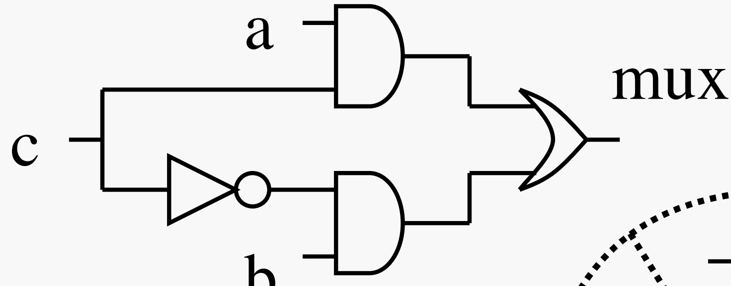
LOGIQUE COMBINATOIRE

ES102 / CM2a

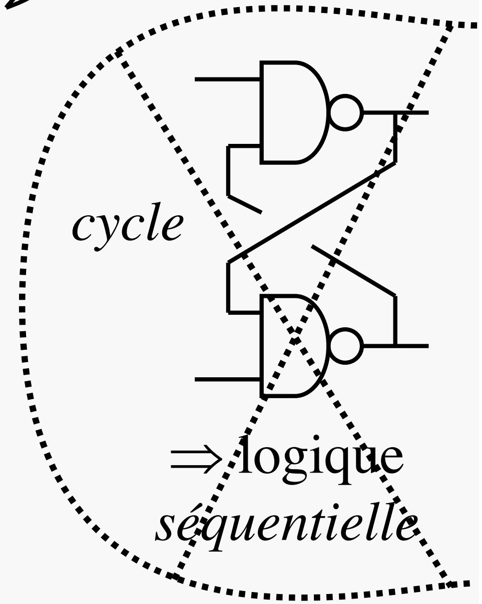
CIRCUIT COMBINATOIRE

- Graphe orienté *acyclique*, où
  - nœuds = portes logiques + ES
  - arcs = connexions électriques, conduisant une valeur binaire de la sortie d'une porte à l'entrée d'une autre



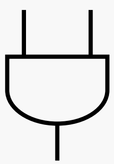
- Propriété *essentielle* : dès ses entrées fixées, un tel circuit rejoint rapidement un état d'équilibre qui ne dépend que d'elles et qui détermine la valeur de la (des) sortie(s) …

  ⇒ indépendance vis-à-vis des valeurs antérieures des entrées !

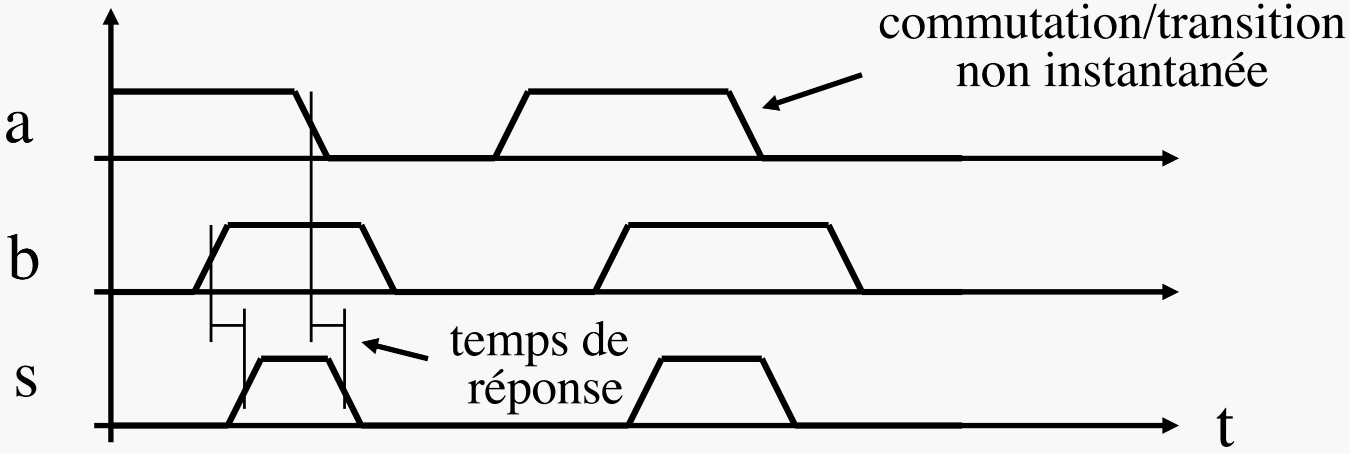


… en un certain *délai*, comme observé sur un *chronogramme* :

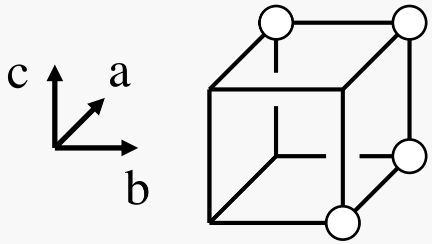
a b



s



UNE FONCTION, DE MULTIPLES RÉALITÉS POSSIBLES



Diverses spécifications

$$\begin{cases} c=1 \Rightarrow mux=a \\ c=0 \Rightarrow mux=b \end{cases}$$

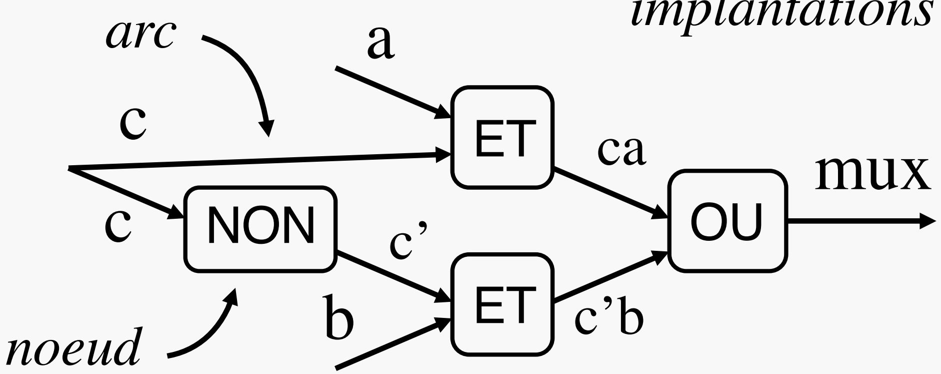
mux

			a	a
	0	1	1	0
c	0	0	1	1
		b	b	

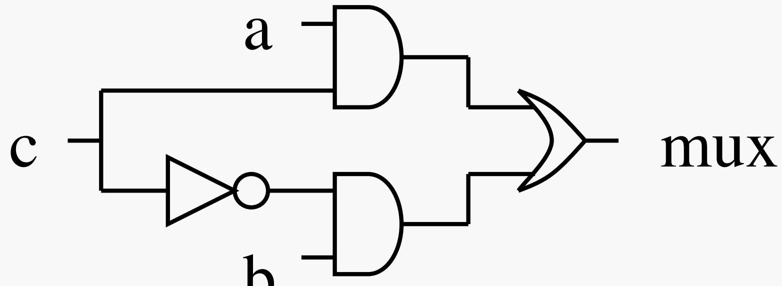
$$\begin{aligned} mux &= a'bc' + abc' + abc + ab'c \\ &= c'b + ab + ca \\ &= c'b + ca \\ &= (c+b)\cdot(c'+a) \end{aligned}$$

Diverses formules booléennes

Diverses implantations



Graphe orienté de sous-fonctions



Circuit combinatoire, implanté en portes

Performances/coût

FORME NORMALE DISJONCTIVE (FND)

Représentation canonique des fonctions booléennes

Autres

Tables *Formules*

FND

Graphes *Circuits*

FORME NORMALE DISJONCTIVE (FND)

caractéristique

- Soit $A \subset \mathbb{B}^n$: 1_A = fonction indicatrice valant 1 sur A et 0 ailleurs

- Cas $A=\{s\}$: s sommet de $\mathbb{B}^n$ ex. : $1_{\{(0,1,0)\}} = x' \cdot y \cdot z'$
  - $1_{\{s\}}$ = ET des *n* variables, certaines complémentées
  - un tel *produit* $\prod$ avec toutes les variables est appelé *minterme*

- Cas $A=\underline{f}$ (support de f) où $f \in \mathcal{F}_n$
  - on a : $f = 1_{\underline{f}} = \boxed{\sum_{s \in \underline{f}} 1_{\{s\}}}$ ex. : mux = a'bc' + abc' + abc + ab'c
  - OU, alias *somme* ($\sum$), de mintermes
  - forme particulière de f, appelée la Forme *Normale* Disjonctive (FND)
  - unique, à l'ordre des variables et termes près
  - forme $\sum\prod$ canonique, mais lourde

mux

				a
	0	1	1	0
c	0	0	1	1
		b	b	

EXPANSION DE BOOLE

Principe récursif formel d’obtention de la FND

Autres

Tables

Formules

FND

arbre

Graphes

Circuits

EXPANSION DE BOOLE

- Soit $f \in \mathcal{F}_n$, *x* l'une des *n* variables, *Y* la liste des *n*-1 autres

- Formules d'expansion de Boole :
  - $f(x,Y) = x' \cdot f(0,Y) + x \cdot f(1,Y)$ ⟶ formule disjonctive
  - $f(x,Y) = [\, x + f(0,Y) \,] \cdot [\, x' + f(1,Y) \,]$ ⟶ formule conjonctive

- f(0,Y) et f(1,Y) sont respectivement appelés *cofacteurs négatif* et *positif* de f par rapport à *x*

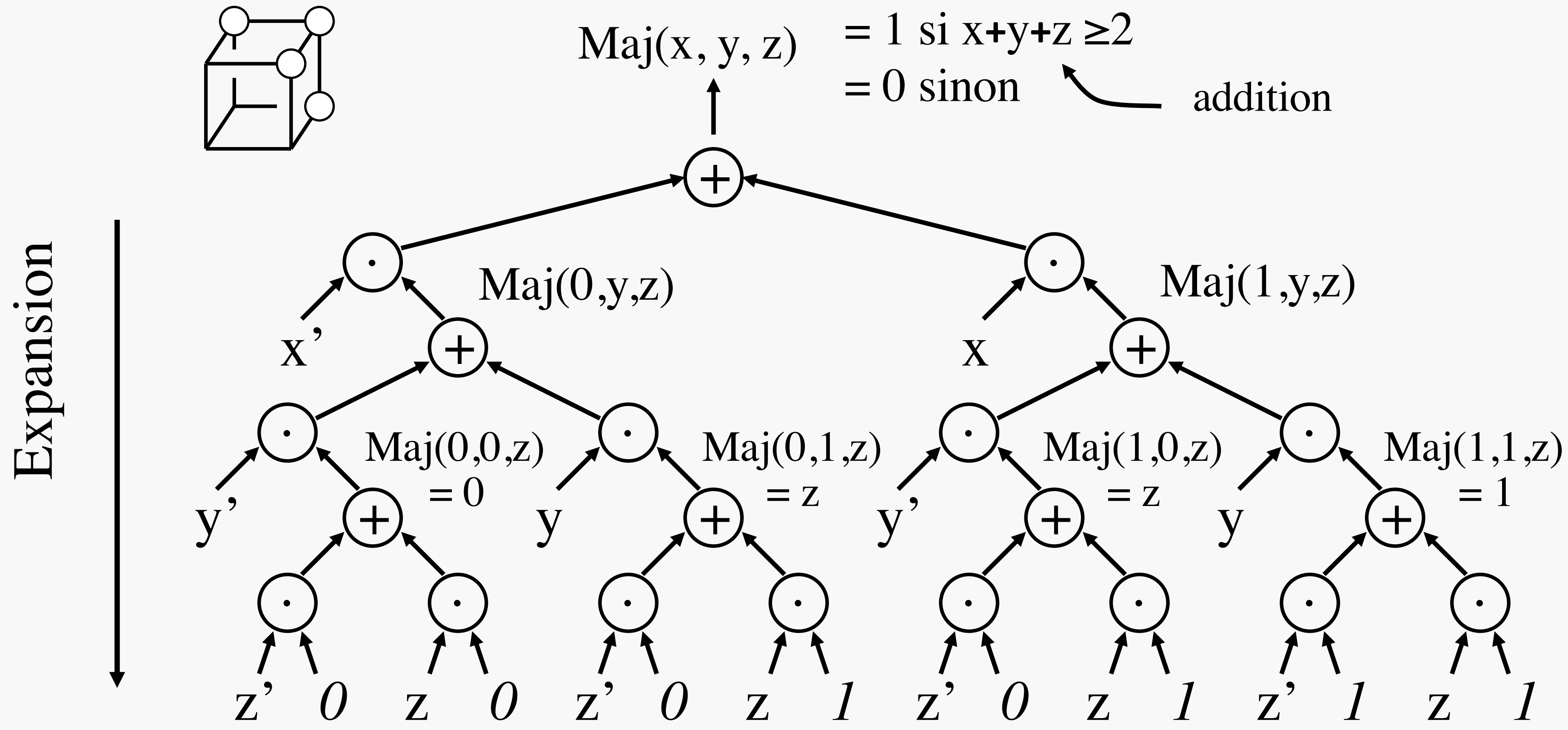
  0 est la négation de 1…

  – exemple : $f(x, y) = x \oplus y$ ⟶ cofacteurs : $f(0, y) = y$ (négatif) $f(1, y) = y'$ (positif)

  ⟶ *PC1/Exo1*

- Récursivité :
  – les cofacteurs sont des fonctions booléennes de *n*-1 variables, auxquelles une expansion est de nouveau applicable
  – il en découle un arbre de calcul de *f*, alternant ET et OU par étages (↵)

EXEMPLE D'EXPANSION DISJONCTIVE



- arbre ⇔ formule parenthèsée à $2n$ niveaux d'imbrication
  - développement par distributivité de · sur + : x·(g+h) = x·g + x·h
  - fournit la FND de Maj : x'yz + xy'z + xyz' + xyz

FORME DISJONCTIVE MINIMALE (FDM)

Recherche de concision, pour implantation

Autres

Tables *Formules*

FDM

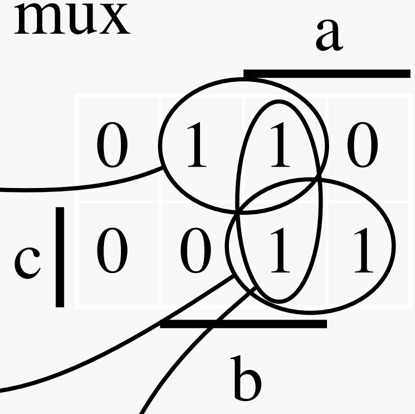
Graphes *Circuits*

FORME DISJONCTIVE MINIMALE (FDM)

- FDM = forme disjonctive ($\sum\prod$) la plus concise possible
  - approche intuitive pour l'instant,
    par regroupement astucieux des 1
    sur tables de Karnaugh fonctionnelles
    - possible pour *n* petit seulement ↵

exemple : mux = c'b + ca

mux

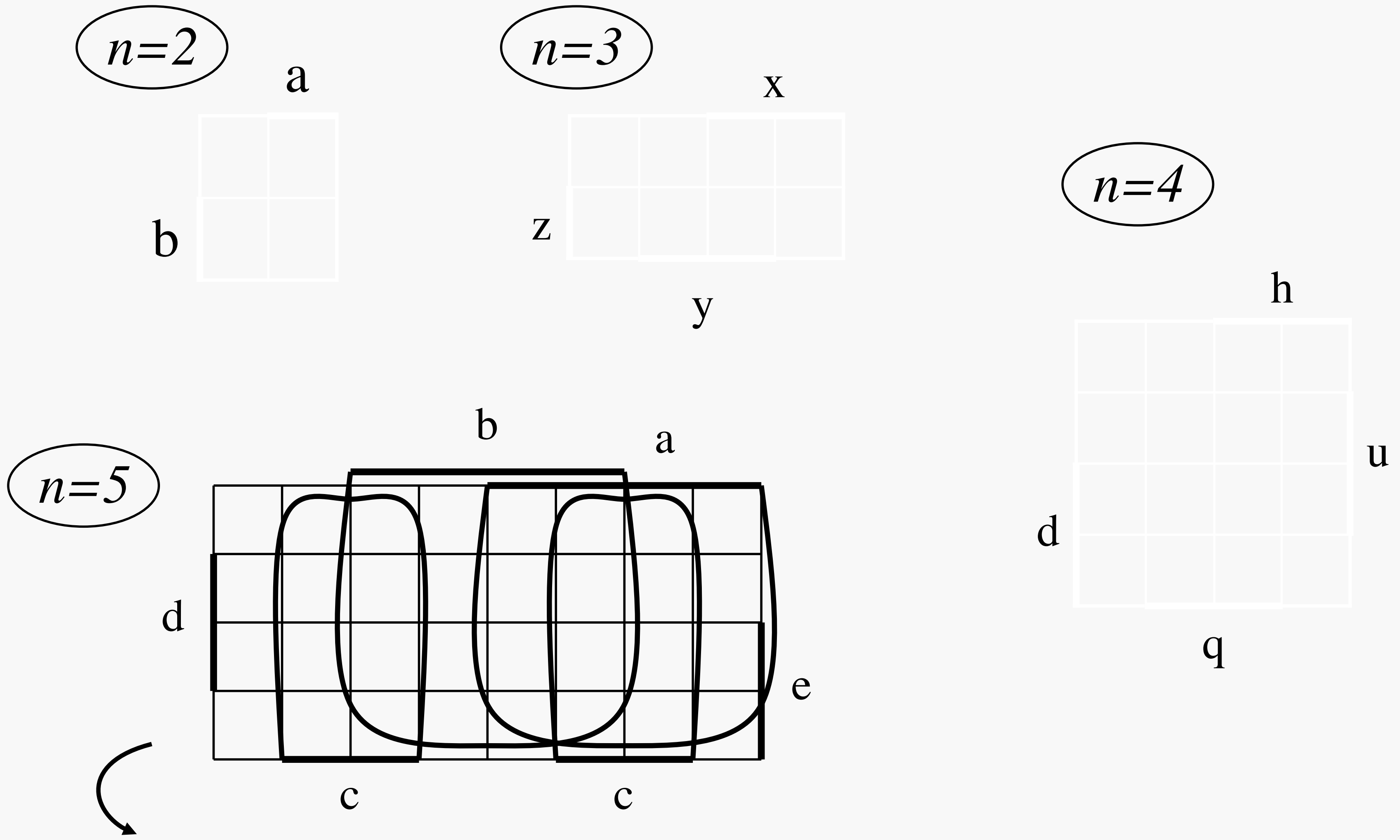


ab superflu
puisque regroupant des 1
déjà pris en compte

  - formalisation au CM4
    - pour *n* quelconque

Ainsi, c'b + ca + ab est une forme $\sum\prod$ *non* minimale de mux.

ART DE LA TABLE



{c=1} coupée en 2 : sacrifiée car impossible d'avoir les zones {a=1}, {b=1} et {c=1} toutes les 3 connexes

$\Rightarrow$ FDM plus dure à *lire*

ALGÈBRE DE BOOLE

Savoir transformer les formules booléennes…

Autres

Tables

Formules

Graphes

Circuits

ALGÈBRE DE BOOLE (1)

- Structure fondée sur l'implication '$\Rightarrow$' en tant qu'ordre partiel
  - 0 : plus petit élément
  - 1 : plus grand élément

$\leq$

$\mathbb{B} = \{ 0, 1 \}$

$0 \Rightarrow 0 \Rightarrow 1 \Rightarrow 1 \not\Rightarrow 0$

- Un ordre partiel sous-tend des opérateurs min ($\wedge$) et max ($\vee$) :
  - $\subset$ sous-tend $\cap$ et $\cup$ $\longrightarrow$ interprétation ensembliste naturelle des propriétés booléennes
  - $\div$ sous-tend PGCD et PPCM
  - l'implication ($\Rightarrow$) sous-tend ET ($\cdot$) et OU ($+$) :

$\wedge$ $\vee$

pour x et y variables binaires, $x \cdot y \begin{array}{c} \Rightarrow x \Rightarrow \\ \Rightarrow y \Rightarrow \end{array} x+y$

  - vraies $\forall$ les valeurs prises par x et y, les implications ci-dessus ont un sens fonctionnel : celui d'un ordre partiel sur les fonctions booléennes :

$$\text{soient } f, g \in \mathcal{F}_n : (f \Rightarrow g) \overset{\Delta}{\Leftrightarrow} \forall X \in \mathbb{B}^n, (f(X) \Rightarrow g(X))$$

Le OU exclusif ($\oplus$) présente quant à lui des propriétés algébriques plus classiques : ($\mathbb{B}$, $\oplus$, $\cdot$) est un corps (fini).

ALGÈBRE DE BOOLE (2)

- Propriétés natives :
  - reflétant bien la dualité 0-1 min/max ·/+
  - ✓ + (OU) et · (ET) : lois internes commutatives et associatives
  - ✓ 0 neutre pour + et absorbant pour ·
  - ✓ 1 neutre pour · et *absorbant pour +*
  - ✓ · distributif sur + et + *distributif sur ·*
  - ✓ unique complément :

    $\forall a,\ \exists! a',\ a+a'=1$ et $a \cdot a'=0$

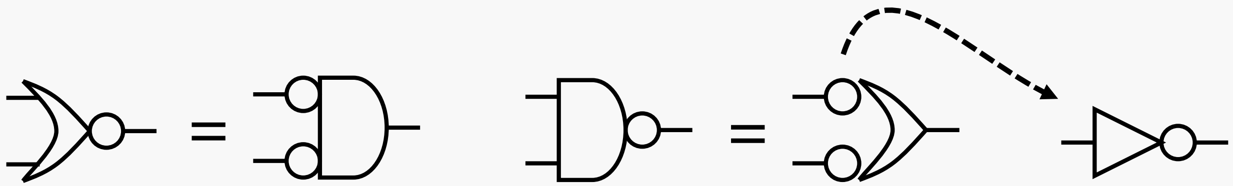
$$\left\{\begin{array}{ll} x+(y\cdot z) = (x+y)\cdot(x+z) & \text{:-o} \\ X\cup(Y\cap Z) = (X\cup Y)\cap(X\cup Z) & \text{:-)} \\ x\vee(y\wedge z) = (x\vee y)\wedge(x\vee z) & \text{:-||} \end{array}\right.$$

- Théorèmes :
  - Idempotence : $\forall x,\ x\cdot x=x$ et $x+x=x$ — $(X\cap X) = X = (X\cup X)$
  - Absorption : $\forall(x,y),\ (y\cdot x)+x=x$ — $(Y\cap X)\cup X = X$

    version duale $\forall(x,y),\ (y+x)\cdot x=x$ — $(Y\cup X)\cap X = X$
  - De Morgan : $\forall(x,y),\ (x+y)' = x'\cdot y'$ — $\overline{X\cup Y}=\overline{X}\cap\overline{Y}$

    version duale $\forall(x,y),\ (x\cdot y)' = x'+y'$ — $\overline{X\cap Y}=\overline{X}\cup\overline{Y}$

Equivalents ensemblistes



→ jeu de bulles

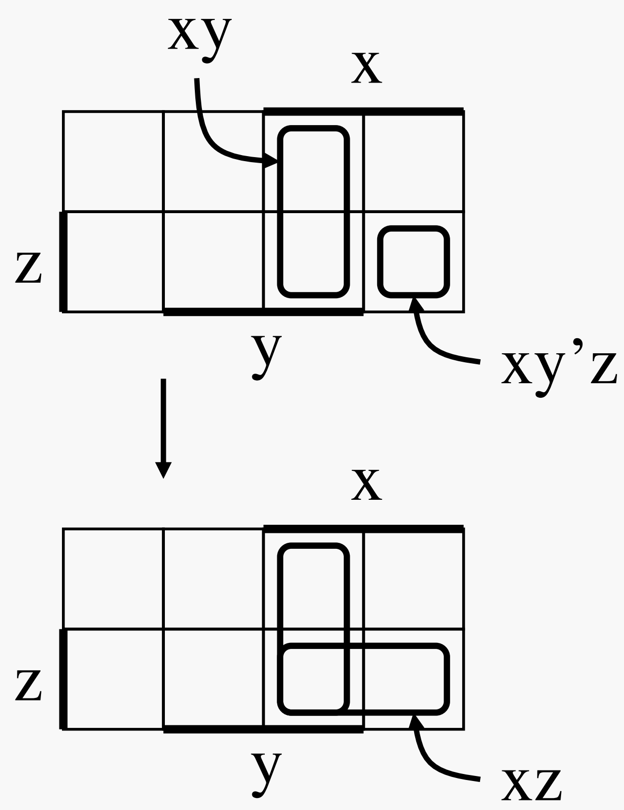
MANIPULATION ALGÉBRIQUE DE FORMULES BOOLÉENNES

- Commodités : 1) précédence de · sur + 2) omission de ·

- Exemple de transformation rigoureuse :

$$
\begin{array}{rl}
 & xy + xy'z \\
\text{absorption} & \\
 & xy + xyz + xy'z \\
\text{distributivité} & \\
 & xy + x(y+y')z \\
\text{complément} & \\
 & xy + x1z \\
\text{élément neutre} & \\
 & xy + xz
\end{array}
$$

- équivalent ensembliste sur table de Karnaugh :

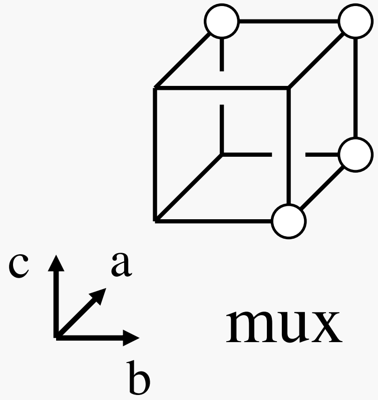


FONCTIONS : DE L'ORDRE À LA CROISSANCE

- f(x,Y) croissante par rapport à *x* ssi f(0,Y) $\Rightarrow$ f(1,Y)

  $\Leftrightarrow$ partout où f(0,Y)=1, f(1,Y) aussi $\rightarrow$ *PC2*

  – exemple : MUX croissant par rapport aux *données* a et b mais *pas* par rapport à la *commande* c



- f est croissante si elle l'est par rapport à chacune de ses variables
  – reconnaissable aux FDM sans variable complémentée : xy, x+y, Maj(a, b, c)

« barre tombée à gauche = slash »

- f est décroissante si /f est croissante
  – reconnaissable aux FDM ne comportant que des variables complémentées
  – le transistor MOS, brique fonctionnelle décroissante $\rightarrow$ *CM3&4*

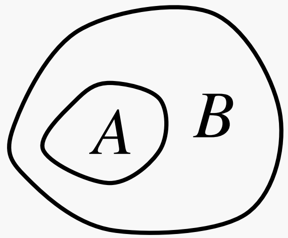
$\rightarrow$ *PC2*

Logique : (a$\Rightarrow$b) $\Leftrightarrow$ $\neg a \vee b$

Fonctions booléennes : (a$\Rightarrow$b) = a'+ b

Ensembles :

$A \subset B \Leftrightarrow \overline{A} \cup B = E$



JEU DE BULLES

La dualité 0-1 en action

Autres

Tables

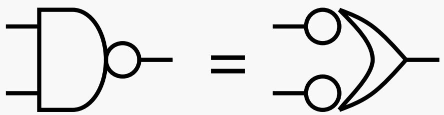
Formules

De Morgan

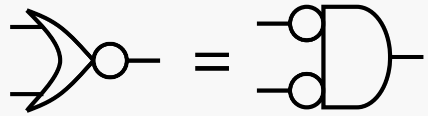
Graphes

Circuits

jeu de bulles

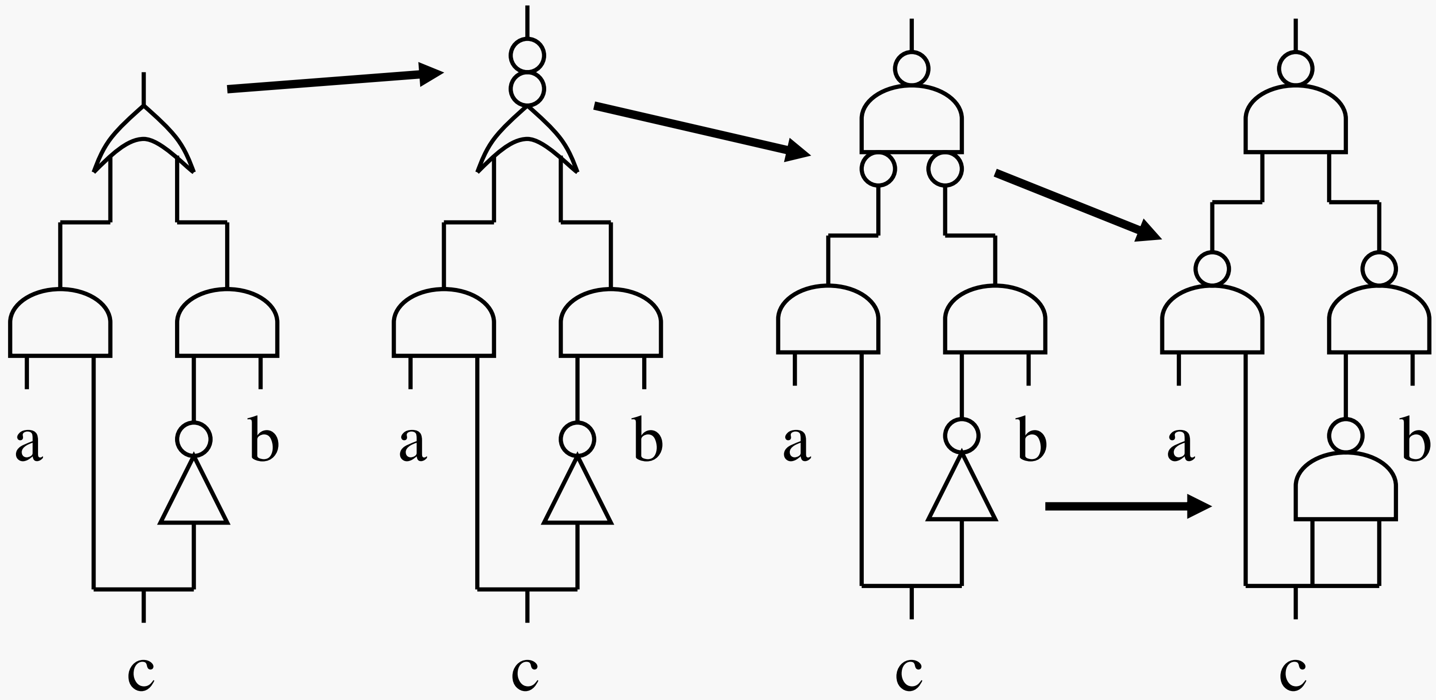


JEU DE BULLES



- Version graphique des lois de De Morgan
  - exploitée ici pour se restreindre à des portes NAND
- Exemple du MUX :

FDM(mux) = ca+c'b



- Procédé généralisable à toute fonction

$\Rightarrow$ portes NAND suffisantes pour implanter toute fonction booléenne

  - NOR aussi, mais NAND technologiquement privilégiée ($\rightarrow$ CM4)

DIAGRAMMES DE DÉCISION BINAIRE (BDD)

Recherche de concision, en se servant de graphes

Autres

Tables

Formules

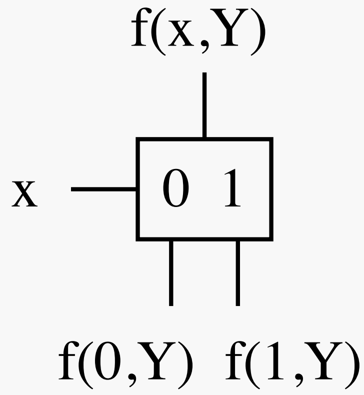
Graphes

Circuits

DIAGRAMME DE DÉCISION BINAIRE

- Expansion de Boole possible avec l'opérateur multiplexeur 2 vers 1 :

$$f(x,Y) = \mathrm{MUX}[\ x, f(1,Y), f(0,Y)\ ]$$



- Application récursive aux deux cofacteurs…
- Fournit un arbre de calcul de *f* (BDD brut ↵) réductible en un graphe plus simple (↵) appelé *diagramme de décision binaire* (BDD en anglais)

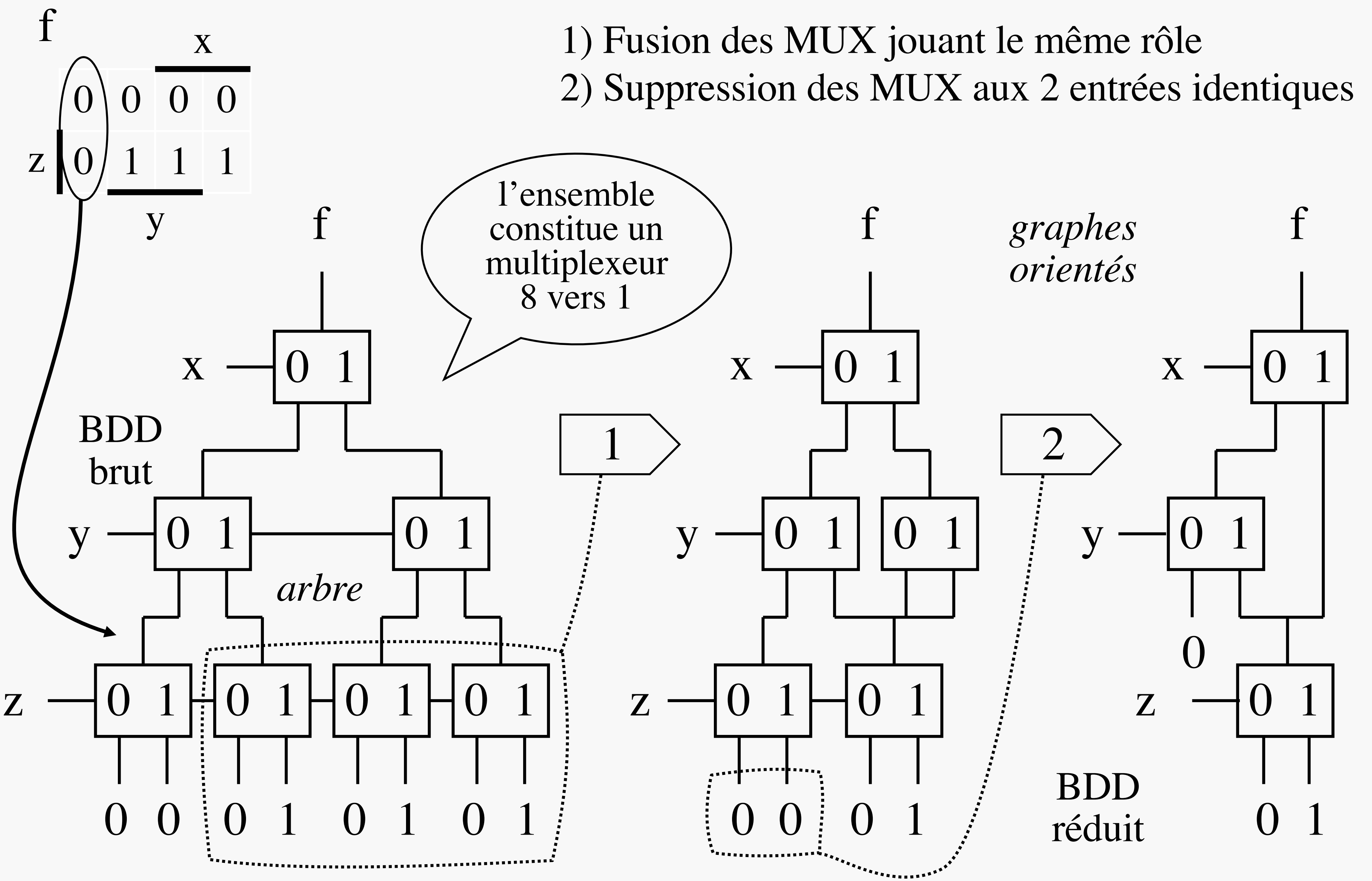
- Un BDD dépend de l'ordre pris pour les variables

- Son intérêt : une fois réduit, il est bien moins lourd que la FND, tout en étant unique (pour l'ordre des variables considéré)

→ représentation de choix dans les logiciels spécialisés

ÉLABORATION / RÉDUCTION D'UN BDD

1) Fusion des MUX jouant le même rôle
2) Suppression des MUX aux 2 entrées identiques



NUMÉRATION & ARITHMÉTIQUE

(SUITE CM1)

ES102 / CM2b

DE BASE EN BASE

47	2						
1	23	2					
	1	11	2				
		1	5	2			
			1	2	2		
				0	1	2	
					1	0	

LSB

MSB

condition d'arrêt

- Conversion décimal/binaire par divisions successives :
  fournie par les restes, jusqu'à quotient nul
  $(47)_{10} = (32 + 8 + 4 + 2 + 1)_{10} = (101111)_2$

  * procédé différent pour les puissances négatives de 2 et la représentation des réels (→ PC2/Exo2)

- Ecritures concises proches de la base 2 grâce aux bases 8 (octal) ou 16 (hexadécimal)

$$(47)_{10} = (10\ 1111)_2 = (2F)_{16}$$

chiffres de 10 à 15 désignés par les lettres A à F

A	10
B	11
C	12
D	13
E	14
F	15

ADDITION EN BASE 2 (rappels CM1+PC1)

- S=A**+**B où A et B deux entiers (≥0) exprimés sur *n* bits :

tranche *carry*

retenues	c_n	c_{n-1}	c_{n-2}	...	c_{i+1}	c_i	...	c_1	$c_0 = 0$
A		a_{n-1}	a_{n-2}	...	a_{i+1}	a_i	...	a_1	a_0
+B		b_{n-1}	b_{n-2}	...	b_{i+1}	b_i	...	b_1	b_0
= S	$\cancel{s_n}$	s_{n-1}	s_{n-2}	...	s_{i+1}	s_i	...	s_1	s_0

Equation fondamentale :

$$2 \times c_{i+1} + s_i = a_i + b_i + c_i$$

$$= (c_{i+1}\ s_i)_2$$

addition

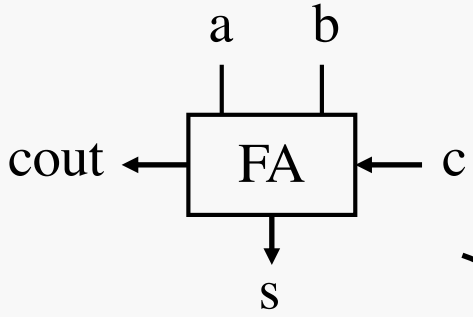
$$\Leftrightarrow \begin{cases} c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i \\ = \mathrm{Maj}(a_i, b_i, c_i) \text{ fonction } \textit{majorité} \\ s_i = a_i \oplus b_i \oplus c_i \text{ fonction } \textit{parité} \end{cases}$$

Comme A et B, S est *aussi* exprimé sur *n* (64, 32...) bits $\Rightarrow$ s_n *n'existe pas !*
c_n ne sert qu'à avertir en cas de *débordement* : A+B≥2^n

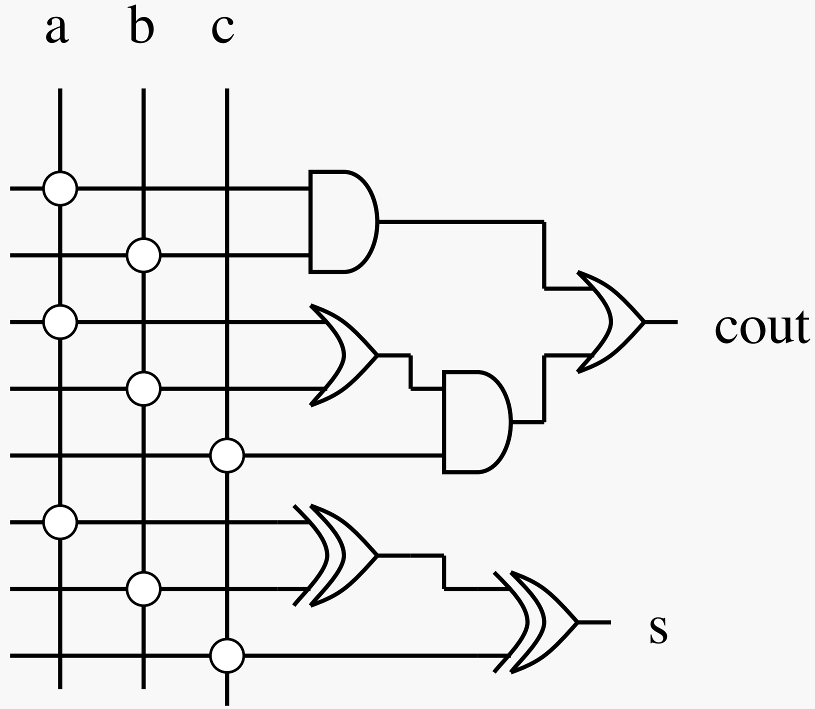
$$\begin{array}{rl} & 1\text{——}1 \\ + & 0\text{——}01 \\ \hline = & \underbrace{0\text{——}0}_{n} \end{array}$$

→ Addition modulo 2^n

ADDITION : IMPLANTATION



surface minimisée



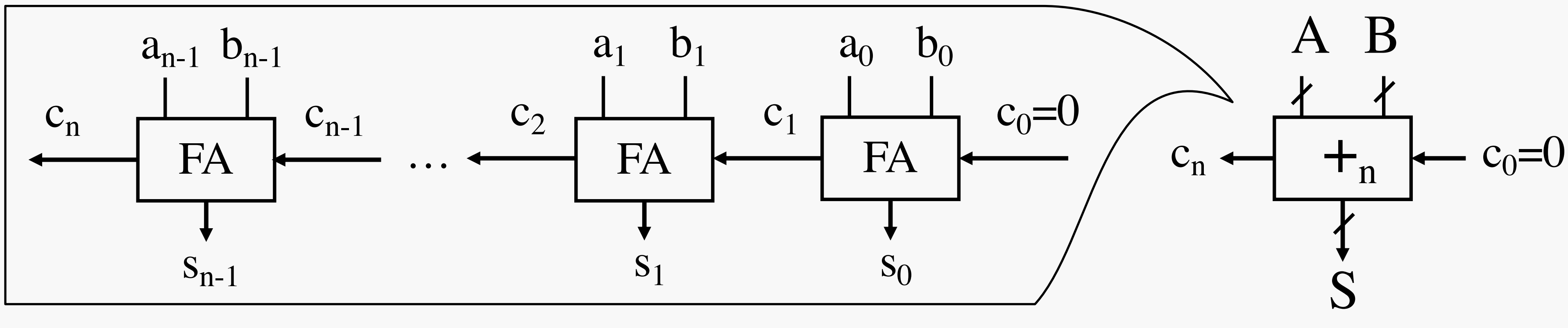
Préparation algébrique :

1) FDM[Maj(a, b, c)] = ab + ac + bc

2) Factoris. : Maj(a, b, c) = ab + c(a+b)

temps minimisé en PC2

Additionneur *n* bits (sous-entendu modulo 2^n) dit *à retenues propagées*



EN QUÊTE DE MOINS…

- Plutôt qu'aborder la soustraction comme l'addition, on cherche à tirer parti de la complémentation
- Soient $I_n = [\![0, 2^n\text{-}1]\!]$,
  $\gamma_n : I_n \to \mathbb{B}^n$ le code positionnel classique sur *n* bits,
  $A \in I_n$ et $\gamma_n(A) = (a_i)_{0 \leq i < n}$
- Complément vectorisé : $(a_i)'_{0 \leq i < n} = (a_i')_{0 \leq i < n}$
- Complément « numérisé » : $A' = \gamma_n^{-1}[\gamma_n(A)']$
- Propriété : $A + A' = 2^n\text{-}1 \quad \Rightarrow \quad A' = 2^n\text{-}1\text{-}A$
  - A' est le complément (arithmétique) de A à 2^n-1
  - dénommé « *complément à 1 de A* » ☺
    qui désigne aussi son code $\gamma_n(A') = \gamma_n(A)'$

(011)'
=(100)

3+3'=7

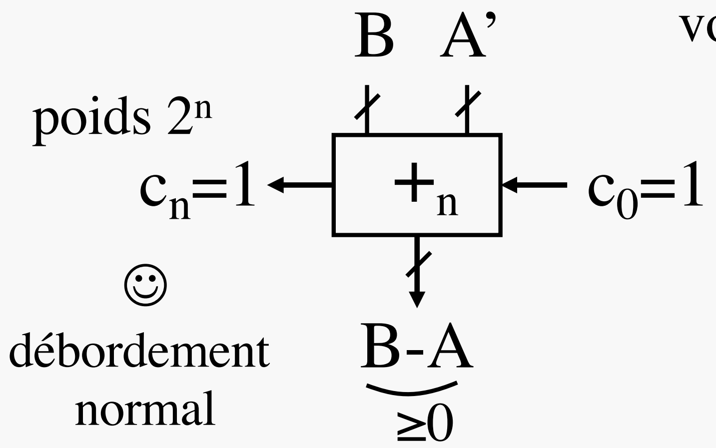
☺
on tient la
soustraction

SOUSTRACTION

- $A + A' = 2^n - 1 \quad \Rightarrow \quad A'+1 = 2^n - A$
  - $A'+1$ est le complément (arithmétique) de A à 2^n
  - dénommé « *complément à 2 de A* » 😐
- $A'+1 \equiv -A \ [2^n]$
- ajouter $A'+1$, c'est soustraire A, modulo 2^n
  - sur un additionneur numérique, calculer $B+A'+1$ fournit exactement $B-A$ si $B \geq A$

+1 réalisable à la volée avec $c_0=1$

$$B+A'+1 = \underbrace{B-A}_{\geq 0} + 2^n \geq 2^n$$



- cela fonctionnera aussi si B<A …

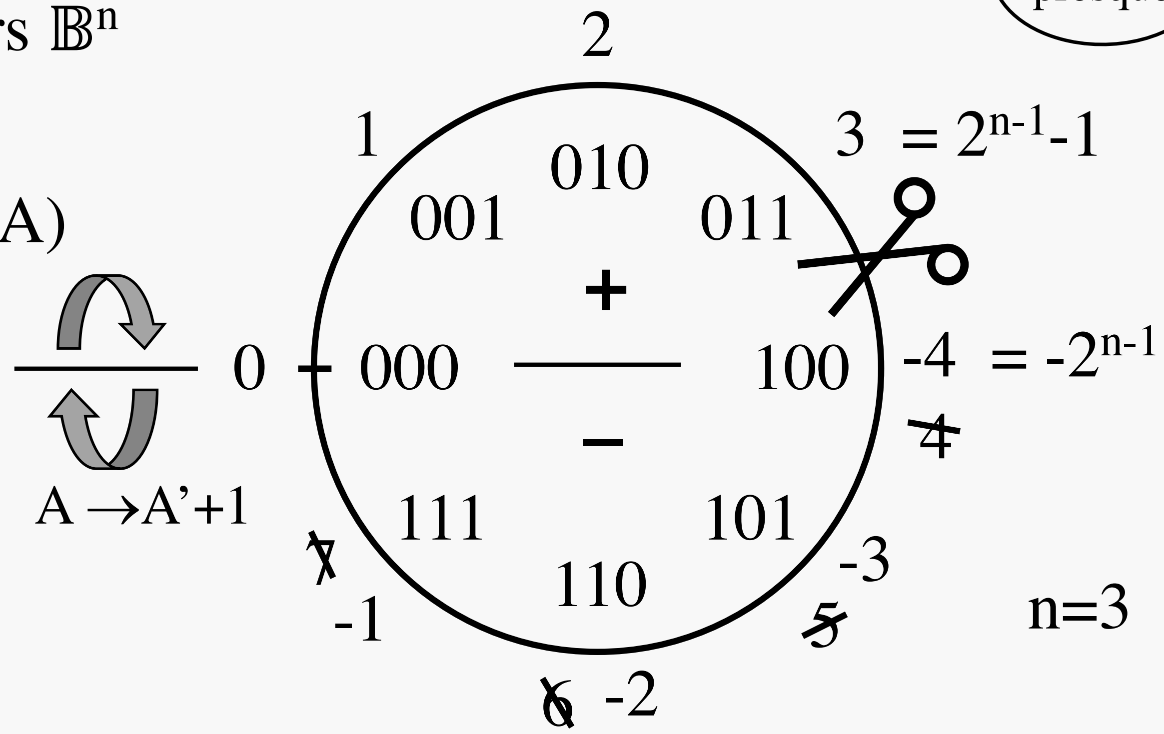
ENTIERS SIGNÉS

- Pertinent de représenter -A par $\gamma_n(2^n - \underbrace{A}_{>0})$
  - obtenu facilement en calculant A’+1
    - exemple : -1 représenté par $\gamma_n(2^n - 1) = 1\text{—}1$
    - mais alors, 1—1 représente-t-il $2^n - 1$ ou -1 ? Il faut choisir…

$$\begin{array}{lr} c_0=1 & 1 \\ C_1 & +\underline{1110} \\ C_2 & =1111 \end{array}$$

- Code γ_{sn} où chaque entier vient avec son opposé :
  - de $[\![-2^{n-1}, 2^{n-1}-1]\!]$ vers $\mathbb{B}^n$
  - $A \geq 0$ codé par $\gamma_n(A)$
  - $A < 0$ codé par $\gamma_n(2^n + A)$
  - code des entiers signés : *int* en C
  - γ_n, code des entiers non signés : *unsigned int* en C

enfin, presque



ENTIERS SIGNÉS : POIDS SUR LA CONSCIENCE

- Les bits ont-ils encore un poids avec γ_{sn} ?
- γ_{sn} choisi tel que le *MSB* a_{n-1} de $\gamma_{sn}(A)$ vaut 1 ssi $A<0$, donc :
  - $a_{n-1}=0 \Leftrightarrow A\geq 0 \Rightarrow A=\gamma_n^{-1}(a_{n-1}\cdots a_0)$
  - $a_{n-1}=1 \Leftrightarrow A<0 \Rightarrow A=\gamma_n^{-1}(a_{n-1}\cdots a_0)-2^n$

d'où $A=\sum_{i=0}^{n-1} 2^i a_i - 2^n a_{n-1}$ et donc $A=-2^{n-1}a_{n-1}+\sum_{i=0}^{n-2} 2^i a_i$

- γ_{sn} est donc également *positionnel*, avec poids identiques à γ_n, sauf celui du MSB a_{n-1}, désormais négatif : -2^{n-1}
  - MSB *pas simple bit de signe*
  - 10—0 représente -2^{n-1}
    - cf. coupure cercle modulo 2^n sur planche précédente
    - mais $+2^{n-1} \notin \gamma_{sn}^{-1}(\mathbb{B}^n)$

$$1—1 \ : \ -2^{n-1} + \sum_{i=0}^{n-2} 2^i = -1$$

NOMBRES FLOTTANTS (1)

- Entiers parfois/souvent inadaptés en pratique
  - notamment pour représenter les grandeurs physiques
  - dynamique : $2^{32} \approx 4 \cdot 10^9 \quad 2^{64} \approx 2 \cdot 10^{19}$
  - précision : entier 1, représentation imprécise du réel 1 !

  $\Rightarrow$ **besoin d'une version binaire de la notation scientifique**
- Principe de la notation : $\pm \;$ mantisse $\times\ 2^{\text{exposant}}$
  - chiffre non nul avant la virgule : ce ne peut être qu'un **1** !
- Exemple : $2{,}5 \;=\; 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} \;=\; (10{,}1)_2$

$$= \underbrace{(1{,}01)_2}_{\substack{\text{mantisse fractionnaire} \\ \text{normalisée } (1 \leq \cdot < 2)}} \cdot 2^{\overset{\text{exposant}}{1}} \;=\; (1{,}\underbrace{010\text{—}0}_{}\,)_2 \cdot 2^{\underbrace{(0\text{—}01)_2}_{}}$$

représentations en mémoire de la mantisse et de l'exposant (sur nombres de bits fixés)

NOMBRES FLOTTANTS (2)

- Position flottante de la virgule, définie par l'exposant (floating point numbers)
- Représentation en machine : (exposant : entier signé)

signe	exposant	mantisse

- Représentation fractionnaire normalisée de la mantisse :

  exposant ajusté t.q. $1 \leq |\text{mantisse}| < 2 \rightarrow \; = 1.b_{-1}b_{-2}b_{-3}b_{-4}\ldots$

- Format 64 bits :

$\pm$	$e_{10}e_9 \cdots e_1e_0$	$b_{-1}b_{-2}b_{-3}b_{-4} \cdots b_{-52}$
1	← 11 →	← 52 →

→ représentation de 2,5 :

0	0—000001	01000000000000—0

- Compromis précision/dynamique

  Dynamique : $2^{\pm 1024} \approx 10^{\pm 308}$ - Précision relative : $2^{-53} \approx 10^{-15}$

- Calculs spécifiques $\Rightarrow$ unités spécialisées, dites flottantes