Plan

- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

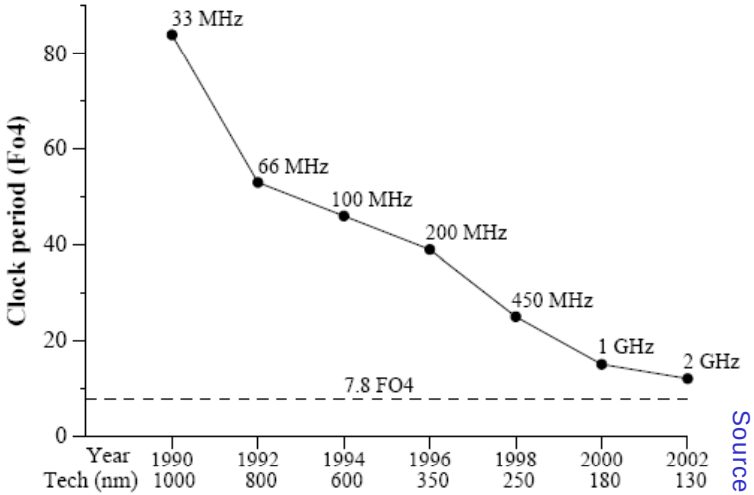
Plan

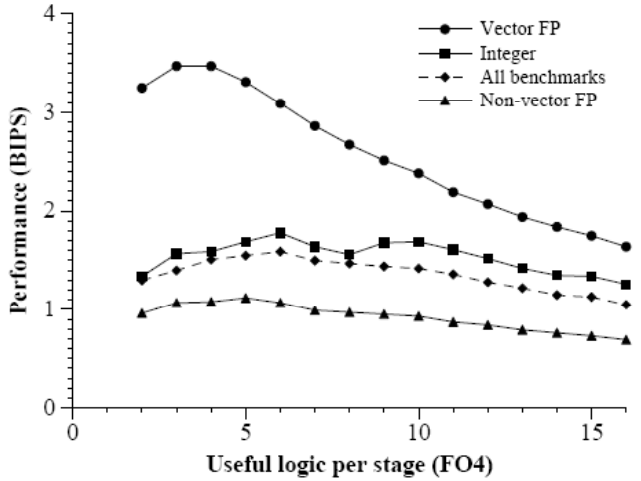
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Allongement du pipeline

- $\Phi_{\text{clock period}} = \Phi_{\text{logic}} + \Phi_{\text{latch}} + \Phi_{\text{skew}} + \Phi_{\text{jitter}}$

  - Clock en # FO4 (*fan-out four*: un inverseur avec 4 copies de sa sortie)
  - Deux approches pour gains en performance (12 ans)
    - délai FO4 ↓
      - 8x: 1000nm → 130nm
    - # FO4 par étage ↓
      - 7x: 84 FO4 → 12 FO4
  - Φ_{logic} décroit en FO4
  - $\Phi_{\text{latch}} + \Phi_{\text{skew}} + \Phi_{\text{jitter}}$ constant en FO4
  - Seuil quand *overhead* ~ *logic*
  - Max: pipeline de 6 à 8 FO4



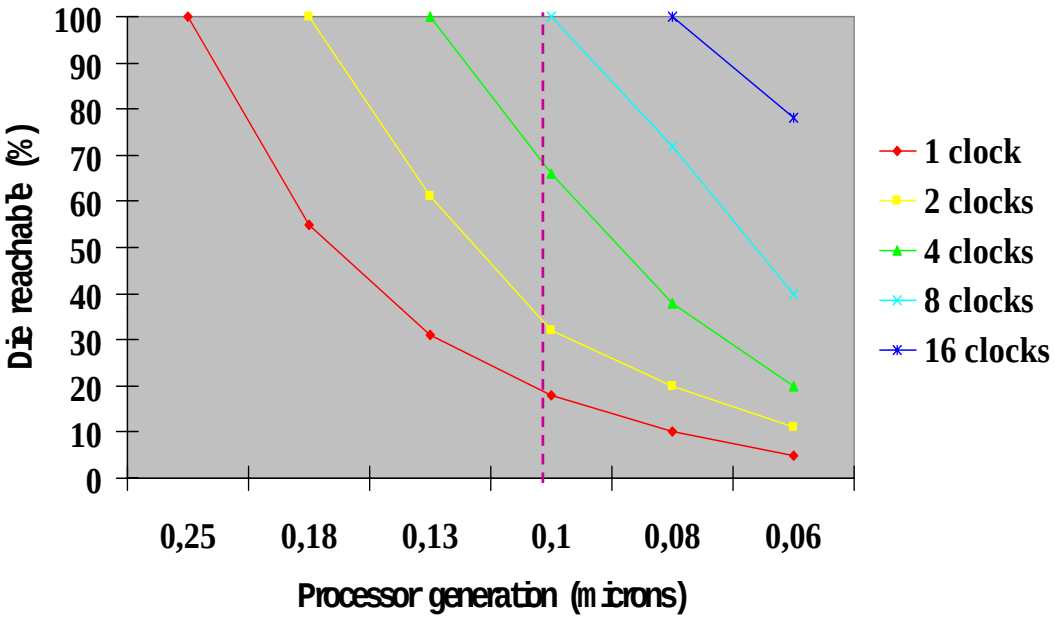


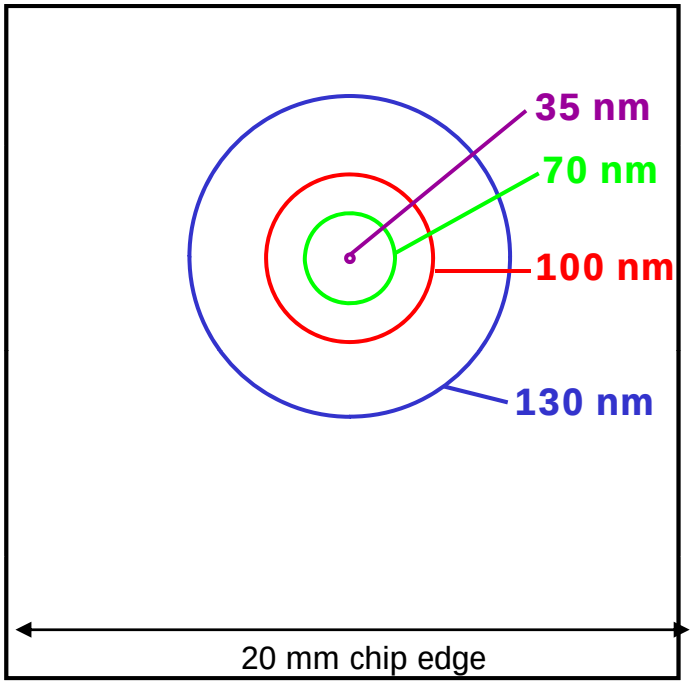
Source: ISCA 2002

Plan

- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Accroissement du degré superscalaire
Délai de propagation



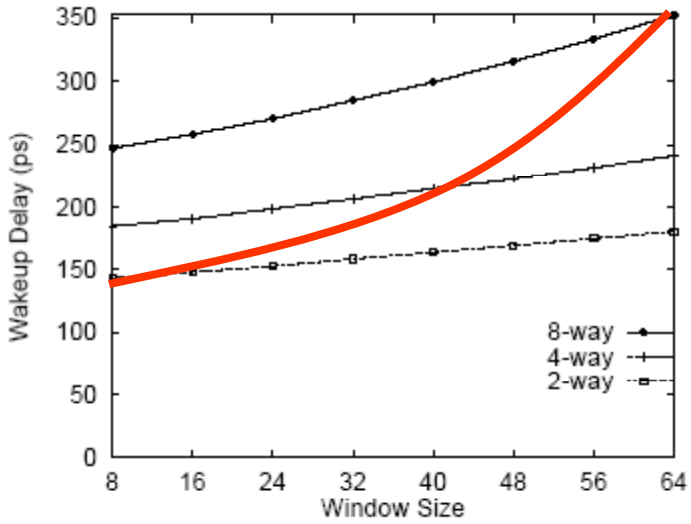


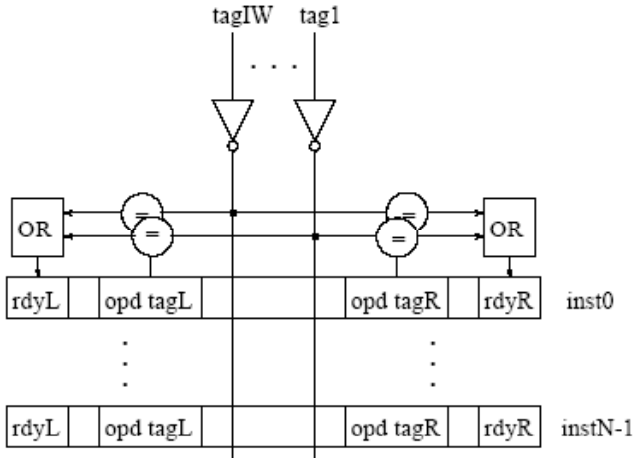
Source: keynote MSP Kathrun McKinley

- Plusieurs cycles pour traverser la puce
- Partitionnement nécessaire

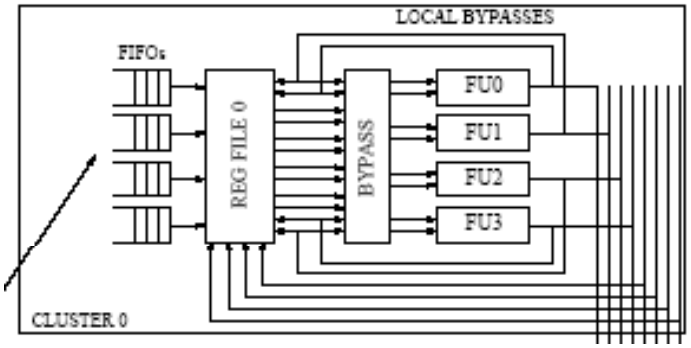
Accroissement du degré superscalaire
Complexité

- Accroissement des délais et de la complexité
- Exemple de la fenêtre d'instructions
  - délai *wakeup* ↑ avec taille
  - restructuration & *clustering* pour réduire délai





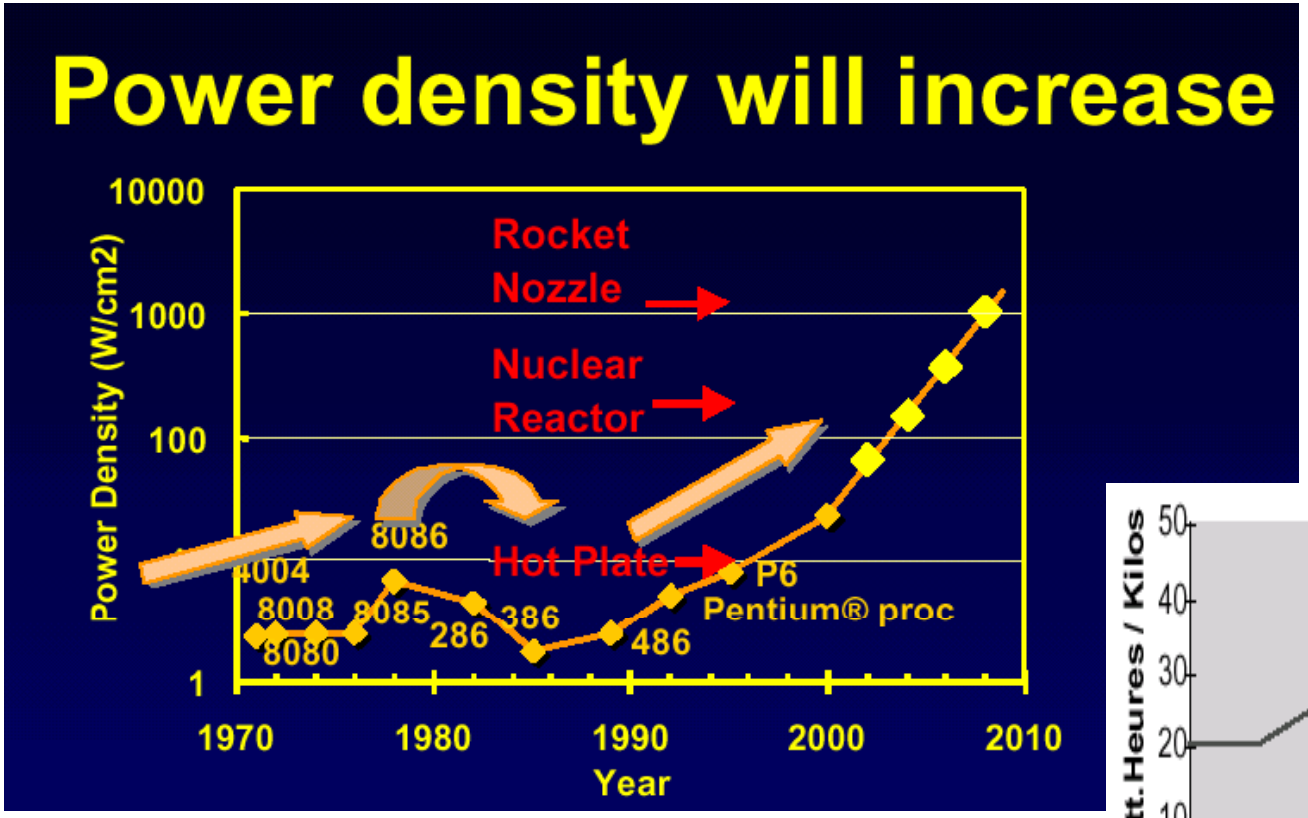
Fenêtre d'instructions



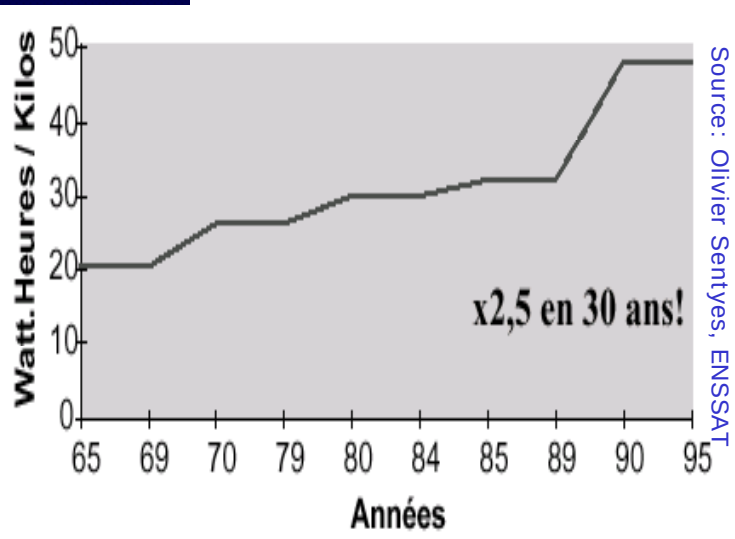
Fenêtre d'instructions dépendantes

« Complexity-Effective Superscalar Processors », by S. Palacharla, N. P. Jouppi et J. E. Smith, In the 24th Annual International Symposium on Computer Architecture, 1997

Accroissement du degré superscalaire
Consommation



Source: Intel



Source: Olivier Sentyes, ENSSAT

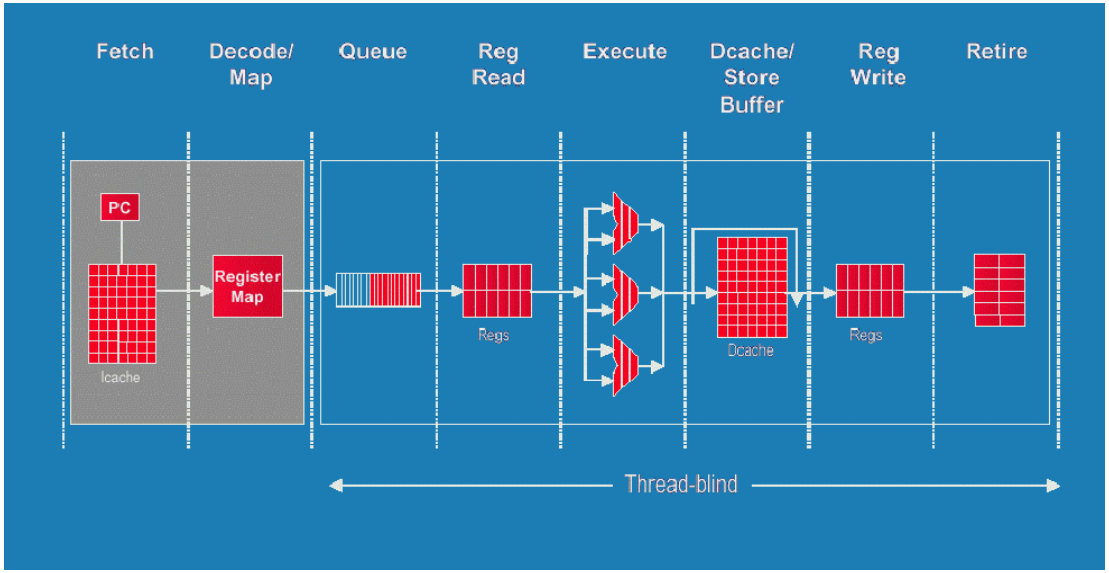
- Consommation/Dissipation: premier facteur limitatif
- Consommation statique & dynamique

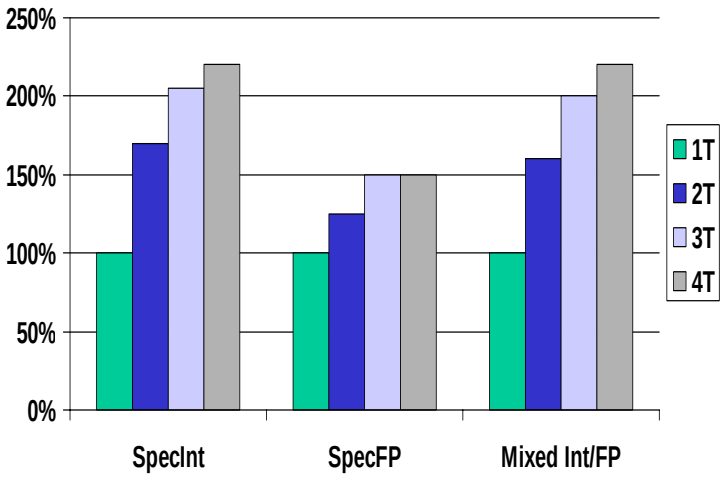
Plan

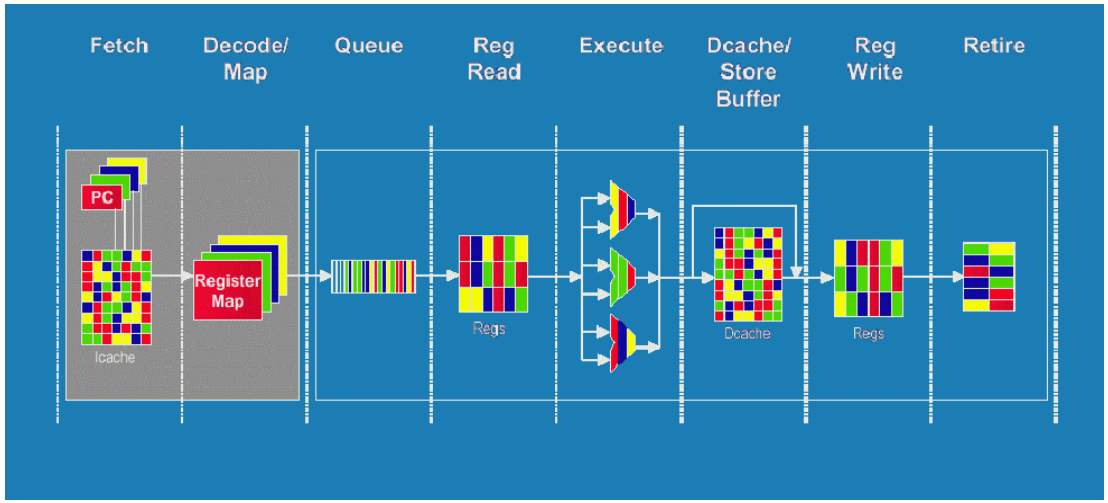
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Processeurs multithreads
Exemple : Alpha 21464 (EV8)

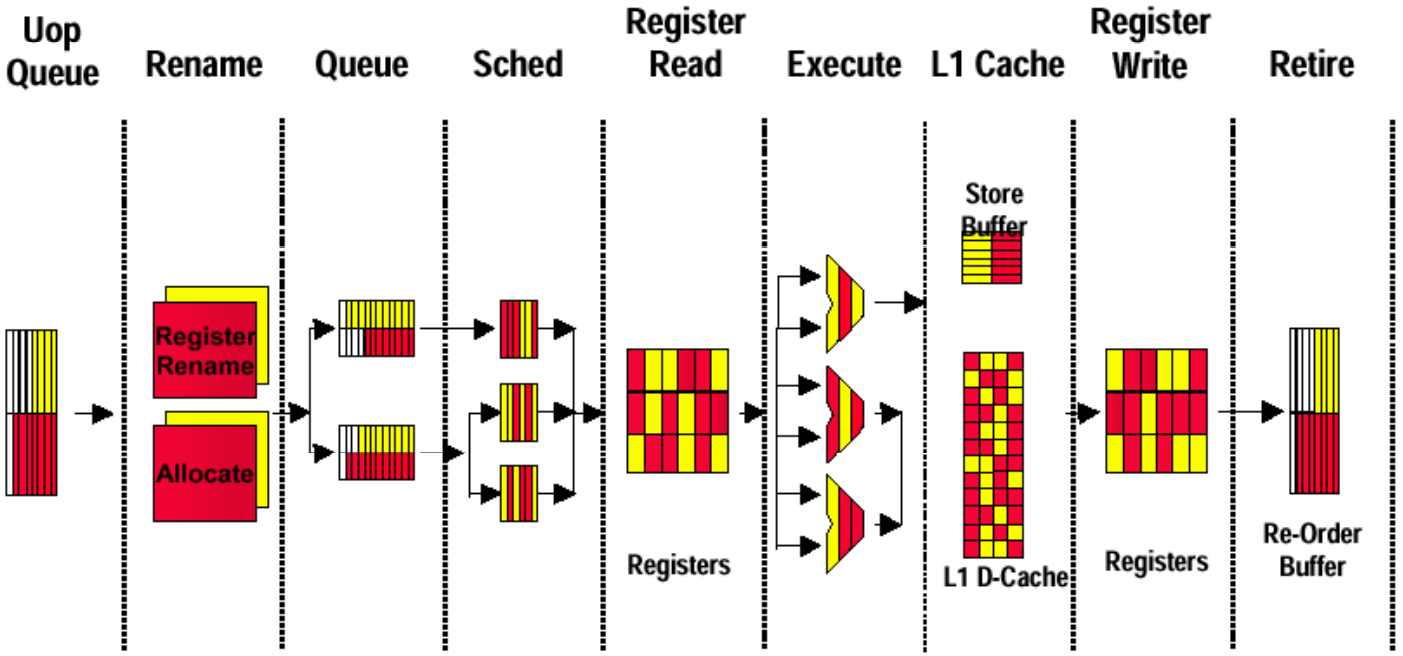
- Simultaneous Multi-Threading (SMT)
- Plusieurs processus s'exécutent en même temps
- Partage des ressources
- Meilleure utilisation/rentabilité des ressources
- Mais mêmes limitations que superscalaires en scalabilité







Processeurs multithreads
Exemple : Intel Xeon Hyper-Threading



Source: *Intel Technology Journal*, *Volume 6, Number 1*, February 2002.

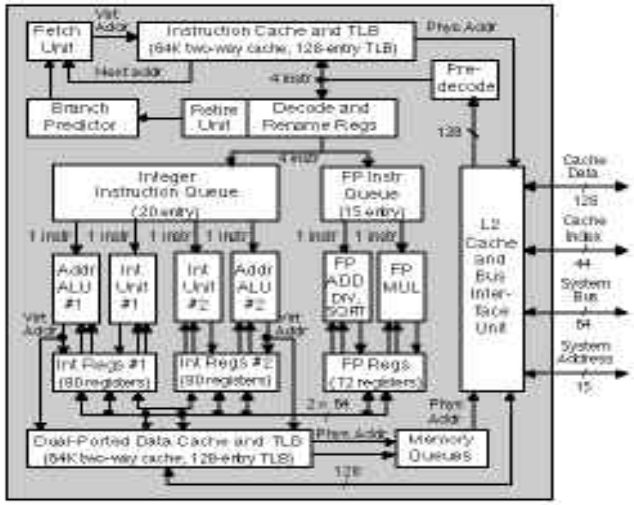
- 2 flots (contre 4 flots pour Alpha 21464)

Plan

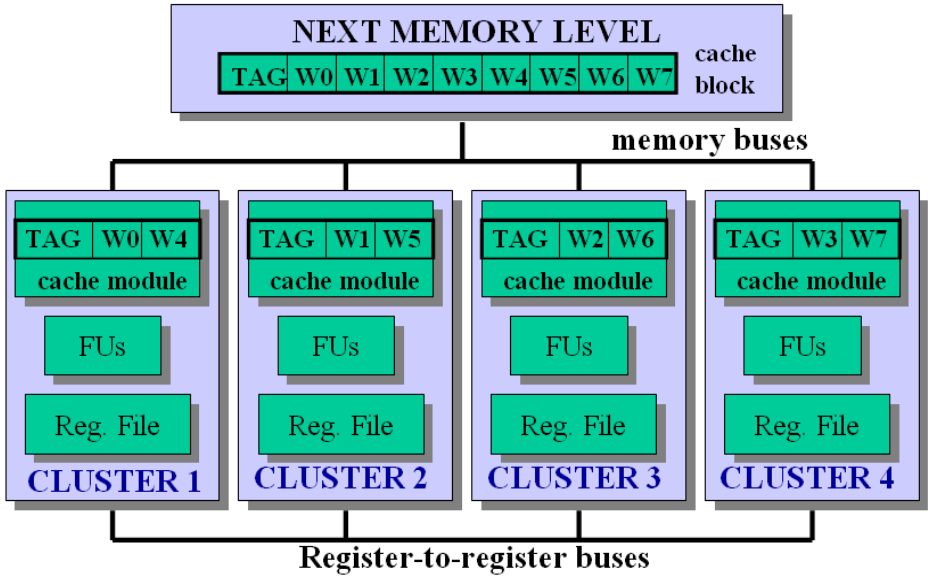
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Architectures en clusters

- Alpha 21264
  - Six unités de calcul regroupées en 3 *clusters*
  - Banc de registres dupliqué; plus de ports; 1 cycle supplémentaire pour synchronisation
- VLIW en clusters
  - séparation FUs, registres, caches



Alpha 21264



E. Gibert, J. Sánchez and A. González, ICS 2002

Plan

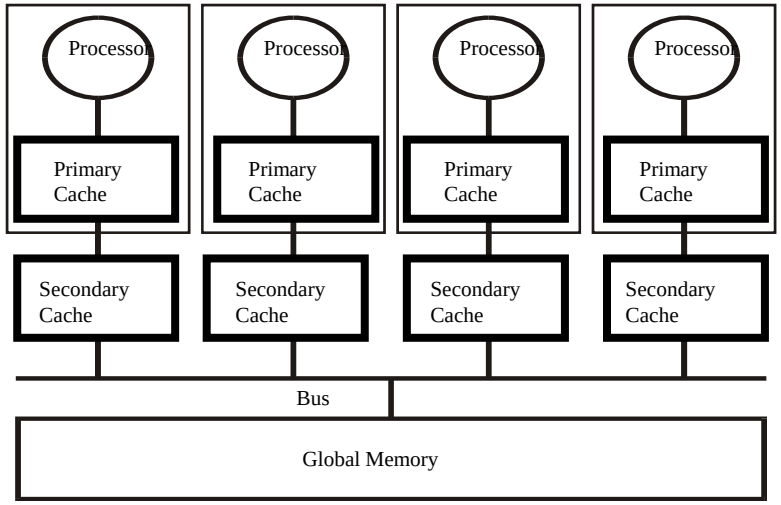
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

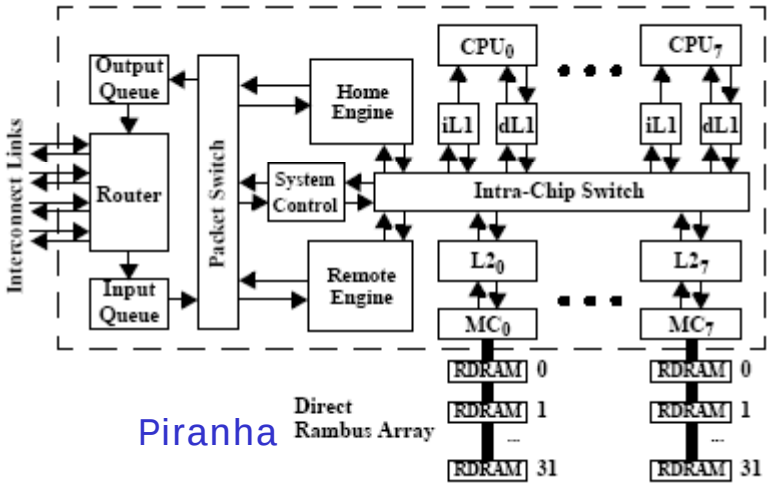
Plan

- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

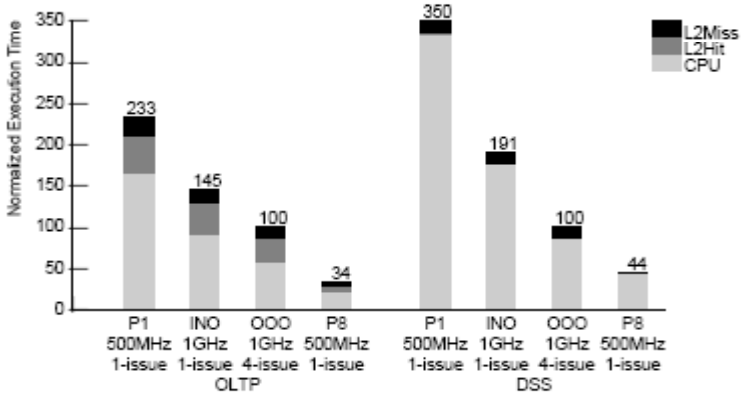
Chip Multi-Processors (CMP)
Principes & modèles existants

- Atouts
  - Simplicité de conception
  - Scalabilité
  - Plus faible consommation
  - Fréquence élevée
- Faiblesses
  - Programmation parallèle de codes « irréguliers »
  - Bande passante mémoire en modèle partagé





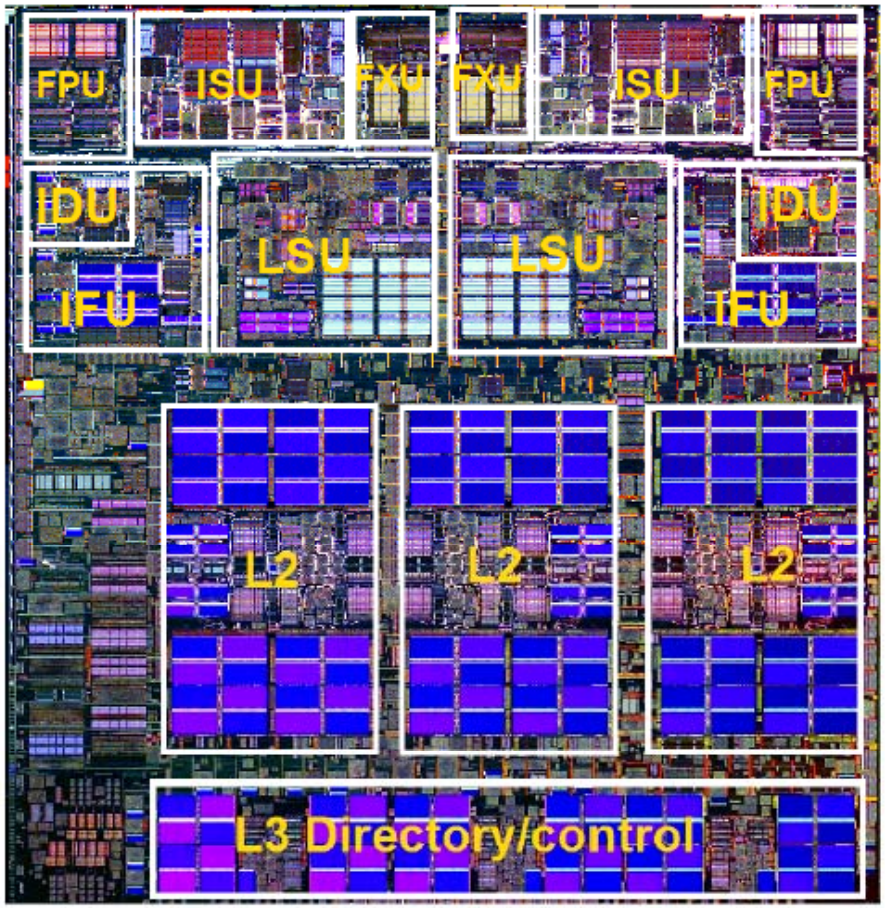
Piranha



- Compaq
  - Piranha
  - 8 Alphas sur une puce

Chip Multi-Processors (CMP)
Principes & modèles existants

- IBM Power 5
  - 2 processeurs
  - SMT (2 threads)
  - 24% surface additionnelle pour SMT
- Autres multicores
  - Intel (P4+HT)
  - AMD (Opteron)
  - nombreux processeurs embarqués (ARM, Philips)

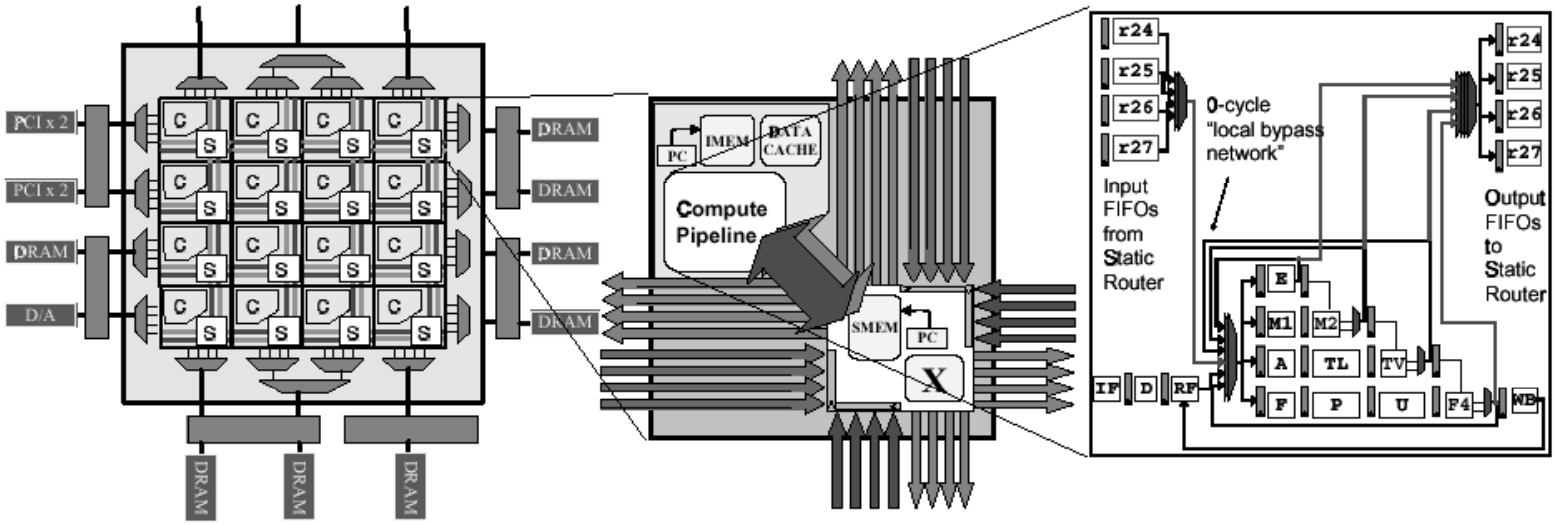


IBM Power5

Plan

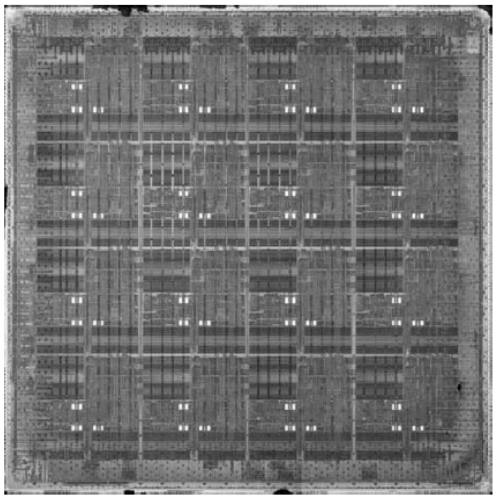
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

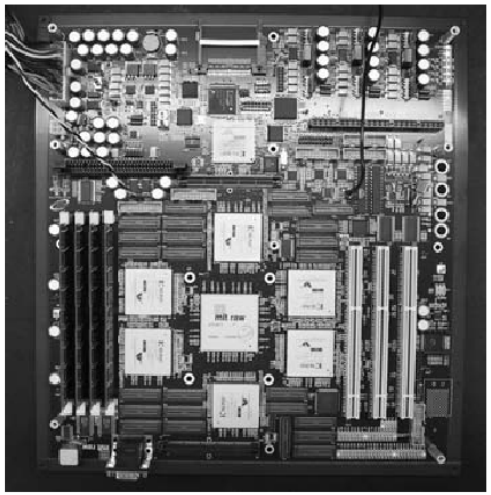
RAW

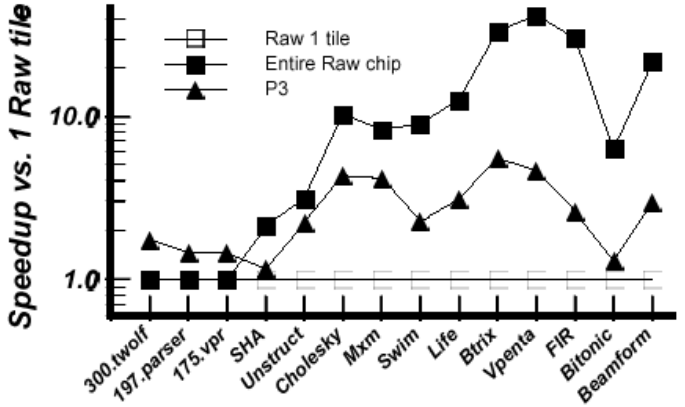


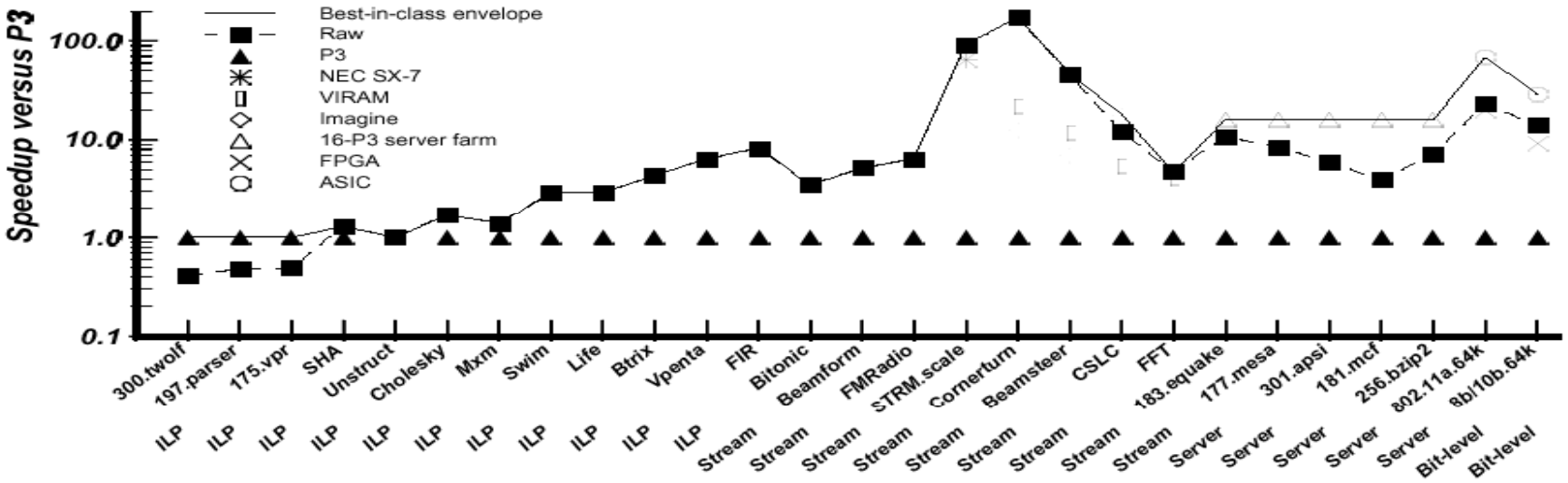
- Buts
  - spécialisation variable d'une architecture (ASIC↔processeur)
  - exploiter ressources parallèles
  - éviter les longs délais de propagation
  - mieux exploiter les *pins* pour la bande passante
- Principes
  - multicore + réseau d'interconnexion programmable

RAW





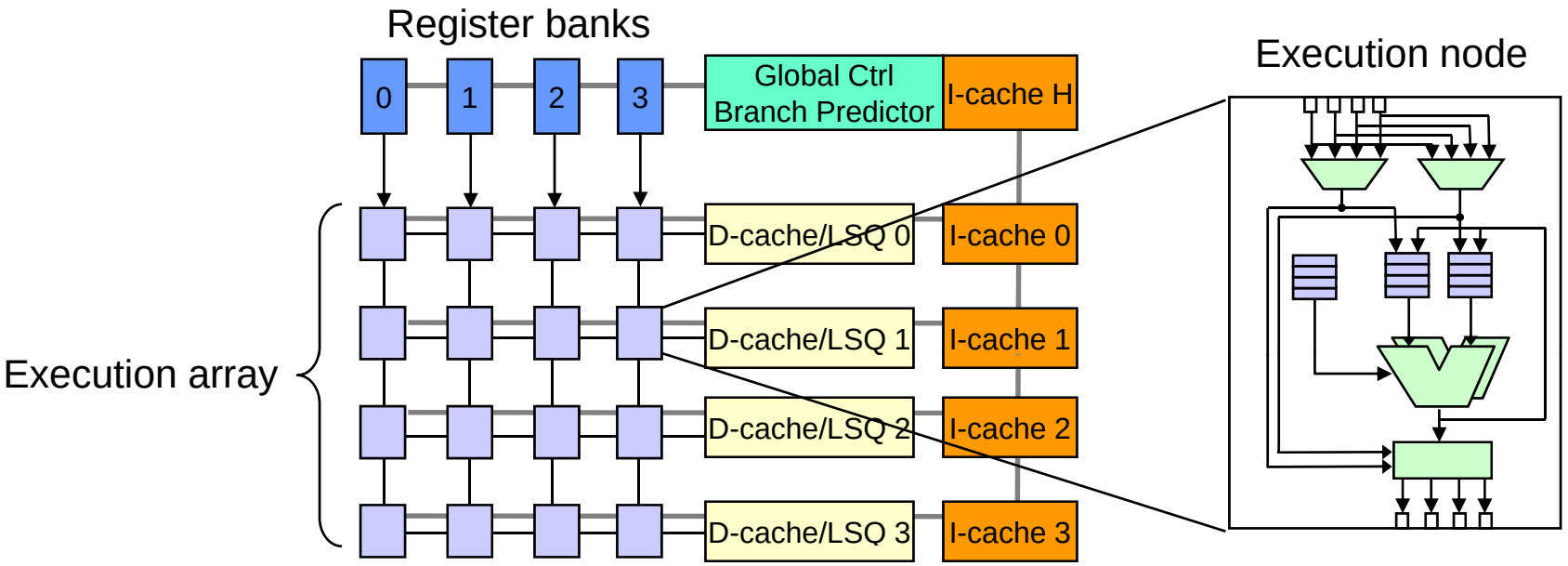

Speedup vs. 1 Raw tile
Raw 1 tile
Entire Raw chip
P3
10.0
1.0
300.twolf
197.parser
175.vpr
SHA
Unstruct
Cholesky
Mxm
Swim
Life
Btrix
Vpenta
FIR
Bitonic
Beamform


Speedup versus P3
Best-in-class envelope
Raw
P3
NEC SX-7
VIRAM
Imagine
16-P3 server farm
FPGA
ASIC
100.0
10.0
1.0
0.1
300.twolf
197.parser
175.vpr
SHA
Unstruct
Cholesky
Mxm
Swim
Life
Btrix
Vpenta
FIR
Bitonic
Beamform
FMRadio
STRM.scale
Cornerturn
Beamsteer
CSLC
FFT
183.equake
177.mesa
301.apsi
181.mcf
256.bzip2
802.11a.64k
8b/10b.64k
ILP
Stream
Server
Bit-level

Plan

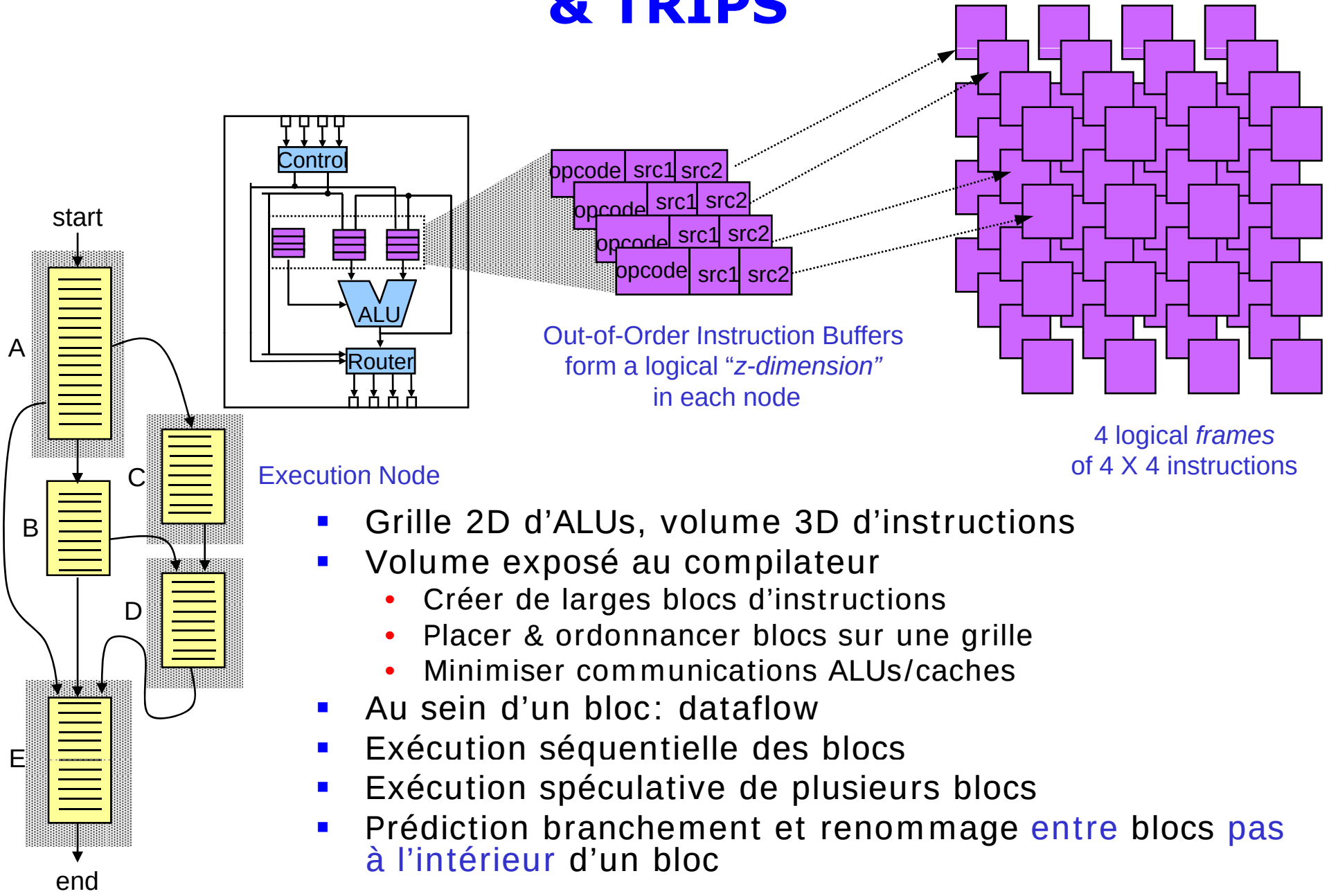
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Grid Processor Architecture (GPA) & TRIPS



- Buts
  - Fréquence élevée, ILP élevé
  - Scalabilité
  - Equilibre architecture/logiciel
- Réduction de la complexité & consommation
- Nouveaux problèmes pour le compilateur
- Prototype en cours de développement

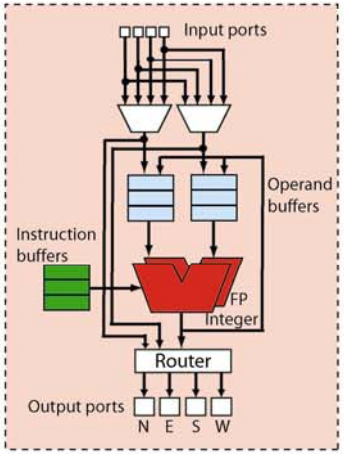
Grid Processor Architecture (GPA) & TRIPS



- Grille 2D d'ALUs, volume 3D d'instructions
- Volume exposé au compilateur
  - Créer de larges blocs d'instructions
  - Placer & ordonnancer blocs sur une grille
  - Minimiser communications ALUs/caches
- Au sein d'un bloc: dataflow
- Exécution séquentielle des blocs
- Exécution spéculative de plusieurs blocs
- Prédiction branchement et renommage entre blocs pas à l'intérieur d'un bloc

Grid Processor Architecture (GPA) & TRIPS

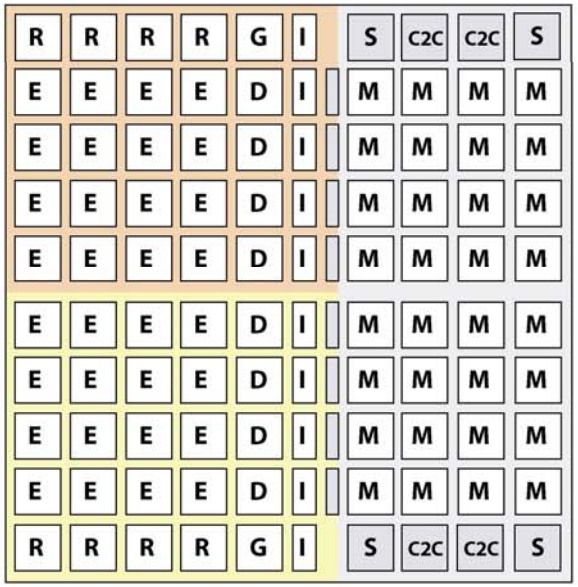
Execution Unit (E tile)



Expected Prototype Chip Specs:

- IBM Cu-11 process
- 18x18 mm chip area
- 500MHz clock
- 2 16-wide processor cores
- 1K instructions mapped per core
- 128KB of L1 I and D storage per core
- 2MB of L2 storage
- 60 GB/s off-chip bandwidth
- 16 Gigaflops/ops peak

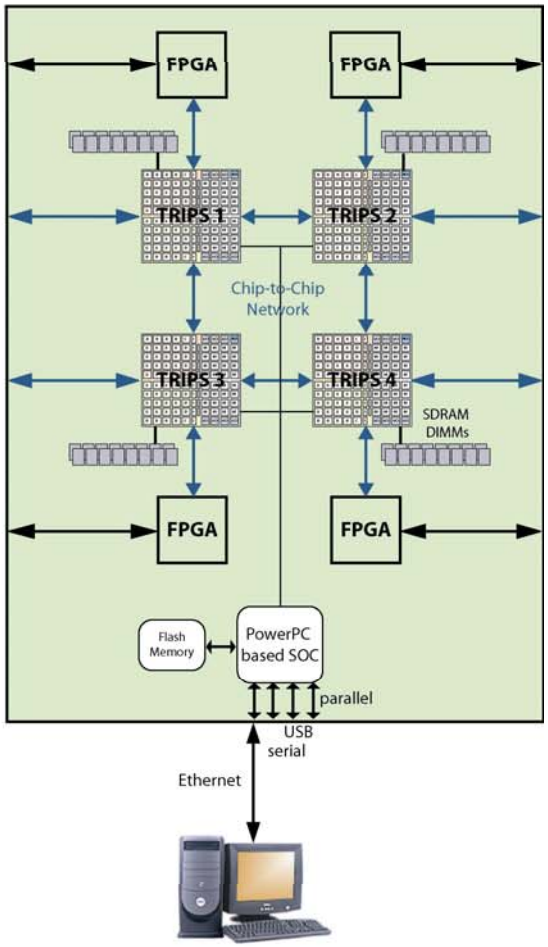
TRIPS Prototype Chip Floorplan



Tiled Architecture and Floorplan

R: Register file tile
E: Execution unit tile (integer and FP)
D: Data cache tile
I: Instruction cache tile
G: Processor control tile
M: Memory tile
S: SDRAM controller
C2C: Chip to chip interconnect

TRIPS Prototype Board

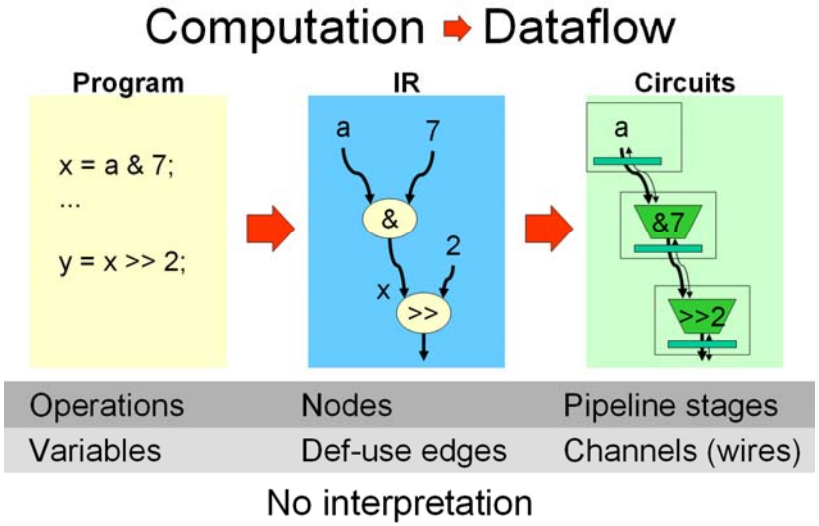


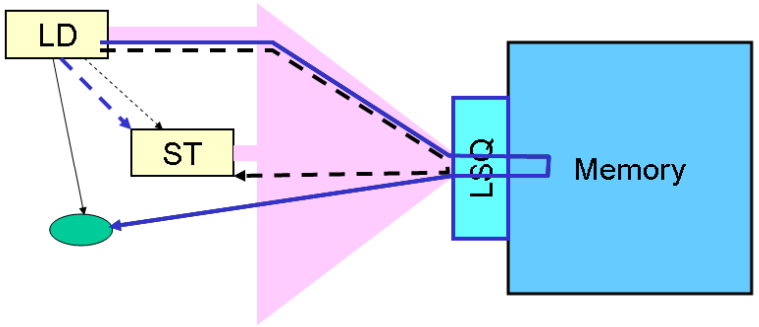
Plan

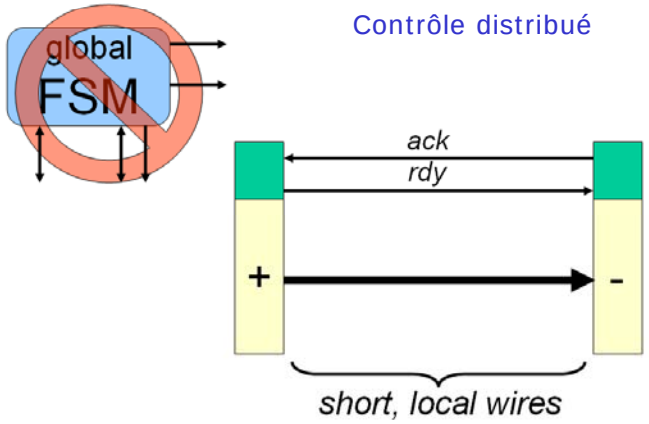
- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Application-Specific Hardware (ASH)

- Traduction de programmes C en hardware
- Pas de contrôle central, contrôle distribué
- Mais mémoire centrale

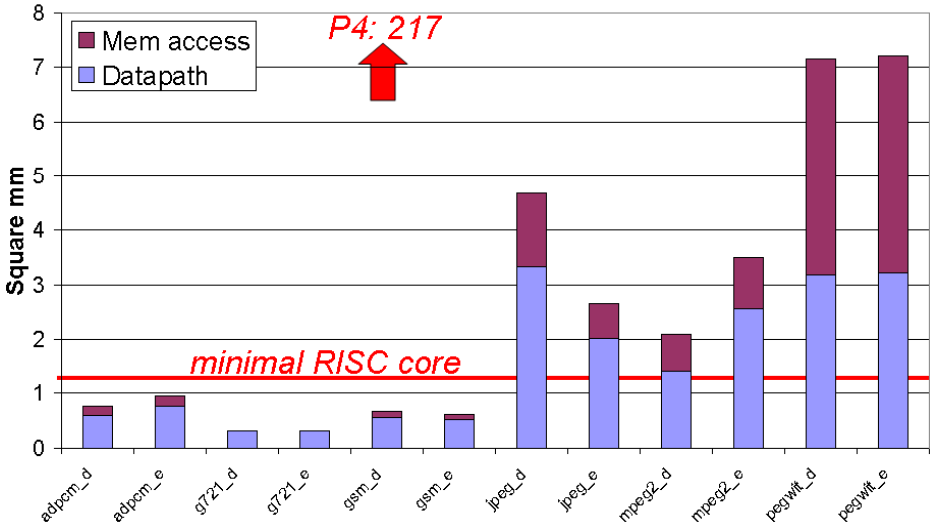


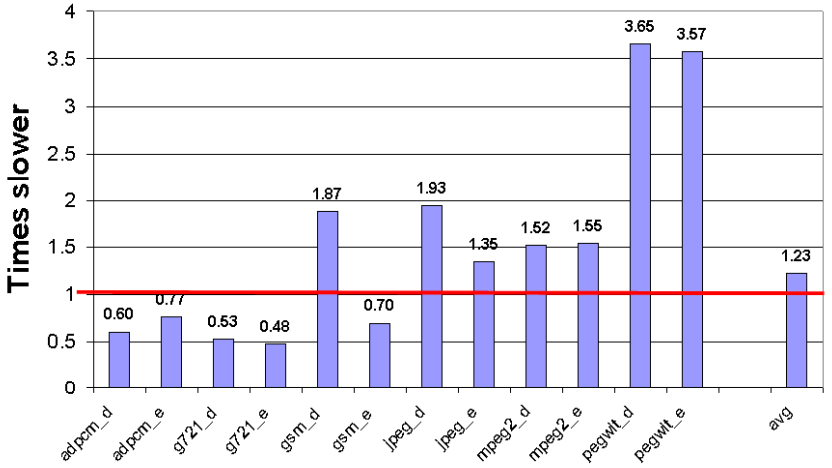


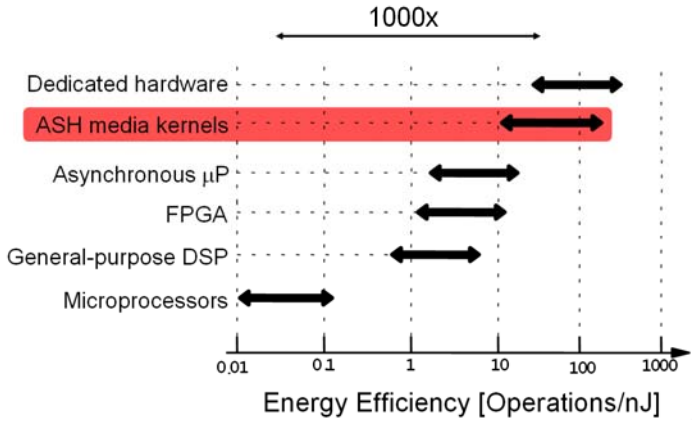


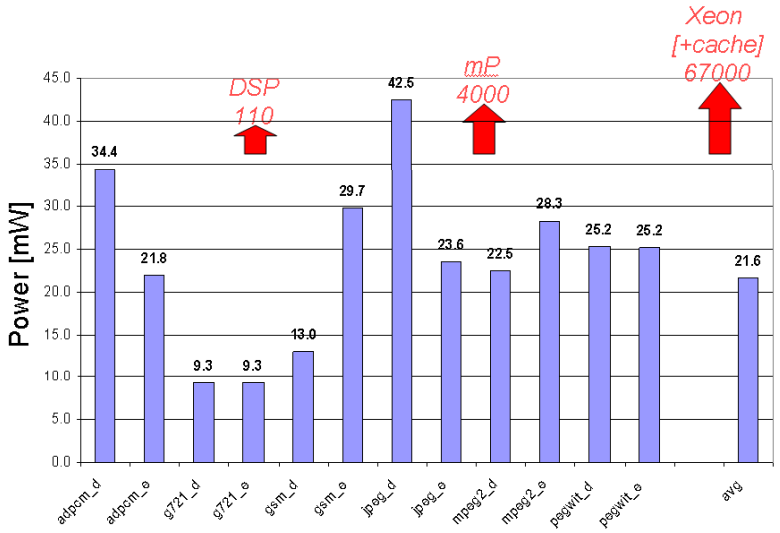
Application-Specific Hardware (ASH)

- Mediabench
- Peu rapide mais très basse consommation
- Espace raisonnable





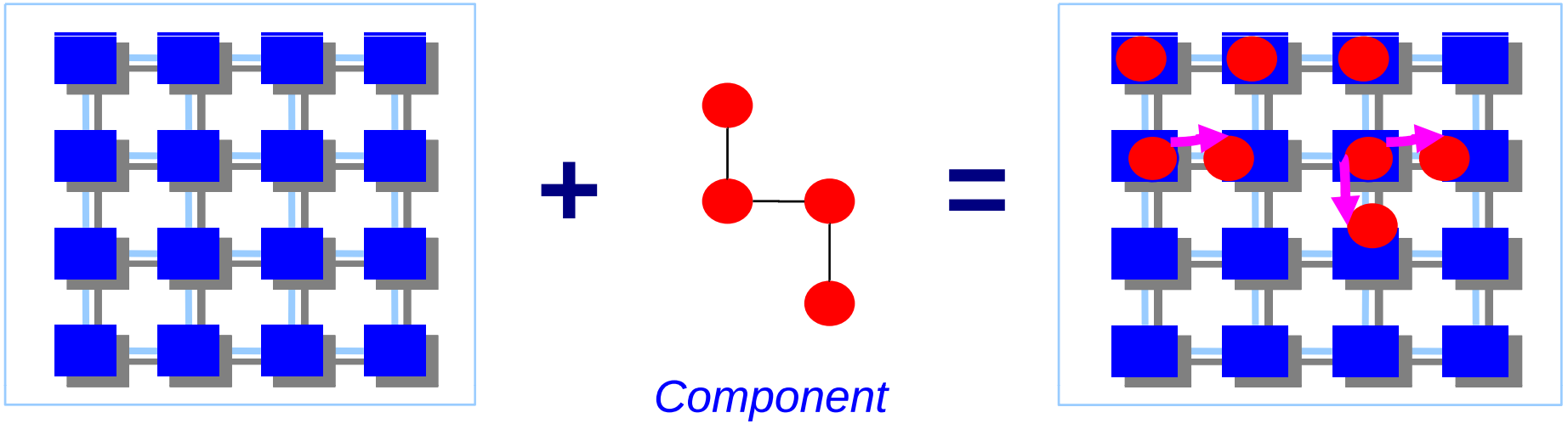




Plan

- Continuer à augmenter fréquence et ILP
  - Allongement du pipeline
  - Accroissement du degré superscalaire
  - Processeurs multithreads (SMTs)
  - Architectures en clusters
- Passer aux CMPs (*Chip Multi-Processors*)
  - Modèles existants
  - Architectures en *tile*
    - RAW
    - GPA/TRIPS
    - ASH
  - Espace & Facilité de programmation

Espace & Facilité de programmation



- Scalabilité = grand nombre de composants
- **Faciliter la programmation parallèle**
  - Modèle simple parallélisme (division)
  - Encapsulation & Communications
  - Expliciter structures de données
- Rôle architecture ou run-time system:
  - Exploiter informations utilisateur
  - Gérer les ressources disponibles

Conclusions

- Très probablement utilisation de l'espace
- Donc parallélisme sur puce
- Sur puce $\Rightarrow$ grande liberté d'architecture, diverses structures
- Principale difficulté: comment extraire le parallélisme sur des applications complexes ?
- Aujourd'hui effort sur architecture OU compilation OU utilisateur
- Réaliser une meilleure symbiose, un meilleur partage des efforts

Stages & Thèses

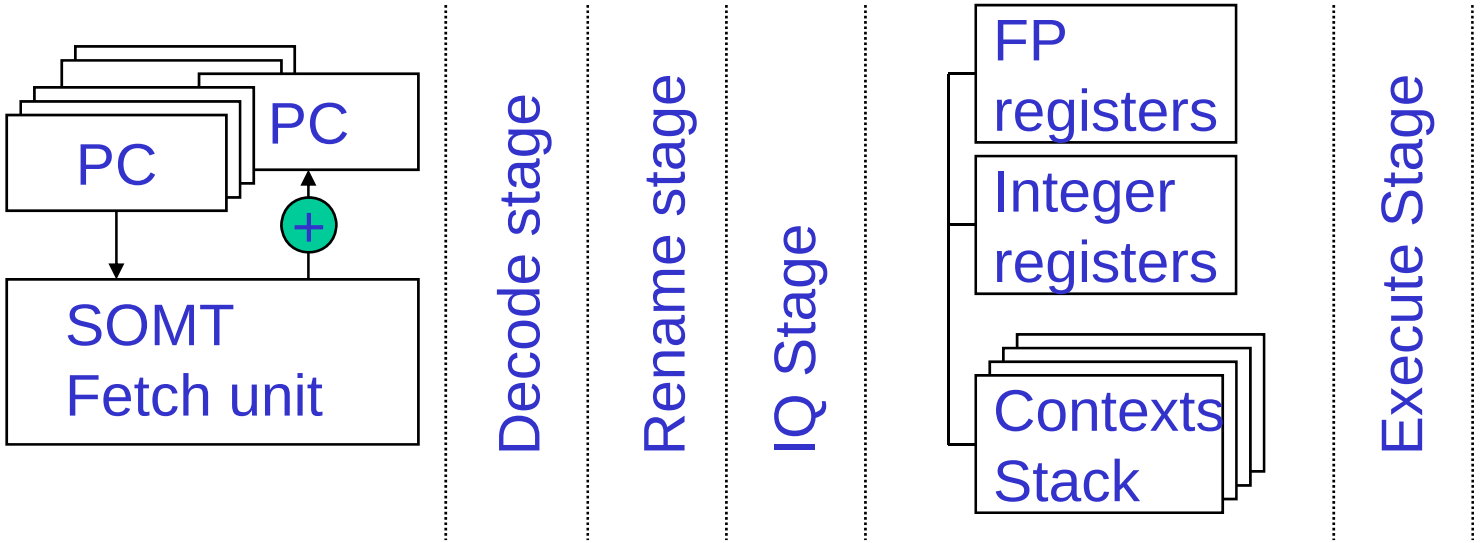
- Architectures CMP & programmation parallèle
- Approches itératives de l'optimisation
- Simulation rapide d'architectures complexes

Symbiotic Processing

- Issues
  - Scaling up processors based on frequency
    - Complex architectures
  - Scaling up processors based on parallelism
    - Complex compilers
  - No much choice, look for (more) **parallelism** beyond current ILP levels
  - Strike right balance between architecture, compiler and **user** effort
  - Focus particularly (but not only) on codes with non-regular control flow/data structures (most difficult cases)

Self-Organized SMT (SOMT)

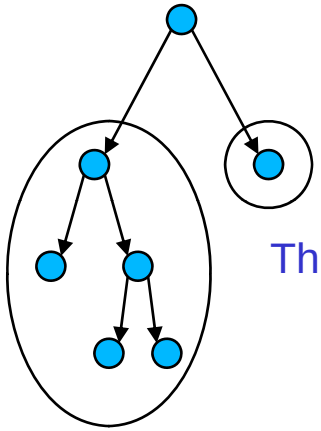
- 1 agent ~ 1 thread
- Dynamic thread creation
  (program enables, architecture decides)
- Thread steering mechanisms (thread swapping)



Experiments

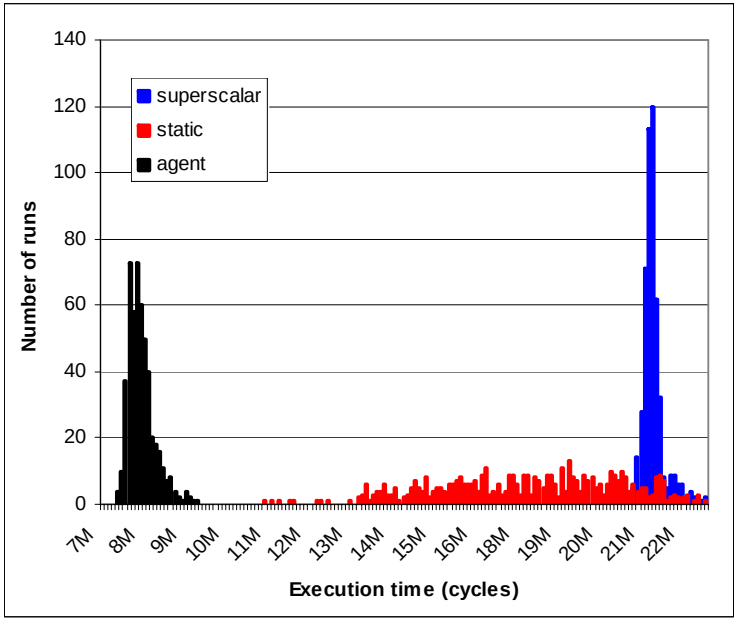
- Benchmarks
  - QuickSort
  - Dijkstra
  - MxV, MxM
  - LZW
  - Perceptron
  - MCF
- 3 versions
  - Superscalar
  - Static parallel
  - Agent program
- Baseline architecture
  - Superscalar
  - 256 in-flight instructions
  - 8 integer ALUs
- SMT (8 threads)
  - ICount 4.4
  - Shared issue, physical registers, FUs
- SOMT
  - SMT + dynamic thread creation & steering

Performance Evaluation
Dynamic Replication



Static Version (unbalanced parallelization)

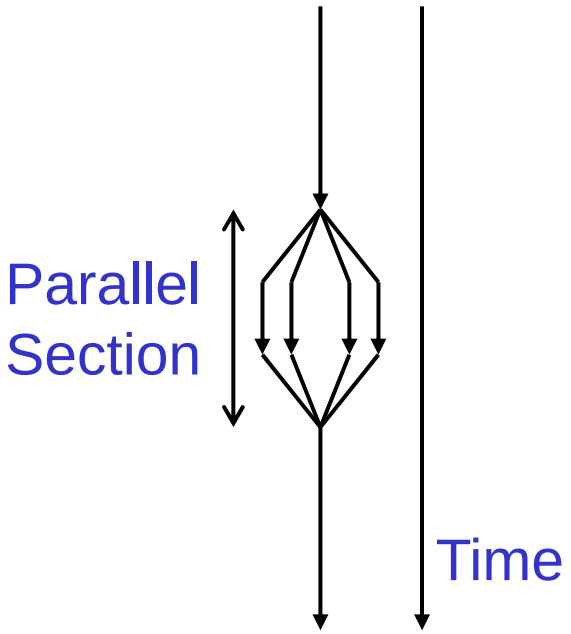
- Dynamic parallelism exploitation in AP+SOMT
- Similar behavior for Dijkstra

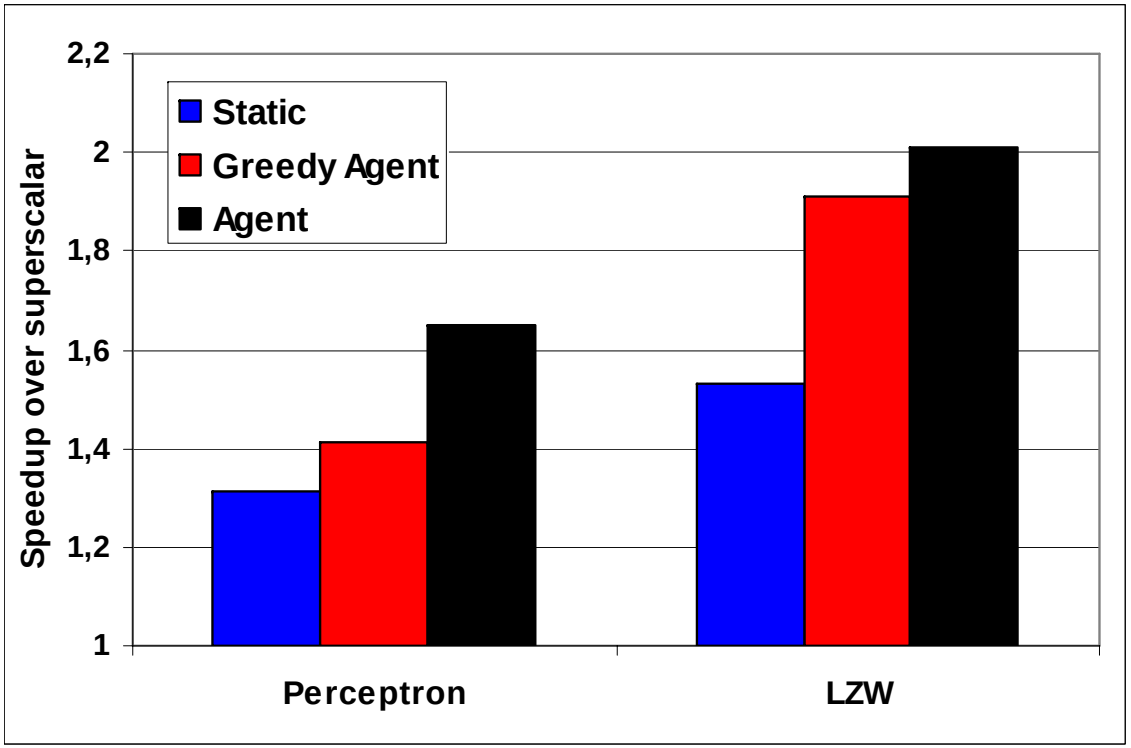


QuickSort: distribution of execution time over 500 datasets

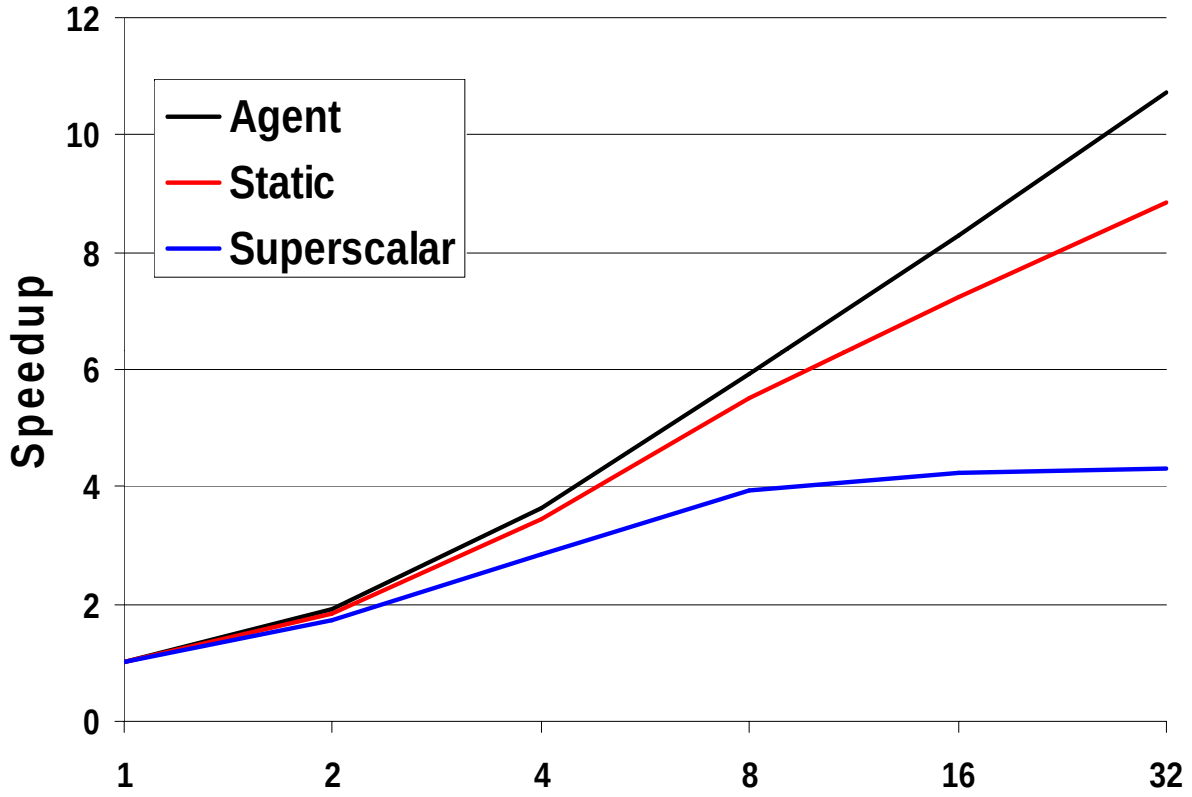
Performance Evaluation

- Slow down thread creation when agents die too quickly (small parallel section)





Performance Evaluation
Scalability



Window size, number of registers… = α.T (number of threads)

Emulating Complex Software Memory Optimizations

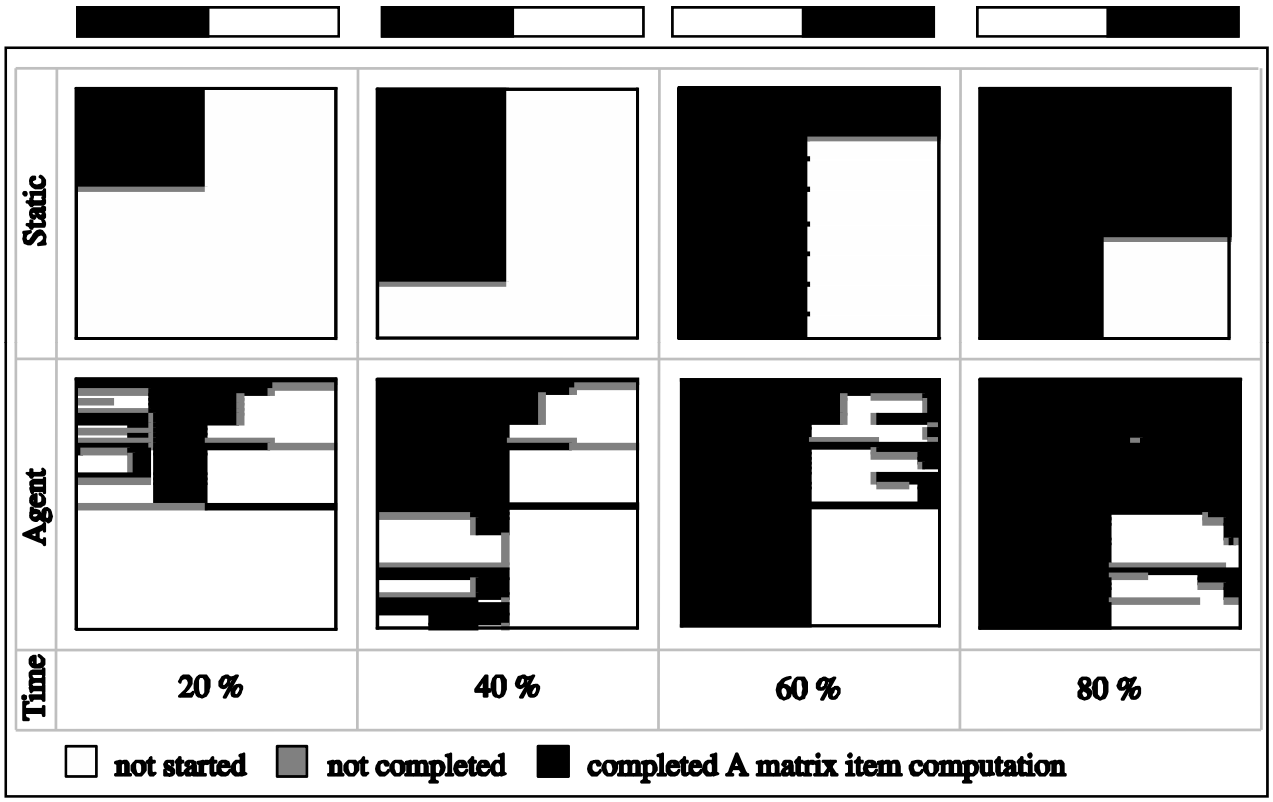
```
void matvec_multiply( int size,
                      double* C,
                      double* A,
                      double* B )
{
  // AP divide
  for( int j = 0; j < size; j++ )
  {
    // AP divide
    for( int i = 0; i < size; i++ )
    {
      C[j] += A[j*size+i]*B[i];
    }
  }
}
```

Agent Programming MxV

```
void matvec_multiply( int size,
                      double* C,
                      double* A,
                      double* B )
{
  for( int ii = 0; ii < size; ii+=B )
  {
    for( int j = 0; j < size; j++ )
    {
      // loop below is parallelized
      for( int i = ii; i < ii + B; i++ )
      {
        C[j] += A[j*size+i]*B[i];
      }
    }
  }
}
```

Tiled and parallelized MxV

Emulating Complex Software Memory Optimizations



Speedup
1.71

Speedup
1.55

Tiling-like behavior with AP+SOMT

- Static optimization: privilege access that reuse data
- Dynamic thread steering: privilege cache well-behaved agents


Gartner
Gartner
Gartner
Gartner