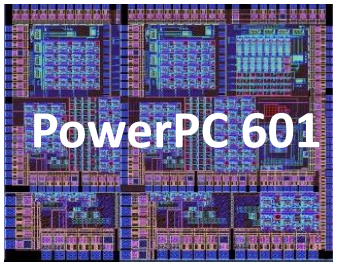
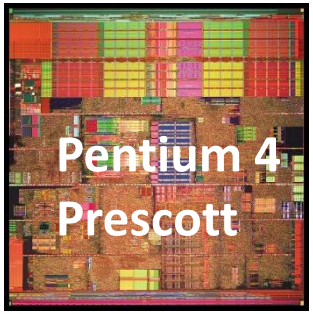
1983



1992



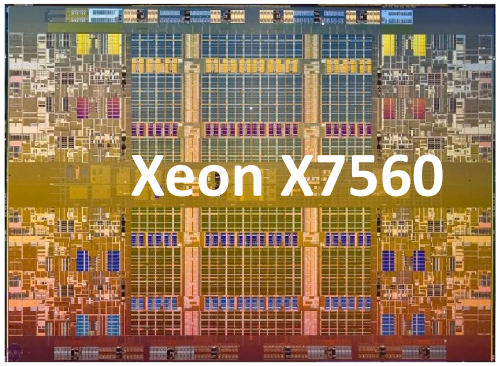
2004

L'évolution des processeurs depuis 1983

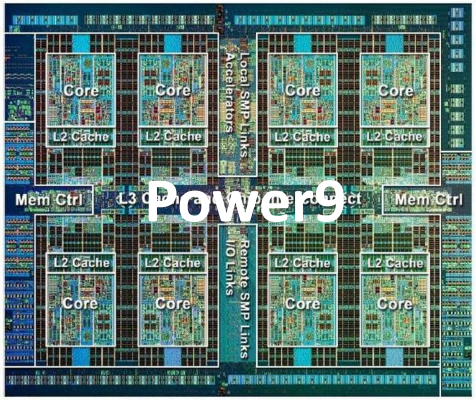
Moore et deux équations

Daniel Etiemble

LRI – Université Paris Sud



2010



2017

2016

Pourquoi 1983 ?

- 18ème anniversaire de la loi de Moore.
- Apple Lisa
- Conception d'un CPU 16 bits par ARM

- **Bernard Goossens devient assistant à Paris 7 (1 Octobre).**

- **Examinons la suite…**

Evolution exponentielle

- *Des lois fondamentales?*
  - **La loi de Moore**
  - Le temps d'exécution d'un programme **Hennessy-Patterson**

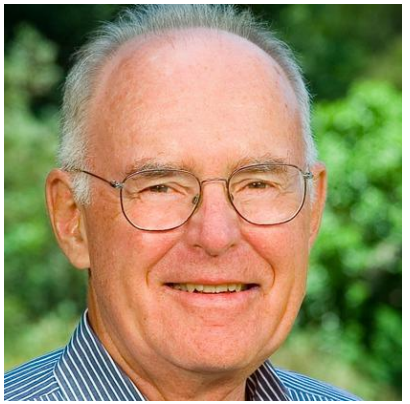
$$T_{ex} = NI * (CPIcpu + CPImem) * T_c = \frac{NI}{IPC * F}$$

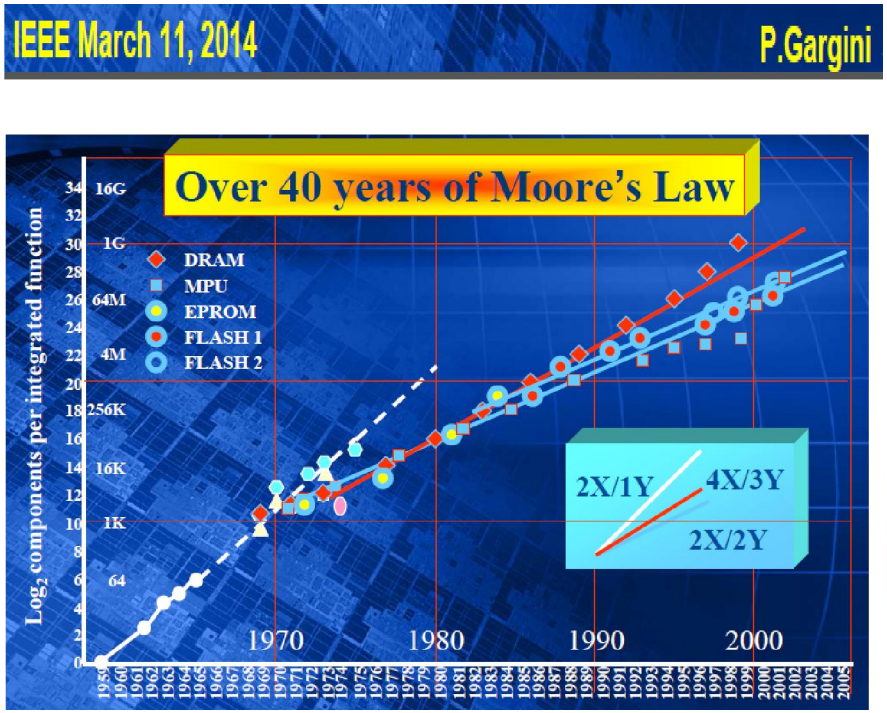
Nb d'instructions — NB Cycles/Instruction Calcul – Attente mémoire — Temps de cycle

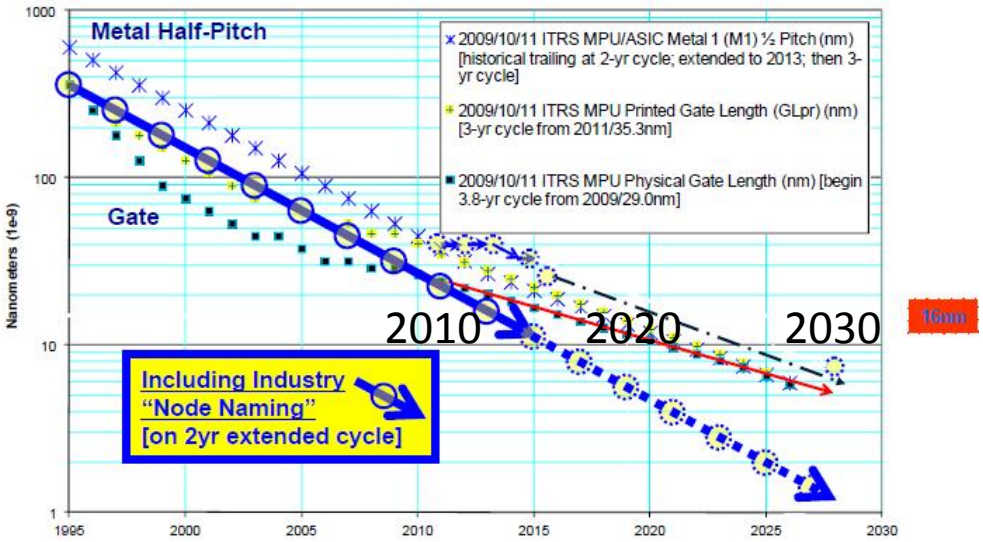
  - Puissance dissipée CMOS

$$P_d = V_{dd} * I_{fuite} + \alpha * \sum C_i * V_{dd}^2 * F$$

La loi de Moore



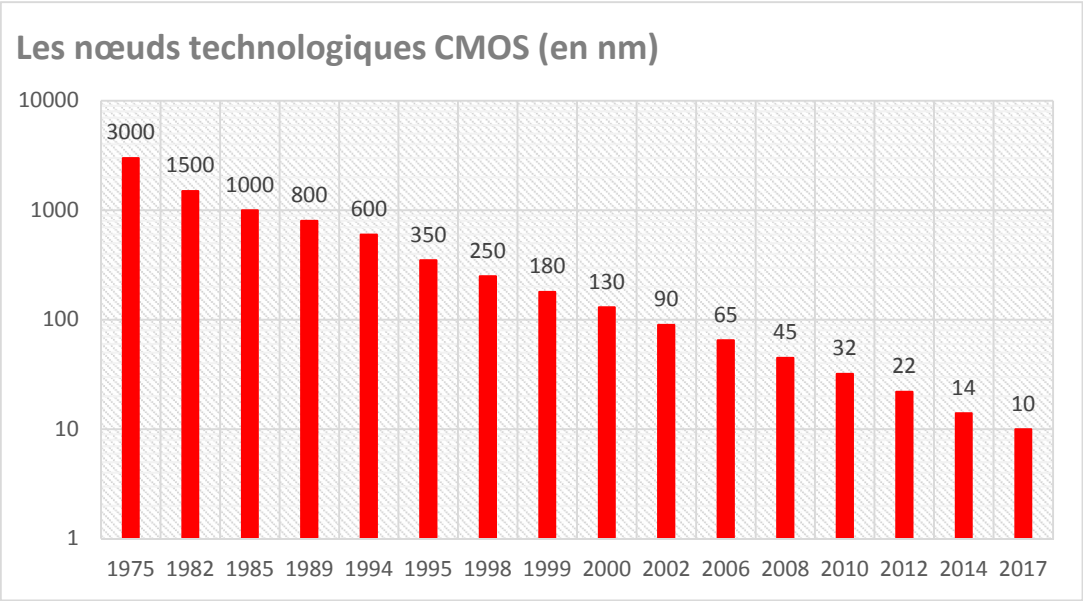




Le Nb de transistors par circuits double tous les N mois (12/18/24)

N augmente

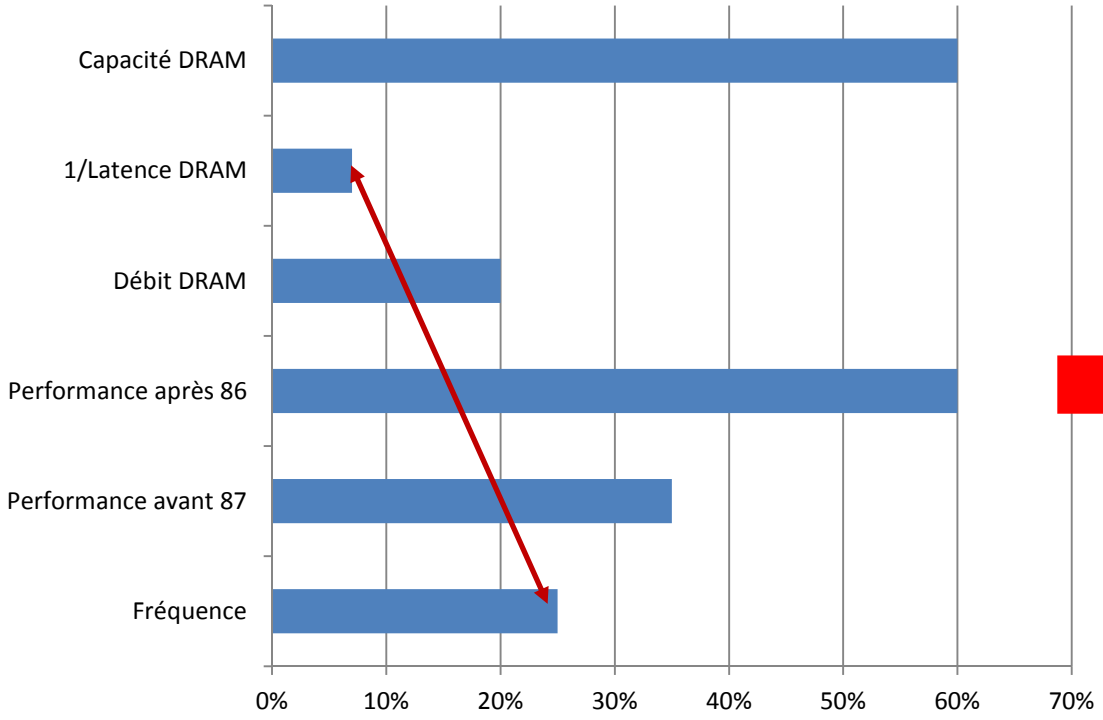
Les « nœuds » technologiques



- D'un nœud au suivant
  - **(en première approximation)**
  - Tp porte/1,4 => Fréquences d'horloge plus élevées
  - Augmentation du Nb de transistors /unité de surface.

Mais différentiels d'exponentielles

- Evolution par an



Hiérarchies de caches

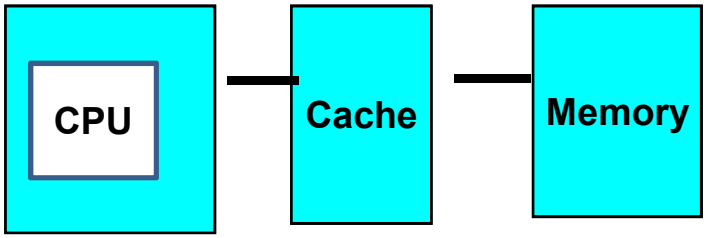
Une ou deux puces

- Opérateurs flottants
  - Coprocesseurs (ex : x87)…
  - Intégrés
- Caches
  - Externes
  - Internes (L1 puis L1-L2 puis…)
- CPU et GPU
  - 2 puces distinctes
  - GPU intégré (APU)
- Plusieurs processeurs
  - Multiprocesseur
  - Multi-cœur

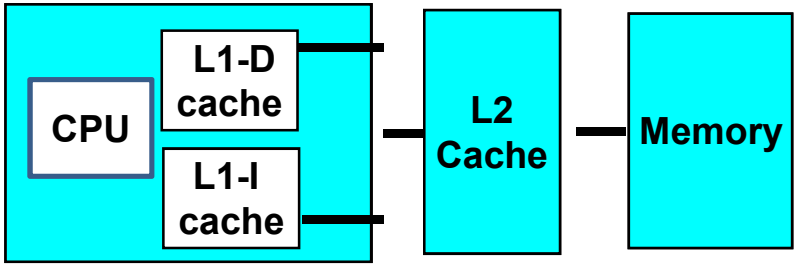
Du coprocesseur à l'intégration dans la même puce

De puces distinctes à une seule puce

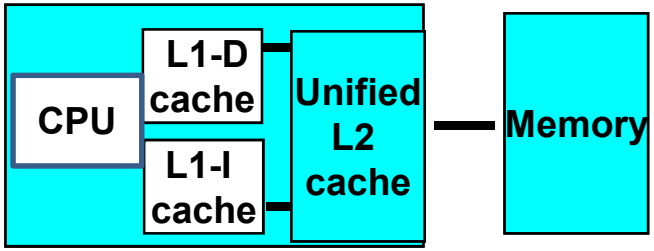
Evolution des hiérarchies de cache



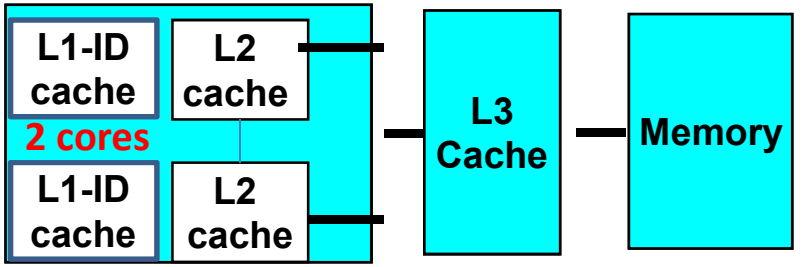
386



Pentium



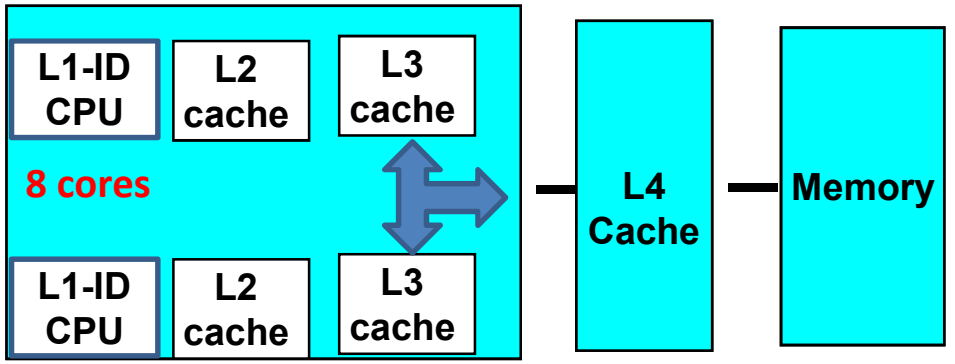
Pentium 4



Power6

Réduit CPImem

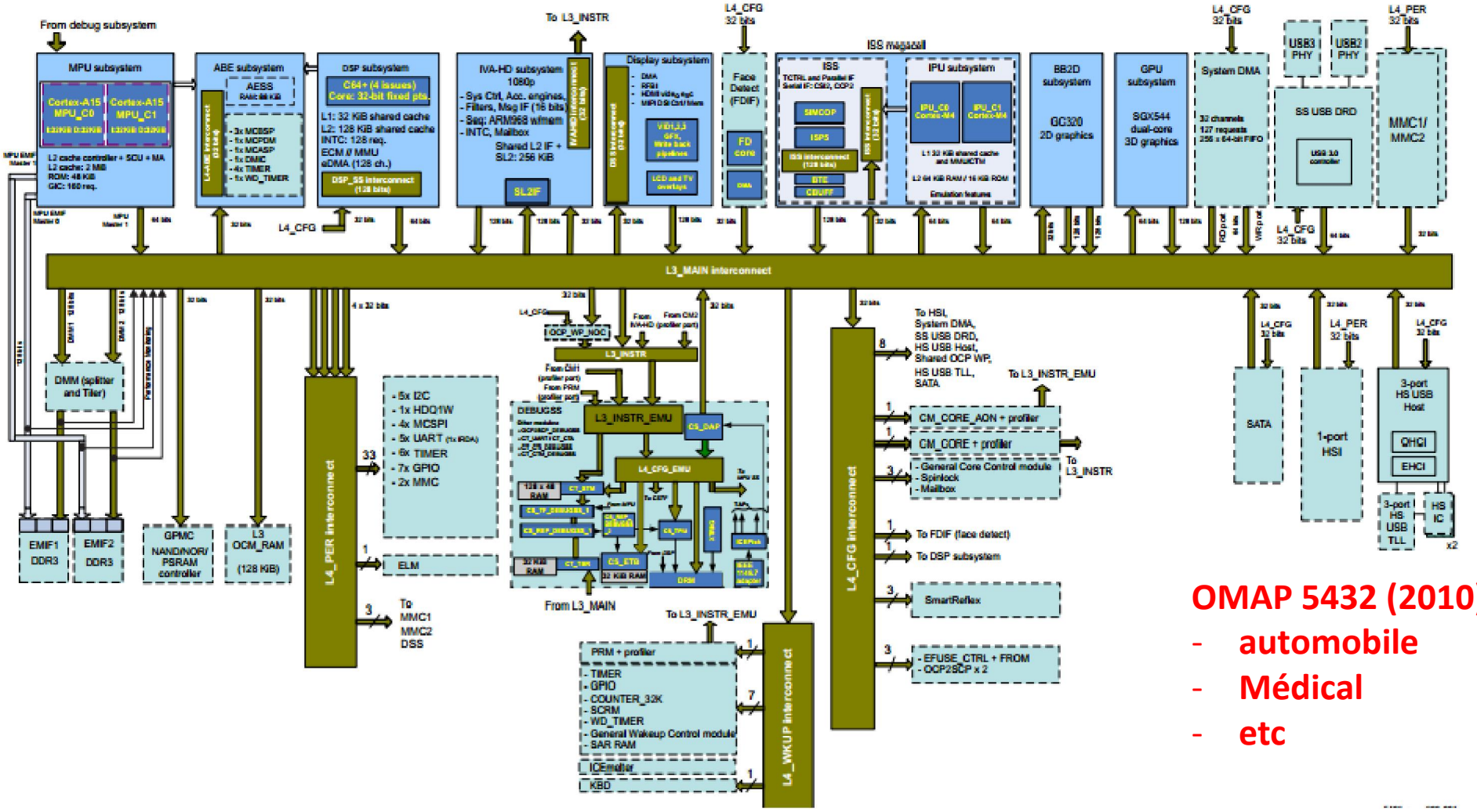
Latence et
Bande passante



Power 8

NUCA

Ajouter des fonctionnalités

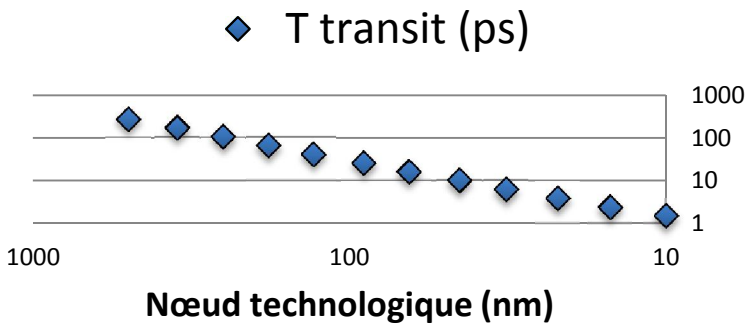


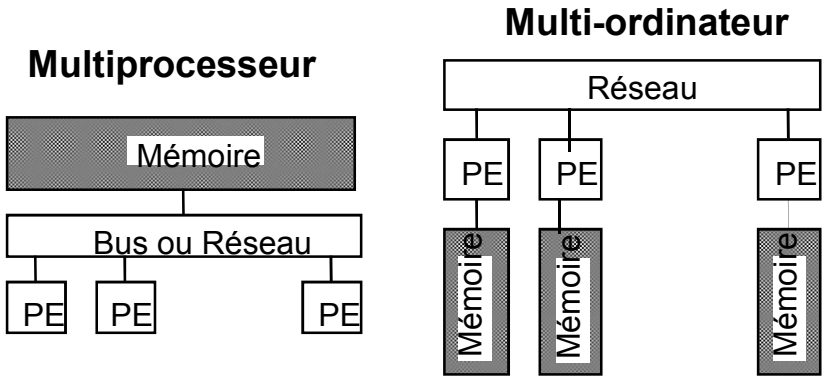
OMAP 5432 (2010)

- **automobile**
- **Médical**
- **etc**

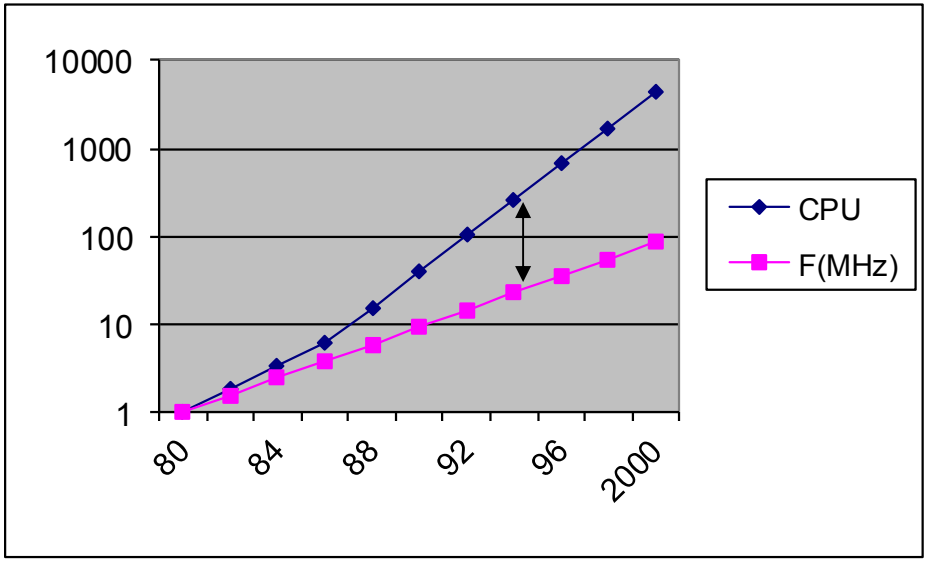
Améliorer les performances

- L'équation : $T_{ex} = \frac{NI}{IPC * F}$
  - Augmenter **F** (nœuds technologiques successifs)
  - Augmenter **IPC**
    - IPC <1 → IPC > 1
    - Superscalaires et VLIW
  - Diminuer **NI**
    - Dans les monoprocesseurs : SIMD, SIMT
    - Les architectures parallèles





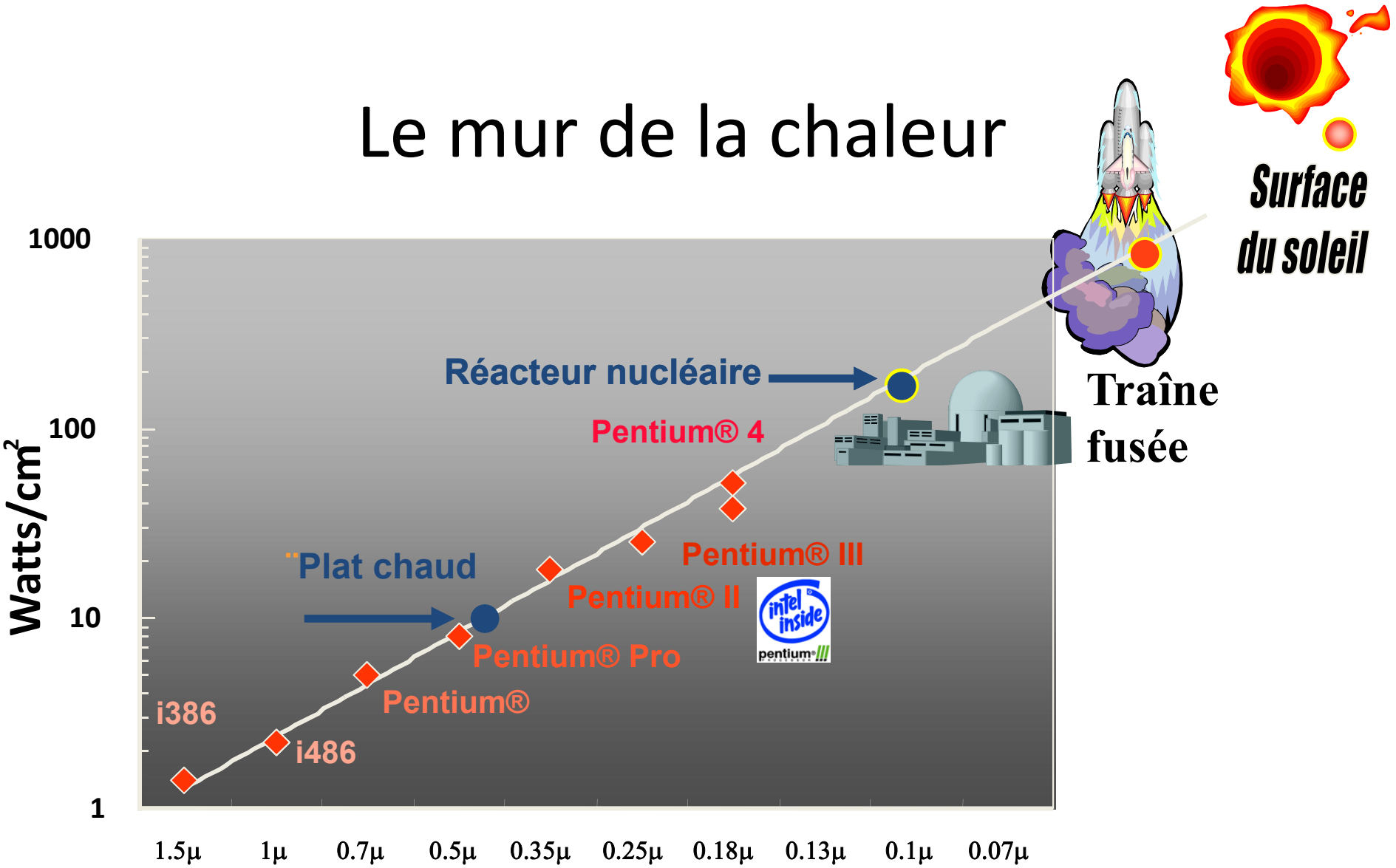
Améliorer la performance : $T_{ex} = \frac{NI}{IPC * \boldsymbol{F}}$



- Environ 25% par an… jusqu'à 4 GHz en 2017
  - Sauf IBM z14 CPU (5.2 GHz) avec refroidissement par eau
  - 2018 : Intel i7-8086K (5 GHz via *overclocking*)
- Limites : **mur de la chaleur** !

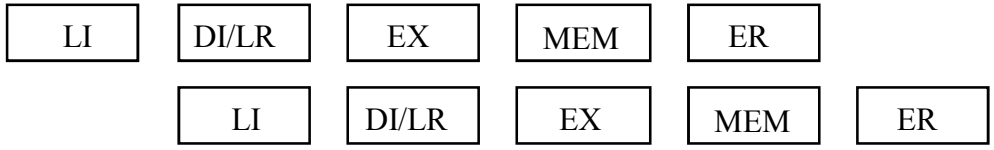
$$P_d = V_{dd} * I_{fuite} + \alpha * \sum \boldsymbol{C_i} * V_{dd}^2 * \boldsymbol{F}$$

Le mur de la chaleur

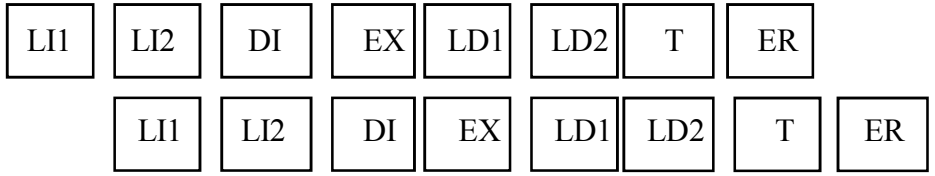


* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

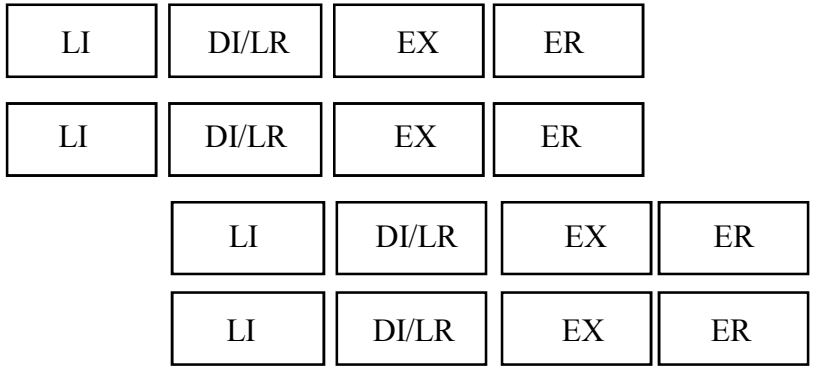
Améliorer la performance : $T_{ex} = \frac{NI}{IPC_{CPU} * F}$



Pipeline



Superpipeline



Superscalaires
- Dans l'ordre (multipipeline)
- Non ordonnés (flot de données)

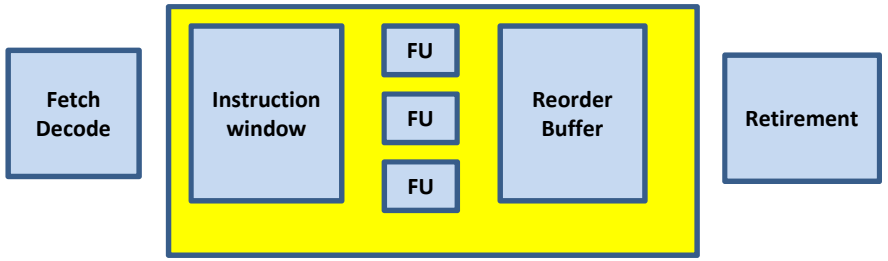
VLIW

Limits on Instruction Level Parallelism (ILP)

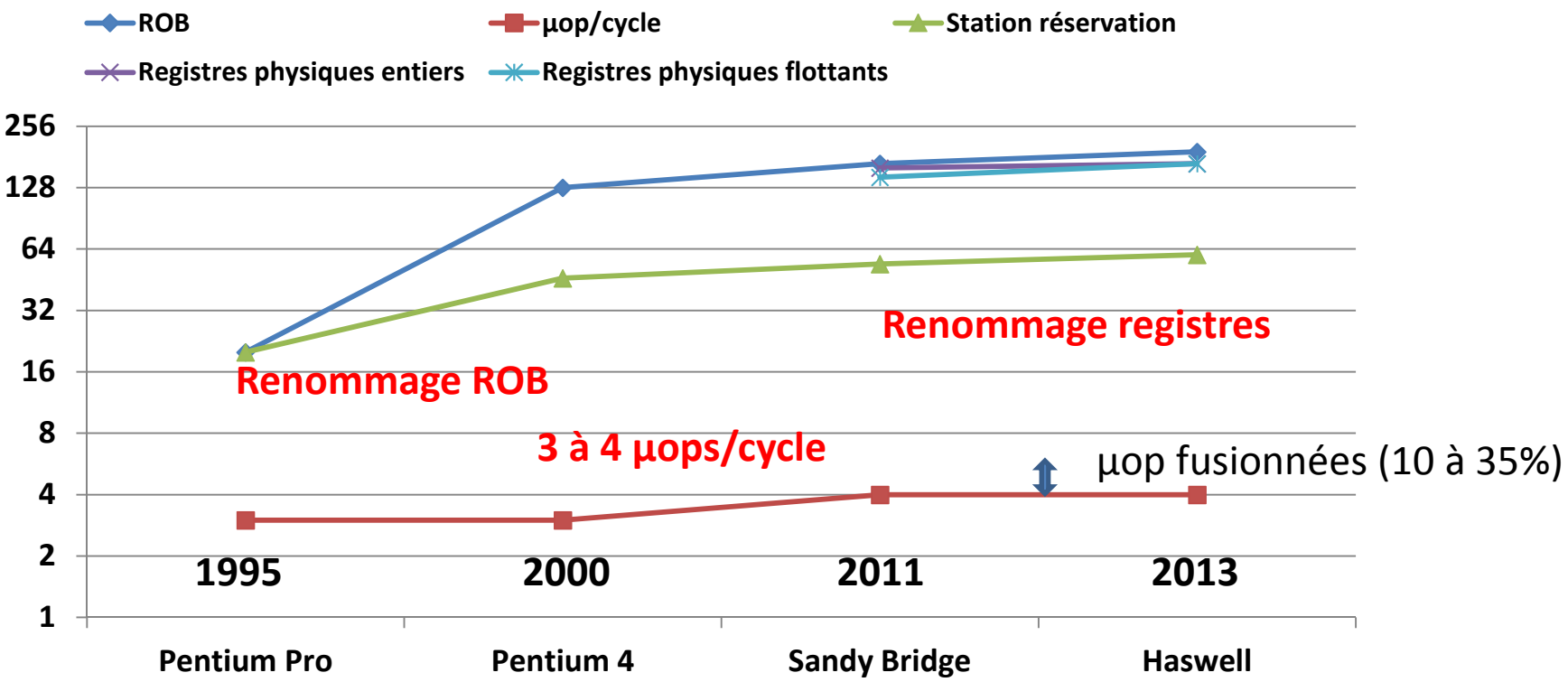
1970: Flynn

Weiss and Smith [1984]	1.58
Sohi and Vajapeyam [1987]	1.81
Tjaden and Flynn [1970]	1.86 (Flynn's bottleneck)
Tjaden and Flynn [1973]	1.96
Uht [1986]	2.00
Smith et al. [1989]	2.00
Jouppi and Wall [1988]	2.40
Johnson [1991]	2.50
Acosta et al. [1986]	2.79
Wedig [1982]	3.00
Butler et al. [1991]	5.8
Melvin and Patt [1991]	6
Wall [1991]	7 (Jouppi disagreed)
Kuck et al. [1972]	8
Riseman and Foster [1972]	51 (no control dependences)
Nicolau and Fisher [1984]	90 (Fisher's optimism)

Limites de l'IPC (CPU)



- Parallélisme d'instructions
  - Nombre d'instructions exécutables/cycle
- **Evolution très limitée de 1990 à 2015 au niveau matériel pour les processeurs « non ordonnés » d'Intel**



Améliorer la performance : $T_{ex} = \frac{NI}{IPC * F}$

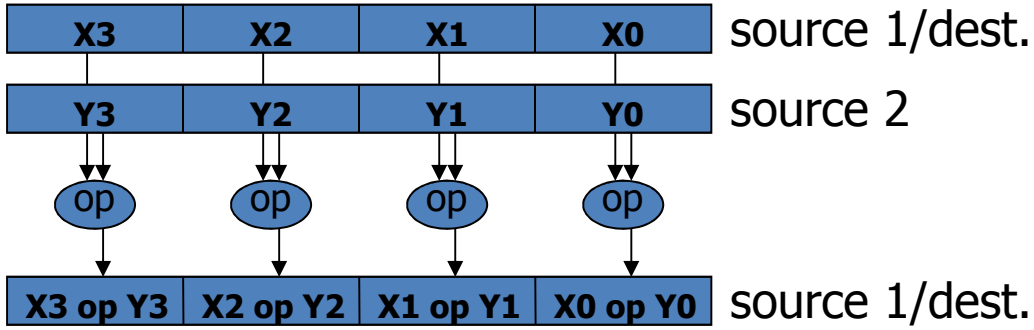
Le parallélisme de données dans les monoprocesseurs

CPU

SIMD

1 instruction avec plusieurs données

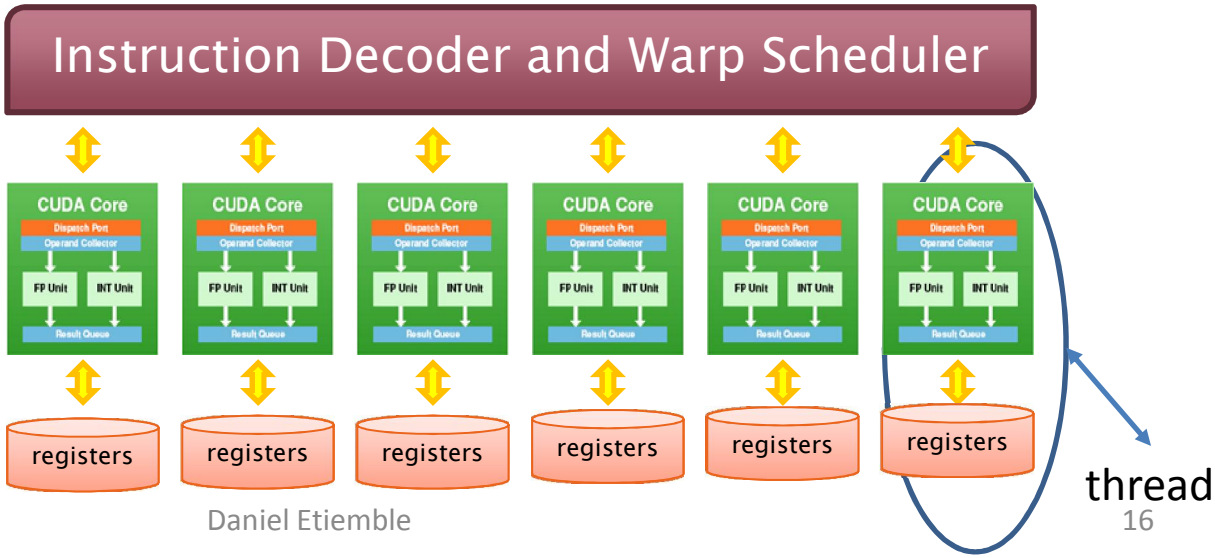
SSE2/3/4 – Neon – Altivec
AVX – AVX2 – AVX 512…



GPU

SIMT

1 instruction pour plusieurs threads



Les instructions SIMD

- Intel
  - 64 bits : MMX (1997)
  - 128 bits : SSE (1999), SSE2 (2000), SSE3 (2004), SSE4 (2006)
  - 256 bits : AVX (2008), AVX2 (2013)
  - 512 bits : AVX-512 (2013)
- ARM : VFP, Neon (2009)
- PowerPC : Altivec (1999)

$$T_{ex} = \boldsymbol{NI} * CPI * T_c = \frac{\boldsymbol{NI}}{IPC * F}$$

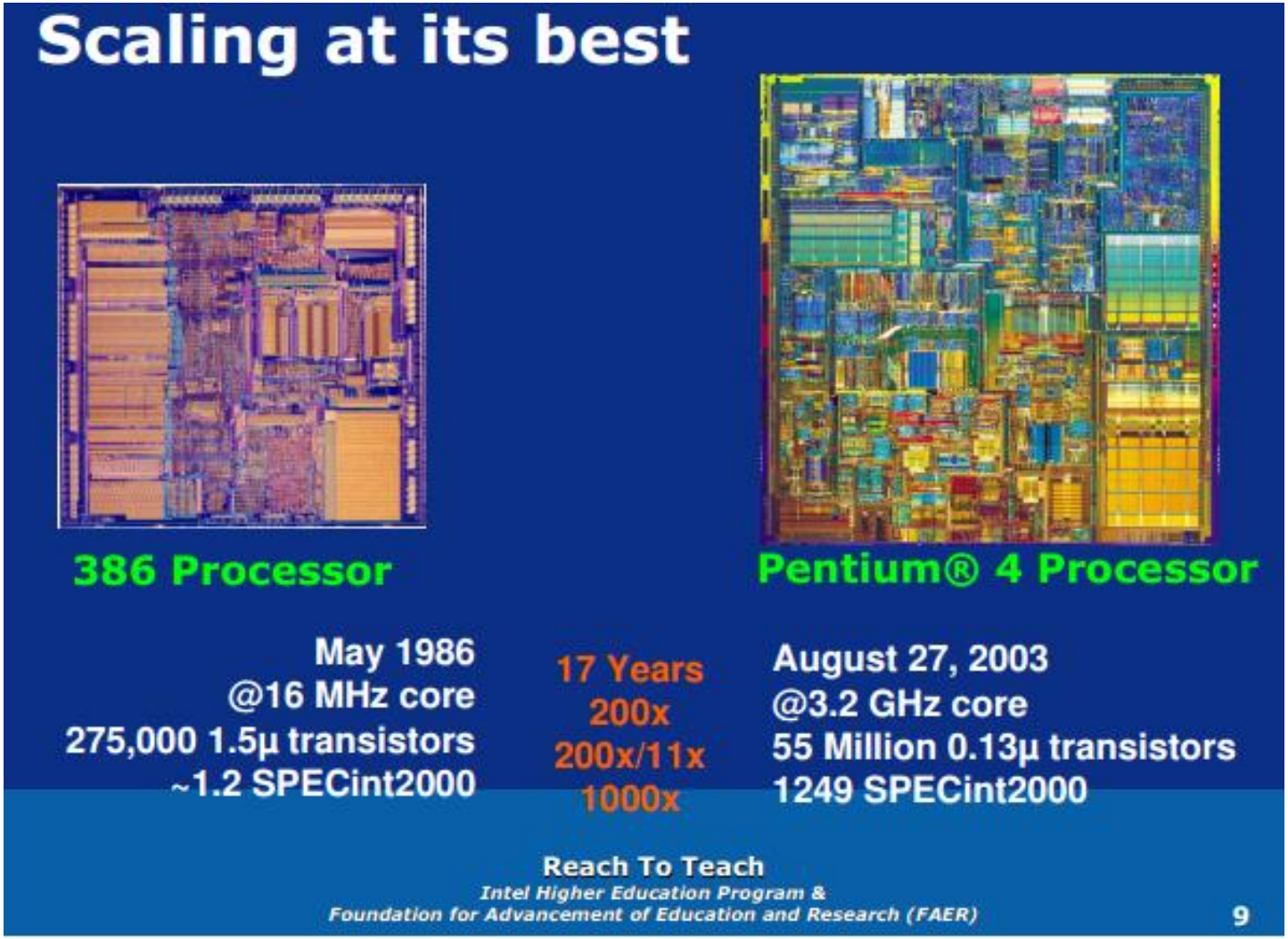
- **Seule manière en 2018 d'augmenter les performances monoprocesseurs**

Free lunch… (par Intel)

F

IPC

NI (SIMD)



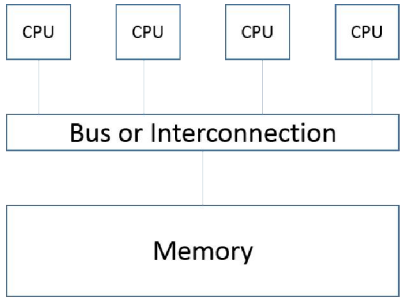
Améliorer la performance : $T_{ex} = \frac{NI}{IPC * F}$

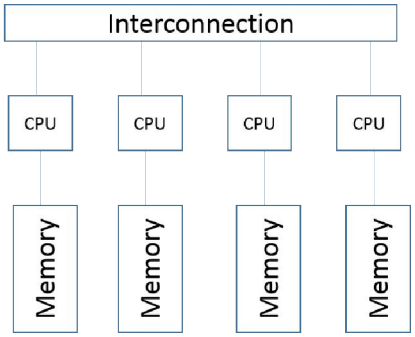
Parallélisme de données ou de threads dans les architectures parallèles

Répartir NI sur plusieurs processeurs (ou cœurs)

- Sauf pour les cas simples ou les applications “embarrassingly parallel”, la répartition dépend de l’architecture, du modèle de programmation, de la loi d’Amdhal, etc.
- Dans certaines architectures, les temps de communication s’ajoutent

OpenMP
Pthreads





MPI

Multiprocesseurs Multi-ordinateurs

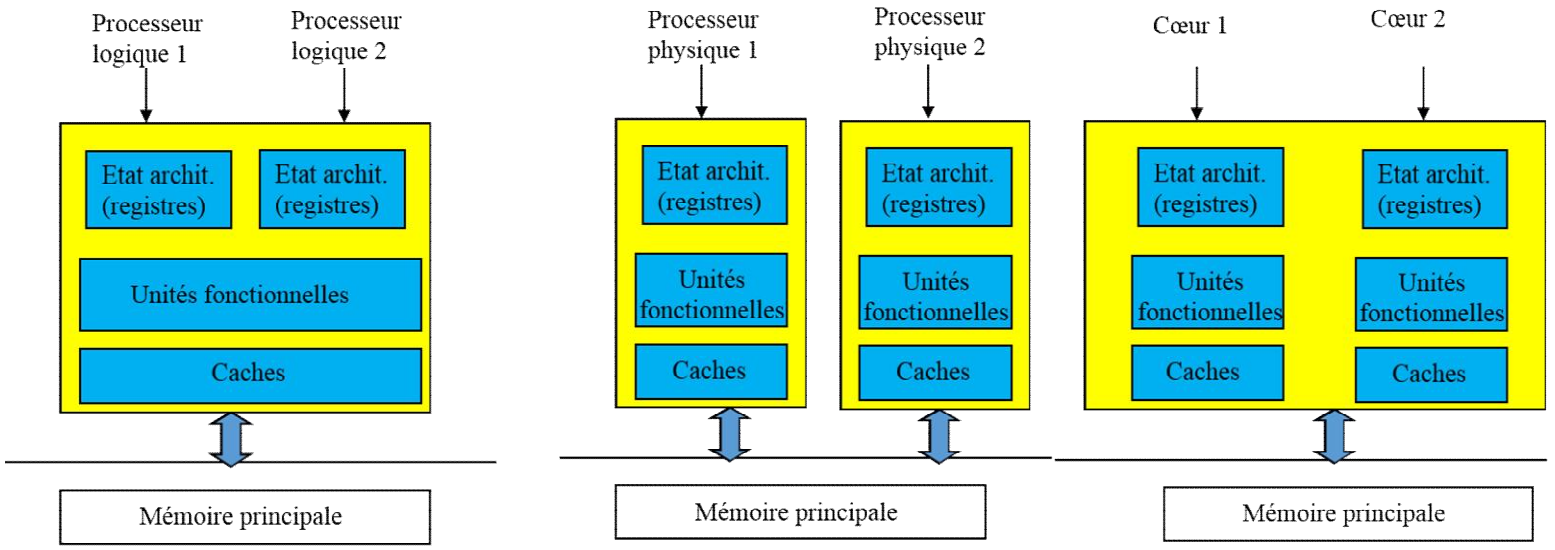
De la programmation séquentielle à la programmation parallèle

- Limité aux serveurs et aux superordinateurs dans la période du “free lunch”

Le virage vers le parallélisme

De l'augmentation de la fréquence d'horloge… au parallélisme

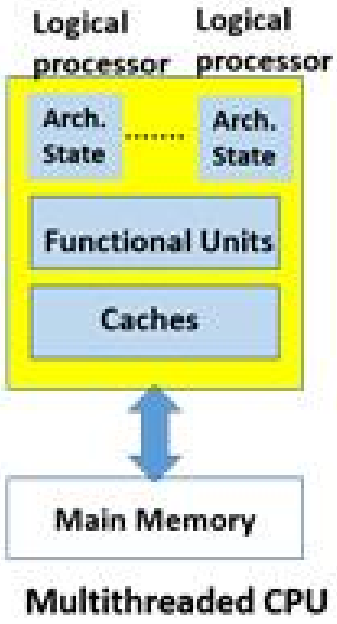
- Hyperthread/Multithread
- Multi-cœurs



CPU multithreads

$$T_{ex} = \frac{NI}{(IPCCP_U + IPC_{Mem}) * F}$$

- Programmes séquentiels
- Plusieurs programmes (multiprogrammation): NI = $\sum NI_i$
- Plusieurs threads (TLP) NI= $\sum NI_i$



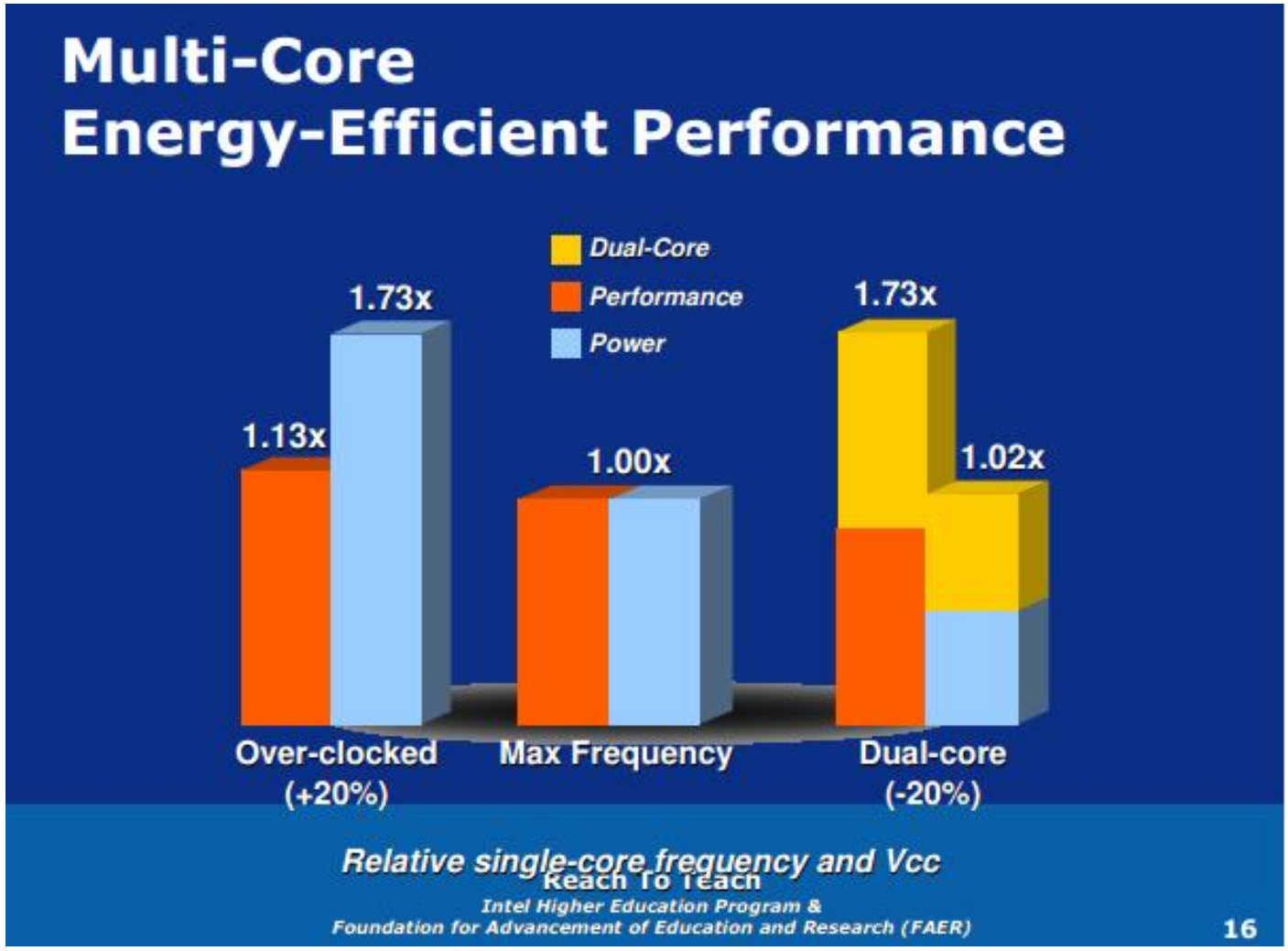
Multithreading grain fin
- Commuter en un cycle d'un thread au suivant sur un aléa du pipeline (défauts cache, instructions longues…)
- Sun Niagara, Oracle servers

Multithreading simultané
- Lancer des instructions de plusieurs threads à chaque cycle
- Intel Hyperthreading (2), IBM Power (2 to 8)

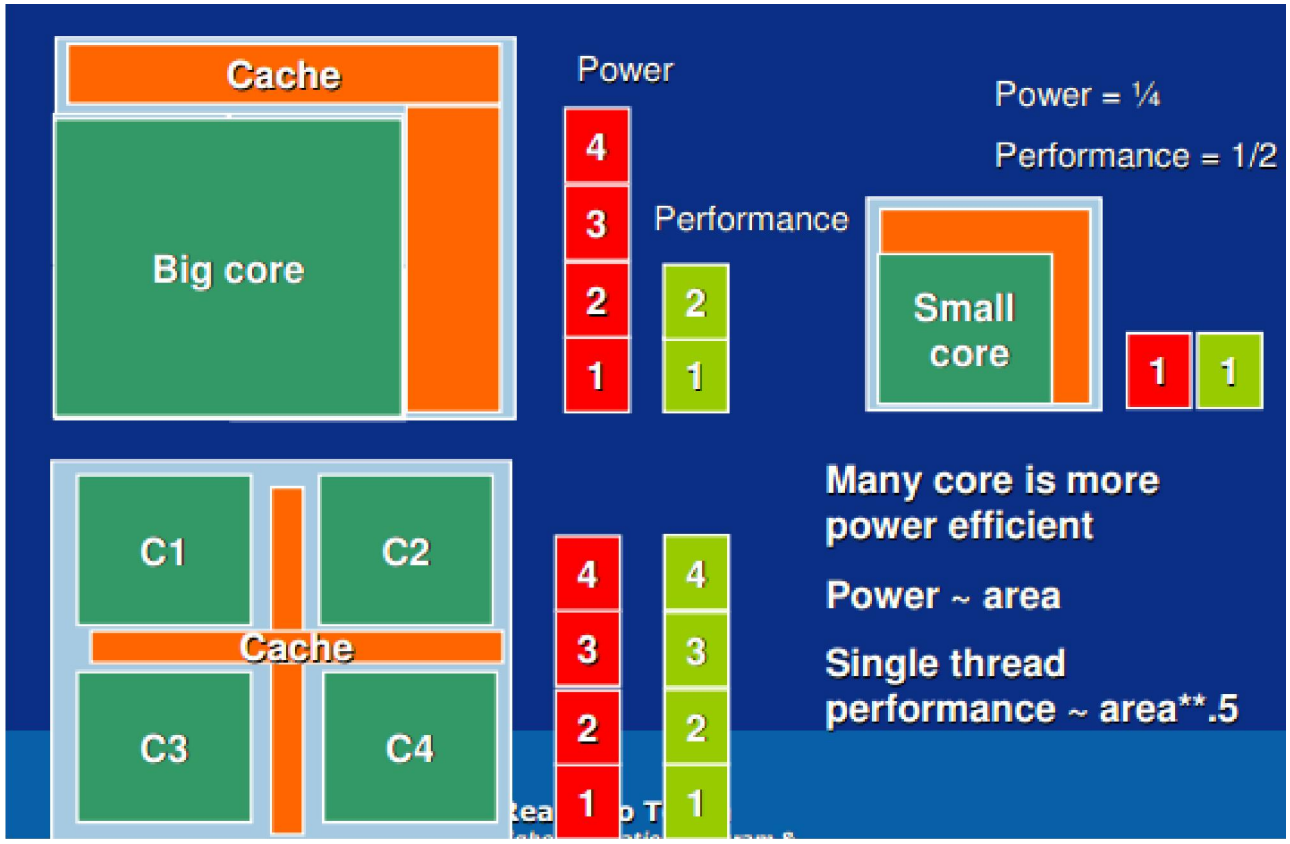
Multithreading réduit CPI_{Mem}

→ **Multi-coeurs avec des coeurs multithreads**

Pourquoi les multi-cœurs ?



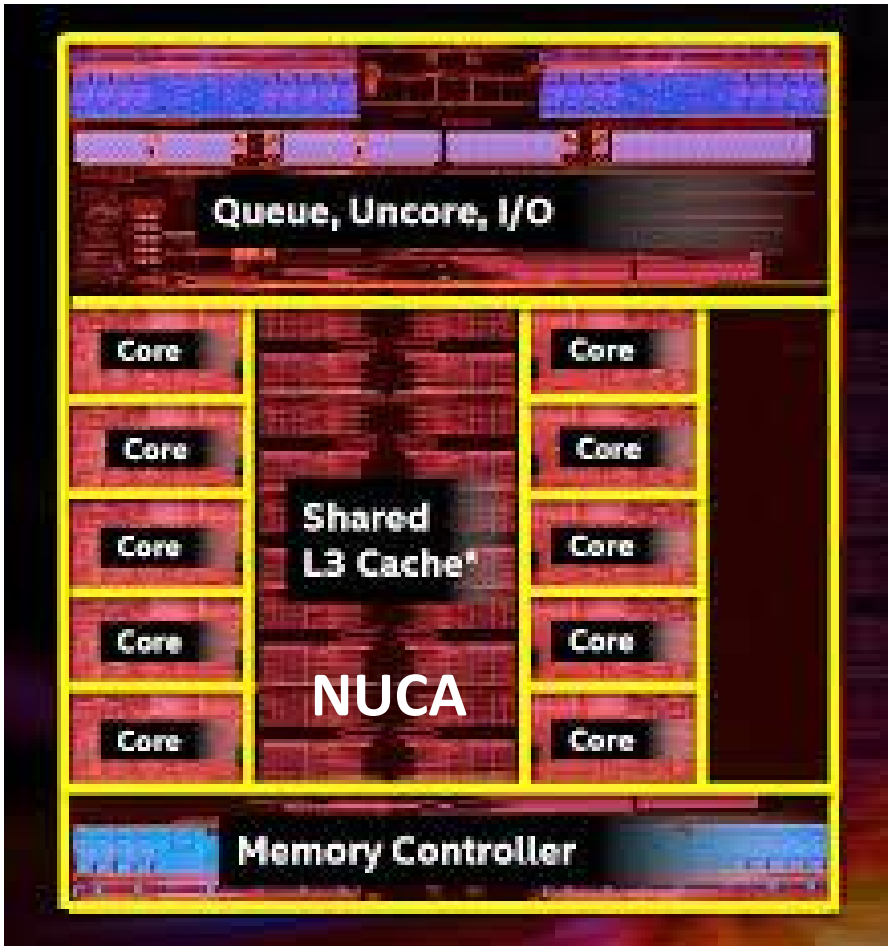
Efficacité énergétique des multi-cœurs



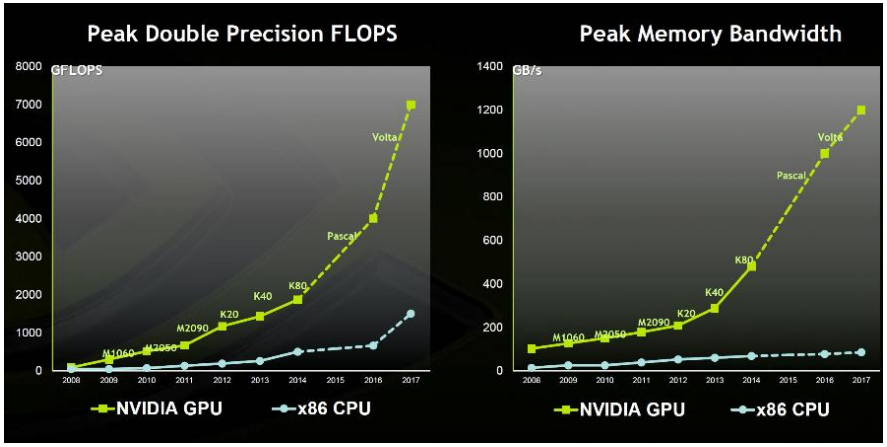
Un multi-cœur Intel de 2016

Broadwell-E

2016
14 nm
3,4 10^9 T
2,46 cm^2

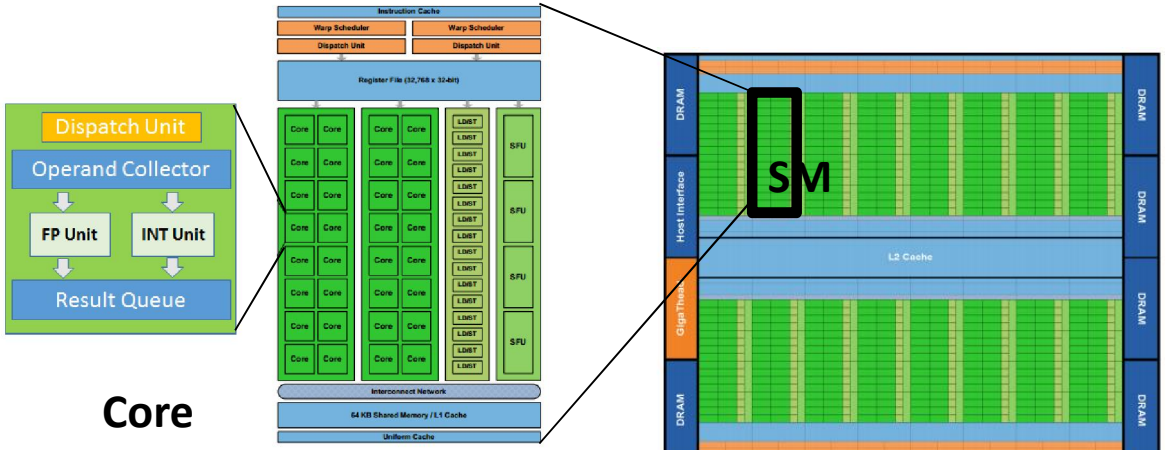


Et les GPU ?



Parallélisme de données massif

Très grand nombre de cœurs



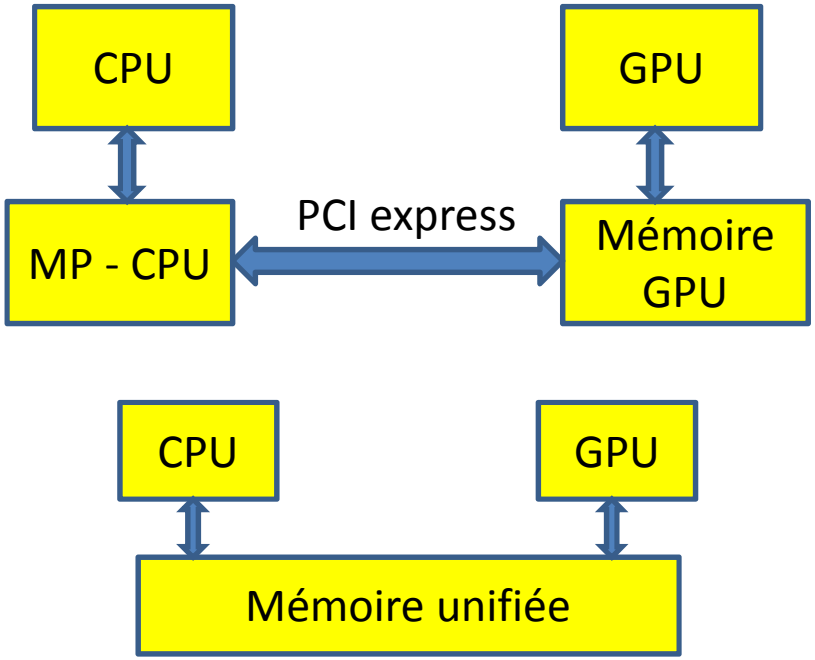
Core

Streaming Multiprocessor

Fermi GPU

CPU + GPU

- GPU = coprocesseur
- Deux modèles de programmation différents
- Un ou deux circuits?
  - CPU + GPU or APU



2014	Tesla K40 + CPU	Nvidia Tegra K1
Single Precision Peak	4.2 TeraFlops	326 GFlops
Single Precision SGEMM	3.8 TeraFlops	290 GFlops
Memory	12GB @ 288GB/s	2GB @ 14.9GB/s
Power (CPU + GPU)	~ 385Watt	<11Watts
Performance Per Watt	10SP GFlops Per Watt	26SP GFlops Per Watt

La seconde équation

- Puissance dissipée CMOS

$$P_d = V_{dd} * I_{fuite} + \alpha * \sum C_i * V_{dd}^2 * F$$

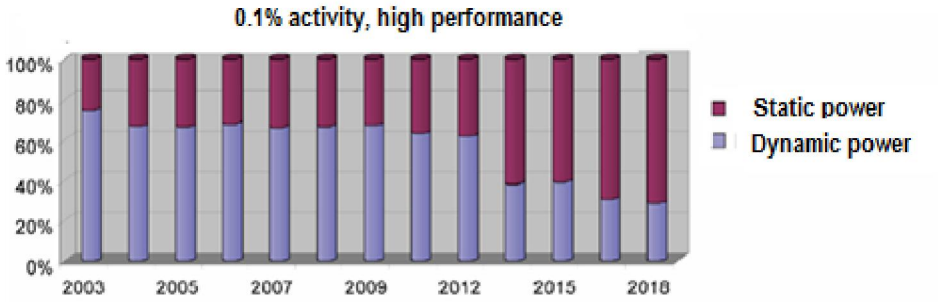
= 0 (autrefois !)

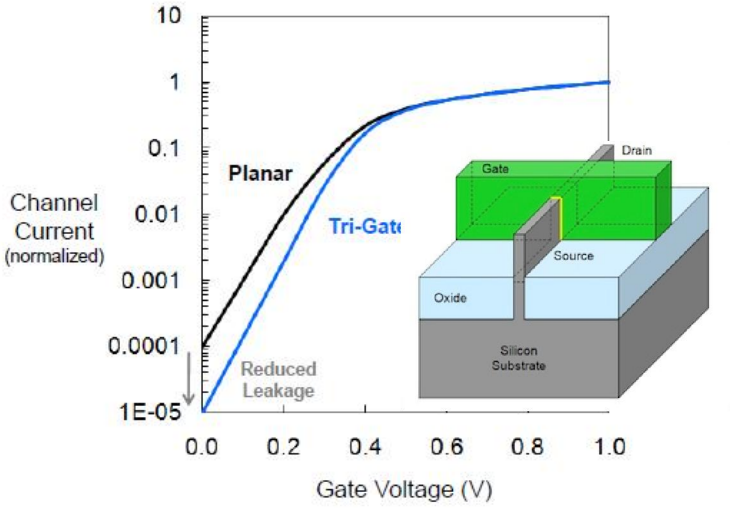
≠ 0

  - V_{dd} : tension d'alimentation
  - C_i : capacités
  - F : Fréquence
  - α : facteur d'activité

Réduire la puissance statique

$$P_{d\ statique} = V_{dd} * I_{fuite}$$





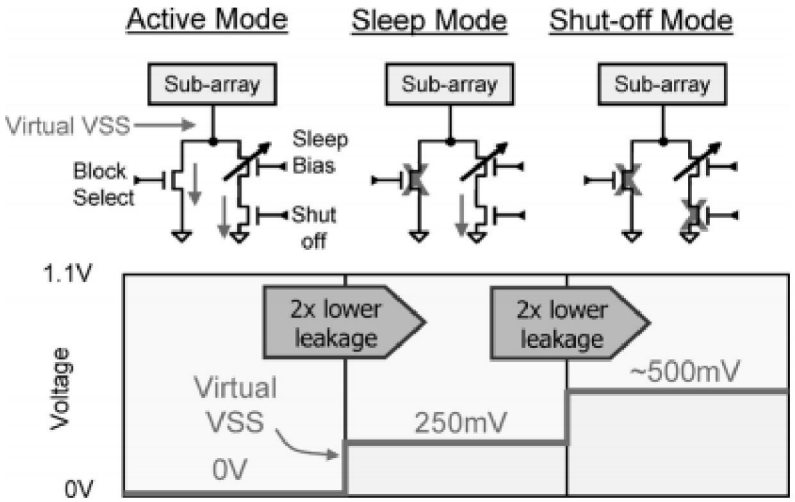
TECHNOLOGIE

- Ex: Intel Tri-gate

CIRCUITERIE

- Ex: Masses virtuelles

Cache L3 du CPU Xeon 65-nm



Réduire la puissance dynamique

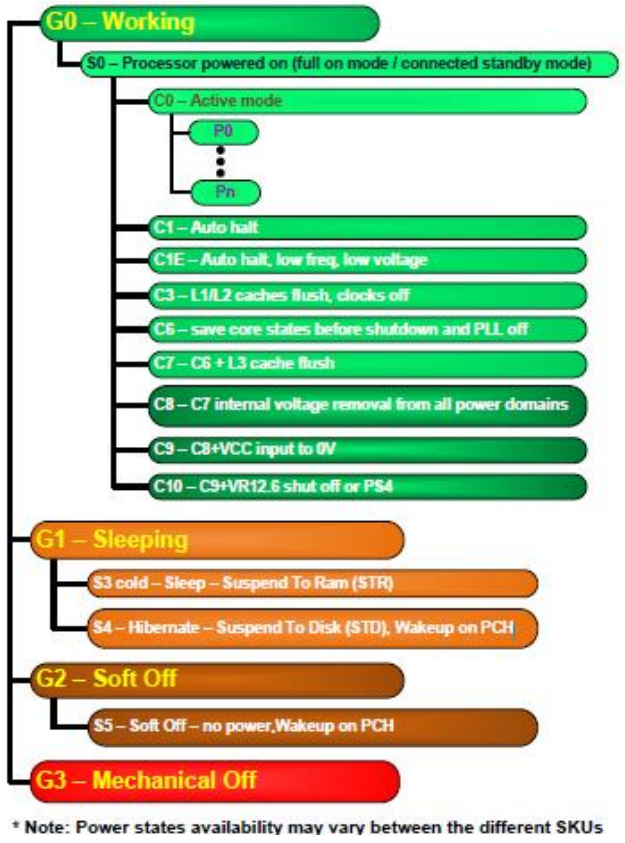
- $P_d = V_{dd} * I_{fuite} + \boldsymbol{\alpha} * \sum C_i * \boldsymbol{V_{dd}^2} * \boldsymbol{F}$
- Circuiterie
  - « *clock gating* » : n'activer que les parties utiles
  - Plusieurs tensions d'alimentation
  - Transistors rapides, moyens et lents (tension de seuil Vt)
- **Architecture**
  - **Fréquence d'horloge (F)**
  - **Découpage en domaines (α, V_{dd}, F)**

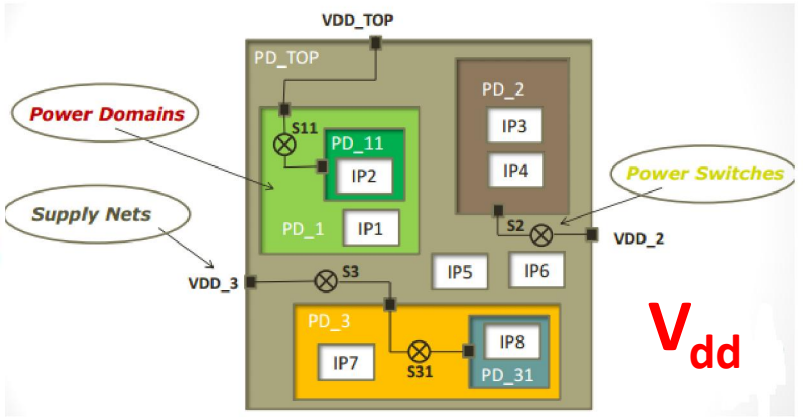
Réduire la puissance dynamique

$$P_{d\ dynamic} = \alpha * \sum C_i * V_{dd}^2 * F$$

α

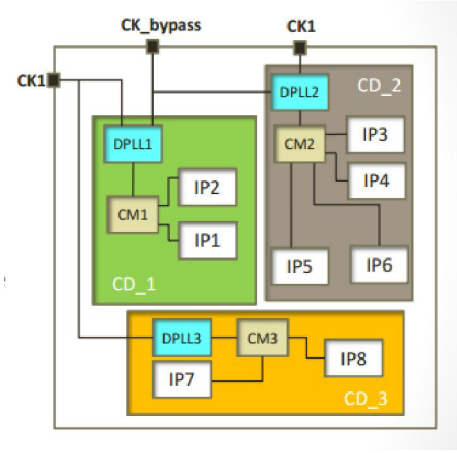
Plusieurs modes de fonctionnement par bloc :
Ex: 5ème génération Intel Cores





V_{dd}

Domaines de puissance



F

Domaines d'horloge

N’activer que les parties utiles

Plusieurs modes de fonctionnement α

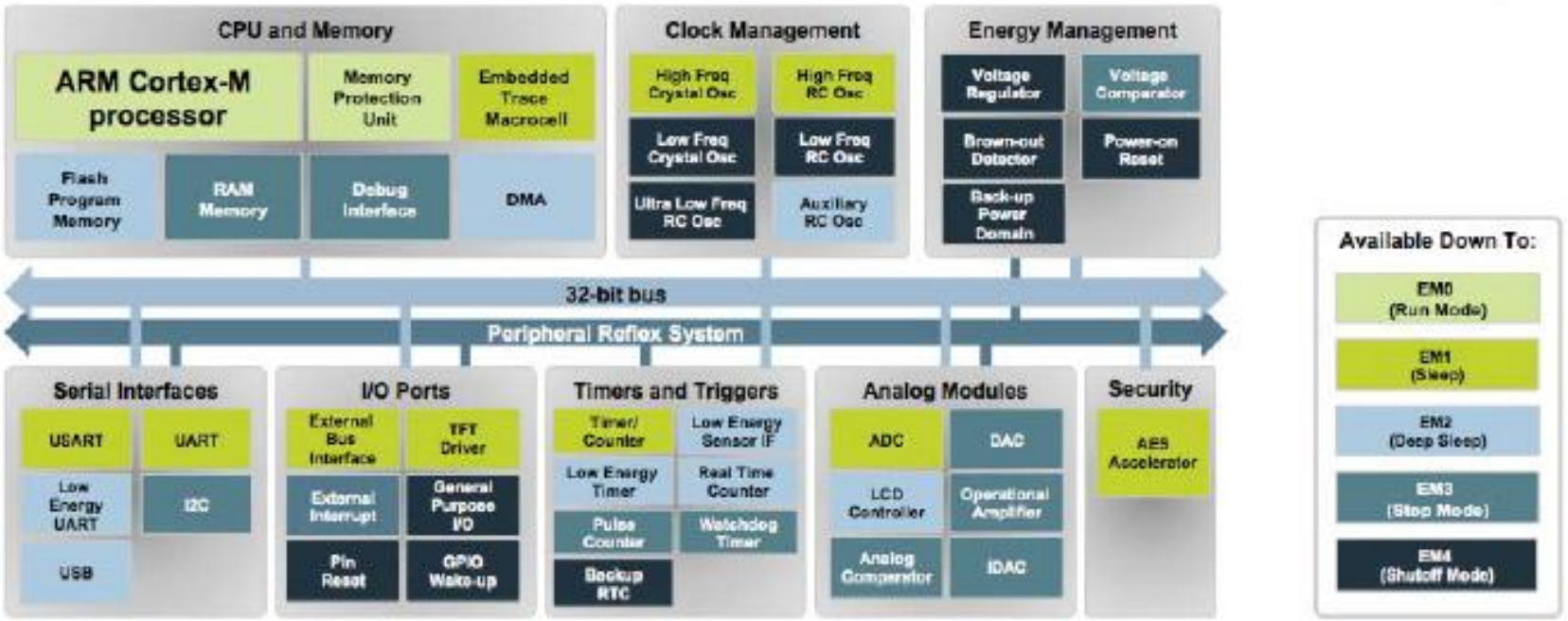
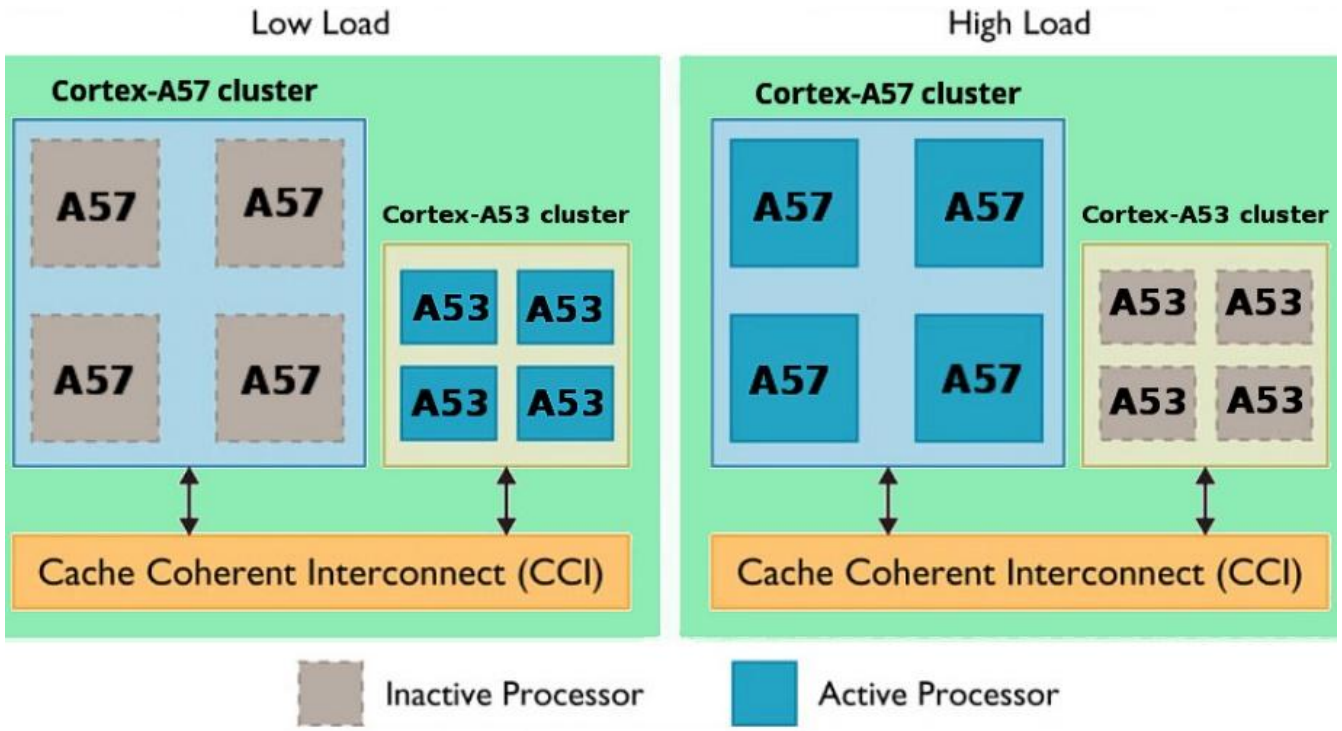


Figure 2. EFM32 Gecko block diagram and the five energy modes.

Adapter le fonctionnement à la charge

α



Circuit Exynox 7420 (Galaxy S6)

Et le futur?

- La loi de Moore : les limites fondamentales sont proches
  - Mais les nœuds 7 nm, 5 nm et 3 nm ont été déjà annoncés !
- Temps d'exécution :
  - F: Modifications importantes peu probables
  - IPC:
    - Limites de l'ILP dans les programmes et exploitable dans les cœurs
    - Nouvelles architectures PIM? (Problèmes du mur mémoire)
  - NI continue de diminuer
    - Plus de travail par instructions
      - Largeur SIMD (256-512-1024…) et taille de données (F16)
      - Nouvelles instructions 2D: *Tensor cores* (Nvidia), Unité de multiplication matricielle (Google TPU), *Intelligent Processing Unit* (Graphcore) …
    - Plus de cœurs
      - Des multi-cœurs aux many-cores. Croissance exponentielle du nombre de cœurs ???
- Puissance dissipée :
  - Tant que le CMOS sera utilisé…

Remarques pour conclure

- La tendance principale (pas les détails) de l'évolution des CPU peut être expliquée par
  - La loi de Moore
  - $T_{ex} = IC * (CPICPU + CPIMem) * T_c = \frac{IC}{IPC * F}$
  - $P_d = V_{dd} * I_{leakage} + \alpha * \sum C_i * V_{dd}^2 * F$
- Valable pour les processeurs programmables aussi longtemps que la technologie CMOS sera utilisée
- Les architectures mixtes HW-SW (FPGA) sont plus difficiles à modéliser.
- L'évolution des architectures de CPU est influencée par de nouvelles applications (IA, IoT, etc.).
- Performance par watt….
- Business.

Questions ?