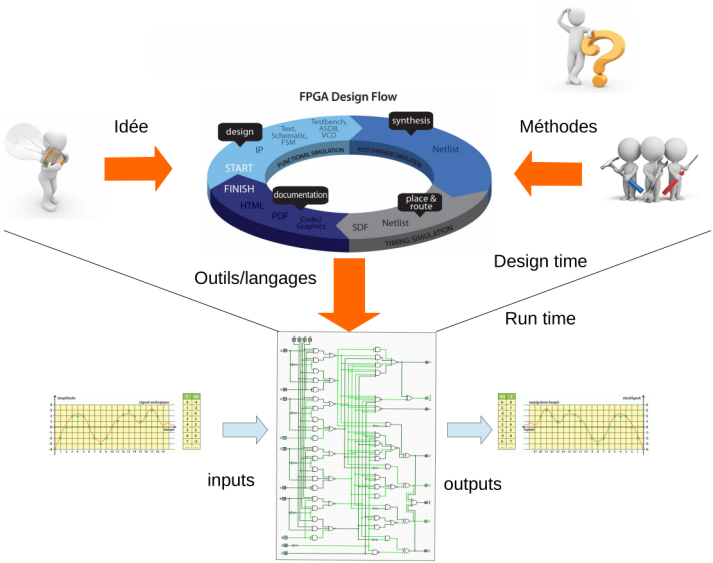
Introduction à l'Electronique Numérique

Jean-Christophe Le Lann

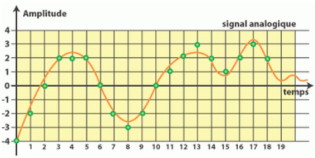
Cours 1A

14 septembre 2026

De l'idée au circuit



Numérisation

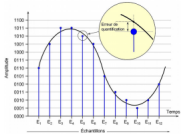


t	val
0	-4
1	-2
2	0
3	2
4	2
5	2
6	0
7	-2
...	...

Deux numérisations :
- Numérisation verticale : les données
- Numérisation horizontale : le temps

Maîtrise / reproductibilité des traitements

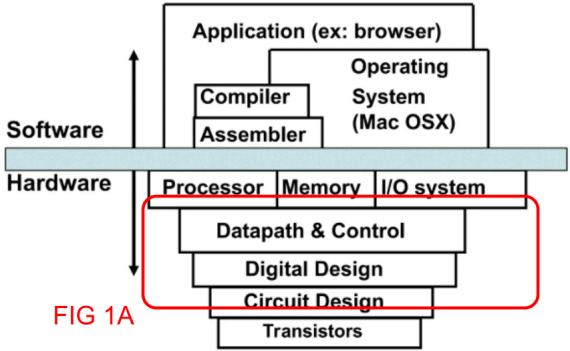
Nouvelles problématiques



Plan

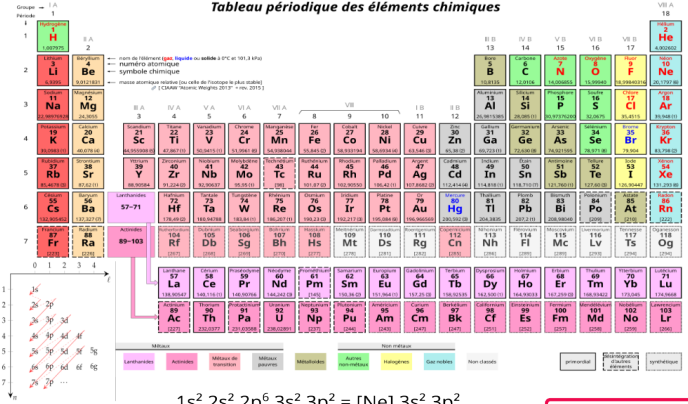
Objectifs

1. Comprendre les fondements de la conception des systèmes numériques :
   - Logique booléenne : combinatoire et séquentielle.
   - Calcul sur silicium, chemin de données
   - Machines d'états finis : contrôleurs / automates
   - Principe des Micro-architectures
   - Initiation à VHDL au niveau logique et structurel
2. Premiers pas vers les systèmes embarqués :
   - Processeurs, périphériques, communication



1– Physique des semi-conducteurs et micro-électronique : un aperçu

Le Silicium : élément chimique



$1s^2\ 2s^2\ 2p^6\ 3s^2\ 3p^2 = [Ne]\ 3s^2\ 3p^2$

14 28.086
Si
Silicon
$[Ne]\ 3s^2 3p^2$
Metalloid

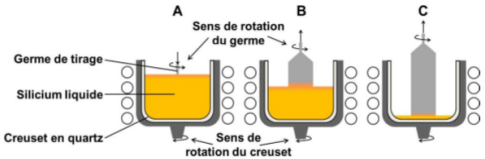
- **Couche K (n=1) :** 2 électrons ($1s^2$) → couche pleine.
- **Couche L (n=2) :** 8 électrons ($2s^2\ 2p^6$) → couche pleine (règle de l'octet).
- **Couche M (n=3) :** 4 électrons ($3s^2\ 3p^2$) → couche incomplète.

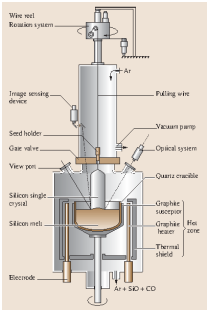
4 électrons de valence

Le Silicium : à partir de la Silice

1. **Extraction** : réduction carbothermique de la silice
   - Réaction chimique à 3000° : SiO2 + 2C $\rightarrow$ Si (liquide) + 2CO (gaz)
   - Source de carbone : charbon, coke,...
   - Pureté 98-99.5 % insuffisante pour l'Electronique (silicium métallurgique MG-Si)
2. **Purification** : Procédé **Siemens**
   - Hydrochloration du silicium métallurgique : trichlorosilane $SiHCl_3$ (gazeux)
   - Si + 3 HCl $\rightarrow$ SiHCl3 + H2
   - Distillation : chauffage à 1100°. Décomposition et obtention de "germes".
   - Pureté 99.9999999% (polycristallin) voire plus pour l'Electronique.
3. **Obtention d'un monocristal** : Procédé **Czokralski**
   - Fusion et germination.
   - Tirage et rotation.
   - Formation du lingot.

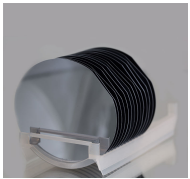
Le Silicium : méthode Czokralski





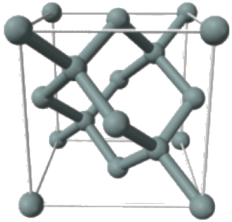
Machine ECM Greentech.

Lingot de Silicium monocristalin

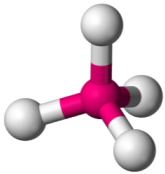


wafer

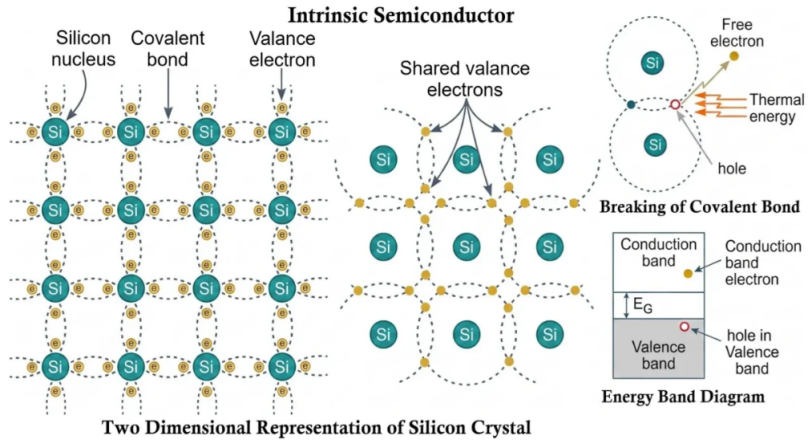
- Inter-pénétration de **deux** réseaux diamant Cubique face centrée (CFC)



- Second réseau décallé du premier d'un **quart** de la diagonale principale
- Le réseau de Bravais sous-jacent reste un réseau CFC unique, mais associé à un motif à deux atomes de silicium : (0,0,0) et (a/4,a/4,a/4)
- Chaque atome de silicium est lié à 4 autres atomes de silicium, qui forment un **tétraèdre**

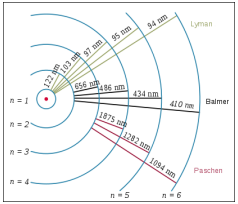


- L'angle entre les **paires de liaisons** est de **109°**

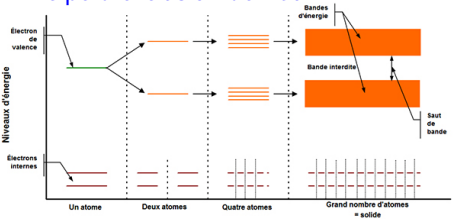


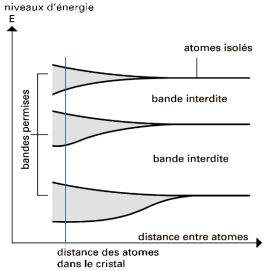
Attention : notez que cette abstraction 2D est éloignée de la réalité 3D

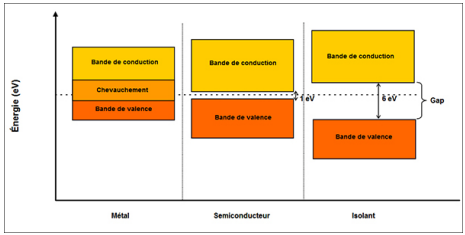
Le Silicium : théorie des bandes



Principe d'exclusion de Pauli

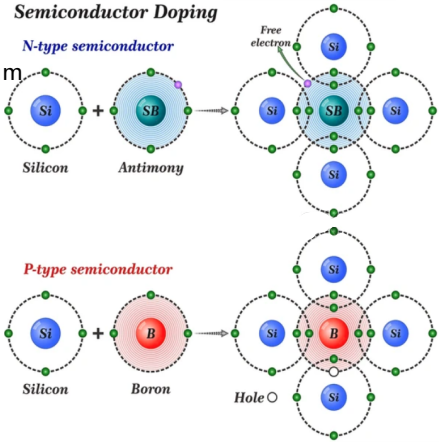






[https ://www.iutenligne.ne]

- On modifie **fortement** la conductivité du silicium par insertion contrôlée d'impuretés.
  - rapport d'environ 1 atome dopant pour 10^4 à 10^9 atomes de silicium
- Deux types d'impuretés : **donneurs** ou **accepteurs** d'électrons
- Deux types de déplacements de charges :
  - **courant d'électrons**
  - **courant de trous**



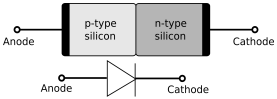
On dispose ainsi de 3 types de sicilium : intrinsèque, type N, type P.

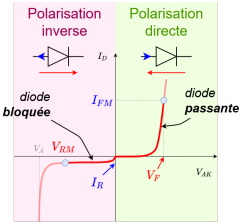
Jonction P-N : mise en contact de deux types de silicium

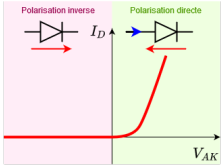
1. Mise en contact et diffusion
   - **Loi de Fick** : les porteurs majoritaires diffusent.
   - Les électrons de N vers P, les trous de P vers N
   - **Recombinaisons** au voisinage immédiat de l'interface.
2. La **zone de déplétion** (ZCE) : zone vidée de ses porteurs libres
   - côté N : des ions donneurs positifs non compensés
   - côté P : des ions accepteurs négatifs non compensés
   - ZCE : région dépourvue de porteurs mobiles mais chargée électriquement.
3. **Champ interne** et barrière de potentiel
   - un champ électrique interne $\vec{E}$, orienté de N vers P, qui s'oppose à la diffusion.
4. Équilibre dynamique :
   - Une barrière de potentiel V_0 s'établit (environ 0,7 V pour le silicium).
   - Le courant de diffusion (dû au gradient) équilibre exactement le courant de conduction (dû au champ électrique).
   - Le niveau de Fermi E_F devient continu à travers tout le matériau.

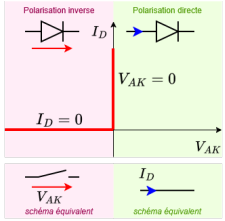
Jonction P-N : polarisation

1. Polarisation directe
   - + côté P, - côté N.
     - La tension externe s'oppose au champ interne.
     - la barrière de potentiel diminue. la ZCE se rétrécit.
     - la diffusion des majoritaires redevient dominante.
     - le courant croît exponentiellement avec V.
   - Loi de Shockley : $I = I_0(e^{V/V_T} - 1)$
2. Polarisation inverse
   - champ externe renforce le champ interne
   - la ZCE s'élargit
   - très faible courant



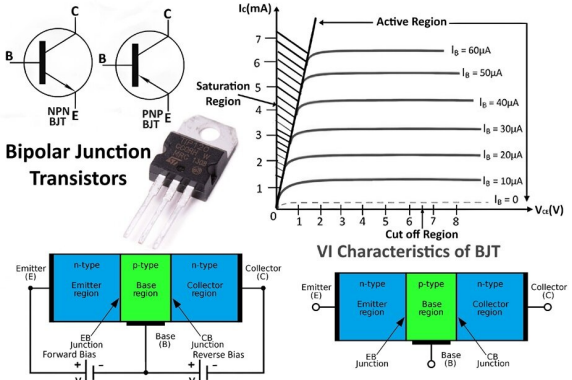






Transistor bipolaire : structure

- Deux jonctions PN dos-à-dos : type NPN ou PNP
- Trois régions : émetteur, base, collecteur
- Base fine et faiblement dopée
- Jonction émetteur-base polarisée en direct
- Jonction base-collecteur polarisée en inverse
- Effet transistor : le courant de base contrôle le courant de collecteur

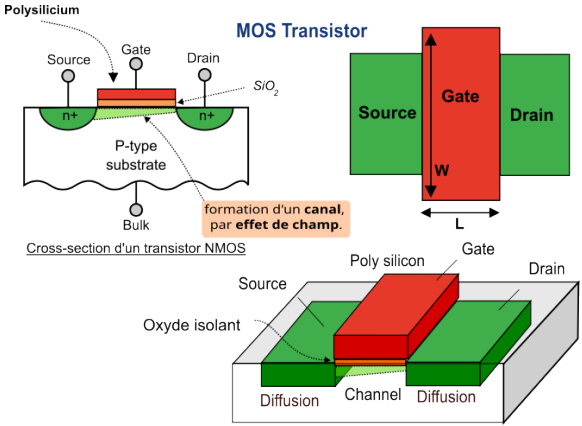


Transistor bipolaire : fonctionnement

- Gain en courant $\beta = I_C/I_B$ (typiquement 50 à 300)
- Régime actif : amplification (usage analogique)
- Régime saturé : interrupteur fermé (les 2 jonctions en direct)
- Régime bloqué : interrupteur ouvert (les 2 jonctions en inverse)
- Relation fondamentale : $I_E = I_B + I_C$
- Applications : amplification, commutation, logique TTL

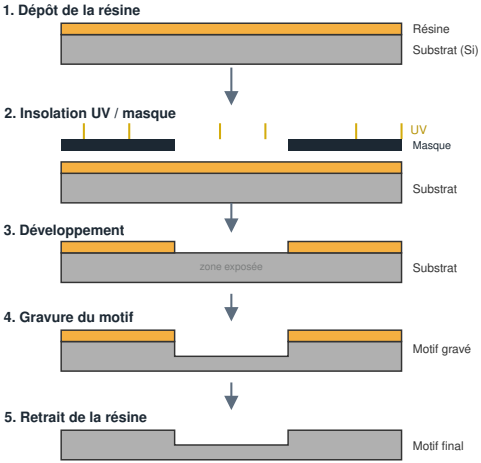
Transistors MOS (nmos)

- Transistor à **effet de champ** (FET) à canal N, constitué d'un substrat P, de deux zones N+ (Source et Drain) et d'une grille isolée par un oxyde (SiO_2).
- Commande **par la tension** grille-source V_{GS}
- Création un canal conducteur (ou non) d'électrons entre Source et Drain.
- Inconvénient : un inverseur NMOS utilise une résistance de pull-up.

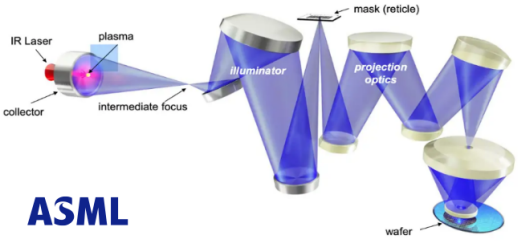


Le procédé de photolithographie

- **Principe** : transférer un motif géométrique d'un masque vers une résine photosensible déposée sur une plaquette (*wafer*)
- **Dépôt** : enduction centrifuge (*spin coating*) d'un film uniforme
- **Insolation** : exposition UV à travers un masque optique portant le motif
- **Développement** : dissolution sélective des zones exposées (résine positive) ou non exposées (négative)
- **Gravure** : transfert du motif dans le substrat (voie humide ou plasma)
- **Retrait** de la résine (*stripping*)
- Résolution limitée par la diffraction $\Rightarrow$ UV profond, immersion, EUV



ASML : le leader de la photo-lithographie



ASML

TWINSCAN EXE:5000

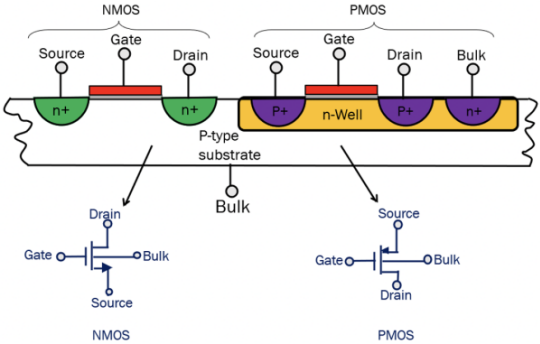
150M€

2– Technologie CMOS : Complementary Metal Oxyde Semiconductor

Paradigme dominant

Fondement de la microélectronique numérique actuelle

- Association de transistors **NMOS** et **PMOS** sur un même substrat.
- NMOS : conduit lorsque la grille est à V_{DD}
- PMOS : conduit lorsque la grille est à $0\,\mathrm{V}$
  - Se substitue à la résistance de pull-up du NMOS.
  - Montage **dual** : série $\rightleftarrows$ parallèle.
- Effet de champ : la **tension** de grille contrôle le canal.
- Isolation par **oxyde de grille** $\Rightarrow$ haute impédance d'entrée.



Physique

Caractérisation du comportement *analogique* du MOS.

1. Blocage $(V_{GS} - V_T \leq 0)$

$$I_D = 0$$

2. Zone linéaire $(0 < V_{DS} < V_{GS} - V_T)$

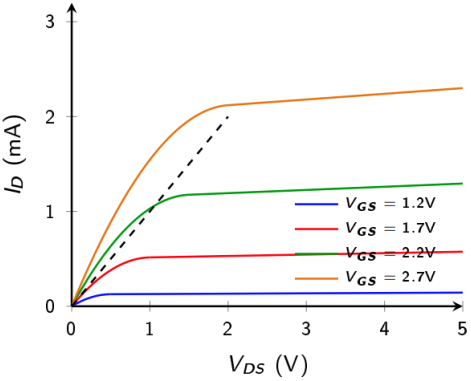
$$I_D = k_n \frac{W}{L}\left[(V_{GS}-V_T)V_{DS} - \frac{V_{DS}^2}{2}\right](1 + \lambda V_{DS})$$

3. Saturation $(V_{DS} \geq V_{GS} - V_T)$

$$I_D = \frac{k_n}{2}\frac{W}{L}(V_{GS} - V_T)^2\,(1 + \lambda V_{DS})$$

$k_n = \mu_n C_{ox}$ (transconductance) · V_T (seuil) · λ (modulation de canal) · W/L (géométrie)

Un seul jeu d'équations décrit tout le comportement, du blocage à la saturation.



Transistor NMOS

- Conduit (interrupteur fermé) si **Grille = 1**
- Bloqué si Grille = 0
- Transmet bien un **0** logique

Transistor PMOS

- Conduit (interrupteur fermé) si **Grille = 0**
- Bloqué si Grille = 1
- Transmet bien un **1** logique

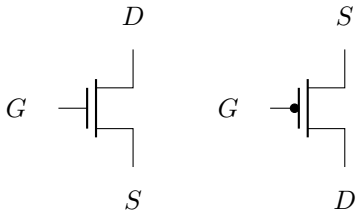


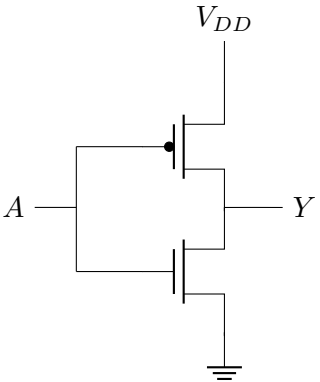
Table de fonctionnement simplifiée

Entrée grille	NMOS	PMOS
0	bloqué	passant
1	passant	bloqué

Idée fondatrice du CMOS

On associe systématiquement un réseau de NMOS (relié à la masse) et un réseau de PMOS *dual* (relié à l'alimentation V_{DD}) pour qu'à chaque instant **un seul** des deux réseaux conduise.

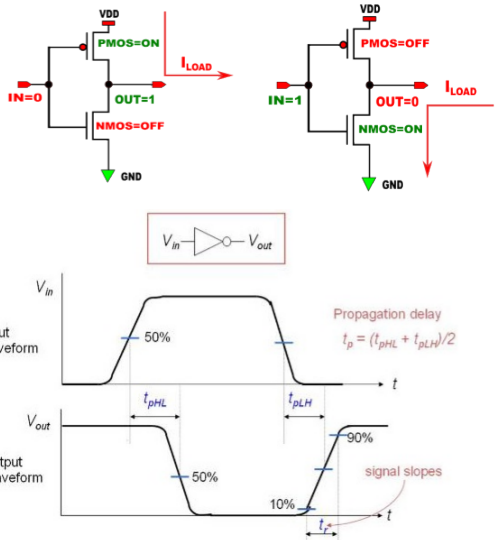
L'inverseur CMOS : la porte de base



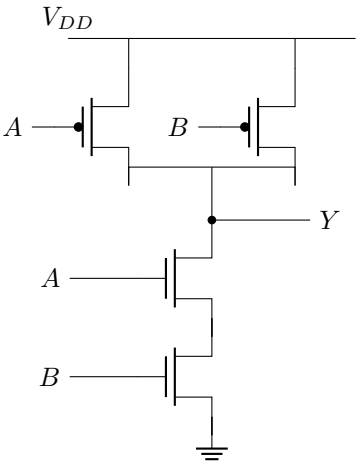
- Entrée A commune aux deux grilles
- PMOS en haut (côté V_{DD})
- NMOS en bas (côté masse)
- Si $A = 0$: PMOS passant, NMOS bloqué $\Rightarrow Y = 1$
- Si $A = 1$: PMOS bloqué, NMOS passant $\Rightarrow Y = 0$

$$Y = \overline{A}$$

L'inverseur CMOS : chronogramme



Porte NAND à 2 entrées : montage dual



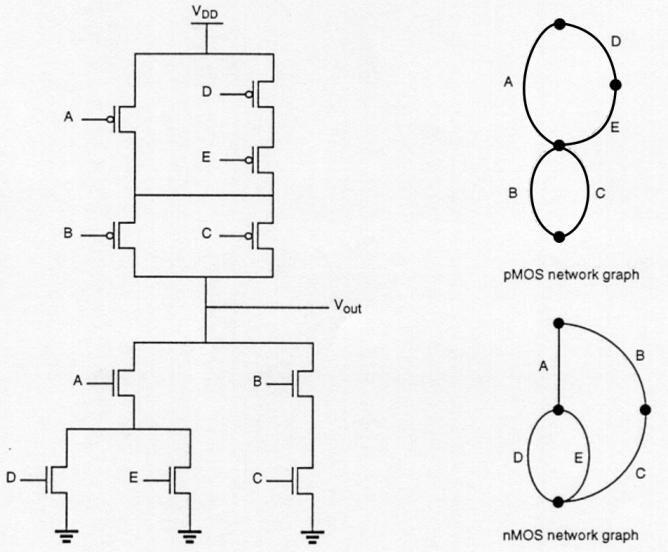
- NMOS A et B en **série** $\rightarrow$ conduit seulement si $A = 1$ ET $B = 1$
- PMOS A et B en **parallèle** (dual)
- $Y = 0$ uniquement si $A = B = 1$

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

Pourquoi ça marche : jamais de court-circuit

- En régime statique, un seul chemin conducteur existe entre V_{DD} et Y, ou entre Y et la masse — jamais les deux.
- Conséquence : **pas de consommation statique** (hors fuites), contrairement aux logiques NMOS seules (résistance de charge) ou bipolaires.
- La consommation du CMOS est essentiellement **dynamique** : elle a lieu lors des commutations (charge/décharge des capacités de sortie).

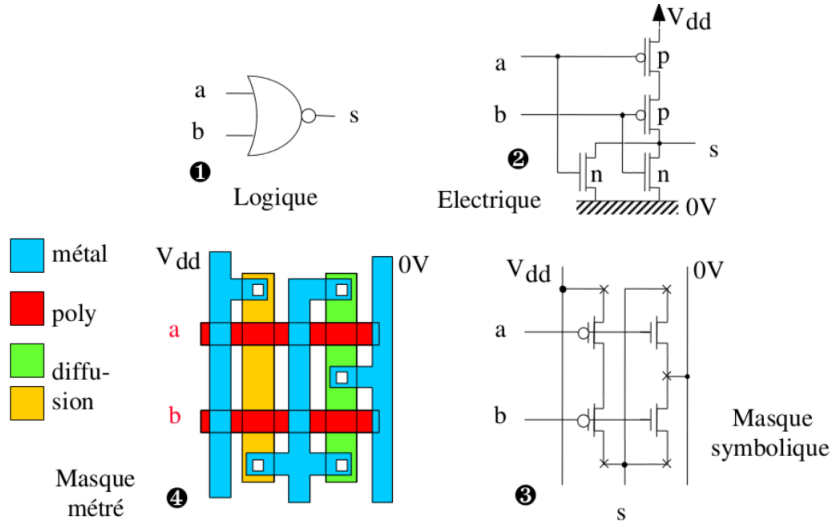
Dualité N-P en CMOS : autre exemple

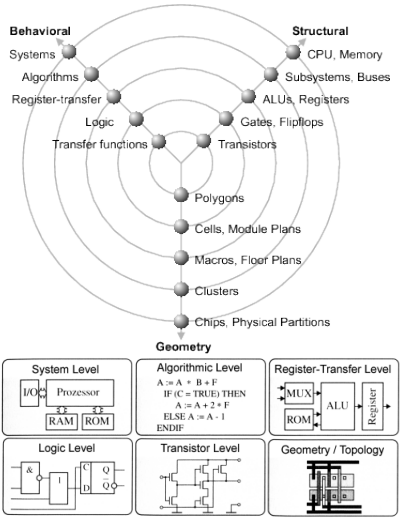


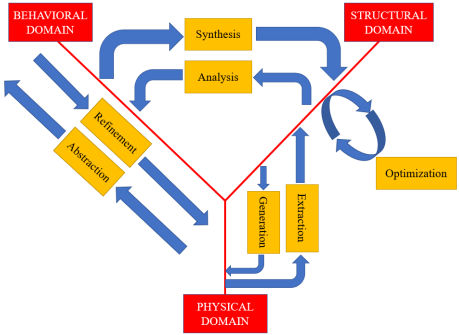
3– Montée en abstraction

Conception des Portes logiques : plusieurs niveaux d'abstraction

[Guyot TIMA 1995]



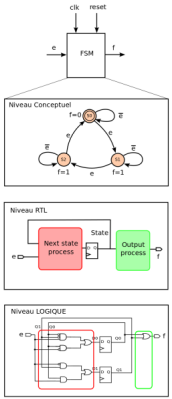

Behavioral
Structural
Systems
CPU, Memory
Algorithms
Subsystems, Buses
Register-transfer
ALUs, Registers
Logic
Gates, Flipflops
Transfer functions
Transistors
Polygons
Cells, Module Plans
Macros, Floor Plans
Clusters
Chips, Physical Partitions
Geometry
System Level
I/O
Prozessor
RAM
ROM
Algorithmic Level
A := A + B + F
IF (C = TRUE) THEN
A := A + 2 * F
ELSE A := A - 1
ENDIF
Register-Transfer Level
MUX
ALU
ROM
Register
Logic Level
&
1
C
D
Q
Q
Transistor Level
Geometry / Topology

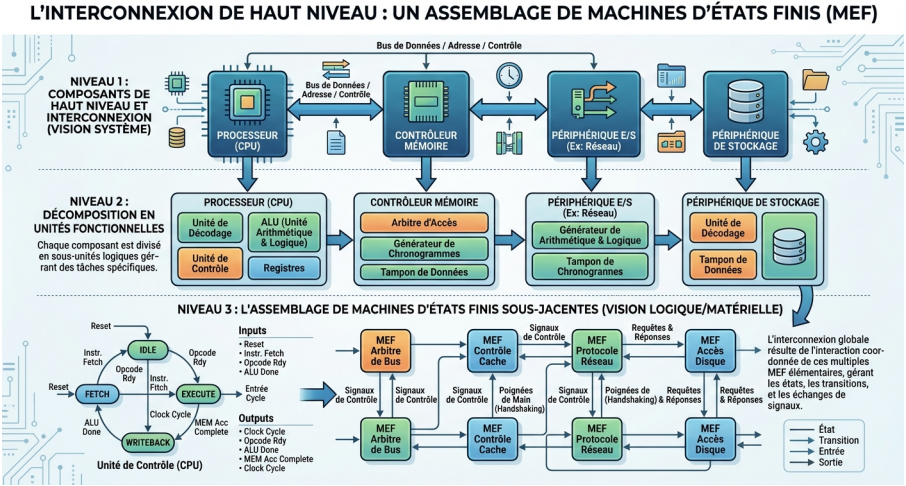

BEHAVIORAL DOMAIN
Synthesis
STRUCTURAL DOMAIN
Analysis
Refinement
Abstraction
Optimization
Generation
Extraction
PHYSICAL DOMAIN

Abstraction et Synthèse automatique

- Complexité des systèmes
- Recours à des langages spécifiques : HDL.
  - Description
  - Simulation (laboratoire virtuel)
  - Synthèse automatique
- ...à différents niveaux d'abstraction.

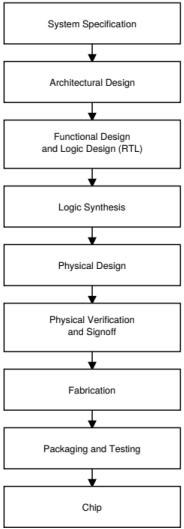
```
architecture RTL of FSM is
  type state_type is (S0,S1,S2);
  signal state,next_state : state_type;
begin

  -- description des bascules d'etat
  state: process(reset_n,clk)
  begin
    if reset_n='0' then
      state <= S0;
    elsif rising_edge(clk) then
      state <= next_state;
    end if;
  end process;

  -- transition 'function'
  next_state_logic : process(state,e)
  begin
    next_state <= state; --default
    case state is
    when S0 =>
      if e='1' then
        next_state <= S1;
      end if;
    when S1 =>
      if e='1' then
        next_state <= S2;
      end if;
    when S2 =>
      if e='1' then
        next_state <= S0;
      end if;
    when others =>
      null;
    end case;
  end process;

  -- output 'function' :
  output_logic : process(state)
  begin
    if state=S0 then
      f <='0';
    else
      f <= '1';
    end if;
  end process;

end RTL;
```



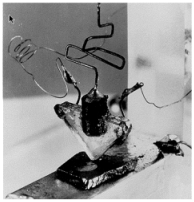

L'INTERCONNEXION DE HAUT NIVEAU : UN ASSEMBLAGE DE MACHINES D'ÉTATS FINIS (MEF)
Bus de Données / Adresse / Contrôle
NIVEAU 1 : COMPOSANTS DE HAUT NIVEAU ET INTERCONNEXION (VISION SYSTÈME)
PROCESSEUR (CPU)
CONTRÔLEUR MÉMOIRE
PÉRIPHÉRIQUE E/S (Ex: Réseau)
PÉRIPHÉRIQUE DE STOCKAGE
NIVEAU 2 : DÉCOMPOSITION EN UNITÉS FONCTIONNELLES
Chaque composant est divisé en sous-unités logiques gérant des tâches spécifiques.
Unité de Décodage
ALU (Unité Arithmétique & Logique)
Unité de Contrôle
Registres
Arbitre d'Accès
Générateur de Chronogrammes
Tampon de Données
Générateur de Arithmétique & Logique
Tampon de Chronogrammes
NIVEAU 3 : L'ASSEMBLAGE DE MACHINES D'ÉTATS FINIS SOUS-JACENTES (VISION LOGIQUE/MATÉRIELLE)
Reset
IDLE
FETCH
EXECUTE
WRITEBACK
Instr. Fetch
Opcode Rdy
ALU Done
Clock Cycle
MEM Acc Complete
Entrée Cycle
Inputs
• Reset
• Instr. Fetch
• Opcode Rdy
• ALU Done
Outputs
• Clock Cycle
• Opcode Rdy
• ALU Done
• MEM Acc Complete
• Clock Cycle
Unité de Contrôle (CPU)
MEF Arbitre de Bus
MEF Contrôle Cache
MEF Protocole Réseau
MEF Accès Disque
Signaux de Contrôle
Requêtes & Réponses
Poignées de Main (Handshaking)
Poignées de (Handshaking) & Réponses
Requêtes & Réponses
L'interconnexion globale résulte de l'interaction coordonnée de ces multiples MEF élémentaires, gérant les états, les transitions, et les échanges de signaux.
État
Transition
Entrée
Sortie

Electronic Design Automation (EDA) :

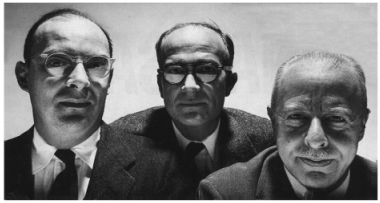
- **Spécification système** : besoins et fonctionnalités
- **Conception architecturale** : partitionnement HW/SW
- **Conception fonctionnelle et RTL** : description HDL (Verilog/VHDL)
- **Synthèse logique** : RTL $\rightarrow$ netlist de portes
- **Conception physique** : placement et routage
- **Vérification physique et signoff** : DRC, LVS, STA
- **Fabrication** : gravure sur wafer (silicium)
- **Packaging et test** : mise en boîtier, tests
- **Puce finale** : circuit intégré prêt à l'emploi



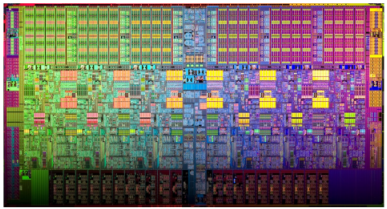
Du transistor (1947) au xeon d'Intel



23 décembre 1947 Bell laboratories



Bardeen, Shockley, Brattain

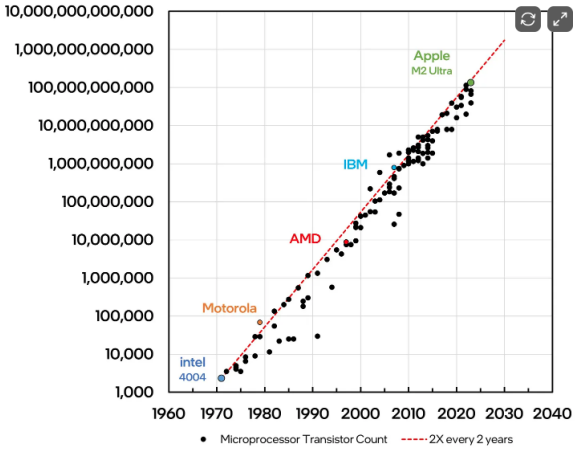


Environ 2 Milliards de transistors

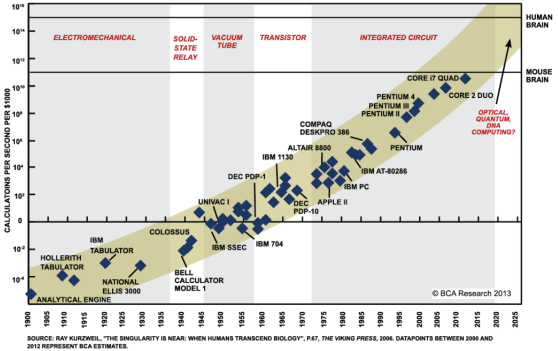
Intel XEON 5680 Core 6 (2010 – obsolete → Cf Xeon Phi) - gravure 32nm
6 coeurs (processeurs) + mémoire cache + contrôleur mémoire + interconnect

Jalons historiques : loi de Moore

- Doublement du nombre de transistors tous les 18 mois.



- Perspective historique plus large.



1971 : Intel 4004

Le cas des mémoires :

1956 1990-2011 2015

Pile h=80km !

Pile h=80m

80 000 x

3Mb

512 Gb

« There is room at the bottom » R.P.Feynman

4– Représentations numériques

Base de numération

- On parle de *base* ou *radix* r.
  - Utilisation des *symboles* de 0 à $r-1$.
  - Notation *positionnelle*
  - Les entiers se décomposent à l'aide des coefficients des puissances de r.
- Base 10 :
  - $142 = 1 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$
- Base 10, nombres décimaux :
  - $142.34 = 1 \times 10^2 + 4 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 4 \times 10^{-2}$
  - Nombre décimal $\frac{a}{10^n} \rightarrow$ nombre fini de symboles.
  - Attention : tous les nombres à virgule ne sont pas décimaux !
  - Attention : lorsqu'on change de base, le nombre de coefficients nécessaires, pour une précision donnée, varie.
    - Exemple : 0.2 ne se représente pas avec un nombre fini de `bits` dans la base 2 !

Notations non positionnelles : trois exemples

Chiffres romains *(additif, symboles à valeur fixe)*

I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000

Exemple : MCMXCIV = 1000 + (1000-100) + (100-10) + (5-1) = **1994**

Système unaire *(additif pur, une marque par unité)*

||||| = 5 ||||||| = 7 *(ou groupes de 5)*

Hiéroglyphes égyptiens *(additif, un symbole par puissance de 10)*

| = 1 ∩ = 10

𓎆 = 100 ⊙ = 1000

24 s'écrit : deux « anses » + quatre « traits » — **l'ordre n'importe pas**

Dans tous ces systèmes, la position d'un symbole ne modifie jamais sa valeur.

La formule générale pour la base 10 est bien entendu :

$$x = \sum_{i=-\infty}^{i=+\infty} c_i.10^i, c_i \in \{0, \ldots, 9\}$$

et se généralise :

$$\forall r, x = \sum_{i=-\infty}^{i=+\infty} c_{i,r}.r^i, c_{i,r} \in \{0, \ldots, (r-1)\}$$

Lors des changements de base, il faut une notation pour préciser la base de représentation.

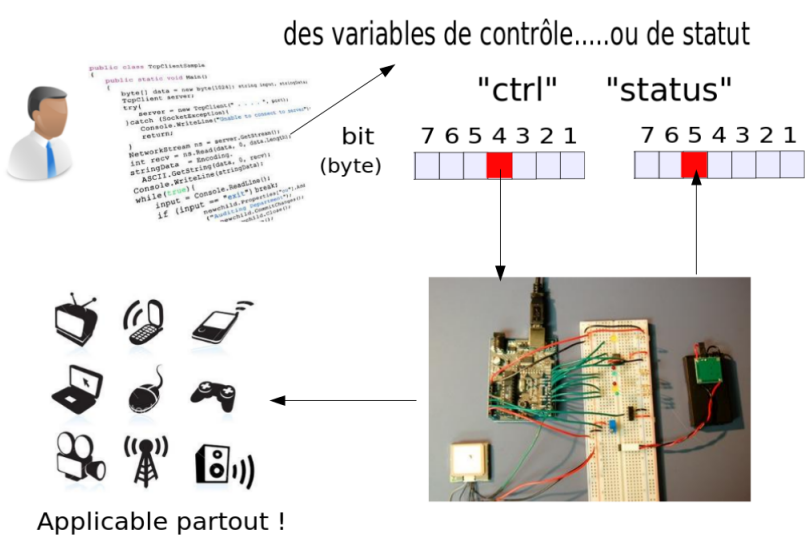
- Par exemple : $142_{10} = 10001110_2$
- En logiciel : $142_{10} = \texttt{0b10001110} = \texttt{0x8e} = \texttt{0o216}$

Compter en binaire : convertir par table

Decimal count	Binary count				
	16s	8s	4s	2s	1s
0					0
1					1
2				1	0
3				1	1
4			1	0	0
5			1	0	1
6			1	1	0
7			1	1	1
8		1	0	0	0
9		1	0	0	1
10		1	0	1	0
11		1	0	1	1
12		1	1	0	0
13		1	1	0	1
14		1	1	1	0
15		1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1
	2^4	2^3	2^2	2^1	2^0
	Powers of 2				

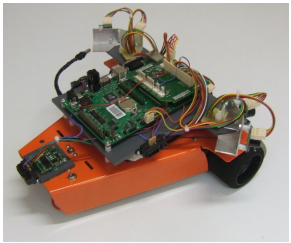
Décimal	Binaire	Octal	Hexadécimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

A quoi ça sert ?



A quoi ça sert ?

SPACIETA Infinity, Août 2012 Biscarosse



JOG : conçu à l'ENSTA Bretagne pour l'enseignement 1A.

Conversions entre bases : par table

$$101101_2 = 10\_1101_2 = 0010\_1101_2 = 2D_{16} = \mathbf{0x2D}$$

Dans le cas de nombres fractionnaires, on rajoute des 0 à droite de la partie fractionnaire. Ainsi :

$$\text{le nombre } 1011.01_2 = 1011.0100_2 = B.4_{16}$$

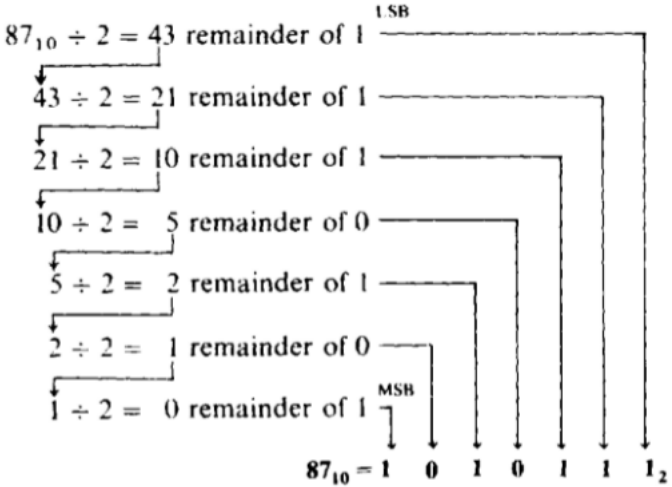
On procède de même pour la traduction octale.

$$(10110001101011.1111)_2 = (101\ 110\ 001\ 101\ 011.111\ 100)_2 = (36153.74)_8$$

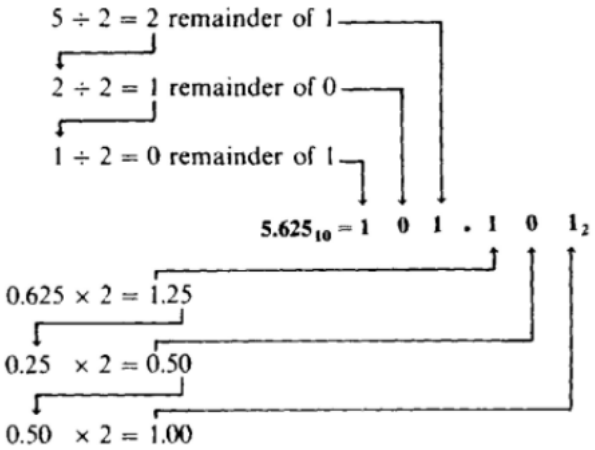
4	7	.	F	E
↓	↓		↓	↓
0100	0111	.	1111	1110

$$47.\text{FE}_{16} \approx 1000111.1111111_2$$

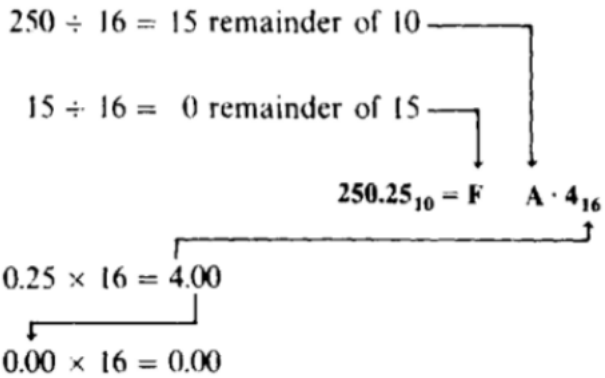
Conversions entre bases : par calcul



Conversions entre bases : par calcul



Conversions entre bases : par calcul



Autres codes binaires : BCD décimal codé binaire

Chiffre	Quartet	Chiffre	Quartet
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Chaque chiffre de 0 à 9 est ici représenté par 4 bits.
Ceci simplifie la conversion : elle revient à étudier la juxtaposition des bits

Ex : 123 = 0001 0010 0011

Autres codes binaires : code de gray ou binaire réfléchi

décimal	binaire classique	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

$0 \rightarrow 0000$

$1 \rightarrow 0001$

$0 \rightarrow 0000$

$1 \rightarrow 0001$

$- - - - -$

$2 \rightarrow 0011$

$3 \rightarrow 0010$

Il existe un algorithme pour passer d'un nombre x à l'autre $x+1$:

- on calcule le nombre de 1 dans x. On inverse le dernier bit de x quand ce nombre de 1 est pair.
- si le nombre de 1 est impair, on inverse le bit à gauche du 1 qui est le plus à droite.

Soit un nombre W de bits et soit $\vec{x} = [x_{w-1}, x_{w-2}, \ldots, x_0]$ de bits. Nous avons vu précédemment (sans la nommer) que la simple fonction :

$$B2U_w(\vec{x}) \doteq \sum_{i=0}^{w-1} x_i 2^i$$

définit la valeur non-signée ("unsigned") de $\vec{x}$, dans $\mathbb{N}$. Les valeurs limites de cette fonction sont :

$$\begin{cases} UMax_w = \sum_{i=0}^{w-1} 2^i = 2^w - 1 \\ UMmin_w = \sum_{i=0}^{w-1} 0 = 0 \end{cases}$$

On a donc :

$$B2U_w : \{0,1\}^w \to \{0, \ldots, 2^w - 1\}$$

De manière similaire, on définit désormais la fonction qui permet de calculer la valeur du vecteur $\vec{x}$ dans le cas d'une valeur signée, c'est-à-dire négative ou positive, **en complément à 2**.

$$B2T_w(\vec{x}) \doteq -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i$$

Désormais, le bit de poids fort, le plus à gauche, contribue de manière négative à la somme totale. Précisons également que le 'T' vient de "two's complement". Calculons les valeurs min et max de cette fonction. Pour $\vec{x} = [10\ldots0]$ et $\vec{x} = [01\ldots1]$ respectivement, on trouve :

$$\begin{cases} TMin_w = -2^{w-1} \\ TMax_w = \sum_{i=0}^{w-2} 2^i = 2^{w-1} - 1 \end{cases}$$

On a donc :

$$B2U_w : \{0,1\}^w \to \{-2^{w-1}, \ldots, 2^{w-1} - 1\}$$

On pourra également dénoter cette fonction $C_{2,w}(\vec{x})$.

Complément à 2 sur w bits

signé	bit vector	non-signé
-4	100	4
-3	101	5
-2	110	6
-1	111	7
0	000	0
1	001	1
2	010	2
3	011	3

Le complément à 1 d'un nombre binaire consiste à simplement inverser tous ses bits. Le complément à 2 d'un nombre binaire s'obtient ainsi :

- On parcourt le nombre en partant de la droite (bit de poids faible).
- On recopie tel quel tous les bits tant qu'on n'a pas rencontré le premier 1.
- On recopie ce premier 1.
- On inverse tous les bits suivants (à gauche de ce 1).

Il existe toutefois une autre méthode, qui se retient plus facilement (peut-être). Elle consiste en une formule simple :

$$-X = \overline{X} + 1$$

Addition classique
(et non « OU » binaire)

Attention ! ! ! !
-134 nécessite 9 bits et non 8

La représentation en magnitude signée est simplement :

$$B2S_w(\vec{x}) \doteq (-1)^{x_{w-1}} \sum_{i=0}^{w-2} x_i 2^i$$

Il existe également une représentation des nombres en complément à 1, très légèrement différente du complément à 2.

$$B2T_w(\vec{x}) \doteq -x_{w-1}(2^{w-1} - 1) + \sum_{i=0}^{w-2} x_i 2^i$$

Ces représentations proposent 2 encodages pour 0 !

Dans le cas de la magnitude signée :
$+0 = 0 \ldots .0$ et $-0 = 10 \ldots 0$

Nombres négatifs : résumé

Signed decimal	8-bit 2's compl.
$+127$	**0111 1111**
$+126$	**0111 1110**
$+125$	**0111 1101**
⋮	⋮
$+1$	**0000 0001**
$+0$	**0000 0000**
-1	**1111 1111**
-2	**1111 1110**
⋮	⋮
-125	**1000 0011**
-126	**1000 0010**
-127	**1000 0001**
-128	**1000 0000**

$$-X = \overline{X} + 1$$

$-125\ ? =$

$125 \quad = \quad 111\,1101$

$/125 \quad = \quad 1\,000\,0010$

$/125 + 1 = 1\,000\,0011$

Noter que les nombres en complément à 2 sont négatifs s'ils présentent un MSB à 1 !
125 se représente sur 7 bits, mais pas -125 : il faut 8 bits

Exemple : -3 sur 3 bits

- $3_{10} = 011_2$
- $-3_{10} = /3_{10} + 1 = 100_2 + 1 = 101_2 = 5_{10}$????
  - 5 ne peut pas être représenté sur 3 bits, si on y représente également les nombres négatifs.

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

Pour un nombre donné de bits,
il faut s'entendre sur le fait
que la représentation d'un nombre est signée ou non

Addition : classique

	Addition
calcul	0110 +0011
retenus	11
résultat	1001

Soustraction : x - y

1. on additionne x et le complément à 2 de y
2. si le résultat présente une retenue finale, on l'oublie !
3. s'il n'y a pas de retenue finale, on prend le complément à 2 du résultat et on place un signe "-" devant le nombre.

exemple : $x = 1101100_2$ et $y = 1011011_2$

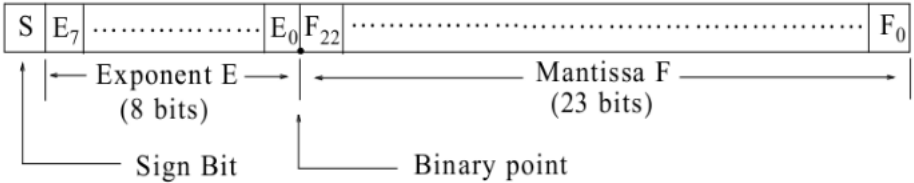
$$1101100 + 0100101 = 1\,0010001$$

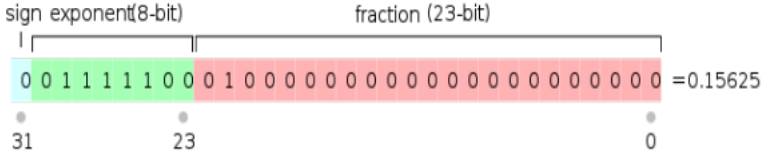
Le résultat est donc $x - y = 0010001_2$

Nombre en virgule fixe : ou *fixed point*

- Déjà vu précédemment...
- Le calculateur numérique devra connaître la position de la virgule, fixée une bonne fois pour toute.
- Impact sur les opérations classiques :
  - Le résultat de l'addition hérite de la position de la virgule identique à celle dans les opérandes
  - Cela est changé pour les autres opérations.

Nombres en virgule flottante : norme ieee754





Exo : calculer le nombre max et min représentable

- L'exposant est biaisé (en excédent) à 127. C'est-à-dire que pour coder un exposant e, on code le nombre $e + 127$.
- Les exposants représentables vont de -126 à 127 (les valeurs 00000000 et 11111111 sont réservées pour des cas exceptionnels)
- Les nombres sont normalisés, c'est-à-dire que la mantisse est entre 1 et 2.

- Si $e = 11111111_2$ et $m = 0$, le nombre représenté est $+$ ou $-$ l'infini. Les opérations sur l'infini sont gérées correctement.
- Si $e = 11111111_2$ et $m \neq 0$, la représentation est considérée comme n'étant pas un nombre (NaN : Not a Number).
- Si $e = 0$ et $m = 0$ on représente le nombre $+0$ ou -0 selon le signe.
- Si $e = 0$ et $m \neq 0$ on dit que le nombre est dénormalisé. C'est-à-dire que la mantisse est écrite sans le 1 implicite.

Exemple de conversion vers ieee 754

Nombre à encoder : $-12{,}375$

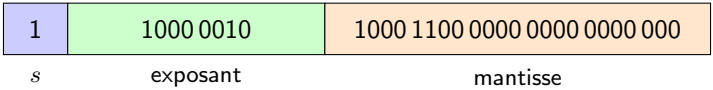
1. **Signe** : le nombre est négatif $\Rightarrow s = 1$
2. **Partie entière** : $12 = 1100_2$
3. **Partie fractionnaire** : $0{,}375$

$$0{,}375 \times 2 = 0{,}75 \to 0 \qquad 0{,}75 \times 2 = 1{,}5 \to 1 \qquad 0{,}5 \times 2 = 1{,}0 \to 1$$

$$\Rightarrow \; 12{,}375 = 1100{,}011_2$$

4. **Normalisation** : $1100{,}011_2 = 1{,}100011_2 \times 2^3$
   - Mantisse : $m = 100011$
   - Exposant réel : $e = 3$
   - Exposant biaisé : $E = 3 + 127 = 130 = 10000010_2$

Étape 3 : assemblage des 32 bits



Résultat final

$$-12{,}375 \longrightarrow \underbrace{1}_{\text{signe}}\ \underbrace{10000010}_{\text{exposant}}\ \underbrace{10001100000000000000000}_{\text{mantisse}}$$

$$\boxed{\texttt{0xC1460000}} \quad \text{(hexadécimal)}$$

Vérification : $(-1)^1 \times 1{,}100011_2 \times 2^{130-127} = -12{,}375$ ✓

IEEE 754 : l'addition n'est pas commutative !

Avertissement

En arithmétique flottante IEEE 754, à cause des **arrondis**, l'addition **n'est pas associative**, et dans certains cas elle peut même sembler **non commutative** lorsqu'on enchaîne plusieurs opérations ou qu'on change l'ordre des termes.

Cause principale : quand on additionne deux nombres d'ordres de grandeur très différents, le plus petit peut être *entièrement absorbé* par l'arrondi du plus grand (*absorption / swamping*).

Exemple classique en double précision

$$a = 1{,}0 \qquad b = 2^{-53} \qquad c = 2^{-53}$$

$$(a + b) + c \neq a + (b + c)$$

L'ordre des opérations change le résultat !

Algorithme matériel (simplifié) pour $x + y$:

1. **Aligner les exposants** : on réécrit le plus petit exposant pour qu'il coïncide avec le plus grand. Les bits de sa mantisse sont **décalés vers la droite** — et les bits qui « sortent » sont **perdus**.
2. **Additionner les mantisses** (avec le bit implicite).
3. **Renormaliser** le résultat si nécessaire.
4. **Arrondir** au format (au pair, par défaut).

Le point critique

C'est à l'étape **1** que se joue tout le drame : si les exposants diffèrent de plus de p bits (où p est la précision), le plus petit nombre est **totalement sacrifié** au profit du plus grand.

Hypothèses : double précision, $\varepsilon_{\text{mach}} = 2^{-52}$. Le plus petit incrément représentable autour de $1{,}0$ est $\text{ulp}(1) = 2^{-52}$.

Calcul de $(a + b) + c$

$$a + b = 1{,}0 + 2^{-53}$$

Or $2^{-53} = \frac{1}{2}\,\text{ulp}(1)$, donc l'arrondi (au pair) donne :

$$a + b = 1{,}0$$

Puis :

$$(a + b) + c = 1{,}0$$

Calcul de $a + (b + c)$

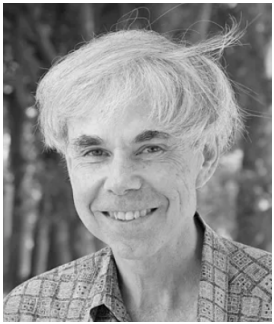
$$b + c = 2^{-53} + 2^{-53} = 2^{-52}$$

Cet incrément vaut exactement $\text{ulp}(1)$, donc il est représentable :

$$a + (b + c) = 1{,}0 + 2^{-52}$$

$$(a + b) + c = 1{,}0 \qquad \neq \qquad a + (b + c) = 1{,}0 + 2^{-52}$$

Douglas Hofstadter et les « faux » nombres réels



Douglas Hofstadter (1945–)
Gödel, Escher, Bach
Prix Pulitzer 1980

« En informatique, il n'y a pas de nombres réels. Il n'y a que des nombres flottants, et ils sont à peu près aussi réels que les personnages de dessins animés. »

IEEE 754 ne code qu'un **ensemble fini** de valeurs, et chaque opération introduit un **arrondi**. Pourtant, cette arithmétique approximative pilote avions, réacteurs et ponts : la fiabilité vient des **algorithmes**, pas des nombres.

Représentation des données symboliques

Certaines données ne sont pas numériques

- Enumérations, collections : couleurs,alphabets,...
- Ou compositions de données numériques

Un encodage de ces données est nécessaire.

Enumération

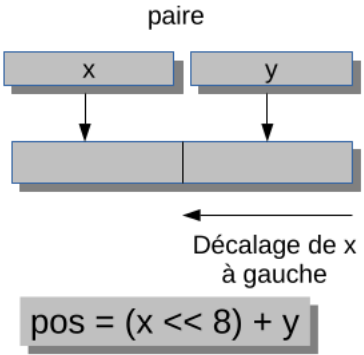
bleu $\rightarrow 0 \rightarrow 000$

blanc $\rightarrow 1 \rightarrow 001$

rouge $\rightarrow 2 \rightarrow 010$

vert $\rightarrow 3 \rightarrow 011$

noir $\rightarrow 4 \rightarrow 100$



- Encodage des caractères usuels

Bits b_4 b_3 b_2 b_1	Column → / Row ↓	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FC	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

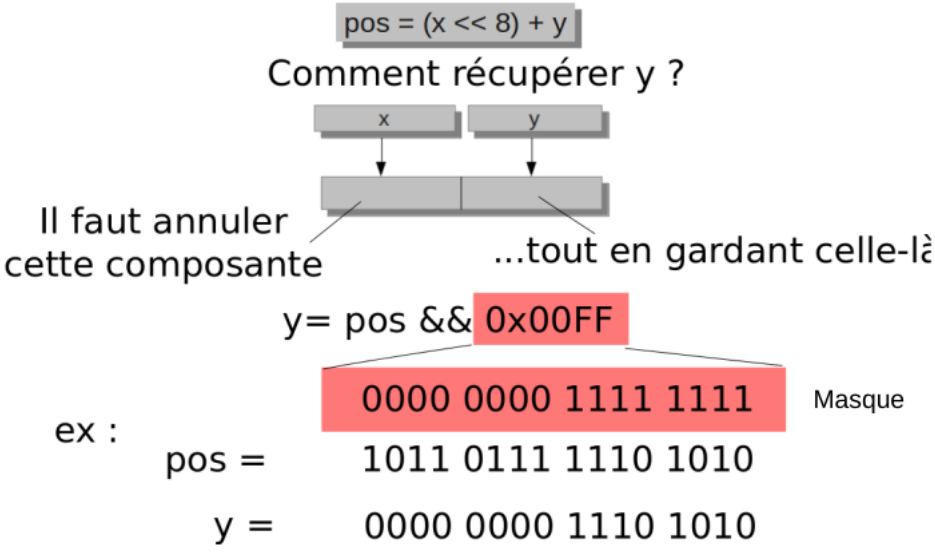
ASCII Art

Définition du nombre d'octets utilisés

Représentation binaire UTF-8	Signification
0xxxxxxx	1 octet codant 1 à 7 bits
110xxxxx **10**xxxxxx	2 octets codant 8 à 11 bits
1110xxxx **10**xxxxxx **10**xxxxxx	3 octets codant 12 à 16 bits
11110xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx	4 octets codant 17 à 21 bits

Ce principe pourrait être étendu jusqu'à huit octets pour un seul point de code, mais UTF-8 pose la limite à quatre[1].

Character		Binary code point	Binary UTF-8	Hexadecimal UTF-8
$	U+0024	0100100	00100100	24
¢	U+00A2	000 10100010	11000010 10100010	C2 A2
€	U+20AC	00100000 10101100	11100010 10000010 10101100	E2 82 AC
𤭢	U+24B62	00010 01001011 01100010	11110000 10100100 10101101 10100010	F0 A4 AD A2


pos = (x << 8) + y
Comment récupérer y ?
x
y
Il faut annuler
cette composante
...tout en gardant celle-là
y= pos && 0x00FF
0000 0000 1111 1111
Masque
ex :
pos = 1011 0111 1110 1010
y = 0000 0000 1110 1010

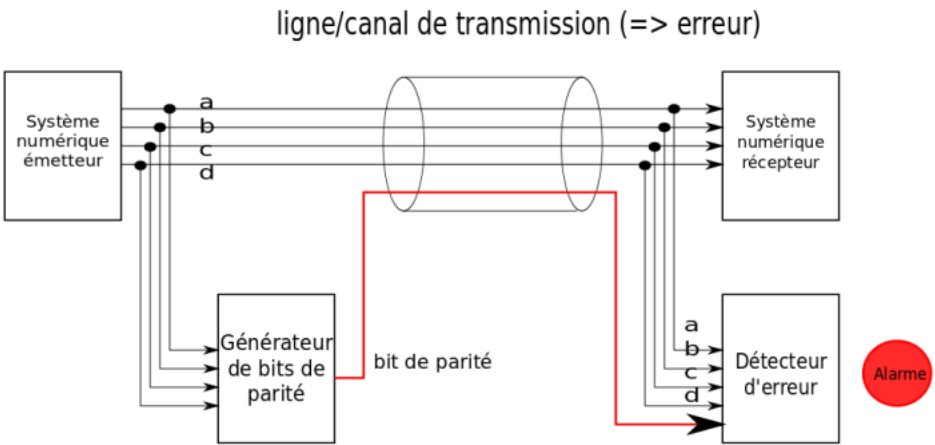
Manipulations binaires *bitwise* traditionnelles

Listing 2.1– Quelques manipulations de bits en C

```
a = a | 0x4; /* mise a 1 du bit 2 de la variable a*/
a |=0x4; /* idem*/
b &= ~(0x4) /* mise a 0 du bit 2 */
b &= ~(1 << 2) /* idem, mais plus explicite*/
c ^= ~(1 << 5) /* inversion du bit 5*/
e >>=2 /* division de e par 4 */
```

Attention : a sera un entier ! On ne verra pas explicitement ses 0 et 1 binaires !

Essentiel pour la programmation des dispositifs embarqués


ligne/canal de transmission (=> erreur)
Système numérique émetteur
a
b
c
d
Système numérique récepteur
Générateur de bits de parité
bit de parité
a
b
c
d
Détecteur d'erreur
Alarme

Codes détecteurs d'erreurs

Entrées	Sortie
D C B A	P
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	0
0 1 0 0	1
0 1 0 1	0
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	0
1 1 0 1	1
1 1 1 1	0
mot	bit de parité

code

La sortie est fonction des entrées, et des entrées seulement.

C'est une fonction combinatoire

Une révolution née à Brest

*En 1993, Claude Berrou et Alain Glavieux (ENST/IMT Atlantique) bouleversent la théorie de l'information en s'approchant de la **limite de Shannon**, que les experts jugeaient inatteignable depuis 1948.*

Les **turbocodes** reposent sur un principe itératif : deux décodeurs élémentaires s'échangent des « croyances » jusqu'à converger vers le message original.

Aujourd'hui : 3G/4G, WiMAX, sondes spatiales (Mars Reconnaissance Orbiter, Messenger), plus de **4 milliards** de turbodécodeurs en circulation.

Claude Berrou (1951–)

Télécom Bretagne
Prix Marconi 2005

Médaille Hamming 2003

5– Logique Booléenne et Combinatoire

6– Logique Séquentielle

7– Micro-architectures FSMD

8– Introduction aux langages de description matériels : VHDL