Systèmes numériques : de l'algorithme aux circuits

Cours 1

Hadrien Barral (prenom.nom@ens.psl.eu)

Version : 1.0

Préambule : détails techniques

- Remplissez la feuille de présence
  - Nom
  - Promo/Département si Ulm
  - Établissement et email si non-Ulm
- Pensez à vous inscrire sur Moodle

Informations pratiques & programme

(cf page du cours)

https://sysnum.di.ens.psl.eu

Lisez ceci

- Pour les horaires : voir page du cours
- Pour les salles : voir https://diplome.di.ens.fr/calendar_fr.html
- Pour le contenu des séances : voir page du cours
- Pour les supports : voir page du cours
- Pour les dates de rendu : voir page du cours et supports liés

Machine d'Anticythère (III^{e} av. J.C.)

Source : Wikipedia

Pourquoi n'est-ce pas un ordinateur ?

Bonnes propriétés pour un ordinateur

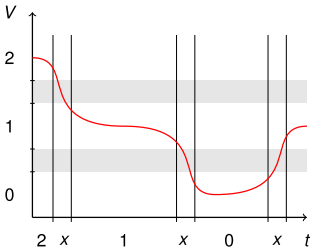
- Électrique (v.s. mécanique)
- Numérique (v.s. analogique)
- Logique suffisamment riche (v.s. calculateur)

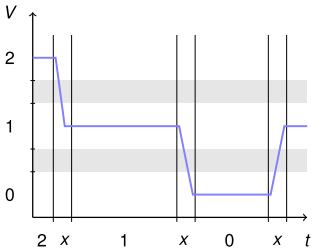
Le signal électrique

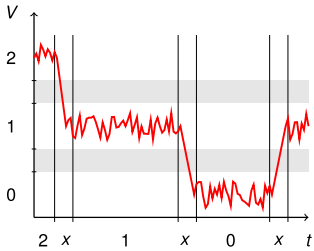
Support de l'information

- Passer de grandeurs physiques à des signaux électriques grâce à :
  - Capteurs
  - Transducteurs
- Possibilité de mesurer/manipuler :
  - la tension
  - le courant
  - la charge...
- C'est pour cela qu'on fait de l'électronique :
  - analogique : traitement de valeurs continues
  - numérique : traitement de valeurs discrètes

Codage numérique de l'information (1/3)





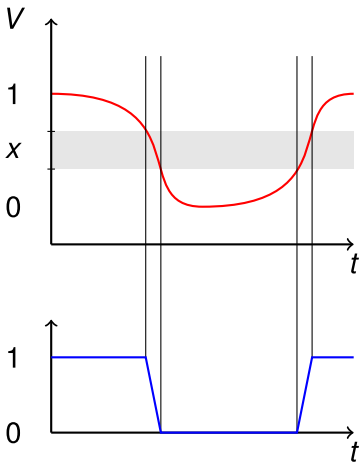


- Discrétiser le signal dans le temps
  - échantillonnage
- Représenter le signal par nombre fini de valeurs (de nombres)
  - quantification

Codage numérique de l'information : intérêts

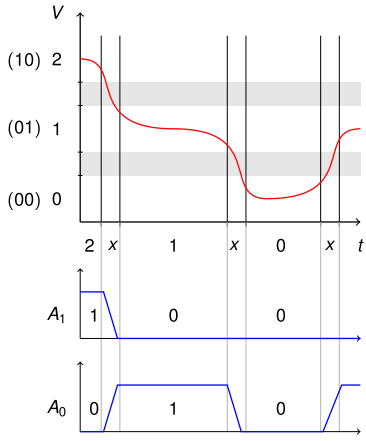
- Possibilité de reproduire sans perte et de façon illimitée l'information
- Indépendance du support utilisé
  - Câble électrique
  - Fibre optique
  - CDROM, disque dur...
- Possibilité d'organiser le traitement de l'information dans le temps

Codage binaire



- Interprétations logiques multiples
  - 0 / 1
  - `Vrai` / `Faux`
- Support électrique très simple
  - Codage utilisant deux valeurs

Codage binaire : travailler en base 2



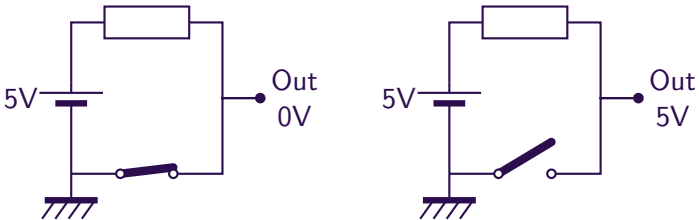
- Comment représenter plus de deux niveaux en binaire ?
  - Utiliser plusieurs fils
  - Les transmettre à tour de rôle
- Chaque élément est un **bit**
  - **b**inary **d**igit
- On peut représenter tous les nombres en base 2.

Le bit

- Convention
  - 2 niveaux de tension (0/5V ou −12/12V ou ...)
  - 2 niveaux de courant électrique
  - Absence/présence de lumière sur une fibre
  - ...
- Interprétations
  - `Vrai` / `Faux` pour des variables de commande ou statut
    - Contrôle
  - Valeur des nombres
    - Calcul

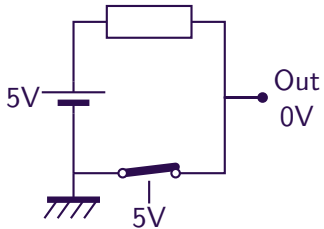
Génération d'un signal électrique binaire

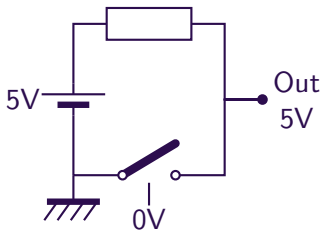
Comment générer deux niveaux de tension ?



- Interrupteur fermé
  → 0V en sortie
- Interrupteur ouvert
  → 5V en sortie

Interrupteur commandé : la théorie

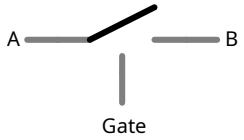




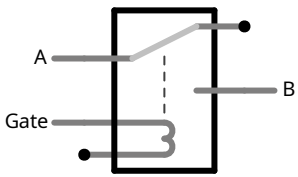
- La sortie dépend de V_{in} et V_{ref}
  - $V_{\text{in}} > V_{\text{ref}} \rightarrow$ interrupteur fermé
  - $V_{\text{in}} < V_{\text{ref}} \rightarrow$ interrupteur ouvert
- C'est bien joli, mais comment faire ceci dans le monde réel ?

Interrupteur commandé : la pratique

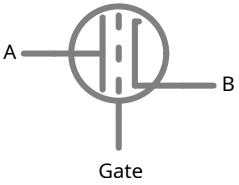
Interrupteur idéal :



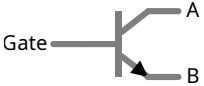
Ce qu'on sait faire :



Relais



Tube à vide



Transistor

Circuits à base d'interrupteur

L'assemblage de relais/triodes/transistors est *Turing-complet*. On peut :

- faire de la logique binaire
- créer de la mémoire

Comment ? On voit ceci un peu plus tard...

- créer des opérateurs arithmétiques
- créer des structures de contrôle

Comment ? Vu dans une séance future...

Réalisations

- **ASIC** (Application Specific Integrated Circuit), à savoir un circuit logique pour une application bien précise
- Logique programmable (CPLD, FPGA, ...)
- **CPU** (Central Processing Unit) : le microprocesseur, circuit universel par excellence
- GPU, NPU, TPU, ...

Algèbre de Boole

Le calcul booléen en 3 lignes :

$B = \{\top, \bot\} = \{\text{Vrai}, \text{Faux}\}$ (seulements 2 états possibles pour une variable logique)

Opérations : `AND`, `OR`, `NOT`, ..., `XOR`, `IMPLIES`, ...

$\wedge$, $\vee$, $\neg$, ..., $\oplus$, $\Longrightarrow$, ...

Pour ceux qui n'ont pas fait MPI

Pour ceux qui n'ont pas vu l'Algèbre de Boole et/ou la logique élémentaire, à regarder pour la semaine prochaine.
Idem, pour ce qui est représentation binaire des nombres et des bases usuelles (2,8,10,16).

Fonctions logiques

Deux catégories

Fonctions combinatoires

La sortie ne dépend que de l'état actuel des entrées.

$$\forall t, s(t) = F(e_0(t), e_1(t), \dots, e_n(t))$$

Fonctions séquentielles

La sortie dépend de l'état actuel des entrées et de leur passé.

$$\begin{aligned} s(t) = F(&e_0(t), e_1(t), \dots, e_n(t),\\ &e_0(t-1), e_1(t-1), \dots, e_n(t-1),\\ &\dots,\\ &e_0(t-\tau), e_1(t-\tau), \dots, e_n(t-\tau)) \end{aligned}$$

Fonctions logiques

Représentations

Plusieurs représentations possibles :

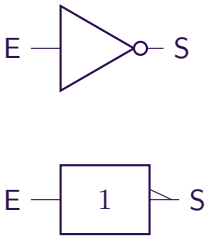
- Table de vérité : En donnant toutes les valeurs possibles pour toutes les entrées possibles.
- Analytique : En donnant l'équation analytique
- Graphique : En utilisant les symboles de fonctions de base
- HDL (Hardware Description Language) : Langage informatique de description du matériel

Fonctions élémentaires

L'inverseur (NOT)

- La sortie est le complément de l'entrée
- La sortie vaut 1 si et seulement si l'entrée vaut 0

Symbole



Équation

$$s = \neg i$$
$$= \overline{i}$$
$$= !i$$

Table de vérité

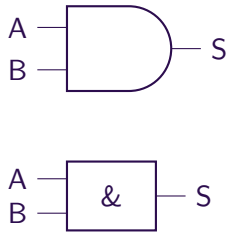
e	s
0	1
1	0

Fonctions élémentaires

Le "et" (conjonction, AND)

- La sortie vaut 1 si et seulement si les deux entrées valent 1
- Si l'une des entrées vaut 0 alors la sortie vaut 0

Symbole



Équation

$$s = a \wedge b$$
$$= a \cdot b$$
$$= a \times b$$

Table de vérité

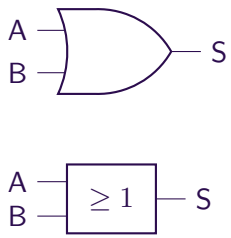
a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

Fonctions élémentaires

Le "ou" (disjonction, OR)

- Si l'une des entrées vaut 1 alors la sortie vaut 1
- La sortie vaut 0 si et seulement si les deux entrées valent 0

Symbole



Équation

$$s = a \vee b$$
$$= a + b$$
$$= a \mid b$$

Table de vérité

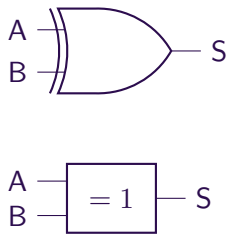
a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

Fonctions élémentaires

Le "ou exclusif" (XOR, MOD2)

- La sortie vaut 1 si une seule entrée est à 1
- La sortie vaut 1 si les deux entrées sont différentes

Symbole



Équation

$$s = a \oplus b$$
$$= a \hat{} b$$

Table de vérité

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

Fonctions élémentaires

Fonction complémentaires

- Il existe des symboles pour combiner une porte avec un `NOT`.

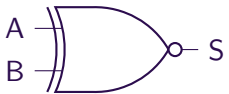
NAND

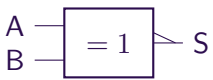
$s = \overline{a \wedge b}$

NOR

$s = \overline{a \vee b}$

XNOR

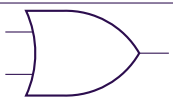
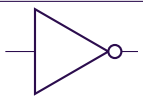
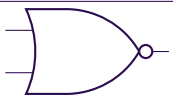
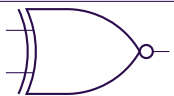




$s = \overline{a \oplus b}$

Notation des portes logiques dans les circuits

Tableau récapitulatif :

	AND $\wedge$ $\cdot$	OR $\vee$ $+$	XOR $\oplus$
			
NOT $\neg$	**NAND**	**NOR**	**XNOR**
			

Exercice

Mise en bouche

1. Exprimer $\oplus$ à partir de : $\wedge$, $\vee$, $\neg$

Simplification algébrique

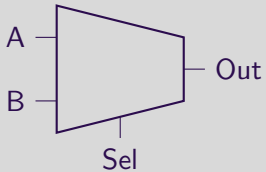
3. Simplifier l'expression suivante :

$$S = (a + b + c) \cdot \left(\overline{a} + \overline{d} \cdot e + f\right) + \overline{\left(\overline{d} + e\right)} \cdot \overline{a} \cdot c + a \cdot b$$

Exercice

MUX

On appelle *multiplexeur* une fonction/porte qui permet de sélectionner l'une de ses deux entrées en fonction d'une troisième entrée de sélection.



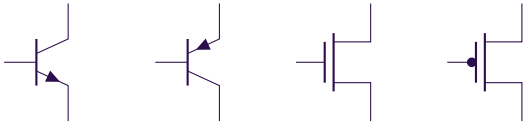
1. En utilisant les opérations de base de l'algèbre booléenne, réaliser un *multiplexeur*.
2. Faire de même pour un *démultiplexeur*.
3. Comment étendre ceci à un *(dé-)multiplexeur* à n entrées ?

Exercice

NAND is enough

1. Montrer la logique NAND est aussi expressive que la logique booléenne, autrement dit que NAND est une *porte universelle*.
   Pour chacune des portes suivantes, on écrira une formule et on dessinera le circuit : NOT, AND, OR, NOR, XOR, XNOR, MUX, DEMUX
2. Faire de même avec NOR.

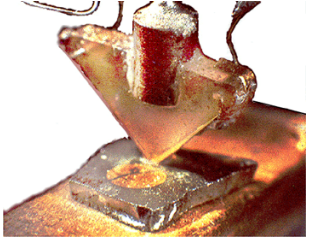
Transistor



Le transistor est plus bien compliqué en physique qu'en informatique. On va s'en tenir autant que possible au modèle "interrupteur".

Histoire

- **1947** : Invention du transistor (Nobel de physique)
- **1957** : Utilisation dans les ordinateurs
- **1958** : Premier circuit imprimé
- **1971** : Premier microprocesseur (Intel 4004)



Source : Jean Vuillemin — Taille : 13cm

Loi de Moore

Un peu d'histoire

Formulée par Gordon E. Moore en 1965 (lien)

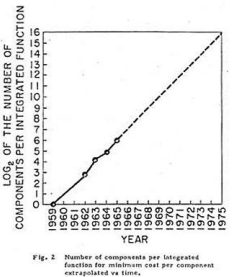
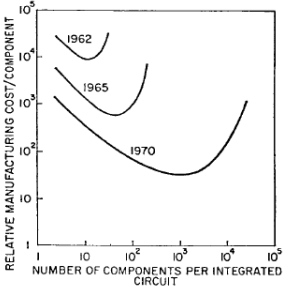


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.



> *The complexity for minimum component costs has increased at a rate of roughly a factor of two per year.*

Loi(s) de Moore

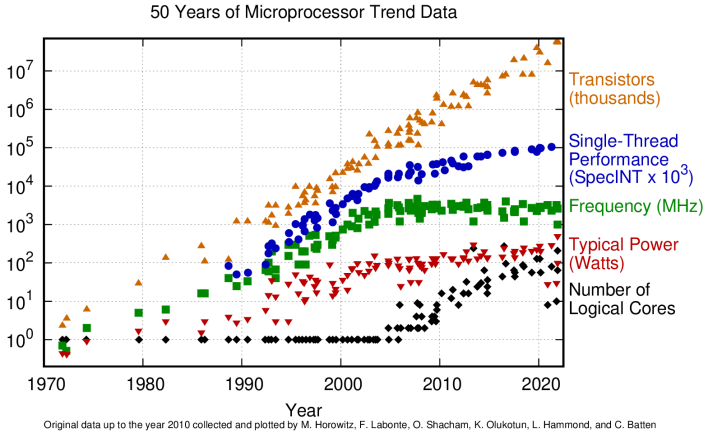
Conséquences

- Cette observation empirique est devenue une prédiction
- Cette prédiction est devenue une feuille de route pour les fondeurs
  - Ajustement des dépenses de R&D...
  - Ajustement des dépenses d'investissement pour les usines...
  - ...dans le but de suivre cette feuille de route
- Cette prédiction s'est étendue à beaucoup d'autres paramètres :
  - La taille des transistors est divisée par deux tous les X ans
  - La puissance de traitement double tous les X ans
  - La fréquence des circuits double tous les X ans
  - La consommation d'énergie est divisée par deux tous les X ans
  - Le nombre de cœurs double tous les X ans

Les (*sic*) lois de Moore sont des lois exponentielles. Peuvent-elles être toujours vraies ?

Loi(s) de Moore

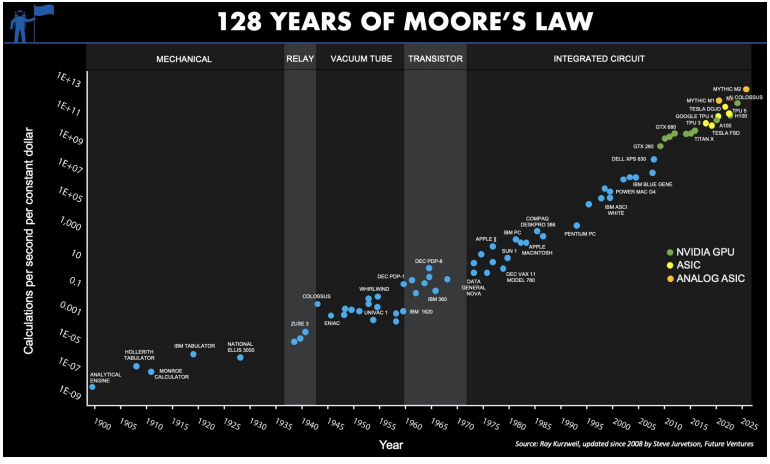
Et en 2025 ?



Source : Karl Rupp & al.

Loi de Moore

Et si on regarde plus loin ? (à prendre avec des pincettes)



Source : Steve Jurvetson

Changement de technologie (*Downsizing*)

Conséquences sur les performances

Facteur de réduction : β (historiquement égal à $\sqrt{2}$)

Évolution des capacités parasites

$$C_{\text{par}}(\beta) = \frac{W}{\beta} \cdot \frac{L}{\beta} \cdot \beta C'_{\text{ox}} = \frac{C_{\text{par}}}{\beta}$$

Évolution de l'énergie consommée par une porte

$$E_{\text{gate}}(\beta) = \frac{C_{\text{par}}}{\beta} \left(\frac{V_{\text{dd}}}{\beta} \right)^2 = \frac{E_{\text{gate}}}{\beta^3}$$

Évolution du temps de propagation des fonctions combinatoires

$$t_{\text{comp}}(\beta) = \frac{t_{\text{comp}}}{\beta}$$

Changement de technologie (*Downsizing*)

Diminuer les coûts et la consommation

- On n'exploite pas le gain en vitesse
  - $F_h(\beta) = F_h$
- Le gain en surface fait diminuer les prix
  - $\text{Area}(\beta) = \frac{\text{Area}}{\beta^2}$
- La consommation diminue
  - $P_{\text{chip}} = \frac{P_{\text{chip}}}{\beta^3}$
- Cette stratégie est particulièrement intéressante dans les systèmes embarqués :
  - transition du haut de gamme vers le milieu, puis bas de gamme (e.g smartphones)
  - ouverture aux appareil très basse consommation (e.g. objets connectés sur batterie)

Changement de technologie (*Downsizing*)

Augmenter les performances

- On exploite le gain en vitesse
  - $F_h(\beta) = \beta F_h$
- On profite du gain en taille des transistors pour accroître la complexité du circuit
  - $\text{Area}(\beta) = \text{Area}$
- La consommation ne change pas
  - $P_{\text{chip}} = P_{\text{chip}}$
- Cette stratégie est particulièrement intéressante pour les processeurs de serveurs :
  - la puissance de calcul profite de l'augmentation de fréquence
  - la puissance de calcul profite de l'augmentation du parallélisme

Changement de technologie (*Downsizing*)

En pratique

En pratique, *on améliore un peu les trois*, de manière pondérée selon l'application.
De plus, les modèles présentés avant ne fonctionnent plus si bien :

- Les vitesses maximum stagnent à cause des problèmes de dissipation thermique.
- On ne peut diminuer sans cesse la tension d'alimentation sans s'éloigner du modèle d'interrupteur idéal : les circuits ont des courants de fuite de moins en moins négligeables
- Les technologues doivent jongler avec des procédés de fabrication de plus en plus complexes (et couteux)

Taille des transistors

Voir :
https://en.wikipedia.org/wiki/List_of_semiconductor_scale_examples

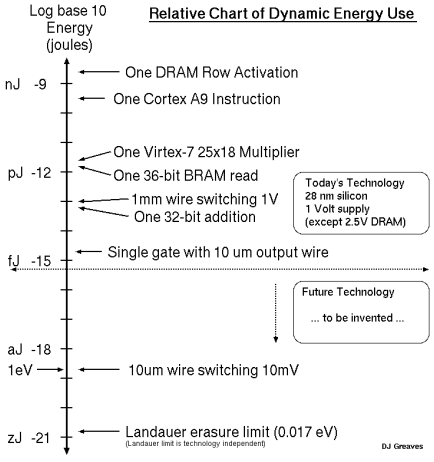
Considérations énergétiques

Principe de Landauer

Le niveau d'énergie minimal nécessaire pour effacer un bit d'information, connu sous le nom de limite de Landauer, est : $kT \ln 2$.

En 2018, nous sommes environ à 10^6 de cette limite.

Peut-on faire un ordinateur qui ne consomme rien ?



Source : David J Greaves

Calcul réversible

Diminuer la température ? Certes.

Ne pas effacer d'information ? Concept de "calcul réversible".
`NOT` est réversible. Qu'en est il de `OR`, `AND`, `XOR` ?

Pour aller plus loin :

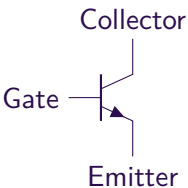
- Billiard-ball computer
- Porte de Toffoli
- Porte de Fredkin

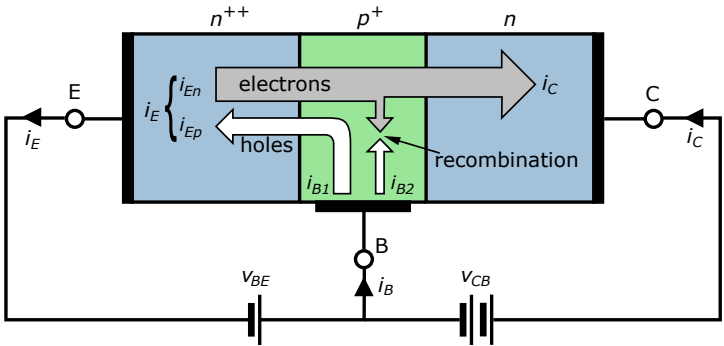
Éléments de base

On veut fabriquer un circuit logique, quels sont mes éléments de base en électronique ?

- transistor
- diode
- résistance
- ~~condensateur / inductance~~

Le transistor dipolaire





Source : Wikipedia

RTL : logique résistance-transistor

Au tableau

Amusez-vous avec https://www.falstad.com/circuit/circuitjs.html

DTL : logique diode-transistor

Au tableau

TTL : logique transistor-transistor

Au tableau

NMOS : logique NMOS

Au tableau

PMOS : logique PMOS

Au tableau