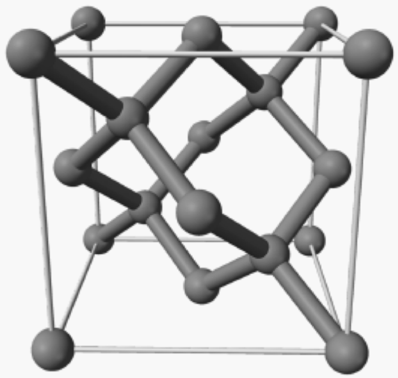
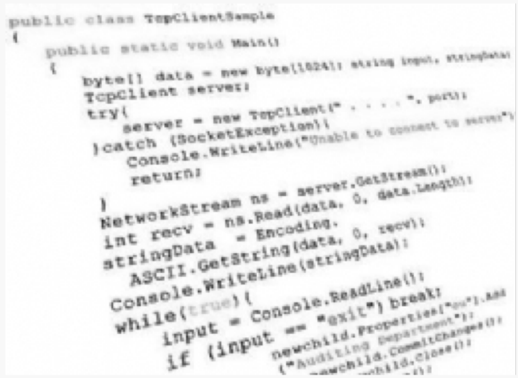
ES102

ÉLECTRONIQUE NUMÉRIQUE

Le lien entre Physique et Informatique

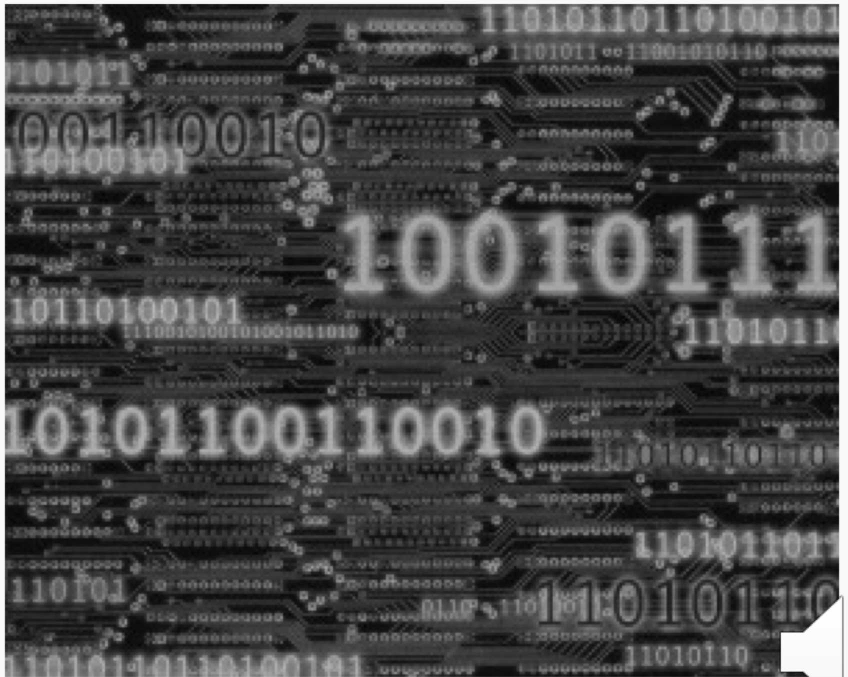


…



IMMERSION DANS LE MONDE BINAIRE

ES102 / CM1



BIT

anglais	*français*
digit	chiffre
digital	numérique

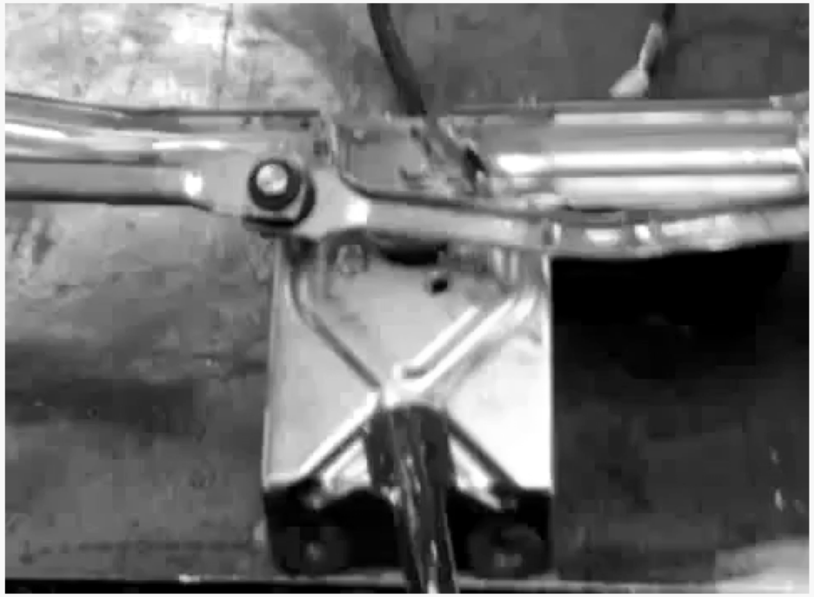
- Contraction de « BInary digiT »
- Grandeur binaire : valant 0 ou 1
  - notation : $\mathbb{B} = \{ 0, 1 \}$
- Bit universel,
  quelle sémantique ?
  - lien avec la logique :
    0 = faux - 1 = vrai
  - lien avec les ensembles :
    0 = rien - 1 = tout

Hors ES102 :
le *bit* comme unité d'entropie H

variables aléatoires
à 2 événements *a* et *b* :
p(a)=0,5 p(b)=0,5 $\rightarrow$ H=1 bit
p(a)=0,9 p(b)=0,1 $\rightarrow$ H$\approx$0,47 bit

- Une fonction à variables et valeur binaires est appelée *fonction booléenne* : $\mathbb{B}^n \rightarrow \mathbb{B}$

QUELQUES BITS CONCRETS



- moteur électrique → m
  - m=0 : arrêté
  - m=1 : tournant

sortie binaire

- commande → c
  (au tableau de bord)
  - c=1 : mise en marche
  - c=0 : demande d'arrêt

entrée binaire

- butée de fin de course → b
  - b=1 : en butée
  - b=0 : butée non atteinte
    (essuie-glace devant les yeux)

$\cancel{m = c}$

$$m = f(c, b)$$

fonction booléenne

à suivre…

POUR REPRÉSENTER LES NOMBRES …

- Notation décimale : $19{,}05 = 1\times10^{1} + 9\times10^{0} + 0\times10^{-1} + 5\times10^{-2}$

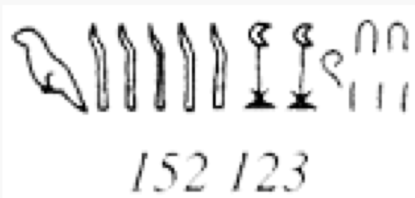
densité de l'Uranium

chiffre   poids : puissance de 10

→ notation dite *positionnelle* :

- car poids de chaque chiffre codé par sa position par rapport à la virgule
- nécessité du 0 !
- vertus : unicité, compacité, nb limité de symboles (autant que la base)
  - adoptée en Europe entre les X^e^ et XV^e^ siècles

- ∃ notations non positionnelles :
  - hiéroglyphes →
  - nombres romains



… DES BITS DE POIDS

- Représentation des entiers positifs par décomposition en base 2 + notation positionnelle

→ soient *n* bits b_0 à b_{n-1},

$$(b_{n-1} \cdots b_1 b_0)_2 \text{ représente l'entier } \sum_{i=0}^{n-1} b_i \cdot 2^i$$

chiffre

poids

- En informatique/électronique, *n* est fixé matériellement
  - *n* = puissance de 2 : 64 désormais sur ordinateur et smartphone
  - exemple sur 32 bits : $(5)_{10} = (00000000\,00000000\,00000000\,00000101)_2$

On n'oubliera pas les '0' à gauche quand on se contentera d'écrire $(5)_{10} = (101)_2$ ...

b_{n-1} appelé *MSB* pour *Most Significant Bit* (bit de poids fort) poids 2^{n-1}

b_0 appelé *LSB* pour *Least Significant Bit* (bit de poids faible) poids 1

PUISSANCES DE 2

- $2^0 = 1$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$

- $2^{10} = 1024$ (kilo)
- $2^{20} \simeq 1{,}05 \cdot 10^6$ (méga)
- $2^{30} \simeq 1{,}07 \cdot 10^9$ (giga)
- $2^{40} \simeq 1{,}10 \cdot 10^{12}$ (téra)

DES BITS POUR CODER

- Soit E un ensemble fini
  - exemple : les différents états possibles d’un système *discret*…
- souhait :
  - désigner chaque élément de E par un code individuel
  - codes chacun constitués de n bits

$\Rightarrow$ besoin d’une fonction de codage : $\gamma : E \to \mathbb{B}^n$

- nécessairement injective $\Rightarrow$ $2^n \geq |E|$ $|E|$ = cardinal de E
  - ex. : la notation binaire des nombres sur n bits est un codage bijectif de l’intervalle entier $[\![0, 2^n\text{-}1]\!]$ vers $\mathbb{B}^n$
- le mot *code* désigne *aussi* la fonction γ elle-même
  - tel le code de Gray ($\to$ PC1), bijectif lui aussi
- parfois besoin de la fonction de décodage : $\gamma^{-1} : \mathbb{B}^n \to E$
  - indéfinie (alias *non spécifiée*) sur les codes inutilisés

COMBIEN DE BITS POUR CODER ?

Le nombre minimal de bits nécessaires est n, tel que :

entropie d'un système à |E| états équiprobables

$$2^{n-1} < |E| \leq 2^n$$

$$\Leftrightarrow \quad n = \lceil \log_2(|E|) \rceil$$

direct mais utilise la partie entière supérieure $\lceil \; \rceil$

$\log_2(x)=\log(x)/\log(2)$

$\log_2(2)=1$

$\log_2(3)\approx 1{,}6$

$\log_2(4)=2$

…

$$\Leftrightarrow \quad 2^{n-1} \leq |E| - 1 < 2^n$$

$$\Leftrightarrow \quad n = \lfloor \log_2(|E| - 1) \rfloor + 1$$

plus compliqué mais utilise seulement la partie entière $\lfloor \; \rfloor$

On ne vise pas toujours le minimum pour n :

On utilise des codes redondants sur un canal de communication bruité (corruption possible des bits) → codage de can

ASCII : UN CODE DE CARACTÈRES

7 bits

$b_3b_2b_1b_0$ \ $b_6b_5b_4$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	*BS*	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	*LF*	SUB	*	:	J	Z	j	z
1011	VT	*ESC*	+	;	K	[	k	{
1100	*FF*	FS	,	<	L	\	l	|
1101	*CR*	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	*DEL*

ASCII :
American Standard Code for Information Interchange

BS = Back Space

LF = Line Feed

FF = Form Feed

CR = Carriage Return

ESC = Escape

DEL = Delete

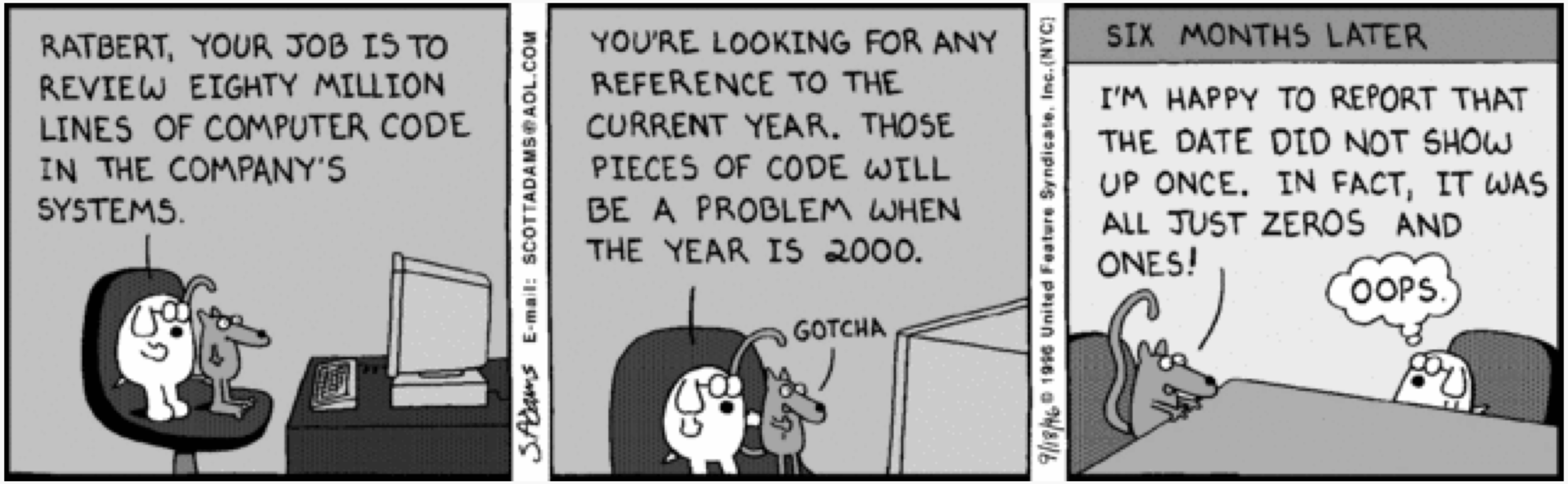
A l'origine, un 8ème bit purement redondant était utilisé

POURQUOI CODER AVEC DES BITS ?

- Exploitation ultime, en tout ou rien, de la physique :

TECHNOLOGIE	Etat 0	Etat 1
Transistor n	Passant vers masse	Non passant
Transistor p	Non passant	Passant vers alim.
DRAM et Flash	Capacité déchargée	Capacité chargée
ROM	Fusible grillé	Fusible intact
Disque dur	$\vec{B}$ dans un sens	$\vec{B}$ dans l'autre
DVD	Surface brute	Alvéole
Fibre optique	Pas de lumière	Lumière

- Meilleur compromis codage/coût $\rightarrow$ PC1/Exo4



Vous avez dit … sémantique ?

DE LA LOGIQUE AU MONDE BOOLÉEN

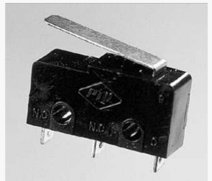
- a, b et s : variables binaires (liées)

Opérateur	Monde logique (faux/vrai)	Monde booléen (0/1)
non/complément	$(s=1) \Leftrightarrow (a=0)$	$s = a'$
ET logique	$(s=1) \Leftrightarrow (a=1) \land (b=1)$	$s = a \cdot b$
OU logique	$(s=1) \Leftrightarrow (a=1) \lor (b=1)$	$s = a+b$

- symboles de la multiplication et de l'addition « détournés » pour désigner ET et OU logique dans le monde booléen
  - car propriétés algébriques ressemblantes, à première vue…
    - 1 neutre pour · et 0 neutre pour +
    - $\land$ et $\lor$ aurait été un choix plus rigoureux, mais moins lisible

FORMULE BOOLÉENNE POUR ESSUIE-GLACE

c



b(utée)

m

$$m=1 \Leftrightarrow (c=1) \vee ((c=0) \wedge (b=0))$$

complément

$$m = c + c' \cdot b'$$

OU logique

ET logique

formule booléenne exprimant la fonction booléenne m(c,b)

c	b	m
0	0	1
0	1	0
1	0	1
1	1	1

expression concise, mais pouvant l'être encore plus…

$$m = c + b'$$

amalgame : formules booléennes souvent appelées « expressions logiques »

FONCTIONS BOOLÉENNES

$$f : \mathbb{B}^n \to \mathbb{B}$$

n variables binaires → $\mathbb{B}^n$

valeur binaire → $\mathbb{B}$

$\mathcal{F}_n$ = ensemble des fonctions booléennes à *n* variables

n=1

x	f
0	?
1	?

f(x)=x’ ← complément aussi noté $\overline{x}$ (voire /x)

x	x’
0	1
1	0

$$|\mathcal{F}_1| = 2^2$$

n=2

x	y	f
0	0	?
0	1	?
1	0	?
1	1	?

x·y

↓

f(x,y)=xy

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

$$|\mathcal{F}_2| = 2^4$$

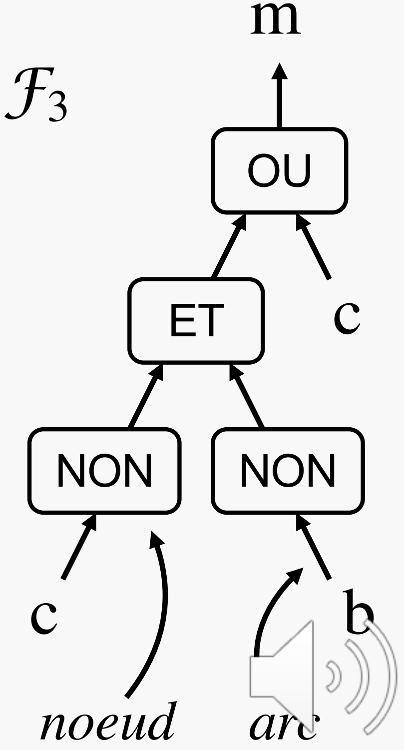
$$|\mathcal{F}_n| = 2^{2^n} \;!!$$

DÉCOMPOSER POUR POUVOIR RÉALISER

{ fonctions booléennes à *n* variables }

- Tout est fonction booléenne…
- Comment implante-t-on un élément arbitraire de $\mathcal{F}_n$?

  Par *décomposition* en fonctions plus simples, prenant la forme d'un graphe orienté :
  - dont les *nœuds* sont des éléments de $\mathcal{F}_1$, $\mathcal{F}_2$, voire $\mathcal{F}_3$
  - dont les arêtes/*arcs* sont des fils véhiculant des valeurs binaires d'un nœud à l'autre



⇒ Opportun d'explorer $\mathcal{F}_1$ et $\mathcal{F}_2$

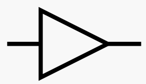
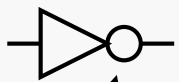
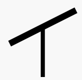
et même un peu $\mathcal{F}_3$ ...

- Diverses techniques de décomposition

  → avant-goût ci-après, étude approfondie au CM2

$\mathcal{F}_1$: 4 FONCTIONS BOOLÉENNES

"∈" $\mathcal{F}_0$ « x barre » "∈" $\mathcal{F}_0$

x					
	0	x	x' (alias $\bar{x}$)	1	**express.**
0	0	0	1	1	**valeurs**
1	0	1	0	1	
	0 logique	identité	complément	1 logique	**nom**
	logical 0	*identity*	*complement*	*logical 1*	***name***
	masse	tampon	inverseur / porte NON	alim.	**nom**
	ground (*Gnd / Vss*)	*buffer*	*inverter / NOT gate*	*power* (*Vdd*)	***name***
					**symbole**

fonction

composant / tension

Sur un symbole de porte, le *rond* représente la complémentation.

power = diminutif de « power supply »

$\mathcal{F}_2$: 16 FONCTIONS BOOLÉENNES

dont 10 "$\notin$"$\mathcal{F}_1$, dont 6 symétriques → opérateurs commutatifs

x	y	x·y	addition modulo 2		x⊕y	x+y	(x+y)’	(x⊕y)’			(x·y)’
0	0	0	0	0	0	0	1	1	1	1	1
0	1	0	0	1	1	1	0	0	0	1	1
1	0	0	1	0	1	1	0	0	1	0	1
1	1	1	0	0	0	1	0	1	1	1	0
nom fonction		ET	porte ET *AND gate*		OU eXclusif	OU	NON OU	NON OU eXclusif	*porte XNOR*		NON ET
name		*AND*			*XOR*	*OR*	*NOR*	X*NOR*			*NAND*
américain symbole du composant									*symboles améri-cains en ES102, car plus lisibles*		
européen		&			=1	≥1	≥1	=1			&

TABLES DE VÉRITÉ : RECHERCHE D’EFFICACITÉ

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Table 1D
standard

$a \oplus b$

b \ a	0	1
0	0	1
1	1	0

Table 2D
standard

$a \oplus b$

		a
	0	1
b	1	0

zone
où a=1

Table 2D
fonctionnelle

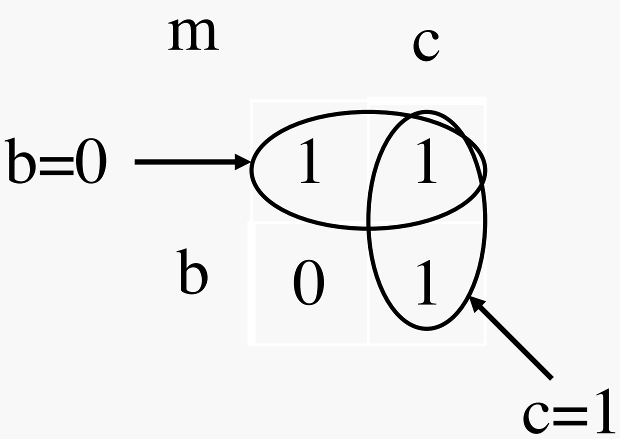
- La table dite *fonctionnelle*
  - exploite astucieusement les bords du tableau
  - des traits (épais) indiquent là où les variables d’entrée valent 1
    - au lieu d’afficher les valeurs des variables, pratique habituelle

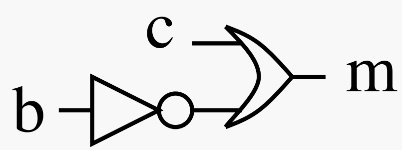
→ utilisation **obligatoire** en ES102 - autres tables **proscrites !**

DERNIER COUP D'ESSUIE-GLACE

m : moteur
c : commande
b : butée

c	b	m
0	0	1
0	1	0
1	0	1
1	1	1





équivalence logique : m=1 ⇔ (c=1) ou (b=0)

équation booléenne : $m = \underbrace{c + b'}$

Forme Minimale Disjonctive

FDM

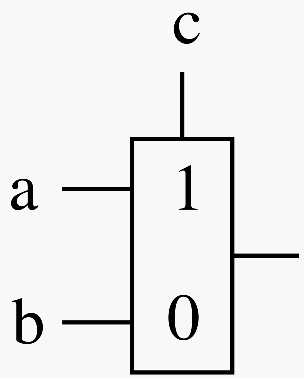
simple forme Σ

ΣΠ en général

ET : conjonction
OU : disjonction

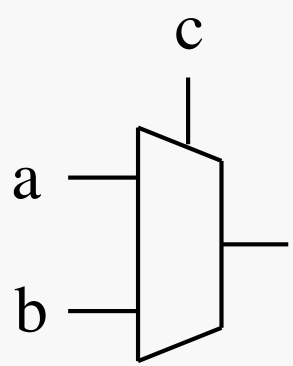
MUX, UN ÉLÉMENT MAJEUR DE $\mathcal{F}_3$

- Multiplexeur 2 vers 1, alias MUX (parfois MUX21)
- Très utile $\rightarrow$ CM2
- Symboles :
  - moderne :



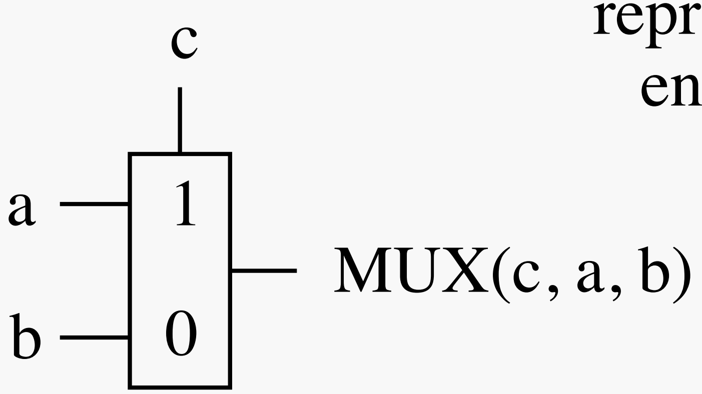
= a si c=1
MUX(c, a, b)
= b si c=0

  - classique :

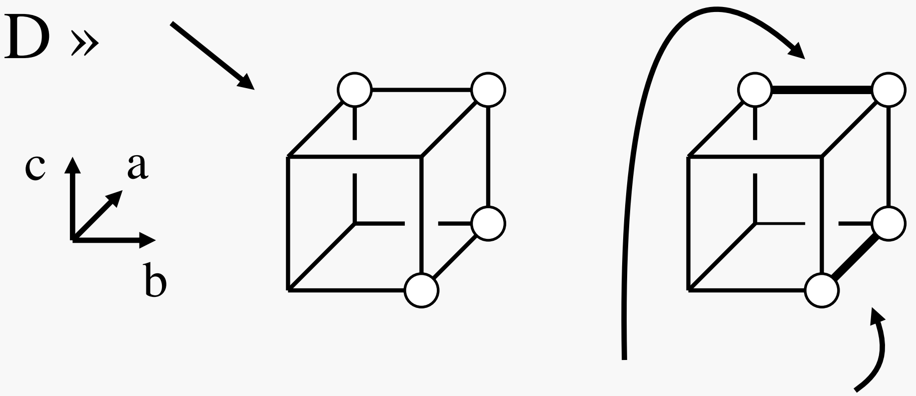


Les valeurs possibles pour le signal de commande *c* apparaissent à l’intérieur du rectangle et désignent l’entrée sélectionnée.

MUX : EXPRESSION / IMPLANTATION



représentable
en « 3D »

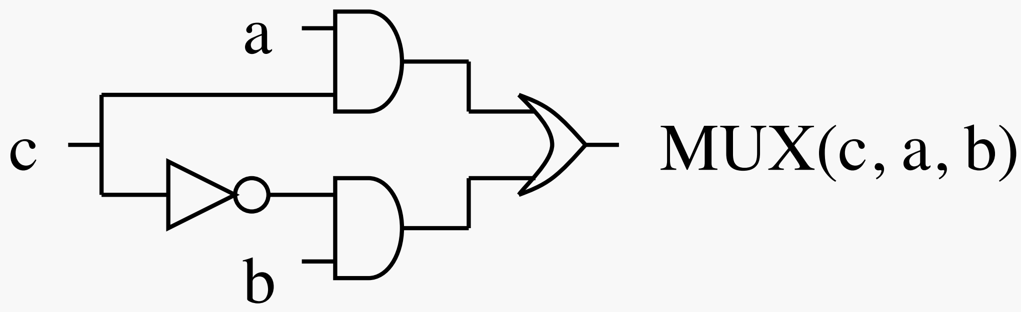


$$\mathrm{MUX}(c, a, b) = \underbrace{ca + c'b}_{\mathrm{FDM}}$$

forme
ΣΠ

- MUX $\in \mathcal{F}_3$
- Sa FDM

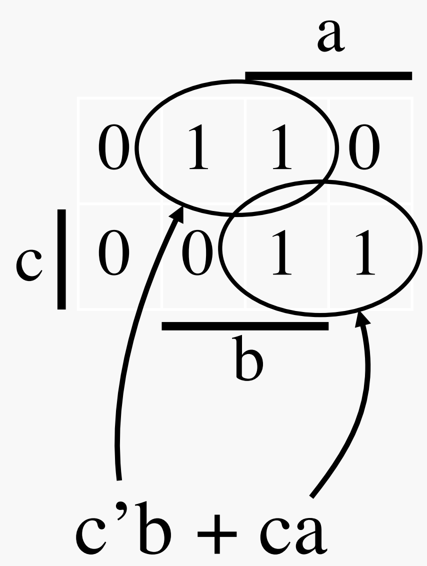
(Forme Disj. Min.)
la décompose en
éléments de $\mathcal{F}_1$ et $\mathcal{F}_2$



- Form(ul)e minimale $\Rightarrow$ encombrement minimal

MUX ET TABLE DE KARNAUGH

- 3 variables, donc 2 sur un même axe
  - ici *a* et *b* sur l'axe horizontal
  - chacune vaut 1 dans une zone rectangulaire *connexe*
    - disposition astucieuse, due à *Karnaugh*



→ table (fonctionnelle) de Karnaugh !

⇒ le couple (a, b) suit une progression horizontale originale :
(0, 0), (0, 1), (1, 1), (1, 0)

- c'est le *code de Gray* pour *n*=2 → PC1

≠ énumération binaire classique : 00, 01, 10, 11
qui aurait coupé la zone {b=1} en 2

ADDITION EN BASE 2 : $\mathcal{F}_3$ AUSSI

- A et B deux entiers exprimés sur *n* chiffres/bits :

tranche numéro *i*

retenues (*carry*)	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	s_n	s_{n-1}	s_{n-2}	…	s_{i+1}	s_i	…	s_1	s_0

en gras, pour ne pas confondre avec le OU logique +

- Pour un calcul particulier à la main, on ferait seulement apparaître les retenues non nulles. Mais, ici, les c_i sont des variables, présentes dans chaque *tranche*, même en i=0.

- $\Sigma_i = a_i \mathbf{+} b_i \mathbf{+} c_i \begin{matrix} \le 19 \\ \le 3 \end{matrix}$ en base $b = \begin{matrix} 10 \\ 2 \end{matrix}$

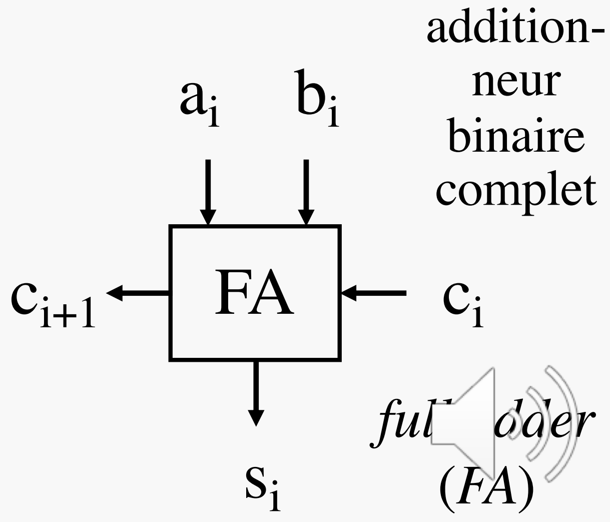
$$= (\,\cdot\;\cdot\,)_b$$

$$c_{i+1} = \Sigma_i / b \qquad s_i = \Sigma_i \% b$$

quotient et reste (modulo) de division euclidienne par *b*

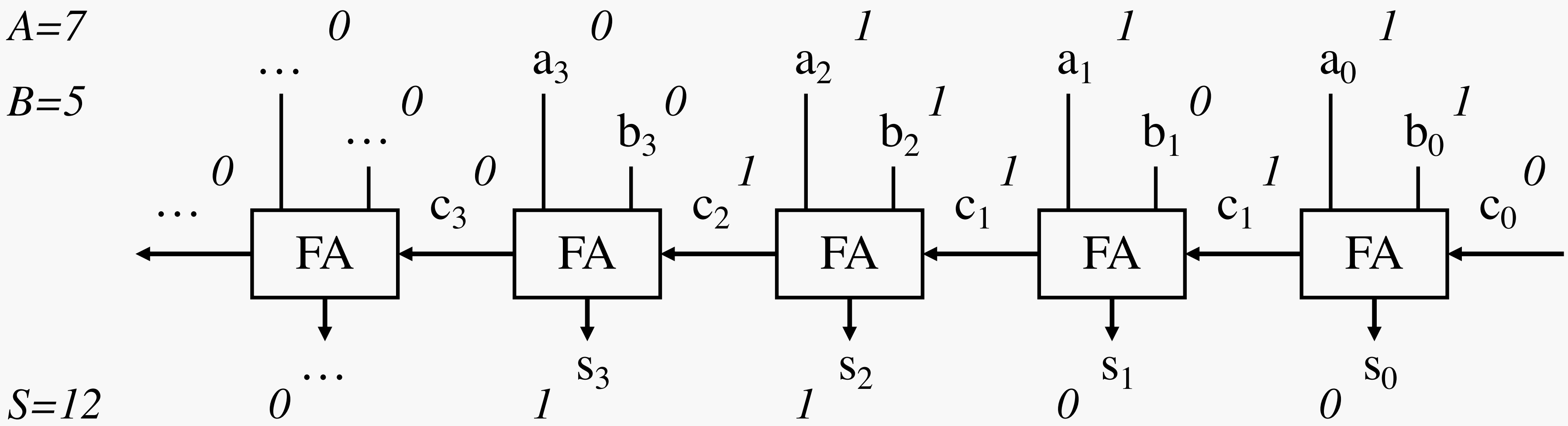
En base 2 :

$$\left.\begin{matrix} s_i\,(a_i, b_i, c_i) \\ c_{i+1}(a_i, b_i, c_i) \end{matrix}\right. \in \mathcal{F}_3$$



ADDITIONNEUR NUMÉRIQUE

mobilisant un Full Adder (FA) par tranche



- dit « à retenues propagées »
  - il en existe d'autres…
- équations booléennes du FA à établir $\rightarrow$ PC1