Université Nice Sophia Antipolis

Langage d'Assemblage et Jeu d'Instructions

Cours (9x2h)

Frédéric MALLET – fmallet@unice.fr

TD (9x2h)

http://deptinfo.unice.fr/~fmallet/laji/

Université Nice Sophia Antipolis

Structure du cours

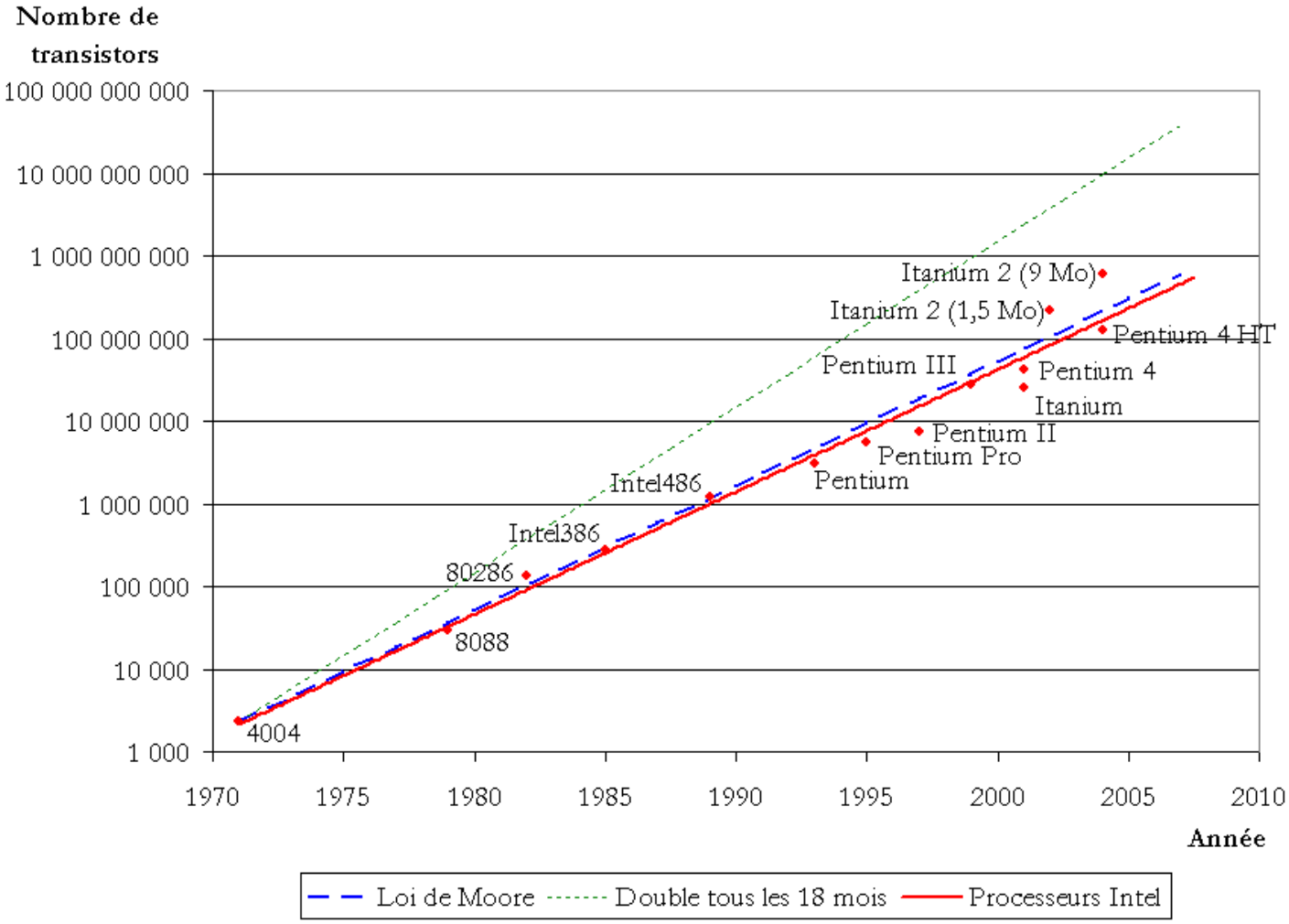
- Performances : évolution et comparaison
- Codage de l'information
- Fonctions logiques et éléments mémoires
- Systèmes à microprocesseurs
- La famille Intel - 80x86
- Le PowerPC d'IBM
- Éléments avancés (parallélisme, mémoire, ...)
- Conception d'architectures numériques - VHDL

Facteurs d'Évolution des performances

- Technologie :
  - Tubes à vide (46-57), transistors (58-64), SSI-MSI (65-71), LSI (72-77), VLSI(78-?)
- Densité : nombre de transistors sur une zone de silicium (< 0.1µm)
  - Loi de Moore : + 50% par an depuis 1970

Illustration de la loi de Moore

http://fr.wikipedia.org/wiki/Fichier:Loi_de_Moore.png

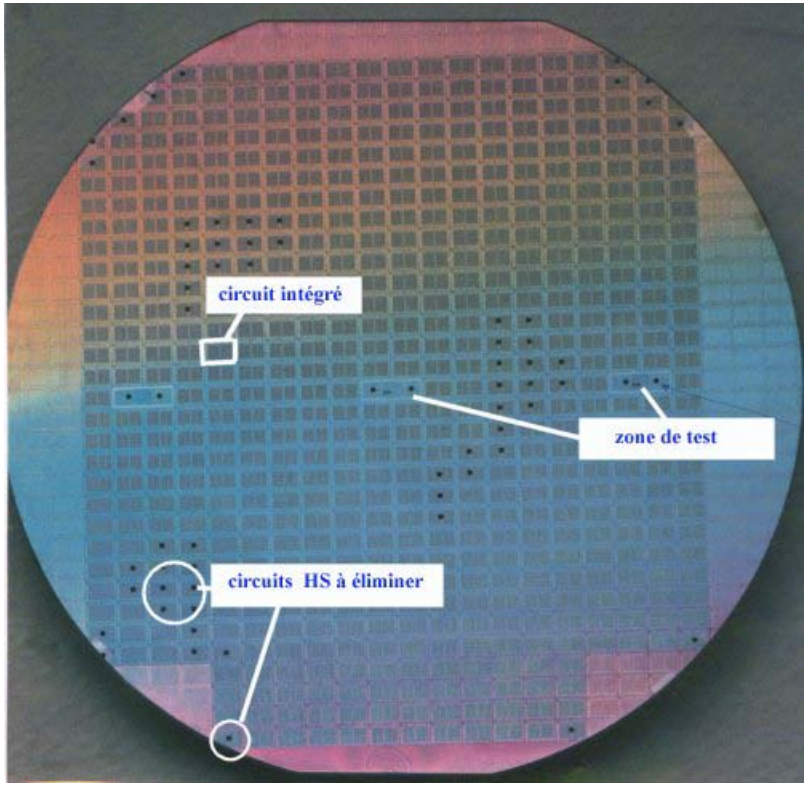


- Technologie
- Densité

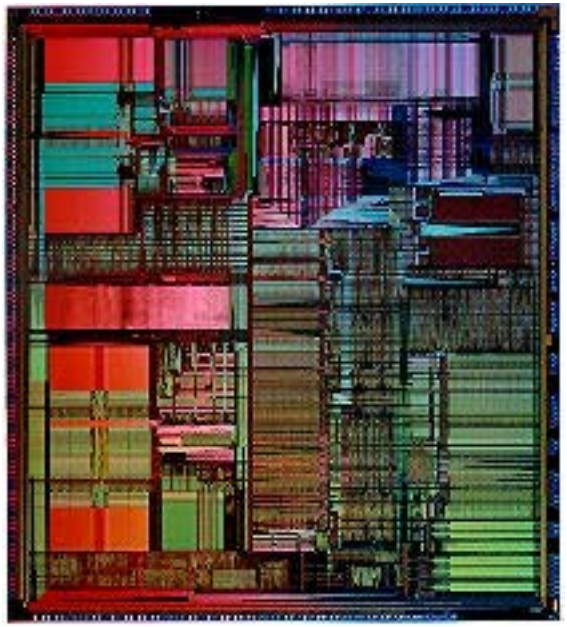
... mais aussi

- **Vitesse :** rapidité des portes logiques et mémoires
  - augmente en moyenne de 13% par an
- **Surface :** taille physique du + grand circuit intégré

Surface et *wafer*



30 cm de diamètre



1 circuit intégré

Université Nice Sophia Antipolis

Comparaison de performances

- Performance = terme très vague
  - le plus souvent associé à la rapidité
  - plusieurs métriques pour décrire les systèmes et sous-systèmes
- Compromis avec la facilité de programmation
  - Architectures orthogonales (RISC)
    - Faciles à programmer, mais moins optimisées
  - Architectures optimisées (DSP)
    - Très optimisées, moins faciles à programmer
  - Architectures multi-cœurs
    - Cœurs plus simples mais parallélisme difficile à programmer

Université Nice Sophia Antipolis

Banc d'essai (benchmark)

- Série de programmes représentatifs d'une famille d'applications donnée
- SPEC : Standard Performance Evaluation Corporation - http://www.spec.org
  - SPEC CPU2006 = CINT2006 + CFP2006
    - Vitesse: SPECint2006 + SPECfp2006
    - Débit: SPECint_rate2006 + SPECfp_rate2006
  - serveur web, graphique, unités de stockage
  - Temps d'exécution total ≠ MIPS ou IPC
  - Tous les constructeurs utilisent les mêmes tests

CINT2006

- Calculs intensifs sur les entiers
  - 400.perlbenchC Derived from Perl V5.8.7: SpamAssassin, MHonArc, and specdiff.
  - 401.bzip2 **C Compression** Julian Seward's bzip2 version 1.0.3, focus on memory rather than I/O.
  - 403.gcc Based on gcc Version 3.2, generates code for Opteron.
  - 429.mcf C Combinatorial Optimization. Network simplex algorithm to schedule public transport.
  - 445.gobmk Plays the game of Go, a simply described but deeply complex game.
  - 456.hmmer **Gene Sequence Protein** sequence analysis using profile hidden Markov models
  - 458.sjeng A highly-ranked chess program that also plays several chess variants.
  - 462.libquantum Simulates a **quantum computer**, running Shor's polynomial-time factorization algorithm.
  - 464.h264ref **H.264/AVC**, encodes a videostream. H.264/AVC should replace MPEG2
  - 471.omnetpp Uses the OMNet++ discrete event simulator: a large Ethernet campus network.
  - 473.astar Pathfinding library for 2D maps, including the well known A* algorithm.
  - 483.xalancbmk A modified version of Xalan-C++, **transforms XML** to other document types.

Moyenne arithmétique et géométrique

- Moyenne arithmétique de n entiers
  - Somme des entiers divisée par n
  - e.g. (4+2+4+82) / 4 = 23, (8+8+8+8)/4 = 8
- Moyenne géométrique de n entiers
  - Racine nième du produit des entiers
  - e.g. $\sqrt[4]{4\times2\times4\times82}=7,16$ $\sqrt[4]{8\times8\times8\times8}=8$
  - Moins sensible aux valeurs accidentelles

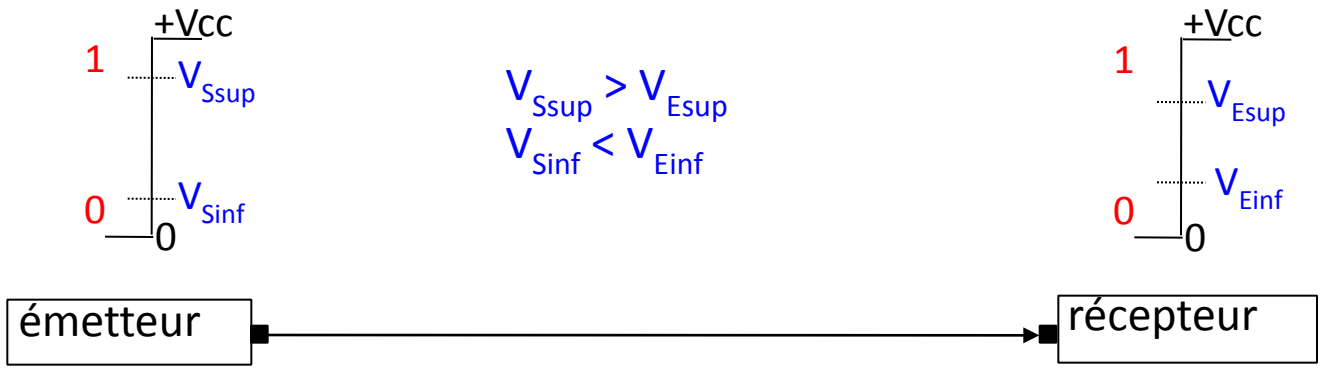
Université Nice Sophia Antipolis

Le monde du numérique

signal logique et mot

Numérique ou analogique ?

- **Analogique :**
  - Échelle continue de valeurs => sensible au bruit
- **Numérique :**
  - Échelle discrète de valeurs : **binaire**, ternaire, ... ?
  - BInary digiT = **BIT** (0 ou 1) : valeurs logiques



Mots de bits

- 1 signal logique représente 2^1 valeurs : bit
- N signaux représentent 2^n valeurs : mot
- 8 signaux représentent 2^8=256 valeurs : octet
- **Codage binaire naturel : entiers non signés**
  - bits ordonnés de 0 à n-1 (de droite à gauche)
  - Le $i^{ème}$ chiffre pèse 2^i : $\sum_{i=0}^{n-1} b_i * 2^i$ (e.g. 0b100111)
  - En décimal, le $i^{ème}$ chiffre pèse 10^i : $\sum_{i=0}^{n-1} b_i * 10^i$
    - $1995 = 1.10^3+9.10^2+9.10^1+5.10^0$

Généralisation en base b

$$N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0$$

Poids fort

Poids faible

$a_i b^i$ → Rang de a_i

Poids de a_i ← a_i ↘ base

a_i = 0 ou 1 => **bit**

- **Exemple :** Base 2

$$10_{10} = 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0$$

$$10_{10} = 1010_2$$

Indique la base

Numération Octale et Hexadécimale

- Octale : $8=2^3$ symboles `0, 1, 2, 3 ... 7`
- Hexadécimale : 16 symboles `0,1,2,3 ...9,A,B,C,D,E,F`
  - Passage de la base 10 à la base 8 ou 16
    - divisions successives par 8 ou 16
  - Passage de la base 2 à la base 8 ou 16
    - décomposition en groupe de 3 ou 4 bits
    - remplacement de chaque groupe par sa valeur dans la nouvelle base

Exemples : $1011101{,}01101_2$

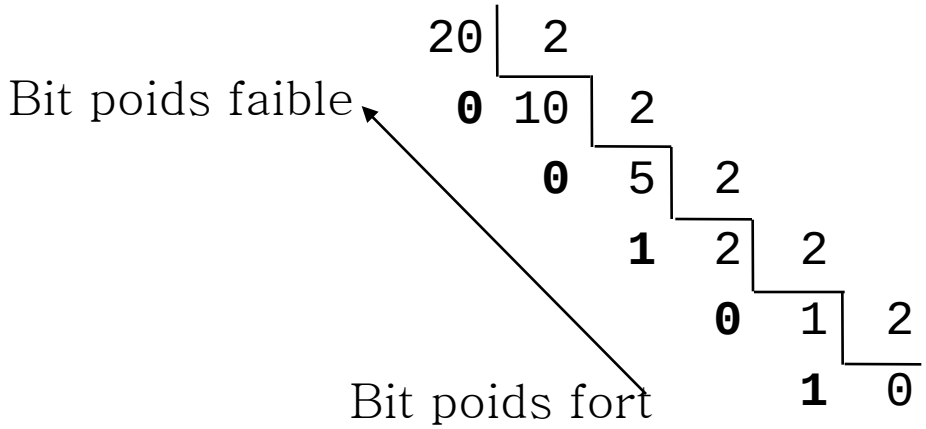
Base 8	$1 \mid 011 \mid 101 , 011 \mid 010$	$= 135{,}32_8$
Base 16	$101 \mid 1101 , 0110 \mid 1000$	$= 5D{,}68_{16}$

Partie fractionnaire (virgule fixe)

- $(0,572)_{10} = 5*10^{-1} + 7*10^{-2} + 2*10^{-3}$
- $(0,011)_{2} = 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$
- $(0,b_{-1}b_{-2}...b_{-n})_{B} = b_{-1}*B^{-1} + b_{-2}*B^{-2} + ... + b_{-n}*B^{-n}$
  - où $0 \leq b_{-i} < B$, $\forall i$

Passage de la base 10 à la base 2

- Partie entière : divisions successives par 2



$20_{10} = 10100_2$

- Partie fractionnaire : multiplications successives par 2

0,375	* 2	=	**0**, 75
0,75	* 2	=	**1**, 5
0,5	* 2	=	**1**

$0{,}375_{10} = 0.011_2$

$20{,}375_{10} = 10100.011_2$

Université Nice Sophia Antipolis

Signe/valeur absolue

- Le signe a 2 valeurs possibles (+ ou -) : 1 bit
  - Le bit de poids fort code le signe
  - les autres bits représentent la valeur absolue
  - e.g.
    - 0b00001100 représente la valeur +12
    - 0b10001100 représente la valeur -12
  - Avec n bits on représente les valeurs
    - de $-(2^{n-1}-1)$ à $+(2^{n-1}-1)$
- 2 problèmes
  - Comment coder 0 ?
  - Circuits spéciaux pour les opérations arithmétiques (cf. TP)

Université Nice Sophia Antipolis

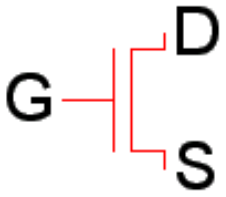
Blocs, fonctions et portes logiques

Blocs et fonctions logiques

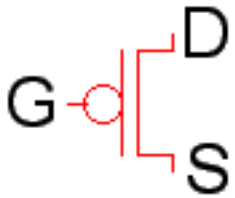
- Un bloc logique manipule des entrées binaires pour produire des sorties
- Chaque sortie réalise une fonction logique :
  - **Les fonctions combinatoires** (e.g. addition)
    - ne dépendent que des entrées E: S = f(E)
  - **Les fonctions séquentielles ou à mémoire**
    - dépendent des entrées et du passé (état Q) : S=f(E,Q)
    - l'état évolue au cours du temps : Q+ = g(E,Q)
    - e.g. compteur

Modèle logique du transistor

Transistor N



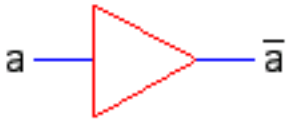
Transistor P



- Transistor N (Très bonne conduction si S = 0)
  - Quand G=0, le transistor est bloquant
  - Quand G=1, le transistor est passant (Source->Drain)
- Transistor P (Très bonne conduction si S = 1)
  - Quand G=0, le transistor est passant (Source->Drain)
  - Quand G=1, le transistor est bloquant

Fonction NON, inverseur

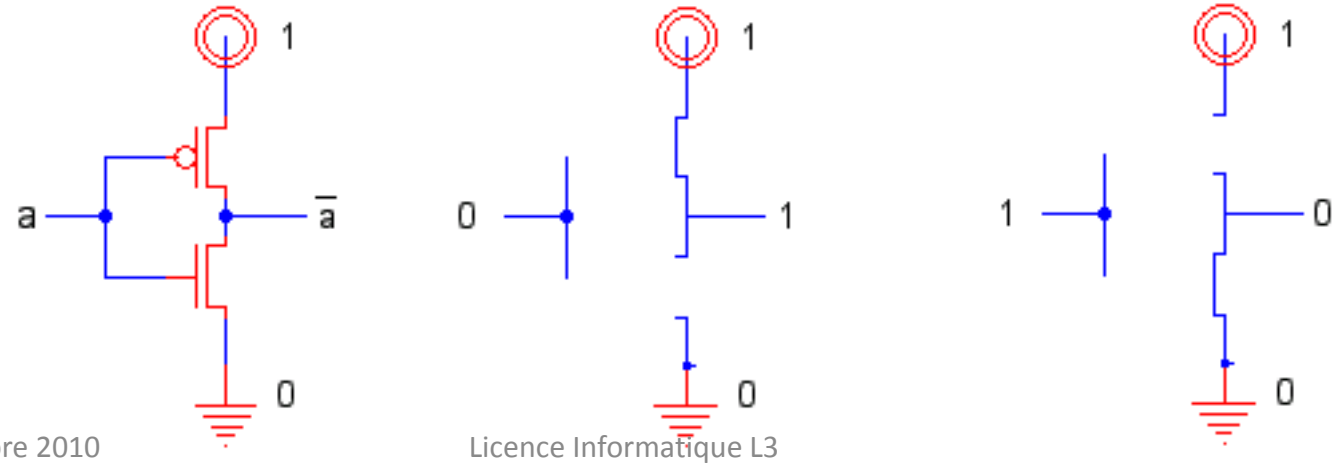
- Fonction logique NON :



- Table de vérité :

a	
0	1
1	0

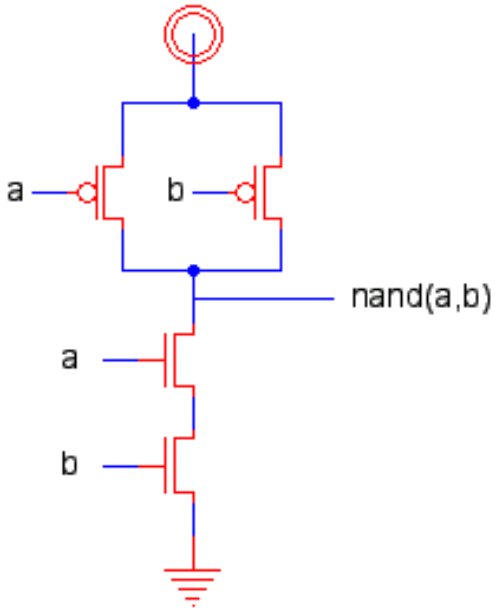
- Porte logique (réalisation) : inverseur
  - 1 transistor N et 1 transistor P
  - Tous les signaux sont de bonne qualité



Portes universelles : NAND

- Toute fonction logique peut être réalisée avec un seul type de "porte universelle":
  - La porte NAND qui réalise la fonction NAND

a	b	$\overline{a \bullet b}$
0	0	1
0	1	1
1	0	1
1	1	0

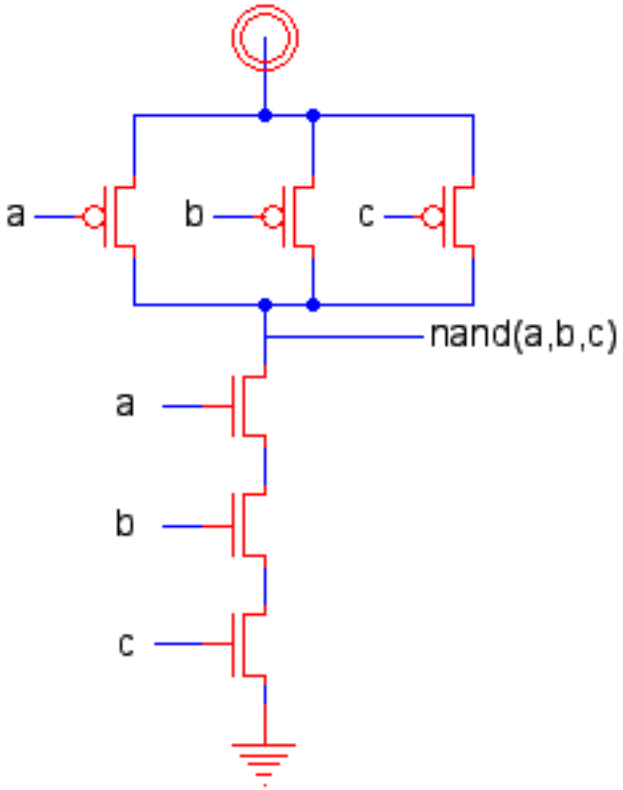


$$\overline{a \bullet a} = \overline{a}$$

Porte NAND à 3 entrées

- Le nombre de transistors en série dans les réseaux N et P est limité (3-5)

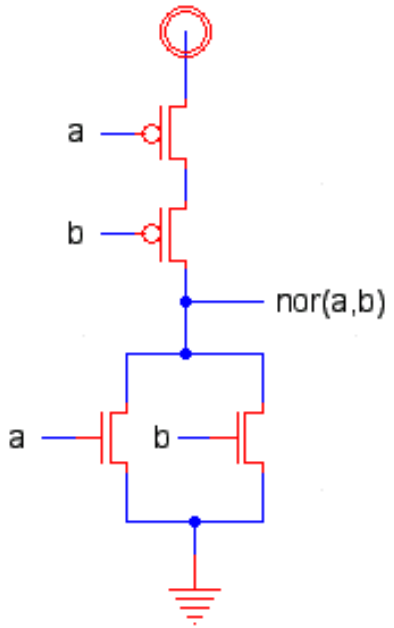
a	b	c	$\overline{a \cdot b \cdot c}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Portes universelles : NOR

- Ou la porte NOR qui réalise la fonction NOR

a	b	$\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0



$$\overline{a+a}=\overline{a}$$

Fonction ET

Produit logique (AND) : a • b

a	b	a•b
0	0	0
0	1	0
1	0	0
1	1	1

$$a \bullet 1 = a,\ a \bullet 0 = 0,\ \overline{a} \bullet \overline{b} = \overline{a + b}$$

$$a \bullet b = \overline{\overline{a} \bullet \overline{b}}$$

– Exercice : réaliser une porte AND avec que des NOR, compter les transistors

Fonction OU

- Somme logique (OR) : a + b

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	1

$$a+1=1,\ a+0=a,\ \overline{a}+\overline{b}=\overline{a \bullet b}$$

– Exercice : réaliser une porte OR avec que des NAND, compter les transistors

Fonction OU exclusif

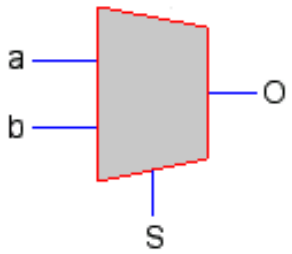
- Ou exclusif (XOR) :

a	b	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

$$a \oplus b = \bar{a} \bullet b + a \bullet \bar{b}$$

Multiplexage

Multiplexeur

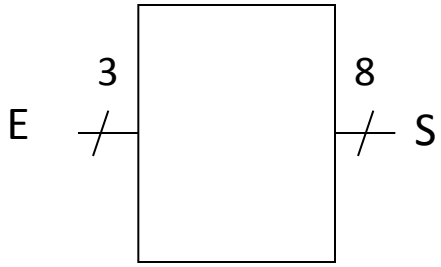


S	O
0	a
1	b

- La fonction multiplexage permet de sélectionner une entrée parmi 2
- Un multiplexeur à 2^n entrées nécessitent n bits de sélection

Décodeur n vers 2^n

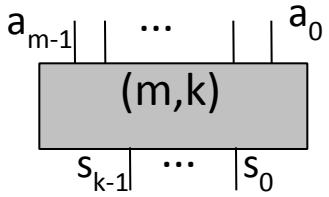
Décodeur 3 vers 8



E_2	E_1	E_0	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	**1**
0	0	1	0	0	0	0	0	0	**1**	0
0	1	0	0	0	0	0	0	**1**	0	0
0	1	1	0	0	0	0	**1**	0	0	0
1	0	0	0	0	0	**1**	0	0	0	0
1	0	1	0	0	**1**	0	0	0	0	0
1	1	0	0	**1**	0	0	0	0	0	0
1	1	1	**1**	0	0	0	0	0	0	0

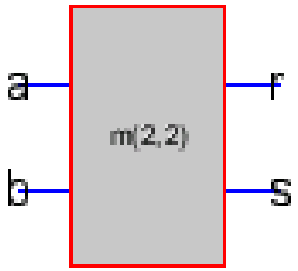
- La fonction décodage 3 bits permet de sélectionner une sortie parmi $2^3=8$
- Un démultiplexeur à 2^n sorties nécessitent n bits d'entrée
- Le numéro de la sortie sélectionnée est codé en binaire naturel sur l'entrée

Compteur (m,k)



- Entrée : mot de m bits Sortie : mot de k bits
- La fonction compteur(m,k)
  - compte le nombre de 1 présents sur ses m entrées
  - donne le résultat sur k bits en *numération simple de position* (binaire naturel)
- Un compteur (2,2) est appelé demi additionneur
- Un compteur (3,2) est un additionneur complet

Demi additionneur

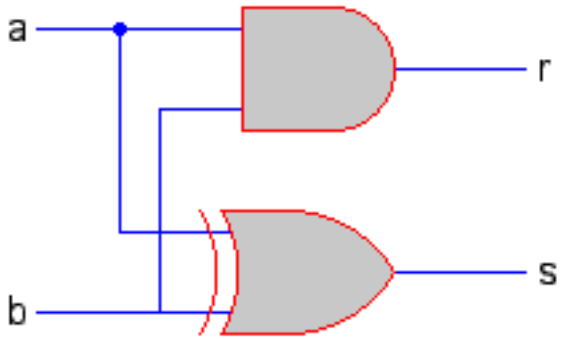


a	b	Nb	r	s
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

- Équation arithmétique : 2r+s = a+b
- Équations logiques

$s = a \oplus b$ $\qquad$ $r = a \bullet b$

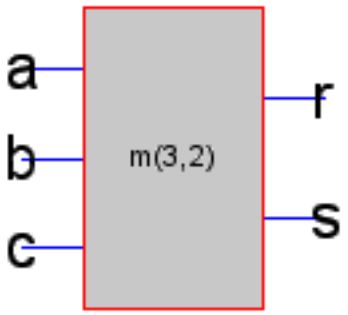
réalisation :



retenue

somme

Additionneur complet



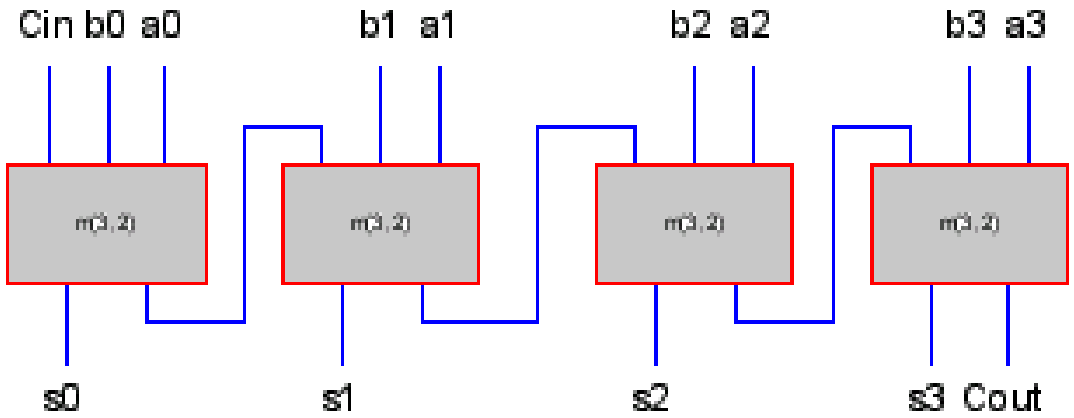
a	b	c	Nb	r	s
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	0	2	1	0
0	0	1	1	0	1
0	1	1	2	1	0
1	0	1	2	1	0
1	1	1	3	1	1

- Équation arithmétique : $2r+s = a+b+c$
- Équations logiques

$$s = a \oplus b \oplus c \qquad r = a \bullet b + a \bullet c + b \bullet c$$

- Plusieurs réalisations possibles (cf. TP)

Additionneur 4 bits



- 4 additionneurs complets en parallèle
- Si Cout = 1, dépassement (overflow)
- Attention: faux parallélisme
  - Le temps de calcul est proportionnel au nombre de bits : temps de propagation de la retenue
  - Il existe des versions avec anticipation de retenue

Le complément à 2

- 2 cas
  - **Nombres positifs :** binaire naturel, le bit de poids fort est 0
  - **Nombres négatifs :** inverser chaque bit de la valeur absolue, ajouter 1, le bit de poids fort est FORCEMENT 1
  - Exemple :
    - +12 est représenté par 00001100
    - -12 est représenté par 11110100

Inverser les bits

11110011

\+ 1

= 11110100

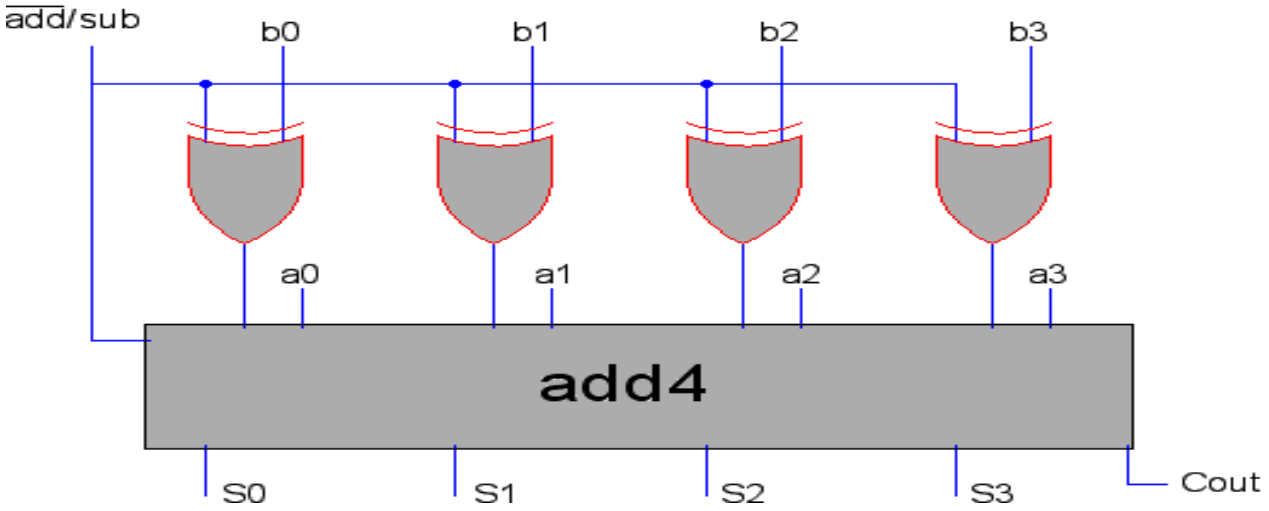
- En circuit :
  - Pas de circuit spécial pour l'addition signée
  - Pour la soustraction, on ajoute l'opposé
- Dissymétrique :
  - Représente les valeurs de -2^{n-1} à $2^{n-1}-1$

Additionneur/Soustracteur : 1 circuit

$$A-B=A+(-B)=A+\overline{B}+1$$

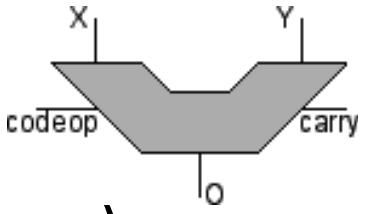
$$-B=\overline{B}+1 \quad \textit{(complément à 2)}$$

- En complément à 2 :

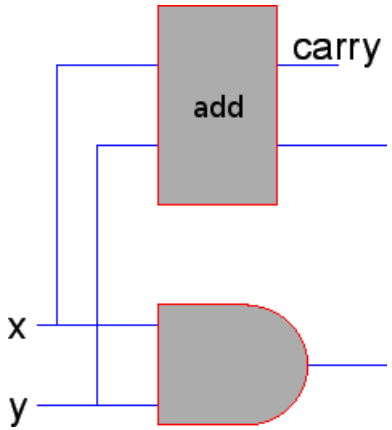


- Quand $\overline{add}$/sub vaut 0, additionneur 4 bits : $0 \oplus b_i = b_i$
- Quand $\overline{add}$/sub vaut 1, soustracteur 4 bits : $1 \oplus b_i = \neg b_i$
  - La retenue entrante du bit de poids faible est 1

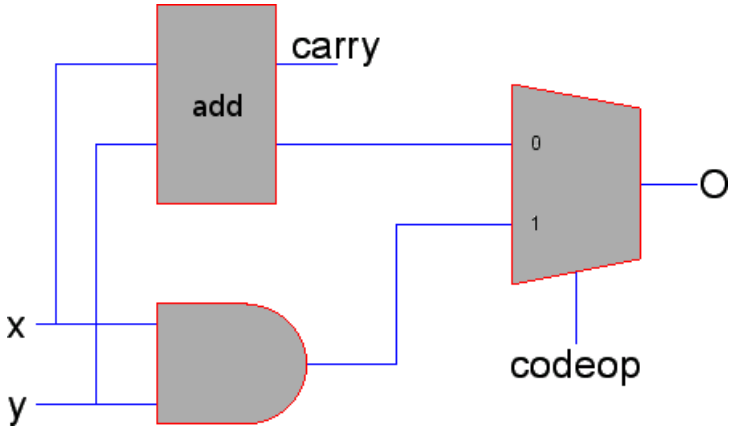
UAL 1-bit



- Unité Arithmétique et Logique
  - Opérations arithmétiques simples (+, -)
  - Opérations logiques (And, Or, Xor, Lsh, Ash)

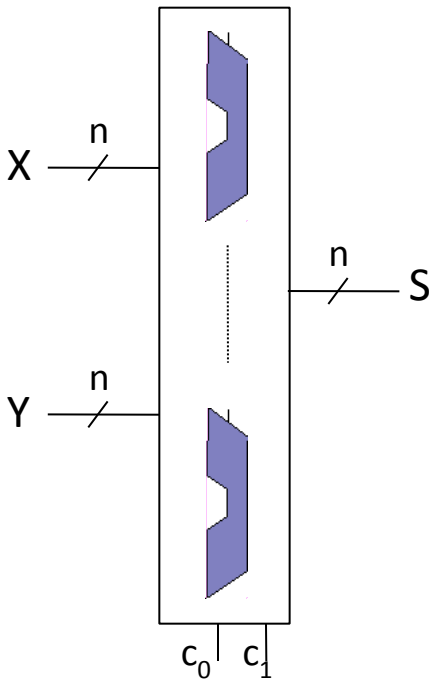


Les opérateurs sont mis en parallèle
A chaque fois qu'une entrée change
les deux opérations sont effectuées



Une seule sortie doit être valide
code opération (1 bit = 2 opérations)

UAL n-bit



- n ALU 1-bit en parallèle
- Les retenues des additionneurs sont cascadées
- Un code opération (c_0, c_1) pour choisir la valeur de sortie
- Une UAL produit aussi des indicateurs (flags):
  - Z : le résultat est nul
  - C : la retenue
  - N : le résultat est négatif
  - V : dépassement de capacité (overflow)

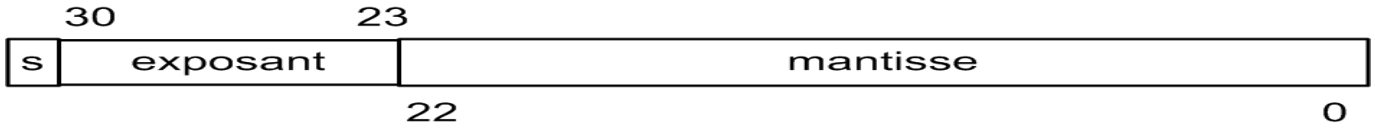
Université Nice Sophia Antipolis

Nombres approchés

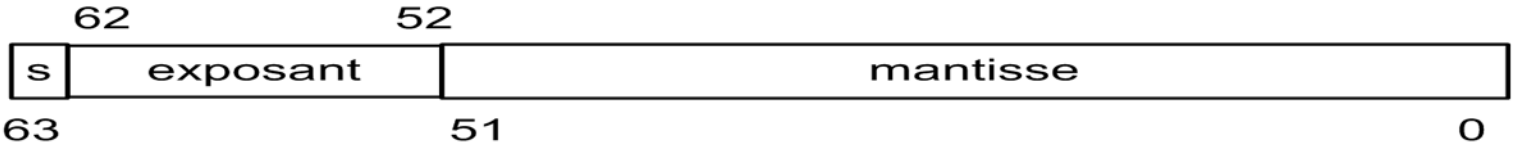
Partie fractionnaire
Représentation à virgule flottante

Nombres approchés

- Nombres à virgule fixe
  - Des bits sont réservés pour la partie imaginaire
- Nombres à virgule flottante : ANSI/IEEE 754
  - Représentés par mantisse x $2^{exposant}$
  - 2 précisions
    - Précision simple sur 32 bits



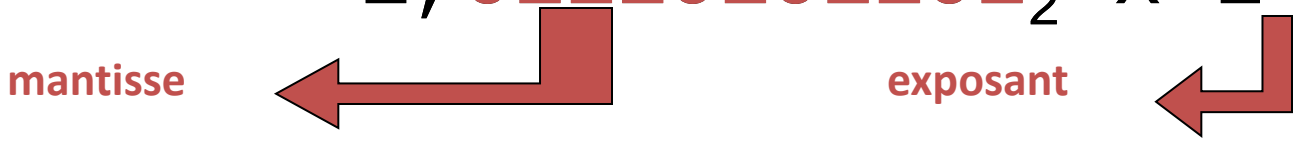
    - Précision double sur 64 bits



La virgule flottante

- On déplace la virgule après le premier chiffre significatif
- On compense avec un exposant
- Exemple :

$$1011101,01101_2 \times 2^0$$
$$101110,101101_2 \times 2^1$$
$$10111,0101101_2 \times 2^2$$
$$1011,10101101_2 \times 2^3$$
$$101,110101101_2 \times 2^4$$
$$10,1110101101_2 \times 2^5$$
$$1,01110101101_2 \times 2^6$$



Simple précision : float

Signe	Exposant	Mantisse	Nombre représenté
S	E=]0,255[	M	$(-1)^S x 2^{E-127} x (1,M)_2$
S	0	>0	$(-1)^S x 2^{-126} x (0,M)_2$
0	0	0	+0
1	0	0	-0
-	255	>0	NaN
0	255	0	$+\infty$
1	255	0	$-\infty$

- $(1,M)_2$ nombre binaire = mantisse préfixée de '1,'
- -0 : nombre négatif dont la valeur absolue est < à 2^{-149}
- NaN : Not a Number (e.g. Racine d'un nombre négatif)
- Infini signifie que la valeur absolue est > à $2^{127} x (2-2^{-23})$

Exemples

- 0 10000000 00000000000000000000000
  - = $(-1)^0 \times 2^{128-127} \times (1,0)_2 = 2$
- 0 10000001 10100000000000000000000
  - = $(-1)^0 \times 2^{129-127} \times (1,101)_2 = 4 \times 1.625 = 6.5$
  - car $(1,101)_2 = 1x2^0+1x2^{-1}+0x2^{-2}+1x2^{-3} = 1,625$
- 0 00000000 00000000000000000000001
  - = $(-1)^0 \times 2^{-126} x 2^{-23} = 2^{-149}$
  - Plus petit nombre que l'on peut représenter en simple précision

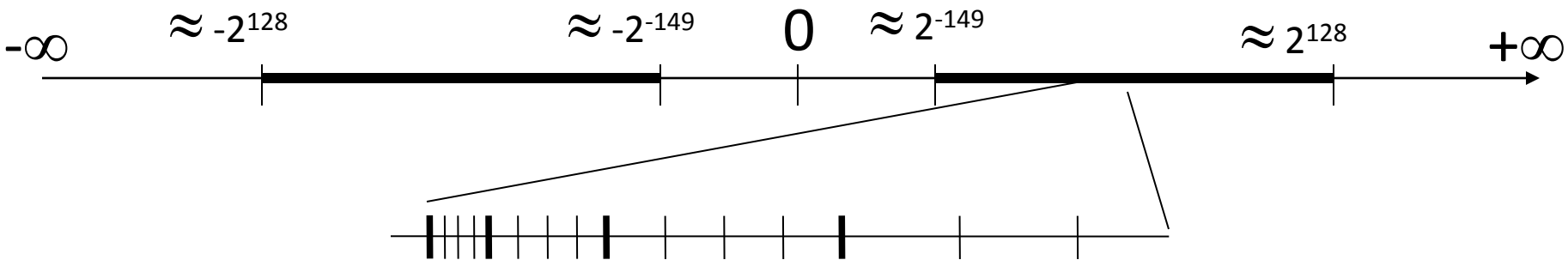
Double précision : double

Signe	Exposant	Mantisse	Nombre représenté
S	E=]0,2047[	M	$(-1)^S x 2^{E-1023} x (1,M)_2$
S	0	>0	$(-1)^S x 2^{-1022} x (0,M)_2$
0	0	0	+0
1	0	0	-0
-	2047	>0	NaN
0	2047	0	$+\infty$
1	2047	0	$-\infty$

- $2047 = 2^{11}-1$: 11 bits d'exposant, 52 bits de mantisse

Compléments sur IEEE 754

- La représentation en complément à 2 sur 32 bits permet de représenter un intervalle de 2^{32} valeurs
- La représentation en virgule flottante permet de représenter un intervalle de 2^{256} valeurs
  - Moins de précisions sur les valeurs très grandes et très petites
  - Bonne précision sur les valeurs entre -1 et 1



Arrondis

- Manipulations étendues :
  - simple précision étendue (43 bits)
  - double précision étendue (80 bits)
- 4 modes :
  - Round-to-nearest : par défaut
    - Si au milieu de 2 valeurs, 0 en poids faible
  - Round-to-zero : conversion en entiers
  - Round-to-+∞ et Round-to--∞

Précision

- $(0,1)_{10} = (1,1001\underline{1100}\ldots)_2 * 2^{-4}$
- $(0,1+0,1)_{10}$

  $=(1,1001\underline{1100}\ldots1101)_2 * 2^{-3}$ En précision infinie

  $=(1,1001\underline{1100}\ldots1100)_2 * 2^{-3}$ En précision sur 43 bits
- $(0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1)_{10}$

  $=(1,1001\underline{1100}\ldots1101)_2 * 2^{-1}$ En précision sur 43 bits

  $=(1,1001\underline{1100}\ldots1110)_2 * 2^{-1}=0.8$ En précision infinie

Représentation des caractères

- Caractères à coder, alphabet + , ! * " %
- Code ASCII (7 bits + 1 bit de parité)
  - **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - permet 128 combinaisons différentes 2^7
  - utilisation de cette table ...
- Code EBCDIC (8 bits)
  - **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode
  - Utilisé principalement par IBM
- Code ANSI
  - **A**merican **N**ational **S**tandard **I**nstitute
  - Utilisés par certains logiciels (Windows)
  - Code ASCII + extensions multilingue

Code iso-latin-1 (iso-8859-1)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STH	ETH	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	CD2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□
9	□	'	'	"	"	•	–	—	˜	™	š	›	œ	□	ž	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

'G' est codé 47h soit $0100\ 0111_2$

Bibliographie

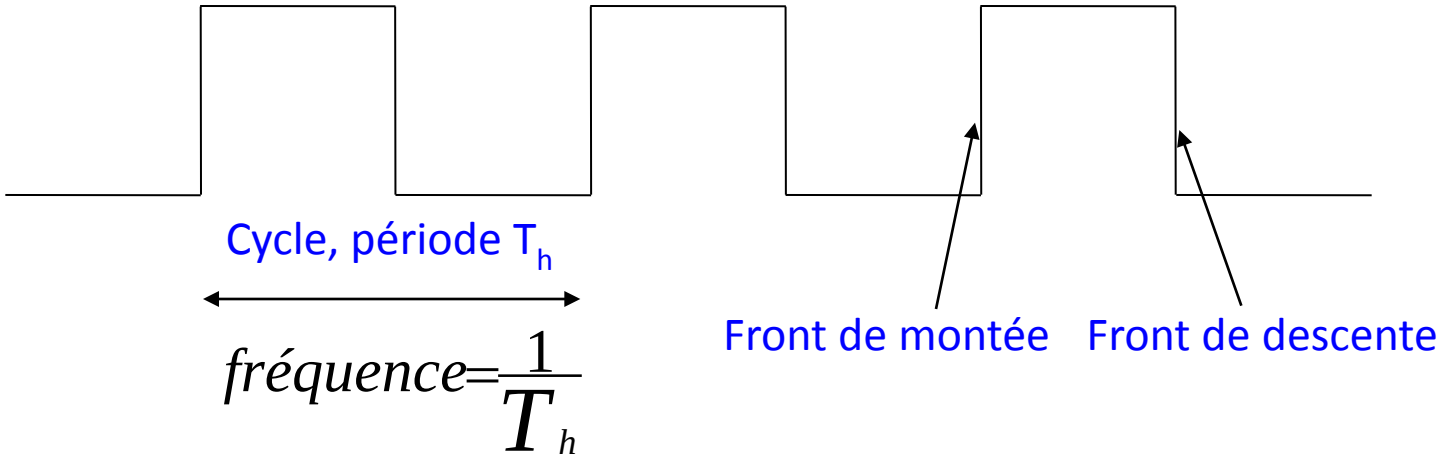
- *« Architecture de l'ordinateur »*
  - N. P. Carter, Schaum's, EdiScience, 2002
- *« Architecture des ordinateurs - Une approche quantitative »*
  - J.L. Hennessy, D.A. Patterson, Vuibert, 2003
- *« Architecture de l'ordinateur »*
  - A. Tanenbaum, Dunod, 5ème édition 2006

Logique séquentielle et mémorisation

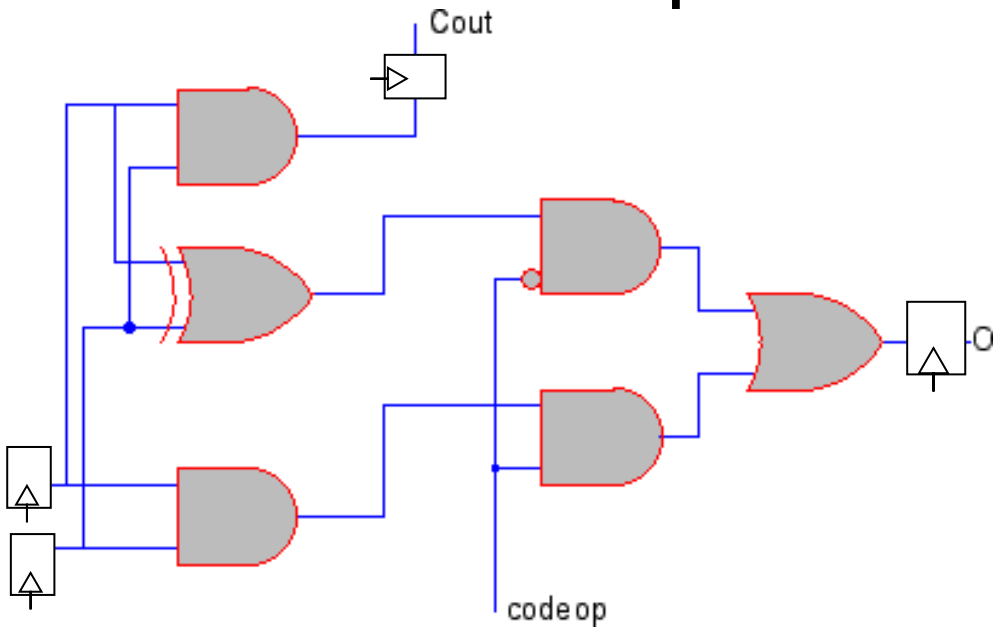
- Les fonctions combinatoires n'intègrent pas la notion de temps et de mémorisation
- Ces notions sont nécessaires au fonctionnement de plusieurs composants
- Notion de temps = Horloge
  - Une action d'un processeur est un enchaînement d'étapes
  - Les horloges sont des signaux périodiques qui déterminent l'instant auquel les différentes parties se synchronisent

Horloge (1/2)

- Logique séquentielle synchrone
  - Synchronisation sur front d'impulsion d'horloge (de montée ou de descente)
  - En général, une seule horloge par processeur
- **Les systèmes sans horloge sont dits asynchrones**



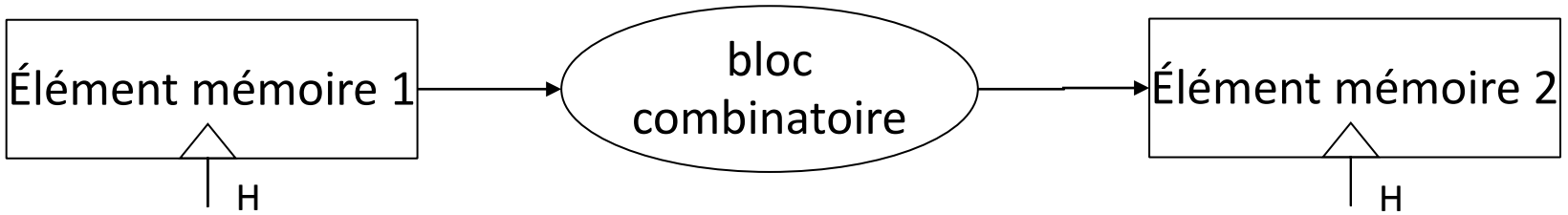
Exemple sur l'UAL 1-bit



- Tous les chemins n'ont pas la même longueur
  - Plus de portes à traverser (Cout, O)
  - Le temps dépend du routage
- Solution facile à mettre en œuvre
  - Synchroniser les entrées/sorties (éléments mémoire)

Horloge 2/2

- Les signaux d'entrée des composants mémoire doivent être valides lors du front d'horloge
  - Stable si les entrées ne changent pas
- La période de H doit être suffisamment grande par rapport au temps de stabilisation du plus lent des blocs combinatoires : $T_h > T_{max}$



Les éléments mémoire

- L'information à mémoriser est modélisée par un Etat Q
- Pour une mémoire élémentaire (1 bit) : Q+ = f(E,Q)
- Ces mémoires peuvent être réalisées avec de la logique combinatoire asynchrone
  - Rétroaction, utilise le délai des portes

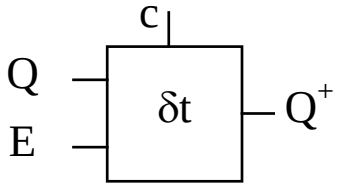
c	E	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

c=0, la barrière est fermée

Q+ = Q

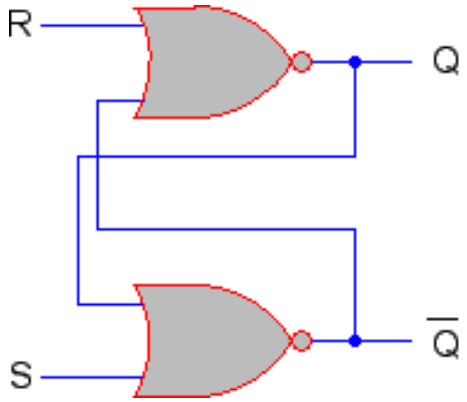
c=1, la barrière est ouverte

Q+ = E



Latch SR asynchrone

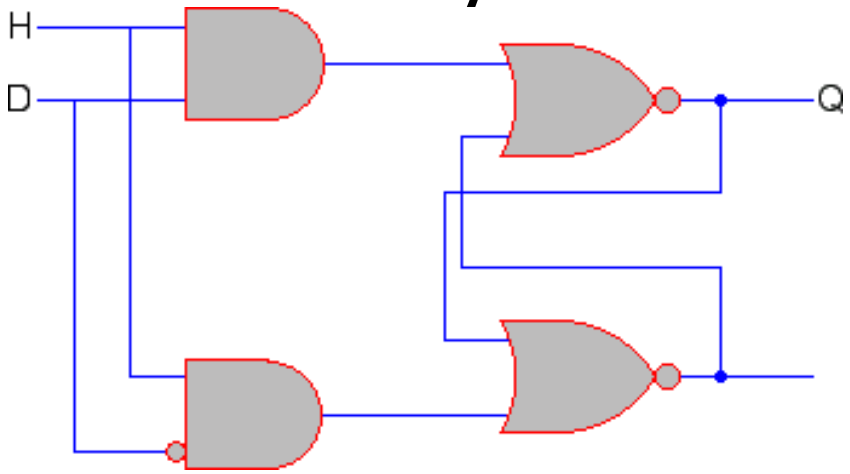
S(et)	R(eset)	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Non utilisé



- Complètement combinatoire
  - **La rétroaction et le délai réalisent la mémorisation**
  - Set : fixe la sortie à 1, Reset : fixe la sortie à 0
  - La combinaison S=1 R=1 est interdite et pose des problèmes d'oscillation (cf. TP2)

Latch D synchrone

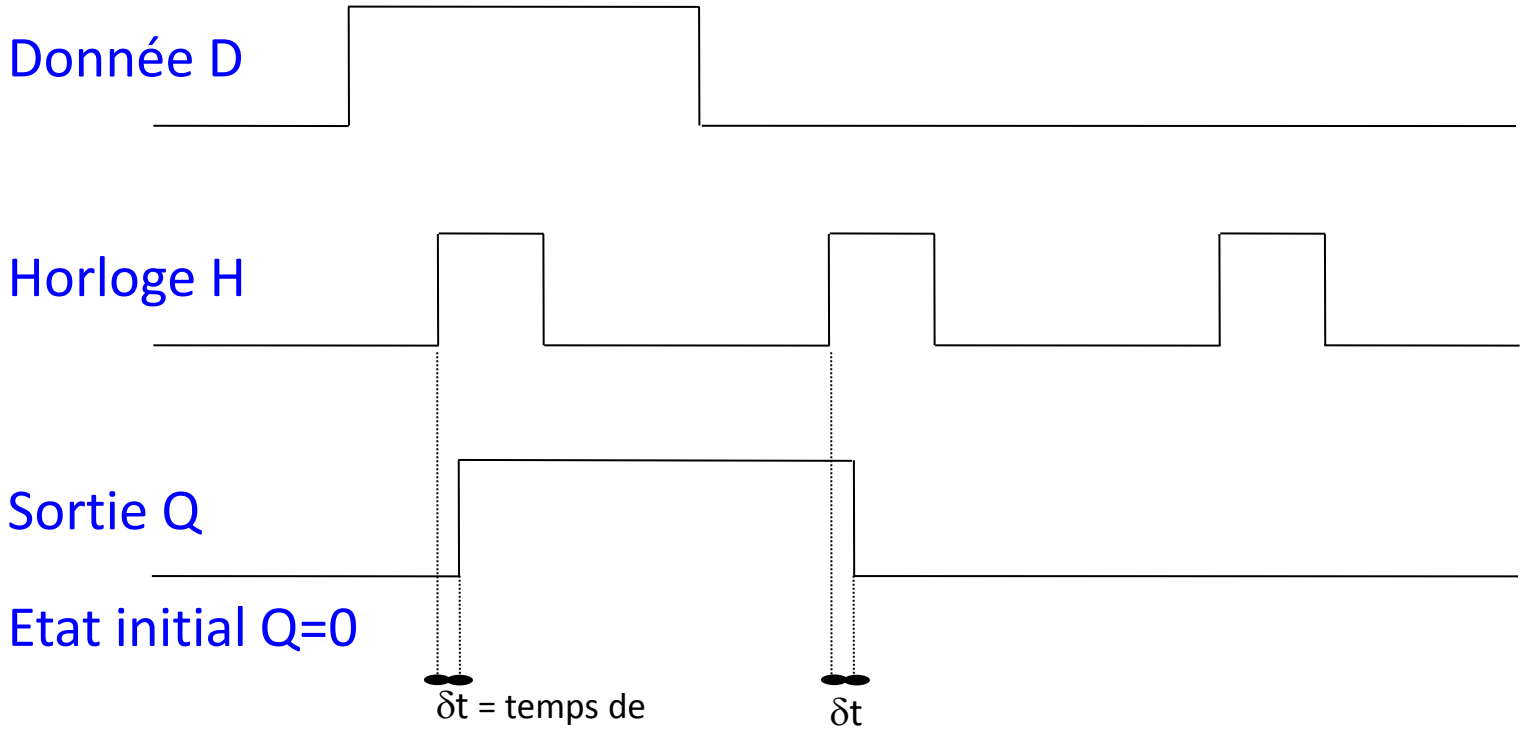
H	D	Q^+
0	0	Q
0	1	Q
1	**0**	0
1	**1**	1



- Le *latch* D est actif sur **niveau haut** de l'horloge
  - H=0, le *latch* est fermé, Q+ = Q
  - H=1, le *latch* est ouvert 'transparent', Q+ = D
- D doit être valide avant le front descendant de H
- R et S ne peuvent plus être à 1 simultanément

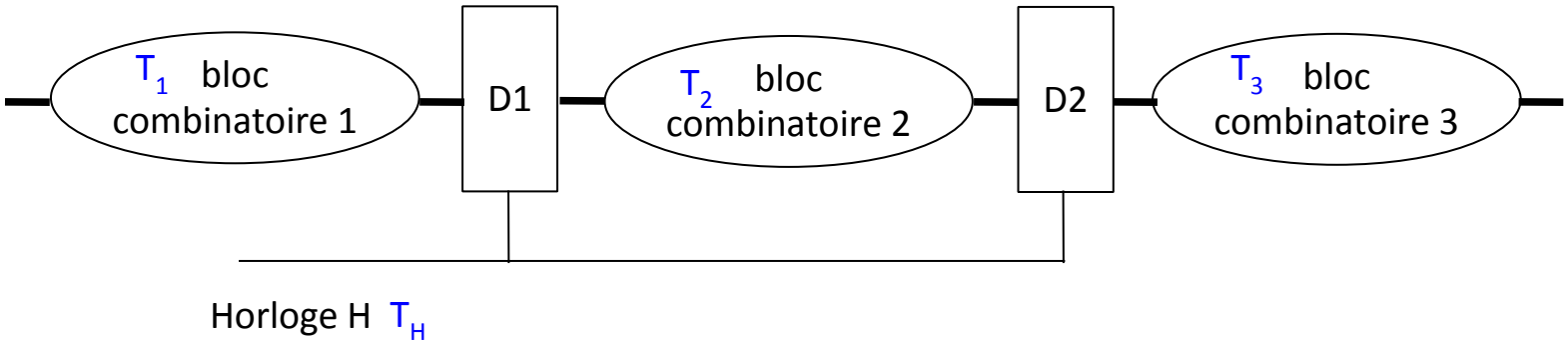
Exemple de bon fonctionnement

- Chronogramme :
  - Ici, l'horloge est périodique mais pas symétrique
  - *Latch* synchrone, car il y a une horloge



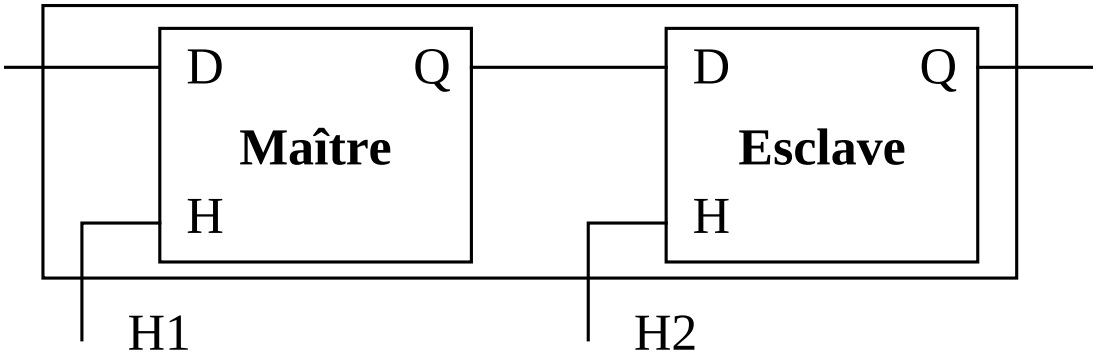
Limitations du latch D

- Un *latch* D peut séparer deux composants
- Mais ne convient pas pour plus de deux
  - Quand H=1, les deux *latchs* D1 et D2 s'ouvrent
  - **Les données traversent les 2 étages en 1 cycle**
    - Ssi $T1+T2 < T_H/2$



Bistable et Maître-Esclave

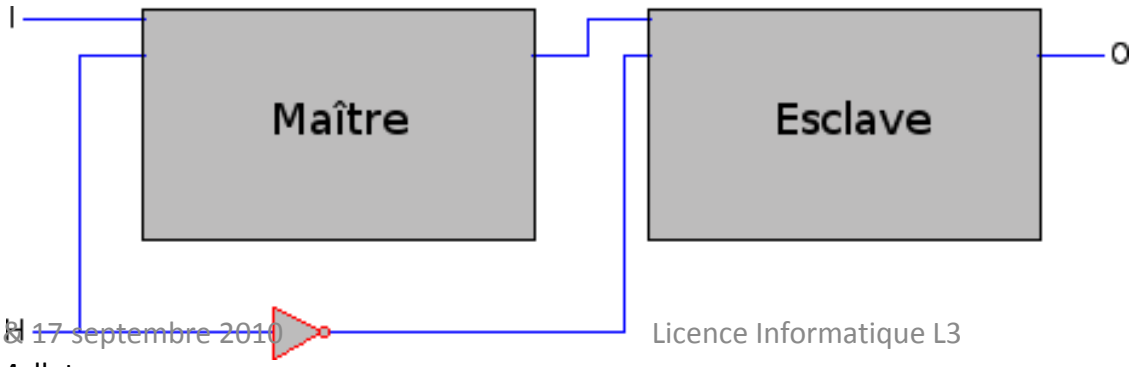
- Un **bistable** est une bascule synchronisée sur front d'horloge, le *latch* D est synchronisé sur niveau
- Réalisation avec une commande Maître-Esclave
  - 1 horloge H1 commande l'écriture : Maître
  - 1 horloge H2 commande la lecture : Esclave

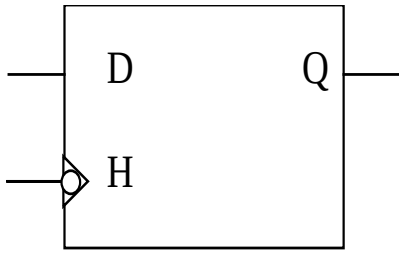


$$H1 \bullet H2 = 0$$

Bistable D - Flip-Flop

- En pratique, 1 seule horloge H génère H1 et H2
- Les sorties sont activées sur front de descente de H
- L'entrée I doit être valide avant et après le front actif
  - Temps d'établissement (*setup*),
  - Temps de maintien (*hold*)

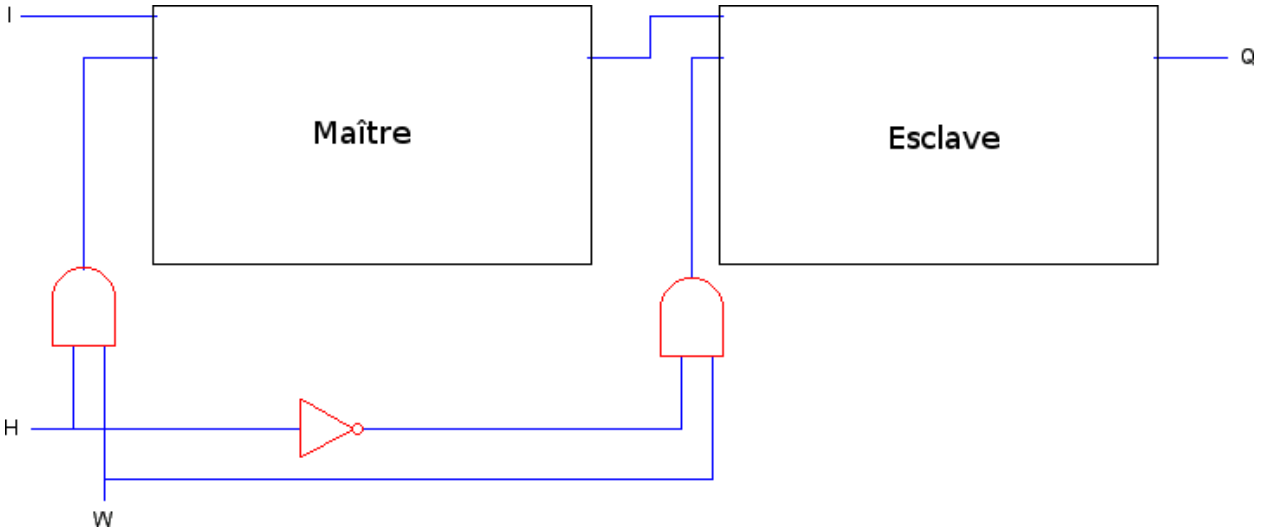


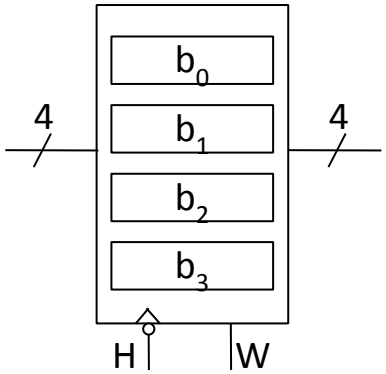


Bistable D

Les registres

- Un registre 1-bit est un bistable D avec 1 entrée W
  - L'entrée **synchrone** W(rite) contrôle l'écriture
  - W=1 : contrôle par l'horloge; W=0 : désactivé
- Un registre n-bit est constitué de n registres 1-bit

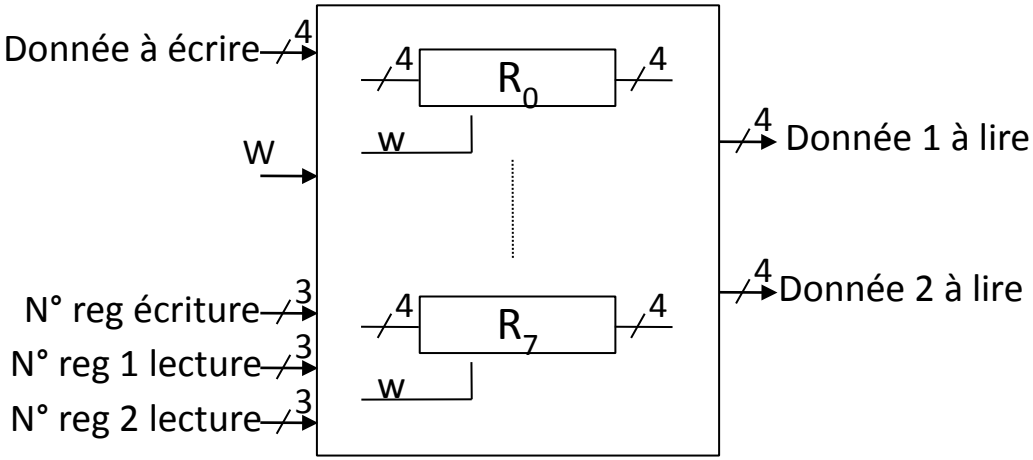




Registre 4-bit

Les bancs de registres

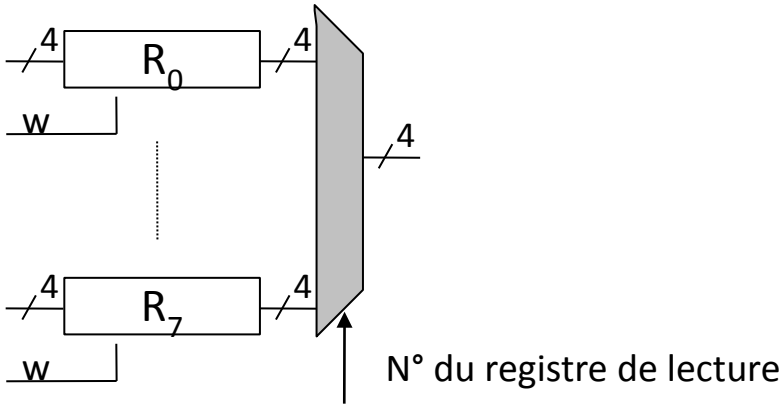
- Les registres peuvent être regroupés en bancs
  - On lit ou écrit en donnant le numéro du registre (adresse)
  - E.g. 2 lectures et 1 écriture dans le même cycle d'horloge
  - E.g. Pour un banc de 8 registres, il faut 3 bits de sélection



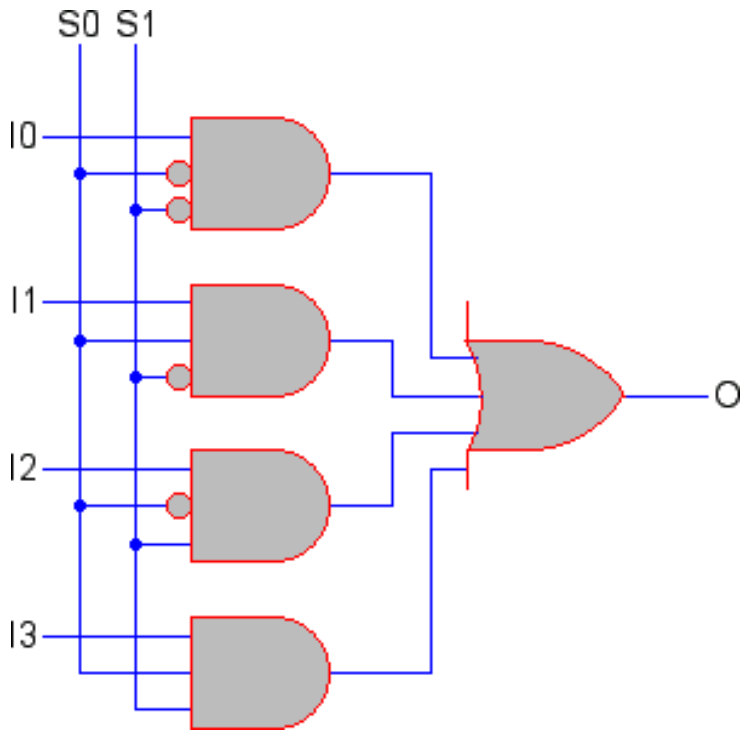
Banc de 8 registres de 4 bits

Commande de lecture - multiplexeur

- Pour la lecture : simple multiplexeur
  - Coût important et temps de traversée = O(n)
  - Que se passe-t-il quand on ne veut pas lire ?
  - Remarque : Il est interdit de connecter ensemble deux sorties logiques sauf ...



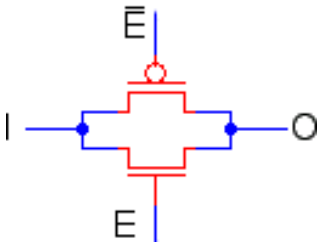
Coût du multiplexeur



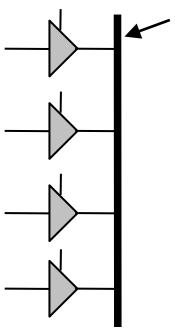
- Ici, seulement 4 entrées
  - 4 portes à 3 entrées
  - 1 porte à 4 entrées
  - Coût = Nombre de portes
    - en O(n)
- Banc de registres : 8 registres
  - 8 portes à 4 entrées
  - 1 porte à 8 entrées ????

Logique à 3 états et bus

- Une sortie 3 états peut prendre un des 3 états :
  - État logique 0, état logique 1
  - État déconnecté (pas logique dit “haute impédance”)
  - Un tel signal est réalisé avec une commande *Enable*
- 1 bus coûte bien moins cher qu’un multiplexeur



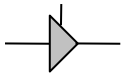
O=I ssi E(nable)=1



Ligne de bit commune (bus 1 bit)

1 contrôleur autorise 1 seule ligne à la fois :
e.g. Décodeur d'adresse

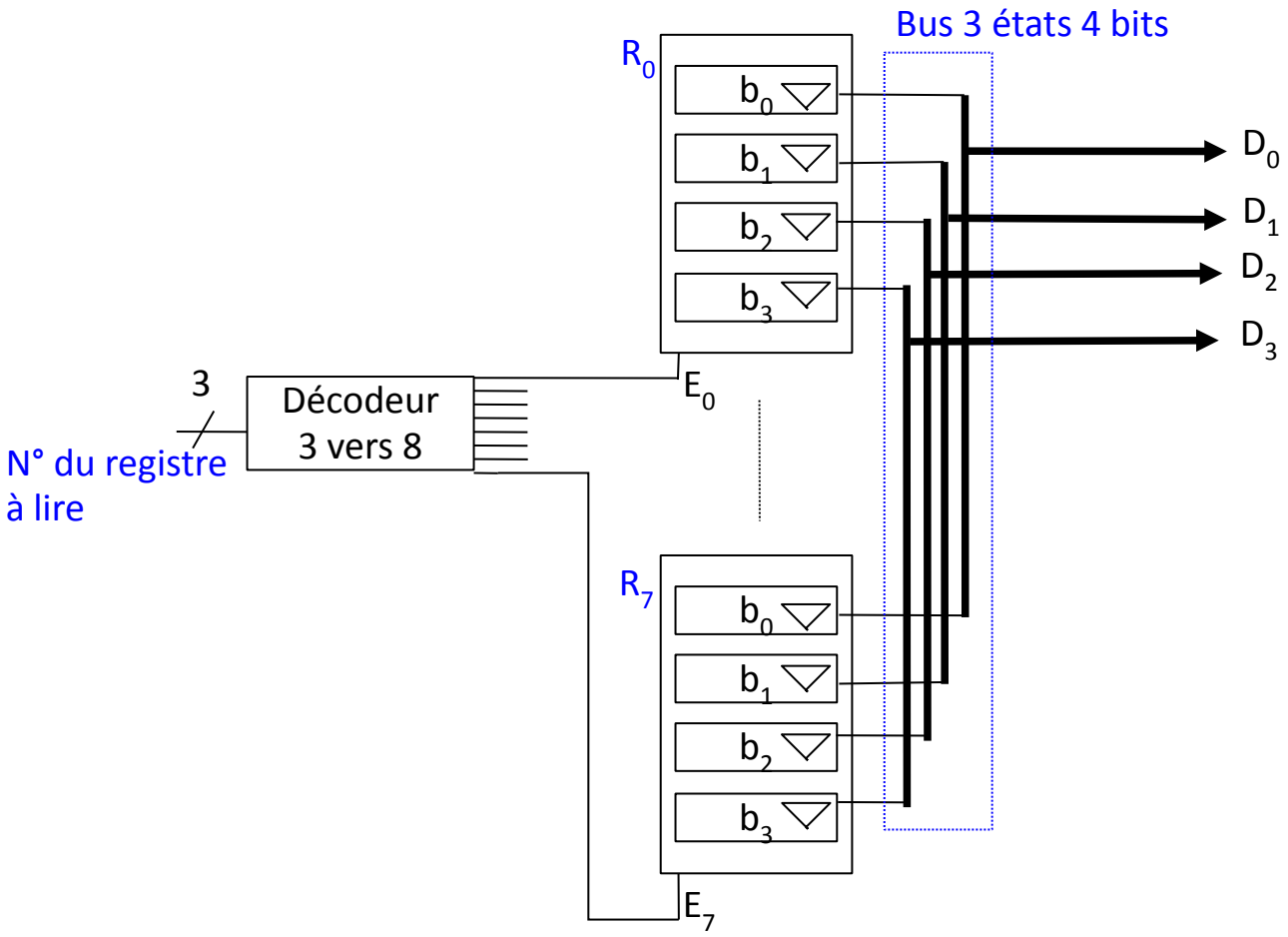
▽ symbolise une sortie à 3 états



Porte de transfert à 3 états

Banc de registres et bus 3 états

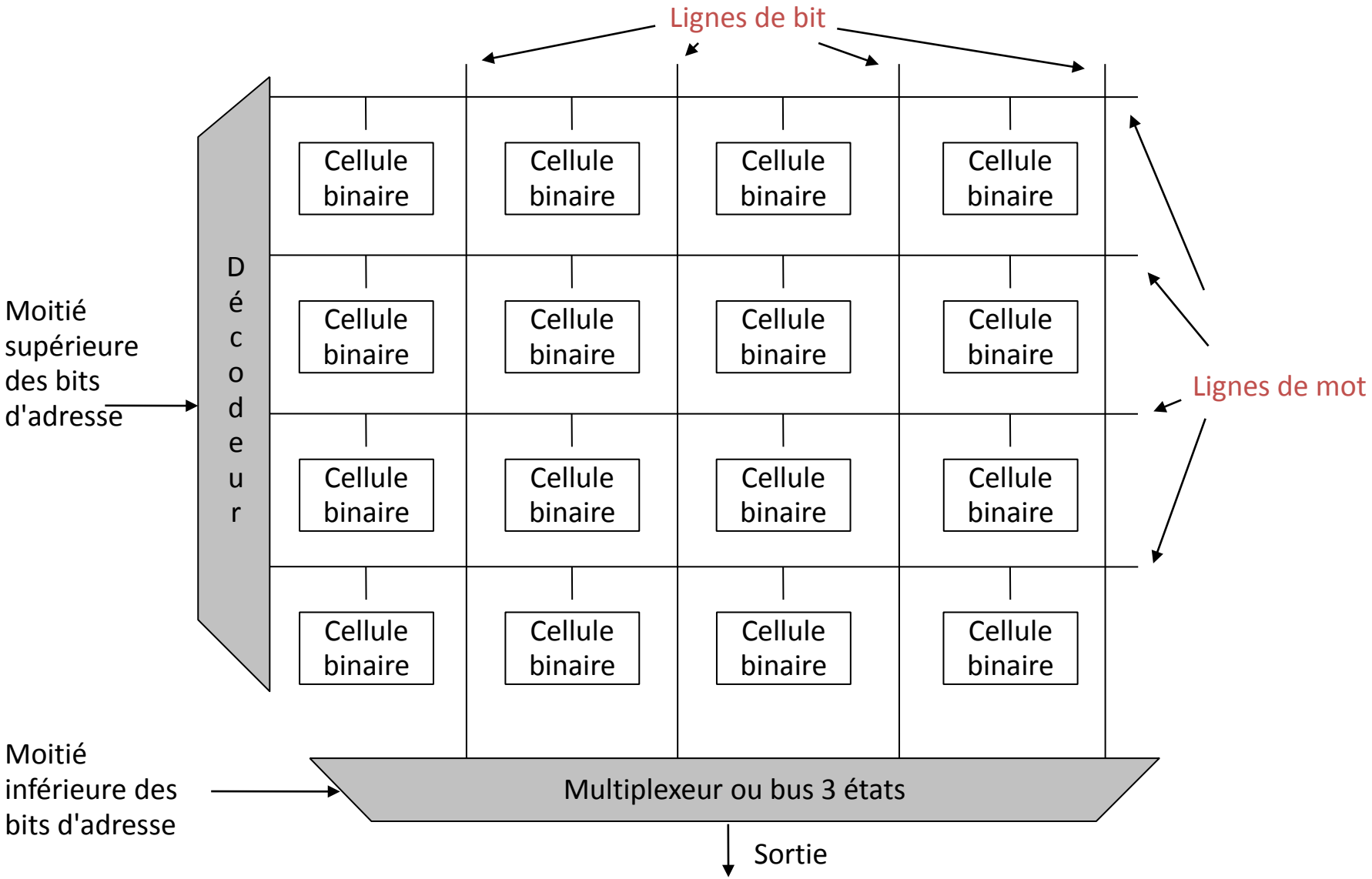
- Plus le bus est long, plus le temps de propagation est grand



Les mémoires RAM

- Réseaux de mémoires élémentaires de taille supérieure aux bancs de registres
- Coût (financier + surface) proportionnel à la rapidité
  - Mémoire rapide (statique) : SRAM (1ns)
    - Petites (cache)
  - Mémoire lente (dynamique) : DRAM (60-70ns)
    - Plus grandes (mémoire principale)
  - Disque dur : mémoire virtuelle (10ms)
    - Beaucoup plus lent (/10^3), beaucoup plus grand

Structure des puces mémoire



Structure des puces mémoire

- Chaque cellule binaire mémorise 1 bit
- Les bits de poids élevé sont chargés dans un décodeur pour activer une ligne de mot
- Les bits de poids faible sélectionnent la ligne de bit (multiplexeur ou contrôleur 3 états)
- Pour avoir une mémoire sur plusieurs bits
  - Plusieurs plans de cellules binaires (8 plans pour 1 octet)
  - Logique plus compliquée de sélection des lignes de bit
- Les matrices ne sont pas carrées :
  - Mémoire 8 bits : 8 cellules binaires par rangée (mot)

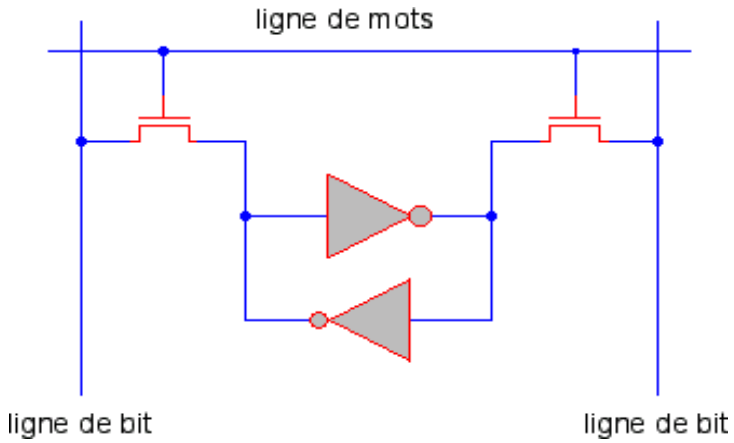
Université Nice Sophia Antipolis

Rapidité des mémoires

- La rapidité des mémoires est déterminée par :
  - La longueur des lignes de mot et des lignes de bit
    - Mémoires modernes : plus de petites matrices
  - La manière dont les cellules binaires sont construites
    - SRAM : 6 transistors, plus rapide mais plus grande
    - DRAM : 1 transistor, 1 pico-condensateur, moins rapide et moins grande
- Temps d'accès (latence)
  - Durée entre l'activation et l'obtention d'une donnée valide (peut être de plusieurs cycles)

Existe aussi en 8 ou 10 transistors
Lectures/Ecritures Multiples

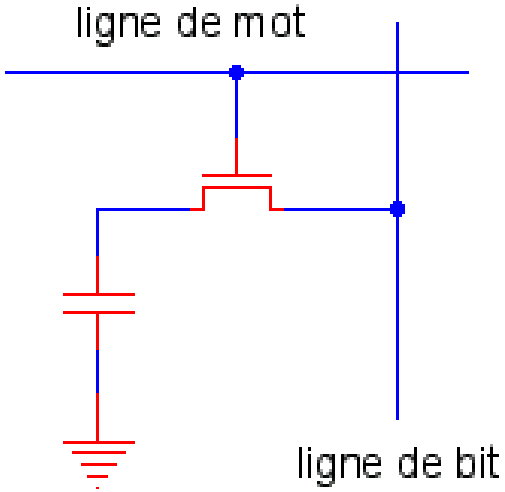
SRAM (Static RAM)



Cellule de SRAM
6 transistors

- Quand la ligne de mot est **inactive**
  - La donnée est mémorisée indéfiniment (statique)
- Quand la ligne de mot est **active**
  - **Lecture** : la valeur et son inverse vers les lignes de bit
    - Puis sélectionnées en sortie
  - **Écriture** : les 2 lignes de bit sont alimentées avec la valeur à mémoriser et son inverse

DRAM (Dynamic RAM)



Cellule binaire DRAM
1 transistor
1 condensateur (1 connexion)

- Lorsque la ligne de mot est **active**, le condensateur est connecté à la ligne de bit
- Sinon, il se décharge petit à petit (dynamique)
- **Les DRAM ont une capacité plus grosse que les SRAM et sont plus lentes**
- Les DRAM doivent être rafraîchies régulièrement

Rafraîchissement de DRAM

- Les rangées sont rafraîchies en lisant la valeur et en la ré-écrivant immédiatement
- Si les rangées sont rafraîchies suffisamment souvent, leur contenu est préservé indéfiniment
- Les DRAM indique un **temps de rafraîchissement**
  - Plusieurs politiques de rafraîchissement existent
  - Par exemple, on peut rafraîchir chaque rangée les unes après les autres

DRAM et DRAM paginées

- DRAM
  - La transmission de l'adresse se fait en 2 phases
    - RAS : l'adresse de la rangée (bits de poids forts)
    - CAS : l'adresse de la colonne (bits de poids faibles)
- DRAM paginées (Fast Page Mode)
  - Lors de RAS tous les bits de la rangée sont mémorisés dans un *latch*
  - Plusieurs CAS successifs permettent de lire tous les bits de la rangée et de réduire le temps d'accès

DRAM Extended Data Out

- Apparues en 1995 en remplacement de FPM
  - Lorsqu'une donnée est lue, l'adresse de la rangée suivante est générée automatiquement
  - Bonne accélération, lorsqu'on lit des données consécutives
  - Cette technologie asynchrone est limitée par la fréquence d'horloge
    - → Mémoires synchrones « avec le processeur »

Université Nice Sophia Antipolis

Mémoires synchrones (1/3)

- SDRAM (Synchronous DRAM) - 1997
  - La mémoire se branche sur la fréquence d'horloge du processeur, le processeur n'a plus besoin d'attendre
  - PC66=66MHz, PC100=100MHz, PC133=133Mhz
- DDR-SDRAM (Double Data Rate SDRAM)
  - Apparue en 1999,
  - Plusieurs canaux de lecture en même temps (2 ou 4)
  - En pratique on couple 2 barrettes mémoires
  - PC1600=100MHz, PC2100, PC2700, PC3200
    - 100MHz => 10^8 transferts/s * 2 (DDR) * 8 octets = $16*10^8$o/s = 1,6Go/s
    - 100MHz => DDR-200 (200 millions de transferts par seconde)

Mémoires synchrones (2/3)

- DDR2-SDRAM (Double Data Rate SDRAM)
  - Buffer en plus de la DDR => double le débit, consomme beaucoup moins que la DDR et n'est pas compatible
  - PC2-3200=100MHz ≡ DDR2-400
    - 100MHz => 10^8 transferts/s * 2 (DDR) * 2 (buffer) * 8 octets = $32*10^8$o/s = 3,2Go/s
  - PC2-4200=4,266Go/s (PC2-4300) ≡ DDR2-533
  - PC2-5300 (5400) ≡ DDR2-667
  - PC2-6400 ≡ DDR2-800
  - PC2-8500 (8600) ≡ DDR2-1066

Mémoires synchrones (3/3)

- DDR3-SDRAM (Double Data Rate SDRAM)
  - 2 fois plus rapide que la DDR2, consomme moins à fréquence identique, n'est pas compatible
  - PC3-6400=100MHz (DDR3-800)
    - 800 M transferts par seconde
    - 100MHz => 10^8 transferts/s * 2 (DDR) * 4 (buffer) * 8 octets = $64*10^8$ o/s = 6,4Go/s
  - PC3-8500=8,533Go/s (DDR3-1066)
  - PC3-10600 (DDR3-1333)
  - PC3-12800 (DDR3-1600)

Université Nice Sophia Antipolis

Conditionnement

- SIMM (Single Inline Memory Module)
  - Circuits imprimés rectangulaires, 32-bit
  - 30 ou 72 broches
- DIMM (Dual Inline Memory Module)
  - 100, 168, 184, 240 broches, 64-bit
  - Plus besoin de coupler 2 barrettes (2 bancs sur la même barrette)
  - Détrompeur pour l'installation
- SO-DIMM (Small-Outline DIMM)
  - Moitié de la taille de la DIMM => portable
  - 72, 100 pins => 32-bits
  - 144, 200 or 204 pins => 64-bits

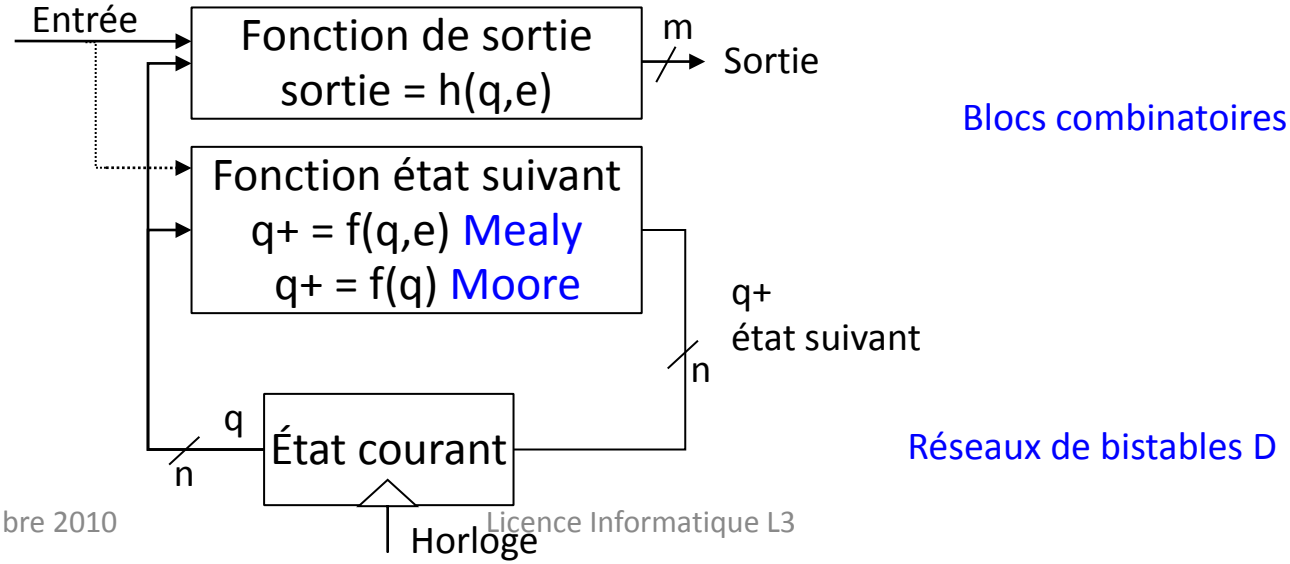
Université Nice Sophia Antipolis

Contrôle

Machines à états finis

Machines à états finis

- Les systèmes de contrôle cadencent les différents composants assemblés (e.g. multiplieur)
- On les décrits avec des machines à états finis
- Mealy et Moore ont le même pouvoir d'expression



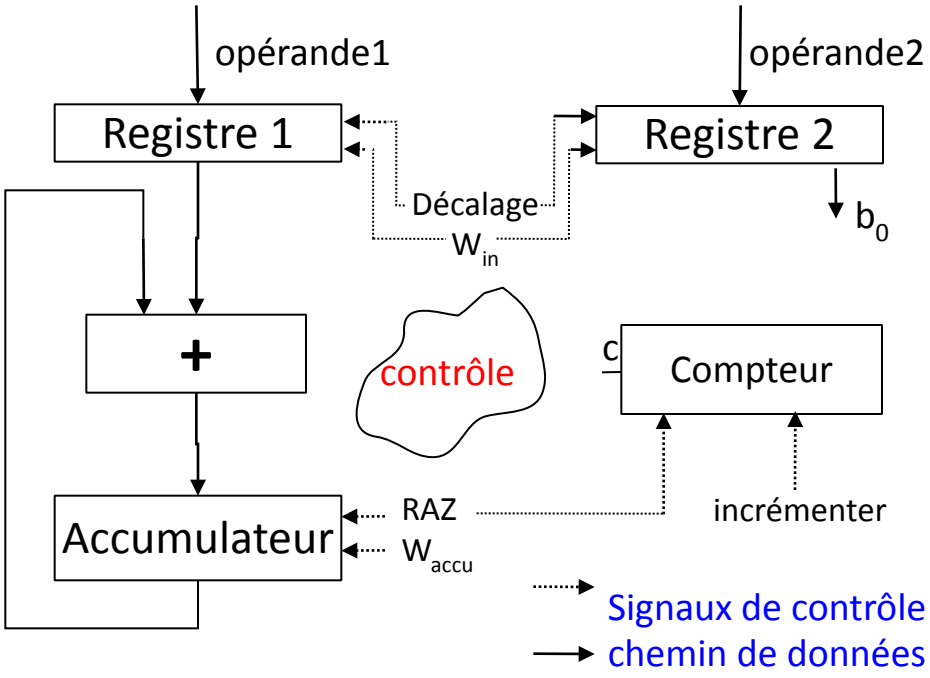
Blocs combinatoires

Réseaux de bistables D

Multiplication binaire non signée

```
  0 0 1 1
X
          0 1 1 1
  0 0 1 1
+
        0 0 1 1 0
  0 1 0 0 1
+
      0 0 1 1 0 0
  0 1 0 1 0 1
+ 0 0 0 0 0 0 0
  0 0 1 0 1 0 1
```

Résultats intermédiaires

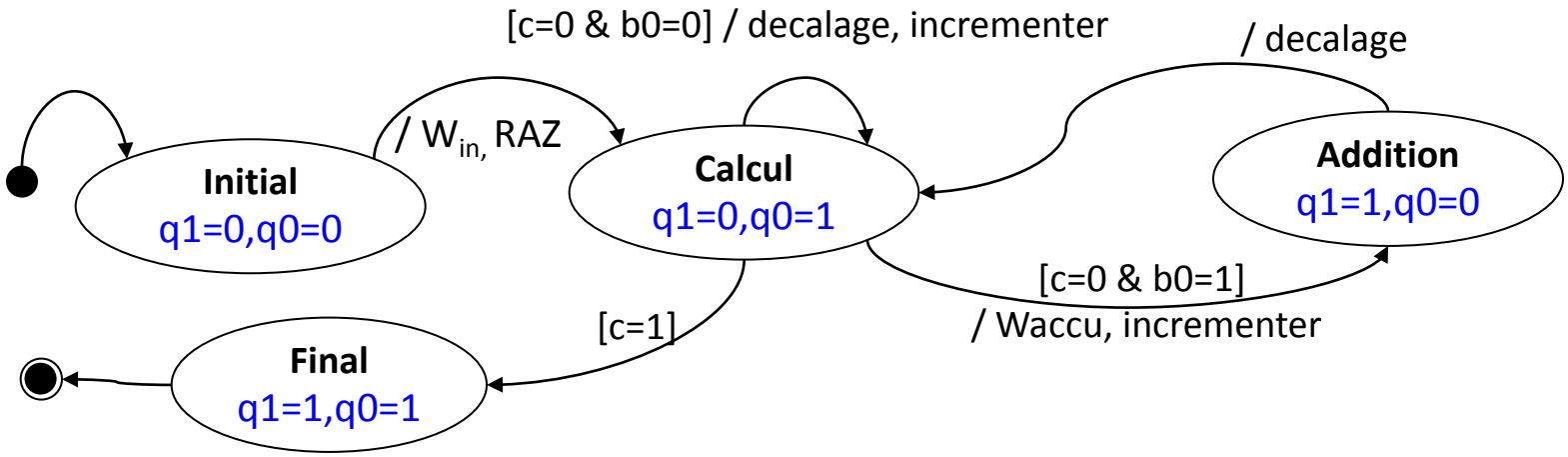


- Une multiplication n bits est équivalente à n additions
- On accumule les résultats intermédiaires dans un registre 2n-bit

Contrôle du multiplieur

- Entrées :
  - b_0 (bit de poids faible du registre 2) et c (compteur=n)
- Sorties :
  - W_{in} (valider les opérandes), W_{accu} (accumuler)
  - Décalage, incrémenter
- 4 états (2 bits q_1 q_0) :
  - Initial : mise à zéro accu et compteur, lire opérandes
  - Calcul : en cours de calcul
  - Addition : b0 vaut 1, on accumule
  - Final : le calcul est terminé

Machine à états pour le multiplieur



c	b_0	q_1	q_0		q_1^+	q_0^+	W_{in}	Raz	W_{accu}	Déc	Incr
-	-	0	0		0	1	1	1	0	0	0
0	0	0	1		0	1	0	0	0	1	1
0	1				1	0	0	0	1	0	1
1	0				1	1	0	0	0	0	0
1	1				1	1	0	0	0	0	0
-	-	1	0		0	1	0	0	0	1	0
-	-	1	1		1	1	0	0	0	0	0

Équations logiques

$$W_{in} = RAZ = \overline{q_0} \bullet \overline{q_1}$$

$$W_{accu} = \overline{c} \bullet b_0 \bullet \overline{q_1} \bullet q_0$$

$$Décalage = \overline{c} \bullet \overline{b_0} \bullet \overline{q_1} \bullet q_0 + q_1 \bullet \overline{q_0}$$

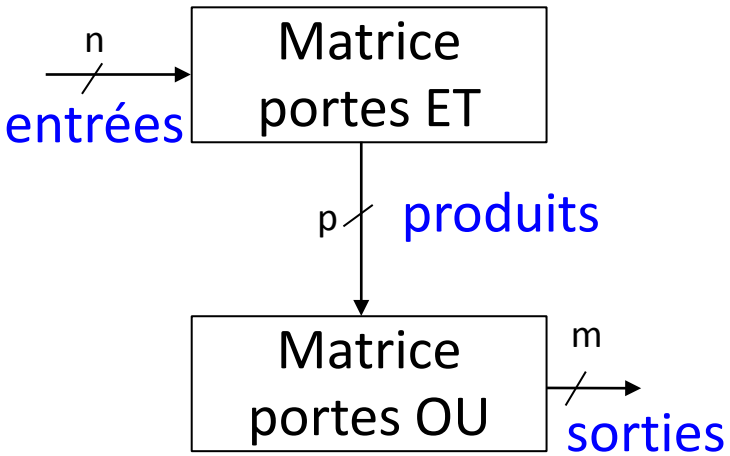
$$Incrémenter = c \bullet q_1 \bullet q_0$$

$$q_0^+ = \overline{W_{accu}}$$

$$q_1^+ = q_0 \bullet q_1 + (c + b_0) \bullet \overline{q_1} \bullet q_0$$

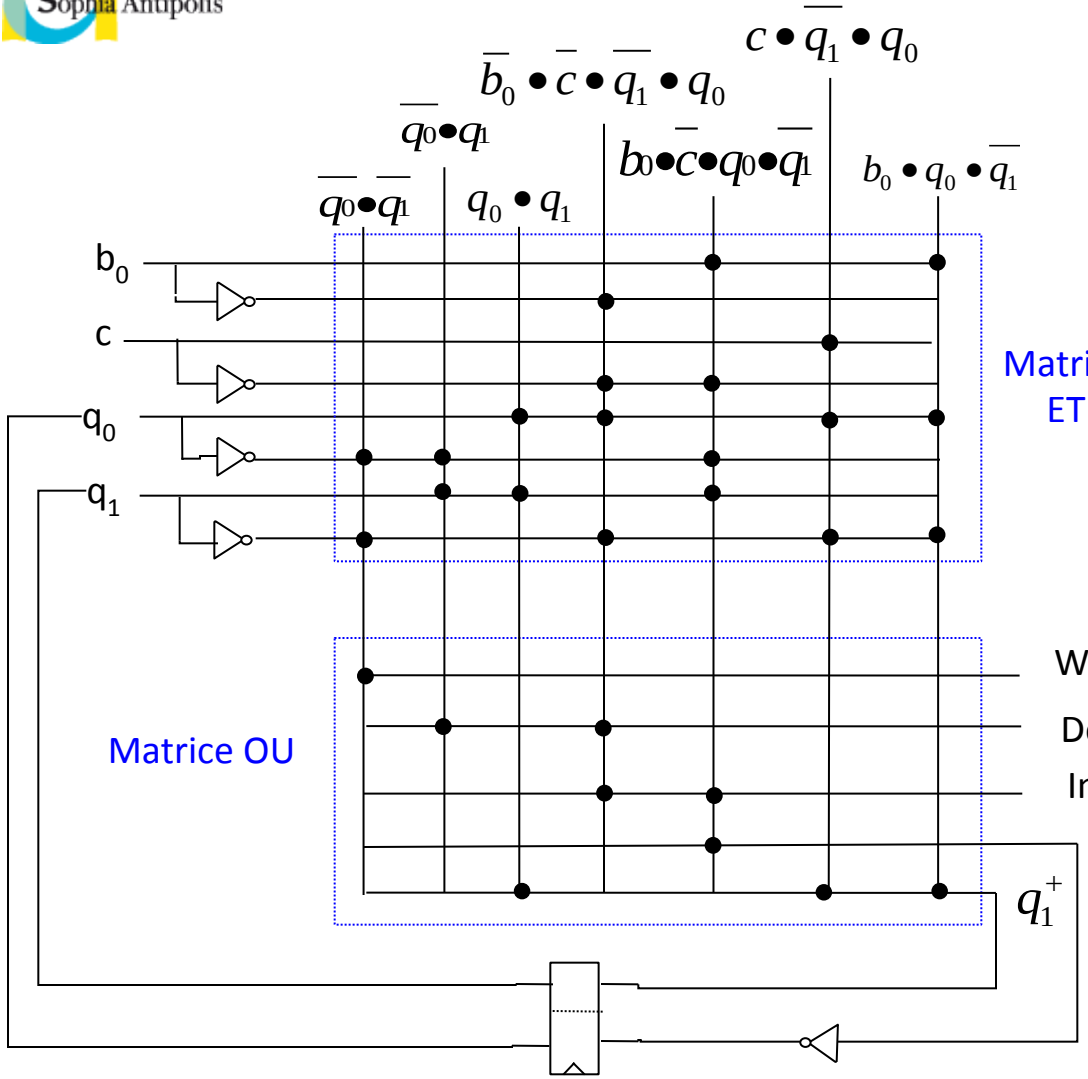
- Ces fonctions logiques peuvent être réalisées simplement avec des réseaux logiques programmables (RLP)

Réseau Logique Programmable



- 3 cas de "programmation"
  - Les 2 matrices sont figées
    - ROM = Mémoire morte
  - Matrice OU figée
    - PAL = Programmable Array Logic
  - Les 2 matrices sont programmables
    - PLA = Programmable Logic Array

Application au multiplieur



$$W_{in} = RAZ = \overline{q_0} \bullet \overline{q_1}$$

$$W_{accu} = \overline{c} \bullet b_0 \bullet \overline{q_1} \bullet q_0$$

$$Décalage = \overline{c} \bullet \overline{b_0} \bullet \overline{q_1} \bullet q_0 + q_1 \bullet \overline{q_0}$$

$$Incrémenter = c \bullet q_1 \bullet q_0$$

$$q_0^+ = \overline{W_{accu}}$$

$$q_1^+ = q_0 \bullet q_1 + (c + b_0) \bullet \overline{q_1} \bullet q_0$$

- marquer les intersections pour programmer le PLA