

# ARCHITECTURE DES MICROPROCESSEURS

# Programme

- Première partie
  - algèbre de Boole
  - circuits logiques
- Deuxième partie
  - L'Unité Centrale du microprocesseur
  - La Mémoire Centrale du microprocesseur
  - Les Périphériques et interfaces associés au microprocesseur
  - Le logiciel de base

# Programme

- Première partie
  - algèbre de Boole
  - circuits logiques
- Deuxième partie
  - L'Unité Centrale du microprocesseur
  - La Mémoire Centrale du microprocesseur
  - Les Périphériques et interfaces associés au microprocesseur
  - Le logiciel de base

# Première partie

## • **Fondements théoriques et technologiques**

*Ordinateur ← Circuits de base très simples dont le comportement fonctionnel est décrit par l'algèbre binaire (algèbre de Boole)*

- I. codage(s) de l'information
- II. portes logiques et algèbre de Boole
- III. circuits logiques de base (combinatoires)
- IV. circuits logiques à mémoire (séquentiels)

# Première partie

- **Fondements théoriques et technologiques**

*Ordinateur ← Circuits de base très simples dont le comportement fonctionnel est décrit par l'algèbre binaire (algèbre de Boole)*

## I. codage(s) de l'information

- II. portes logiques et algèbre de Boole
- III. circuits logiques de base (combinatoires)
- IV. circuits logiques à mémoire (séquentiels)

# Première partie

## *codage(s) de l'information*

- Utilisation
  - le stockage en mémoire
  - la manipulation des données
  - la communication avec les périphériques
- Architecture de Von Neumann

# Première partie

*codage(s) de l'information*

- Un octet :  $2^8 = 256$  valeurs différentes
- Codage des nombres en binaire non-signé : sur un octet, nombres de 0 à 255

contenu	1	0	1	0	0	0	1	1
Poids	128	64	32	16	8	4	2	1
	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
total = 163	128		32				2	1

contenu	$c_7$	$c_6$	$c_5$	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$
Poids	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
total =	$c_7 \cdot 2^7 + c_6 \cdot 2^6 + c_5 \cdot 2^5 + c_4 \cdot 2^4 + c_3 \cdot 2^3 + c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0$							

# Première partie

*codage(s) de l'information*

- Codage en complément à deux (modulo 256).

Sur un octet, codage de -128 à +127 :

**1010 0011**

**163 - 256 = -93**

- Code ASCII (sur 7 bits)

A chaque octet correspond un caractère.

Exemples :

**0100 0101**

**caractère 'E'**

**0110 0101**

**caractère 'e'**

## Table ASCII

Table ASCII standard (codes de caractères de 0 à 127)

000	(nul)	016	(dle)	032	sp	048	0	064	Ø	080	P	096	'	112	p
001	(soh)	017	(dc1)	033	!	049	1	065	À	081	Q	097	a	113	q
002	(stx)	018	(dc2)	034	"	050	2	066	ß	082	R	098	b	114	r
003	(etx)	019	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	(eot)	020	(dc4)	036	\$	052	4	068	D	084	T	100	d	116	t
005	(eng)	021	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	(ack)	022	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	(bel)	023	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	(bs)	024	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	(vt)	027	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	(np)	028	(fs)	044	,	060	<	076	L	092	\	108	l	124	
013	(cr)	029	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	(so)	030	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	(si)	031	(us)	047	/	063	?	079	O	095	_	111	o	127	

Table ASCII étendue (codes de caractères de 128 à 255)

128	ç	144	É	160	á	176	—	192	+	208	ö	224	ó	240	€SHY;
129	ü	145	æ	161	i	177	—	193	-	209	đ	225	ß	241	±
130	é	146	Œ	162	ô	178	—	194	-	210	Ê	226	Ô	242	—
131	â	147	ô	163	ú	179	—	195	+	211	Ë	227	Ó	243	¾
132	ã	148	ö	164	ñ	180	—	196	-	212	È	228	Ñ	244	‰
133	à	149	ò	165	Ñ	181	À	197	+	213	i	229	Ö	245	§
134	å	150	û	166	*	182	Å	198	ä	214	Í	230	µ	246	÷
135	ç	151	ù	167	°	183	À	199	ë	215	Í	231	þ	247	•
136	ê	152	ÿ	168	¢	184	®	200	+	216	Ï	232	Þ	248	°
137	ë	153	ó	169	®	185	—	201	+	217	+	233	Ú	249	—
138	è	154	Ü	170	¬	186	—	202	-	218	+	234	Û	250	•
139	í	155	ø	171	¤	187	+	203	-	219	—	235	Ù	251	—
140	í	156	£	172	¤	188	+	204	!	220	—	236	ÿ	252	—
141	ì	157	Ø	173	¡	189	¢	205	-	221	—	237	Ý	253	—
142	ää	158	×	174	«	190	¥	206	+	222	í	238	—	254	—
143	ää	159	f	175	»	191	+	207	¤	223	—	239	’	255	—

# Première partie

## • **Fondements théoriques et technologiques**

*Ordinateur ← Circuits de base très simples dont le comportement fonctionnel est décrit par l'algèbre binaire (algèbre de Boole)*

I. codage(s) de l'information

**II. portes logiques et algèbre de Boole**

III. circuits logiques de base (combinatoires)

IV. circuits logiques à mémoire (séquentiels)

# Première partie

## *Portes logiques et algèbre de Boole*

- Circuits logiques élaborés à partir de composants électroniques primaires : *les transistors*
- Une porte logique = assemblage de transistors
- Certains principes de base de l'algèbre de Boole permettent l'analyse des circuits logiques
- Sujets abordés :

→les portes logiques

→l'algèbre de Boole (un raccourci)

→la réalisation des fonctions booléennes

→la recherche de circuits équivalents

# Première partie

## *Portes logiques et algèbre de Boole*

- **Sujets abordés :**

→les portes logiques

→l'algèbre de Boole (un raccourci)

→la réalisation des fonctions booléennes

→la recherche de circuits équivalents

# Première partie

## *Portes logiques et algèbre de Boole*

- Sujets abordés :

→ **les portes logiques**

→ l'algèbre de Boole (un raccourci)

→ la réalisation des fonctions booléennes

→ la recherche de circuits équivalents

# Première partie

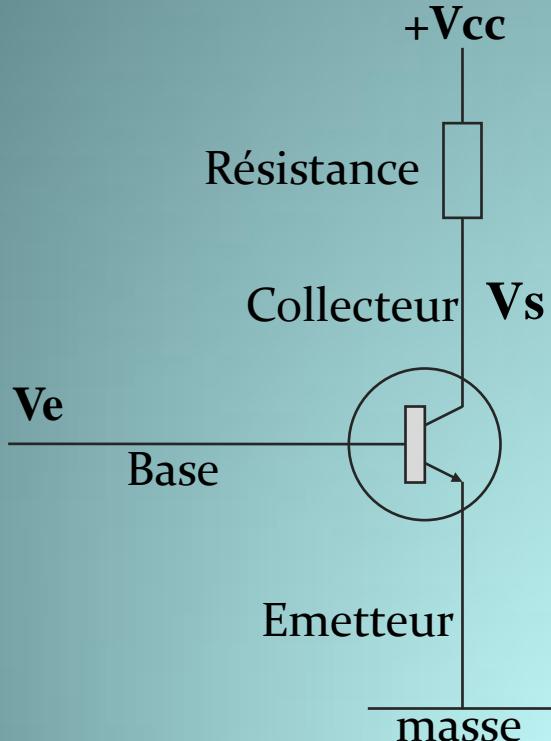
*Portes logiques et algèbre de Boole  
les portes logiques*

- le fondement matériel des ordinateurs actuels ← diverses fonctions logiques
  - ↑
  - Circuit logique → Porte logique
- Circuit logique à base de transistors → un comportement binaire
  - deux états logiques*
  - \*un signal électrique compris entre zéro et un volt (**état binaire ZERO**)
  - \*un signal compris entre deux et cinq volts au plus (**état binaire UN**)
- Un *transistor* est utilisé comme *interrupteur électronique* très rapide

# Première partie

## Portes logiques et algèbre de Boole

### les portes logiques



trois connexions externes :

- ❖ le collecteur
- ❖ la base
- ❖ l'émetteur

#### • **Principe de fonctionnement :**

- \* la tension d'entrée  $V_e \leq$  une valeur critique, le transistor se  bloque  → un interrupteur ouvert (grande résistance). La tension de sortie  $V_s$  (Collecteur) est au niveau  haut  ( $\approx V_{cc}$  , Voltage Collector Current)
- \* la tension  $V_e >$  la valeur critique , le transistor → un conducteur (résistance quasi nulle) forçant la tension de sortie  $V_s$  au  niveau bas  : une valeur voisine de celle de la masse.

⇒ **le signal  $V_s$  est au niveau haut lorsque  $V_e$  est au niveau bas et réciproquement → circuit inverseur logique ( porte NON ).**

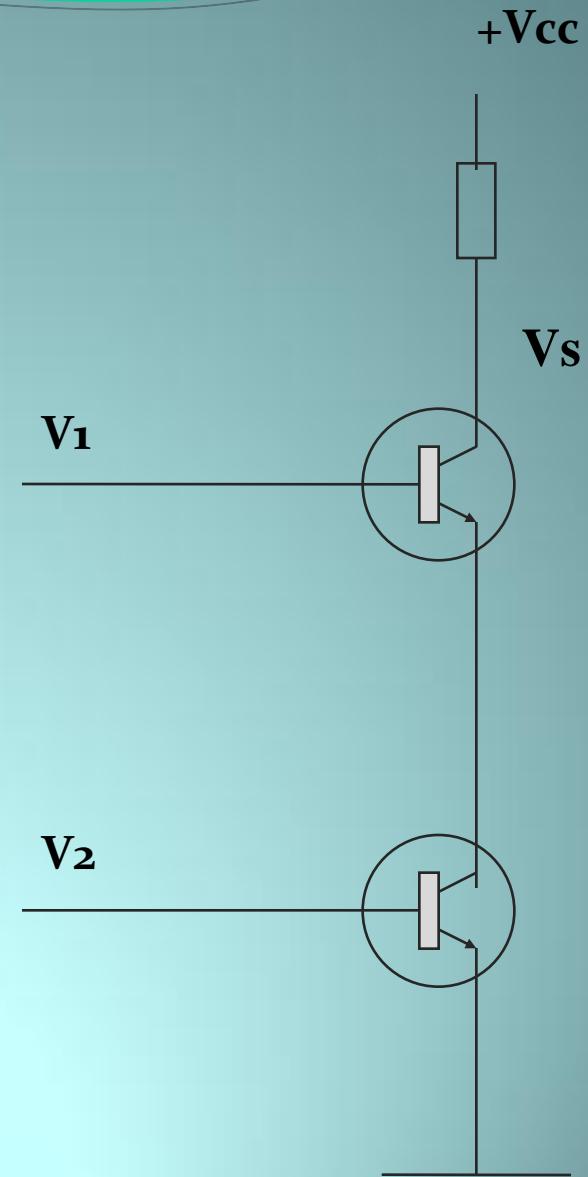
- *Le temps de basculement nécessaire au passage d'un état à l'autre est de quelques nanosecondes ( $1\text{ns} = 10^{-9}\text{ s}$ )*

# Première partie

*Portes logiques et algèbre de Boole  
les portes logiques*

- **Exercice 1**

Associons deux transistors *en série* :



# Première partie

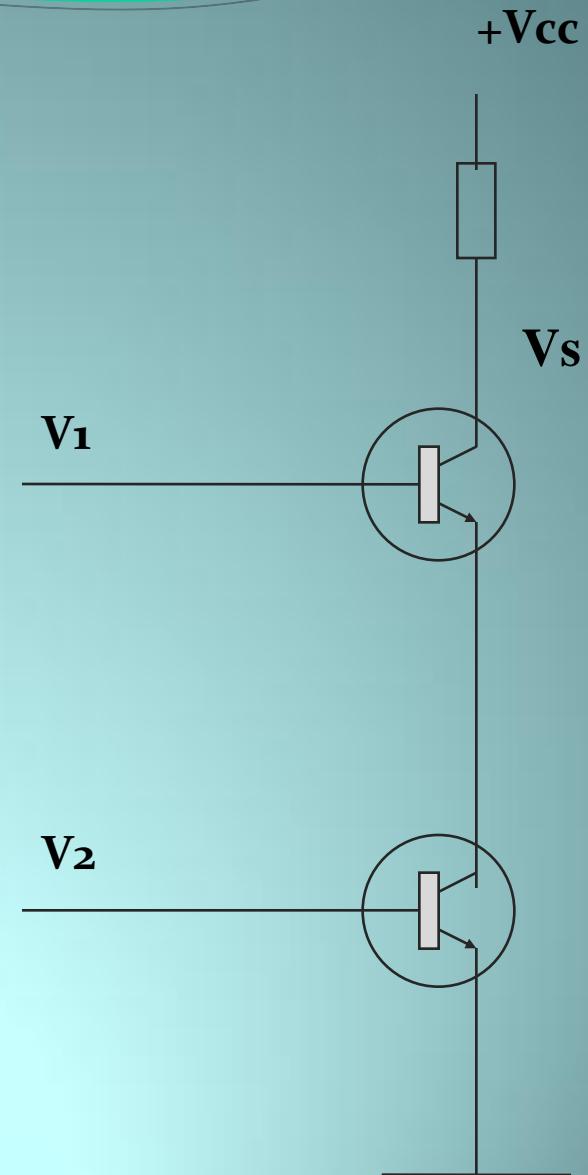
## *Portes logiques et algèbre de Boole les portes logiques*

- **Exercice 1**

Associons deux transistors *en série* :

- Si les tensions d'entrée  $V_1$  et  $V_2$  sont au niveau haut, les transistors sont conducteurs et  $V_s$  est forcé au niveau bas.
- Si l'une seulement des deux entrées est au niveau bas ( $V_1$  ou  $V_2$ ) le transistor correspondant n'est plus conducteur et la sortie  $V_s$  reste au niveau haut.

⇒ Ainsi  $V_s$  est au niveau bas(0) ssi  $V_1$  et  $V_2$  sont au niveau haut(1) (porte NAND).

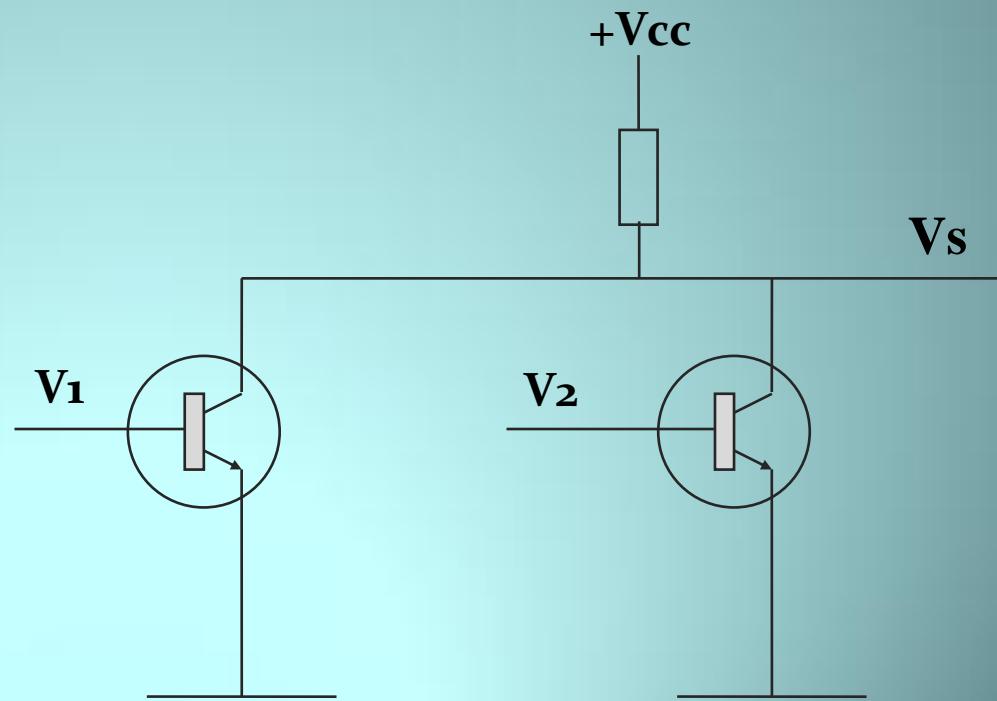


# Première partie

*Portes logiques et algèbre de Boole  
les portes logiques*

- **Exercice 2**

Associons deux transistors *en parallèle* :



# Première partie

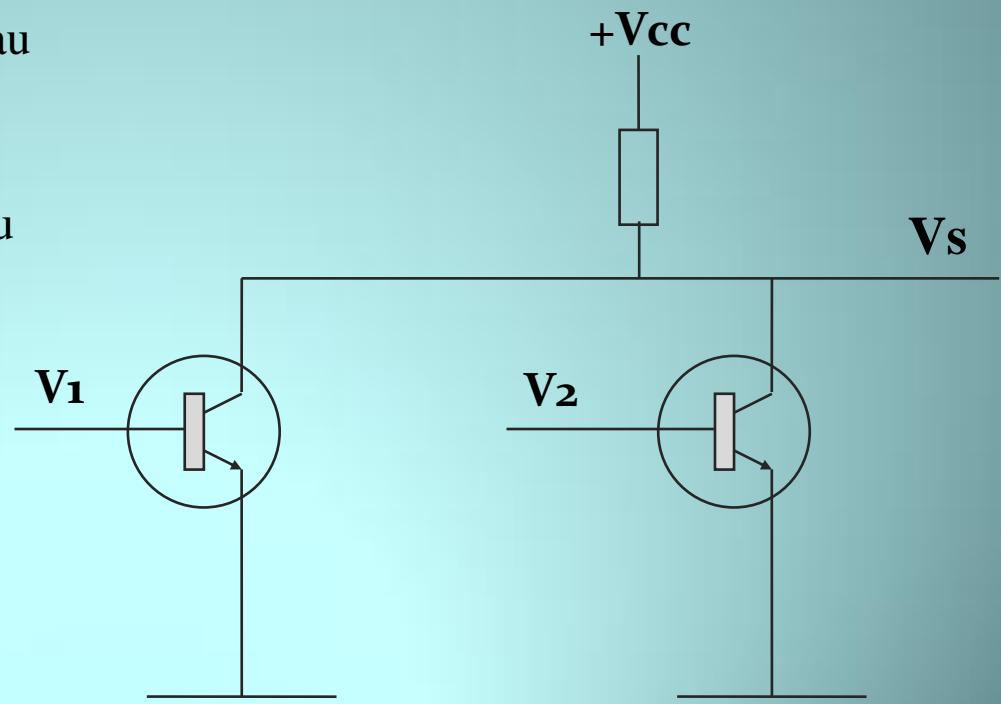
## *Portes logiques et algèbre de Boole* les portes logiques

- **Exercice 2**

Associons deux transistors *en parallèle* :

➤ Si l'une des entrées  $V_1$  ou  $V_2$  est au niveau haut, le transistor correspondant est conducteur et force la sortie  $V_s$  au niveau bas.

⇒ La sortie  $V_s$  est au niveau haut si et seulement si les deux entrées sont au niveau bas (NOR).



# Première partie

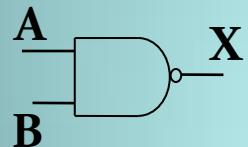
*Portes logiques et algèbre de Boole*  
les portes logiques

**NON (NOT)**  
1 transistor



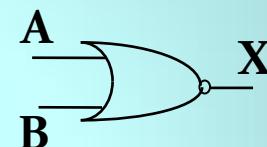
A	X
0	1
1	0

**NON ET  
(NAND)**  
2 transistors



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

**NON OU  
(NOR)**  
2 transistors

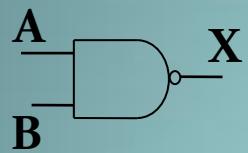


A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

# Première partie

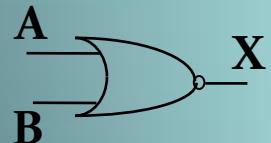
*Portes logiques et algèbre de Boole*  
les portes logiques

**NON ET  
(NAND)**  
2 transistors

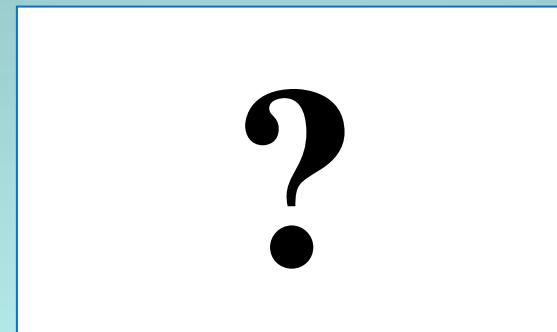


A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

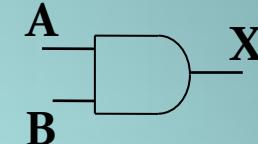
**NON OU  
(NOR)**  
2 transistors



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

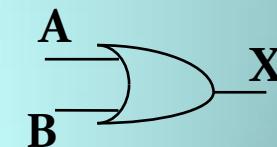


**ET  
(AND)**



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

**OU  
(OR)**

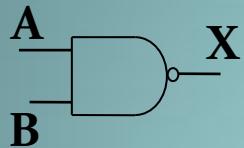


A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

# Première partie

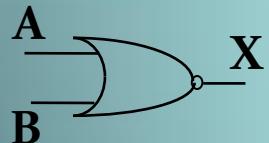
## Portes logiques et algèbre de Boole les portes logiques

NON ET  
(NAND)  
2 transistors



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

NON OU  
(NOR)  
2 transistors



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

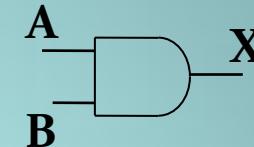
→ Pour les portes ET, OU, les sorties des circuits NON-ET et NON-OU sont connectées à un inverseur.

On note

$$\begin{aligned} X &= \text{NON-NON-ET}(A, B) \\ &= \text{ET}(A, B) \end{aligned}$$

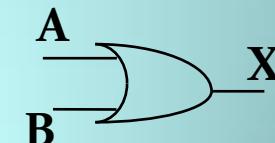
$$\begin{aligned} X &= \text{NON-NON-OU}(A, B) \\ &= \text{OU}(A, B) \end{aligned}$$

ET  
(AND)  
NON NON ET  
3 transistors



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

OU  
(OR)  
NON NON OU  
3 transistors



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

# Première partie

*Portes logiques et algèbre de Boole  
les portes logiques*

## REMARQUE :

deux familles technologiques dominent le domaine des *circuits logiques* :

- la *technologie bipolaire* (basée sur le transistor à jonction) :

.TTL (Tr. Tr. Logic)	rapides & chers
.ECL (Emitter Coupled Logic)	

- la *technologie unipolaire* MOS (Metaloxyd Semiconductor, basée sur le transistor à effet de champ) :

PMOS, NMOS, CMOS	moins rapides que les circuits bipolaires
HMOS et XMOS	aussi rapides et ... moins chers

# Première partie

## *Portes logiques et algèbre de Boole*

- **Sujets abordés :**

→les portes logiques

→l'**algèbre de Boole (un raccourci)**

→la réalisation des fonctions booléennes

→la recherche de circuits équivalents

# Première partie

*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

- **Boole: 1815-1864, mathématicien anglais**
- **Partie de ses travaux relative aux fonctions et variables binaires (variables booléennes)**
- **Une fonction booléenne à n variables fournit un résultat qui dépend uniquement de la valeur binaire de ses variables**
- **EXAMPLE:** une fonction  $f$  d'une variable binaire  $A$  peut être définie en précisant que:

$$f(A)=1 \text{ si } A=0$$

$$f(A)=0 \text{ si } A=1$$

de quelle fonction s'agit-il ?

# Première partie

## *Portes logiques et algèbre de Boole* *l'algèbre de Boole*

➤ Une fonction booléenne à n variables présente  $2^n$  états possibles et on peut décrire complètement cette fonction à l'aide d'une table de  $2^n$  lignes appelée **table de vérité (TV)** de la fonction.

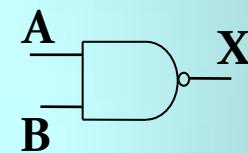
➤ **EXEMPLE 1 :** les tables de vérité des fonctions ou portes de base dans lesquelles chaque ligne de la table de vérité (TV) indique la valeur prise par cette fonction pour une configuration binaire des n variables (ici, n=1 ou 2)

NON (NOT)  
1 transistor



A	X
0	1
1	0

NON ET  
(NAND)  
2 transistors



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

# Première partie

*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

➤ **Le nombre N des fonctions booléennes à n variables est fini**

➤ **Exemple : pour n=1 , 4 fonctions ; n=2, 16 fonctions , etc... )**



A	X	X	X	X
0	0	0	1	1
1	0	1	0	1

# Première partie

*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

➤ **Exercice** : décrire les fonctions disponibles pour la porte logique de 2 entrées

➤ Nombre de fonction ?

➤ Les représenter dans un tableau



# Première partie

*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

A	B	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0	0	0	1	0	0	0	1	0	0	1	0	1	1	0	1	1	1
0	1	0	0	1	0	0	1	1	0	0	1	0	1	1	0	1	1
1	0	0	0	0	1	0	0	1	1	1	0	0	1	1	1	0	1
1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1

n entrées → nombre des combinaisons  $a=2^n$ , nombre de fonctions possible  $N = 2^a$

$$N = 2^a \text{ avec } a = 2^n$$

# Première partie

*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

**EXAMPLE 2 :** soit la table de vérité  
d'une fonction de trois variables  
réalisant la *fonction majoritaire*

$$M = f(A, B, C)$$

M vaut zéro si la majorité des variables  
d'entrée vaut zéro et un sinon

A	B	C	M
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

# Première partie

## *Portes logiques et algèbre de Boole* *l'algèbre de Boole*

**EXAMPLE 2 :** soit la table de vérité d'une fonction de trois variables réalisant la *fonction majoritaire*

$$M = f(A, B, C)$$

M vaut zéro si la majorité des variables d'entrée vaut zéro et un sinon

A	B	C	M	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A} \cdot B \cdot C$
1	0	0	0	
1	0	1	1	$A \cdot \bar{B} \cdot C$
1	1	0	1	$A \cdot B \cdot \bar{C}$
1	1	1	1	$A \cdot B \cdot C$

### Expression logique

→ utile pour la réalisation des fonctions à l'aide de portes logiques

→ établie à partir :

\*des variables d'entrée qui fournissent à la fonction M *un résultat égal à 1*

\*des conventions suivantes:

X indique valeur 0 (complément de 1)

X indique valeur 1 pour X

. ou rien pour le ET (multiplication)

+ pour le OU (addition)

**M vaut 1 (vrai)ssi :**

**A=0 ET B=1 ET C=1 OU ...**

$$M = \bar{A}BC + A\bar{B}C + ABC + A\bar{B}\bar{C}$$

# Première partie

*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

**EXERCICE :** réaliser le schéma logique représentant la fonction majoritaire M, à l'aide des portes de base ET, OU et NON.

# Première partie

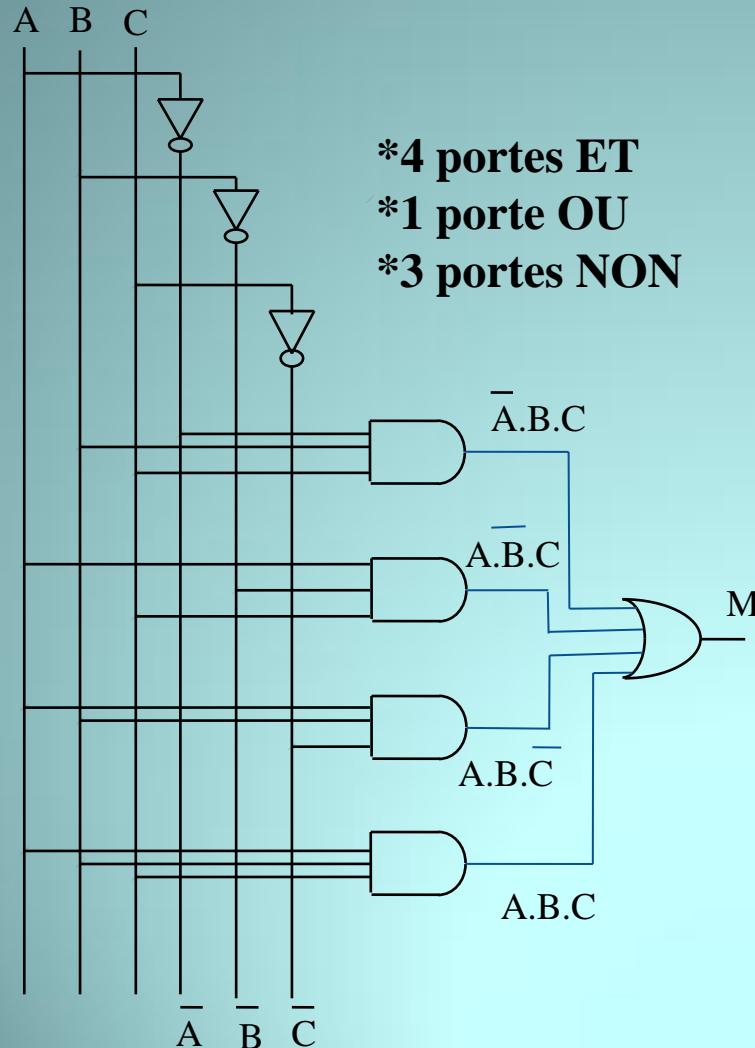
*Portes logiques et algèbre de Boole*  
*l'algèbre de Boole*

## Étapes de réalisation

1. écrire l'équation de la fonction à partir de la TV
2. réaliser l'inversion de toutes les variables d'entrée pour disposer de leur complément
3. prévoir une porte ET pour chaque terme égal à 1 dans la colonne valeur
4. établir le câblage des portes ET avec les entrées appropriées
5. réunir l'ensemble des sorties des portes ET vers une porte OU dont la sortie est le résultat de la fonction

# Première partie

## *Portes logiques et algèbre de Boole* *l'algèbre de Boole*



### Étapes de réalisation

1. écrire l'équation de la fonction à partir de la TV
2. réaliser l'inversion de toutes les variables d'entrée pour disposer de leur complément
3. prévoir une porte ET pour chaque terme égal à 1 dans la colonne valeur
4. établir le câblage des portes ET avec les entrées appropriées
5. réunir l'ensemble des sorties des portes ET vers une porte OU dont la sortie est le résultat de la fonction

# Première partie

## *Portes logiques et algèbre de Boole*

- **Sujets abordés :**

→les portes logiques

→l'algèbre de Boole (un raccourci)

→**la réalisation des fonctions  
booléennes**

→la recherche de circuits équivalents

# Première partie

*Portes logiques et algèbre de Boole*

*la réalisation des fonctions booléennes*

- ❖ **il n'est pas toujours possible de disposer de portes ET et OU ayant autant d'entrées que nécessaire (restrictions en pratique)**
- ❖ **il est souvent impératif de réaliser la fonction avec un seul type de porte (NON-ET ou NON-OU le plus souvent )**

**Les portes NON-ET (NAND) ou NON-OU (NOR) sont dites complètes car on peut réaliser avec seulement l'une ou l'autre n'importe quelle fonction booléenne → ces portes constituent les éléments de base des circuits intégrés d'aujourd'hui.**

# Première partie

*Portes logiques et algèbre de Boole*

*la réalisation des fonctions booléennes*

**EXERCICE 1** : réaliser une porte OU à quatre entrées

**EXERCICE 2** : réaliser des portes NON, ET et OU d'abord à partir de portes NON-ET (NAND) puis à partir de portes NON-OU (NOR) ; le choix effectué sera justifié ultérieurement à l'aide de la loi de De Morgan.

# Première partie

*Portes logiques et algèbre de Boole*

*la réalisation des fonctions booléennes*

**EXERCICE 1** : réaliser une porte OU à quatre entrées à l'aide de trois portes OU à deux entrées chacune

*indication : on remarquera que*

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

**EXERCICE 2** : réaliser des portes NON, ET et OU d'abord à partir de portes NON-ET (NAND) puis à partir de portes NON-OU (NOR) ; le choix effectué sera justifié ultérieurement à l'aide de la loi de De Morgan.

# Première partie

*Portes logiques et algèbre de Boole*

*la réalisation des fonctions booléennes*

De Morgan

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

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



A	B	$\overline{A}$	$\overline{B}$	$A+B$	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

# Première partie

*Portes logiques et algèbre de Boole*

la réalisation des fonctions booléennes

De Morgan

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

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



A	B	$\overline{A}$	$\overline{B}$	$A+B$	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

NON-ET (NAND (A, B) =  $\overline{A \cdot B}$ )

$$\overline{A} = \overline{A} \cdot \overline{A}$$

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

$$A \cdot B = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A} \cdot \overline{B} \cdot \overline{A} \cdot \overline{B}}$$

NON-OU (NOR (A, B) =  $\overline{A+B}$ )

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

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

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

# Première partie

*Portes logiques et algèbre de Boole*

*la réalisation des fonctions booléennes*

De Morgan

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

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



A	B	$\overline{A}$	$\overline{B}$	$A+B$	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

NON-ET (NAND (A, B) =  $\overline{A \cdot B}$ )

$$\overline{A} = ?$$

$$A+B = ?$$

$$A \cdot B = ?$$

NON-OU (NOR (A, B) =  $\overline{A+B}$ )

$$\overline{A} = ?$$

$$A \cdot B = ?$$

$$A+B = ?$$

# Première partie

## *Portes logiques et algèbre de Boole*

- **Sujets abordés :**

→les portes logiques

→l'algèbre de Boole (un raccourci)

→la réalisation des fonctions booléennes

→**la recherche de circuits équivalents**

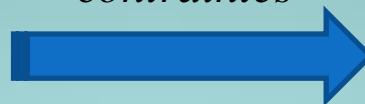
# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

- \*le coût en nombre de boîtiers,
- \*la surface d'implantation sur la plaque de circuit imprimé,
- \*la consommation électrique,
- \*etc...

*contraintes*



**réduire le nombre de portes nécessaires à la réalisation des systèmes logiques**



*recherche*

**un équivalent qui réalise la même fonction dans les mêmes conditions.**

*À l'aide des lois et théorèmes de l'algèbre de Boole*

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

## EXEMPLE:

considérons le circuit réalisant la fonction **AB+AC**, constitué de trois portes ( 2 ET et 1 OU ). La table de vérité ci-contre montre que  $AB+AC = A(B+C)$

A	B	C	AB	AC	AB+AC	B+C	A(B+C)
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

## EXEMPLE:

considérons le circuit réalisant la fonction  $AB+AC$ , constitué de trois portes ( 2 ET et 1 OU ). La table de vérité ci-contre montre que  $AB+AC = A(B+C)$

→ Deux fonctions sont équivalentes.  
⇒ la seconde est "le meilleur" qui nécessite seulement deux portes ( 1 ET et 1 OU )

A	B	C	AB	AC	AB+AC	B+C	A(B+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

## PRINCIPALES LOIS DE L'ALGEBRE DE BOOLE

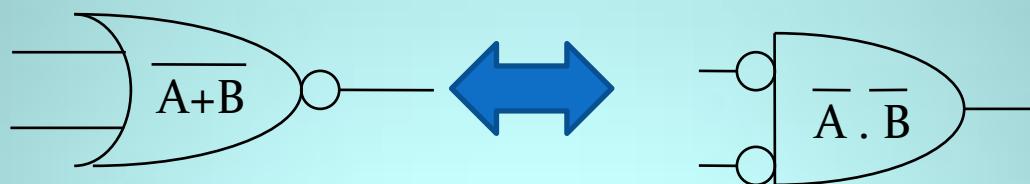
(permettant de réduire les fonctions sous forme optimale)

nom de la loi	forme ET	forme OU
loi d'identité	$1A = A$	$0+A = A$
loi de nullité	$0A = 0$	$1+A = 1$
loi d'idempotence	$AA = A$	$A+A = A$
loi d'inversion	$A\bar{A} = 0$	$A+\bar{A} = 1$
loi commutative	$AB = BA$	$A+B = B+A$
loi associative	$(AB)C = A(BC)$	$(AB)+C = A+(B+C)$
loi distributive	$A+BC = (A+B)(A+C)$	$A(B+C) = AB+AC$
loi d'absorption	$A(A+B) = A$	$A+AB = A$
loi de DE MORGAN	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A+B} = \overline{A} \overline{B}$

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents



**Représentations symboliques différentes**

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

Une fonction booléenne bien utile : **OU – exclusif**

Cette fonction vaut 1 si et seulement si ses entrées sont différentes  $X = \overline{A}B + A\overline{B}$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



# Première partie

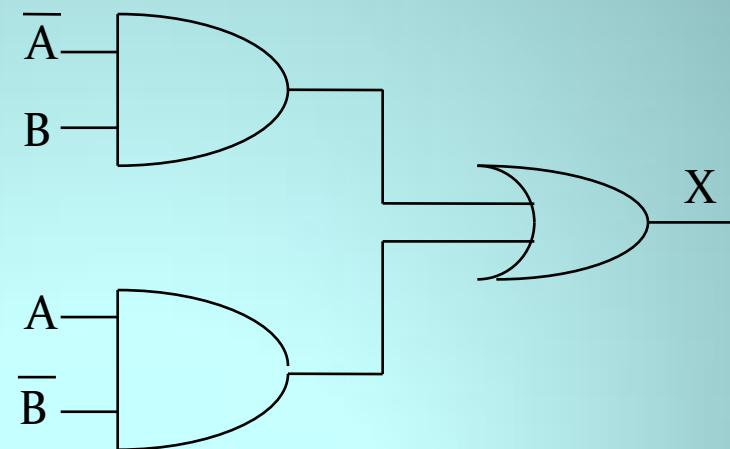
*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

**Une fonction booléenne bien utile : OU – exclusif**

Cette fonction vaut 1 ssi ses entrées sont différentes  $X = \overline{A}B + A\overline{B}$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

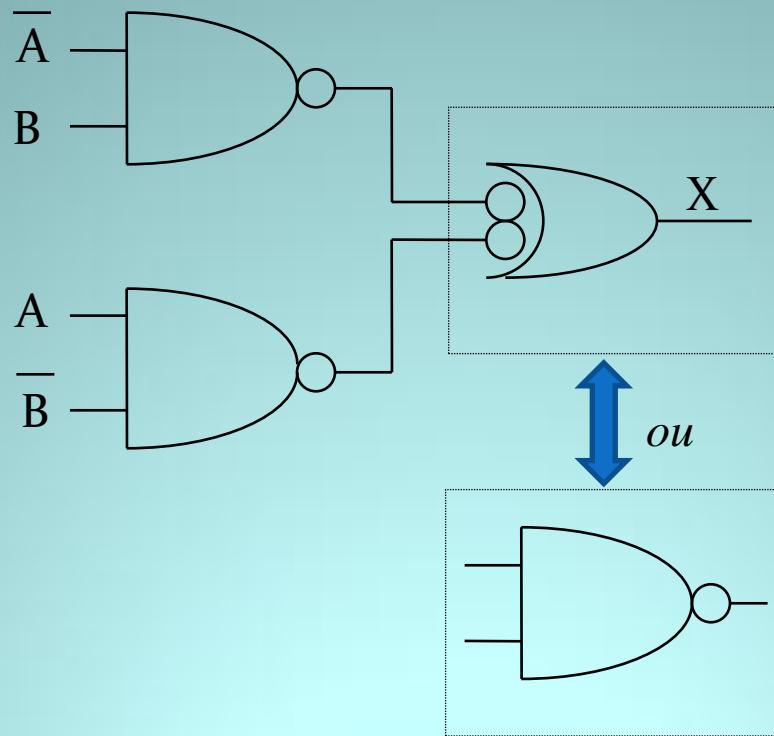
**EXERCICE:**

**Réaliser un circuit OU-exclusif à l'aide de portes NON-ET (NAND) uniquement**

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents



Le circuit comporte :

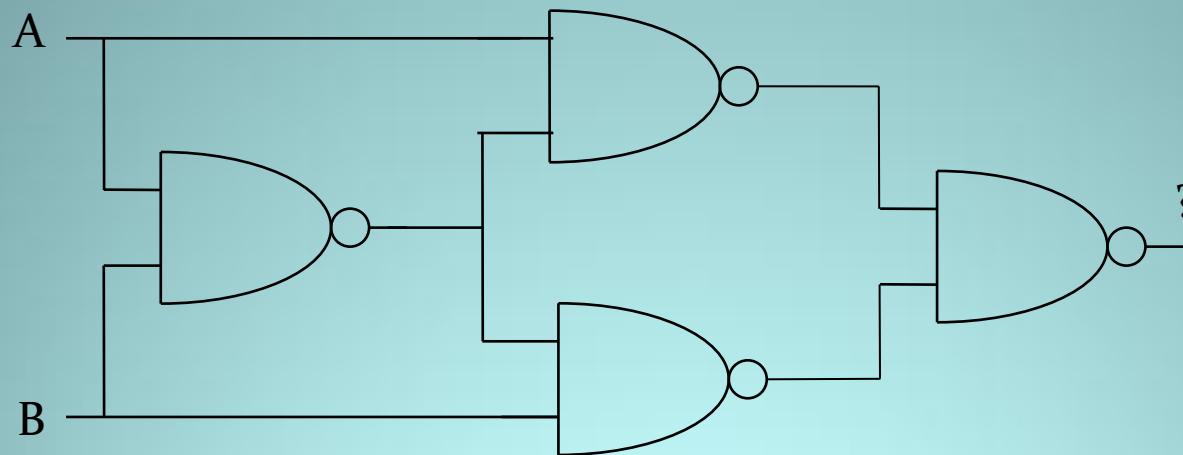
- \*2 inverseurs
- \*3 portes NAND
- 8 transistors

Première solution

# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

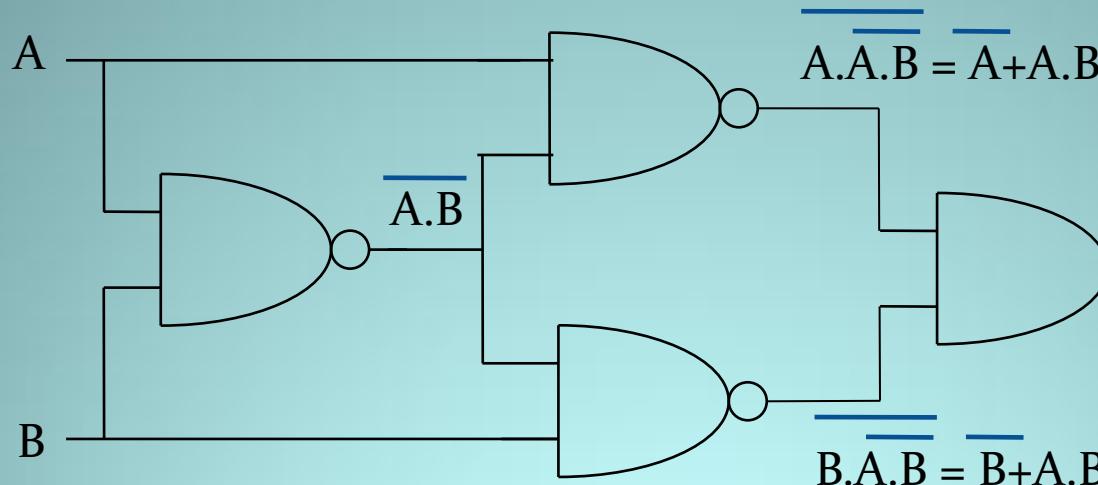


*Deuxième solution*

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents



$$(\overline{\overline{A}} + A \cdot B) (\overline{\overline{B}} + A \cdot B) = (\overline{\overline{A}} + A \cdot B) + (\overline{\overline{B}} + A \cdot B) = A (\overline{\overline{A}} + \overline{\overline{B}}) + B (\overline{\overline{A}} + \overline{\overline{B}}) = \overline{A} \cdot B + A \cdot \overline{B}$$

Deuxième solution

# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

## Simplification de fonctions booléennes par les tableaux de KARNAUGH

- ❖ Méthode permettant de former des « sommes de produits »
- ❖ On écrit la table de vérité comme un tableau à deux entrées en gardant adjacentes les entrées qui ne diffèrent que d'une seule valeur binaire
- ❖ On peut ainsi "voir" les produits de littéraux utiles prendre la forme de rectangles dont les points ont la valeur 1
- ❖ Utilisable pour les fonctions booléennes jusqu'à 4 variables

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

Exemple : Soit une fonction  $f$  de deux variables  $a$  et  $b$  donnée par sa table de vérité

a	b	f	
0	0	1	$\overline{ab}$
0	1	1	$\overline{a}b$
1	0	0	
1	1	1	$ab$

f	b		
	0	1	
a	0	?	?
	1	?	?

$$f(a,b) = \overline{\overline{a}} + \overline{a}b + ab \rightarrow \text{simplifier ?}$$

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

Exemple : Soit une fonction  $f$  de deux variables  $a$  et  $b$  donnée par sa table de vérité

a	b	f	
0	0	1	$\overline{ab}$
0	1	1	$\overline{a}b$
1	0	0	
1	1	1	$ab$

f	b		
	0	1	
a	0	1	1
	1	0	1

$$f(a,b) = \overline{\overline{a}} + \overline{a}b + ab \rightarrow \text{simplifier ?}$$


$$f(a,b) = \overline{a}(\overline{b} + b) + ab = \overline{a} + ab$$

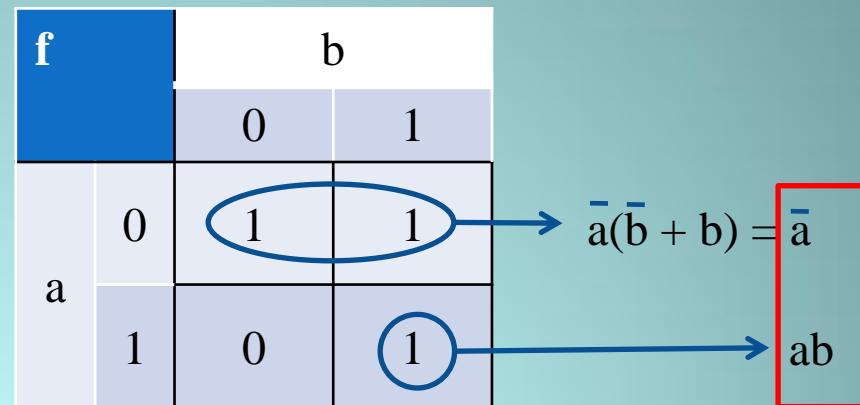
# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

Exemple : Soit une fonction  $f$  de deux variables  $a$  et  $b$  donnée par sa table de vérité

a	b	f	
0	0	1	$\bar{a}\bar{b}$
0	1	1	$\bar{a}b$
1	0	0	
1	1	1	$ab$



$$f(a,b) = \bar{a}\bar{b} + \bar{a}b + ab \rightarrow \text{simplifier ?}$$

→  $f(a,b) = \bar{a}(\bar{b} + b) + ab = \bar{a} + ab$

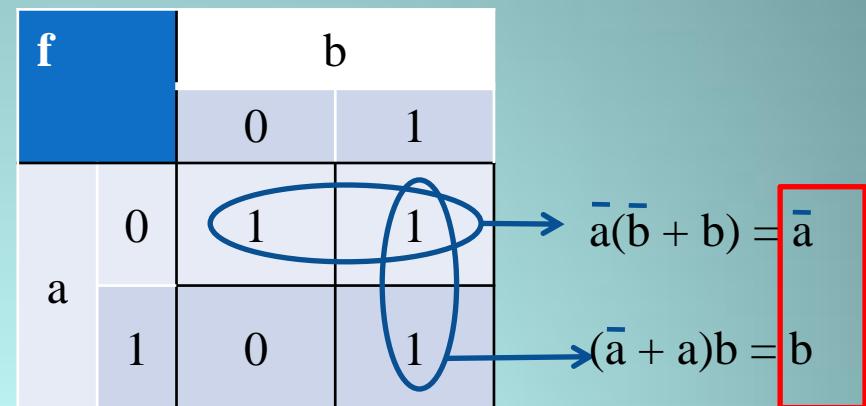
# Première partie

Portes logiques et algèbre de Boole

la recherche de circuits équivalents

a	b	f	
0	0	1	$\overline{ab}$
0	1	1	$\overline{a}b$
1	0	0	
1	1	1	ab

$$f(a,b) = \overline{ab} + \overline{a}b + ab$$



$$f(a,b) = \overline{a}\overline{b} + (\overline{a} + a)b = \overline{a} + b$$



# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

**EXERCICE 1:** retrouver la loi d'absorption

$$[a + ab = a ; a(a+b)=a]$$

**EXERCICE 2:** simplifier  $\overline{pq}+pq$  à l'aide de tableau de Karnaugh

# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

**Liste des rectangles possibles :**

- ❖ **un point quelconque**
- ❖  **$2^i$  cases adjacentes verticalement/horizontalement (y compris extrémités)**
- ❖ **un carré  $2^i \times 2^i$  (y compris extrémités)**
- ❖ **la table entière**

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

**EXEMPLE 1 :** soit une fonction booléenne à trois variables  $f$  donnée par sa table de vérité

f		yz			
		00	01	11	10
x	0	0	0	1	0
	1	0	1	1	1

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

f		yz			
		00	01	11	10
x	0	0	0	1	0
	1	0	1	1	1

$$f(x, y, z) = \textcolor{red}{xz} + \textcolor{red}{yz} + xy$$

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

**EXEMPLE 2 :** soit une fonction booléenne à trois variables  $f$  donnée par sa table de vérité

f		yz			
		00	01	11	10
x	0	1	1	1	1
	1	1	1	0	1

# Première partie

*Portes logiques et algèbre de Boole*

la recherche de circuits équivalents

**EXEMPLE 2 :** soit une fonction booléenne à trois variables  $f$  donnée par sa table de vérité

f		yz			
		00	01	11	10
x	0	1	1	1	1
	1	1	1	0	1

```
graph LR; A(( )) --> B(( )); C(( )) --> D(( )); E(( )) --> F(( )); G(( )) --> H(( )); I(( )) --> J(( )); K(( )) --> L(( )); M(( )) --> N(( )); O(( )) --> P(( )); Q(( )) --> R(( )); S(( )) --> T(( )); U(( )) --> V(( )); W(( )) --> X(( )); Y(( )) --> Z(( ));
```

$$f(x, y, z) = \bar{x} + \bar{y} + \bar{z} = xyz = \text{NAND}(x, y, z)$$

# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

**EXERCICE 1 : simplifier l'expression de la fonction majoritaire**

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

**en utilisant un tableau de Karnaugh puis (autre méthode) les lois de l'algèbre de BOOLE.**

**EXERCICE 2 : simplifier  $\overline{A + \overline{B}C + \overline{A}B}$  de deux manières différentes.**

# Première partie

*Portes logiques et algèbre de Boole*

*la recherche de circuits équivalents*

**EXERCICE 1 : simplifier l'expression de la fonction majoritaire**

$$M = \bar{A}BC + A\bar{B}C + ABC\bar{C} + ABC$$

en utilisant un tableau de Karnaugh puis (autre méthode) les lois de l'algèbre de BOOLE. *Karnaugh comme exemple 1. Boole recopier 2 fois de ABC*

**EXERCICE 2 : simplifier  $\overline{A + \bar{B} C} + \bar{A}B = \bar{A}B + \bar{A}\bar{C}$  en développant le 1<sup>er</sup> élément**

$$\overline{A + \bar{B} C} = 1 \text{ si } A=0 \text{ et } \bar{B} C = 0 \quad [(\text{Si } \bar{B}=0 \text{ ou } C=0) \rightarrow (B=1 \text{ ou } C=0)]$$

# Première partie

## • **Fondements théoriques et technologiques**

*Ordinateur ← Circuits de base très simples dont le comportement fonctionnel est décrit par l'algèbre binaire (algèbre de Boole)*

- I. codage(s) de l'information
- II. portes logiques et algèbre de Boole

## **III. circuits logiques de base (combinatoires)**

- IV. circuits logiques à mémoire (séquentiels)

# Première partie

## *Circuits logiques de base (combinatoires)*

➤ Il est possible de réaliser des fonctions logiques en partant de leur T.V. et en les construisant à partir de portes individuelles

➤ En pratique

Les concepteurs de systèmes logiques disposent de circuits plus ou moins complexes dont les plus courants sont :

\*les circuits intégrés logiques (CIL)

\*les circuits logiques combinatoires (multiplexeurs, décodeurs,

\*comparateurs, réseau logique programmable)

\*les circuits de traitement ou de calcul (décaleur, additionneur, UAL)

\*les horloges

# Première partie

## *Circuits logiques de base (combinatoires)*

➤ Il est possible de réaliser des fonctions logiques en partant de leur T.V. et en les construisant à partir de portes individuelles

➤ En pratique

Les concepteurs de systèmes logiques disposent de circuits plus ou moins complexes dont les plus courants sont :

### \*les circuits intégrés logiques (CIL)

\*les circuits logiques combinatoires (multiplexeurs, décodeurs, comparateurs, réseau logique programmable)

\*les circuits de traitement ou de calcul (décaleur, additionneur, UAL)

\*les horloges

# Première partie

*Circuits logiques de base (combinatoires)*

*les circuits intégrés logiques (CIL)*

- Appelés CIL, puce ou "chip"
- Ce sont les composants que l'on trouve dans les :
  - systèmes logiques,
  - systèmes numériques,
  - Ordinateurs

Exemple :

le schéma d'un CIL / SSI (small scale integration) :

circuit CMOS 4011

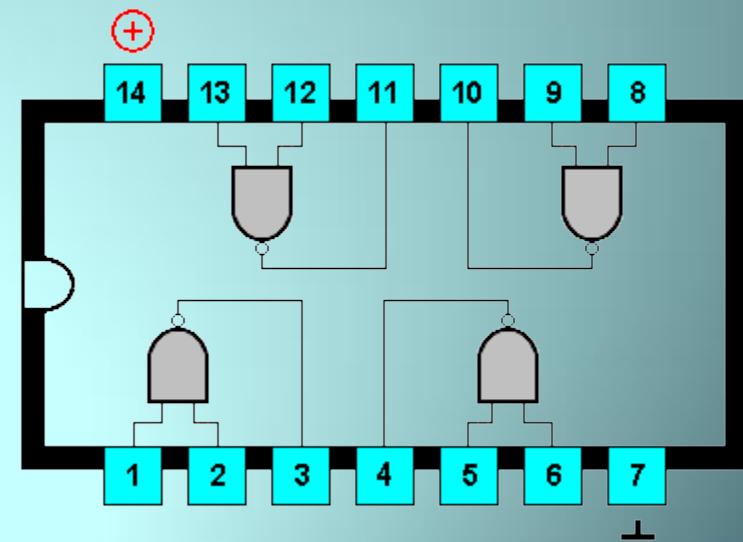
→ Surface de la puce 5x5 mm<sup>2</sup>

→ un boîtier

\*rectangulaire en plastique (l=5 à 20 mm, L=20 à 50 mm)

\*dispose de deux rangées // de connexions et porte le nom de : DIL ou DIP (dual in line package) à 14 broches

→ contient 4 portes NAND



# Première partie

*Circuits logiques de base (combinatoires)*

*les circuits intégrés logiques (CIL)*

❖ On distingue 4 familles de CIL selon la densité d'intégration (nombre de portes ou de transistors internes par circuit ou par mm<sup>2</sup>) :

Type circuit	Nature	nb portes/circuit	nb portes/nb broches
SSI	Small scale Integration	1 à 10	<1
MSI	Medium scale integration	10 à 100	5 à quelques dizaines
LSI	Large scale integration	100 à 100 000	25 à 1000
VLSI	Very large scale integration	> 100 000	Plusieurs milliers

❖ Dans les circuits VLSI actuels, on envisage un million de portes par circuit et encore plus dans les ULSI

*Intérêt essentiel* : permettre la construction de circuits logiques très complexes, peu encombrants, peu coûteux ...

# Première partie

*Circuits logiques de base (combinatoires)*

## Sujets abordés :

- \*les circuits intégrés logiques (CIL)
- \*les circuits logiques combinatoires (multiplexeurs, décodeurs, comparateurs, réseau logique programmable)
- \*les circuits de traitement ou de calcul (décaleur, additionneur, UAL)
- \*les horloges

# Première partie

*Circuits logiques de base*

*les circuits logiques combinatoires*

**Deux catégories de circuits dans les systèmes logiques :**

\**Circuits combinatoires* :

Sortie[t] = fonction ( Entrées[t] )

Exemple : circuit de la fonction majoritaire

\**Circuits séquentiels ou circuits logiques à mémoire* :

Sortie [instant t] = fonction (Entrées[t], Entrées[t-1]],...,Sortie [t-1], Sortie [t-2],... )

Sujets abordés :

⇒les caractéristiques des principaux circuits MSI combinatoires (multiplexeur, décodeur, comparateur et les réseaux logiques programmables)

# Première partie

*Circuits logiques de base*

*les circuits logiques combinatoires*

## Multiplexeur

Ce circuit dispose de  $n$  entrées ( $E_0, \dots, E_{n-1}$ ),  
d'une sortie  $S$  et de  $m$  lignes de sélection  
( $C_0, \dots, C_{m-1}$ )

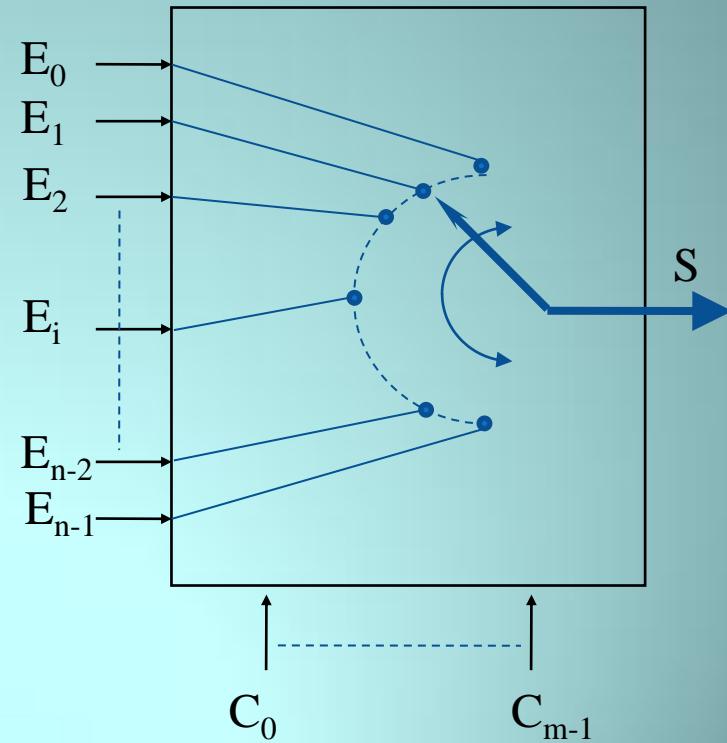
$$n = 2^m$$

On dit qu'il est du type 1 parmi  $2^n$

Son rôle :

une liaison entre sa sortie et une entrée  
parmi les  $n$  selon la configuration binaire  
présente sur les  $m$  lignes de sélection.

$\Rightarrow$ fonction d'aiguillage



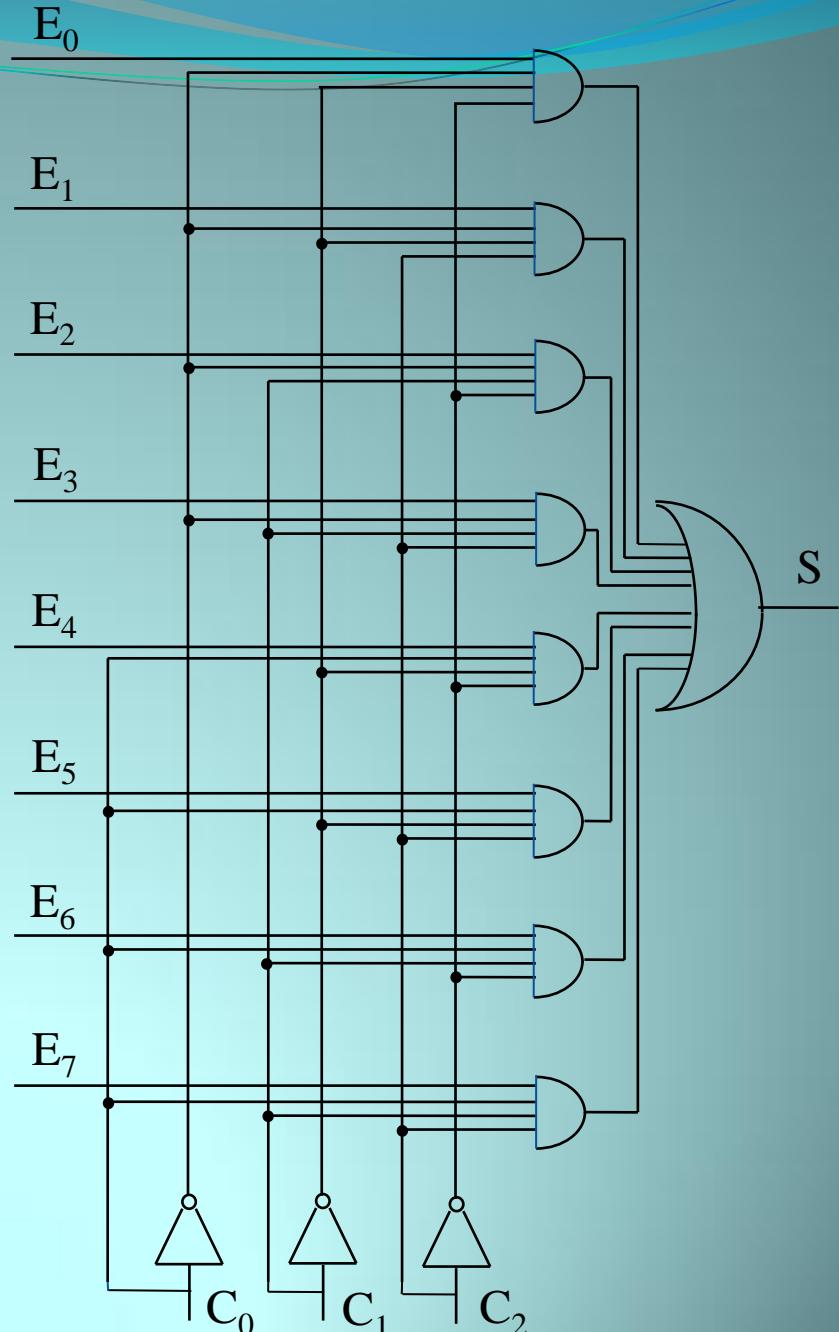
# Première partie

## Circuits logiques de base

### les circuits logiques combinatoires

#### EXEMPLE : le multiplexeur à 8 entrées

- ✓ Les 3 lignes de sélection  $C_0$ ,  $C_1$  et  $C_2$  différencient  $2^3 = 8$  configurations binaires et ceci permet de spécifier quelle entrée (de  $E_0$  à  $E_7$ ) sera dirigée vers la sortie S.
- ✓ Quelle que soit la valeur binaire présente sur  $C_0$ ,  $C_1$  et  $C_2$ , 7 portes parmi les huit ont leur propre sortie à la valeur 0.
- ✓ La huitième porte ET, la seule qui soit sélectionnée, voit sa sortie suivre les variations (0 ou 1) de son entrée associée  $E_i$  ( $0 \leq i \leq 7$ )



# Première partie

## *Circuits logiques de base*

### *les circuits logiques combinatoires*

#### **Application du multiplexeur :**

réalisation d'une *conversion parallèle/série*

- Supposons une information codée sur huit bits, présente (en parallèle) sur les lignes d'entrée  $E_0$  à  $E_7$ .
- Si l'on envoie successivement sur les lignes de sélection (ou commandes)  $C_0$ ,  $C_1$  et  $C_2$  les valeurs binaires de 000 à 111 de façon cyclique, les huit bits du mot d'entrée sont transmis sur la ligne de sortie les uns à la suite des autres (*en série*).

# Première partie

*Circuits logiques de base*

*les circuits logiques combinatoires*

## **EXERCICE :**

**réaliser la fonction majoritaire à l'aide d'un multiplexeur**

# Première partie

*Circuits logiques de base*

*les circuits logiques combinatoires*

## **EXERCICE :**

**réaliser la fonction majoritaire à l'aide d'un multiplexeur**

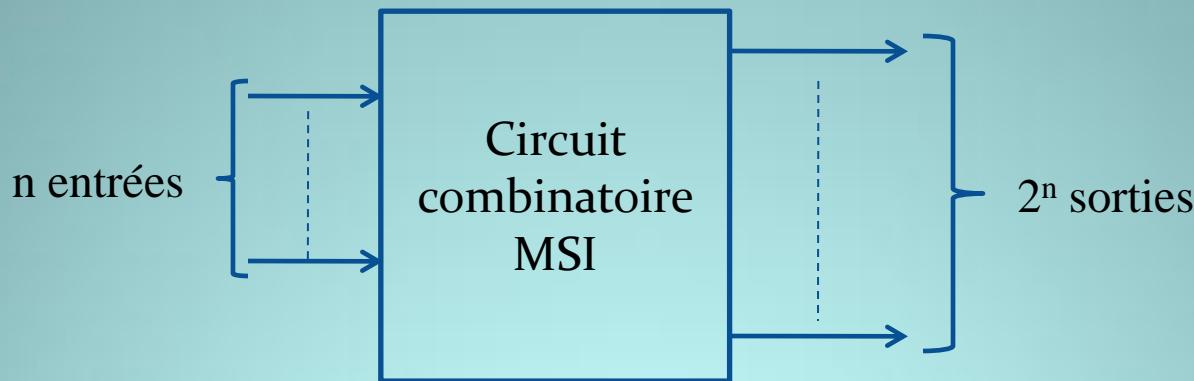
Pour les entrées de sélection il faut inverser les A,B et C soit à l'intérieur de multiplexeur soit grâce aux inverseurs câblés extérieurement.

# Première partie

## Circuits logiques de base

### les circuits logiques combinatoires

## Décodeur



Pour une configuration binaire en entrée, une seule sortie  
parmi les  $2^n$  possibles est active

# Première partie

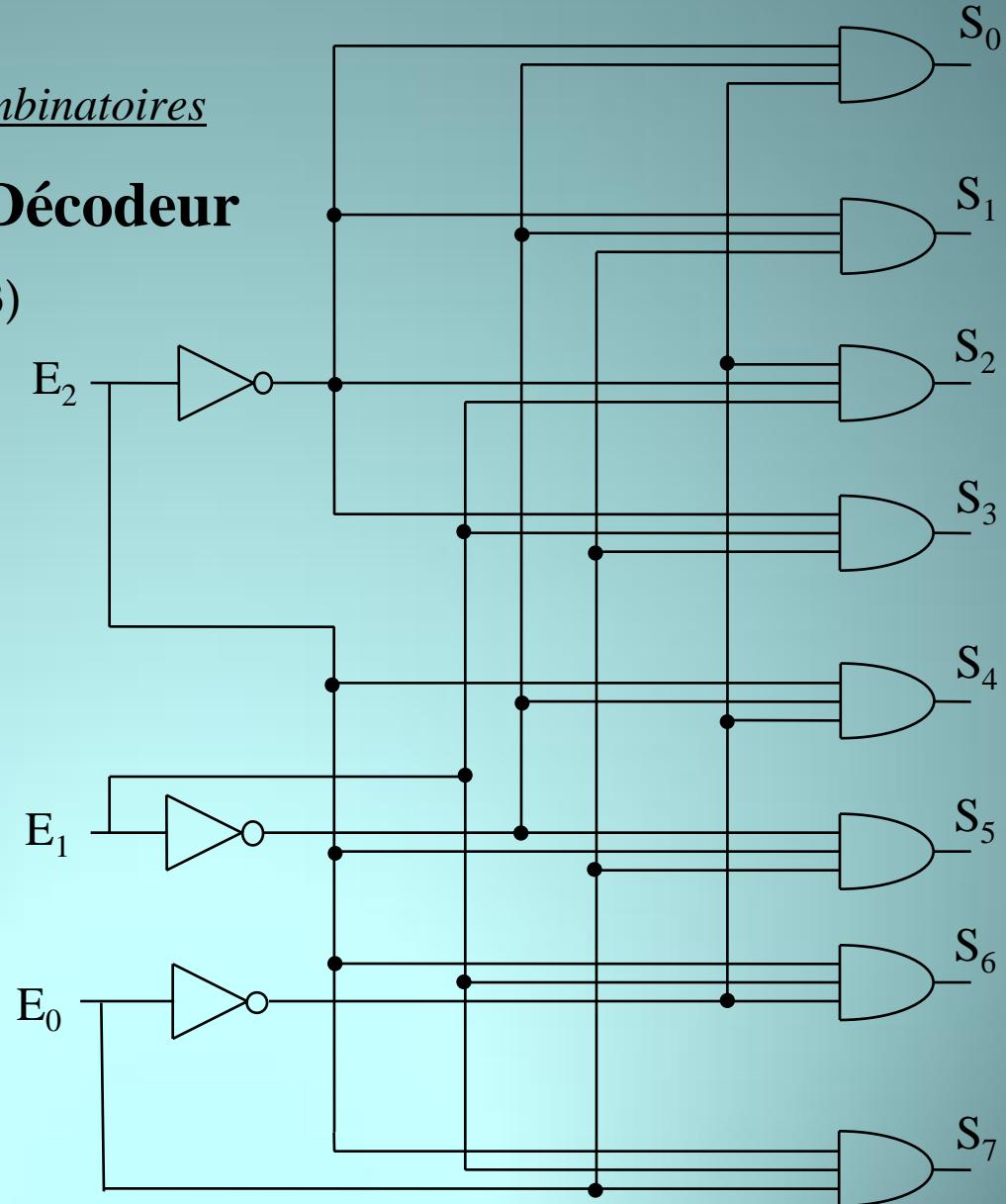
Circuits logiques de base

les circuits logiques combinatoires

## Décodeur

Exemple : décodeur 3 vers 8 ( $n=3$ )

E <sub>2</sub>	E <sub>1</sub>	E <sub>0</sub>	S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>	S <sub>6</sub>	S <sub>7</sub>
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0		1					0
0	1	1	0			1				0
1	0	0	0				1			0
1	0	1	0				1			0
1	1	0	0					1	0	
1	1	1	0	0	0	0	0	0	0	1



# Première partie

## *Circuits logiques de base*

### les circuits logiques combinatoires

#### Application 1 : la sélection de circuit mémoire

Soit une mémoire constituée de 8 circuits (ou boîtiers) de 1Ko chacun (D0, D1,...,D7).

Le circuit n° 0 contient les octets d'adresse 0-1023

Le circuit n° 1 contient les octets d'adresse 1024-2047 ...

D7 1FFF↔1C00	D6	D5	D4	D3 (FFF↔C00)	D2 3071↔2048 (BFF↔800)	D1 2047↔1024 (7FF↔400)	D0 1023↔0 (3FF↔000)
-----------------	----	----	----	-----------------	------------------------------	------------------------------	---------------------------

Les trois bits de poids fort de l'adresse (sur 16 bits) sont utilisés à la sélection des boîtiers mémoire (1 parmi 8)

adresse 16 bits      (o : non utilisé      x : 0 ou 1)

ooo 0 11 xx xxxx xxxx

bits de poids fort  
⇒ bits d'adresse

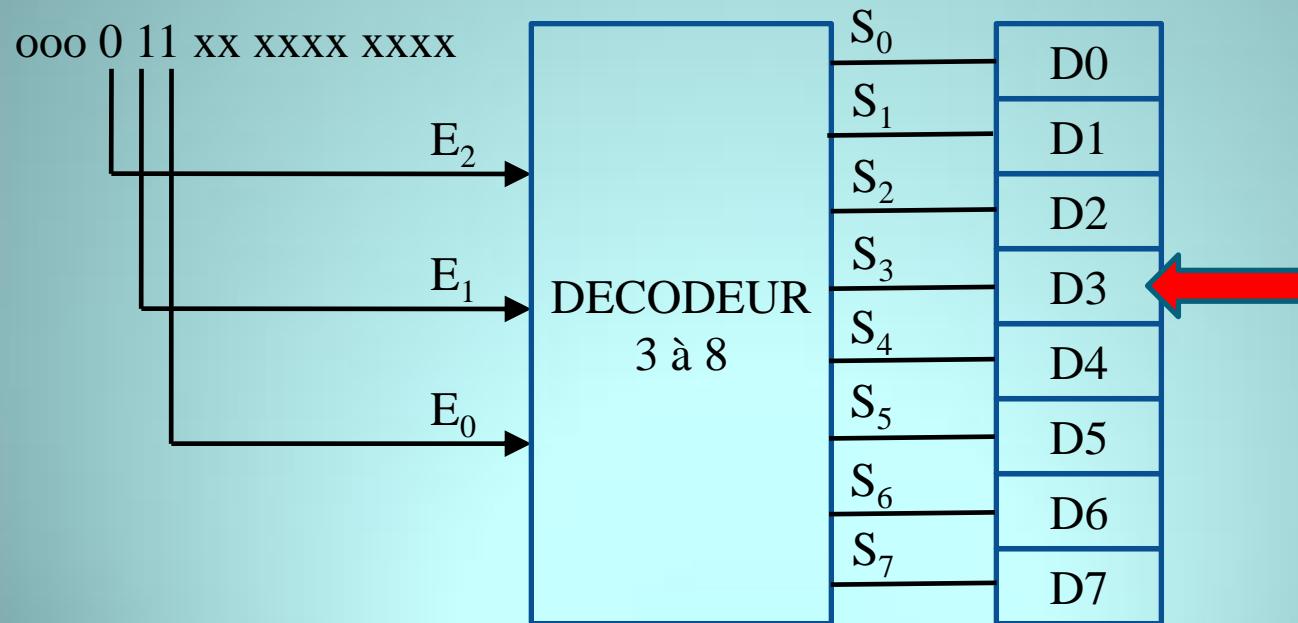
# Première partie

## Circuits logiques de base

### les circuits logiques combinatoires

Sur le décodeur 3 à 8 précédent, ses bits d'adresse sont reliés aux entrées  $E_0$ ,  $E_1$  et  $E_2$ . Selon l'état binaire présent sur ces bits, une seule sortie est à l'état 1 et les autres sont à l'état zéro.

Par exemple, l'adresse 000 0 11 xx xxxx xxxx de cette forme permet de sélectionner un octet du boîtier n° 3:



# Première partie

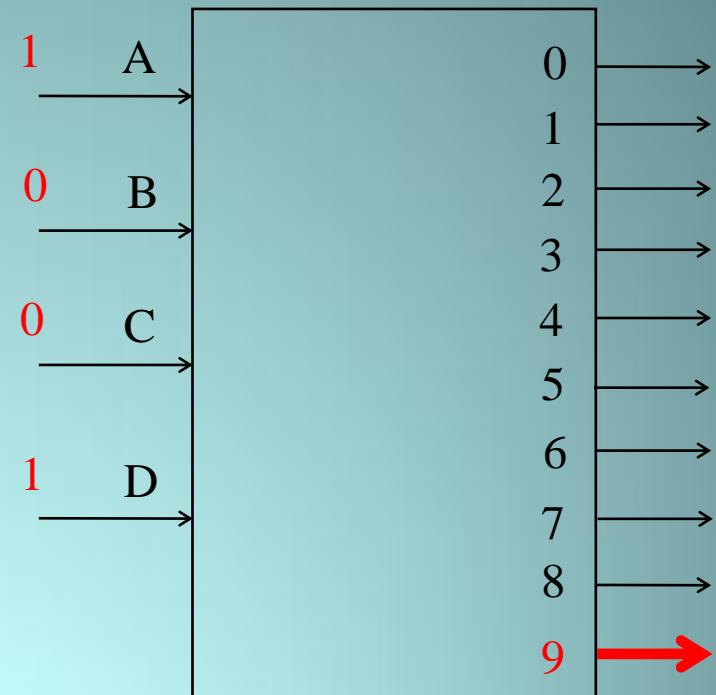
## *Circuits logiques de base*

### les circuits logiques combinatoires

Application 2 : on trouve des circuits spéciaux de type 4 vers 10 qui permettent le décodage des nombres décimaux codés en DCB (Décimal Codé Binaire)

Exemple : 312 est codé 0011 0001 0010 en DCB

*Entrées non prises en compte* : 1010 à 1111



# Première partie

## Circuits logiques de base

### les circuits logiques combinatoires

## comparateur

- ❖ Comparaison de deux mots de n bits chacun

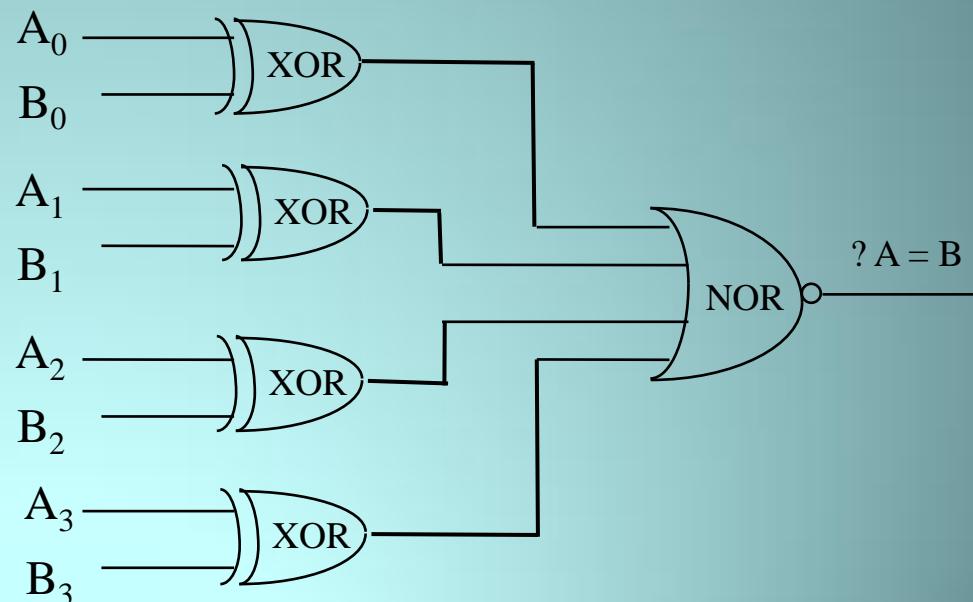
Exemple : un comparateur 4 bits :

$A=B$  ssi  $\text{NOR}(0,0,0,0) = 1$

$\text{XOR}(A_i, B_i) = 0$  ssi  $A_i = B_i$

- ❖ les circuits comparateurs du marché comportent en général 3 sorties :

$A=B$ ,  $A < B$ ,  $A > B$



# Première partie

## *Circuits logiques de base*

### *les circuits logiques combinatoires*

## les réseaux logiques programmables

- ❖ On sait réaliser une fonction logique à partir de sa T.V. ( avec ET, le produit, OU, la somme, etc...)
- ❖ Il existe des *composants* tout prêts adaptés à la réalisation directe des fonctions logiques mises sous forme "*somme de produits*"
- ❖ Ce sont les réseaux logiques programmables PLA ou FPLA (Field Programmable Logic Array)

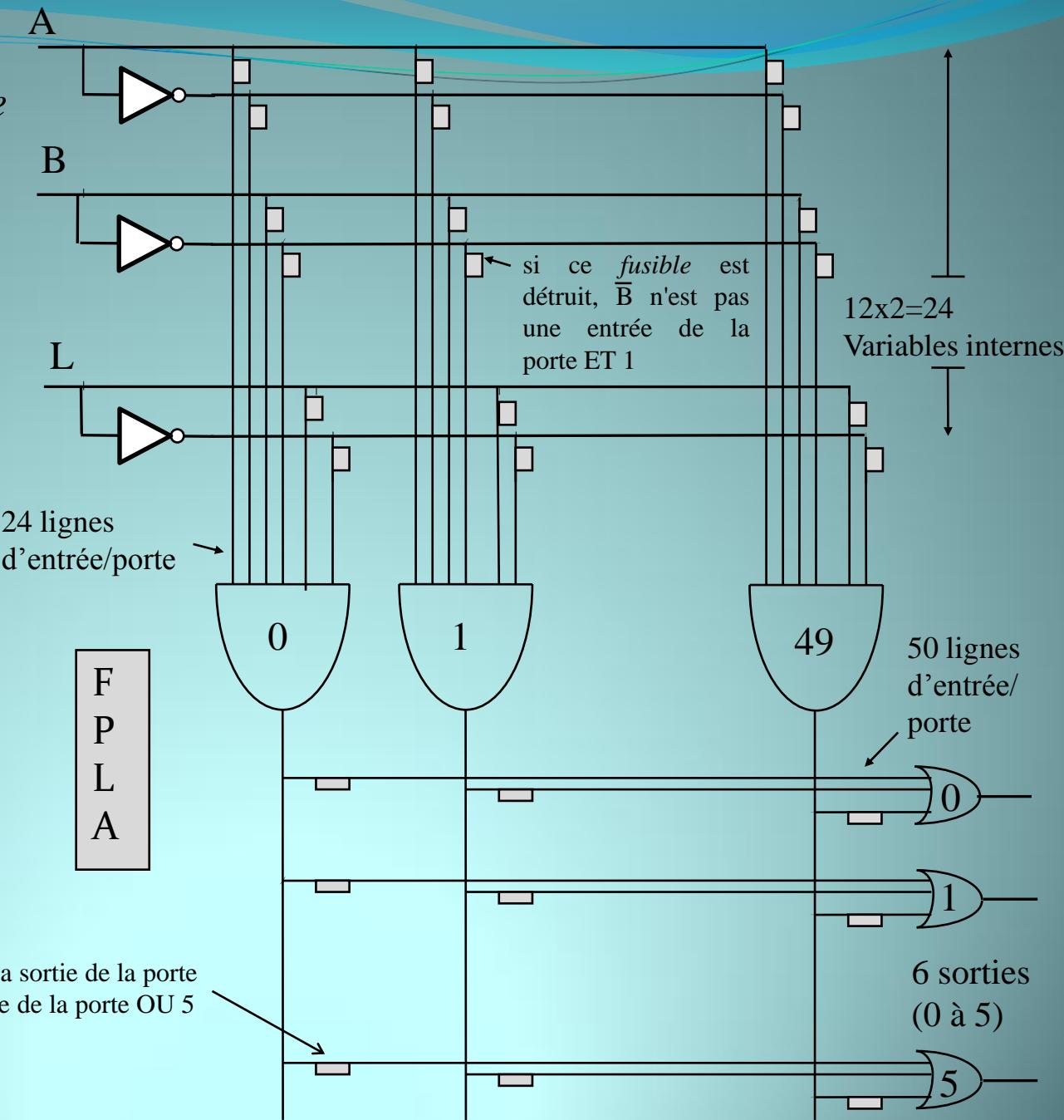
Exemple : le circuit FPLA suivant comprend 12 entrées et 6 sorties, 50 portes ET et se présente dans un boîtier de 20 broches (12 + 6 + GND + Vcc)

# Première partie

## Circuits logiques de base les circuits logiques combinatoires

❖ Le FPLA ci-contre a 12 entrées, 6 sorties, 50 portes ET et 6 portes OU. Les petits carrés sur le schéma représentent les fusibles permettant la programmation du FPLA.

❖ Chaque entrée des portes dispose d'un fusible → les 1500 fusibles sont intacts lors de la fabrication



# Première partie

*Circuits logiques de base*

*les circuits logiques combinatoires*

***EXERCICE*** : réaliser, avec ce circuit FPLA la fonction logique majoritaire M

# Première partie

## *Circuits logiques de base*

### *les circuits logiques combinatoires*

**Remarque 1** : ce circuit FPLA pourrait être programmé pour traiter plusieurs fonctions indépendantes : il est donc ici sous-utilisé

**Remarque 2** : intérêt des circuits FPLA pour la réalisation des fonctions logiques combinatoires, par rapport aux autres modes de conception analysés.

# Première partie

*Circuits logiques de base (combinatoires)*

## Sujets abordés :

\*les circuits intégrés logiques (CIL)

\*les circuits logiques combinatoires (multiplexeurs, décodeurs, comparateurs, réseau logique programmable)

**\*les circuits de traitement ou de calcul  
(décaleur, additionneur, UAL)**

\*les horloges

# Première partie

*Circuits logiques de base*

*les circuits de calcul*

**Sujets abordés :**

- le décaleur
- l'additionneur
- l'UAL (Unité Arithmétique et Logique)

# Première partie

*Circuits logiques de base*

*les circuits de calcul*

Sujets abordés :

- le décaleur
- l'additionneur
- l'UAL (Unité Arithmétique et Logique)

# Première partie

## Circuits logiques de base

### les circuits de calcul → décaleur

Le décaleur permet de décaler un mot d'information d'un bit (à droite ou à gauche) en fonction d'un signal de commande C :

$C=1$  : décalage à droite

$C=0$  : décalage à gauche

Les 8 bits de sortie  $S_0$  à  $S_7$  correspondent à l'entrée  $D_0, \dots, D_7$  après décalage.

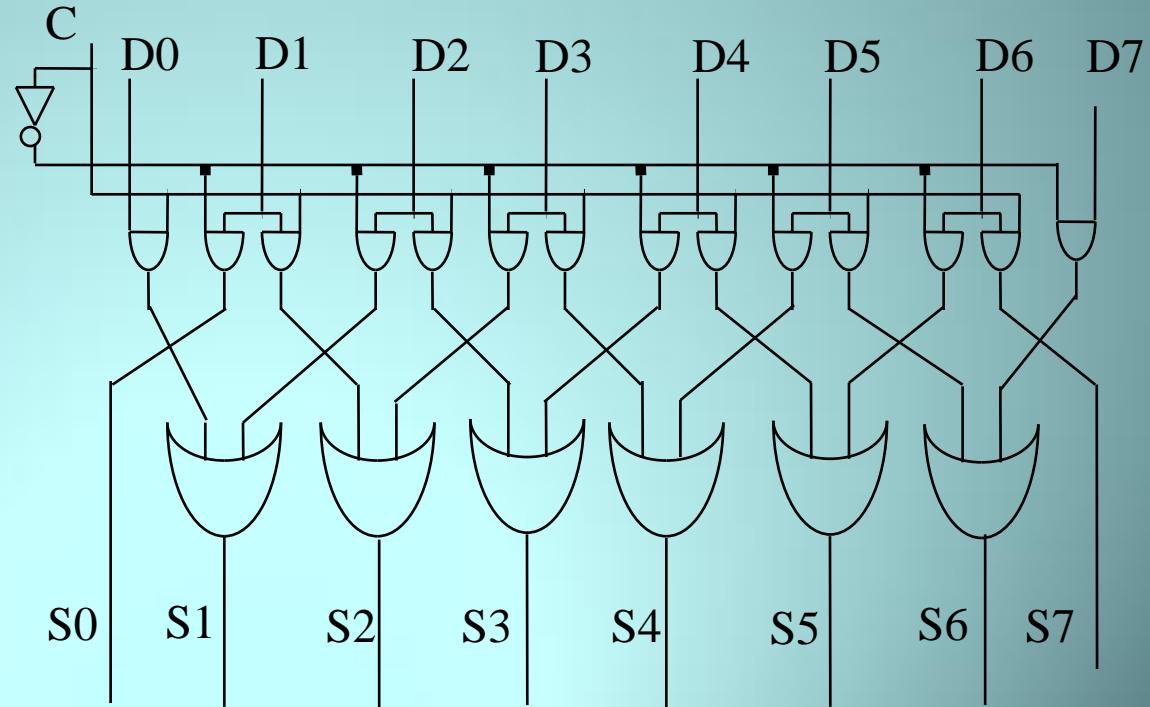
En fait , pour  $i=1, \dots, 6$

$$S_i = D_{i-1} \cdot C + D_{i+1} \cdot \bar{C}$$

$$S_0 = D_1 \cdot C$$

$$S_7 = D_6 \cdot C$$

Remarque : lors d'un décalage à gauche [resp. à droite], un bit peut être perdu à gauche [resp. à droite] et un bit zéro peut être généré à droite [resp. à gauche]



# Première partie

*Circuits logiques de base*

*les circuits de calcul*

Sujets abordés :

- le décaleur
- l'additionneur
- l'UAL (Unité Arithmétique et Logique)

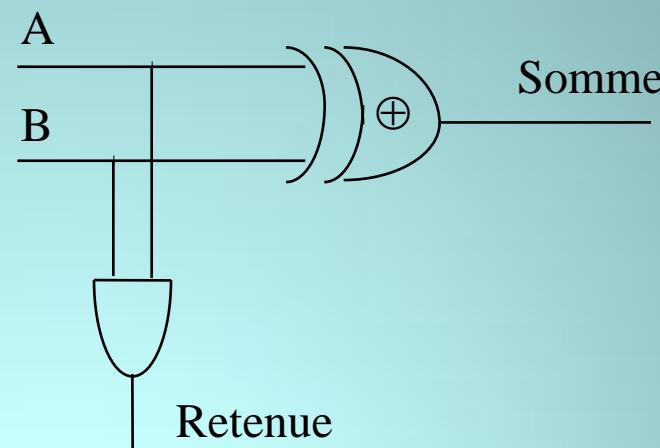
# Première partie

## Circuits logiques de base

les circuits de calcul → additionneur

1/2 additionneur pour réaliser la somme de deux bits

A	B	Somme	Retenue
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



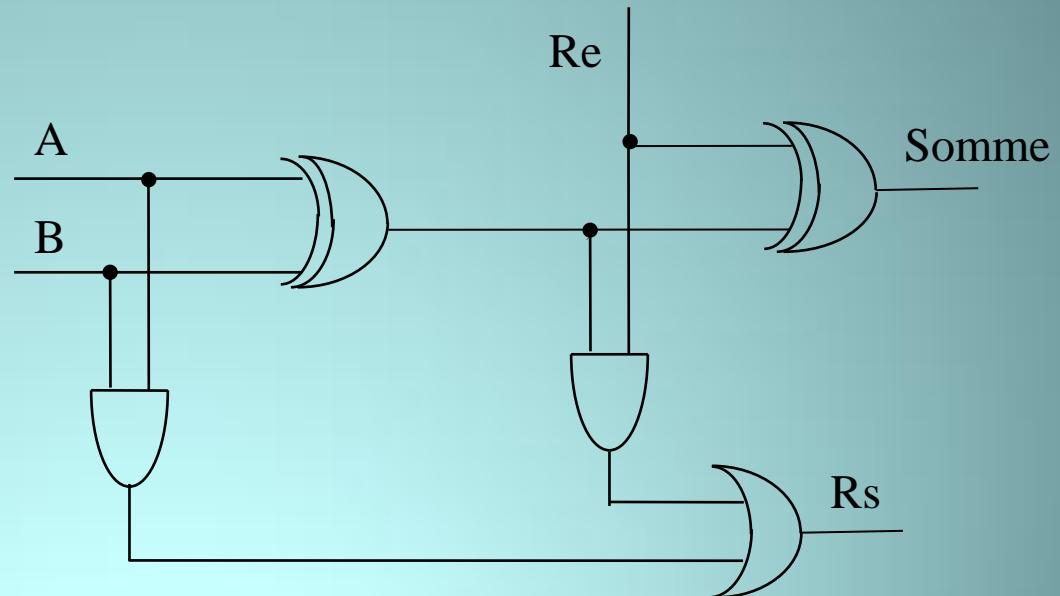
# Première partie

## Circuits logiques de base

les circuits de calcul → additionneur

**1 additionneur complet** pour effectuer la somme de deux nombres binaires quelconques

A	B	Retenue Re	Somme	Retenue Rs
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



EXERCICE : Ecrire les expressions logique de Somme et Rs en fonction de A, B et Re

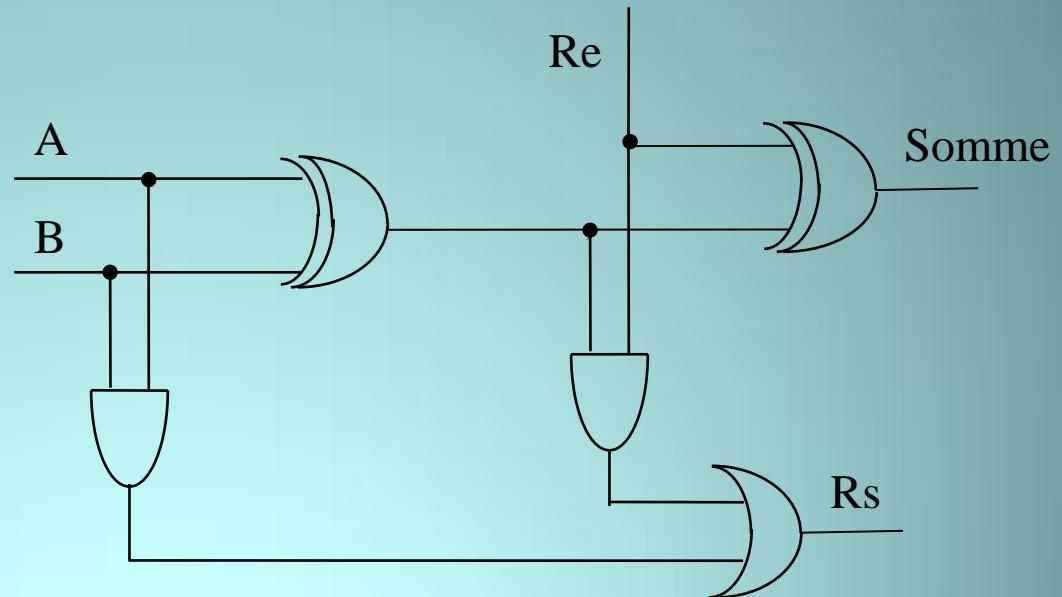
# Première partie

*Circuits logiques de base*

les circuits de calcul → additionneur

## 1 additionneur complet 1 bit

A	B	Retenue Re	Somme	Retenue Rs
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



EXERCICE : Ecrire les expressions logique de Somme et Rs en fonction de A, B et Re

Réponse :  $Rs = A \cdot B + \text{XOR}(A, B) \cdot Re$

et

$\text{Somme} = \text{XOR}(\text{Re}, \text{XOR}(A, B))$

# Première partie

## *Circuits logiques de base*

les circuits de calcul → additionneur

*Généralisation :*

**pour construire un additionneur de nombres de n bits, il suffit d'utiliser n additionneurs «en cascade», la sortie de retenue Rs d'un étage correspondant à l'entrée de retenue Re de l'étage qui suit.**

# Première partie

*Circuits logiques de base*

*les circuits de calcul*

Sujets abordés :

- le décaleur
- l'additionneur
- l'**UAL (Unité Arithmétique et Logique)**

# Première partie

## *Circuits logiques de base*

### les circuits de calcul → UAL

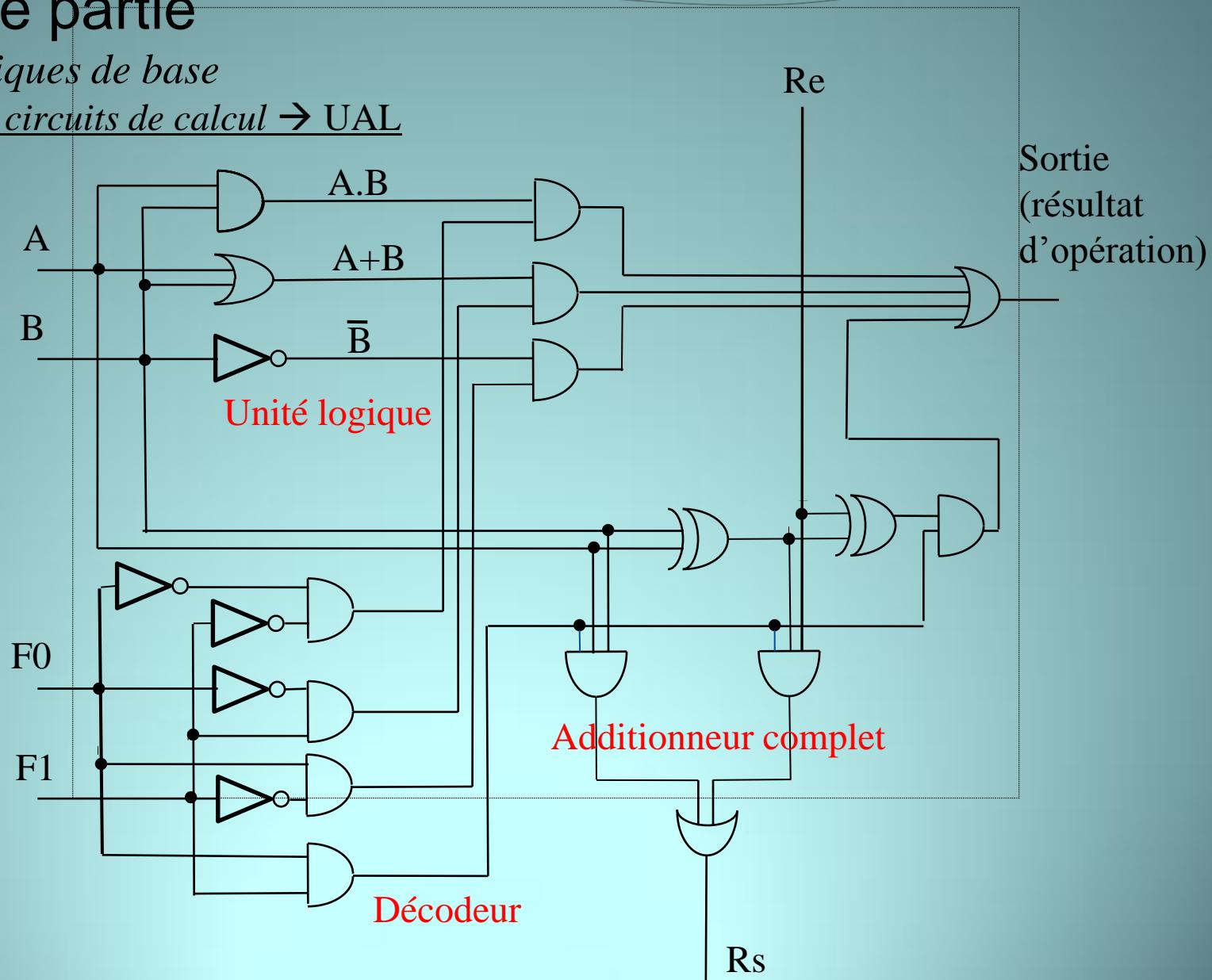
- Les ordinateurs disposent d'une Unité réalisant :
  - ❖ les opérations logiques de base : ET, OU, NON
  - ❖ la somme de deux nombres binaires
- Pour traiter des nombres sur n bits, on associe par blocs de n les différents opérateurs sur 1 bit

# Première partie

Circuits logiques de base

les circuits de calcul → UAL

**EXEMPLE :**  
UAL de 1 bit



# Première partie

*Circuits logiques de base*

*les circuits de calcul → UAL*

## Questions

- 1. Combien d'opérations l'UAL traite-t-elle ?**
- 2. Lesquelles ?**
- 3. Comment choisit on l'opération à effectuer ?**
- 4. Où est transmis le résultat ?**
- 5. Quelle est la nature du résultat ?**
- 6. Quel est le rôle joué par l'additionneur complet ?**
- 7. Comment réaliser les quatre opérations de base sur des nombres de plusieurs bits ?**

# Première partie

*Circuits logiques de base*

les circuits de calcul → UAL

## Réponses

1. 4 opérations
2. A ET B, A OU B, NON B et A + B (addition)
3. A partir d'un décodeur à deux bits: entrés  $F_0$  et  $F_1$  (00,01, 10,11)
4. à la sortie
5. Le résultat de l'une des 4 opérations; les 3 autres sorties des opérateurs transmettent la valeur zéro vers la sortie
6. Il réalise la somme arithmétique  $A+B$  transmise en sortie en tenant compte d'une éventuelle entrée de retenue et transmet à l'extérieur sa sortie de retenue  $R_s$
7. On associe plusieurs UAL. 1 bit en // : ce sont des circuits ou microprocesseurs en tranches => on peut bâtir des unités de traitement sur mesure

# Première partie

*Circuits logiques de base (combinatoires)*

## Sujets abordés :

- \*les circuits intégrés logiques (CIL)
- \*les circuits logiques combinatoires (multiplexeurs, décodeurs, comparateurs, réseau logique programmable)
- \*les circuits de traitement ou de calcul (décaleur, additionneur, UAL)
- \*les horloges

# Première partie

## *Circuits logiques de base les horloges*

❖ But :

pour assurer

la séquentialité

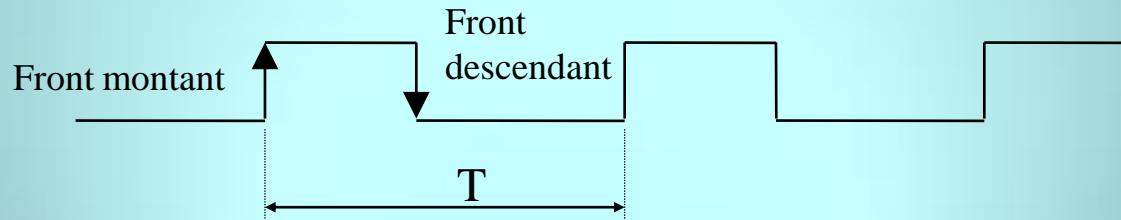
la synchronisation temporelle

des variables logiques , on utilise des *horloges*

❖ **HORLOGE** : système logique qui émet des *impulsions calibrées* à intervalle de temps réguliers

❖ L'intervalle de temps entre deux impulsions représente le *temps de cycle* ou la *période T* de l'horloge

La fréquence ( $1/T$ ) est comprise entre 1 et 100 MHz, soit T de 1  $\mu$ s (microseconde) à 100  $\mu$ s

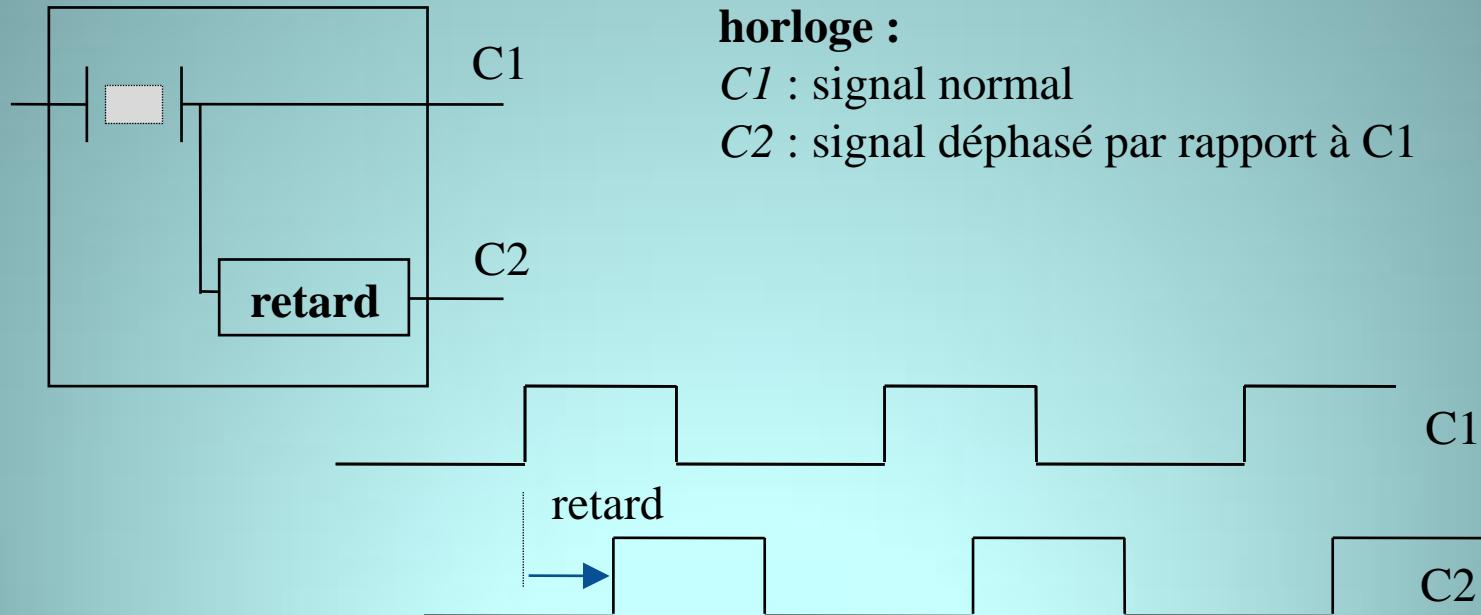


# Première partie

## *Circuits logiques de base*

### les horloges

- ❖ Pour plus de précision, les horloges sont pilotées par des oscillateurs à quartz



# Première partie

## *Circuits logiques de base les horloges*

Ainsi, on peut réaliser un séquencement déterminé en attribuant des actions à chacun de ces instants : 4 actions différentes pourraient être déterminées à partir de C1 et C2 :

- ❖  $\overline{C1}$  et  $\overline{C2}$
- ❖  $\overline{C1}$  et  $C2$
- ❖  $C1$  et  $\overline{C2}$
- ❖  $C1$  et  $C2$

En général, les horloges délivrent des signaux symétriques : le temps de maintien au niveau Haut est égal au temps de maintien au niveau Bas

# Première partie

*Circuits logiques de base  
les horloges*

**EXERCICE : comment fournir un signal asymétrique à partir de signaux symétriques ?**

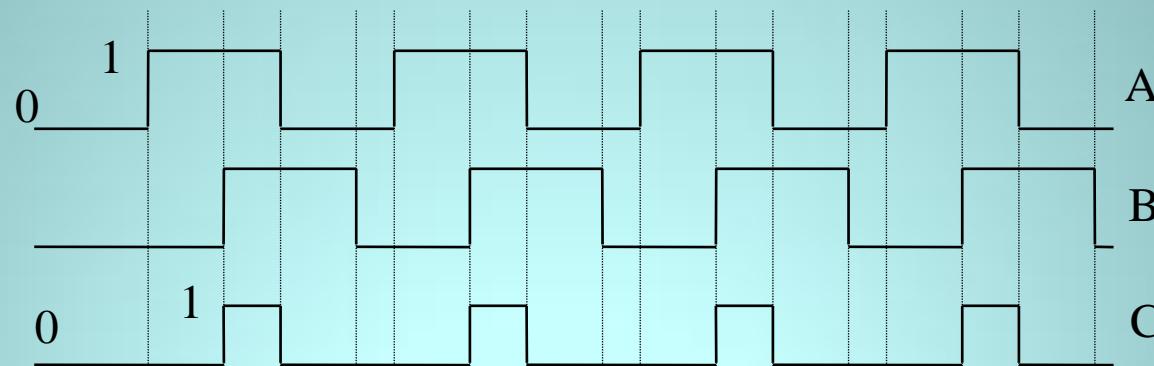
# Première partie

## *Circuits logiques de base*

### les horloges

**EXERCICE : comment fournir un signal asymétrique à partir de signaux symétriques ?**

**Réponse :  $C = A \text{ ET } B$**



# Première partie

## • **Fondements théoriques et technologiques**

*Ordinateur ← Circuits de base très simples dont le comportement fonctionnel est décrit par l'algèbre binaire (algèbre de Boole)*

- I. codage(s) de l'information
- II. portes logiques et algèbre de Boole
- III. circuits logiques de base (combinatoires)
- IV. circuits logiques à mémoire (séquentiels)**

# Première partie

## *Circuits logiques à mémoire*

**La *mémoire* d'un ordinateur : organe essentiel car les programmes et les données y sont stockés**

**But de l'étude :**

- ✓ examiner les composants qui forment les *mémoires*
- ✓ voir la façon d'associer ces composants pour former des *circuits de mémorisation*

# Première partie

## *Circuits logiques à mémoire*

### Sujets abordés :

- ❖ les bascules (flip-flops)
- ❖ les bascules (flip-flops) déclenchées sur front d'horloge
- ❖ les circuits interrupteurs trois états
- ❖ bus de données
- ❖ Registre
- ❖ organisation des mémoires

# Première partie

## *Circuits logiques à mémoire*

Sujets abordés :

### ❖ les bascules (flip-flops)

- ❖ les bascules (flip-flops) déclenchées sur front d'horloge
- ❖ les circuits interrupteurs trois états
- ❖ bus de données
- ❖ Registre
- ❖ organisation des mémoires

# Première partie

## *Circuits logiques à mémoire les bascules*

**But : mémoriser un élément binaire**

→un dispositif qui se "souvienne" de la valeur enregistrée  
à l'aide de deux portes logiques NON-OU ou NON-ET

# Première partie

## Circuits logiques à mémoire

les bascules → Bascule RS

### Bascule RS

❖ deux entrées

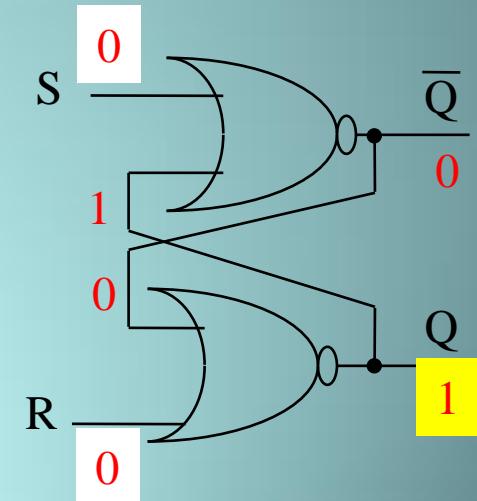
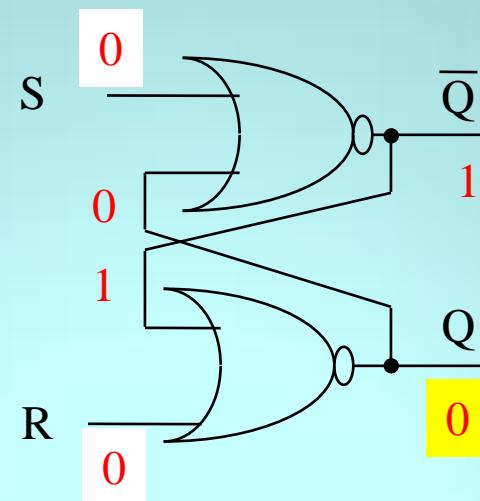
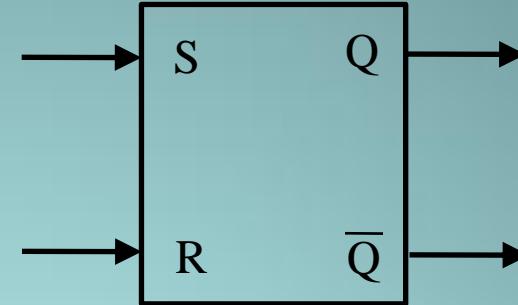
S[et] pour la mise à 1 de la bascule

R[eset] pour la mise à l'état zéro

❖ deux sorties Q et NON(Q)

complémentaires

❖ A l'inverse des circuits combinatoires, la valeur de sortie de la bascule ne dépend pas que des valeurs des variables d'entrée, mais aussi de la valeur antérieure de la sortie



# Première partie

## *Circuits logiques à mémoire*

les bascules → Bascule RS

**Exercice :**

Montrer que

→ quand  $R[\text{eset}] = 0$ ,  $S[\text{et}]$  passe de 0 à 1,  $Q = 1$  ( $\overline{Q} = 0$ )

→ quand  $S[\text{et}] = 0$ ,  $R[\text{eset}]$  passe de 0 à 1,  $Q = 0$  ( $\overline{Q} = 1$ )

Avec la remarque précédente, quel est le fonctionnement de la bascule RS ?

# Première partie

## *Circuits logiques à mémoire*

les bascules → Bascule RS

### Exercice :

Montrer que

→ quand R[eset] = 0, S[et] passe de 0 à 1, Q = 1 ( $\overline{Q} = 0$ )

→ quand S[et] = 0, R[eset] passe de 0 à 1, Q = 0 ( $\overline{Q} = 1$ )

Avec la remarque précédente, quel est le fonctionnement de la bascule RS ?

⇒ l'effacement des valeurs introduites est sans effet

On peut donc dire qu'une bascule RS "se souvient" de l'action antérieure de R ou S (passage temporaire de 0 à 1)

C'est cette propriété de mémorisation qui est mise en œuvre dans les mémoires.

# Première partie

## *Circuits logiques à mémoire*

les bascules → Bascule RS

### Exercice

- ❖ Que valent Q et  $\bar{Q}$  si R = S = 1 ?
- ❖ Avec la condition précédente, on fait maintenant passer tout les deux (R et S) à zéro.

Que se passe-t-il si S passe à zéro avant R ?

Que se passe-t-il si R passe à zéro avant S ?

### Remarques :

- ❖ la bascule RS peut présenter une instabilité de fonctionnement lorsque ses entrées R et S sont égales à 1 en même temps
- ❖ on ne peut pas prévoir à l'avance quel état stable va prendre la bascule lorsque R et S ne sont pas passées à zéro simultanément

# Première partie

## *Circuits logiques à mémoire*

Sujets abordés :

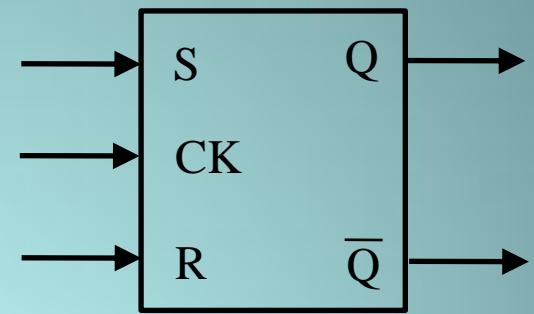
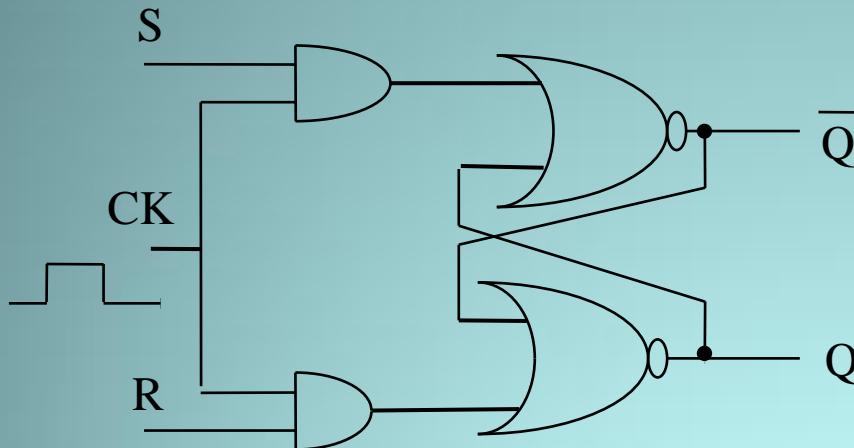
- ❖ les bascules (flip-flops)
- ❖ **les bascules (flip-flops) déclenchées sur front d'horloge**
- ❖ les circuits interrupteurs trois états
- ❖ bus de données
- ❖ Registre
- ❖ organisation des mémoires

# Première partie

Circuits logiques à mémoire

les bascules déclenchées sur front d'horloge

Bascule RS recommandée par un niveau d'horloge



Changer l'état d'une bascule à des instants bien précis

→ un système *commandé par une horloge (Clock)*

**CK = 0 : sortie(ET) = 0 quels que soient R et S; la bascule reste à l'état pris antérieurement**

**CK = 1 : les portes ET ouvertes, les entrées R et S sont transmises sur la bascule.**

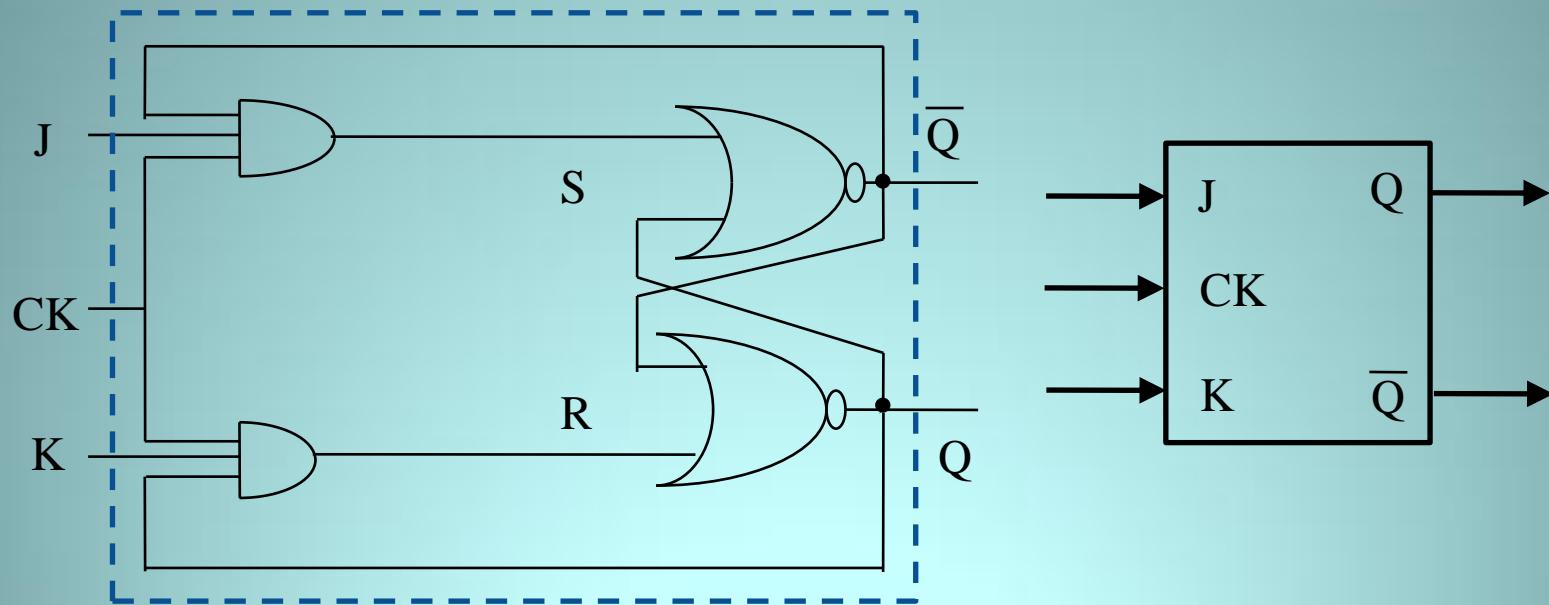
# Première partie

Circuits logiques à mémoire

les bascules déclenchées sur front d'horloge

## Bascule JK

⇒ éviter l'ambiguïté de fonctionnement de la bascule RS par la bascule JK



⇒ Les sorties Q et  $\bar{Q}$  sont rebouclées sur les entrées K et J via des portes ET

# Première partie

## Circuits logiques à mémoire

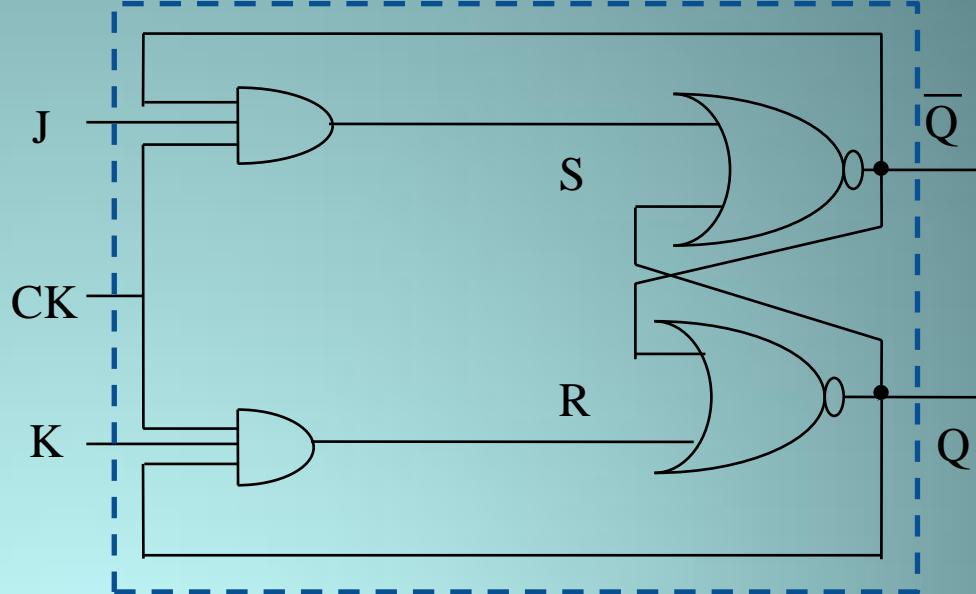
les bascules déclenchées sur front d'horloge

### Exercice

Quel est l'effet de  $J = K = 1$

- si  $Q = 0$  ?
- si  $Q = 1$  ?

On suppose que J et K passent ensuite à zéro.



# Première partie

## Circuits logiques à mémoire

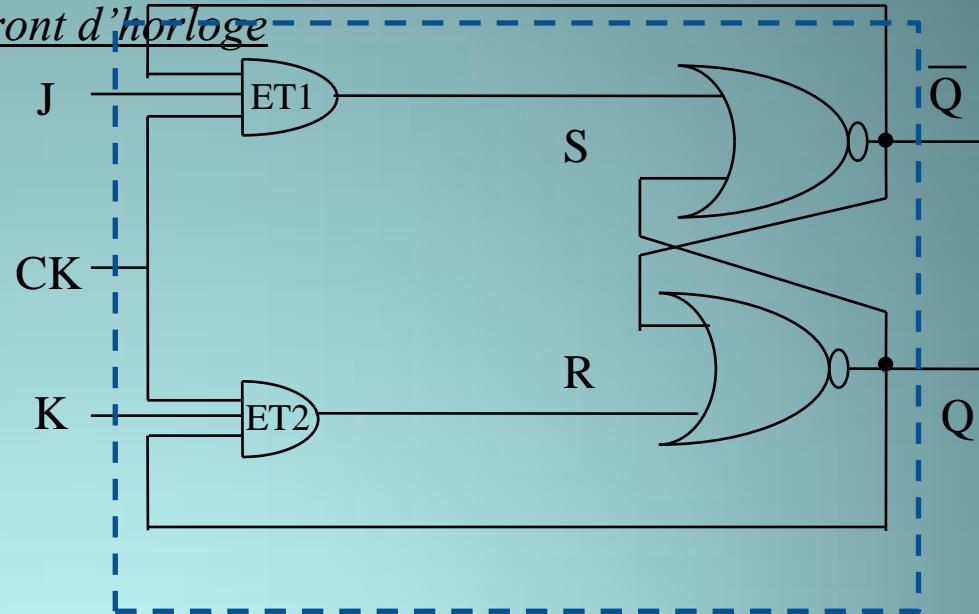
les bascules déclenchées sur front d'horloge

### Exercice

Quel est l'effet de  $J = K = 1$

- si dans l'état d'origine  $Q = 0$  ?
- si dans l'état d'origine  $Q = 1$  ?

On suppose que J et K passent ensuite à zéro.



### REPONSE :

➤ dans l'état d'origine  $Q=0$

ET1 active ( $S=1$ )  $\Rightarrow$  Set  $\Rightarrow Q=1$

➤ dans l'état d'origine  $Q=1$

ET2 active ( $R=1$ )  $\Rightarrow$  Reset  $\Rightarrow Q=0$

ET2 bloquée, si J et K repassent à 0

ET1 bloquée, si J et K repassent à 0

Et, de plus ces deux états stables sont prévisibles: il n'y a plus d'ambiguïté comme avec RS.

### Remarque 1:

- les entrées J et K ont un effet sur la bascule seulement lorsque l'entrée d'horloge est à 1.
- les bascules JK ont un rôle essentiel dans la constitution des circuits appelés compteurs.

# Première partie

*Circuits logiques à mémoire*

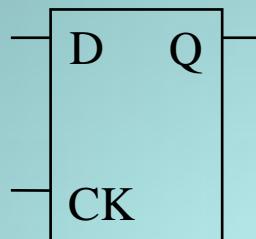
*les bascules déclenchées sur front d'horloge*

les bascules (flip-flops) déclenchées sur front d'horloge

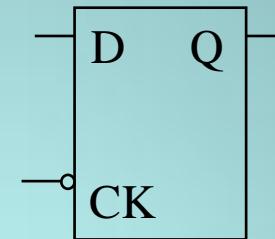
**Remarque** : la propriété de mémorisation  
d'une bascule est garantie lorsque  
l'impulsion d'horloge est de durée très faible



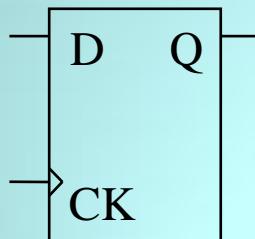
**la transition (ou front, "edge")**



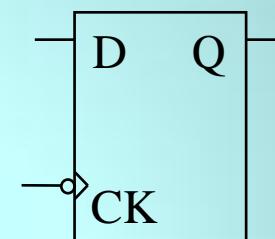
**niveaux** haut (CK=1)



**niveaux** bas (CK=0)



**front** montant (transition 0 à 1)  
*leading edge*



**front** descendant (transition 1 à 0)  
*trailing edge*

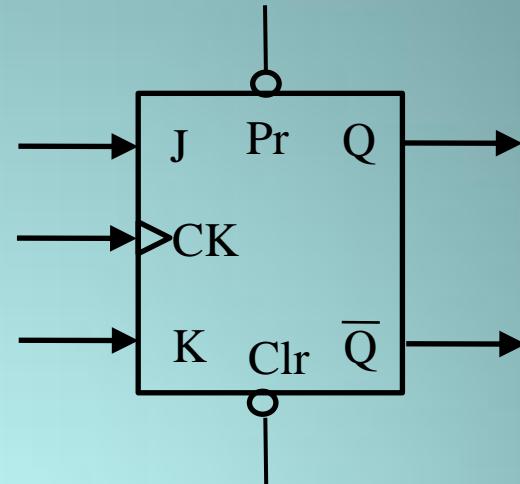
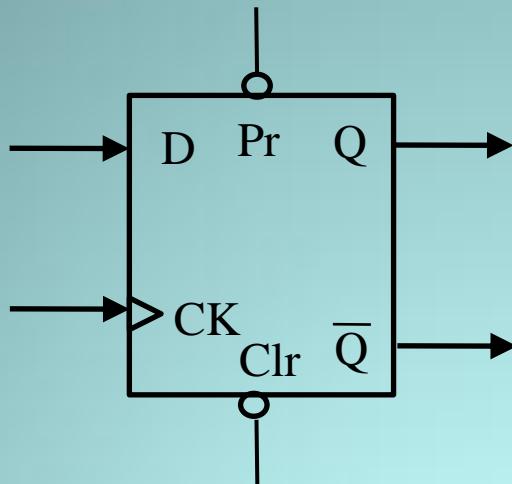
\*Ceci  
s'applique  
aussi aux  
bascules JK

# Première partie

*Circuits logiques à mémoire*

*les bascules déclenchées sur front d'horloge*

**les bascules (flip-flops) commercialisées**



**Deux entrées supplémentaires :**

\***Clear ou Reset** force la bascule directement à l'état  $Q = 0$

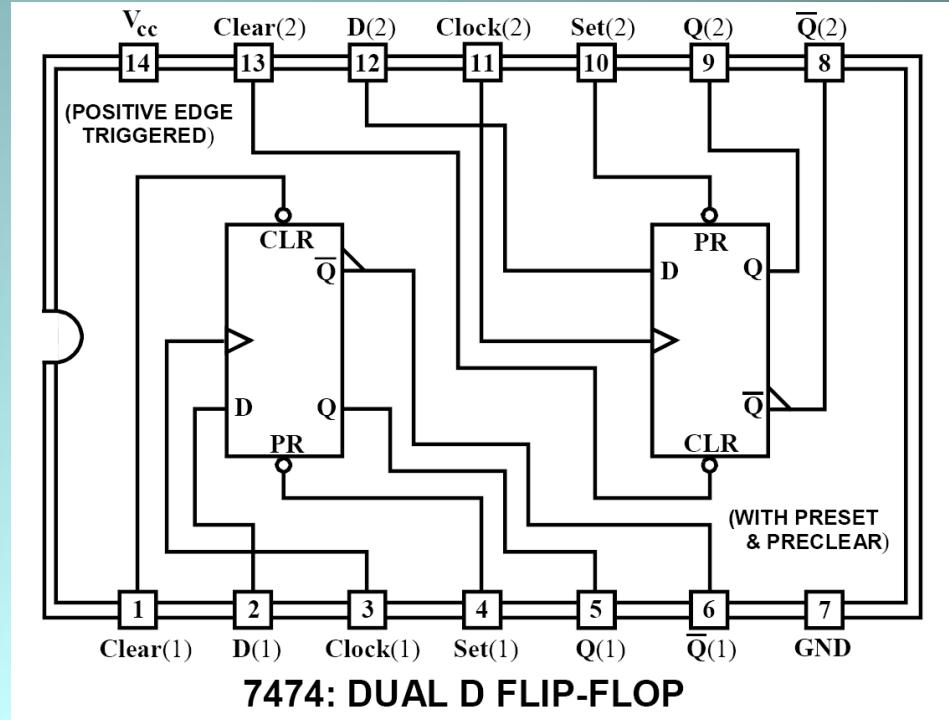
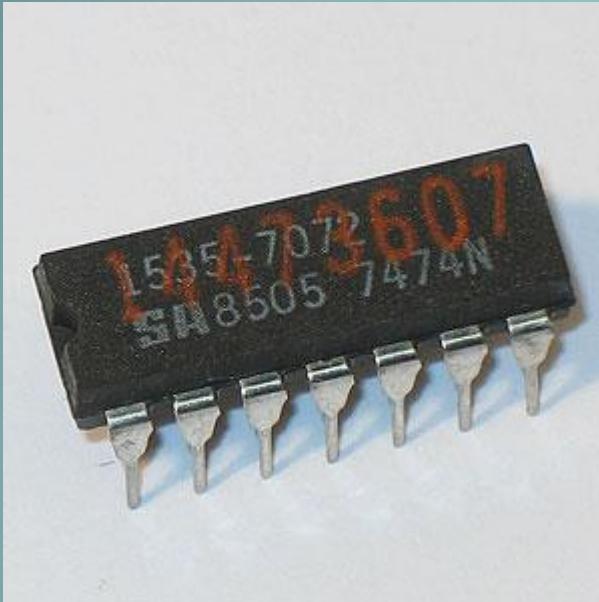
\***Preset ou Set** force la bascule directement à l'état  $Q = 1$

# Première partie

*Circuits logiques à mémoire*

les bascules déclenchées sur front d'horloge

Exemple : circuit 7474



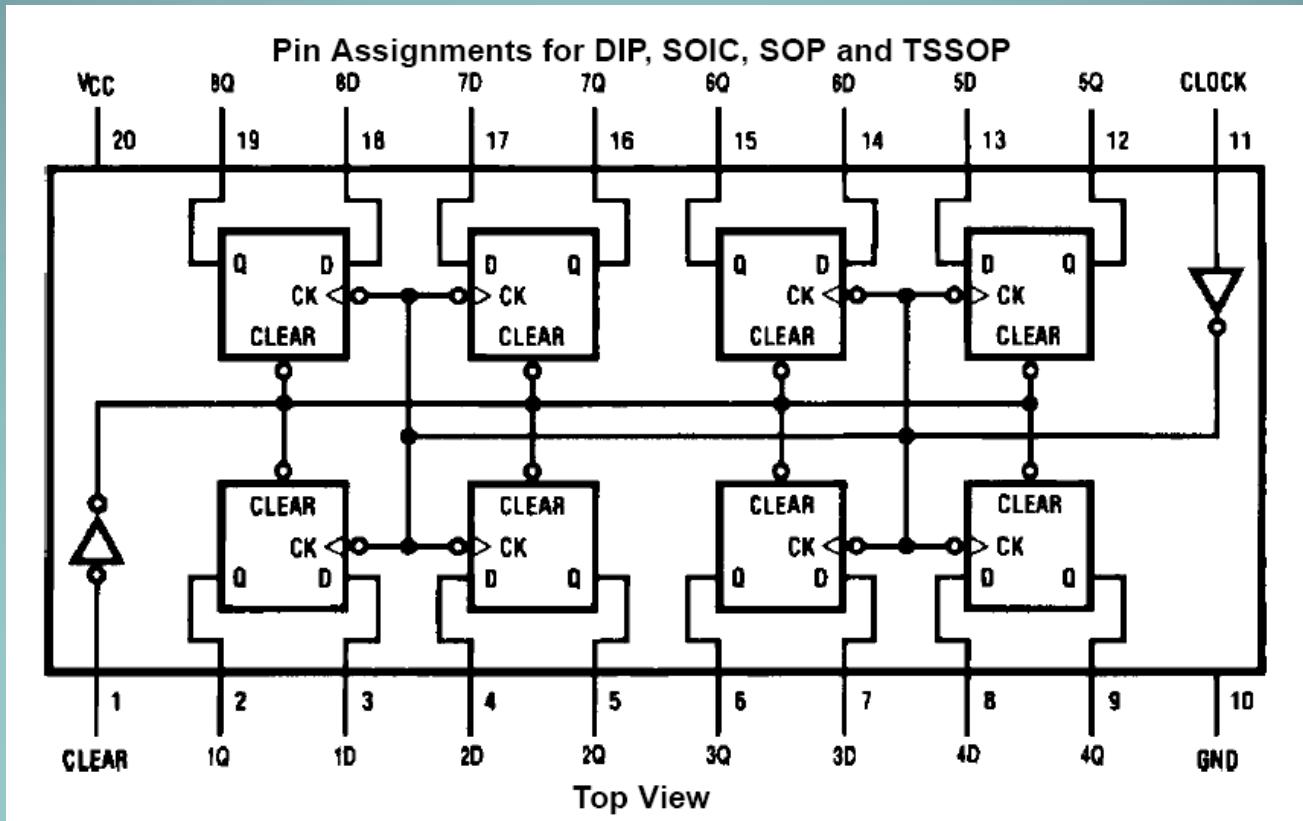
2 bascules D identiques et indépendantes (seule l'alimentation électrique est commune)

# Première partie

## Circuits logiques à mémoire

les bascules déclenchées sur front d'horloge

Exemple :  
circuit  
74273



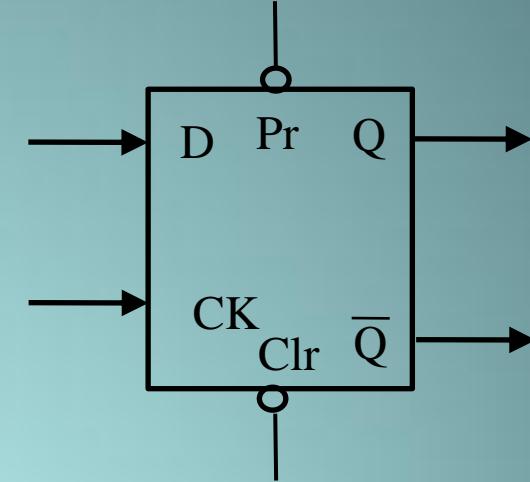
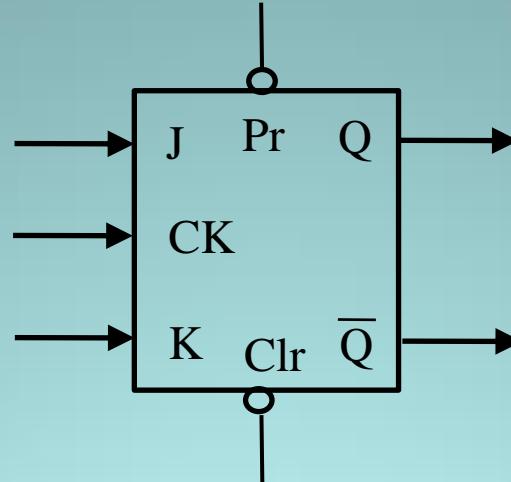
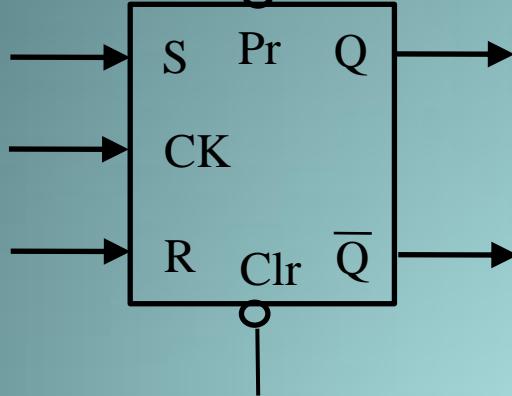
8 bascules D identiques et non indépendantes (ces bascules n'ont pas de sortie  $\bar{Q}$  et pas d'entrée PR)

⇒ un *registre* de huit bits permettant de mémoriser un mot d'information de 8 bits.

# Première partie

## Circuits logiques à mémoire

les tables de vérité



S(t)	R(t)	$Q(t+1) = \underline{S(t)+Q(t)R(t)}$
0	0	Q(t)
0	1	0
1	0	1
1	1	interdit

J(t)	K(t)	$Q(t+1) = J(t)\overline{Q(t)} + \overline{K(t)}Q(t)$
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

D(t)	$Q(t+1) = D(t)$
0	0
1	1

# Première partie

## Circuits logiques à mémoire

### les applications des bascules

#### APPLICATION 1 :

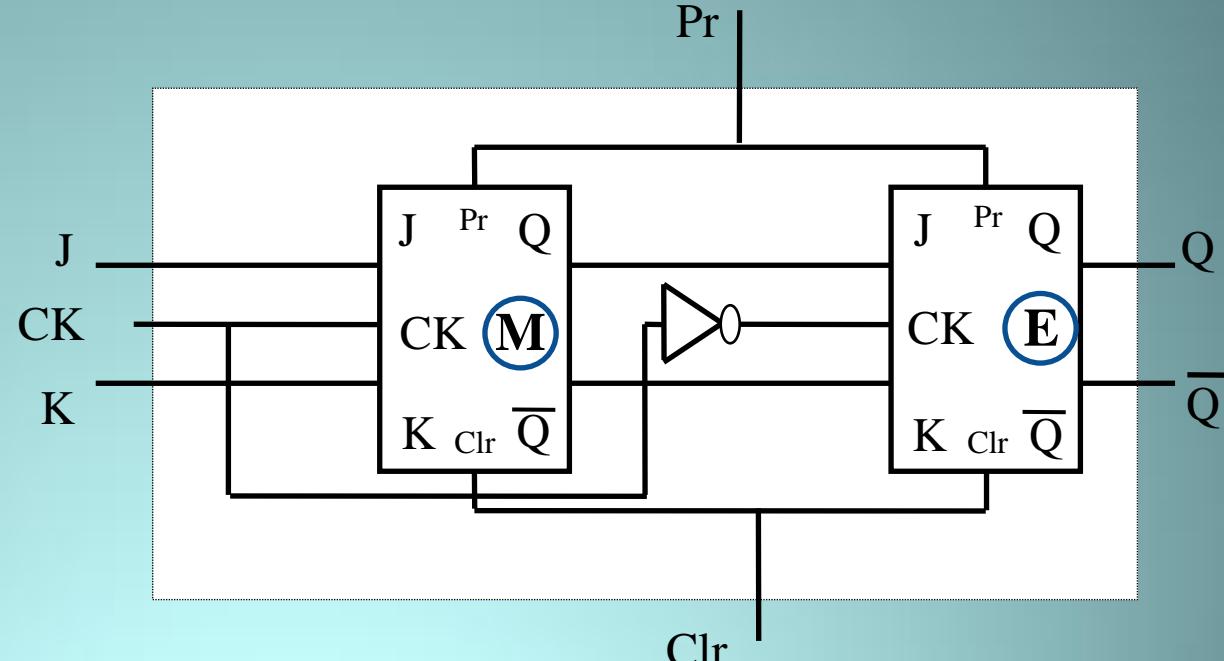
##### BASCULE

##### MAITRE/ESCLAVE

les sorties changent lorsque le signal d'horloge est *encore* à l'état haut → problème

⇒ l'horloge ne doit pas être activée trop longtemps de façon à ce que Q et  $\bar{Q}$  ne changent pas pendant que l'horloge est encore activée  
⇒ MAITRE/ESCLAVE :

Cette bascule M/E peut être *lue et modifiée en même temps*



- {
- \*CK active (=1) → M activée et E inactivée
  - \*CK inactivée (=0) → M inactivée et E prend l'état de M et les sorties sont modifiées après que CK soit passé à zéro

# Première partie

*Circuits logiques à mémoire*

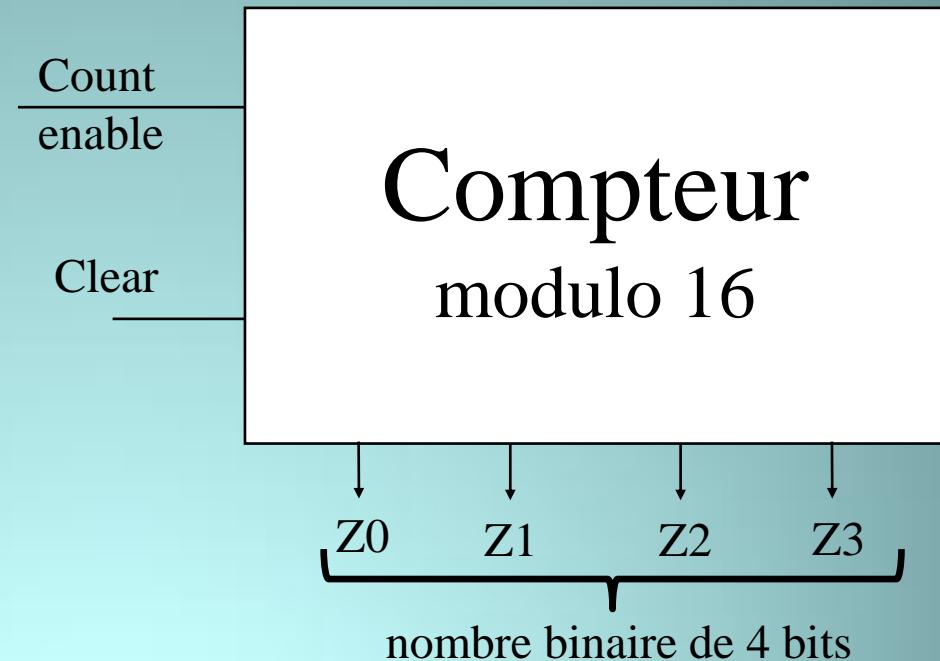
les applications des bascules

## APPLICATION 1 : COMPTEUR

Définition : un compteur est une machine séquentielle conçue pour prendre cycliquement une séquence prédéterminée de  $k$  états (nombres consécutifs)

$$S_0, S_1, \dots, S_{k-1}$$

avec  $S_{i+1} = S_i + 1$  modulo  $k$



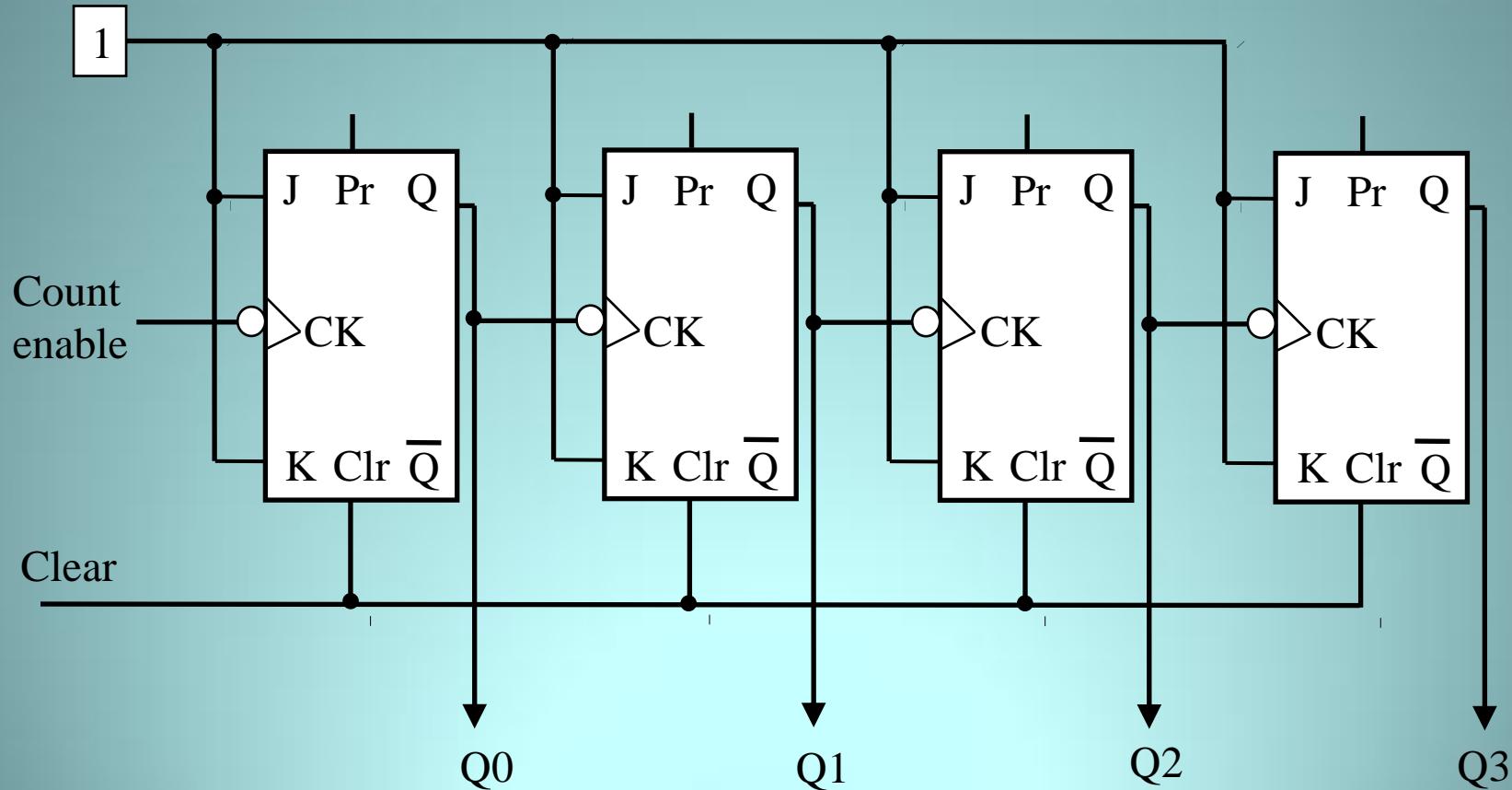
Cette séquence est prise en réponse à des impulsions sur une ligne d'entrée « count enable » : chaque impulsion d'entrée incrémente l'état de un.

# Première partie

Circuits logiques à mémoire

les applications des bascules

**APPLICATION 1 : COMPTEUR asynchrone modulo 16** constitué de 4 bascules JK (M/E)



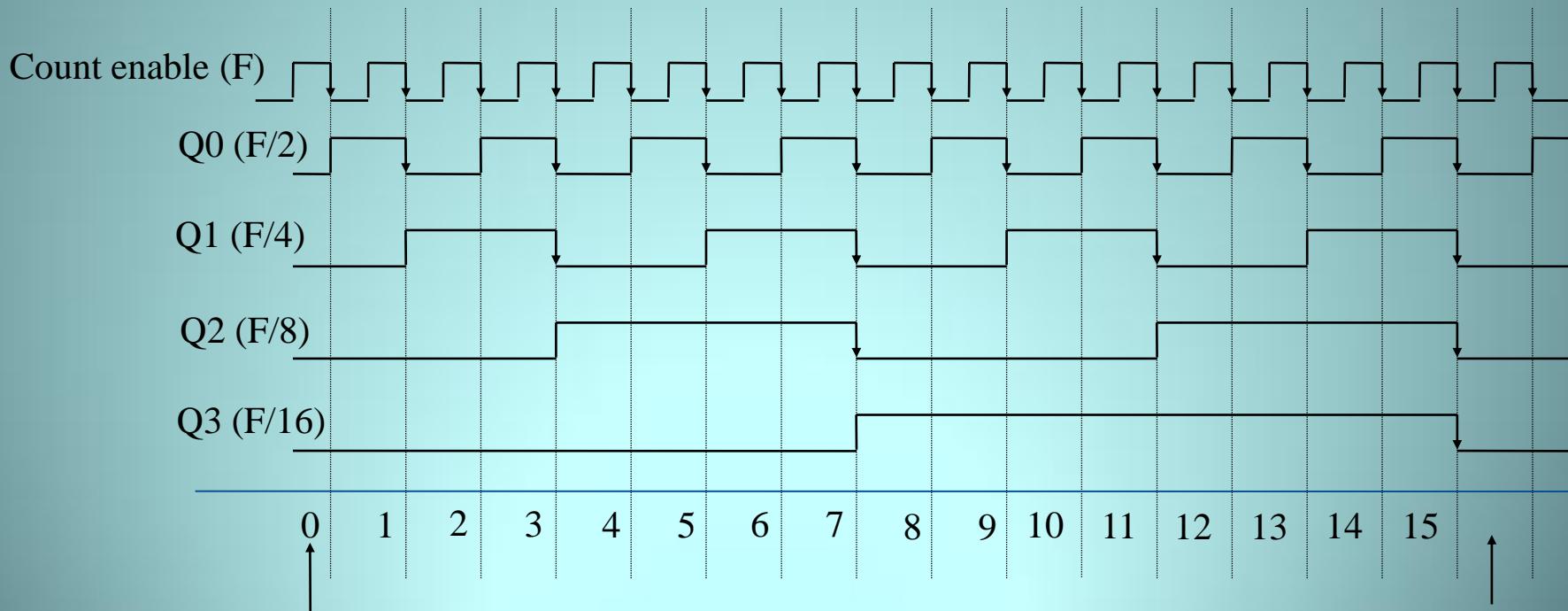
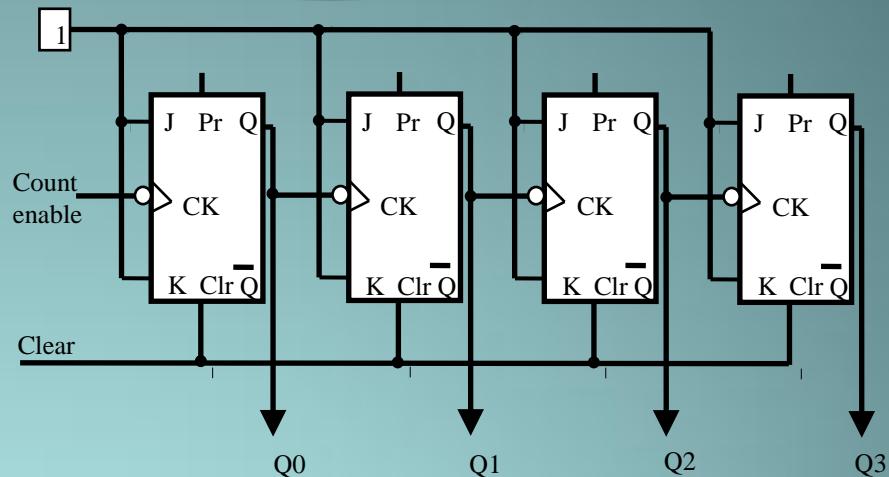
**EXERCICE :** établir un chronogramme de Q<sub>0</sub>Q<sub>1</sub>Q<sub>2</sub>Q<sub>3</sub>, c'est à dire représenter les valeurs des sorties à chaque top d'horloge.

# Première partie

Circuits logiques à mémoire  
les applications des bascules

## APPLICATION 1 : COMPTEUR

*asynchrone modulo 16* constitué de 4 bascules JK (M/E)



CHRONOGRAMME DU COMPTEUR SYNCHRONE modulo 16 (0 à 15)

# Première partie

Circuits logiques à mémoire

les applications des bascules

## APPLICATION 1 : COMPTEUR

*asynchrone modulo 16* constitué de 4 bascules JK (M/E)

## ⇒ APPLICATION : GENERATION DE SIGNAUX TEMPORELS

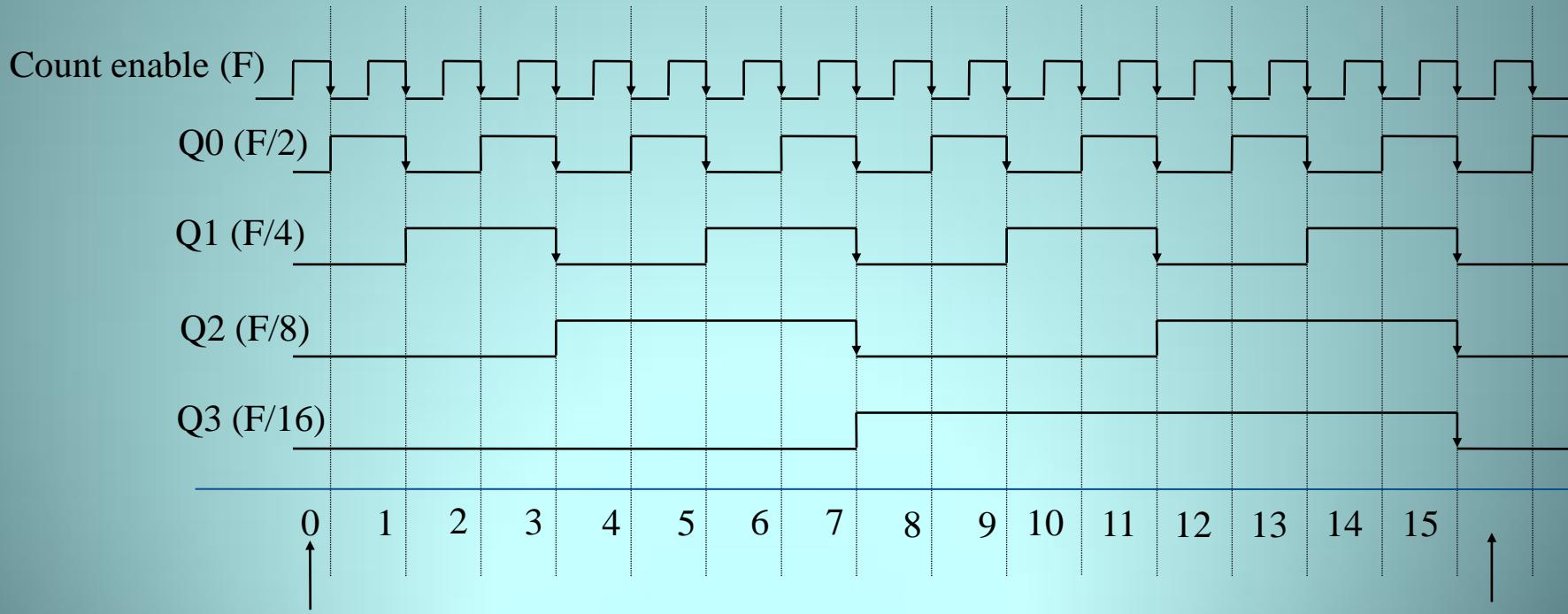
Si CE est reliée à une horloge de fréquence F :

\*sortie Q0 → les impulsions de fréquence  $F/2$

\*sortie Q1 → les impulsions de fréquence  $F/4$

etc...

le compteur se comporte ici comme un diviseur de fréquence



CHRONOGRAMME DU COMPTEUR SYNCHRONE modulo 16 (0 à 15)

# Première partie

*Circuits logiques à mémoire*

*les applications des bascules*

**EXERCICE : Circuits et chronogrammes compteurs synchrones**

**\*modulo 8**

**\*modulo 6**

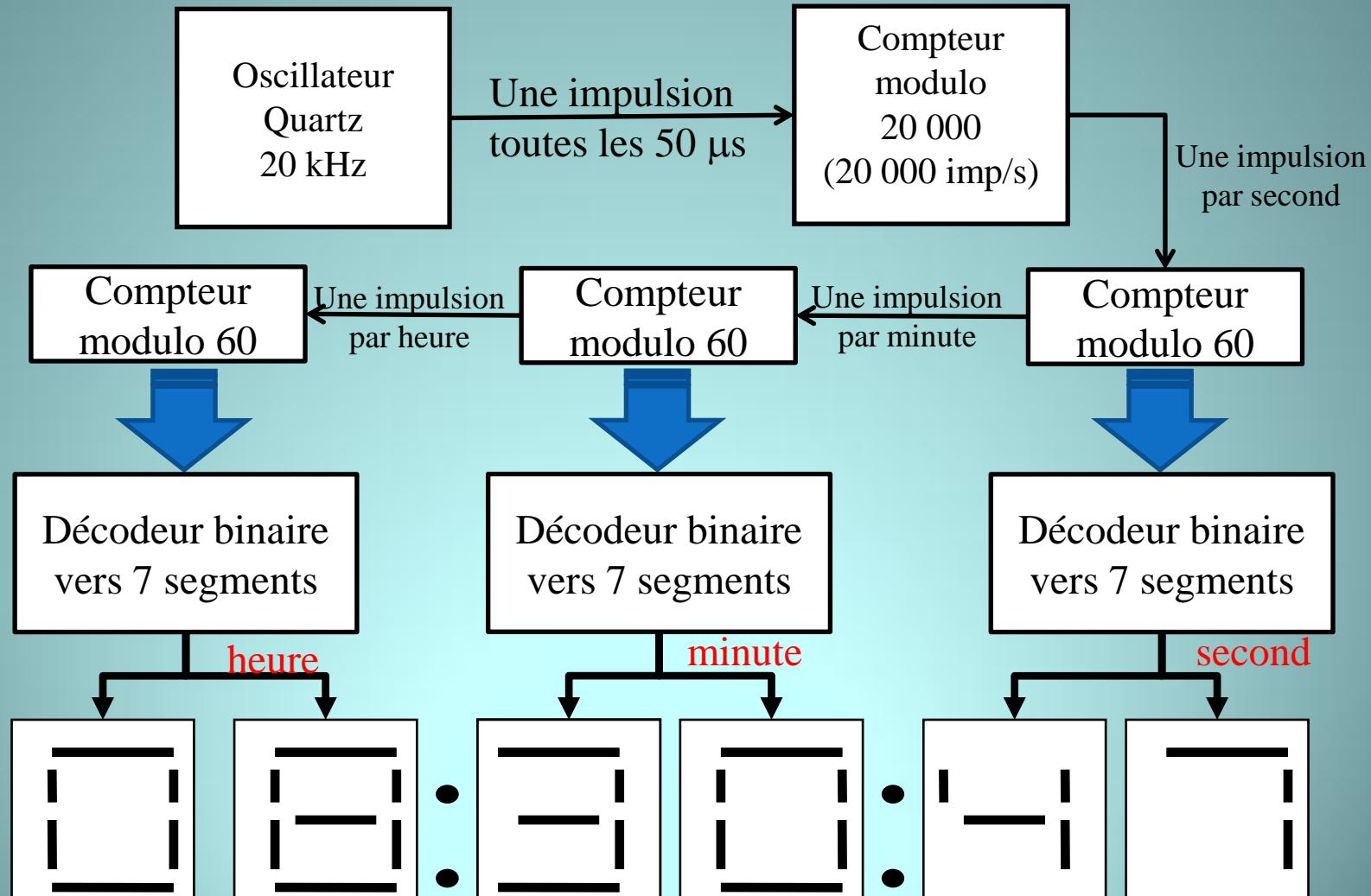
# PRINCIPE DE REALISATION D'UNE MONTRE DIGITALE A QUARTZ

(circuits séquentiels et combinatoires)

## Première partie

Circuits logiques à mémoire

les applications des bascules



# Première partie

## *Circuits logiques à mémoire*

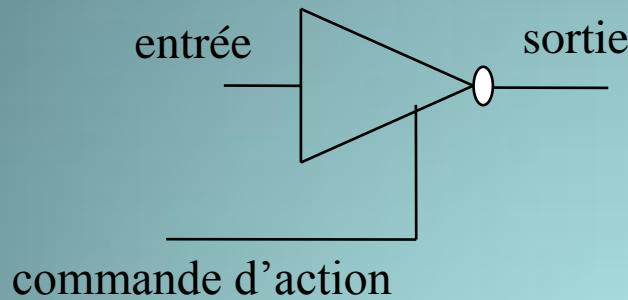
Sujets abordés :

- ❖ les bascules (flip-flops)
- ❖ les bascules (flip-flops) déclenchées sur front d'horloge
- ❖ **les circuits interrupteurs trois états**
- ❖ bus de données
- ❖ Registre
- ❖ organisation des mémoires

# Première partie

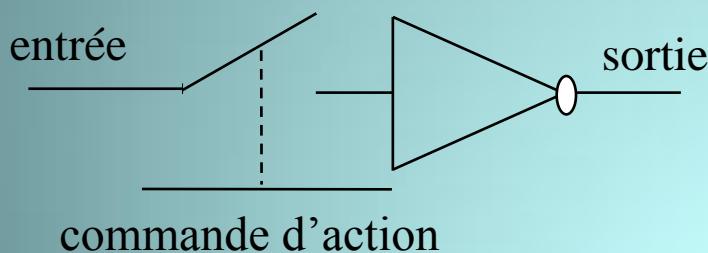
## Circuits logiques à mémoire

### les circuits interrupteurs trois états



la sortie vaut :

- ❖ 0 ou 1 selon l'entrée (commande d'action active, l'interrupteur fermé)
- ❖ état déconnecté ou haute impédance (commande d'action à zéro, l'interrupteur ouvert).



De plus, c'est un circuit AMPLIFICATEUR de puissance

Ce circuit permet une *déconnexion* pendant un temps déterminé afin d'éviter les interférences entre les entrées et les sorties des bascules (internes)

# Première partie

## *Circuits logiques à mémoire*

### Sujets abordés :

- ❖ les bascules (flip-flops)
- ❖ les bascules (flip-flops) déclenchées sur front d'horloge
- ❖ les circuits interrupteurs trois états
- ❖ **bus de données**
- ❖ Registre
- ❖ organisation des mémoires

# Première partie

## *Circuits logiques à mémoire bus de données*

- ❖ **Un bus est une structure d'interconnexion constituée d'un ensemble de lignes (fils) sur lesquel(le)s transitent des informations que s'échangent deux unités qui communiquent.**



- La *transmission en parallèle* : les  $n$  (ici 8) bits d'information à transmettre le sont en une seule fois sur les  $n$  lignes de données → rapide
  - La *transmission en série* : les  $n$  bits d'information sont transmis sur une seule ligne de données, 1 bit à la fois, séquentiellement → moins encombrante
- Dans la plupart des *micro-architectures*, les registres sont connectés entre eux par un bus d'entrée et un bus de sortie d'information.

# Première partie

## Circuits logiques à mémoire bus de données

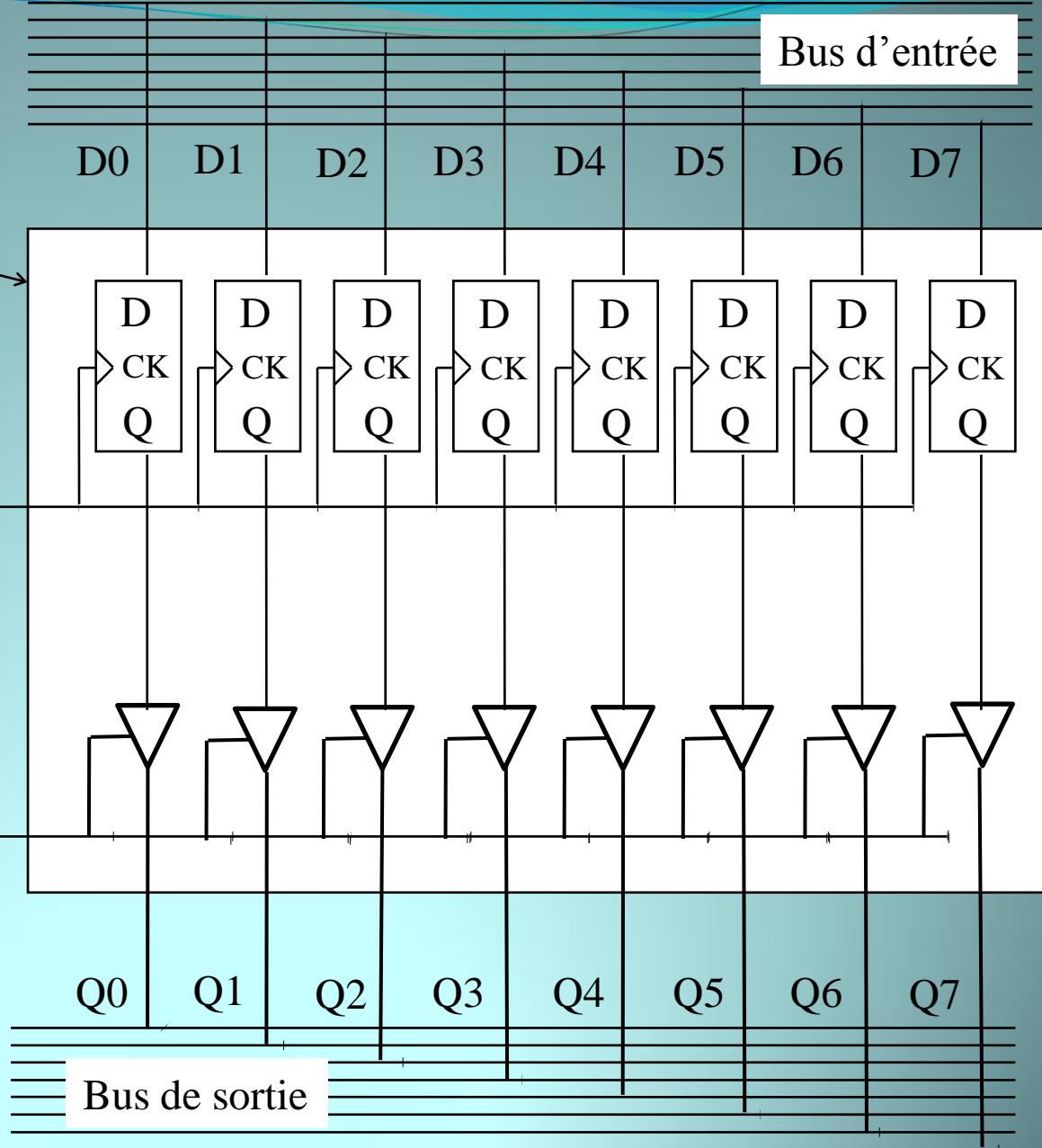
SCHEMA  
LOGIQUE D'UN  
REGISTRE 8 BITS

Registre 8 bits

Deux lignes de commandes

**CK** : bus d'entrée → registre le  
chargement quand  $CK = 1$

**OE** (Output Enable) :  
registre → les lignes du bus de  
sortie (tant que OE est actif)



# Première partie

*Circuits logiques à mémoire  
bus de données*

**EXERCICE : imaginer comment réaliser le chargement d'un registre Ri dans un registre Rj**

# Première partie

*Circuits logiques à mémoire  
bus de données*

**EXERCICE : imaginer comment réaliser le chargement d'un registre Ri dans un registre Rj**

**REPONSE :** sortie Ri sur bus d'entrée Rj / activation OEi et Ckj / en fin de chargement, OEi et CKj sont remis à l'état de repos ...

# Première partie

## *Circuits logiques à mémoire*

Sujets abordés :

- ❖ les bascules (flip-flops)
- ❖ les bascules (flip-flops) déclenchées sur front d'horloge
- ❖ les circuits interrupteurs trois états
- ❖ bus de données
- ❖ Registre
- ❖ **organisation des mémoires**

# Première partie

## Circuits logiques à mémoire

### organisation des mémoires

Schéma  
logique d'une  
mémoire 4x3  
(soit 12 bits)

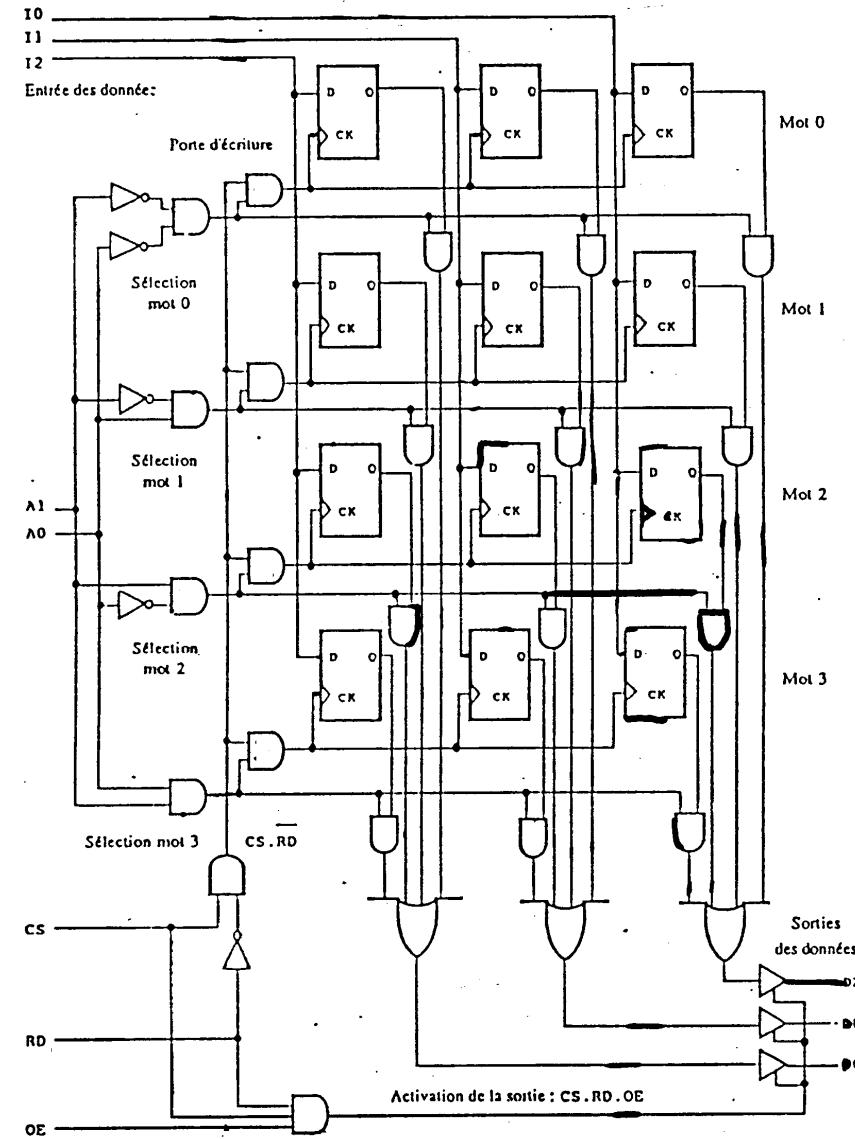


Schéma logique d'une mémoire 4 x 3 (soit 12 bits),

# Première partie

*Circuits logiques à mémoire*

organisation des mémoires

1. Combien de mots ?
2. Combien de bits par mot ?
3. Capacité de cette mémoire ?
4. Nombre d'entrées ; précisez la répartition
5. Nombre de sorties ?
6. Grâce à quelles broches peut-on sélectionner le mot de mémoire sur lequel porte l'opération précisée par RD ? Pourquoi ?
7. Quelle est l'opération réalisée sur le mot de mémoire adressée si la mémoire est sélectionnée (CS=1) avec RD=0 ?
8. Qu'en est-il des autres mots de la mémoire ? Pourquoi ?
9. Décrire l'opération de lecture.

- Combien de mots ?** 4 mots (numéro 0 à 3)
- Combien de bits par mot ?** 3 bits ; toute opération de L/E porte sur 1 mot de 3 bits
- Capacité de cette mémoire ?**  $4 \times 3 = 12$  bits
- Nombre d'entrées ; précisez la répartition** 8 entrées, dont :
  - 3 entrées de données ( $I_0, I_1, I_2$ ) : bus d'entrée
  - 2 adresses ( $A_0$  et  $A_1$ ) : bus d'adresse
  - 3 commandes :
    - $CS$  (Chip Select) pour la sélection du boîtier ( $CS=1$ )
    - $RD$  : lecture ( $RD=1$ ) et écriture ( $RD=0$ )
    - $OE$  (Output Enable) pour activer les lignes de sorties ( $OE=1$ )
- Nombre de sorties ?** 3 sorties :  $D_0, D_1, D_2$  lignes de données (à droite)
- Grâce à quelles broches peut-on sélectionner le mot de mémoire sur lequel porte l'opération précisée par RD ? Pourquoi ?**

$A_0$  et  $A_1$  sont des lignes d'adresses entrées d'un décodeur 2 vers 4 qui permet la sélection du mot :

$A_0 = A_1 = 0 \Rightarrow$  porte d'écriture 0

...

$A_0 = A_1 = 1 \Rightarrow$  porte d'écriture 3

- Quelle est l'opération réalisée sur le mot de mémoire adressée si la mémoire est sélectionnée ( $CS=1$ ) avec  $RD=0$  ?**

C'est une écriture et la liaison verticale interne  $CS.\overline{RD}=1$ . La sortie des portes d'écriture alimente les entrées CK des bascules internes; ainsi le mot adressé est chargé avec les bits présents sur les lignes du bus d'entrée.

- Qu'en est-il des autres mots de la mémoire ? Pourquoi ?**

Ils sont inchangés, car non sélectionnés par les portes d'écriture (décodeur)

## 9. Décrire l'opération de lecture.

\*décodage d'adresse du mot à lire

\*ligne  $CS.\overline{RD}$  inactive ( $RD=1$  et  $CS=1$ )  $\Rightarrow$  portes écriture inactives

\*donc aucune bascule n'est modifiée

\*la sortie du décodeur d'adresse valide les portes ET associées aux sorties Q des bascules constituant le mot mémoire à lire ; seuls les bits de celui-ci sont transmis vers les portes OU de sortie en bas du schéma

\*ainsi, le contenu du mot mémoire à lire apparaît sur les portes OU

# Première partie

## *Circuits logiques à mémoire*

### organisation des mémoires

#### REMARQUE :

\*Pour une **lecture**, les lignes I0, I1 et I2 ne sont pas utilisées car le mot est positionné sur les lignes D0, D1 et D2.

\*Pour une **écriture**, les sorties sont inutilisées car les bits présents sur les lignes d'entrée sont chargés dans le mot de mémoire adressée.

\*Pour éviter des interférences sur le bus de sortie, les portes OU de sortie ne sont pas directement reliées aux lignes de sorties des données : on utilise des **circuits interrupteurs à trois états** comme dans le cas du registre 8 bits.

*La sortie est activée seulement lorsque RD.CS.OE=1*

*Dès que l'un des signaux CS, RD, OE est à zéro, les lignes de sorties sont dans un état de haute impédance et sont alors isolées du reste du système.*

# Première partie

*Circuits logiques à mémoire*

*organisation des mémoires*

EXERCICE 1 : comment écrire 110 dans le mot d'adresse 1 ?

EXERCICE 2 : comment lire le contenu du mot d'adresse 3 ?

# Programme

- Première partie
  - algèbre de Boole
  - circuits logiques
- Deuxième partie
  - L'Unité Centrale du microprocesseur
  - La Mémoire Centrale du microprocesseur
  - Les Périphériques et interfaces associés au microprocesseur
  - Le logiciel de base