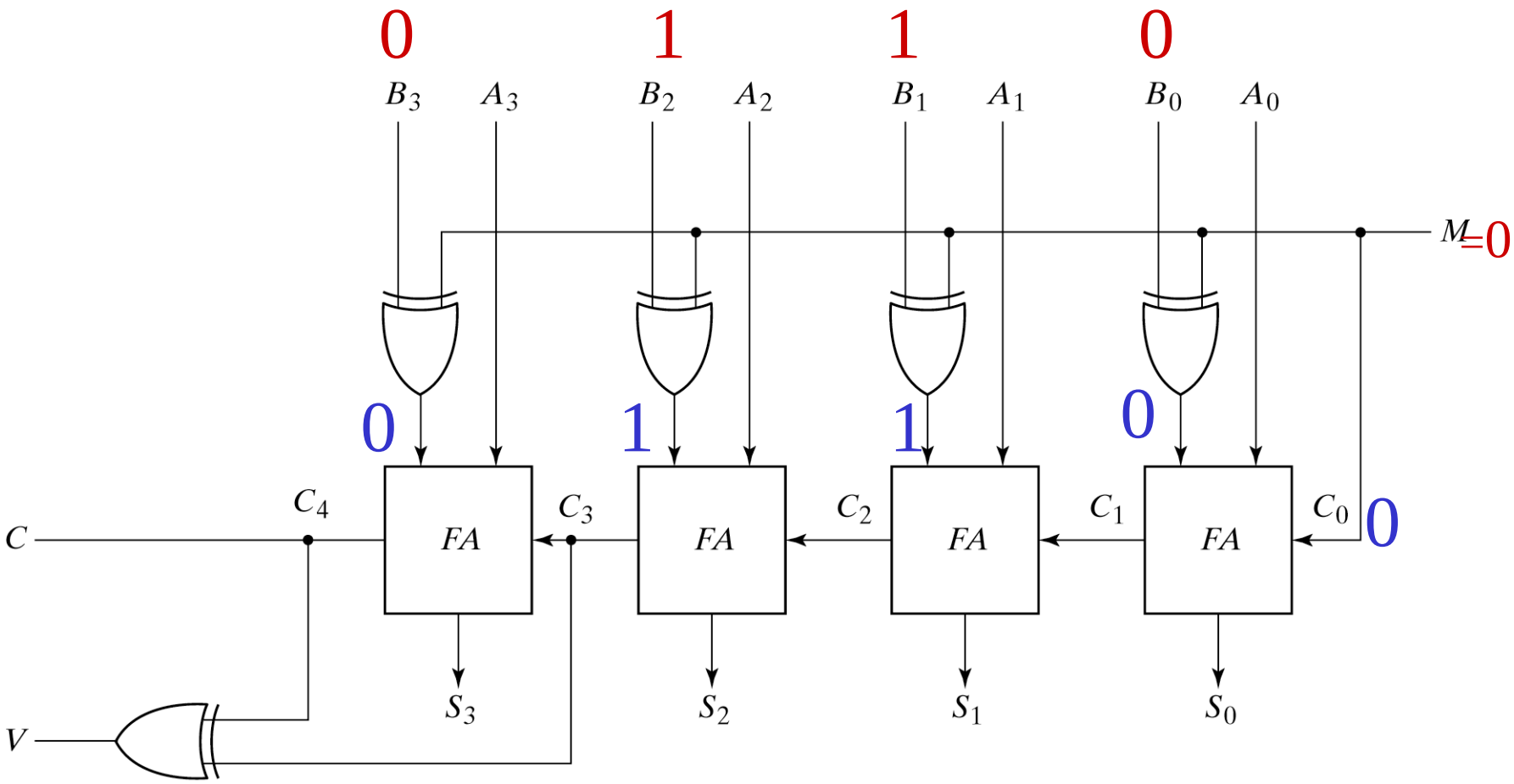


Fig. 4-13 4-Bit Adder Subtractor

Additionneur soustracteur complément à deux

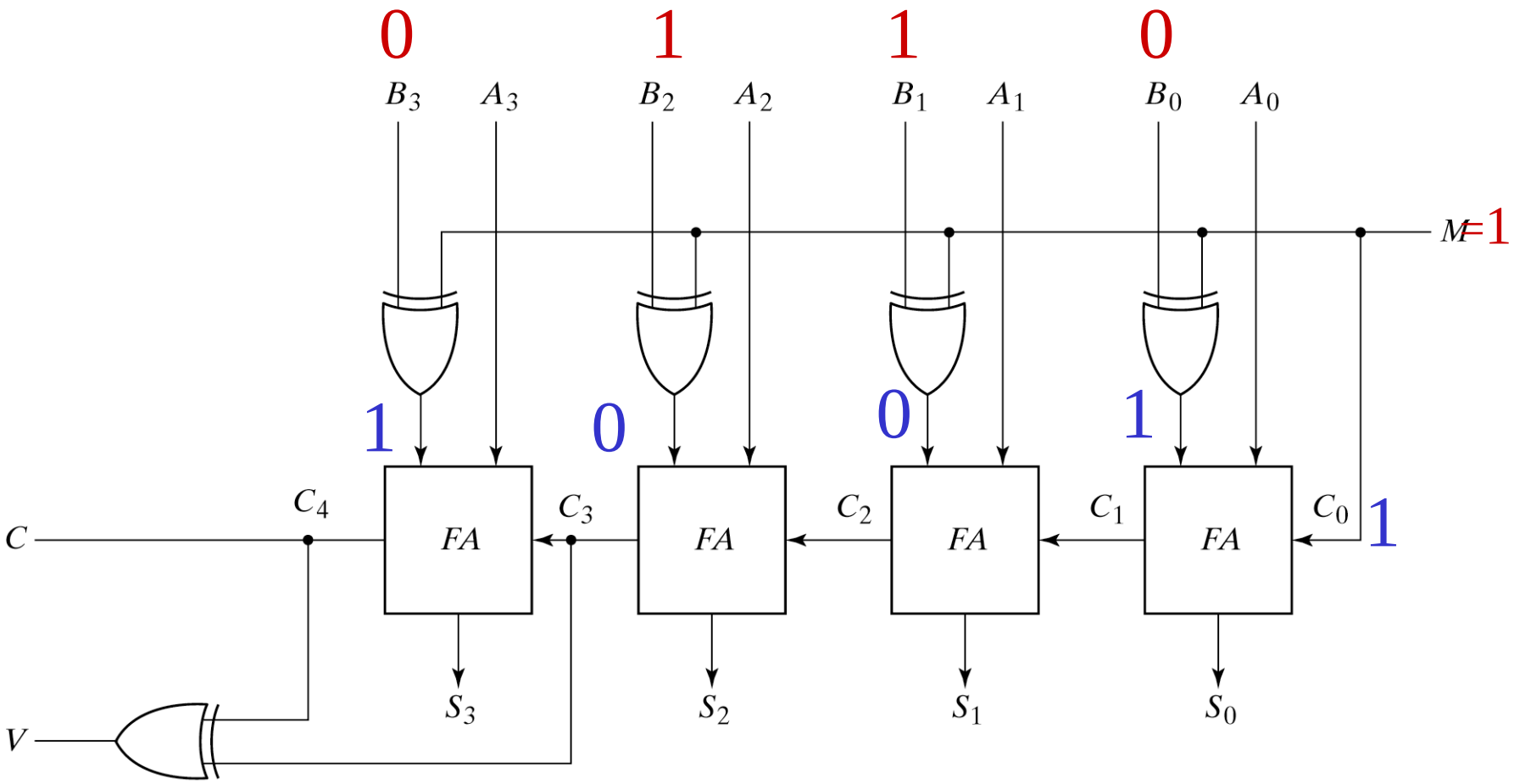


Fig. 4-13 4-Bit Adder Subtractor

Additionneur-4 bits -Exemples

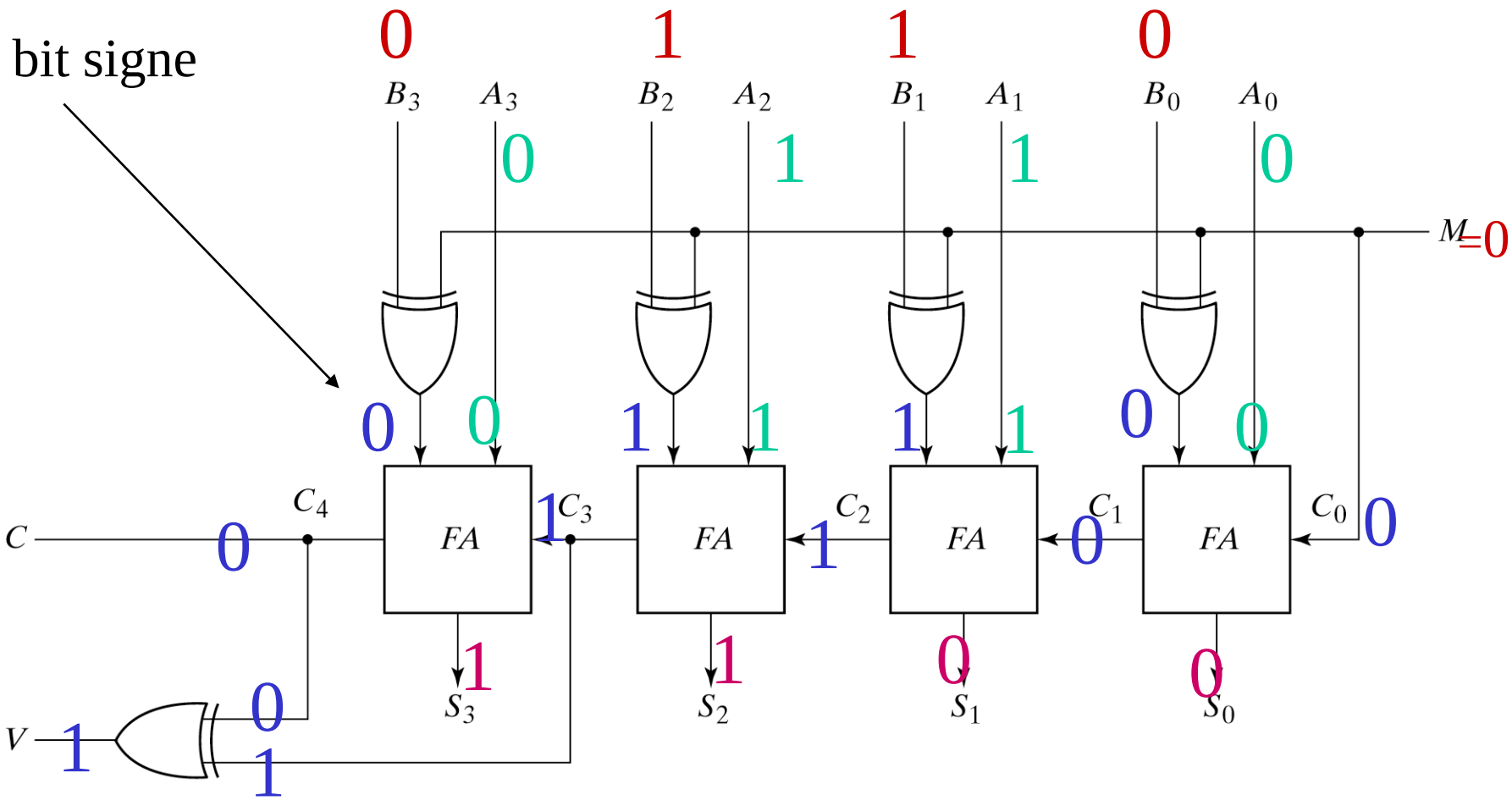
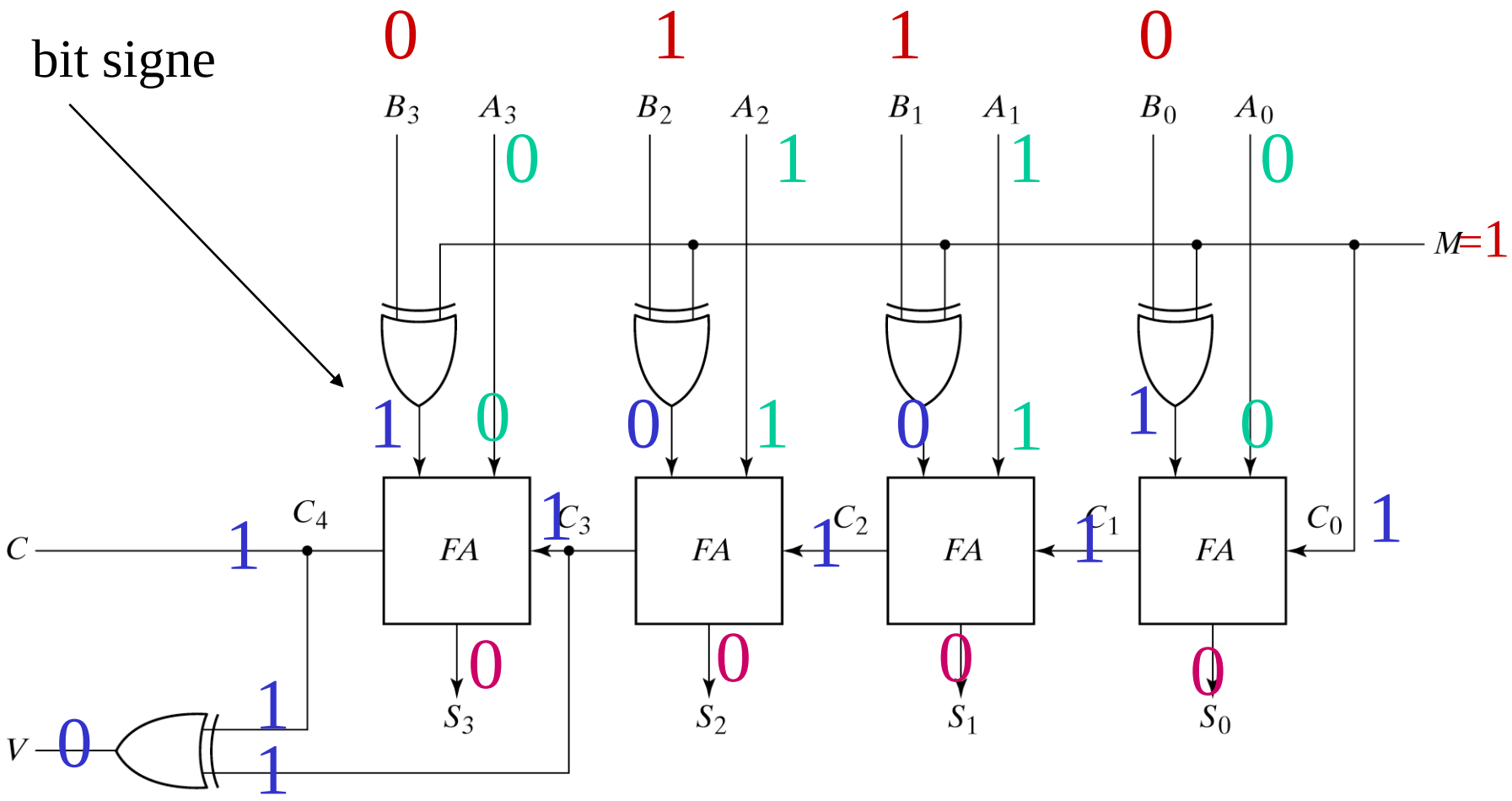


Fig. 4-13 4-Bit Adder Subtractor

débordement

Additionneur-4 bits -Exemples



Pas de débordement

Fig. 4-13 4-Bit Adder Subtractor

Formes standards SDP & PDS - Exemple

A partir d'une table de vérité arbitraire

- **Partie I**

1- Obtenir l'expression Somme de produits pour F

2- déterminer la réalisation a deux niveaux de F sans simplification

3- simplifier F avec les identités de Boole

4- déterminer la réalisation a deux niveaux de F

5- comparer les deux réalisations de F

- **Partie II**

•Répéter la partie 1 avec le produit de sommes.

Expression somme de produits

Table de vérité arbitraire

i	A	B	C	F	Minterms
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	1	**→m_2 = A'BC'**
3	0	1	1	1	**→m_3 = A'BC**
4	1	0	0	0	
5	1	0	1	1	**→m_5 = AB'C**
6	1	1	0	0	
7	1	1	1	1	**→m_7 = ABC**

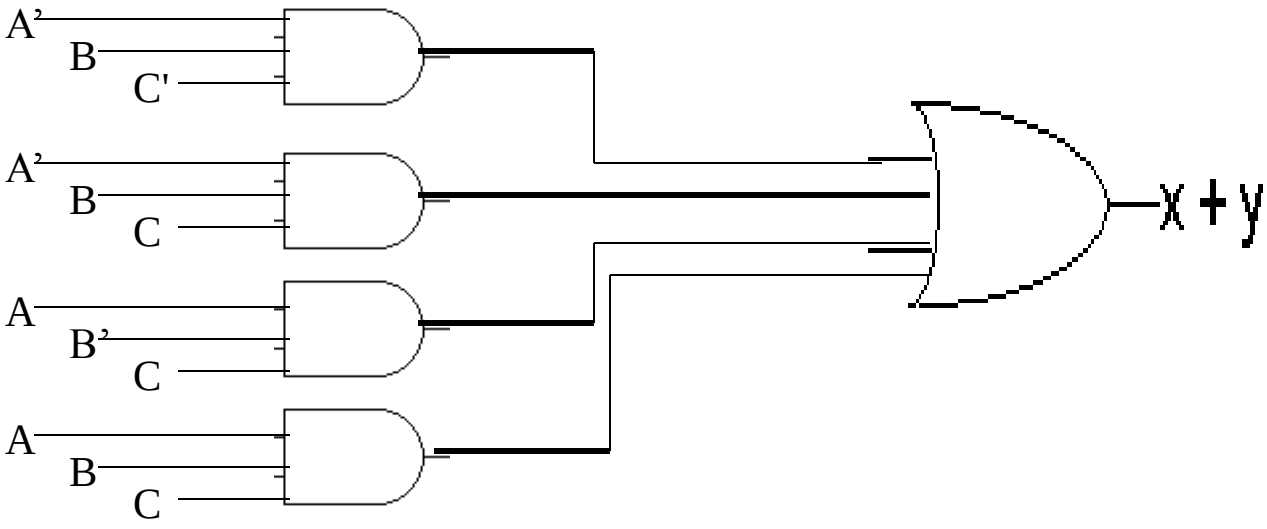
Expression somme de produits

1- **Expression somme de produits**

a) $F = m_2 + m_3 + m_5 + m_7$

$= A'BC' + A'BC + AB'C + ABC$

b) réalisation a deux niveaux (non simplifiée)



Expression somme de produits

c) Simplification

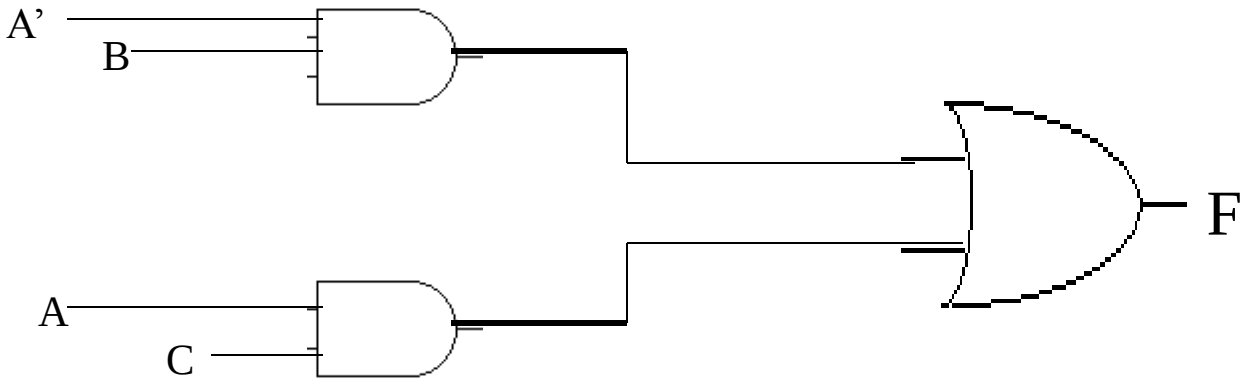
En utilisant l'identité absorption $xy + xy' = x$

On peut simplifier $\mathbf{m_2}$ avec $\mathbf{m_3}$ et $\mathbf{m_5}$ avec $\mathbf{m_7}$

$m_2 + m_3 = (A'B)\,C' + (A'B)\,C = A'B$

$m_5 + m_7 = (AC)B' + (AC)B = AC$

Alors $\mathbf{F = A'B + AC}$ réalisation a deux niveaux



Circuit avec 3 portes au lieu de 5

Expression Produit de sommes

2- Expression Produit de sommes

i	A	B	C	F	Maxterms
0	0	0	0	0	$\rightarrow M_0 = A+B+C$
1	0	0	1	0	$\rightarrow M_1 = A+B+C'$
2	0	1	0	1	
3	0	1	1	1	
4	1	0	0	0	$\rightarrow M_4 = A'+B+C$
5	1	0	1	1	
6	1	1	0	0	$\rightarrow M_6 = A'+B'+C$
7	1	1	1	1	

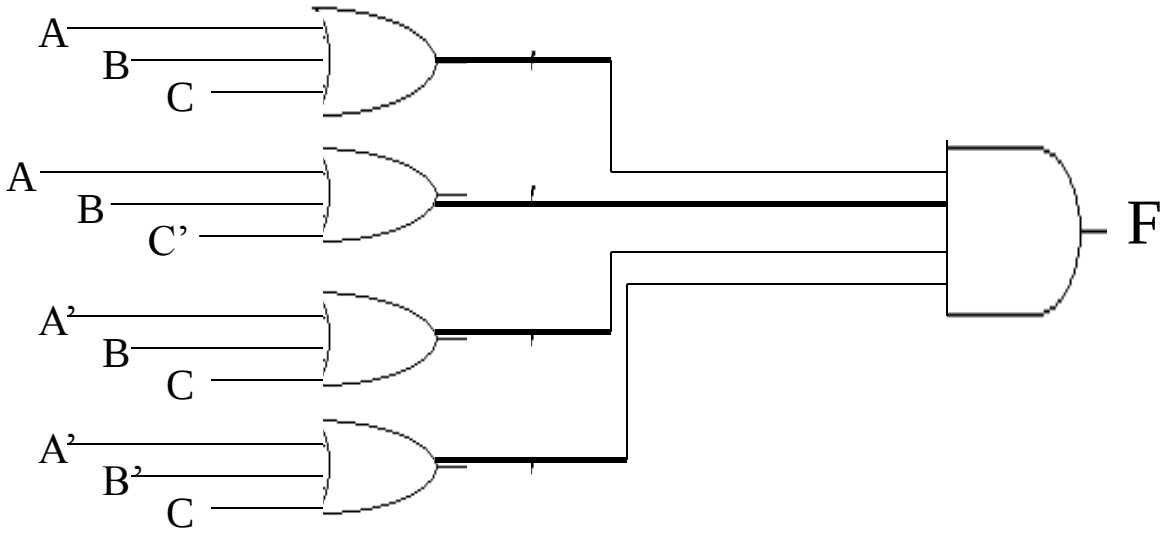
Expression Produit de sommes

a) Fonction

$F = M_0.M_1.M_4.M_6$

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

circuit logique a deux niveaux



Expression Produit de sommes

b**) Simplification**

en utilisant (X + Y) (X+Y') = X

Vérification

X	Y	Y'	X+ Y	X + Y'	**(X + Y) (X+Y')**
0	0	1	0	1	0
0	1	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1

$M_0.M_1$ = [(A+B) + C] [(A+B) + C']= **A+B**

$M_4.M_6$ = [A' + C) + B] [(A'+C) + B'] = **A'+C**

F = (A+B)(A'+C)

Expression Produit de sommes

c) Réalisation a deux niveaux