

# Signaux et systèmes

## *Introduction à la conception de circuits numériques*

Vianney Lapotre

Maître de conférences

# Objectifs

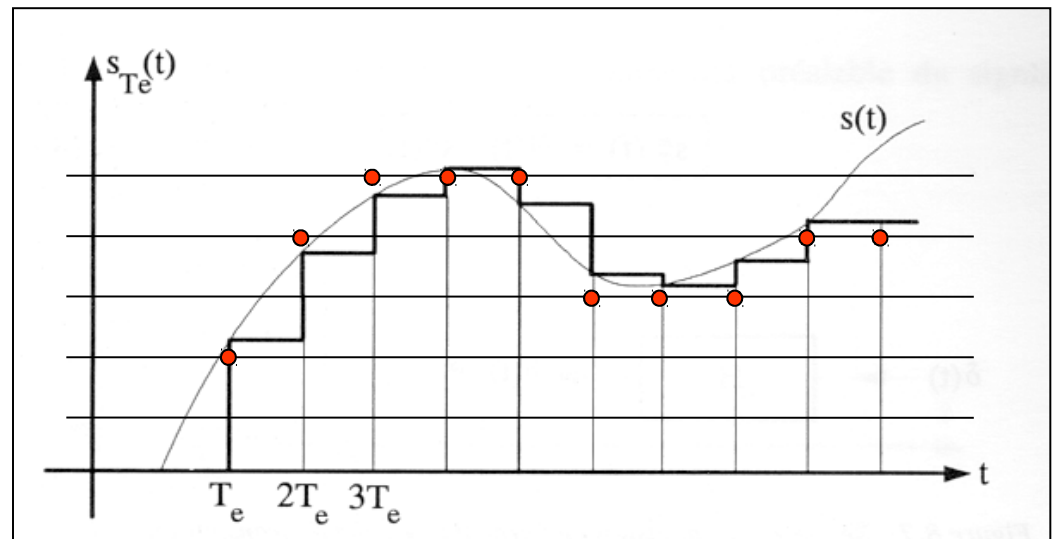
- Rappels
  - Numérisation de l'information
  - Nombres binaires et Hexadécimaux
  - Algèbre de Boole
- Les composants numériques simples
  - Portes et circuits logiques
  - Multiplexeurs et démultiplexeurs
  - Bascules
  - Registres
- Les éléments d'un processeur simple
- Le projet
- Évaluation

# Rappels

## *Numérisation de l'information*

# Échantillonnage

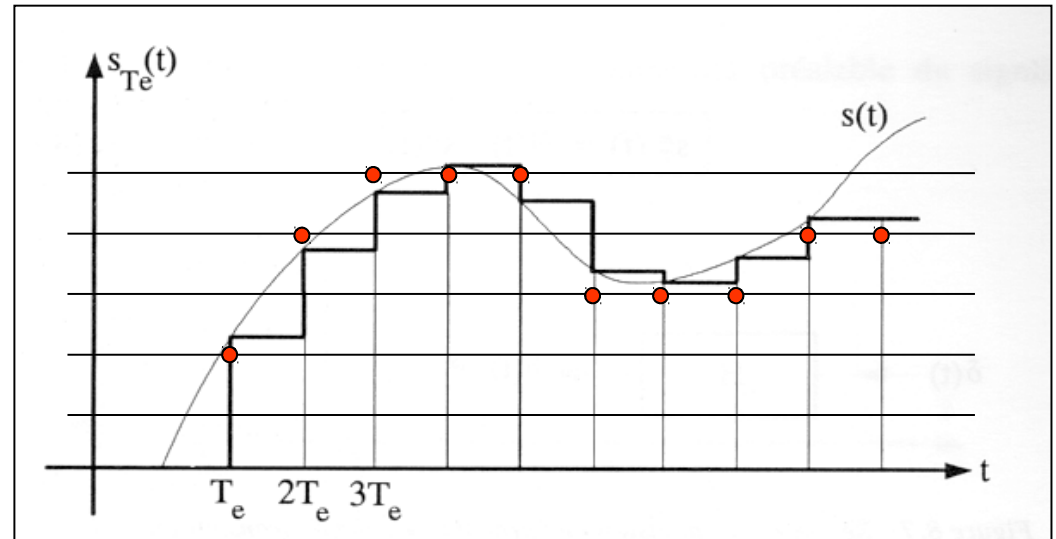
- Cette étape consiste à hacher le signal analogique en petites tranches temporelles selon une période bien définie par une horloge.
- L'amplitude ( $A$ ) du signal au moment du top de l'horloge est prise comme référence pour cette tranche. C'est cette valeur qui sera codée.



# Quantification

- Supposons que le signal soit compris entre 3 et -4 volts. Les valeurs correspondantes sur 3 bits sont :

- 011      3V
- 010      2V
- 001      1V
- 000      0V
- 111      -1V
- 110      -2V
- 101      -3V
- 100      -4V



- les valeurs de l'Echantillonneur sont remplacées par un codage binaire après arrondi.

# Rappels

## *Représentation des nombres binaires*

# Nombres positifs (1)

- Coder un nombre avec seulement deux symboles, généralement les deux chiffres 0 et 1
  - On note  $(11001)_2$  afin de ne pas confondre avec 11001 (onze mille un)

Binaire	Décimal
0	$0 = 0 \times 2^0$
1	$1 = 1 \times 2^0$
10	$2 = 1 \times 2^1 + 0 \times 2^0$
11	$3 = 1 \times 2^1 + 1 \times 2^0$
100	$4 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$
101	$5 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
110	$6 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
111	$7 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

## Nombres positifs (2)

- Passage de la notation décimale à la notation binaire
  - $(23)_{10} = (010111)_2 = (10111)_2$

$(23)_{10}$	32 ( $2^5$ )	16 ( $2^4$ )	8 ( $2^3$ )	4 ( $2^2$ )	2 ( $2^1$ )	1 ( $2^0$ )
= 0x32	0					
+ 1x16		1				
+ 0x8			0			
+ 1x4				1		
+ 1x2					1	
+ 1x1						1



# Addition

- 1 bit + 1 bit
  - $0 + 0 = 0$
  - $0 + 1 = 1$
  - $1 + 0 = 1$
  - $1 + 1 = 10$

$$5 = 1 \times 4 + 0 \times 2 + 1 \Rightarrow 1 \ 0 \ 1$$

$$7 = 1 \times 4 + 1 \times 2 + 1 \Rightarrow 1 \ 1 \ 1$$

$$\text{Somme bit à bit:} \quad 1 \ 1 \ 0 \ 0$$

Vérification du résultat:

$$1 \times 8 + 1 \times 4 + 0 \times 2 + 0 = 12$$

qui est bien égal à  $5 + 7$ .

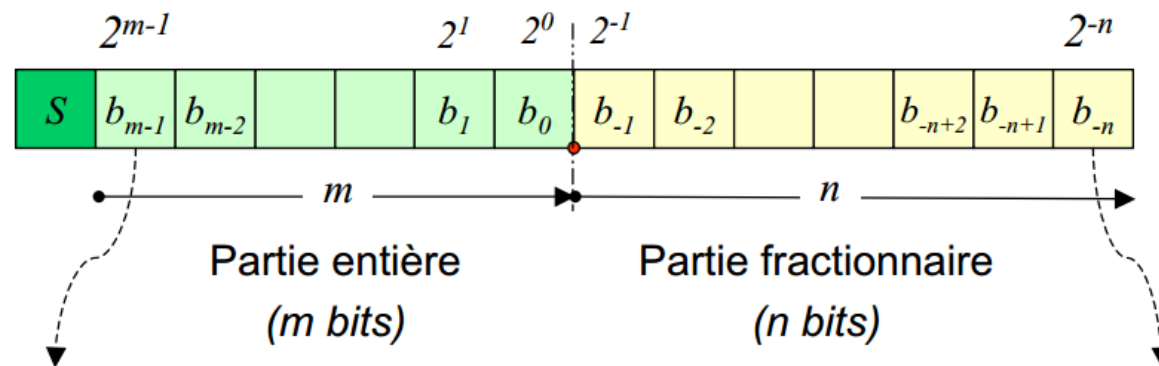
# Nombres négatifs

- Les nombres positifs sont codés normalement tandis que les nombres négatifs sont codés selon leur complément à deux.
  - Du positif au négatif: inverser et ajouter 1.
  - Du négatif au positif: inverser et ajouter 1.

Poids en binaire =>		32	16	8	4	2	1
Valeur en binaire	21	0	1	0	1	0	1
Inverser les bits		1	0	1	0	1	0
Ajouter 1	<b>-21</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>
Inverser les bits		0	1	0	1	0	0
Ajouter 1	21	0	1	0	1	0	1

# Nombres à virgule fixe

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i 2^i$$



Domaine de définition  
du codage :

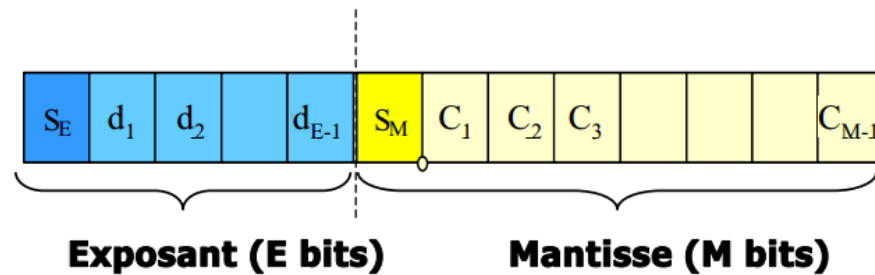
$$D = [-2^m, 2^m - 2^{-n}]$$

Précision du codage  
Pas de quantification :

$$q = 2^{-n}$$

# Nombres à virgule flottante

- L'exposant associé à la donnée est codée au sein de celle-ci
- Les données sont composées de deux parties
  - Exposant
  - Mantisse



- Le codage dépend de la norme utilisée

# Multiples (1)

- 1 tétrade ou Nibble: quatre bits
- 1 octet: 8 bits
- 1 ko (kilo-octets): 1 000 octets
- 1 Mo (Méga-octets): 1 million d'octets
- 1 Go (Giga-octets): 1 milliard d'octets.

# Multiples (2)

## Multiples normalisés [\[ modifier | modifier le code \]](#)

La normalisation des [préfixes binaires](#) de 1998 par la [Commission électrotechnique internationale](#) spécifie les préfixes suivants pour représenter les puissances de 2 :

- kibi pour « **kilo binaire** » ;
- mébi pour « **méga binaire** » ;
- gibi pour « **giga binaire** » ;
- tébi pour « **téra binaire** » ;

et ainsi de suite.

Concernant les multiples de l'octet, cela donne<sup>5</sup> :

- 1 **kibioctet** (Kio) =  $2^{10}$  octets = 1 024 octets
- 1 **mébioctet** (Mio) =  $2^{20}$  octets = 1 024 Kio = 1 048 576 octets
- 1 **gibioctet** (Gio) =  $2^{30}$  octets = 1 024 Mio = 1 073 741 824 octets
- 1 **tébioctet** (Tio) =  $2^{40}$  octets = 1 024 Gio = 1 099 511 627 776 octets
- 1 **pébioctet** (Pio) =  $2^{50}$  octets = 1 024 Tio = 1 125 899 906 842 624 octets
- 1 **exbioctet** (Eio) =  $2^{60}$  octets = 1 024 Pio = 1 152 921 504 606 846 976 octets
- 1 **zébioctet** (Zio) =  $2^{70}$  octets = 1 024 Eio = 1 180 591 620 717 411 303 424 octets
- 1 **yobioctet** (Yio) =  $2^{80}$  octets = 1 024 Zio = 1 208 925 819 614 629 174 706 176 octets

Les préfixes kilo, méga, giga, téra, etc. correspondent aux mêmes multiplicateurs que dans tous les autres domaines : des puissances de 10. Appliqué à l'informatique, cela donne :

- 1 **kiloctet** (ko) =  $10^3$  octets = 1 000 octets
- 1 **mégaoctet** (Mo) =  $10^6$  octets = 1 000 ko = 1 000 000 octets
- 1 **gigaoctet** (Go) =  $10^9$  octets = 1 000 Mo = 1 000 000 000 octets
- 1 **téraoctet** (To) =  $10^{12}$  octets = 1 000 Go = 1 000 000 000 000 octets
- 1 **pétaoctet** (Po) =  $10^{15}$  octets = 1 000 To = 1 000 000 000 000 000 octets
- 1 **exaoctet** (Eo) =  $10^{18}$  octets = 1 000 Po = 1 000 000 000 000 000 000 octets
- 1 **zettaoctet** (Zo) =  $10^{21}$  octets = 1 000 Eo = 1 000 000 000 000 000 000 000 octets
- 1 **yottaoctet** (Yo) =  $10^{24}$  octets = 1 000 Zo = 1 000 000 000 000 000 000 000 000 octets

Source : <https://fr.wikipedia.org/wiki/Octet>

# Rappels

## *Représentation des nombres hexadécimaux*

# Nombres Hexadécimaux positifs (1)

- Coder un nombre avec 16 symboles

Hexa	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- Conversion Hexadécimal / Décimal

Hexadécimal	Décimal
8	$8 = 8 \times 16^0$
C	$12 = 12 \times 16^0$
AC	$172 = 10 \times 16^1 + 12 \times 16^0$
A5C	$2652 = 10 \times 16^2 + 5 \times 16^1 + 12 \times 16^0$



# Nombres Hexadécimaux positifs (2)

## ■ Conversion Hexadécimal / Binaire

Décimal	Binaire				Hexadécimal
	8 ( $2^3$ )	4 ( $2^2$ )	2 ( $2^1$ )	1 ( $2^0$ )	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	A
11	1	0	1	1	B
12	1	1	0	0	C
13	1	1	0	1	D
14	1	1	1	0	E
15	1	1	1	1	F

# Rappels

## *Algèbre de Boole*

# Qu'es aquò ?

- George Boole (1815-1864) était un mathématicien autodidacte anglais
  - Il développa une algèbre permettant de manipuler les propositions logiques au moyen d'équations mathématiques où les énoncés VRAI et FAUX sont représentés par les valeurs 1 et 0, tandis que les opérateurs ET et OU deviennent des opérateurs algébriques de multiplication et d'addition

# Notions théoriques (1)

- Un algèbre de Boole c'est :
  - Un ensemble noté  $E$
  - Deux éléments particuliers de  $E$  :  $0$  et  $1$ 
    - ▶ Correspondant respectivement à FAUX et VRAI
  - Deux opérations binaire sur  $E$  :  $+$  et  $\bullet$ 
    - ▶ Correspondant respectivement au OU et ET logiques
  - Une opération unaire sur  $E$  :  $\bar{\phantom{x}}$ 
    - ▶ Correspondant à la négation logique

## Notions théoriques (2)

■ On acceptera les postulats suivant :

- $0 \bullet 0 = 0$
- $0 \bullet 1 = 1 \bullet 0 = 0$
- $1 \bullet 1 = 1$
- $1 + 1 = 1$
- $1 + 0 = 0 + 1 = 1$
- $0 + 0 = 0$
- $\overline{0} = 1$
- $\overline{1} = 0$

## Notions théoriques (3)

- Des postulats précédant découlent les axiomes suivants
  - Soient  $a$ ,  $b$  et  $c$  des éléments de  $E$

Commutativité	$a+b=b+a$	$a \cdot b=b \cdot a$
Associativité	$(a+b)+c=a+(b+c)$	$(a \cdot b) \cdot c=a \cdot (b \cdot c)$
Distributivité	$a \cdot (b+c)=a \cdot b+a \cdot c$	$a+(b \cdot c)=(a+b) \cdot (a+c)$
Élément neutre	$a+0=a$	$a \cdot 1=a$
Complémentation	$a+\bar{a}=1$	$a \cdot \bar{a}=0$

## Notions théoriques (4)

- Quelques théorèmes de base mais indispensables

	Nom	Forme 1	Forme 2
1	Involution	$\overline{\overline{a}} = a$	-
2	Idempotence	$a+a=a$	$a \cdot a=a$
3	Élément absorbant	$a+1=1$	$a \cdot 0=0$
4	Absorption	$a+a \cdot b=a$	$a \cdot (a+b)=a$
5	De Morgan	$\overline{a+b}=\overline{a} \cdot \overline{b}$	$\overline{a \cdot b}=\overline{a} + \overline{b}$
6	-	$a+(\overline{a} \cdot b)=a+b$	$a \cdot (\overline{a} + b)=a \cdot b$

# Table de vérité (1)

- La table de vérité d'une fonction logique est un tableau énumérant les valeurs logiques d'une fonction pour les différentes combinaisons des valeurs de ses variables indépendantes
  - En circuits logiques, on parlera de correspondance entre la sortie et les entrées



## Table de vérité (2)

- Quelques exemples

$S=A+B$		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

$S=\overline{A \cdot B}$		
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

$S=\overline{A+B}$		
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

$S=\overline{A}$	
A	S
0	1
1	0

## Table de vérité (3)

- À vous de jouer !

$S=A \cdot B + C + \overline{D}$				
A	B	C	D	S
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

## Table de vérité (3)

- À vous de jouer !



$S=A \cdot B + C + \overline{D}$				
A	B	C	D	S
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

# Les composants numériques simples

## *Portes et circuits logiques*



# Portes logiques élémentaires (1)

- Chaque porte correspond à une fonction logique précise

Opérateur logique	Nom Français	Nom Anglais	Symbole (USA)	Table de vérité															
$A \bullet B$	ET	AND		<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
$A + B$	OU	OR		<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	



## Portes logiques élémentaires (2)

- Chaque porte correspond à une fonction logique précise

Opérateur logique	Nom Français	Nom Anglais	Symbole (USA)	Table de vérité															
$\overline{A \bullet B}$	NON ET	NAND		<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
$\overline{A + B}$	NON OU	NOR		<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	


# Portes logiques élémentaires (3)

- Chaque porte correspond à une fonction logique précise

Opérateur logique	Nom Français	Nom Anglais	Symbole (USA)	Table de vérité															
$A \oplus B = \overline{A \cdot B} + A \cdot \overline{B}$	OU exclusif	XOR		<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
$\overline{A \oplus B} = \overline{\overline{A \cdot B} + A \cdot \overline{B}}$	NON OU exclusif	NXOR		<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	1
A	B	F																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

## Portes logiques élémentaires (4)

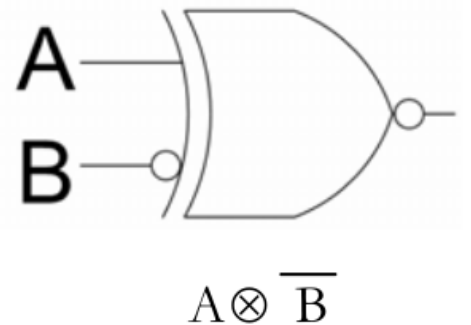
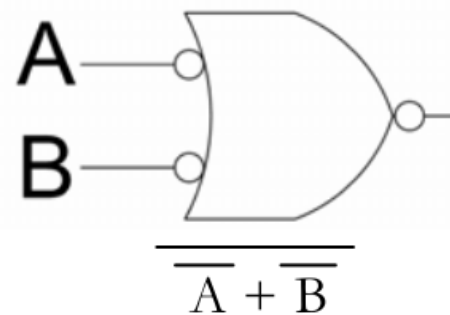
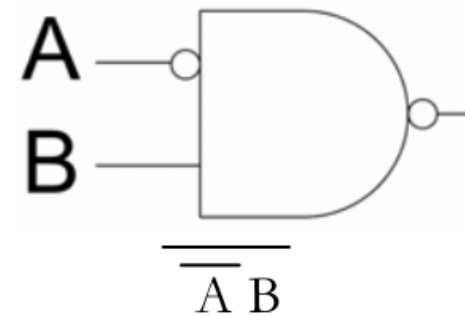
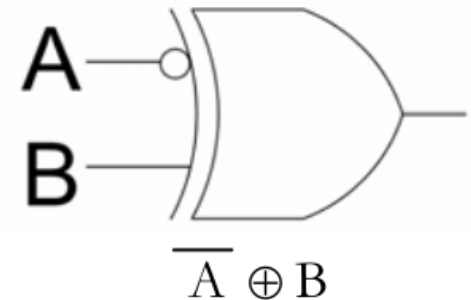
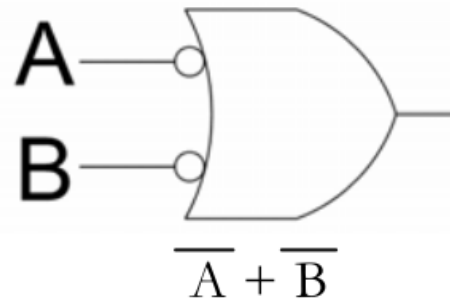
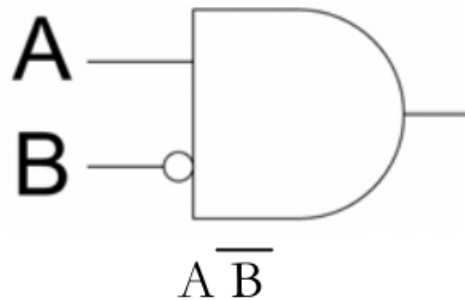
- Chaque porte correspond à une fonction logique précise

Opérateur logique	Nom Français	Nom Anglais	Symbole (USA)	Table de vérité						
$\overline{A}$	NON	NOT		<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0
A	F									
0	1									
1	0									



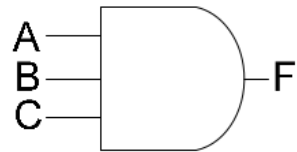
# Portes logiques élémentaires (5)

- Inversion des entrées

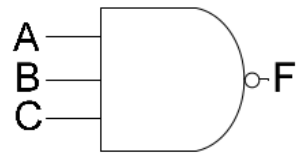
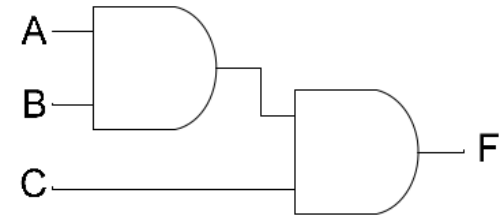


# Portes logiques élémentaires (6)

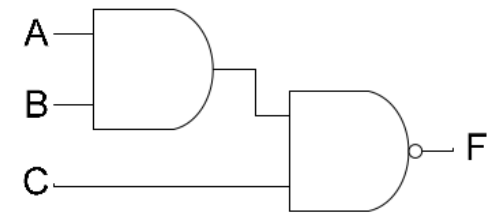
- Entrées multiples



≡



≡

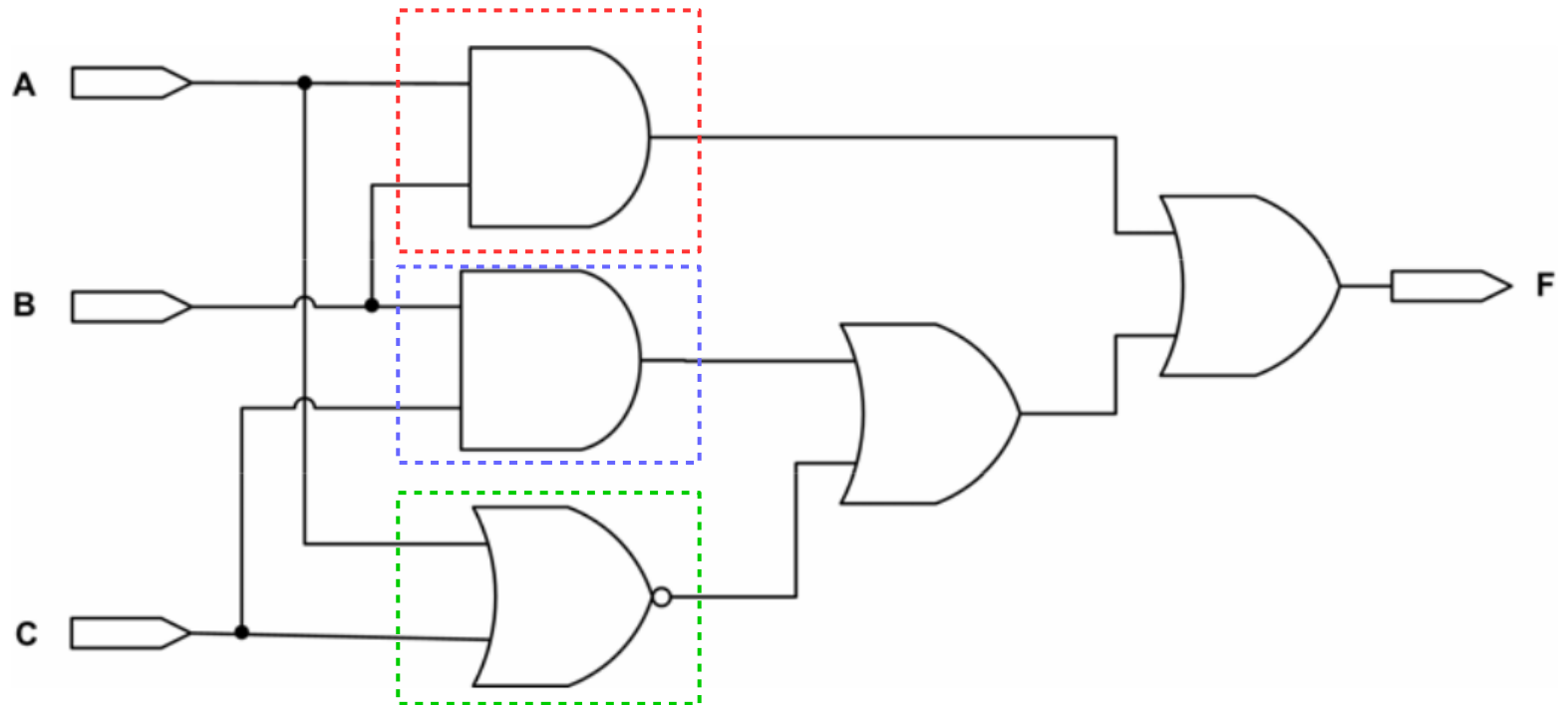


# Notion de synthèse de circuits logiques

- Soit  $F = (A \cdot B) + (B \cdot C) + \overline{(C + A)}$ 
  - Le circuit logique correspondant est ?

# Notion de synthèse de circuits logiques

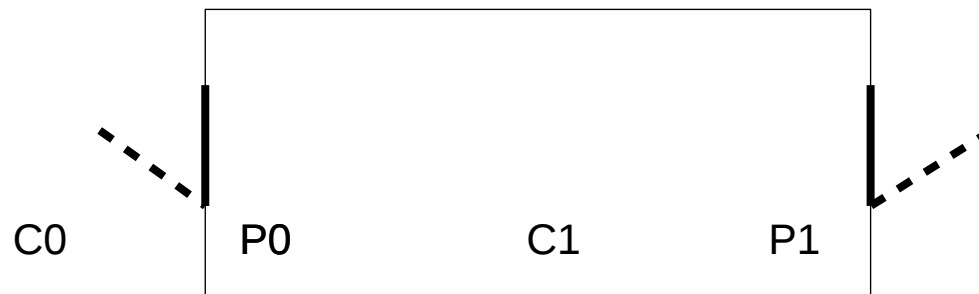
- Soit  $F = (A \cdot B) + (B \cdot C) + (C + A)$ 
  - Le circuit logique correspondant est :



# De la table de vérité à la fonction logique (1)

## ■ Scénario

- Sas avec ouverture de porte automatique (simplifié)
  - ▶ C0 : capteur d'ouverture de la porte P0
  - ▶ C1 : capteur d'ouverture de la porte P1



C0	C1	P0	P1
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	1

↔ Fonction logique ?

# De la table de vérité à la fonction logique (2)

- La forme normal disjonctive
  - Écrire l'équation sous la forme d'une somme de produits.
    - Pour cela, il faut repérer dans la table de vérité toutes les combinaisons pour lesquelles la(les) sortie(s) vaut(valent) 1.

C0	C1	P0	P1
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	1

$$P0 = C0 \cdot \overline{C1}$$

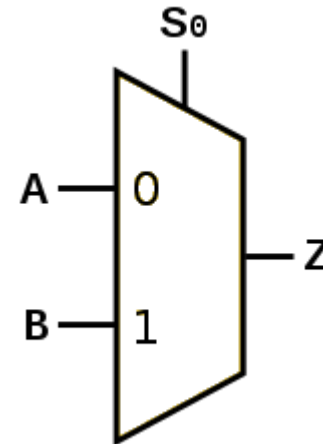
$$P1 = (\overline{C0} \cdot C1) + (C0 \cdot C1)$$

# Les composants numériques simples (De)multiplexeur, bascule et registre

# Multiplexeur (1)

- La sortie de ce circuit correspond à la sélection d'une entrée parmi N
  - La sélection est commandée par le signal S
  - Les signaux A,B et Z sont codés sur un nombre de bit identique

$S_0$	Z
0	A
1	B





## Multiplexeur (2)

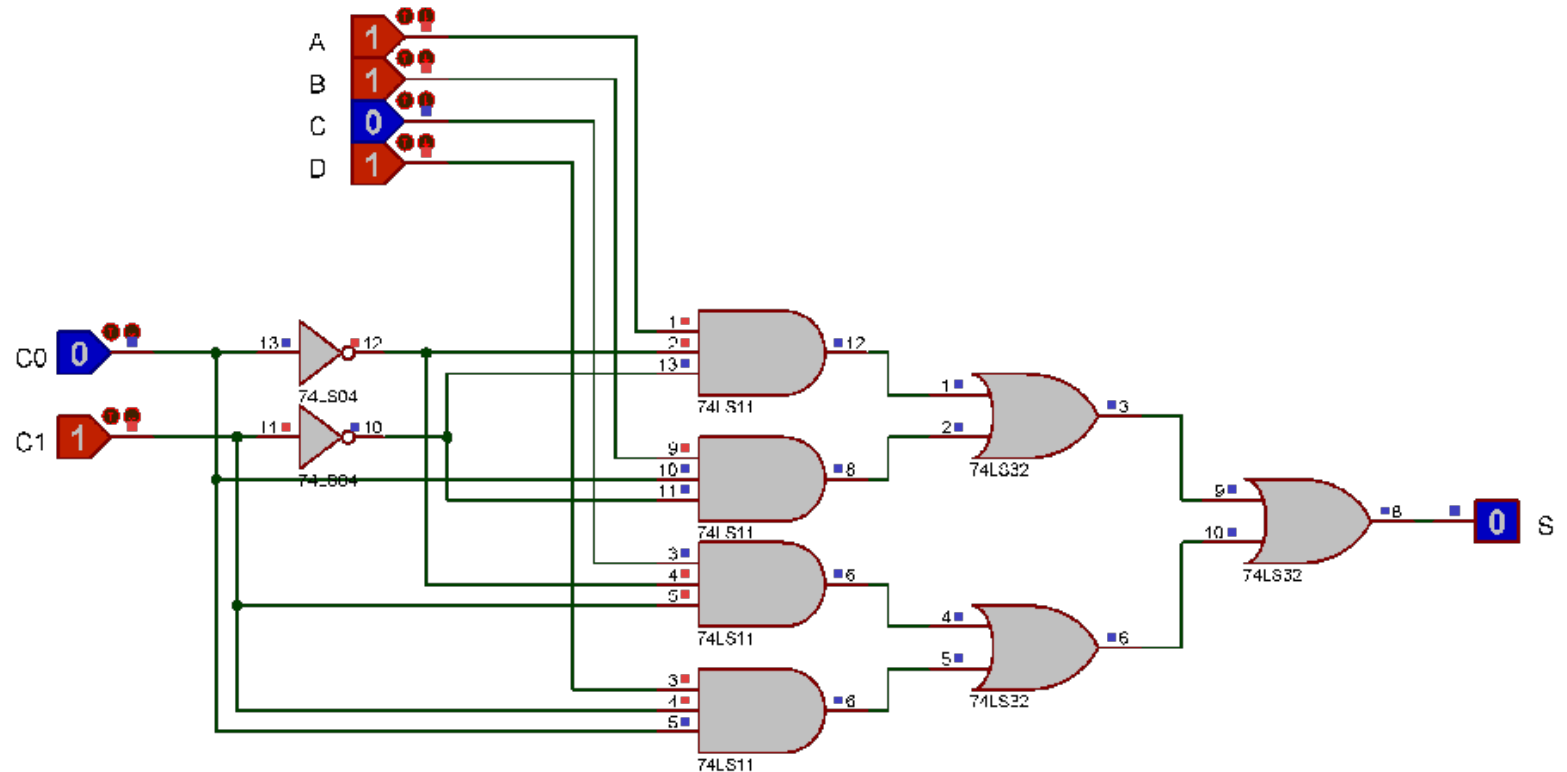
- Table de vérité pour 4 entrées
  - S codé sur 2 bits  $\Rightarrow 2^2 = 4$

S		Z
0	0	A
0	1	B
1	0	C
1	1	D

- Taille (en bits) du signal S pour 8, 12 et 32 entrées ?

# Multiplexeur (3)

- Les portes logiques au cœur de tout !

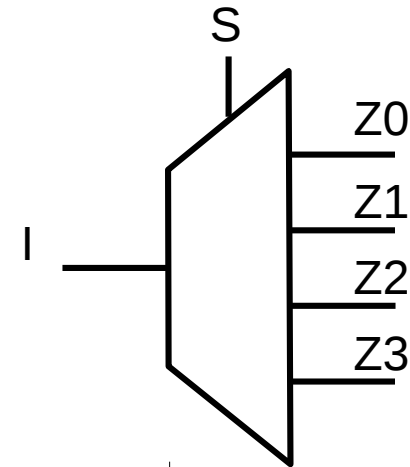


Source: <https://fr.wikipedia.org/wiki/Multiplexeur#/media/File:4to1mux.png>

# Démultiplexeur

- Ce circuit permet de redistribuer sur plusieurs sorties les informations en provenance d'une entrée unique

S		Z0	Z1	Z2	Z3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



# Bascules (1)

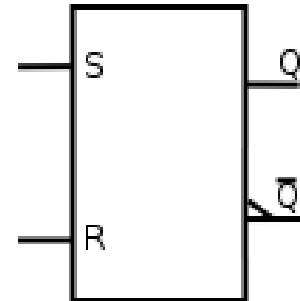
- Une bascule est un circuit logique dont l'état de la sortie ne dépend pas toujours uniquement des entrées du circuit
- En effet, l'état de la sortie est maintenu même après la disparition du signal de contrôle
  - L'état précédent intervient donc dans le fonctionnement du circuit
  - On parle alors de logique séquentielle

## Bascules (2)

- La bascule est l'élément de base de la logique séquentielle. Elle permet de construire :
  - Compteurs, registres, registres à décalage, mémoires...
- Il existe 2 familles de bascules
  - Asynchrone
    - ▶ La sortie peut changer d'état à tout moment dès qu'une entrée varie
  - Synchrone
    - ▶ La sortie ne peut changer d'état uniquement au rythme d'une horloge
      - Changement sur front ou sur niveau

# Bascule Asynchrone RS (1)

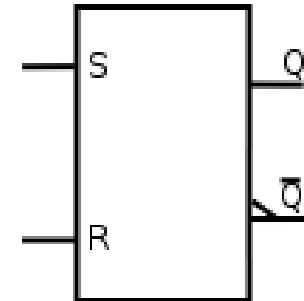
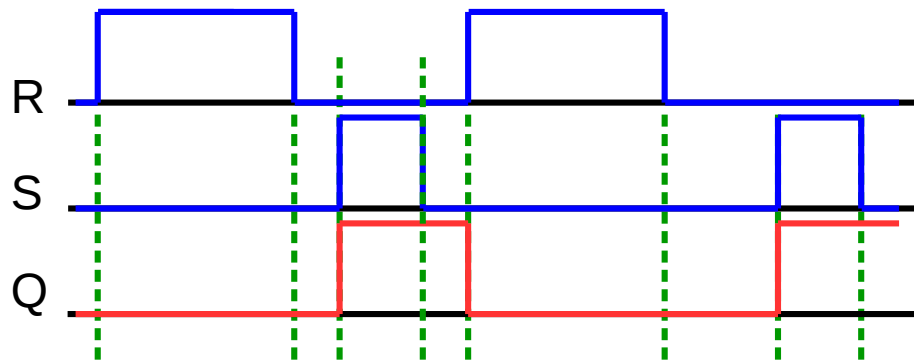
- 2 entrées de contrôle
  - **Set** : mise à 1
  - **Reset** : mise à 0
- 2 sorties
  - **Q** : sortie non-inversée
  - **$\overline{Q}$**  : sortie inversée
- Table de vérité



S	R	Q	$\overline{Q}$	Remarque
0	0	Q	$\overline{Q}$	mémoire
0	1	0	1	mise à 0
1	0	1	0	mise à 1
1	1	0	0	interdit

# Bascule Asynchrone RS (2)

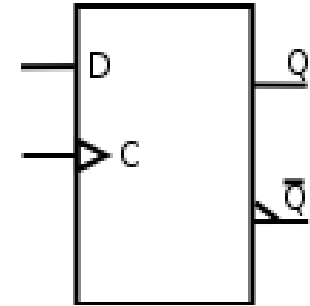
## ■ Chronogramme



S	R	Q	$\bar{Q}$	Remarque
0	0	Q	$\bar{Q}$	mémoire
0	1	0	1	mise à 0
1	0	1	0	mise à 1
1	1	0	0	interdit

# Bascule synchrone D (1)

- 2 entrées de contrôle
  - **D** : entrée de données
  - **C** : Horloge
- 2 sorties
  - $Q_n$  : sortie non-inversée
  - $\overline{Q}_n$  : sortie inversée
- Table de vérité

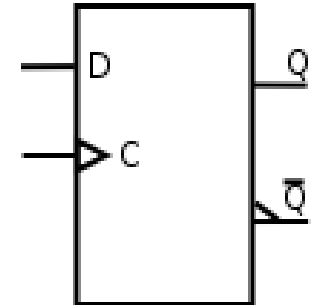
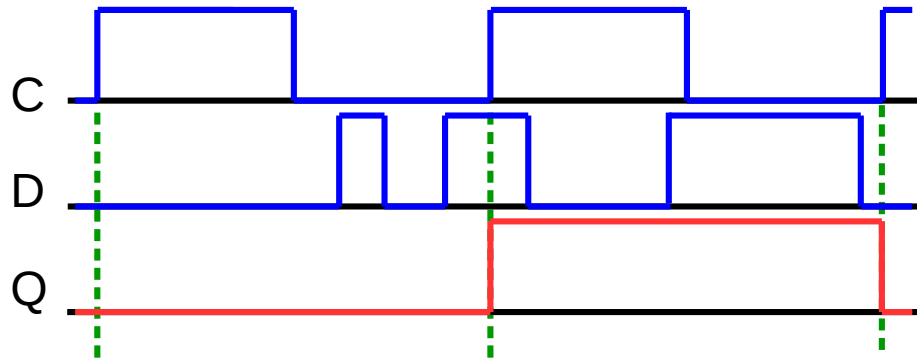


D	C	$Q_n$	$\overline{Q}_n$	Remarque
0	↑	0	1	Q recopie D
1	↑	1	0	Q recopie D
X	0	$Q_{n-1}$	$\overline{Q}_{n-1}$	mémoire
X	1	$Q_{n-1}$	$\overline{Q}_{n-1}$	mémoire



# Bascule synchrone D (2)

## ■ Chronogramme



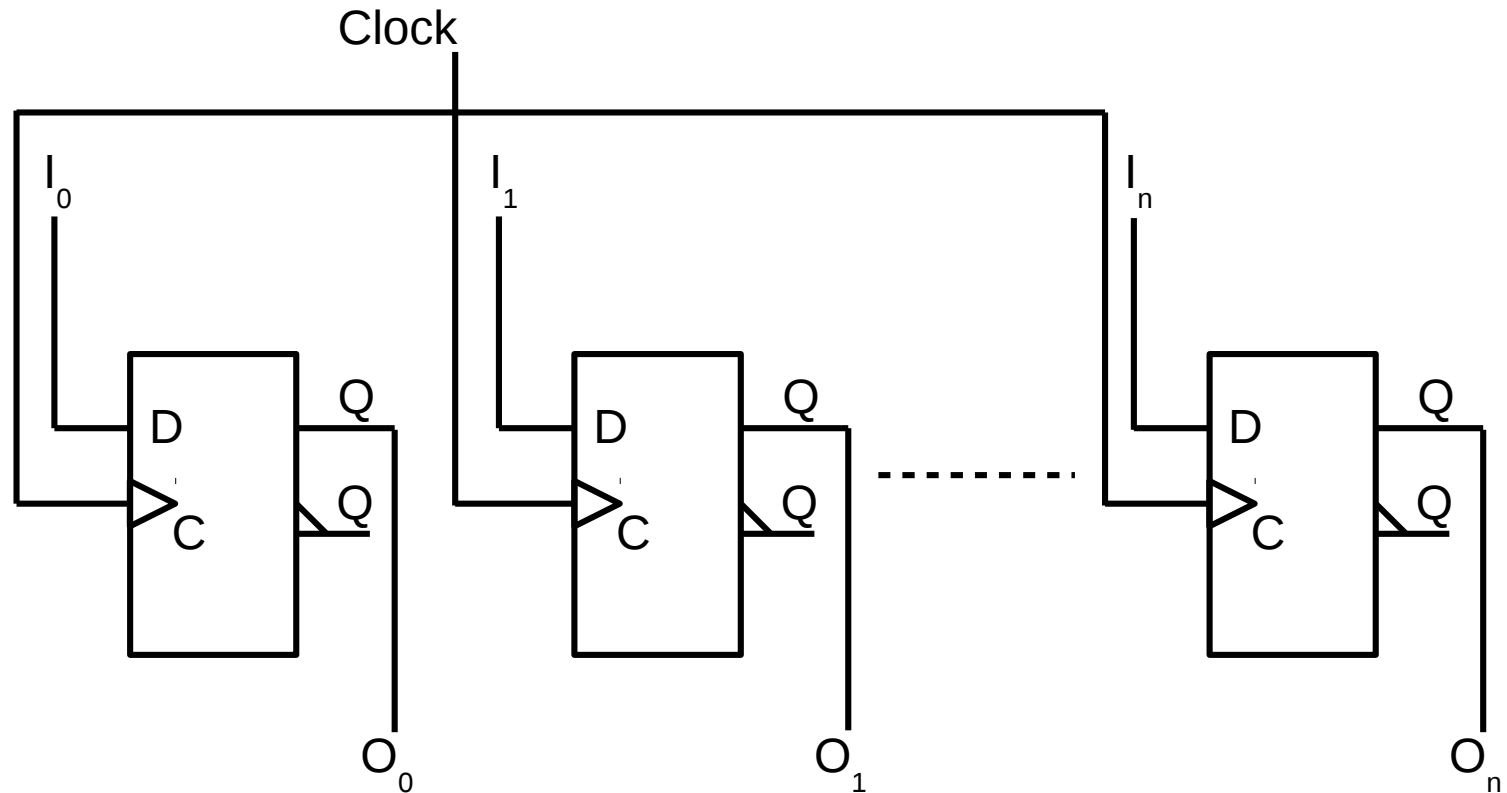
D	C	$Q_n$	$\overline{Q}_n$	Remarque
0	↑	0	1	Q recopie D
1	↑	1	0	Q recopie D
X	0	$Q_{n-1}$	$\overline{Q}_{n-1}$	mémoire
X	1	$Q_{n-1}$	$\overline{Q}_{n-1}$	mémoire

## Bascules (3)

- Il existent d'autres bascules
  - Asynchrone
    - ▶ RS, Verrou D, RSH-RST
  - Synchrone
    - ▶ D, JK, T, Bascule de Schmitt

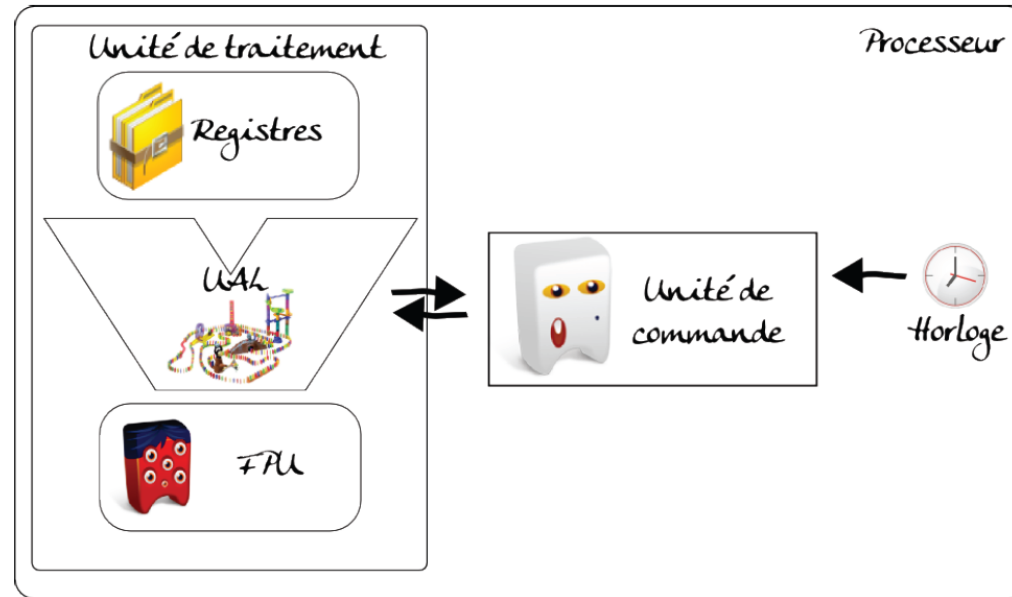
# Registres

- Un registre N-bits peut être réalisé à l'aide de N bascules



# Les éléments d'un processeur simple

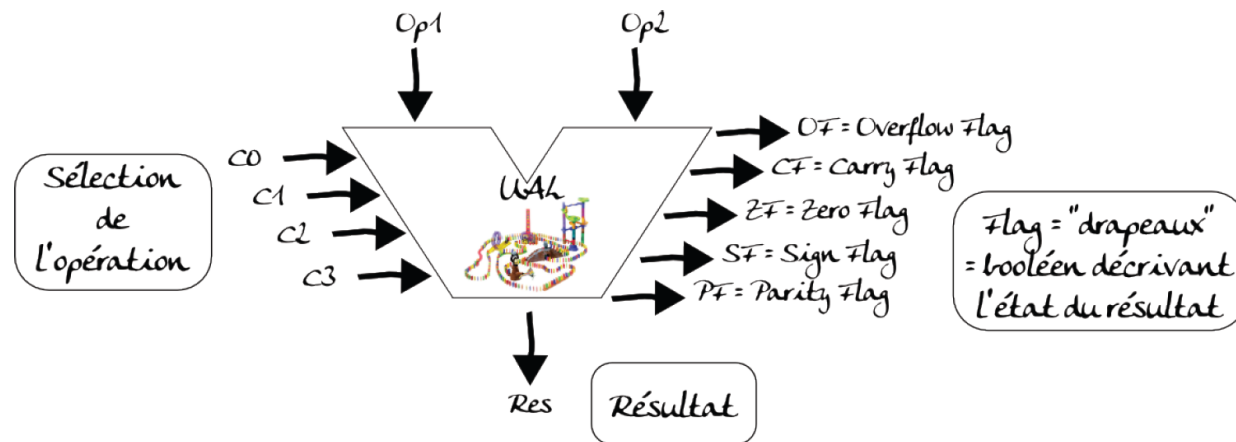
# Vision abstraite du processeur



- Registres = emplacements de stockage d'accès rapide
- UAL : unité de calcul (entiers et booléens)
- FPU : unité de calcul sur les « réels »
- Unité de contrôle : interprète les instructions
- Horloge : cadence le processeur (fréquence en MHz/GHz)

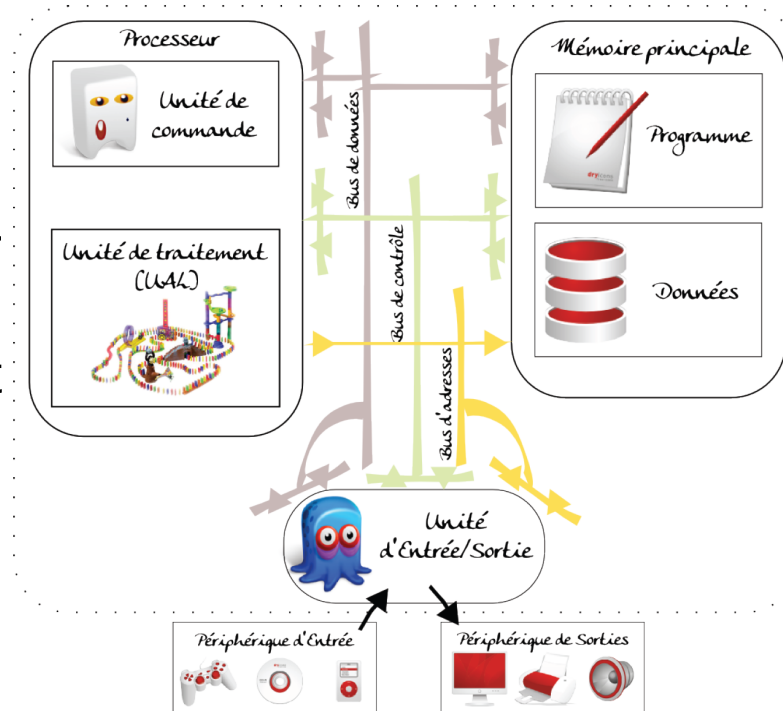
# UAL

- Unité chargée
  - des opérations arithmétiques
    - ▶ ADD (+), SUB (-), MUL (\*), DIV (:), INC (+ 1), DEC (- 1)
  - des opérations logiques
    - ▶ AND, OR, XOR, NOT, CMP
    - ▶ LSL, LSR, ASR (décalages)



# Vision abstraite d'un ordinateur

- Bus: nappe de fils conduisant l'information
  - Bus de données
    - ▶ @ d'un élément requis par le processeur
    - ▶ @ où écrire un élément envoyé par le processeur
  - Bus d'adresses
    - ▶ Transporte les informations de contrôle entre le processeur et les autres éléments

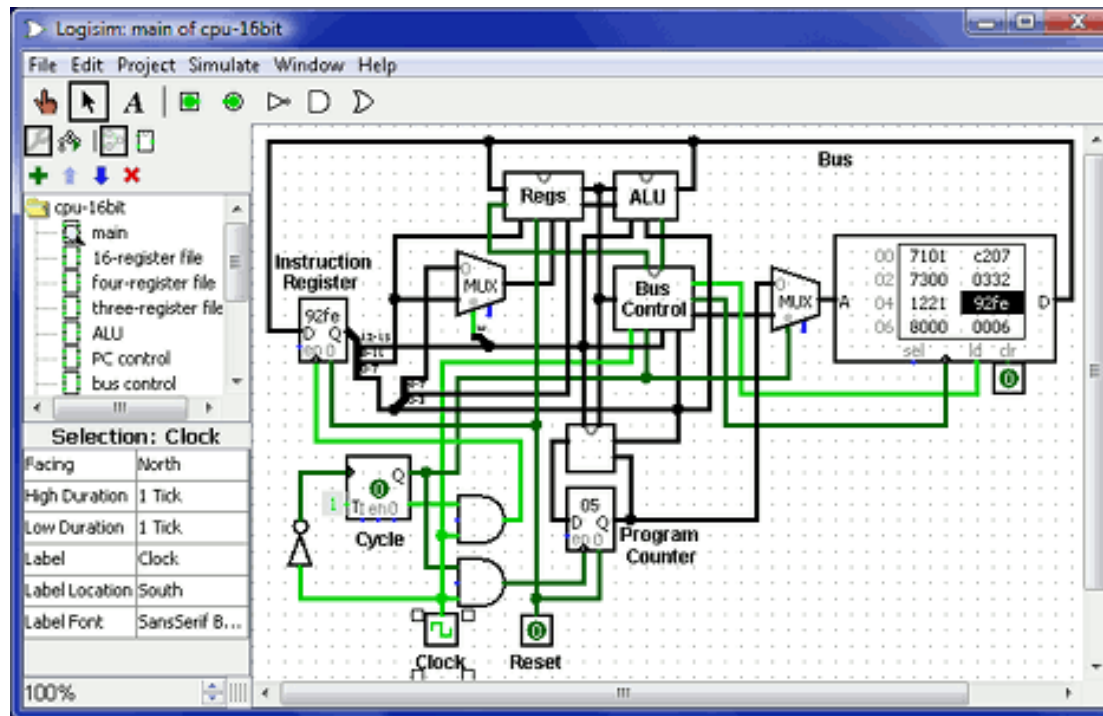


# Le projet



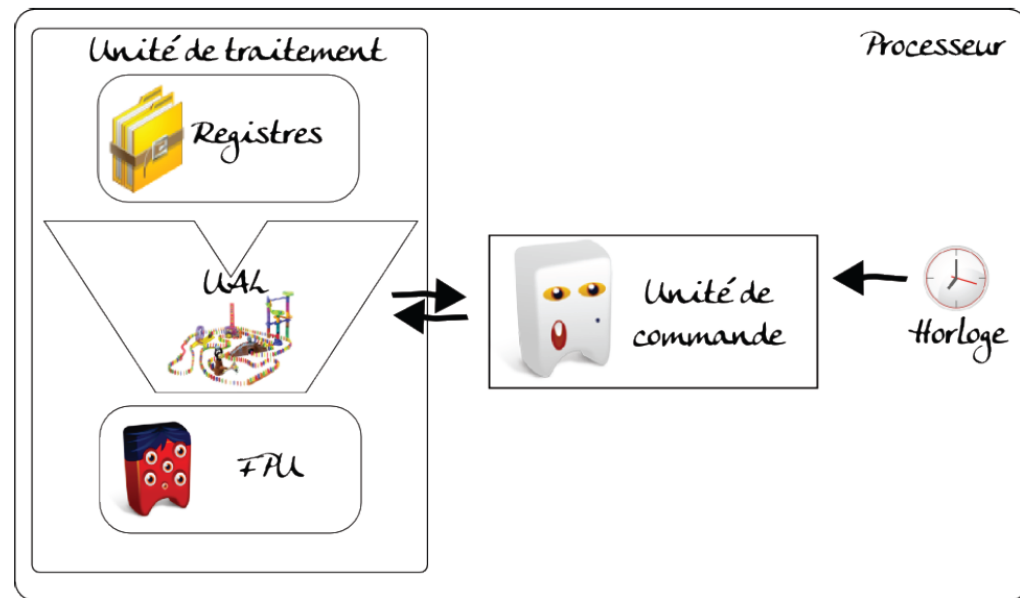
# Logisim

- <http://www.cburch.com/logisim/>



# Le projet

- 12H de TD/TP
  - Réalisation d'un processeur simple



# Évaluation

# Évaluation

- 1 évaluation écrite en fin de module
  - Avant dernière ou dernière séance de TD/TP
  - ~ 1H
    - ▶ Question de cours
    - ▶ Algèbre de Boole et table de vérité
    - ▶ Composants numériques simples
      - Portes logiques et Bascules
      - Chronogrammes
- Évaluation du projet
  - Un rapport écrit → 1/4
  - Réalisation → 3/4