ELEC-H-305

Circuits logiques et numériques

2011-2012

Cours 2

Dragomir Milojevic

Dragomir.Milojevic@ulb.ac.be

Aujourd'hui

1. Codes
2. Codes correcteurs — parité
3. Algèbre de Boole
4. Fonctions logiques
5. Réalisation matérielle

1. Codes

Codes

- En décimal: tout nombre est composé de chiffre, chaque chiffre est décrit à l'aide des 10 symboles {0,9}.
- Pour représenter chaque chiffre en binaire: il faut $log_2 10$ de bits (en arrondissant sur le premier nombre entier supérieur: 4 bits)
- Lors de la représentation d'un chiffre à l'aide des 4 bits nous avons 16 codes, dont 6 inutiles (due au arrondissement):

1010, 1011, 1100, 1101, 1110, 1111

- Pour octal et hexadécimal c'est plus pratique, avec 3 et 4 bits respectivement il n'y a pas de codes inutiles (respectivement 8 et 16 symboles).

1. Codes

Trois classes de codes

1. Codes **pondérés** (*weighted codes*)
   - 8421 *Binary Coded Decimal* (BCD) — chaque chiffre codé séparément
   - 2421 Code
   - 642-3 Code excédent 3 (*Excess 3*)

   } Codes auto-complémentaires

2. Codes **non-pondérés**
   - Code Gray
   - Code ASCII

3. Codes **détecteurs d'erreur**

1. Codes pondérés

Codes pondérés

Tables

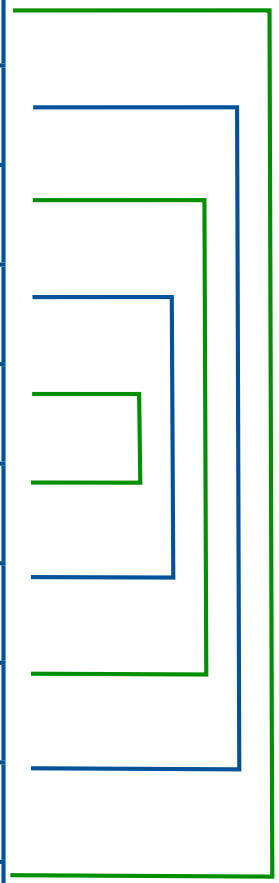
Auto-complémentaire (inversion)

Plusieurs codes possibles

642-3 - à titre d'exemple
(– est un signe, il s'agit d'un nombre < 0!**)**

Codes auto-complémentaires

Décimal	8421 (BCD)	2421	642-3
0	0000	0000	0000
1	0001	0001	0101
2	0010	0010	0010
3	0011	0011	1001
4	0100	0100	0100
5	0101	1011	1011
6	0110	1100	0110
7	0111	1101	1101
8	1000	1110	1010
9	1001	1111	1111



1. Code BCD

Aspect pratique : Addition en BCD

Exemple: $(532+268)_{10}$

```
 532        0101   0011   0010
+268       +0010   0110   1000
               1      1
_____________________________
 800        1000   1010   1010    Résultat ≥ 10 correction:+6(0110)

                  0110   0110
_____________________________
          (1000   0000   0000)2
           ↓      ↓      ↓


_____________________________
          (   8      0      0)10
```

On reporte!!! la dizaine
qui en binaire nécessite +6

1. Codes non-pondérés

Code Gray (code cyclique)

- Principe de la *Look-up Table:* la conversion ne suis pas une règle, on peut faire ce que l'on veut...
- **Code Gray** — deux codes voisins ne diffèrent que par la valeur d'un seul bit (pour éviter des probs.).
- Lors de transitions (passage d'une valeur à l'autre — le comptage p.e.) il n'y a pas deux bits qui changent de valeur en même temps.
- Plusieurs codes sont possibles (on peut imaginer un algorithme permettant de générer de tels codes).

Décimal	Gray
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

1. Codes non-pondérés

Code ASCII

(***American Standard Code for Information Interchange***)

- Utilisé pour coder les caractères dans des systèmes de traitement d'information numérique.
- C'est un code sur 8 bits: donc 256 possibilités (2^8), mais en pratique on n'utilise que 128.
- Utilisé pour coder les lettres en MAJ et MIN, chiffres, ponctuation, caractères de contrôle...
- Attention à l'interprétation des codes (caractères ou les nombres)

Différence entre un fichier texte et un fichier binaire?

1. Codes non-pondérés: ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

a ou 97 ?

2. Codes détecteurs d'erreur

Distance de *Hamming*

La distance de Hamming quantifie la distance (la différence) entre deux séquences de symboles.

Exemple

```
La distance entre 0101001 et

                  0111100 est de 3.
```

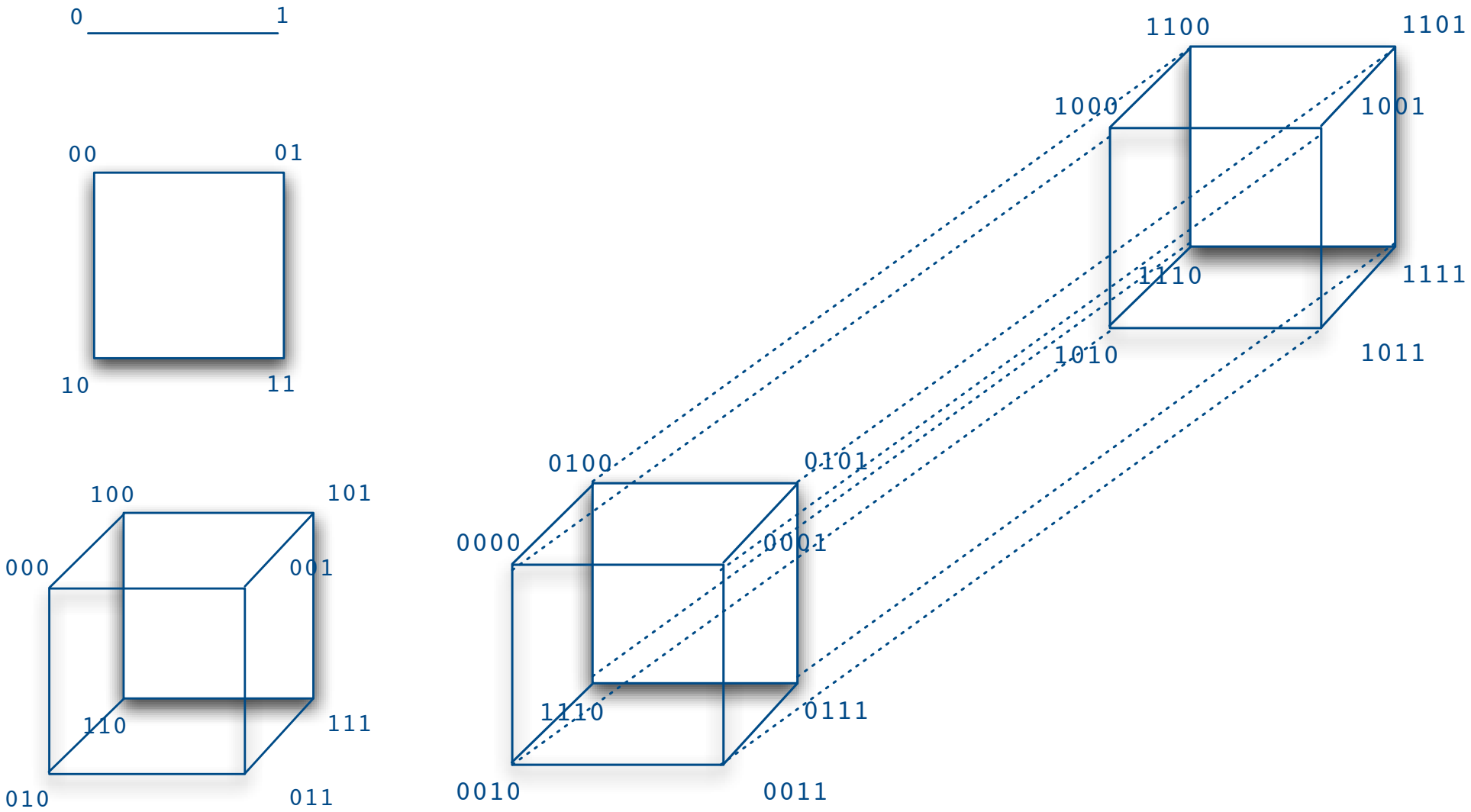
Un *n*-cube est un cube à *n*-dimensions avec 2^n sommets.

Un *n*-bit string est représenté comme un des sommets binaires d'un *n*-cube.

Sommets adjacents ont une distance de Hamming de 1.

2. *n*—cubes

Représentation graphique des *n*—cubes pour *n* = 1, 2, 3 et 4 ...



2. Codes détecteurs d'erreur

Principe général

- Avec un mots de *n*-bits on peut coder 2^n mots différents.
- On peut imaginer utiliser que la moitié des mots pour les **codes corrects** et l'autre moitié pour les **codes erronés**.
- En fait pour un mot de *n*-bits, en ajoutant 1 bit en plus nous avons *n* mots supplémentaires que l'on pourrais qualifier d'erronés.
- Pour un code détecteur d'erreur - dans certains conditions (erreur de transmission p.e.):

Le **code correct** est **transformé** en **code erroné**.

2. Exemple

Codes de parité paire ou impaire (*Even (odd) parity codes*)

- Un mot de *n*-bits est codé à l'aide de (n+1) bits
- Nous avons le même nombre de mots corrects et le même nombre de mots erronés.
- Critère de correction représenté par le bit (n+1):
  - On compte le nombre de '1' dans le mot, le bit de parité y compris — on a deux situations possibles:
    - **Bit de parité paire — '1' → si c'est un nombre paire**
    - **Bit de parité impaire — '1' → si c'est un nombre impaire**

2. Parité paire et impaire

Exemple

Pour chaque **mot** on compte le nombre des ‘1’ (bit de parité y compris)

0 ‘1’ bit parité impaire =1

1 ‘1’ bit parité paire =1

2 ‘1’ bit parité impaire =1

.

.

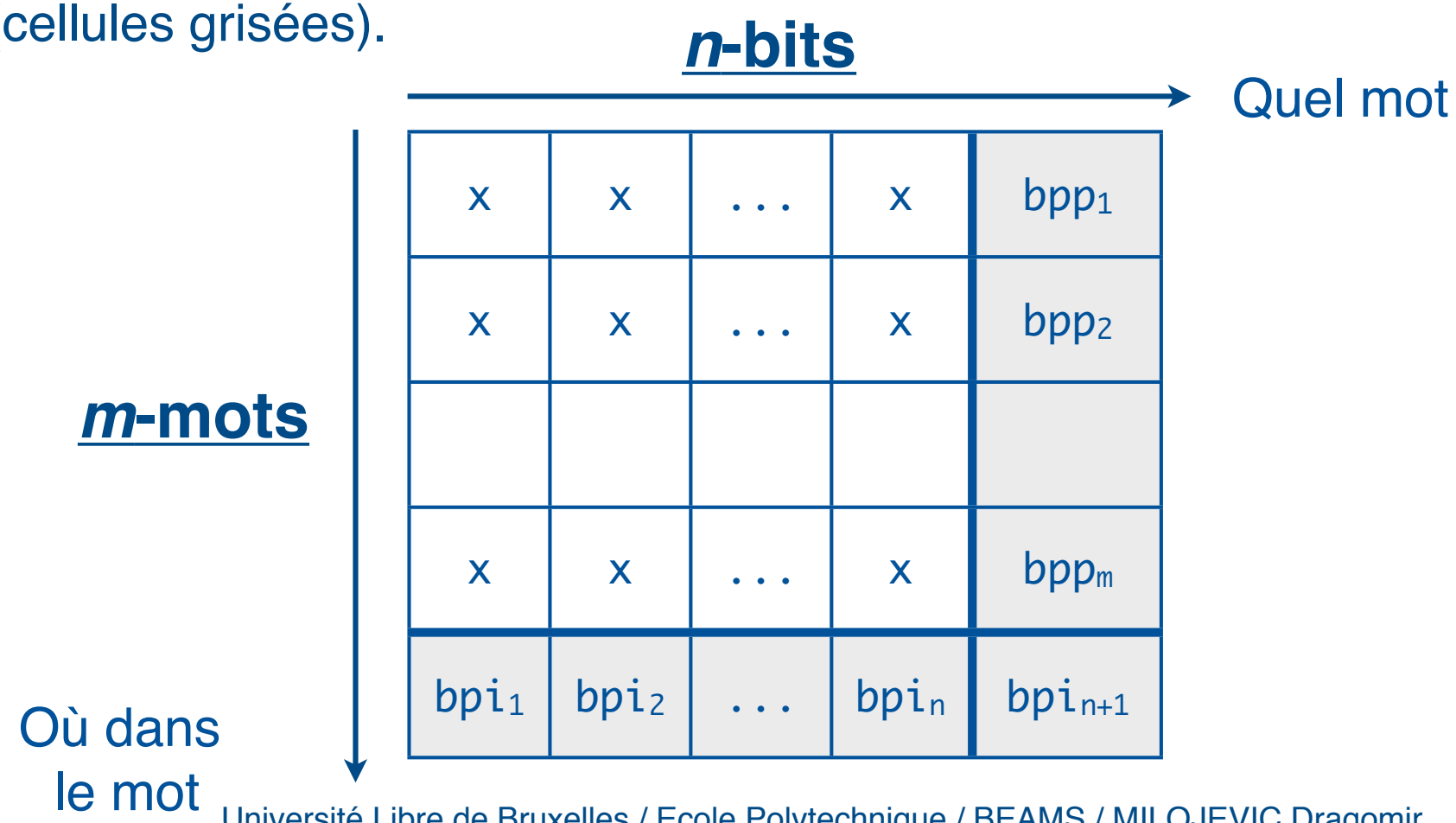
.

b_4	b_3	b_2	b_1	b_0	Paire	Impaire
0	0	0	0	0	0	1
0	0	1	0	0	1	0
1	1	0	0	0	0	1
0	1	1	1	0	1	0
1	0	1	1	1	0	1
1	1	1	1	1	1	0

2. Exemple d'utilisation

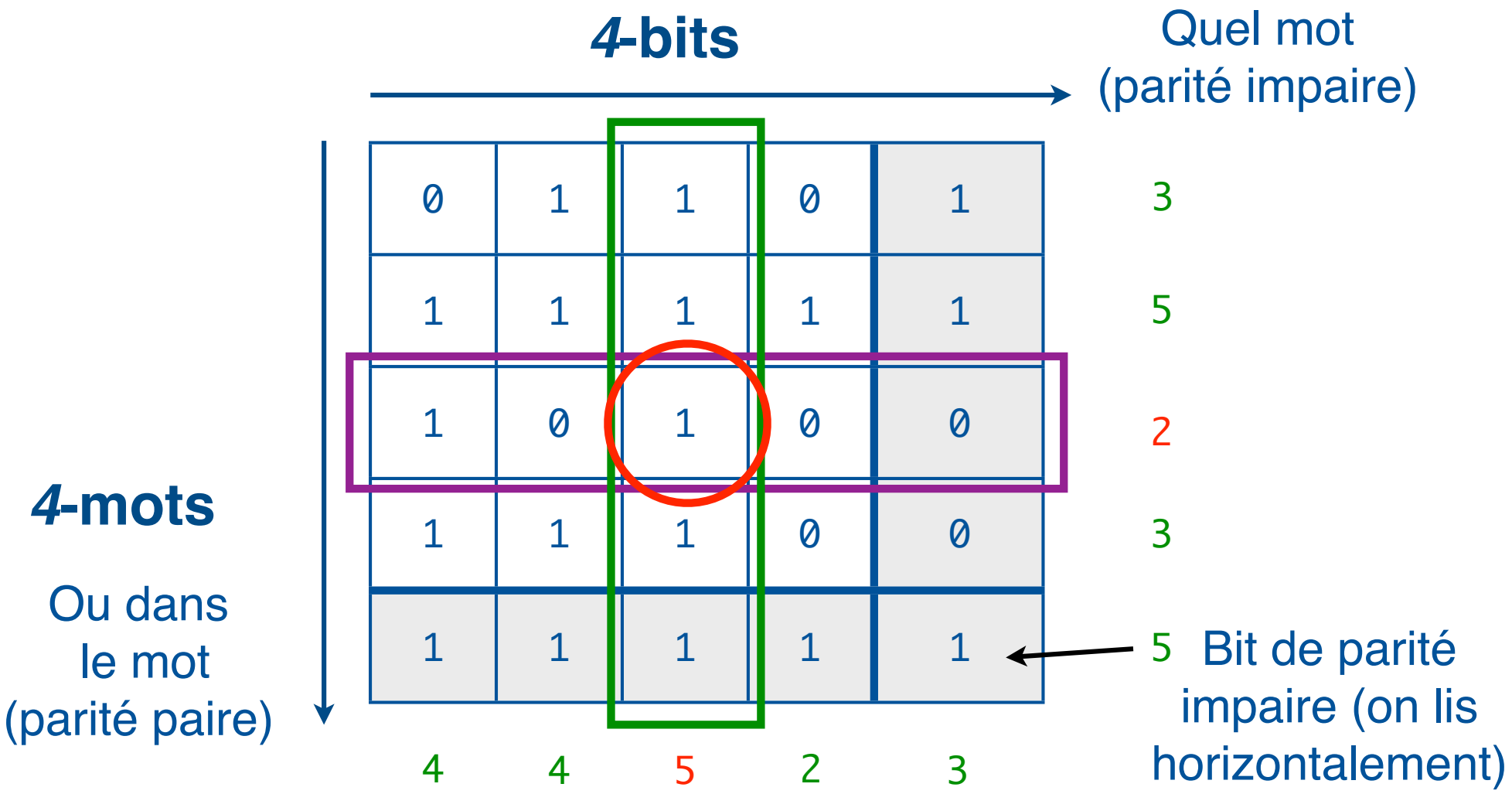
On veut transmettre *m*-mots de *n*-bits

On ajoute à chaque mot un bit de parité (les mots à transmettre sont alors de (n+1) bits). A la fin de la transmission, on ajoute un mot de (n+1) bits de parité, calculés sur tous les bits des mots transmises ayant le même poids (cellules grisées).



2. Exemple concret

On veut transmettre *4*-mots de *4*-bits — 25 bits en tout



3. Algèbre de Boole

Repères historiques

1847 - G. Boole :
"The mathematical analysis of logic"

1904 - E. V. Huntigton :
Définition formelle d'algèbre de Boole, telle que l'on la présente aujourd'hui

1938 - C. E. Shannon :
Montre le lien entre l'algèbre de Boole à deux valeurs et les propriétés de certains circuits électriques. Naissance de l'algèbre de commutation (*Switching Algebra*)

> *"Shannon's thesis is possibly the most important, and also the most famous, master's thesis of the century"*.
>
> Howard Gardner, Harvard University

3. Algèbre de Boole

Définition de l'algèbre de Boole

- Algèbre de Boole est un quadruplet $\{B, ', \bullet, +\}$, où:
  - B est un ensemble de deux éléments
  - $'$ est l'opérateur de complément (parfois barre verticale au dessous de symbole)
  - $\bullet$ est l'opérateur ET
  - $+$ est l'opérateur OU
- En fonction de la définition de l'ensemble B et des 3 opérateurs, on peut définir **plusieurs** algèbres de Boole.

3. Algèbre de Boole

Cas à 2 valeurs

Parmi les différentes Algèbres de Boole nous nous intéressons à celle introduite par Shannon.

Algèbre de Boole à **deux valeurs** (*Two-valued Boolean algebra*):

1. B={0,1} où '0' - faux et '1' - vrai
2. Deux opérateurs binaires :

\+ OU inclusif (OR) et • ET (AND)

Opérateurs : Représentation par **Table de Vérité**

Attention

- **et + ne sont pas des opérateurs arithmétiques !**

3. Algèbre de Boole: Axiomes

Pour qu'un quadruplet $\{B, ', \bullet, +\}$ soit une algèbre de Boole, les axiomes suivantes doivent être satisfaites:

Axiome 1. B est **fermé** pour + et pour •

Axiome 2. B a un **neutre** pour + (noté 0) et pour • (noté 1)

Axiome 3. B est **commutatif** par rapport à + et •

Axiome 4. • **distribue** + et + distribue •

Axiome 5. **Complément** de x

Axiome 6. Il y a au moins 2 éléments x,y du B tels que $x \neq y$

Associativité est une conséquence de ce qui précède:

$$(a+b)+c=a+(b+c) \text{ et } (a\bullet b)\bullet c=a\bullet(b\bullet c)$$

3. Algèbre de Boole

Cas à 2 valeurs - opérateurs ET et OU

Les opérateurs • et + définis par des **Tables de Vérités (TdV)** suivantes:

x	y	x•y
0	0	0
0	1	0
1	0	0
1	1	1

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

On peut vérifier les 6 axiomes posés par E. V. Huntigton pour le cas de l'algèbre de Boole à deux valeurs B={0,1}

3. Axiomes

Axiome 1 : B={0,1} est fermé

Le résultat de chaque opération est bien dans le B.

En examinant les tables de vérités pour les opérateurs +, •, ‘ :

x	x’
0	1
1	0

x	y	x•y
0	0	0
0	1	0
1	0	0
1	1	1

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

3. Axiomes

Axiome 2 : Neutre (pour les valeurs de B)

0 et 1 sont les neutres pour + et ● respectivement

On peut vérifier à l'aide de la définition des opérateurs + et ● (cf. les TdV des opérateurs).

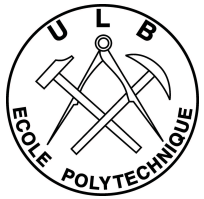
Pour + :

$$0 + 0 = 0$$
$$1 + 0 = 1$$

Pour ● :

$$0 \bullet 1 = 0$$
$$1 \bullet 1 = 1$$

3. Axiomes



Axiome 3 : Commutativité

+ et • sont commutatifs

+ est commutatif, on peut vérifier:

$$x + y = y + x$$
$$0 + 1 = 1 + 0$$
$$1 = 1$$

• est commutatif:

$$x \bullet y = y \bullet x$$
$$0 \bullet 1 = 1 \bullet 0$$
$$0 = 0$$

3. Axiomes

Axiome 4 : Distributivité de • par rapport à + [a]

$$x \bullet (y+z) = x \bullet y + x \bullet z$$

x	y	z	y+z	x•(y+z)	x•y	x•z	x•y+x•z
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

[a] **Preuve par table de vérité**

3. Axiomes

Axiome 4 : Distributivité de + par rapport à •

$$x+(y \bullet z)=(x+y) \bullet (x+z)$$

x	y	z	y•z	x+(y•z)	x+y	x+z	(x+y)•(x+z)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

3. Axiomes

Axiome 5: Compléments

‘0’ et ‘1’ sont des compléments respectifs:

0 + 0’ = 0 + 1 = 1

0 + 1’ = 0 + 0 = 0

0 • 0’ = 0 • 1 = 0

0 • 1’ = 0 • 0 = 0

Le complément introduit un troisième opérateur : NON (NOT)

x	y=NOT(x)
0	1
1	0

3. Axiomes

Axiome 6 :
Il y a au moins 2 éléments x,y du B tels que x ≠ y

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

Il y a deux éléments différents dans le $B=\{0,1\}$.

Et $0\neq1$, $1\neq0$

3. Théorèmes

Les 7 théorèmes de base de l'algèbre de Boole

On peut les prouver à l'aide des Axiomes ou des TdV.

- **Th1:** Idempotence pour + et •

  $x+x=x$ et $x \bullet x=x$

- **Th2:**

  $x+1=1$ et $x \bullet 0=0$

- **Th3:** Absorption

  $x \bullet (x+y)=x$

3. Théorèmes

- **Th4:** Involution

  (x')'=x

- **Th5:** Associativité

  (x+y)+z=x+(y+z) et (x•y)•z=x•(y•z)

- **Th6:** Lois de De Morgan

  (x+y)'= x' • y' et (x•y)' = x' + y'

- **Th7:** Consensus

  x•y + x'z + y•z = x•y + x'z

3. Démonstrations des Ax./Th.

- En partant d'un énoncé chaque étape de démonstration est franchi à l'aide d'un des axiomes ou des théorèmes déjà prouvé
- On précise à chaque étape l'axiome ou le théorème appliqué
- Exemple:
  Preuve de **Th1**

x+x = (x+x)•1	Ax2. Neutre
= (x+x)(x+x')	Ax5. Complément
= x+x•x'	Ax4. Distributivité
= x+0	Ax5. Complément
= x	Ax2. Neutre

3. Démonstrations par TdV

- Algèbre de Boole à deux variables:
  - Théorèmes sont des problèmes combinatoires:
    on peut prouver un énoncé à l'aide d'une **Table de Vérité**
- Nombre de variables détermine le nombre de lignes de la TdV
- Pour chaque côté de l'expression nous avons une TdV (2 donc)
- Deux côtés de l'égalité doivent avoir les mêmes valeurs: on compare les deux TdV
- L'identité de deux colonnes preuve le théorème

3. Exemple de démonstration

Théorème de De Morgan

$(x+y)' = x' \bullet y'$

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

3. Exemple de démonstration

Théorème de De Morgan

$(x \bullet y)' = x' + y'$

x	y	x•y	(x•y)'	x'	y'	x' + y'
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

3. Principe de dualité

- Dans une algèbre de Boole, tout résultat se présente sous deux formes **duales**.
- Soit `S` un résultat, son dual `S*` s'obtient en **permutant** systématiquement :
  - les opérateurs `+` et `•`
  - les symboles `0` et `1` de `B`

Si un résultat S est vrai dans une algèbre de Boole, il en est de même pour son dual!

3. Principe de dualité

Exemples

- Soit S le résultat suivant :

  $\forall x,\ x \in B\ ,\ x+x=x\ ,$ **règle d'idempotence**

  son dual S* est :

  $\forall x,\ x \in B\ ,\ x \bullet x = x$

- De Morgan:
  - Puisque:
    $(x+y)' = x' \bullet y'$ est vraie
  - alors
    $(x \bullet y)' = x' + y'$ est vraie aussi

3. Principe de dualité

Généralisation

On peut généraliser pour une expression Booléenne à n – variables.

$$\forall\ x_i,\ x_i \in B\ ,\ E(x_1,\ \ldots,\ x_n) \rightarrow E^*(x_1,\ \ldots,\ x_n)$$

Exemple d'application:

Extension de De Morgan sur n – variables

$$(x_1 + \ldots + x_n)' = x_1' \bullet \ldots \bullet x_n'$$

et aussi

$$(x_1 \bullet \ldots \bullet x_n)' = x_1' + \ldots + x_n'$$

3. Algèbre de Boole

Algèbre à 2 valeurs

C'est une algèbre complète:

- B={0,1}
- Deux opérateurs:
  - ET (AND)
  - OU (OR)
  - plus 1 opérateur dérivé NON (NOT)
- Pour chaque opérateur on construit un composant physique.
- Assemblage des différents composants (portes) permet la réalisation physique des expressions.
- Plusieurs modes de représentation:
  A. Fonctions Logiques
  B. Logigrammes
  C. Tables de vérité

4. Fonctions logiques

- **Variables** a, b, c, ...
- **Opérateurs** ET, OU, NON (AND, OR et NOT)
- **Evaluation** =
- **Parenthèses**
  sens de gauche à droite, depuis les “entrées” jusqu’aux “sorties”
  - **Entrées:** arguments d’une fonction logique
  - **Sorties:** évaluation de(s) la(es) fonction(s)
- Une fonction logique pour chaque variable de sortie. On peut avoir dans un même système plusieurs sorties, chaque sortie aura une fonction logique qui lui est propre.

4. Fonctions logiques

Formes de représentation des fonctions logiques:

A. **Fonctions** — pratique pour une manipulation manuelle, “crayon et papier”; représentation compacte pour un faible nombre de variables.

B. **Schématique** — représentation graphique; proche du monde de la réalisation physique; facile à comprendre (mais peut être très compliqué pour un grand nombre arguments); beau à voir.

C. **Tables de Vérité** — Représentation tabulaire, proche de la représentation de type mémoire numérique (style *Look-up-Table*).

Quelque soit la forme initiale, la fonction peut être transformée d’une forme à l’autre, même de façon automatique.

4. Fonctions logiques

Exemple de formalisme

F = xz+xy'z'+x'yz'

= x•z+x•y'•z'+x'•y•z'

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

= OU(ET(x,z), ET(x,y',z'), ET(x',y,z'))

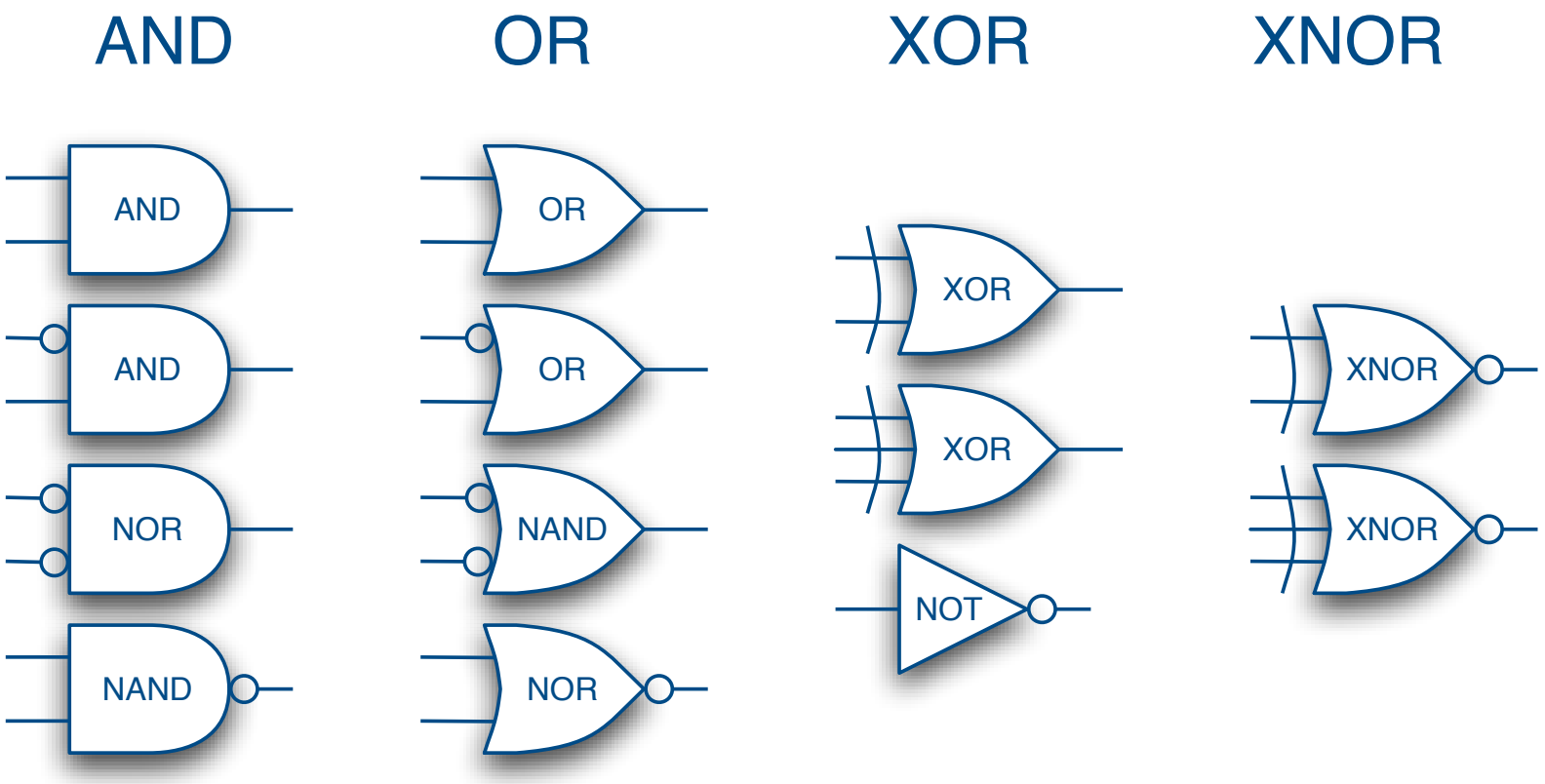
= OU(ET(x,z), ET(x,NON(y), NON(z)), ET(NON(x),y, NON(z)))

Le résultat est au format de type **Somme de Produit — SdP** (un opérateur **OU** qui réuni les différents termes **ET**).

4. Logigrammes

Schémas : assemblages des opérateurs élémentaires

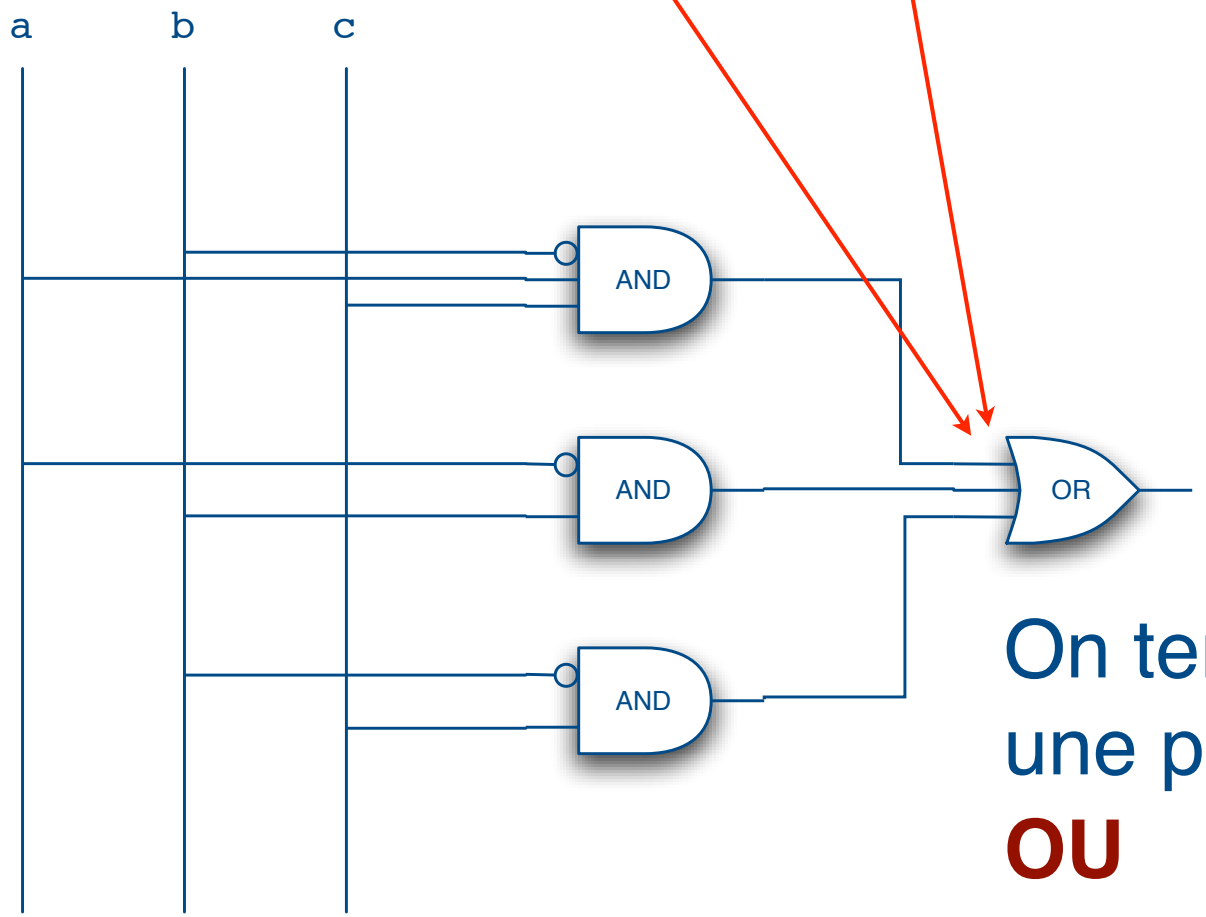
Symboles des opérateurs élémentaires par classe:



4. Logigrammes

Un circuit sous forme de **Somme de Produits - SdP** $(\sum\pi)$

$$s = (a \cdot \overline{b} \cdot c) + (\overline{a} \cdot b) + (\overline{b} \cdot c)$$

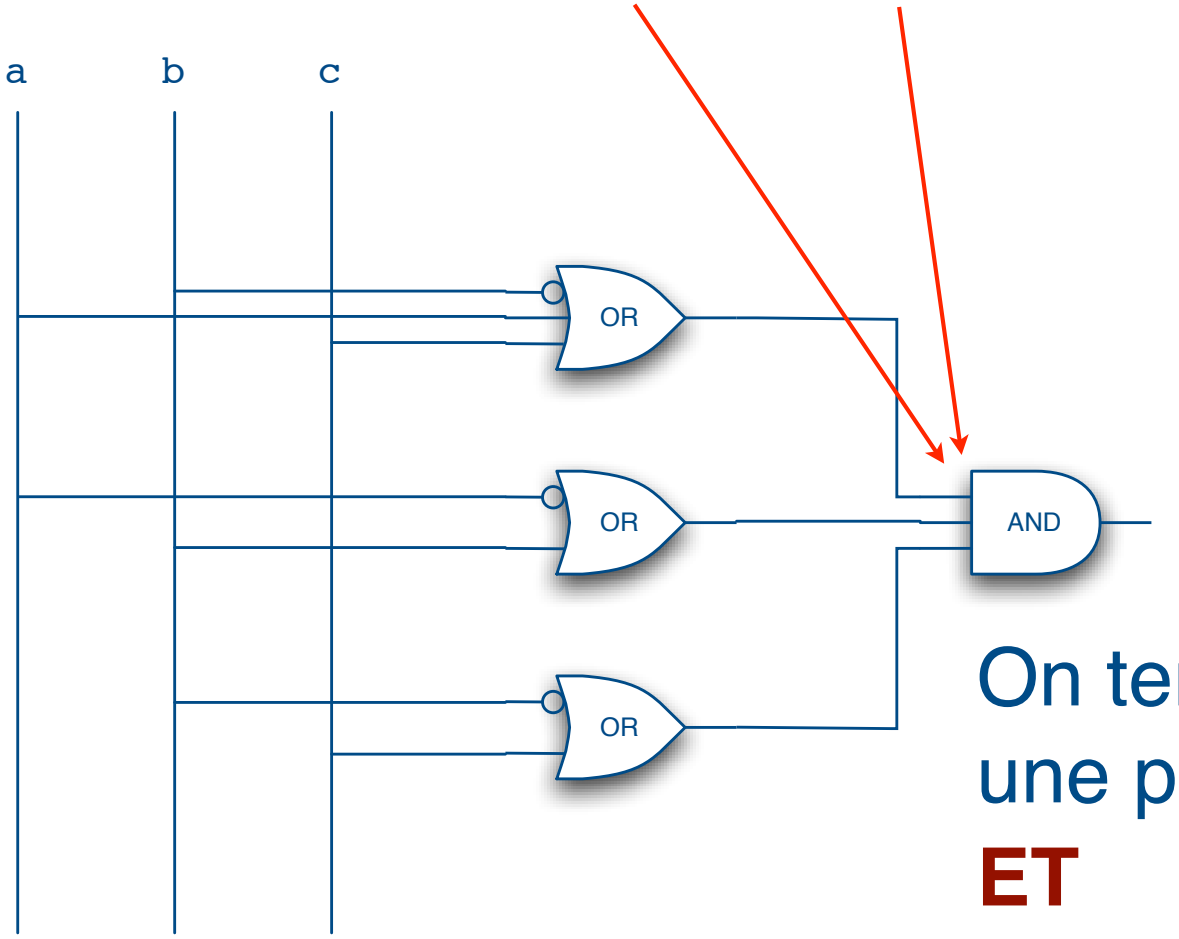


On termine par une porte **OU**

4. Logigrammes

Un circuit sous forme de **Produit de Sommes - PdS** $(\pi\Sigma)$

$$s = (a + \overline{b} + c) \cdot (\overline{a} + b) \cdot (\overline{b} + c)$$



On termine par une porte **ET**

4. Tables de Vérité

Table de Vérité (TdV) et Somme de Produits

x	y	z	F	
0	0	0	1	+
0	0	1	1	+
0	1	0	0	
0	1	1	1	+
1	0	0	0	
1	0	1	0	
1	1	0	1	+
1	1	1	1	+

F vaut 1 lorsque x ET y ET z valent:

0 ET 0 ET 0 **ou**

0 ET 0 ET 1 ...

SdP

F = x'y'z' + x'y'z + x'yz + xyz' + xyz

4. Tables de Vérité

Table de vérité (TdV) et Produit de Somme

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

En partant à l'envers: F vaut 1 lorsque xyz ne valent pas 010 ...

F = (x'yz')'
(xy'z')'
(xy'z)'

PdS

F = (x+y'+z)(x'+y+z)(x'+y+z')

5. Réalisation matérielle

- Composants **physiques** (électroniques) permettant la réalisation **matérielle (*hardware*)** des portes logiques (et donc des fonctions logiques)
- Différentes technologies: relais (électro-mécaniques), tubes électroniques, transistors, ... A partir du moment où le composant est capable de jouer le rôle d'interrupteur on peut réaliser les différentes portes logiques.
- Au départ ces composants ont été crées pour des besoins “électriques”:
  - **Diode — Redressement** : Transformer l'alternatif en continu
  - **Transistor — Amplification** : Augmenter l’amplitude de la tension (courant)

5. Réalisation matérielle

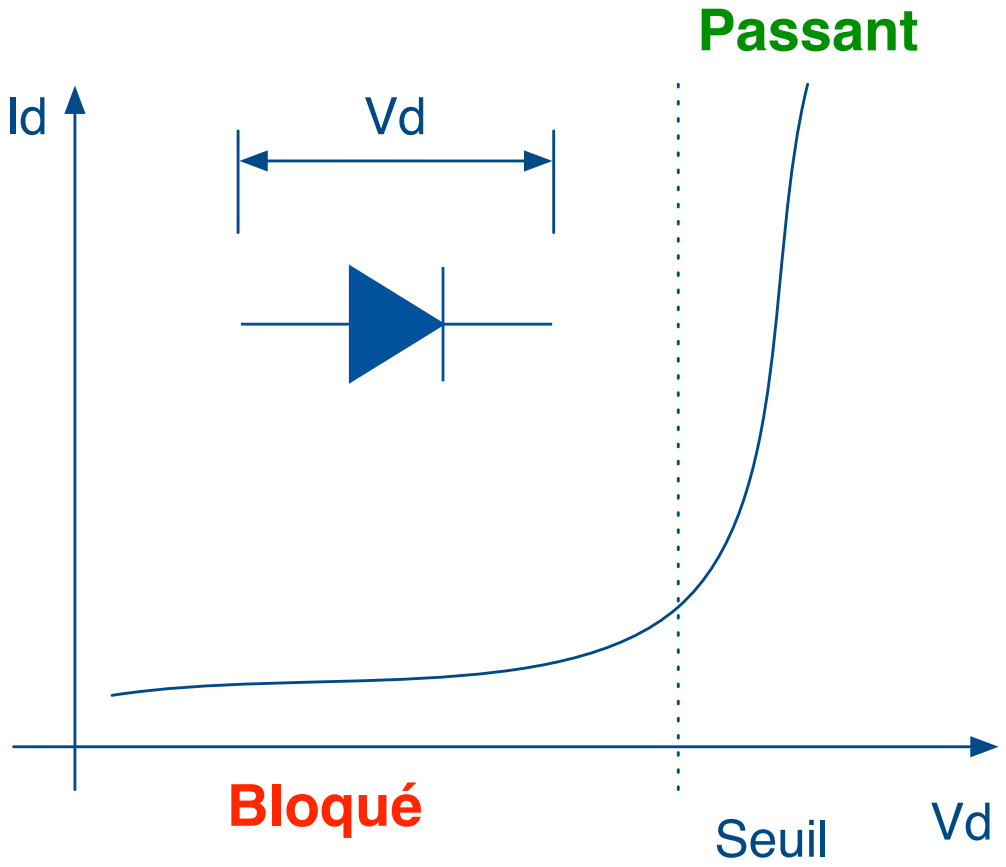
Le choix de la technologie de fabrication de CI a un impact sur certains nombre de paramètres, toujours d'actualité:

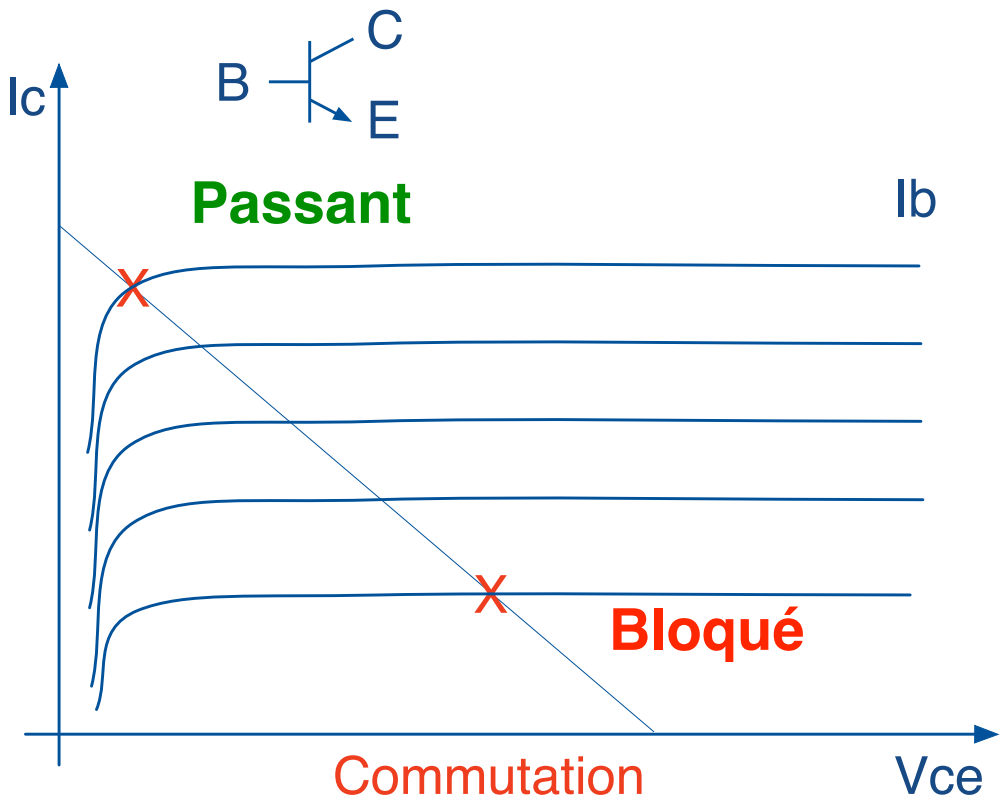
1. **Espace utilisé - encombrement**
   En fonction de la technologie le nombre de portes logiques:
   - SSI - *Small Scale Integration*: 1 à 10 portes par circuit
   - MSI - *Medium Scale Integration*: 10 à 100
   - LSI - *Large Scale Integration*: 100 à 100 000
   - VLSI - *Very Large Scale Integration*: entre 100 000 à 1 million
   - Actuellement ~ milliard
2. **Fréquence maximale de commutation**
   Vitesse de circuit = "vitesse de calcul de la fonction logique"
3. **Puissance dissipée**
   Lié à la consommation (écolo, autonomie) et refroidissement

5. Réalisation matérielle

Diode: un interrupteur

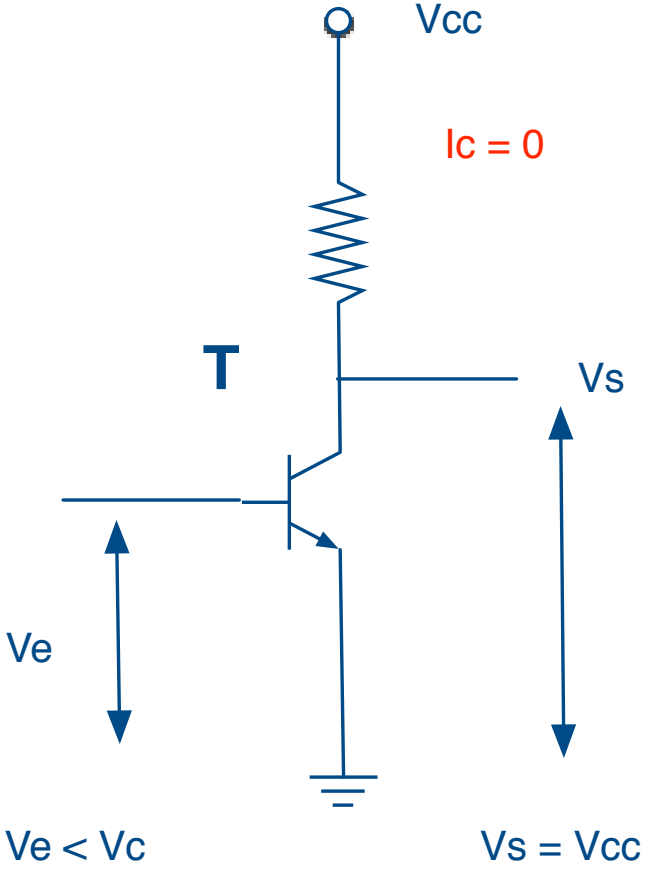
Transistor: amplificateur qui peut fonctionner comme un interrupteur



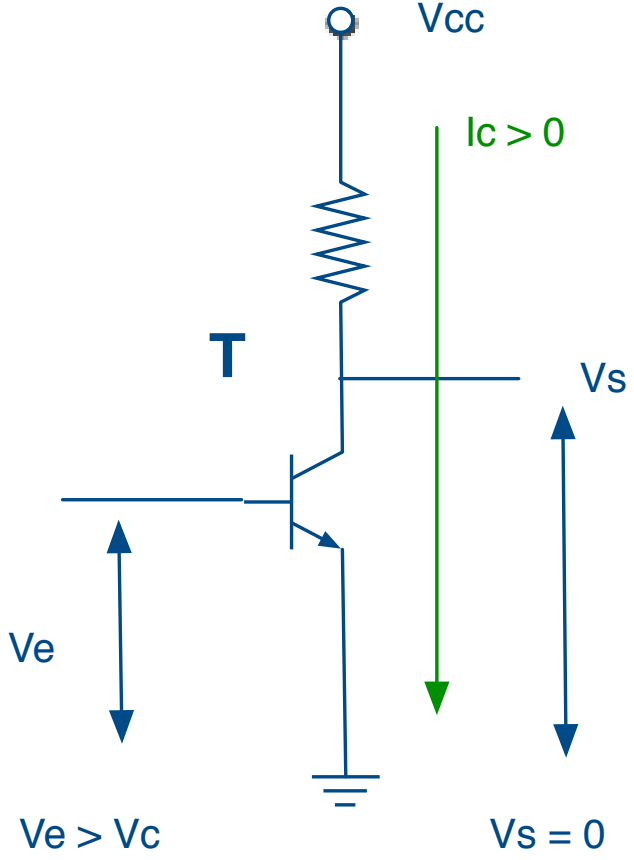


5. Réalisation matérielle

Montage: Transistor comme interrupteur



T est bloqué



T est passant

Vcc - Alimentation

Ve - Tension “Entrée”

Vs - Tension “Sortie”

Vc - Seuil

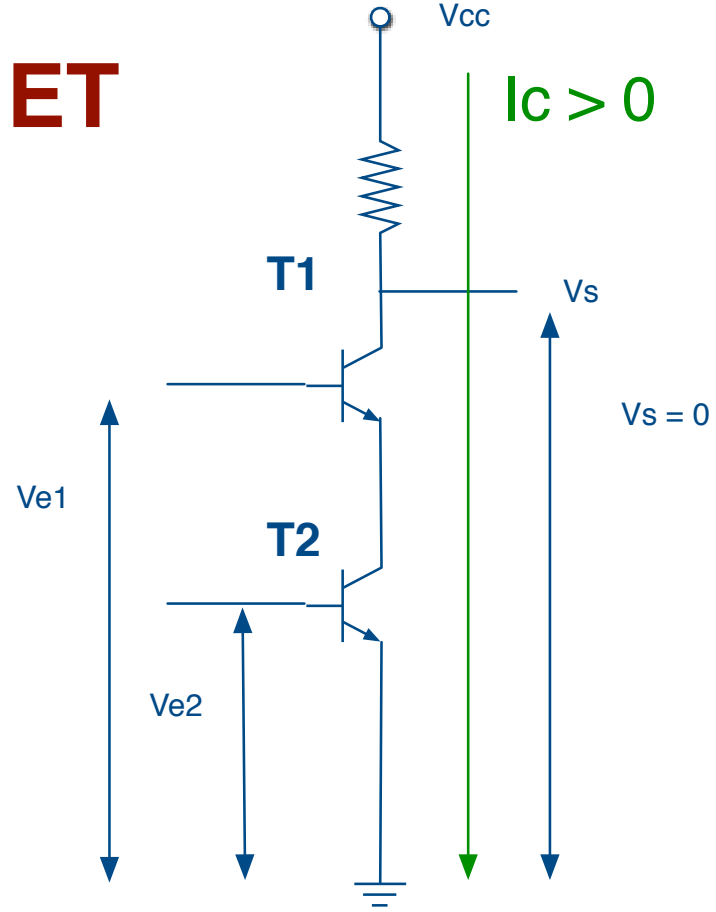
Vs=Vcc - ‘1’ logique

Vs=0 - ‘0’ logique

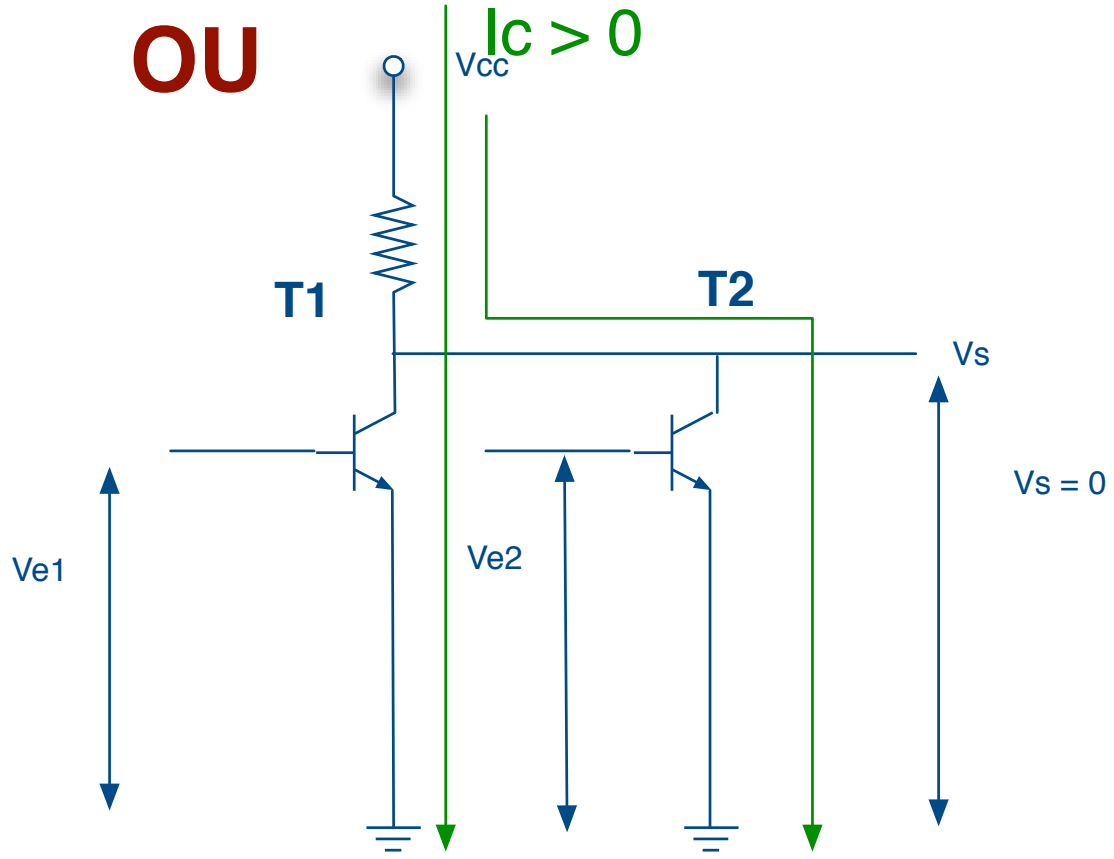
5. Réalisation matérielle

Portes logiques à l'aide des transistors: NAND et NOR

Mise en série et parallèle de 2 transistors.



Ve1 ET Ve2 > Vc



Ve1 OU Ve2 > Vc