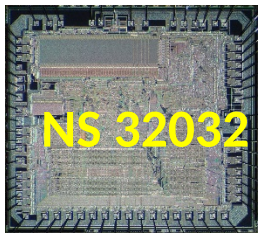
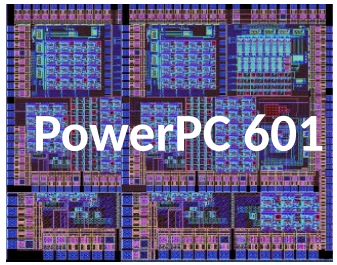
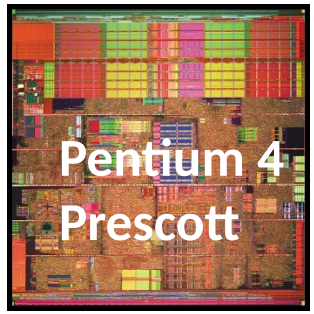


1983



1992



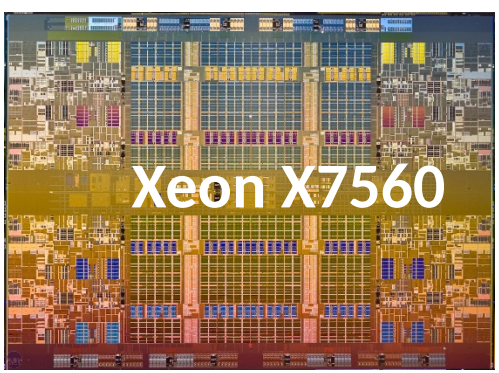
2004

45 ans d'évolution des CPU

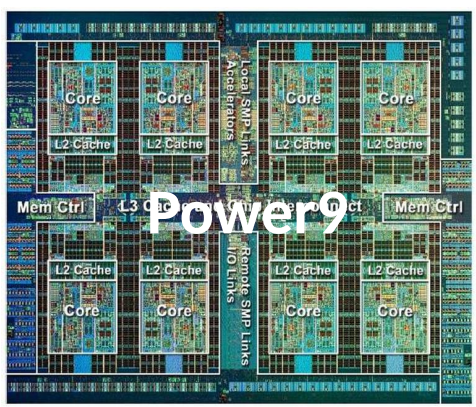
Moore et 2 équations.

Daniel Etiemble

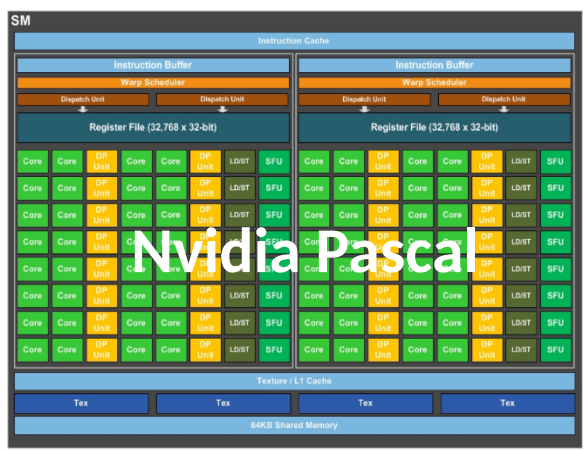
LISN– Université Paris Saclay



2010



2017



2016

Plan

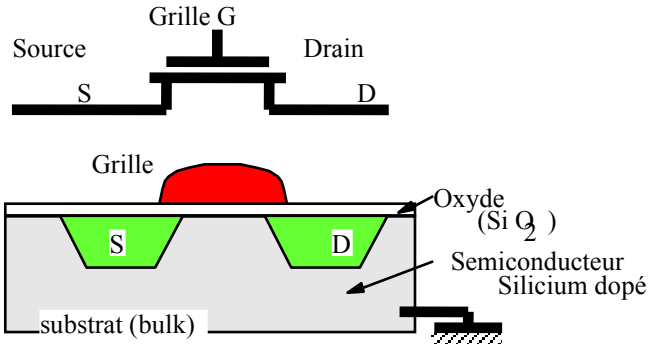
- La préhistoire
- 1 loi et 2 équations
- Améliorer les performances (NI, IPC, F)
- Du monoprocesseur aux multi-coeurs
- Limiter la puissance dissipée
- Pour conclure

La préhistoire

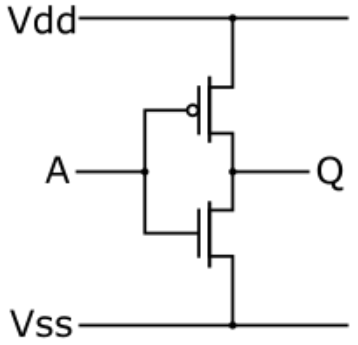
- Avant la technologie MOS (années 70)
- Beaucoup de concepts sont déjà là
  - Pipeline : IBM 7030 (1959)
  - Superscalaires : CDC 6600 (1966)
  - Caches : IBM 360/85 (1968)
  - Stations de réservation : IBM 360/91 (1969)
  - Mémoire virtuelle : Atlas, Multics, etc. (années 60)
  - Machines parallèles : Burroughs D825 (1962)…
  - Etc.

Technologie MOS et CPU

- PMOS : Intel 4004 (1971)
- NMOS : Intel 8080 (1974)
- CMOS : RCA 1802 (1976), 80C286 (Intersil 1982), Intel depuis 80386 (1985)…



Transistor MOS



Inverseur CMOS

Evolution exponentielle

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

$$Tex = NI * (CPI\text{-}CPU + CPI\text{-}Mem) * Tc$$

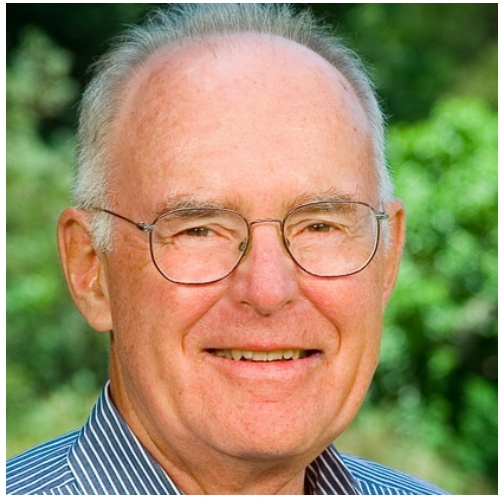
Nb d'instructions — NB Cycles/Instruction (Calcul – Attente mémoire) — Temps de cycle

  - Puissance dissipée CMOS

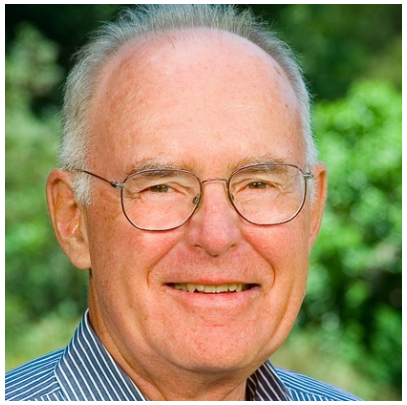
$$P_d = V_{dd} * I_{fuite} + \alpha * \sum C_i * V_{dd}^2 * F$$

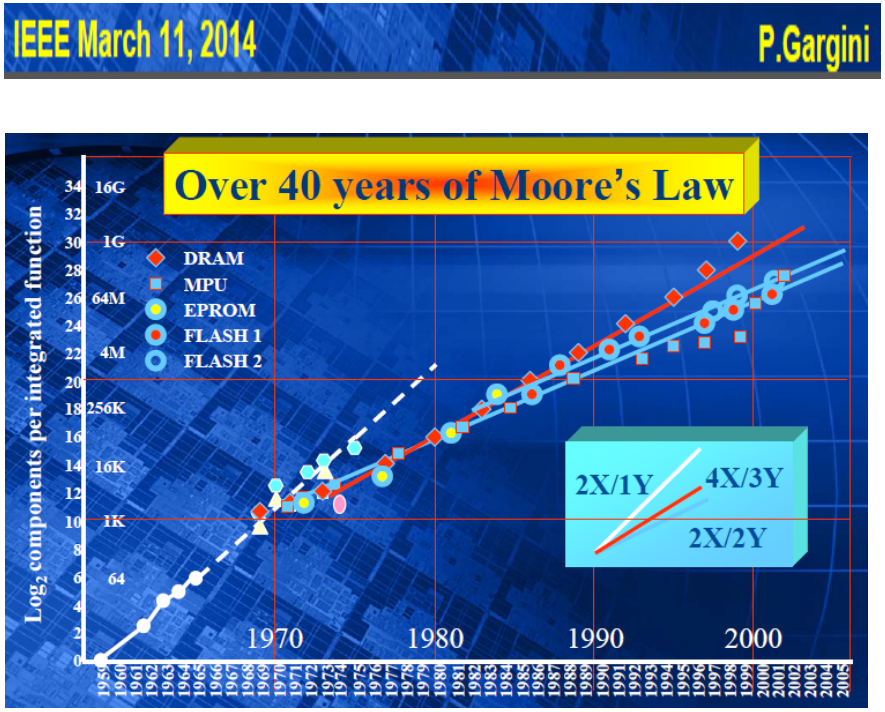
La loi de Moore : une **exponentielle**

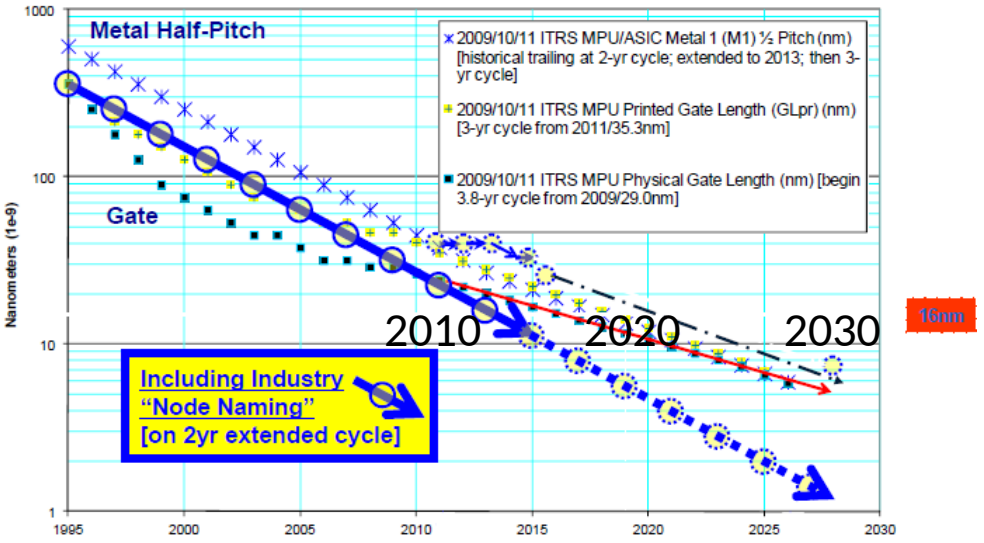
- Version correcte
  - Le nombre de transistors par puce double tous les 12 / **18** / 24 **mois**
- Versions « incorrectes »
  - Doublement tous les deux ans
    - De la performance
    - Du nombre de cœurs par puce
    - Etc.



La loi de Moore





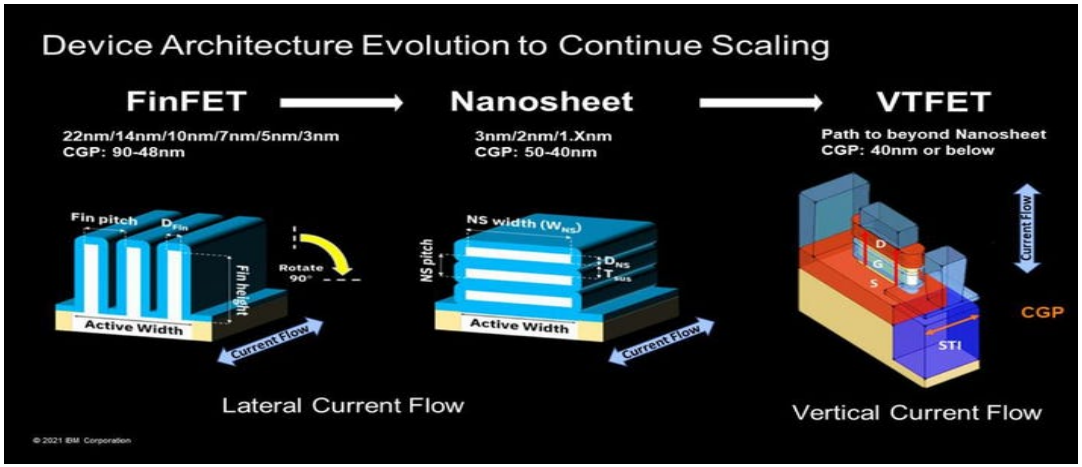


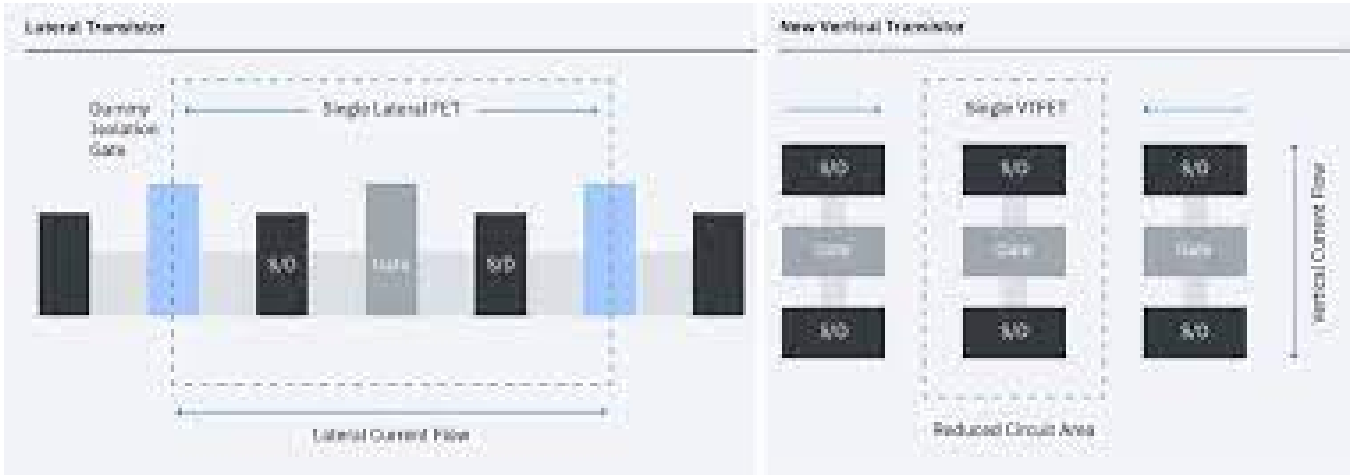
Le Nb de transistors par circuits double tous les N mois (12/18/24)

N augmente

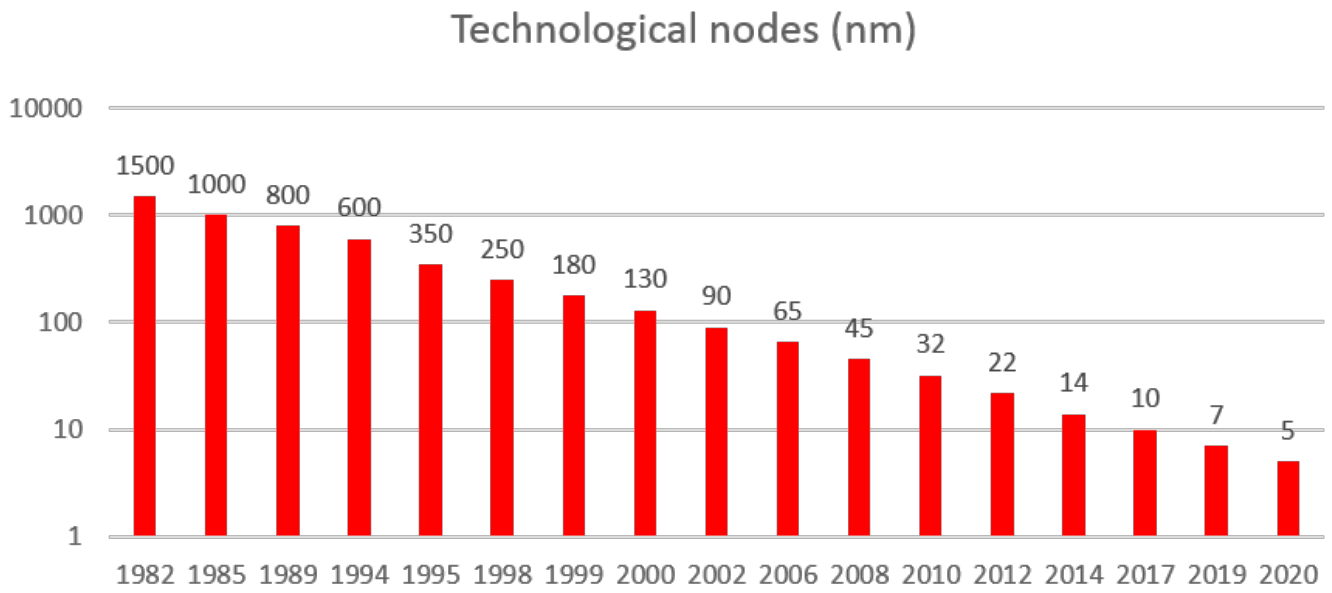
De nouveaux progrès (2021)

Les FET verticaux (IBM et Samsung)



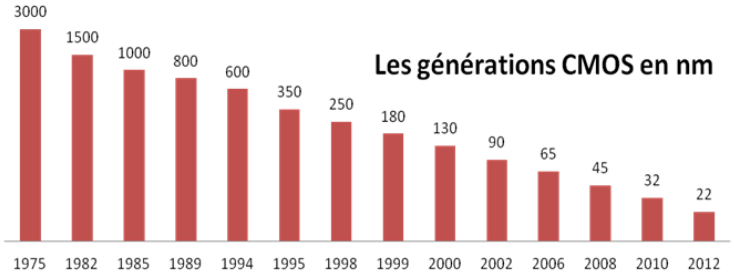


Les « nœuds » technologiques



- 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.

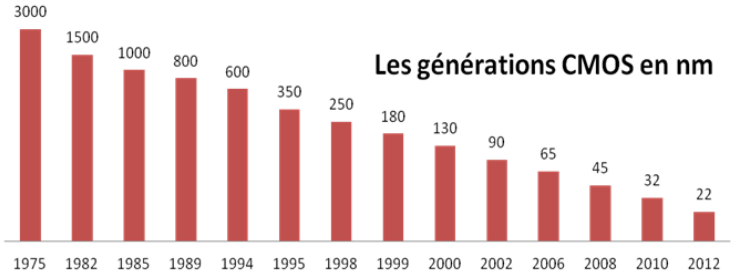
« Scaling »



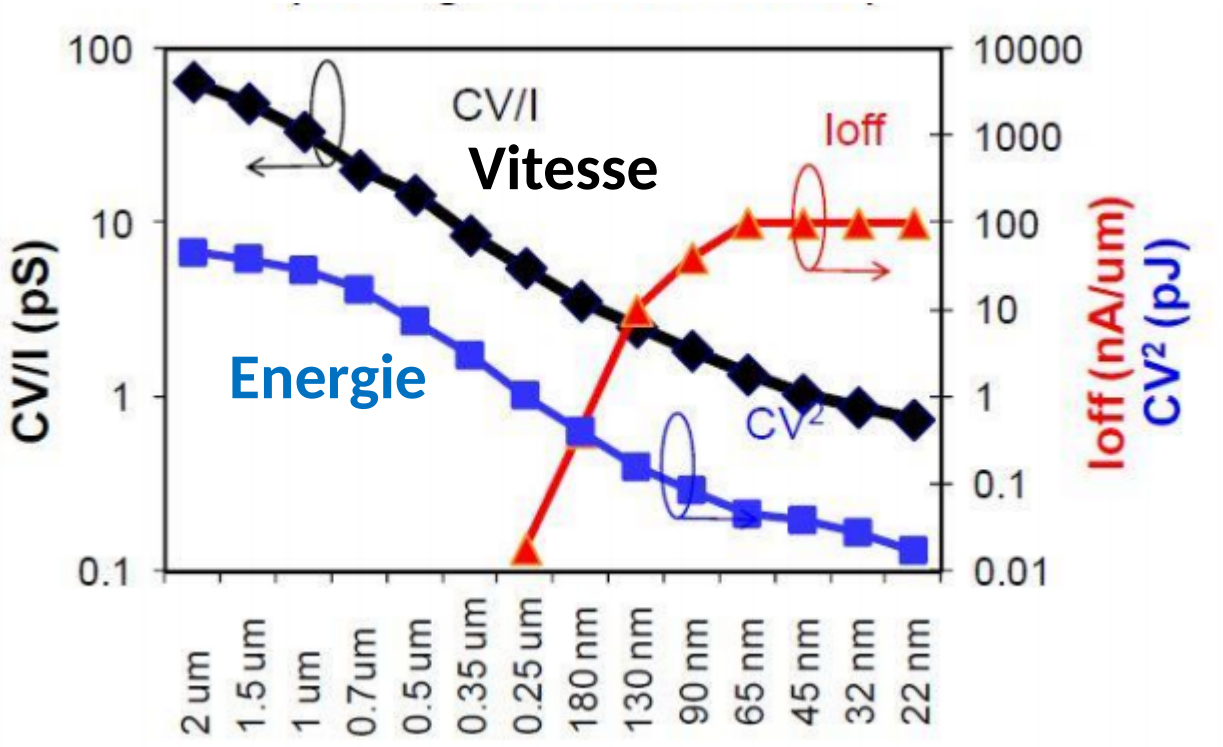
- Le bon vieux temps (R. Dennard, IEEE JSSC, 1974)
  - Réduction d'un facteur $\lambda = 0.7$ soit $\lambda^2 \approx 1/2$

Paramètre	Scaling total	Tension fixée	Scaling « saturé »
Dimensions (W, L, tox)	λ	λ	λ
Surface	λ^2	λ^2	λ^2
V_{dd}	λ	1	1
Courant	λ	$1/\lambda$	1
Capacité	λ	λ	λ
Délai	λ	λ^2	λ
Densité puissance	1	$1/\lambda^3$	$1/\lambda^2$

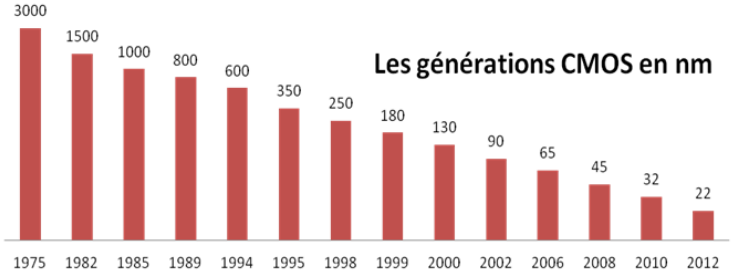
« Scaling »



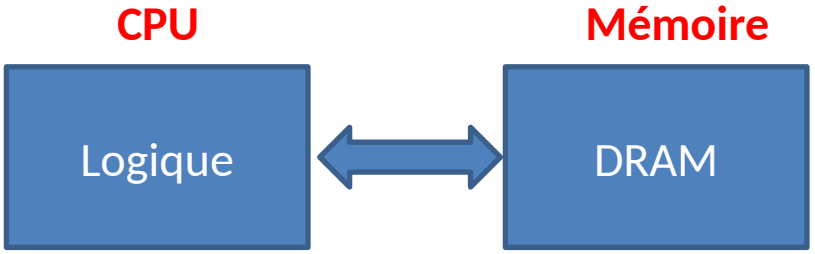
- Plus réaliste



« Scaling »

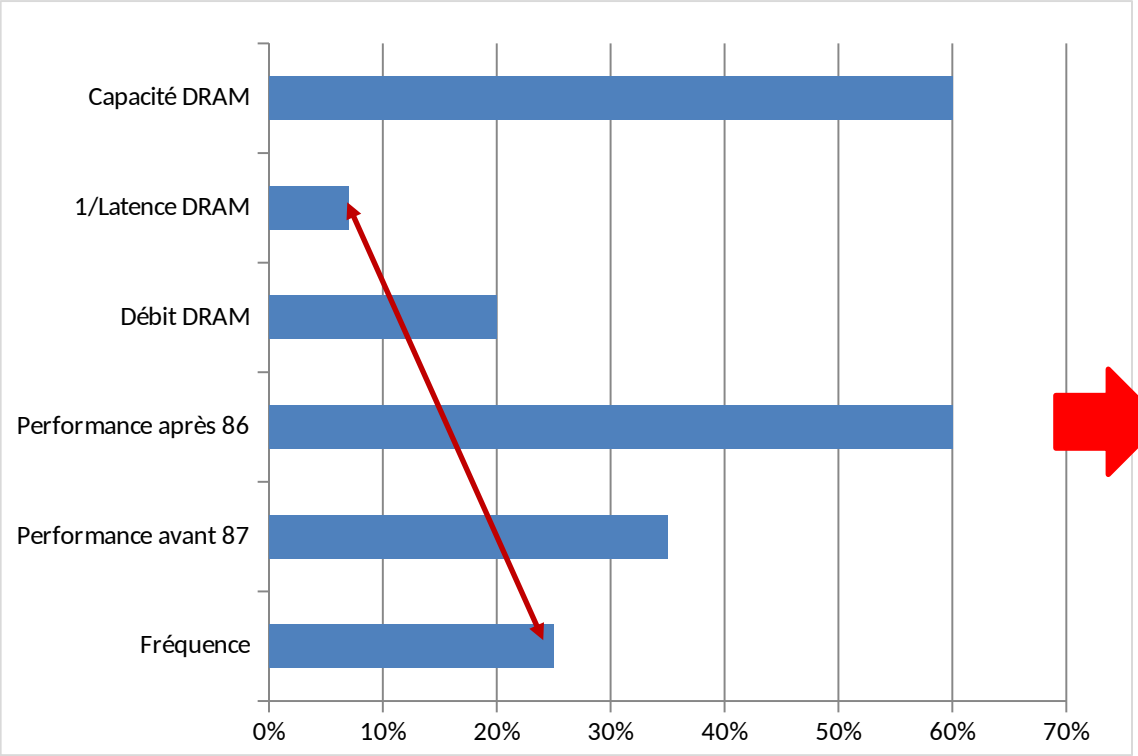


- Surface :
  - Nb Transistors/unité de surface : x2
- Vitesse des portes
  - Délai : /1,4
  - *Mais impact croissant des interconnexions*
- Mais



Mais différentiels d'exponentielles

- Evolution par an



Hiérarchies de caches

L’utilisation des transistors

- Intégrer plus de fonctions dans une même puce
  - De deux puces à une puce
- Augmenter les « capacités » d’une fonction
  - Ex : taille des caches
- Ajouter de nouvelles fonctions

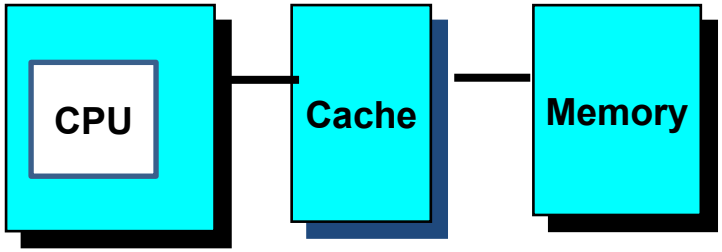
Une ou deux puces

- 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

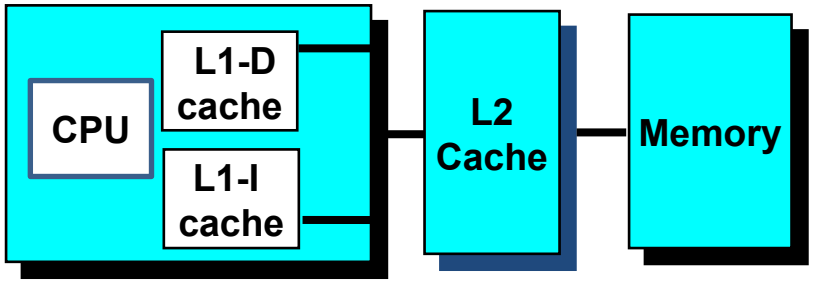
Du coprocesseur à l'intégration dans la même puce

De puces distinctes à une seule puce

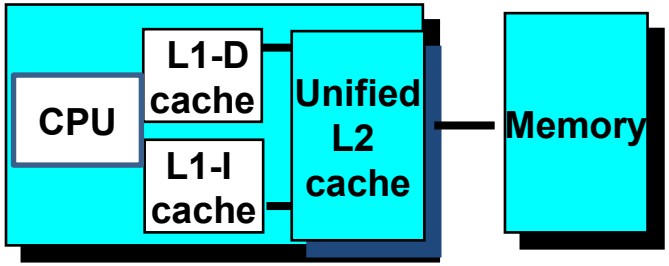
Evolution des hiérarchies de cache



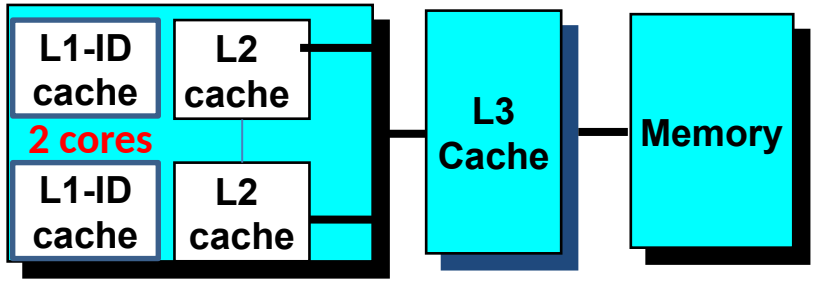
386



Pentium



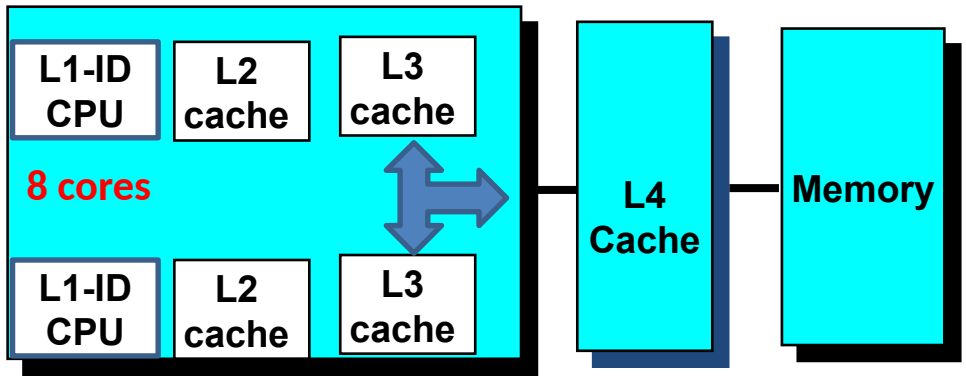
Pentium 4



Power6

Latence et

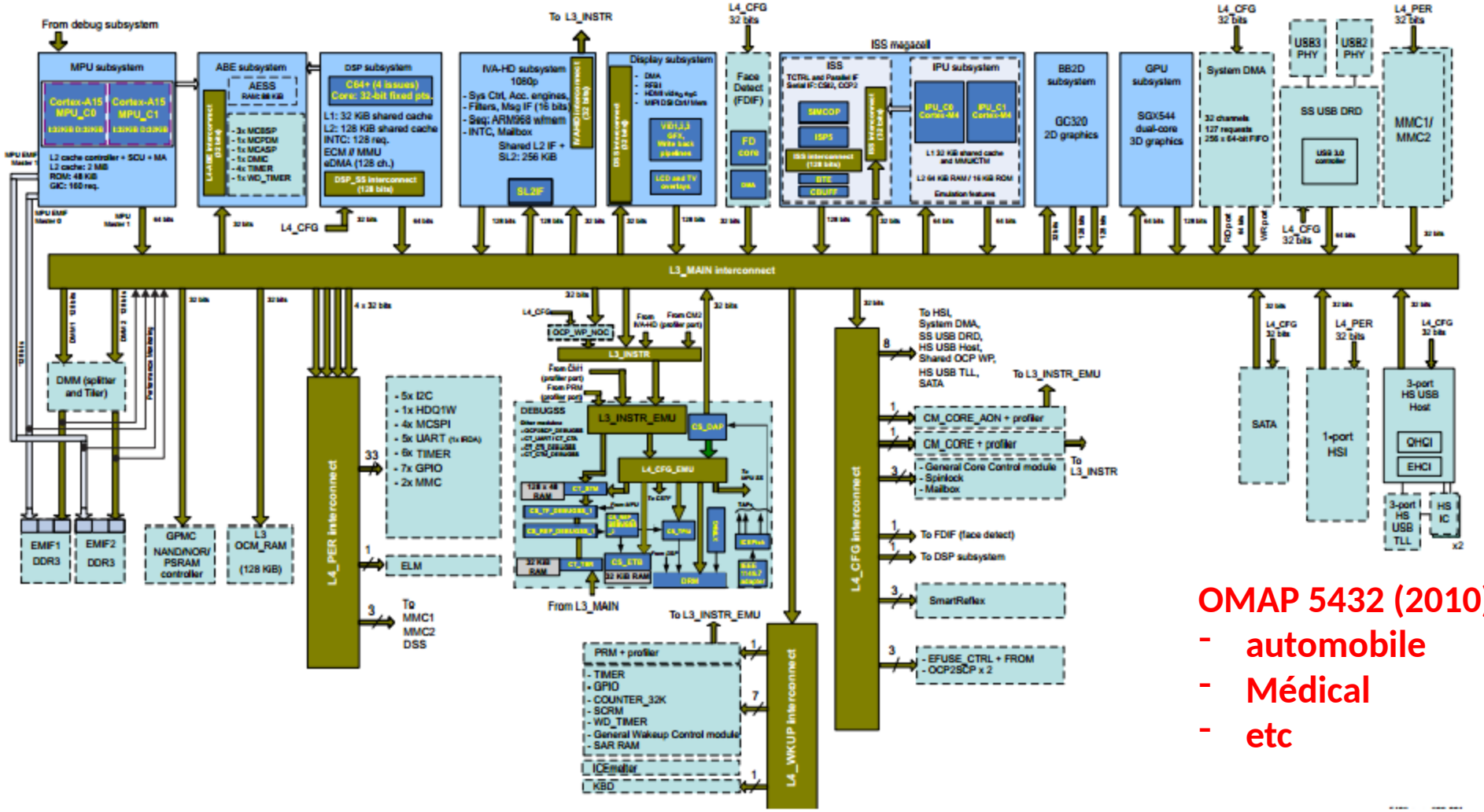
Bande passante



Power 8

NUCA

Ajouter des fonctionnalités

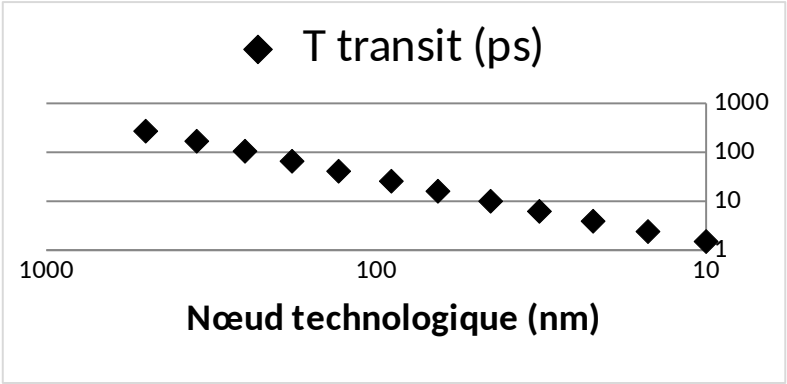


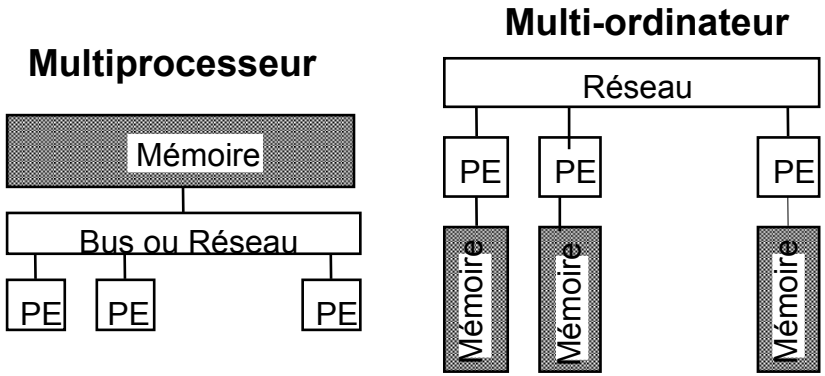
OMAP 5432 (2010)

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

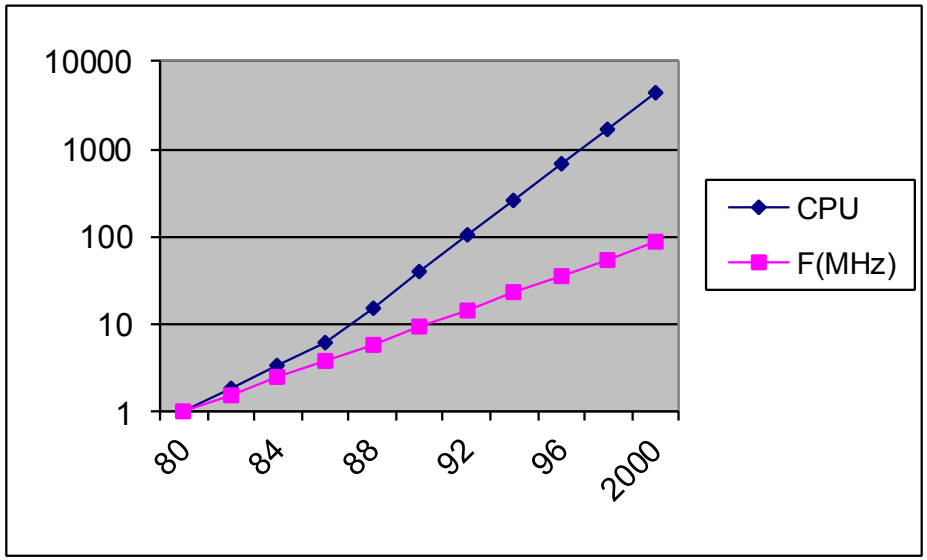
Améliorer les performances

- L'équation Tex=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 les performances



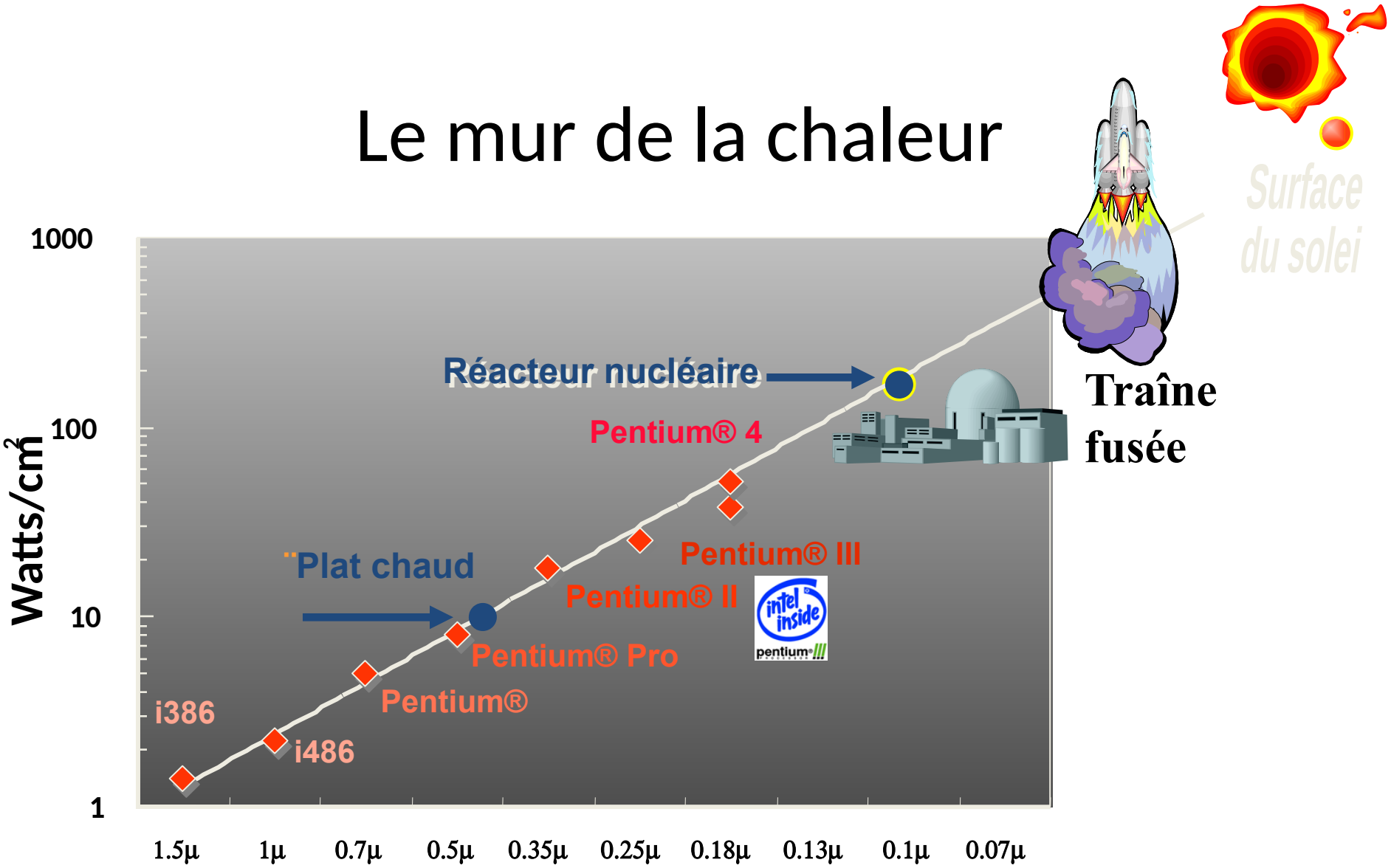
Tex = NI / (IPC * **F**)

- Environ 25% par an… jusqu'à 4 GHz en 2017
  - Sauf IBM z14 CPU (5.2 GHz) avec refroidissement par eau
  - 2022 Intel Core i9-13900K (9 GHz via *overclocking*)

- Limites : **mur de la chaleur** !

$$P_{dyn} = alpha * \sum C_i * V_{dd} * V_{dd} * F$$

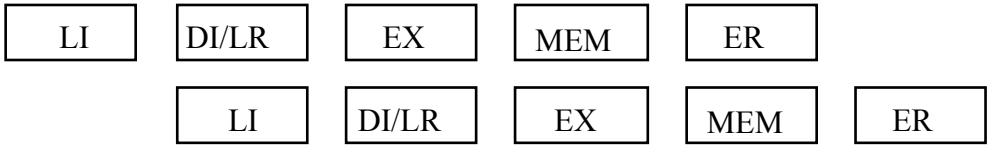
Le mur de la chaleur



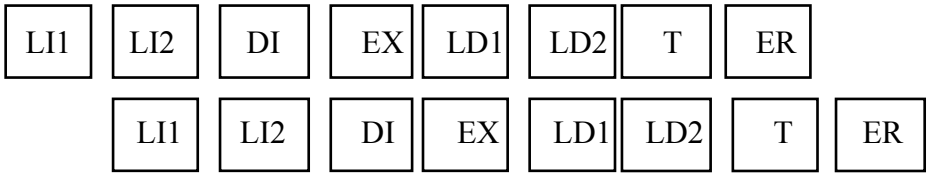
* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

Améliorer les performances

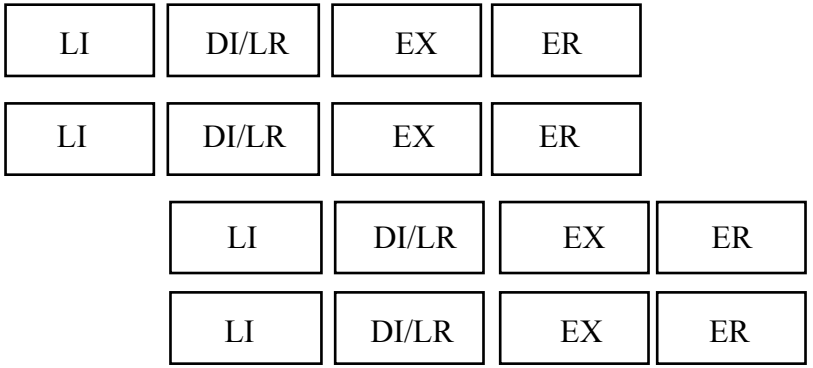
Tex = NI / (**IPC** * **F**)



Pipeline



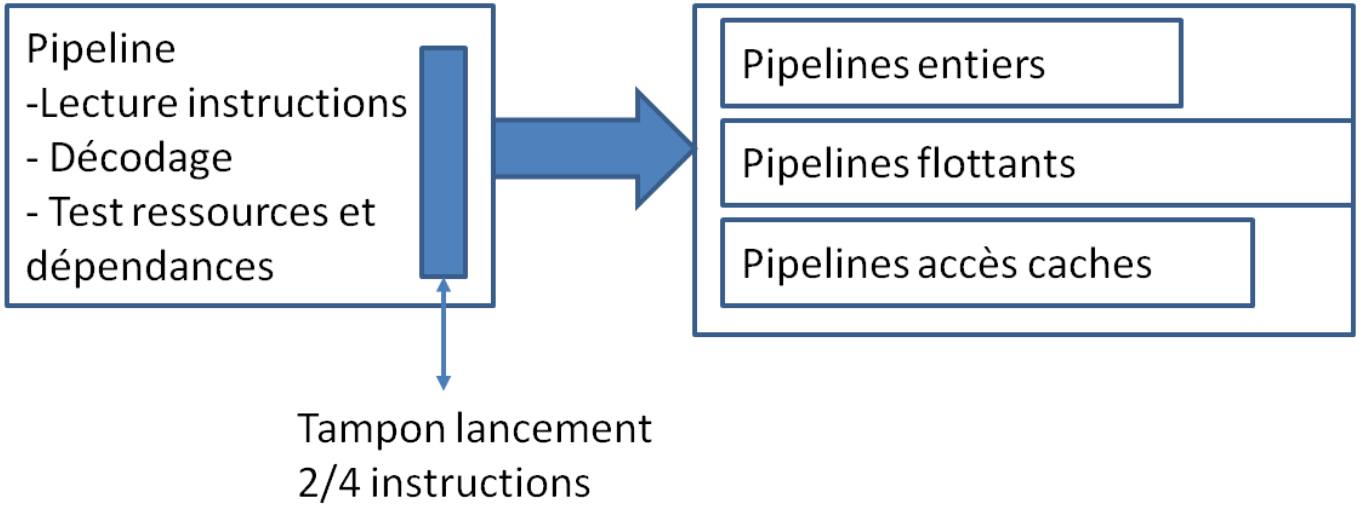
Superpipeline



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

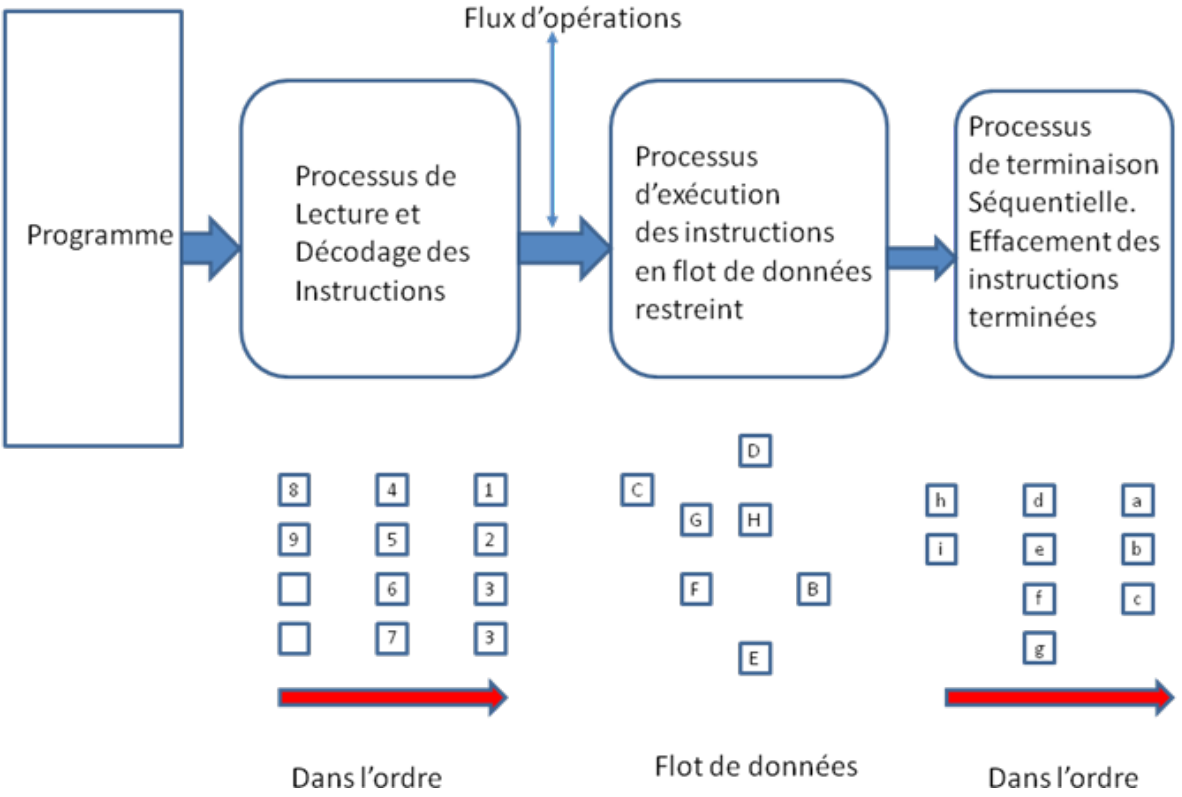
VLIW

Superscalaires dans l'ordre



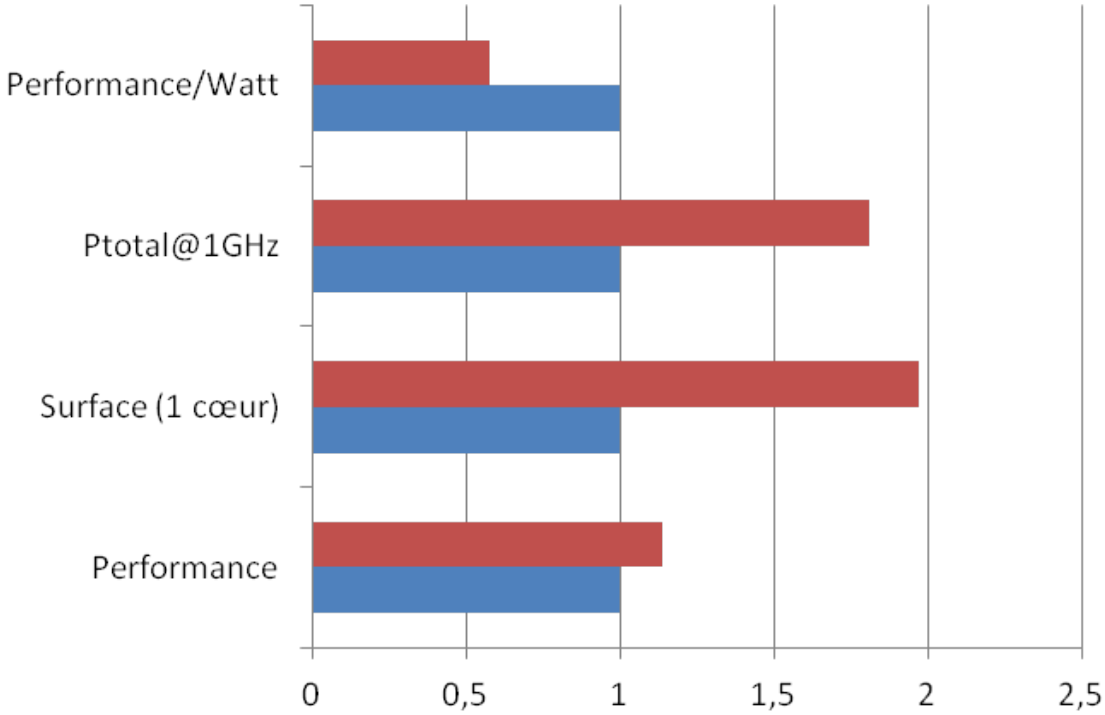
- Alpha 21064 (1992) et 21164 (1995)
- Pentium (1993), Atom – Bonnel (2008)
- ARM Cortex A8 (1995)
- Etc…

Superscalaires « flot de données »



- Pentium Pro (1995) aux iCores
- AMD K6 (1997) aux Zen
- ARM Cortex A9 (2008), A15, …
- IBM PowerPC 601 (1993), Power 4 (2002)…

1 comparaison dans l'ordre – non ordonné



ARM Cortex

- A7 : dans l'ordre
- A9 : flot de données

- Même technologie
- Même période

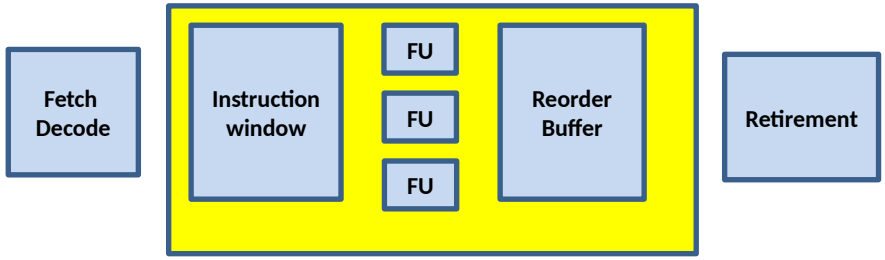
- A9 : plus performant, surface et puissance dissipée supérieure, performance énergétique inférieure
- Tendance générale : Intel, IBM, ARM…

Limits on Instruction Level Parallelism (ILP)

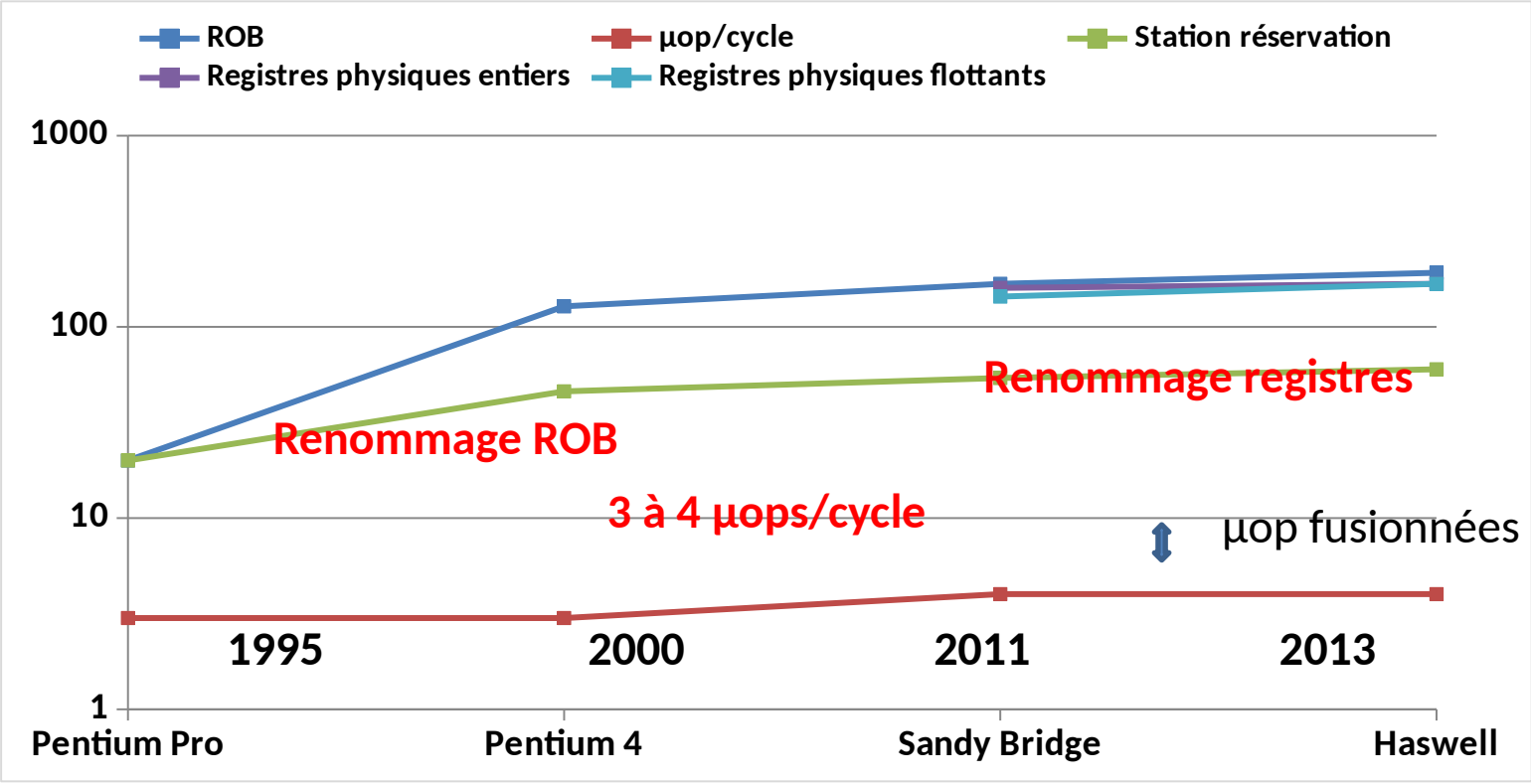
1970: Flynn

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 depuis les années 90 au niveau matériel pour les processeurs « non ordonnés » d'Intel**



Améliorer les performances

Tex = **NI** / (IPC * **F**)

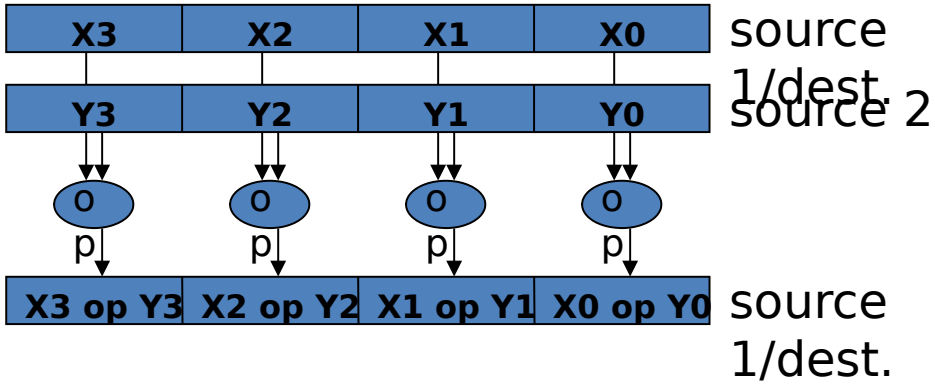
Le parallélisme de données dans les monoprocesseurs

CPU

SIMD

1 instruction avec plusieurs données

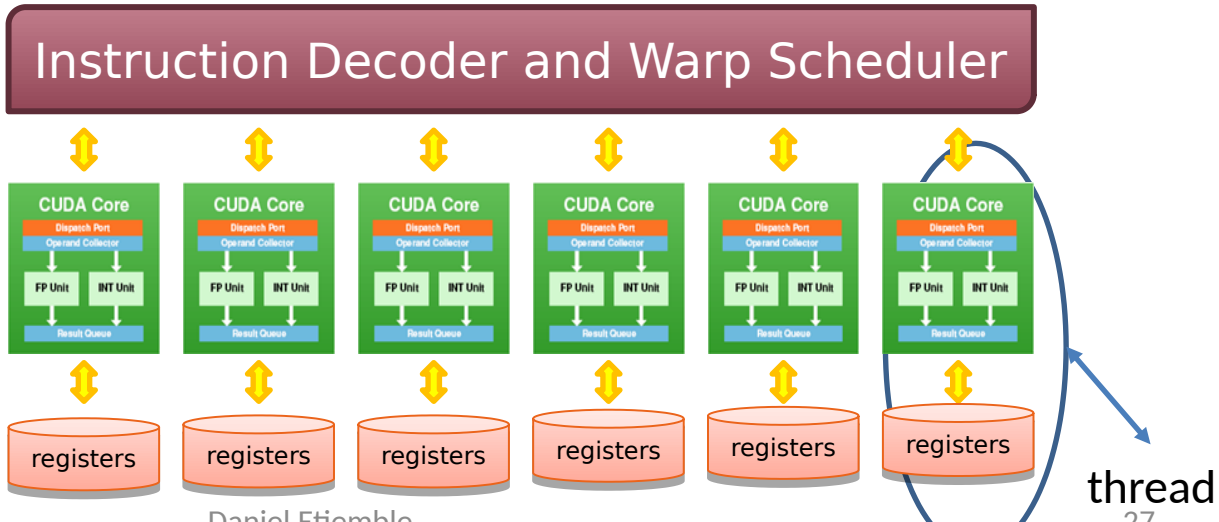
SSE2/3/4 – Neon – Altivec
AVX – AVX2 – AVX 512…



GPU

SIMT

1 instruction pour plusieurs threads



Les instructions SIMD/vectorielles

Tex = **NI** / (IPC * **F**)

- 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)
- ***F limité, IPC limité : NI = seule manière d'augmenter vraiment les performances monoprocesseurs***
- ARM : SVE
- RISC-V : extension vectorielle

SIMD

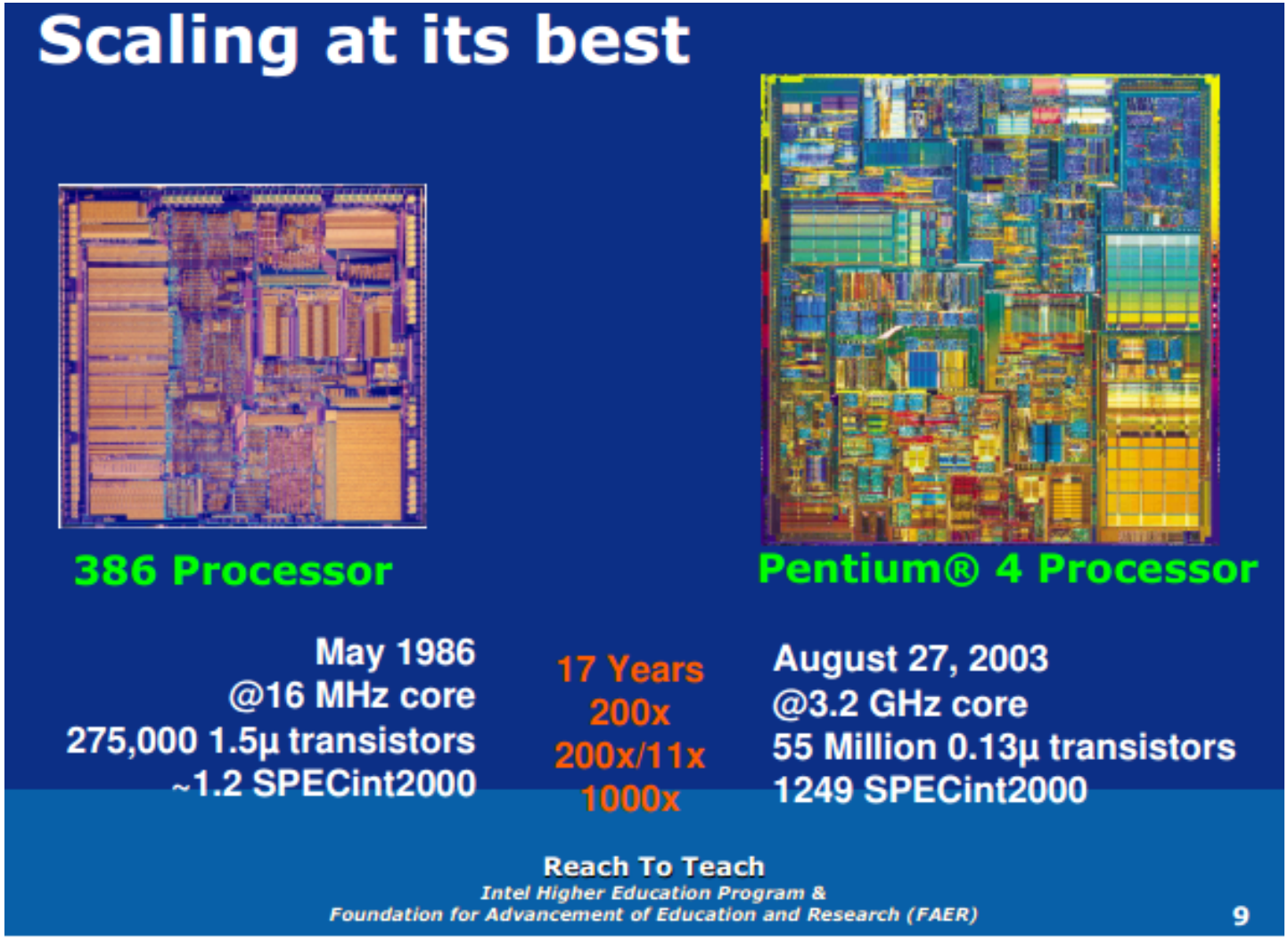
Vectoriel

Free lunch… (par Intel)

F

IPC

NI (SIMD)



Améliorer les performances

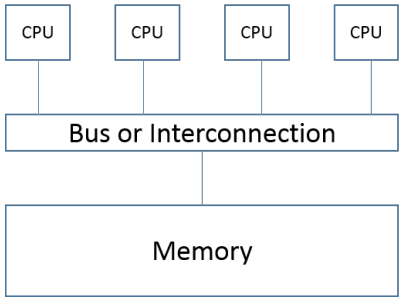
Tex = **NI** / (IPC * **F**)

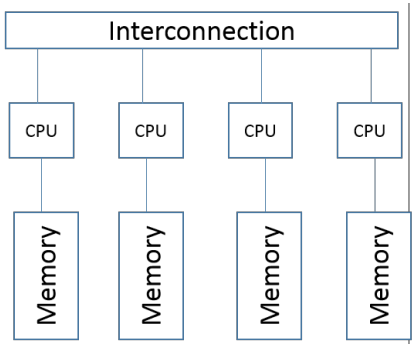
Parallélisme de données ou de threads dans les architectures parallèles

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

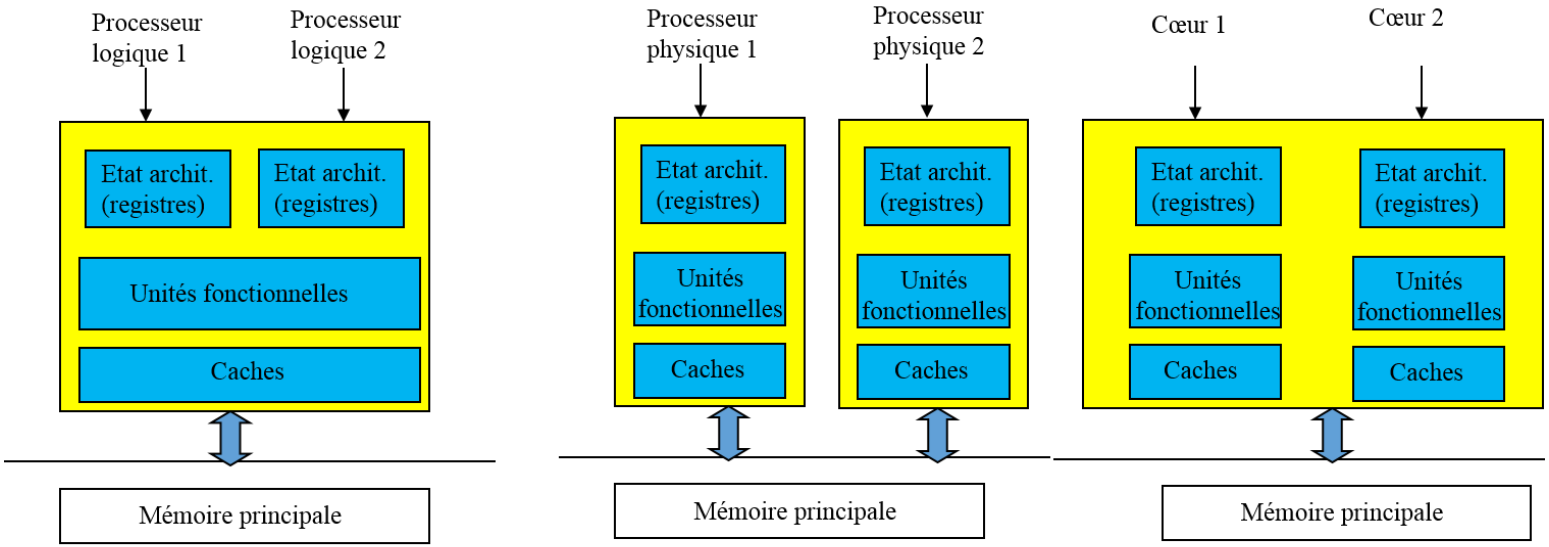
De la programmation séquentielle à la programmation parallèle

- 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

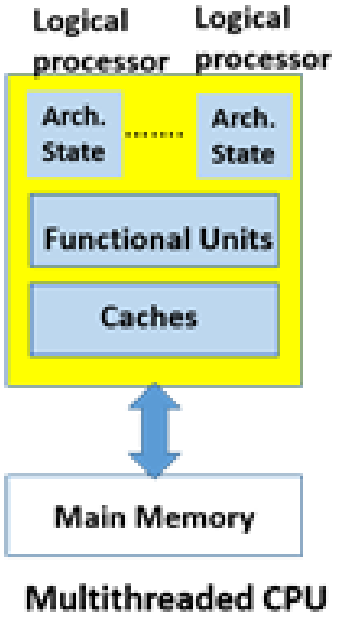
- Hyperthread/Multithread
- Multi-cœurs



CPU multithreads

Tex = **NI** / (**IPC** * **F**)

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



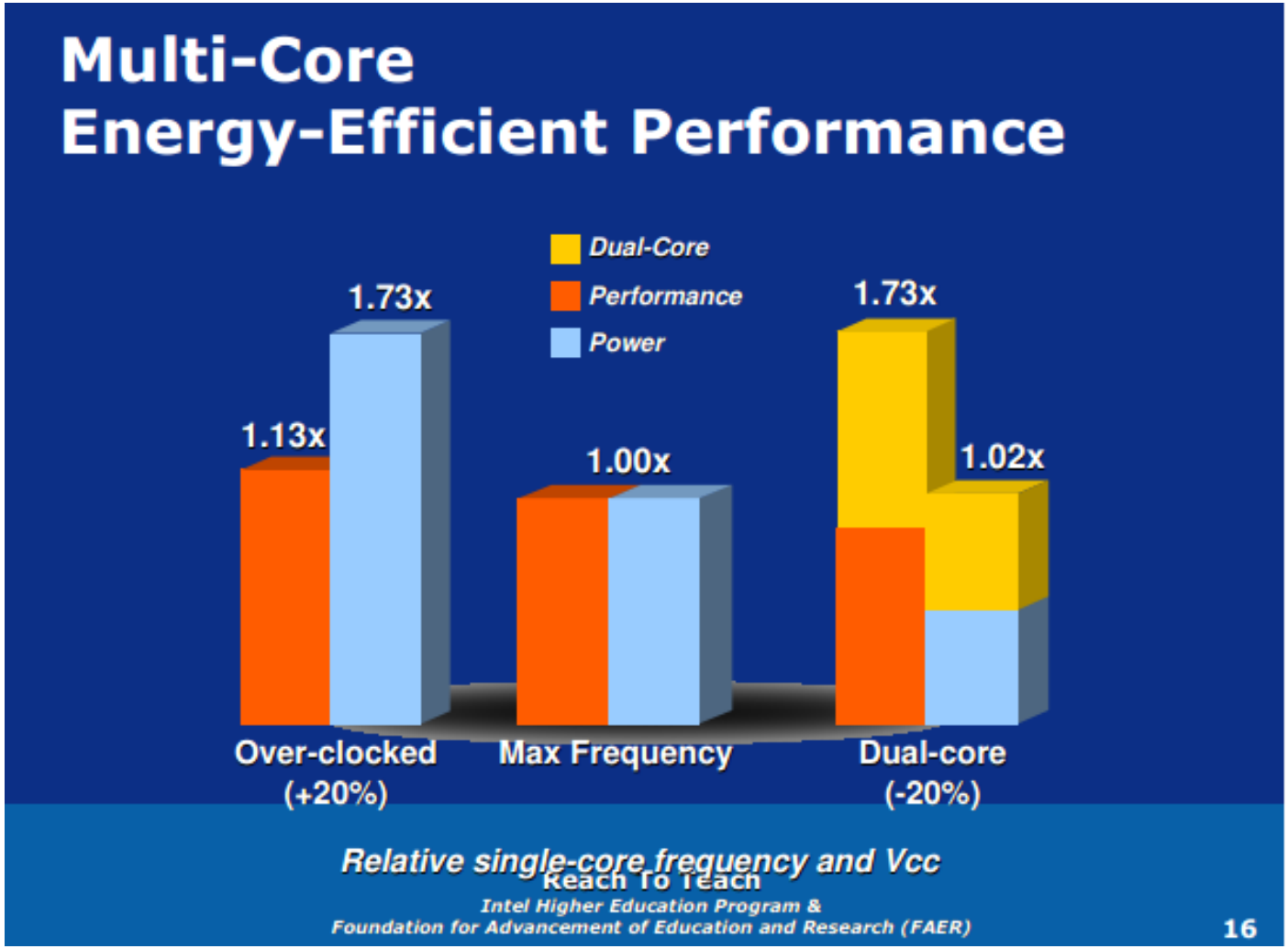
Multithreading grain fin
- Commuter d'un thread à un autre à chaque cycle d'horloge
- Sun Niagara, serveurs Oracle

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

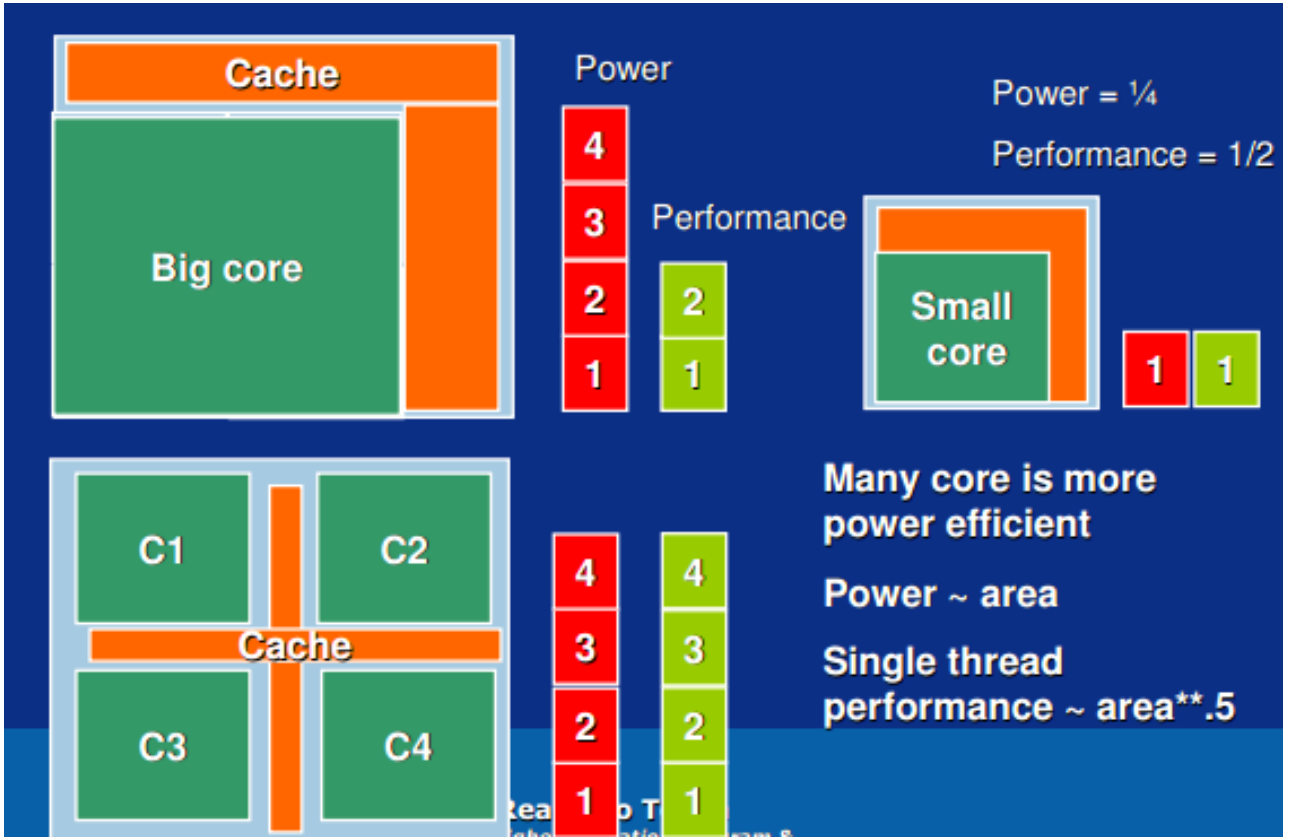
Multithreading réduit CPI_{Mem}

Multi-coeurs avec des coeurs multithreads

Pourquoi les multi-cœurs ?



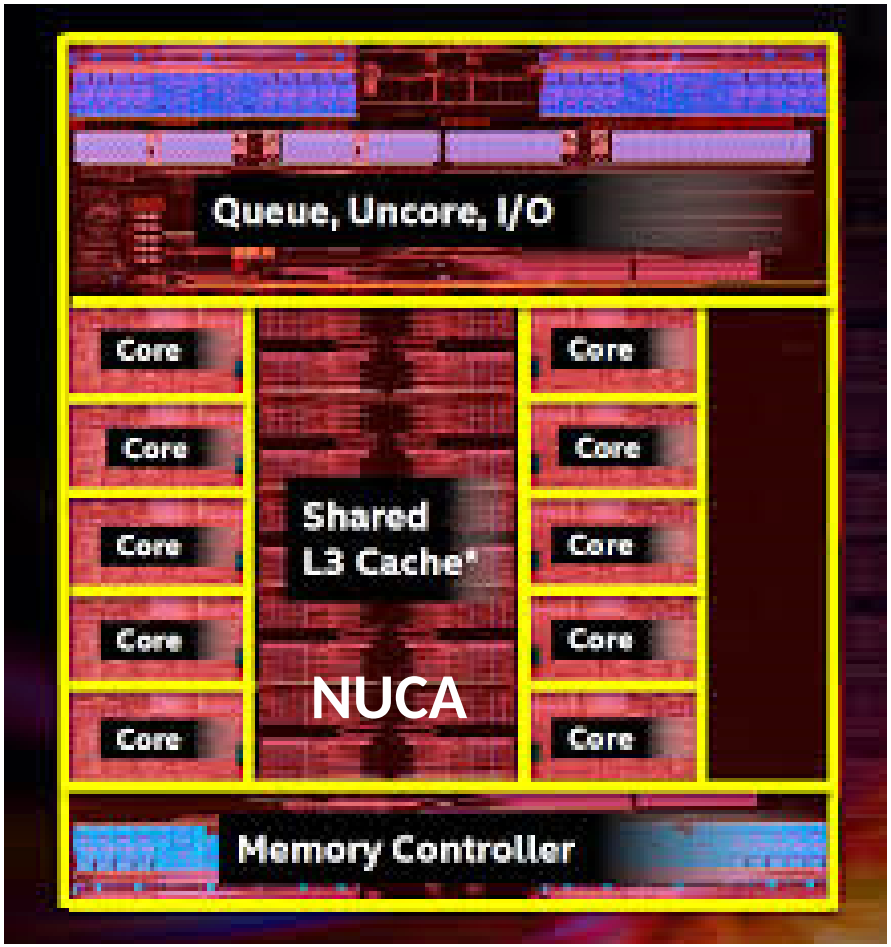
Efficacité énergétique des multi-cœurs



Un multi-cœur Intel de 2016

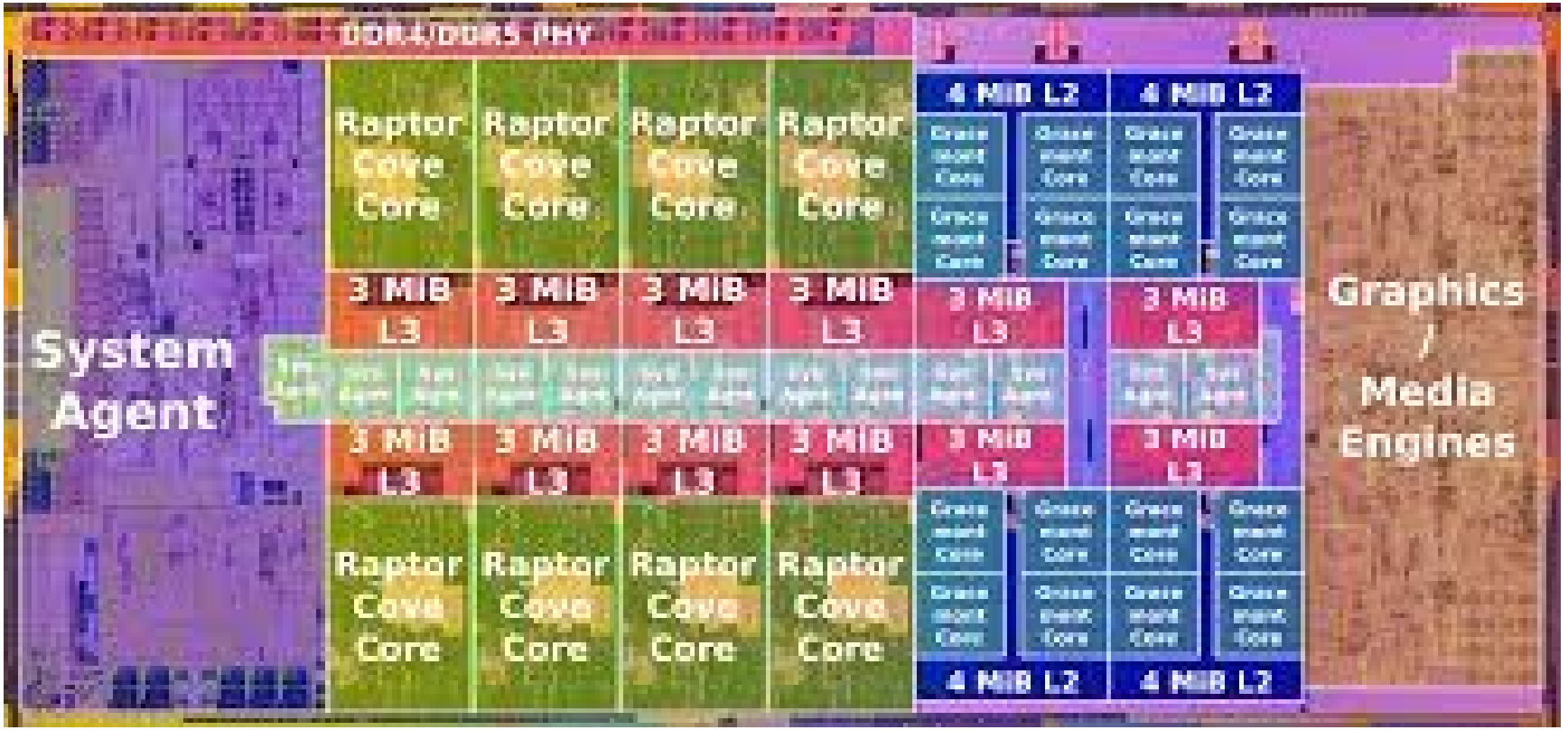
Broadwell-E

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

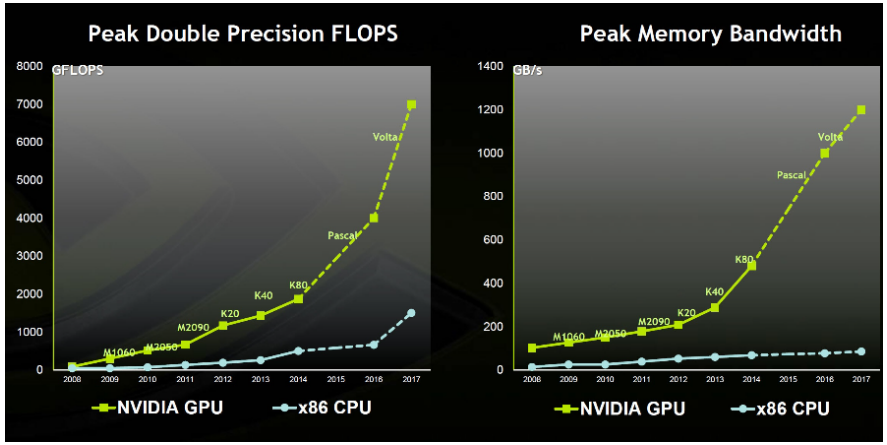


Un multi-cœur Intel de 2022

Intel Raptor Lake 16 coeurs E, 8 coeurs P

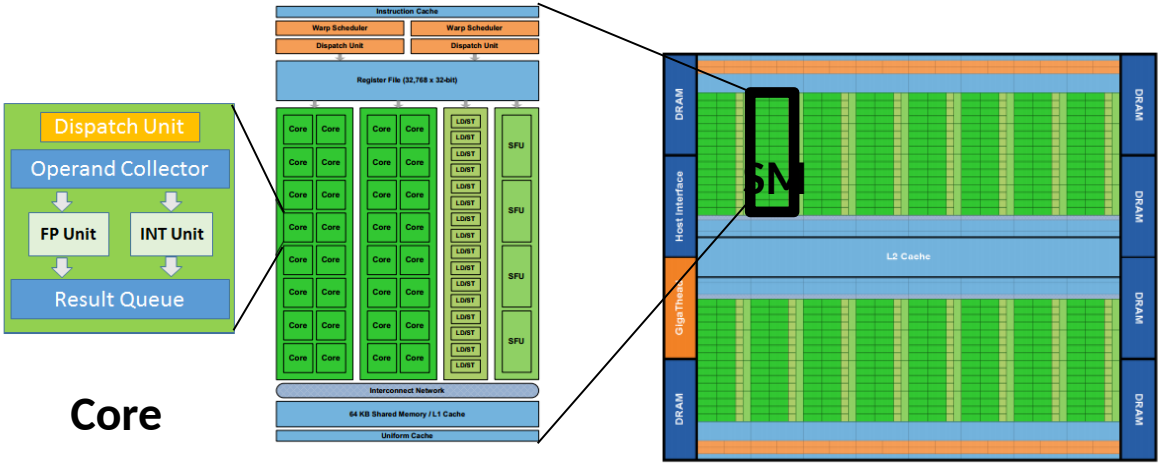


Et les GPU ?



Parallélisme de données massif

Très grand nombre de cœurs



Core

Streaming Multiprocessor

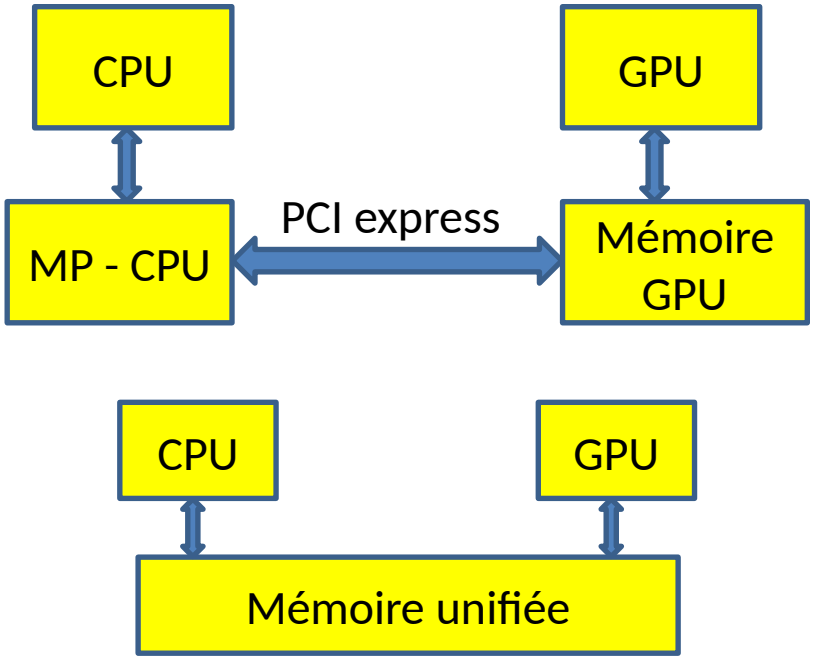
Fermi GPU

Le GPU Nvidia Volta

21 milliards de transistors.

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

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

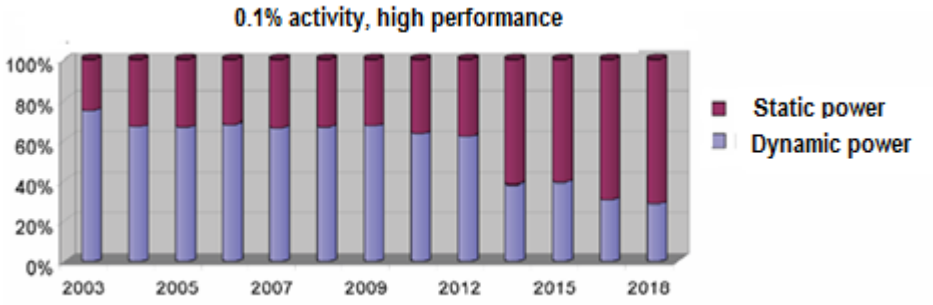
= 0 (autrefois !)

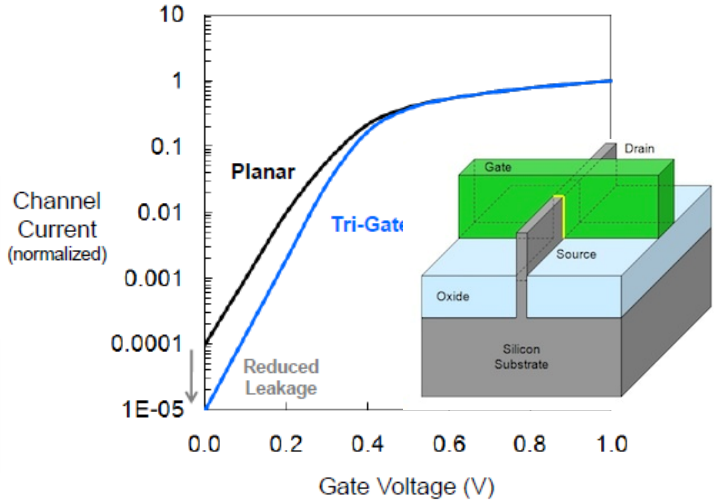
≠ 0

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

Réduire la puissance statique

$$P_{dstatique} = V_{dd} * I_{fuite}$$





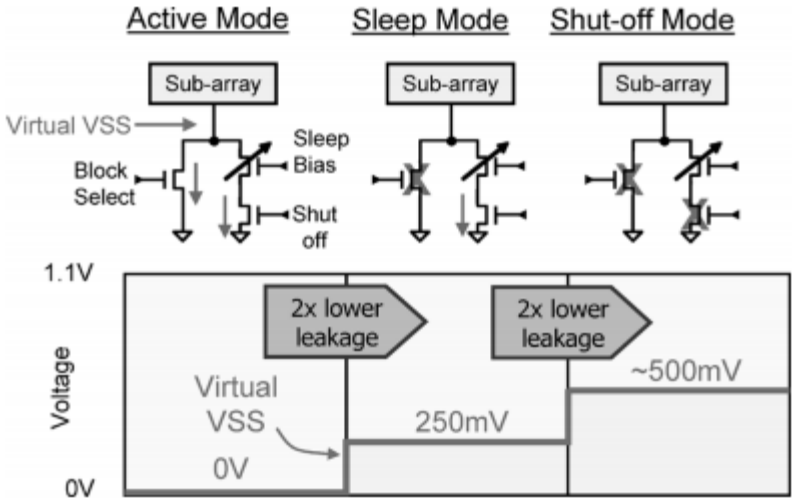
TECHNOLOGIE

- Ex: Intel Tri-gate

CIRCUITERIE

- Ex: Masses virtuelles

Cache L3 du CPU Xeon 65-nm



Réduire la puissance dynamique

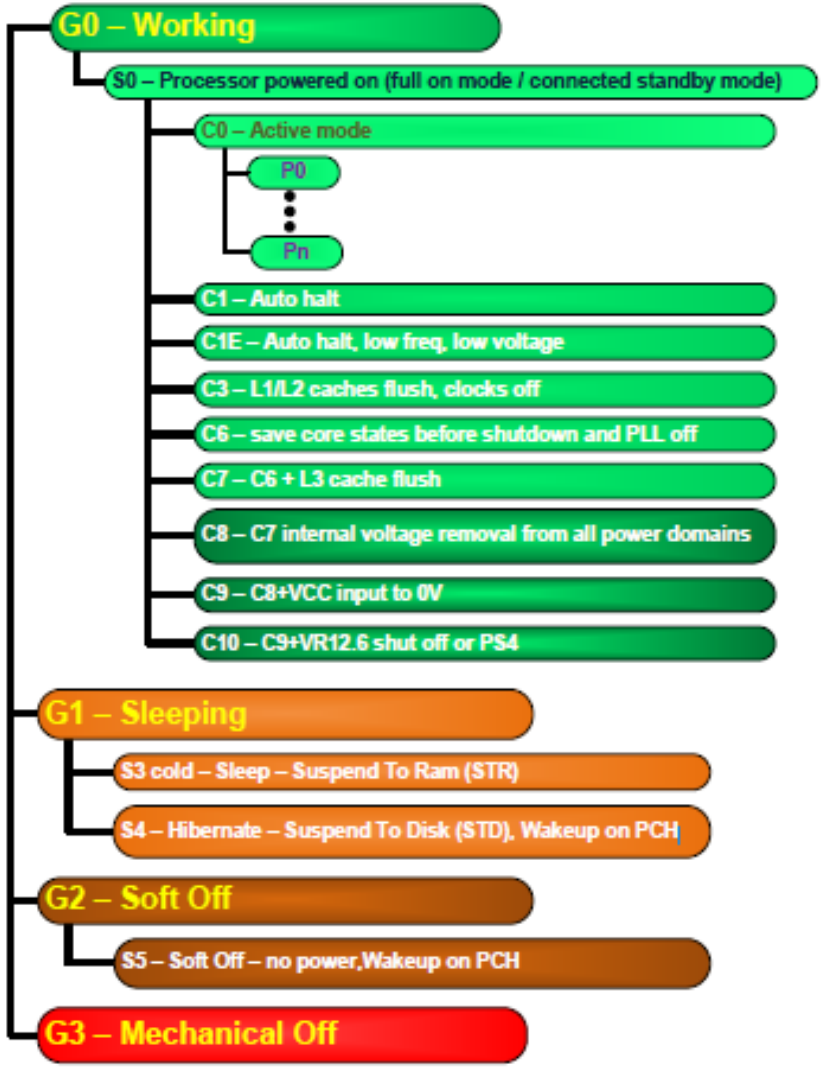
- Circuiterie
  - « *clock gating* » : n'activer que les parties utiles
  - Plusieurs tensions d'alimentation
  - Transistors rapides, moyens et lents (tension de seuil Vt)
- **Architecture**
  - **Fréquence d'horloge (F)**
  - **Découpage en domaines (V_{dd}, F)**

Réduire la puissance dynamique

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

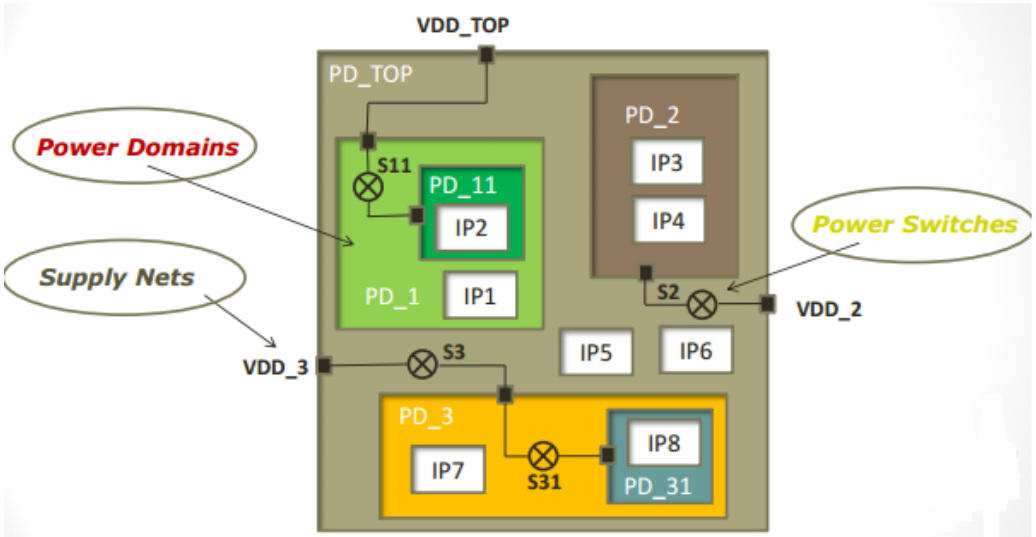
α

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

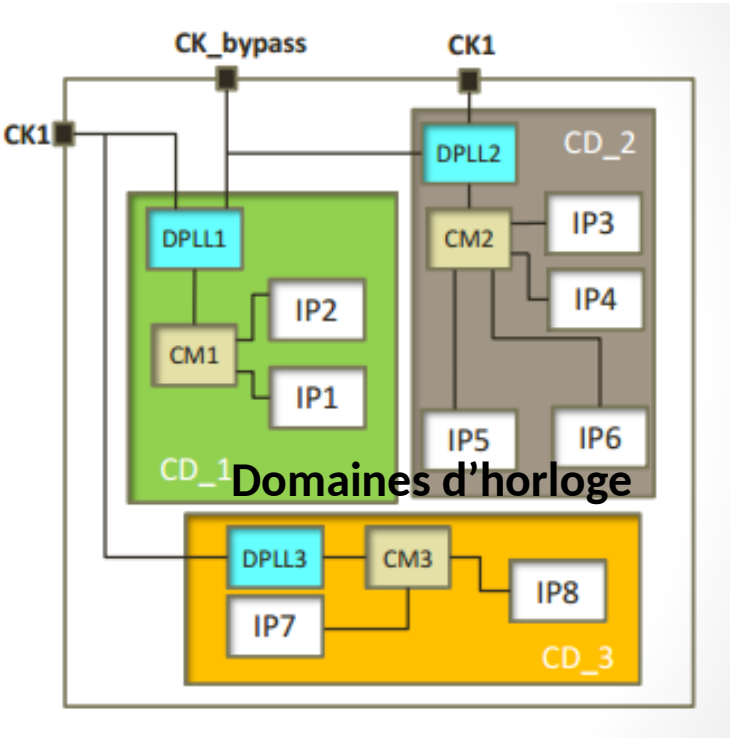


* Note: Power states availability may vary between the different SKUs

Réduire la puissance dynamique



Domaines de puissance



Domaines d'horloge

V_{dd}

F

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

N’activer que les parties utiles

Plusieurs modes de fonctionnement α

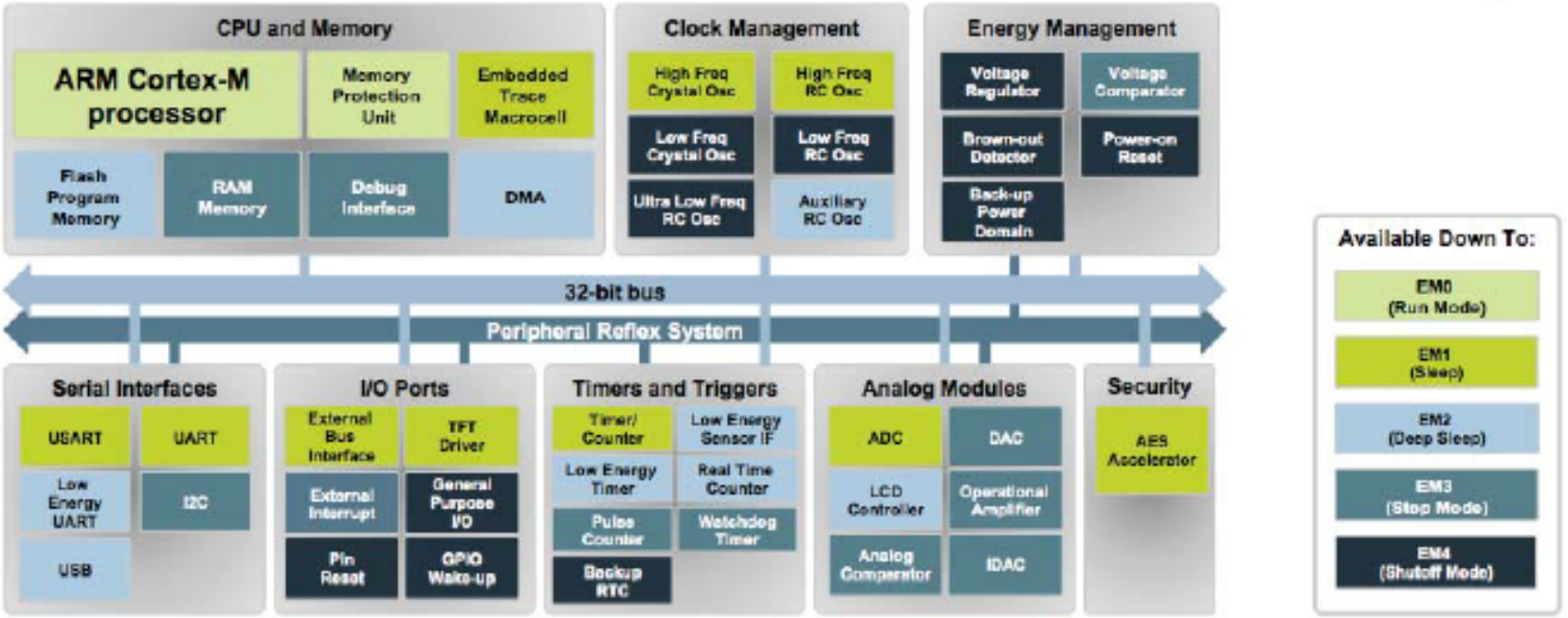
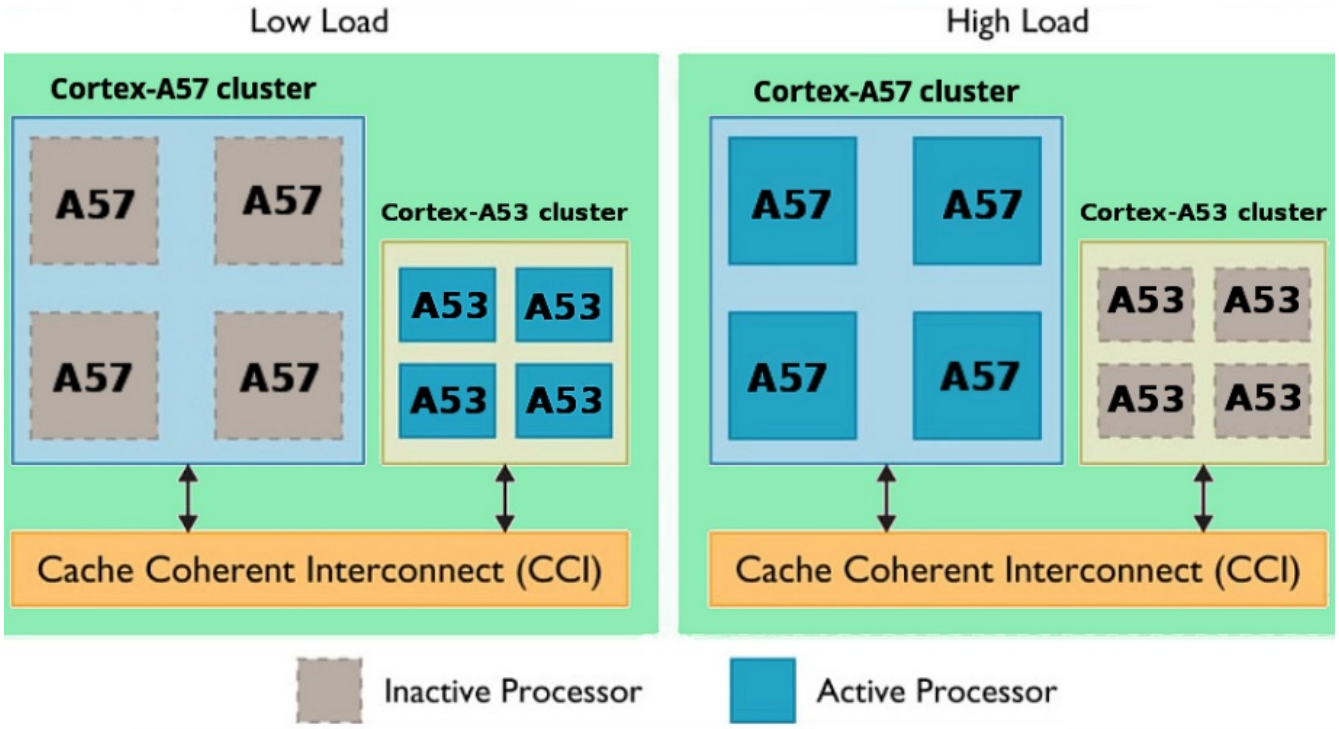


Figure 2. EFM32 Gecko block diagram and the five energy modes.

Adapter le fonctionnement à la charge

α



Circuit Exynox 7420 (Galaxy S6)

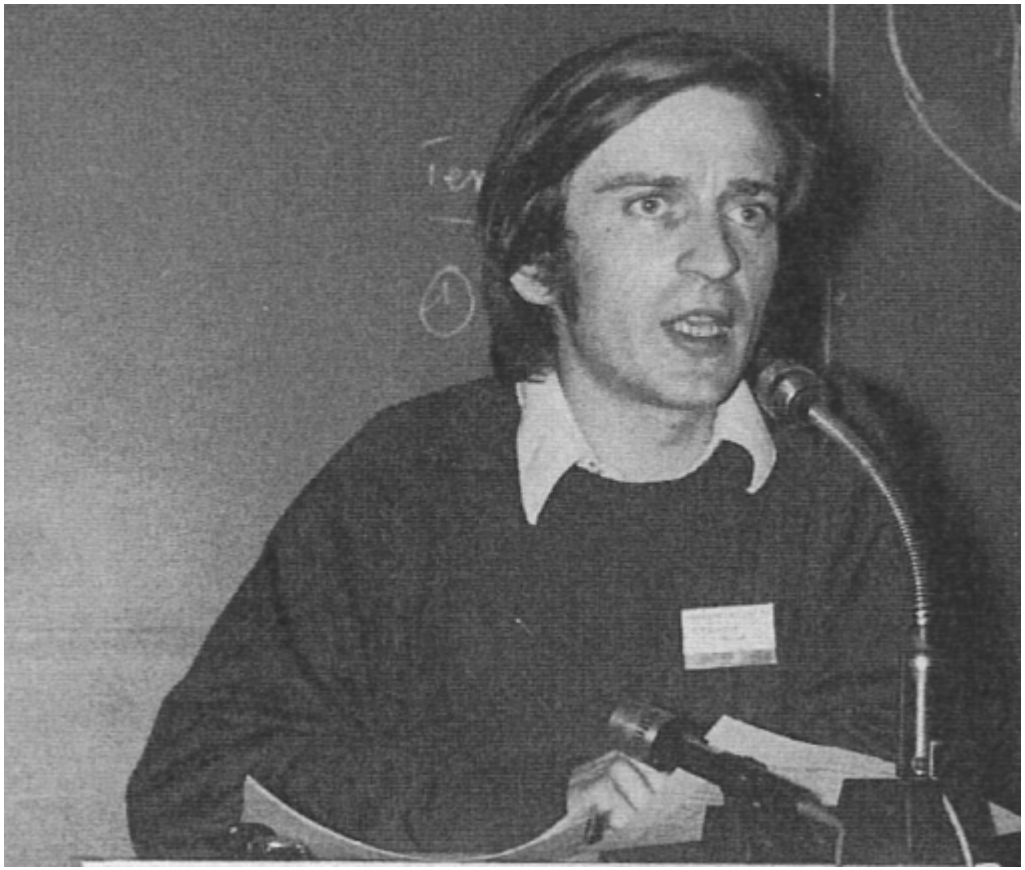
Et le futur?

- La loi de Moore : les limites fondamentales sont proches
  - Mais les nœuds 7 nm, 5 nm sont là. Le 3 nm est déjà annoncé !
- 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…

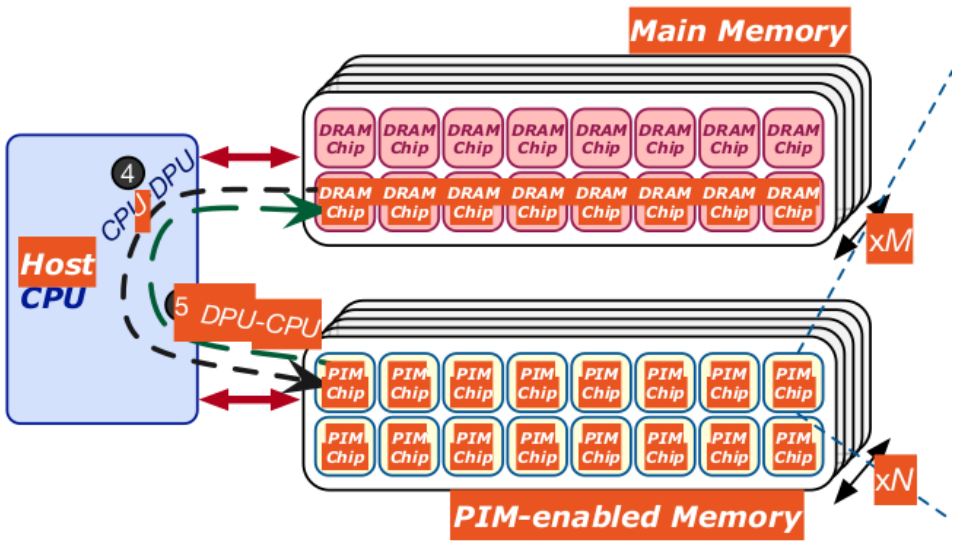
Remarques pour conclure

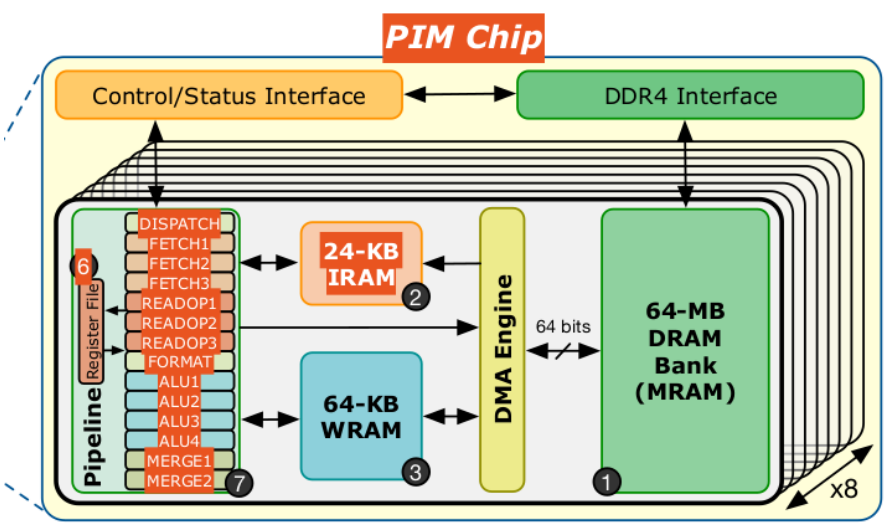
- La tendance principale (pas les détails) de l'évolution des CPU peut être expliquée par
  - La loi de Moore
  - Temps d'exécution = f(NI, CPI, Tc)
  - Puissance dissipée CMOS
- Valable pour les processeurs programmables aussi longtemps que la technologie CMOS sera utilisée
- Les architectures mixtes 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 ?



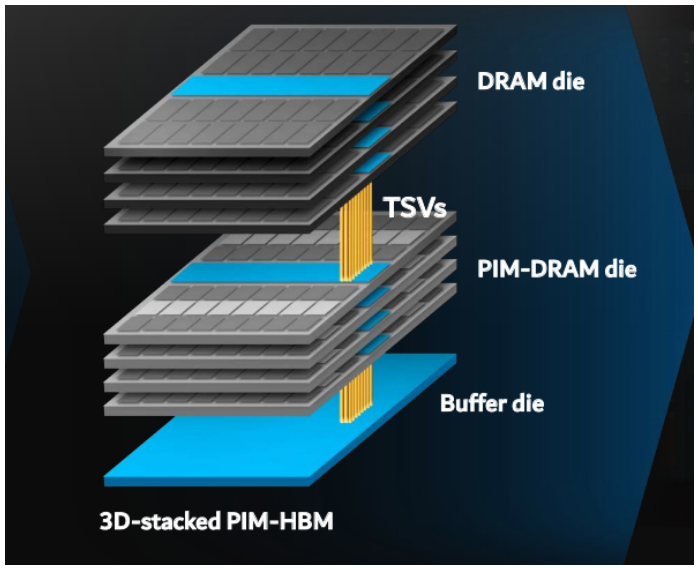
PIM - UPMEM



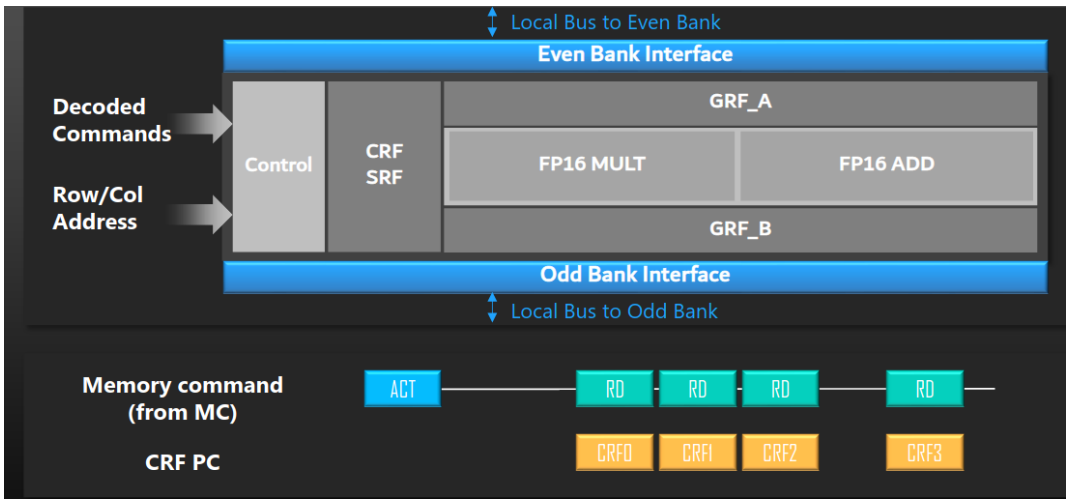


CPU en technologie DRAM

PIM – Aquabolt XL (Samsung)



DRAM HBM avec puce PIM
- Intégration 3D
- Technologie TSV



Puce PIM