ELECINF 102 : Processeurs et Architectures Numériques

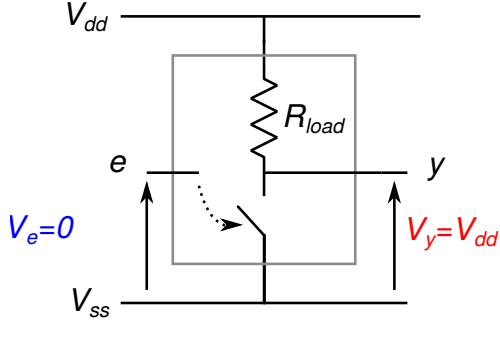
De l'architecture des portes logiques à l'impact de la nanoélectronique sur l'économie des TICs. . .

Tarik Graba
tarik.graba@telecom-paris.fr

Plan

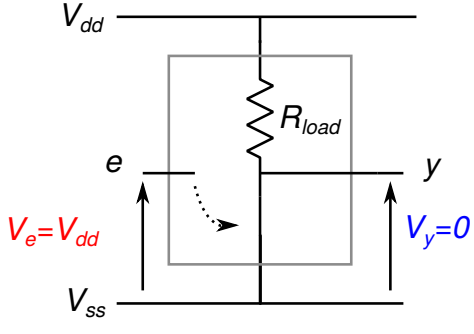
Construisons un inverseur
Inventaire des éléments nécessaires

- Une source d'alimentation : V_{dd}
- Une charge résistive : R_{load}
- Un interrupteur **piloté** par une tension :
  - $V_g = 0$ Interrupteur ouvert



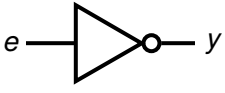
Construisons un inverseur
Inventaire des éléments nécessaires

- Une source d'alimentation : V_{dd}
- Une charge résistive : R_{load}
- Un interrupteur **piloté** par une tension :
  - $V_g = 0$ Interrupteur ouvert
  - $V_g = V_{dd}$ Interrupteur fermé



Construisons un inverseur
Inventaire des éléments nécessaires

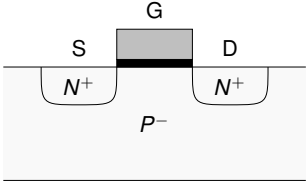
- Une source d'alimentation : V_{dd}
- Une charge résistive : R_{load}
- Un interrupteur **piloté** par une tension :
  - $V_g = 0$ Interrupteur ouvert
  - $V_g = V_{dd}$ Interrupteur fermé
- Une convention logique : $0 \equiv V_{ss}$, $1 \equiv V_{dd}$



Plan

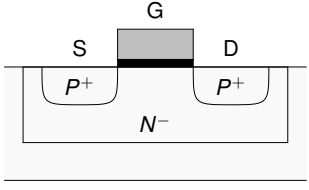
Logique CMOS

Complementary Metal Oxyde Semiconductor logic



Transistor nMOS

- Canal N
- Courant d'électrons
- Passant si $V_{gs} > V_T$

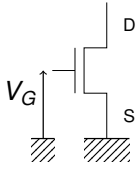


Transistor pMOS

- Canal P
- Courant de trous
- Passant si $V_{gs} < -|V_T|$

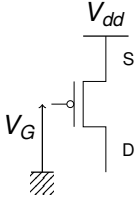
Le transistor MOS

Deux interrupteurs électroniques



Transistor nMOS

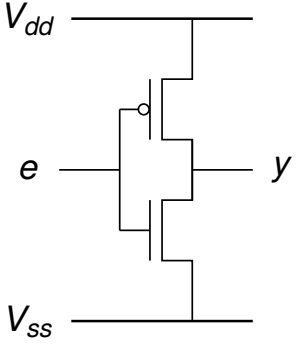
- $V_G = V_{ss}$
  $\Rightarrow$ interrupteur ouvert
- $V_G = V_{dd}$
  $\Rightarrow$ interrupteur fermé



Transistor pMOS

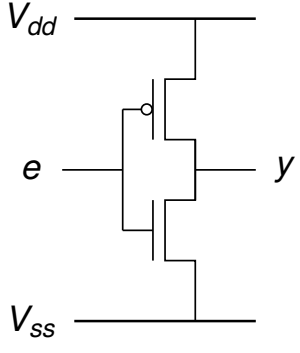
- $V_G = V_{ss}$
  $\Rightarrow$ interrupteur fermé
- $V_G = V_{dd}$
  $\Rightarrow$ interrupteur ouvert

Logique CMOS
L'inverseur CMOS



Logique CMOS
L'inverseur CMOS

- Entrée logique $e = 0$
  - $\rightarrow$ $V_e = 0$
    - $\rightarrow$ nMOS bloqué
    - $\rightarrow$ pMOS passant
  - $\rightarrow$ $V_y = V_{dd}$
- $\rightarrow$ Sortie logique $y = 1$



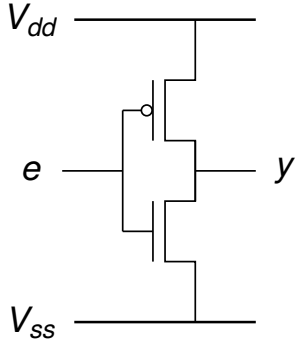
Logique CMOS
L'inverseur CMOS

- Entrée logique $e = 0$
  - $\rightarrow$ $V_e = 0$
    - $\rightarrow$ nMOS bloqué
    - $\rightarrow$ pMOS passant
  - $\rightarrow$ $V_y = V_{dd}$

$\rightarrow$ Sortie logique $y = 1$

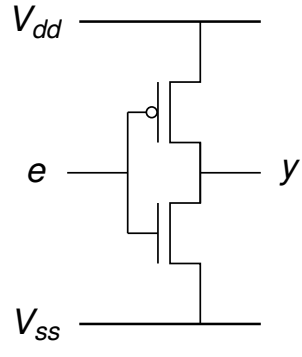
- Entrée logique $e = 1$
  - $\rightarrow$ $V_e = V_{dd}$
    - $\rightarrow$ nMOS passant
    - $\rightarrow$ pMOS bloqué
  - $\rightarrow$ $V_y = 0$

$\rightarrow$ Sortie logique $y = 0$



Logique CMOS
L'inverseur CMOS

- Entrée logique $e = 0$
  - $\rightarrow$ $V_e = 0$
    - $\rightarrow$ nMOS bloqué
    - $\rightarrow$ pMOS passant
  - $\rightarrow$ $V_y = V_{dd}$
- $\rightarrow$ Sortie logique $y = 1$
- Entrée logique $e = 1$
  - $\rightarrow$ $V_e = V_{dd}$
    - $\rightarrow$ nMOS passant
    - $\rightarrow$ pMOS bloqué
  - $\rightarrow$ $V_y = 0$
- $\rightarrow$ Sortie logique $y = 0$

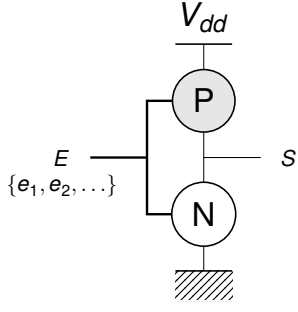


Consommation uniquement à l'occasion de nouveaux calculs (transitions)

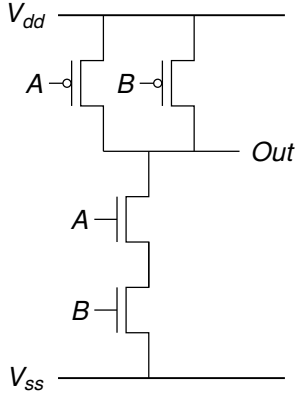
Logique CMOS
Généralisation à une porte complexe

Une porte avec les entrées $\{e_1, e_2, \ldots\}$ et la sortie S.

- Deux réseaux duaux :
  - nMOS : permet la mise à 0
  - pMOS : permet la mise à 1
- Les deux réseau ne doivent jamais être passants en même temps
- Pour que S soit une fonction logique :
  - Si N est passant P bloqué
  - Si P est passant N bloqué



Logique CMOS
La porte non-et (NAND)

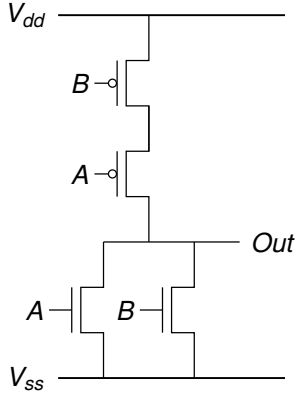


Logique CMOS
exerçons nous...

- Comment peut on construire la porte NOR à deux entrées ?

Logique CMOS
exerçons nous...

- Comment peut on construire la porte NOR à deux entrées ?



Logique CMOS

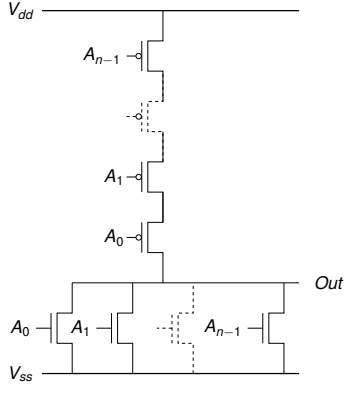
exerçons nous...

- Comment peut on construire la porte NOR à *n* entrées ?

Logique CMOS

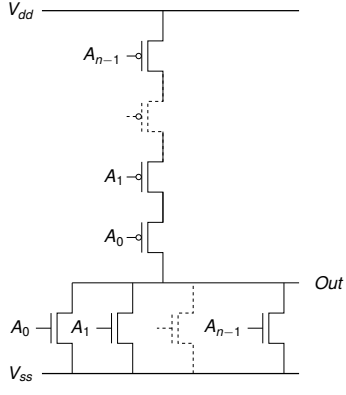
exerçons nous...

- Comment peut on construire la porte NOR à *n* entrées ?



Logique CMOS
exerçons nous...

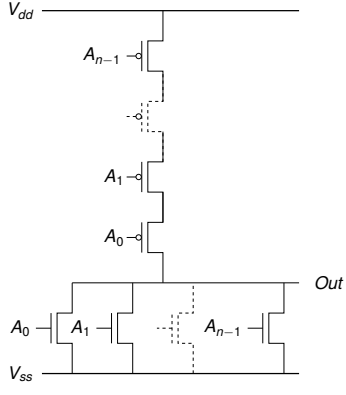
- Comment peut on construire la porte NOR à *n* entrées ?



- Attention : dans la pratique pas plus de 3/4 entrées (modèle trop simpliste du transistor)

Logique CMOS
exerçons nous...

- Comment peut on construire la porte NOR à *n* entrées ?



- Attention : dans la pratique pas plus de 3/4 entrées (modèle trop simpliste du transistor)
- Porte complexe = Association de portes simples.

Logique CMOS
exerçons nous...

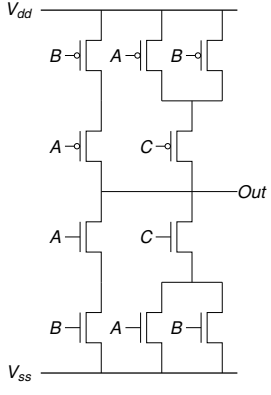
- Combien de transistors pour la fonction : $F(a,b,c) = \overline{a \cdot b + b \cdot c + c \cdot a}$?

Logique CMOS
exerçons nous...

- Combien de transistors pour la fonction : $F(a,b,c) = \overline{a \cdot b + b \cdot c + c \cdot a}$?
- On peut factoriser les transistors

Logique CMOS
exerçons nous...

- Combien de transistors pour la fonction : $F(a,b,c) = \overline{a \cdot b + b \cdot c + c \cdot a}$?
- On peut factoriser les transistors



Logique CMOS
exerçons nous...

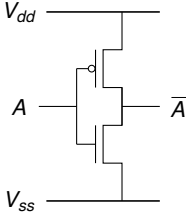
- Combien de transistors pour la fonction ou exclusif : $F(a,b) = a \cdot \overline{b} + \overline{a} \cdot b$?

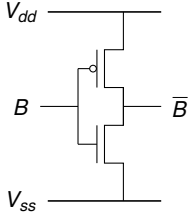
Logique CMOS
exerçons nous...

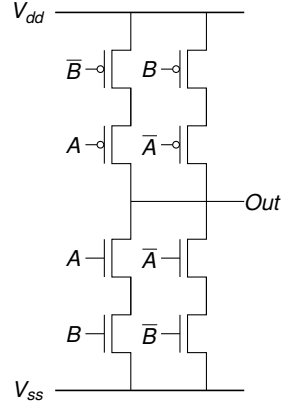
- Combien de transistors pour la fonction ou exclusif : $F(a,b) = a \cdot \overline{b} + \overline{a} \cdot b$?
- Il faut plusieurs portes en logique CMOS.

Logique CMOS
exerçons nous...

- Combien de transistors pour la fonction ou exclusif : $F(a,b) = a \cdot \overline{b} + \overline{a} \cdot b$?
- Il faut plusieurs portes en logique CMOS.







Plan

Un peu d'histoire
Loi(s) de Moore

Intel 1969 - 106 employés

https://commons.wikimedia.org/wiki/File:Intel_Mountain_View_in_1969.jpg

Un peu d'histoire
Loi(s) de Moore

- Gordon Moore, cofondateur d'Intel.
- 1965 : « La complexité des semiconducteurs double tous les ans **à coûts constants** »

Le triomphe du « **technology push** »

Un peu d'histoire
Loi(s) de Moore

- Gordon Moore, cofondateur d'Intel.
- 1965 : « La complexité des semiconducteurs double tous les ans **à coûts constants** »
- Constatation devenue par la suite auto-prédictive.
  - Ajustement des dépenses de R&D. . .
  - Ajustement des investissements dans les usines . . .
  - . . . pour suivre la prédiction.

Le triomphe du « **technology push** »

Un peu d'histoire
Loi(s) de Moore

- Gordon Moore, cofondateur d'Intel.
- 1965 : « La complexité des semiconducteurs double tous les ans **à coûts constants** »
- Constatation devenue par la suite auto-prédictive.
  - Ajustement des dépenses de R&D. . .
  - Ajustement des investissements dans les usines . . .
  - . . . pour suivre la prédiction.
- Extrapolation à :

  « La fréquence de fonctionnement des . . . double tous les. . . »
  « La consommation des . . . est divisée par deux tous les. . . »

Le triomphe du « **technology push** »

Un peu d'histoire
Loi(s) de Moore

- Gordon Moore, cofondateur d'Intel.
- 1965 : « La complexité des semiconducteurs double tous les ans **à coûts constants** »
- Constatation devenue par la suite auto-prédictive.
  - Ajustement des dépenses de R&D. . .
  - Ajustement des investissements dans les usines . . .
  - . . . pour suivre la prédiction.
- Extrapolation à :

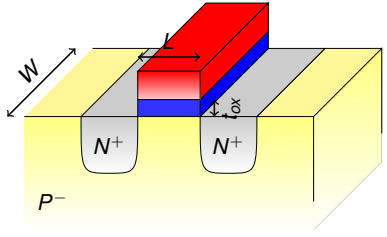
  « La fréquence de fonctionnement des . . . double tous les. . . »
  « La consommation des . . . est divisée par deux tous les. . . »

Le triomphe du « **technology push** »

Critères de performances

- La **surface** :
  Plus le circuit est petit, meilleur est le rendement de fabrication et donc plus faible est le coût de fabrication.
  - Réduire la taille des transistors (technologue)
  - Réduire le nombre de transistors (architecte)
- La **vitesse** :
  Plus la logique est rapide, plus on peut effectuer de calculs dans la même durée de temps.
  - Comment augmenter la fréquence d'horloge ?
- La **consommation** :
  Le « calcul » implique une consommation d'énergie
  - Comment minimiser cette consommation ? (objets connectés . . .)
  - Comment évacuer la chaleur dissipée ? (serveurs pour le cloud . . .)

Le transistor MOS
Une vision plus physique



Le courant dans le transistor **passant**

$$I_{DS_{max}} = K_n \cdot (V_{dd} - V_{TN})^2 \text{ avec } K_n = \frac{1}{2}\mu_{0N} \cdot C'_{ox} \frac{W_N}{L_N}$$

La capacité parasite de la grille du transistor

$$C_{ox} = C'_{ox} W_N \cdot L_N$$

Temps de calcul d'une porte logique
Capacité parasite

- La sortie d'une porte en logique CMOS est reliée :
  - A des fils de connexion
  - A des entrées de portes CMOS (grilles de transistors)

Temps de calcul d'une porte logique
Capacité parasite

- La sortie d'une porte en logique CMOS est reliée :
  - A des fils de connexion
  - A des entrées de portes CMOS (grilles de transistors)
- Ses éléments sont équivalents à une unique capacité parasite C_{par} :
  - qui doit être chargée dans les transitions montantes de la sortie de la porte
  - qui doit être déchargée dans les transitions descendantes de la sortie de la porte

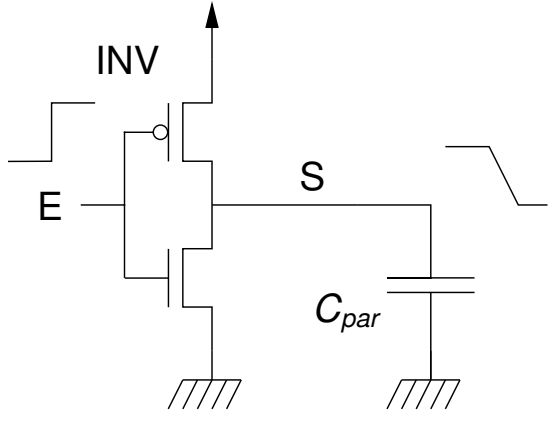
Temps de calcul d'une porte logique
Capacité parasite

- La sortie d'une porte en logique CMOS est reliée :
  - A des fils de connexion
  - A des entrées de portes CMOS (grilles de transistors)
- Ses éléments sont équivalents à une unique capacité parasite C_{par} :
  - qui doit être chargée dans les transitions montantes de la sortie de la porte
  - qui doit être déchargée dans les transitions descendantes de la sortie de la porte
- Le temps de calcul d'une porte logique est directement lié à ce temps de charge et de décharge

Temps de calcul d'une porte logique

Cas d'une transition montante à l'entrée d'un inverseur

- Le courant traversé par les transistors pour la charge ou la décharge de la capacités parasite C_{par} est $I_{DS_{max}}$



Temps de calcul d'une porte logique
Temps de calcul d'un inverseur

Relation courant/tension dans la capacité parasite

$$I_{C_{par}} = C_{par} dV_{C_{par}}/dt$$

Temps de calcul d'une porte logique
Temps de calcul d'un inverseur

Relation courant/tension dans la capacité parasite

$$I_{C_{par}} = C_{par} dV_{C_{par}}/dt$$

Courant de décharge quasi-constant à travers le transistor NMOS

$$I_{C_{par}} \approx I_{DSmax} = K_n \cdot (V_{dd} - V_{tn})^2$$

Temps de calcul d'une porte logique

Temps de calcul d'un inverseur

Relation courant/tension dans la capacité parasite

$$I_{C_{par}} = C_{par} dV_{C_{par}}/dt$$

Courant de décharge quasi-constant à travers le transistor NMOS

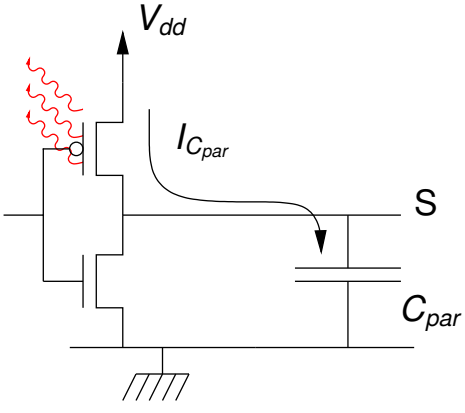
$$I_{C_{par}} \approx I_{DSmax} = K_n \cdot (V_{dd} - V_{tn})^2$$

Décharge de V_{dd} à 0

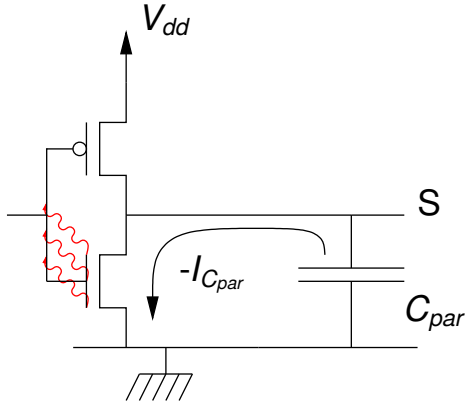
$$t_{calc} = C_{par} \frac{\Delta V}{I_{DSmax}} = C_{par} \frac{V_{dd}}{K_n \cdot (V_{dd} - V_{tn})^2}$$

Consommation de la logique CMOS

Énergie dissipée et énergie stockée



Transition montante



Transition descendante

Consommation de la logique CMOS

Bilan d'énergie

Charge : Énergie fournie par l'alimentation

$$E_{V_{dd}} = C_{par} \int_0^{V_{dd}} V_{dd} \, \mathrm{d}V_s = C_{par} {V_{dd}}^2$$

Charge : Énergie potentielle stockée dans la capacité

$$E_{Cpar} = C_{par} \int_0^{V_{dd}} V_s \, \mathrm{d}V_s = C_{par} \frac{{V_{dd}}^2}{2}$$

Consommation de la logique CMOS

Bilan d'énergie

Charge : Énergie fournie par l'alimentation

$$E_{V_{dd}} = C_{par} \int_0^{V_{dd}} V_{dd} \, \mathrm{d}V_s = C_{par} {V_{dd}}^2$$

Charge : Énergie potentielle stockée dans la capacité

$$E_{Cpar} = C_{par} \int_0^{V_{dd}} V_s \, \mathrm{d}V_s = C_{par} \frac{{V_{dd}}^2}{2}$$

En moyenne $C_{par} \frac{{V_{dd}}^2}{2}$ dissipée (ou consommée) à chaque transition de la sortie de la porte

Consommation de la logique CMOS

Extrapolation à un circuit intégré

- Soit C_{total} la capacité parasite totale du circuit
- Soit F_h la fréquence de fonctionnement du circuit
- Soit T_{act} la probabilité de transition des portes à chaque cycle d'horloge ($T_{act} \approx 0.3$)

Consommation de la logique CMOS

Extrapolation à un circuit intégré

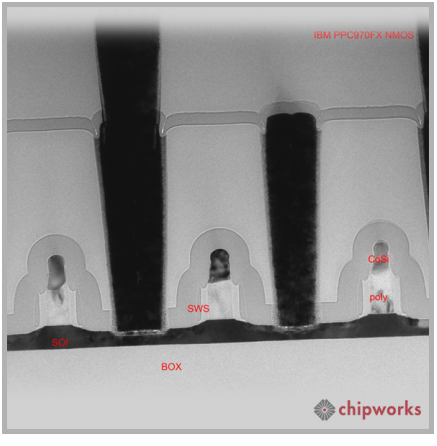
- Soit C_{total} la capacité parasite totale du circuit
- Soit F_h la fréquence de fonctionnement du circuit
- Soit T_{act} la probabilité de transition des portes à chaque cycle d'horloge ($T_{act} \approx 0.3$)

Puissance consommée par le circuit :

$$P_{circuit} \approx T_{act} F_h C_{total} {V_{dd}}^2$$

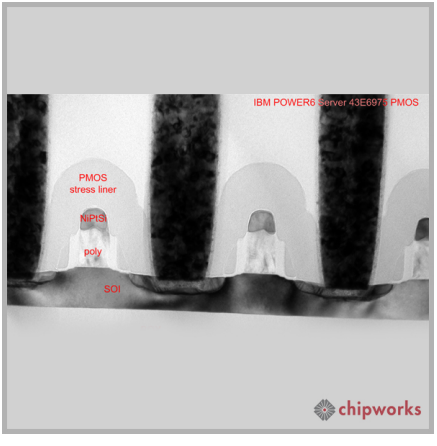
Plan

Transistor en 2004



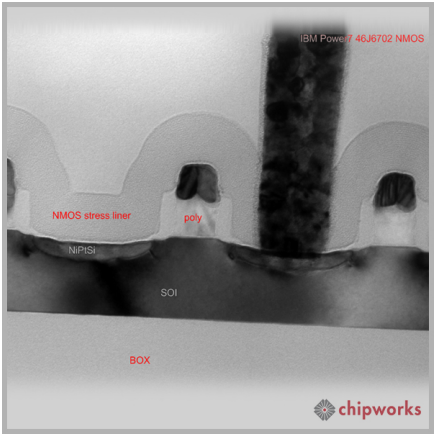
PPC970fx (90nm)

Transistor en 2008



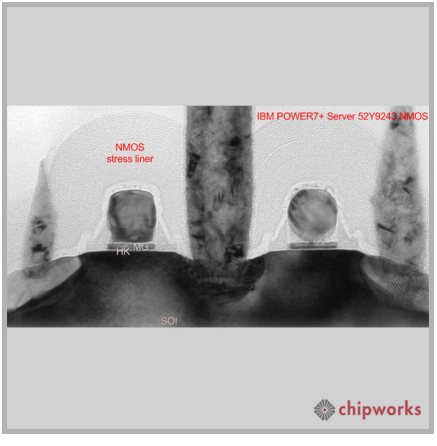
Power6 (65nm)

Transistor en 2011



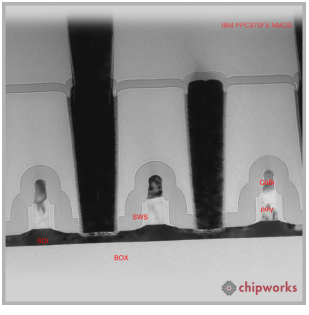
Power7 (45nm)

Transistor en 2013

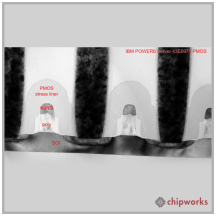


Power7+ (32nm)

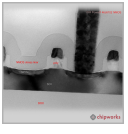
A l'échelle



2004
90nm
PPC970fx



2009
65nm
Power6



2011
45nm
Power7

2013
32 nm
Power7+

Crédits

Les images sont issues de l'analyse de l'évolution des technologies IBM faite par ChipWorks Inc.
L'analyse ainsi que les images originales étaient accessibles en 2014 ici :
`http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/ibm-continues-major-source-chip-innovation/`

Changement de technologie
"Downsizing théorique"

- Générations technologiques :
  - La longueur de grille minimale est caractéristique d'une génération technologique (90nm, 65nm, 40nm, 28nm, ...)
  - À chaque génération, les *fondeurs* visent une réduction de la surface d'un facteur **2**
  - Les *fondeurs* investissent les milliards nécessaires pour cela ...

Changement de technologie
"Downsizing théorique"

- Générations technologiques :
  - La longueur de grille minimale est caractéristique d'une génération technologique (90nm, 65nm, 40nm, 28nm, ...)
  - À chaque génération, les *fondeurs* visent une réduction de la surface d'un facteur **2**
  - Les *fondeurs* investissent les milliards nécessaires pour cela ...
- On utilise un facteur de réduction $\beta = \sqrt{2}$
  - division par β de la largeur W et la longueur L des transistors
  - division par β de l'épaisseur d'oxyde de grille T_{OX}
  - division par β de la tension d'alimentation V_{dd} des circuits
  - division par β de la tension de seuil V_T des transistors

Changement de technologie
Conséquences sur les performances

Évolution des capacités parasites

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

Évolution de l'énergie consommée par une porte

$$E_{porte}(\beta) = \frac{C_{par}}{\beta}(\frac{V_{dd}}{\beta})^2 = \frac{E_{porte}}{\beta^3}$$

Évolution du temps de propagation des fonctions combinatoires

$$t_{calc}(\beta) = \frac{t_{calc}}{\beta}$$

Changement de technologie
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
  - $Surf(\beta) = \frac{Surf}{\beta^2}$
- La consommation diminue.
  - $P_{circuit}(\beta) = T_{act}(F_h)\frac{E_{circuit}}{\beta^3} = \frac{P_{circuit}}{\beta^3}$

Changement de technologie
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
  - $Surf(\beta) = \frac{Surf}{\beta^2}$
- La consommation diminue.
  - $P_{circuit}(\beta) = T_{act}(F_h)\frac{E_{circuit}}{\beta^3} = \frac{P_{circuit}}{\beta^3}$
- Cette stratégie est particulièrement intéressante dans l'embarqué :
  - transition du haut de gamme vers le milieu, puis bas de gamme (smartphones),
  - ouverture à de nouvelles utilisations (objets connectés).

Changement de technologie
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
  - $Surf(\beta) = Surf$
- La consommation ne change pas
  - $P_{circuit}(\beta) = P_{circuit}$

Changement de technologie
Augmenter les performances

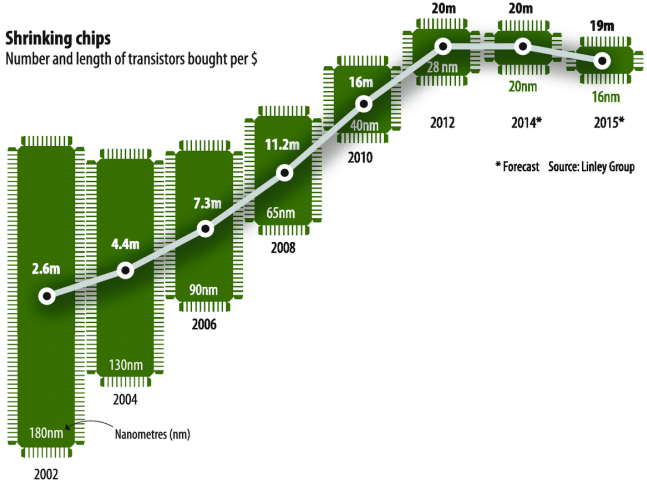
- On exploite le gain en vitesse
  - $F_h(\beta) = \beta F_h$
- On profite du gain en taille des transistors pour accroître la complexité du circuit
  - $Surf(\beta) = Surf$
- La consommation ne change pas
  - $P_{circuit}(\beta) = P_{circuit}$
- Cette stratégie est particulièrement pour les processeurs 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
Dans la pratique

Dans la pratique ça ne fonctionne plus si bien :

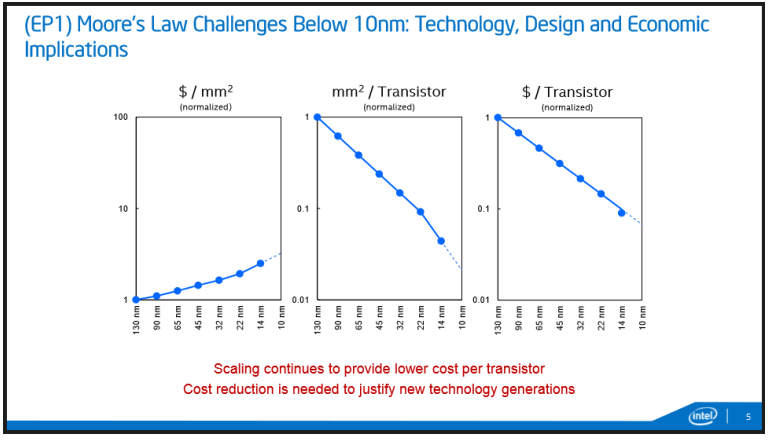
- Les vitesses maximum stagnent depuis le début des années 2000 (3 à 4 Ghz) à cause des problèmes de dissipation thermique.
- On ne peut diminuer sans cesse la tension d'alimentation sans s'éloigner du modèle d'interrupteur idéal : les circuits ont des courants de fuite de moins en moins négligeables
- Les technologues doivent jongler avec des procédés de fabrication de plus en plus complexes (et couteux) pour continuer à suivre la "loi de Moore".
- On a plusieurs fois prédit la fin de la loi de Moore pour des raisons "scientifiques" (physique du transistor) mais il semble depuis 2014 que le plus grave problème soit économique !

Evolution technologique
(prevision 2013) La fin de la loi de Moore ?

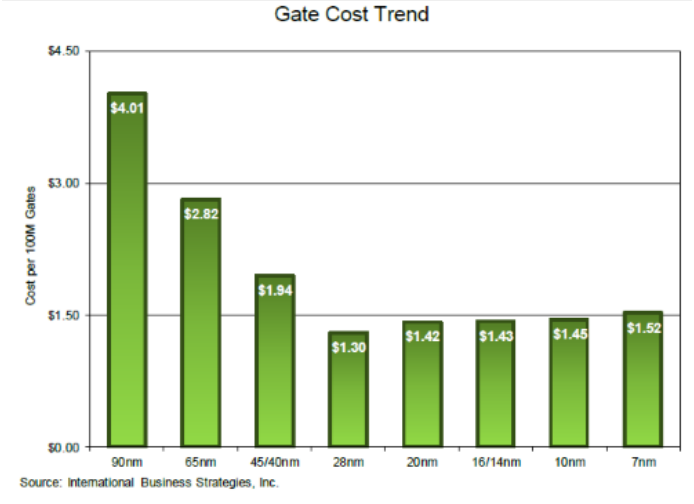


Evolution technologique

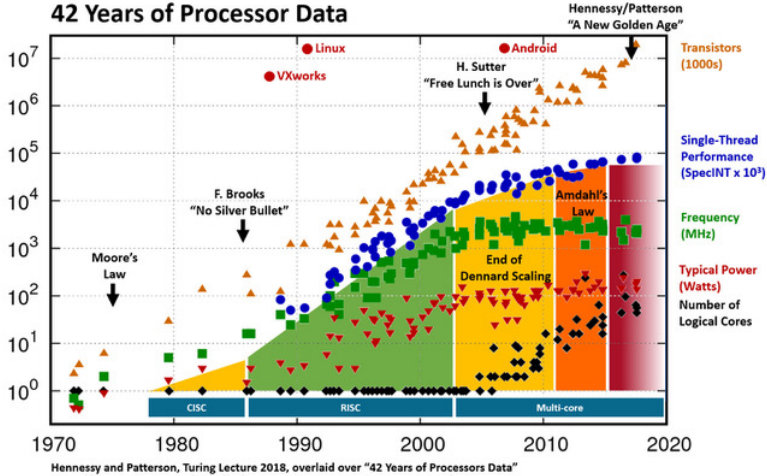
(2015) Pas pour Intel ?



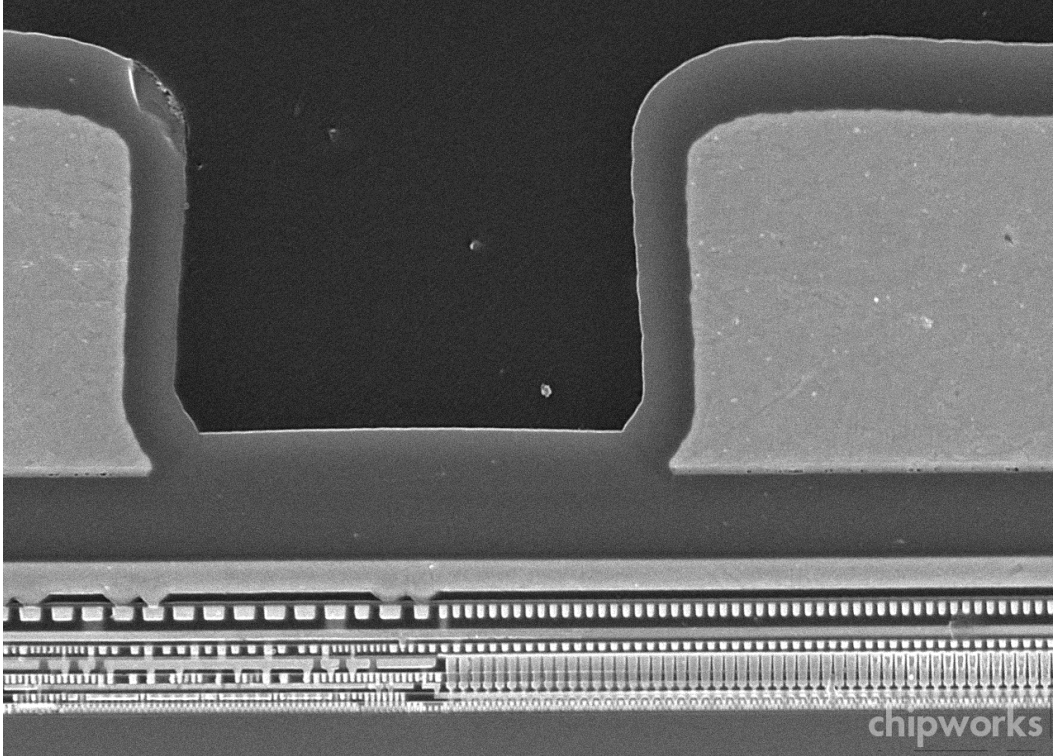
Evolution technologique
(prevision 2016) Décidement la controverse continue



Evolution technologique



Hennessy and Patterson, Turing Lecture 2018, overlaid over "42 Years of Processors Data"
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp


chipworks

Gordon Moore Fishing

source https://commons.wikimedia.org/wiki/File:Gordon_moore_fishing.jpg