# Simultaneous Multithreading and the Case for Chip Multiprocessing

**John Mellor-Crummey**

**Department of Computer Science**
**Rice University**

**johnmc@rice.edu**

# Microprocessor Architecture (Mid 90's)

- Superscalar (SS) designs were the state of the art
  - —multiple instruction issue
  - —dynamic scheduling: h/w tracks instruction dependencies
  - —speculative execution: looks past predicted branches
  - —non-blocking caches: multiple outstanding memory accesses
- Transistors shrinking, count per chip doubling every 24 months
  - —more transistors supports more logic
  - —smaller transistors enable higher clock frequencies
- Apparent path to higher performance?
  - —more functional units
  - —wider instruction issue
  - —support for more speculation

# Impediments to Higher Performance

**Claim: ⇑ issue width of SS will provide diminishing returns**

**Factors in 1996**

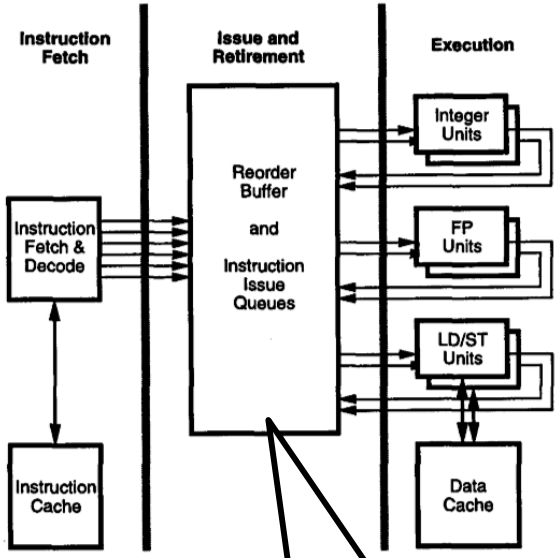
- **Fundamental circuit limitations**
- **Limited amount of instruction-level parallelism**
  - **—"one in every five instructions is a branch"**
    **Hennessy and Patterson, Computer Architecture, 1996**
- **Long latency memory operations**

# Superscalar Designs (e.g. R10K, PA-8K)

- 3 phases
  - —instruction fetch and decode
  - —issue and retirement
  - —execution
- Circuit-level performance limiters?
  - —wider issue widths increase need for deeper instruction issue Q's for sufficient ||-ism
    - – claims:

      quadratic increase in size of issue Q

      limiting factor: long wires for broadcasting tags of issued instructions limit cycle time




PA-8K:
56-ins Q
20% die area

  - —execution phase: quadratic ⇑ register file, quadratic ⇑ bypass logic
    - – Farkas et al '96: 8-issue perf only 20% > 4-issue (reg file complexity limits perf)

**Circuit technology impact**

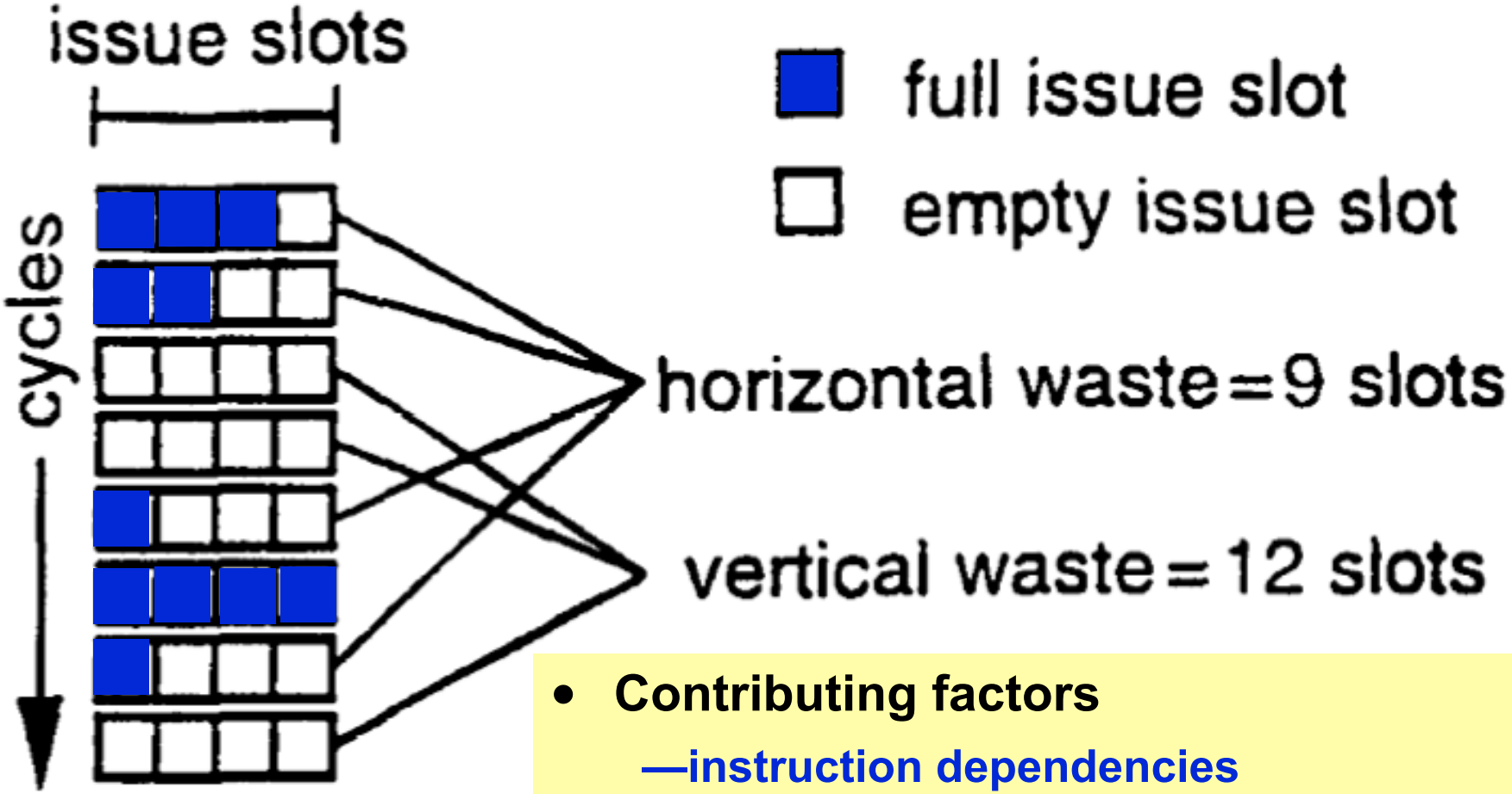
- Delays ⇑ as issue queues ⇑ and multi-port register files ⇑
- Increasing delays limit performance returns from wider issue

The case for a single-chip multiprocessor, Olukotun et al., ASPLOS-VII, 1996.

# Instruction-level Parallelism Concerns

## Issue Waste

Simultaneous multithreading: maximizing on-chip parallelism, Tullsen et. al. ISCA, 1995.




- **Contributing factors**
  - —instruction dependencies
  - —long-latency operations within a thread

**How do they contribute to waste?**

# Sources of Wasted Issue Slots and Remedies

- TLB miss
  - —larger TLB; h/w instruction prefetching; h/w or s/w data prefetch
- I cache miss
  - —larger icache, more icache associativity; h/w prefetch
- D cache miss
  - —larger, more associativity, prefetching, more dynamic execution
- Control hazard
  - —speculative execution; aggressive if-conversion
- Branch misprediction
  - —better prediction logic; lower mispredict penalty
- Load delays (L1 hits)
  - —shorten load latency; better scheduling
- Instruction dependences
  - —better instruction scheduling
- Memory conflict (multiple access to same location in a cycle)
  - —improved instruction scheduling

# How Much IPC is There?

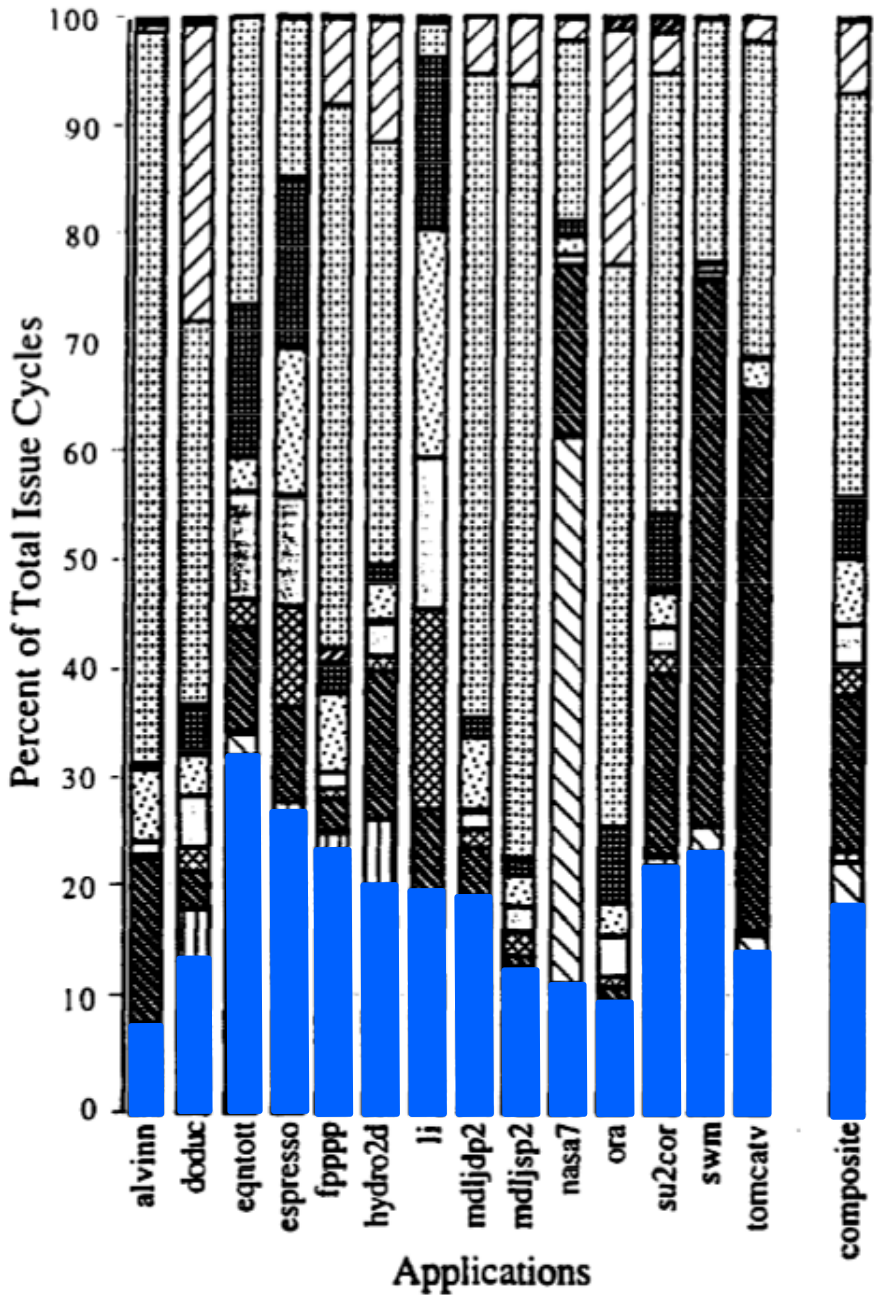
- **Approach: study applications and evaluate their characteristics**
  - **—assess quantity and character of parallelism present**
- **Are there any pitfalls to this approach?**
- **Is there any better approach?**
  - **—what kinds of applications will be important for next-generation architectures?**
    - **speech recognition, e.g., Apple's Siri, Microsoft Cortana**

**Intel's Recognition, Mining, Synthesis Research Initiative**

**Compute-Intensive, Highly Parallel Applications and Uses**
Intel Technology Journal, 9(2), May 19, 2005
DOI: 10.1535/itj.0902
http://www.intel.com/technology/itj/2005/volume09issue02/

graphics, computer vision, data mining, large-scale optimization

# Simulations of 8-issue Superscalar





Simultaneous multithreading: maximizing on-chip parallelism, Tullsen et. al. ISCA, 1995.

**Summary: Highly underutilized**

**Applications: most of SPEC92**

- On average < 1.5 IPC (19%)
- Dominant waste differs by application
- Short FP dependences: 37%

# Analysis of 8-issue Simulations

- No dominant cause of wasted cycles, but 61% vertical waste
- No "silver bullet" solution
  - —no single latency-tolerance technique likely to help dramatically
- Even if memory latencies eliminated, utilization < 40%
- Tullesen et. al. claim
  - —"instruction scheduling targets several important segments of the wasted issue bandwidth, but we expect that our compiler has already achieved most of the available gains in that regard"

    (using Multiflow trace-scheduling compiler)
- If specific latency hiding mechanisms limited, then need general latency hiding solution to increase parallelism

# What Should be Next?

- If not ⇑ superscalar issue width, then what?
- Alternatives
  - —single chip multiprocessor
  - —simultaneous multithreaded processor
- How should we decide?
- Best approach depends upon application characteristics

# The Case for a Single-chip Multiprocessor

# The Case for a Single Chip Multiprocessor

**Two motivations**

- **"Technology push"**
  - **—delays ⇑ as issue queues ⇑ and multi-port register files grow**
  - **—increasing delays limit performance returns from wider issue**
- **"Application pull"**
  - **—limited IPC is a problem**

# Comparing Alternative Designs

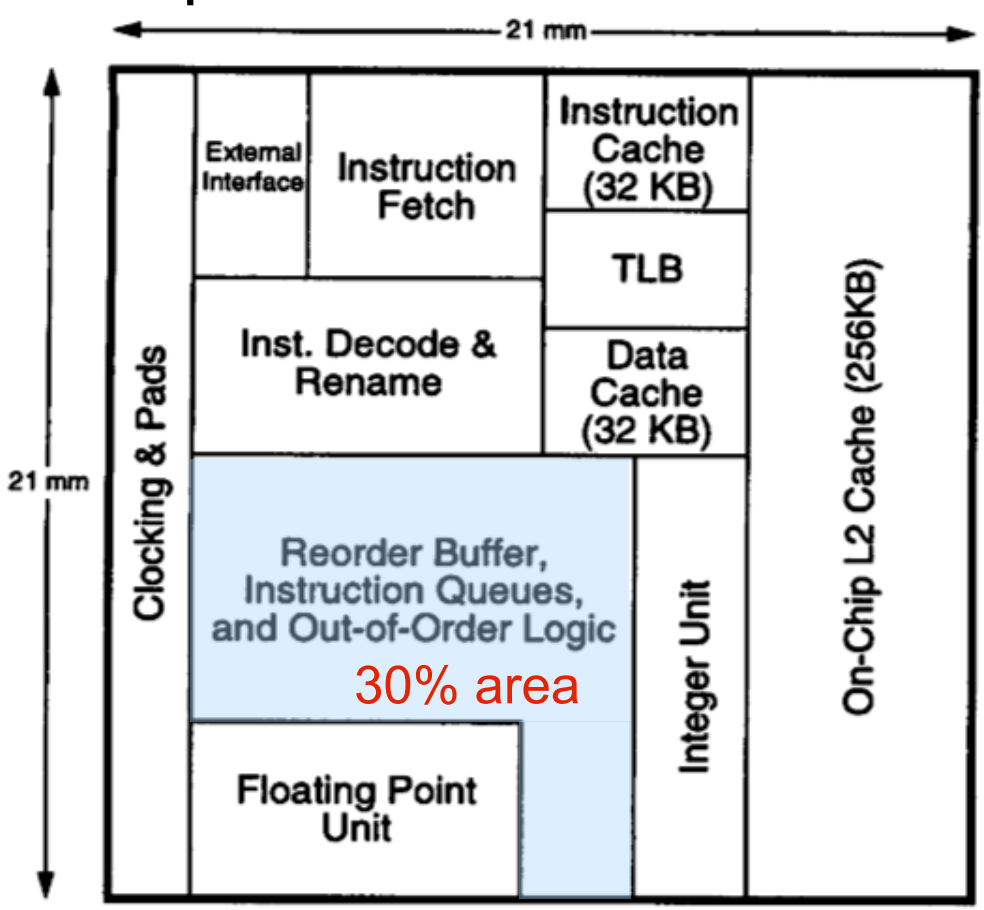
- **6-issue superscalar architecture**
- **4 x 2-issue superscalar multiprocessor architecture**

| | 6-way SS | 4x2-way MP |
|---|---|---|
| # of CPUs | 1 | 4 |
| Degree superscalar | 6 | 4 x 2 |
| # of architectural registers | 32int / 32fp | 4 x 32int / 32fp |
| # of physical registers | 160int / 160fp | 4 x 40int / 40fp |
| # of integer functional units | 3 | 4 x 1 |
| # of floating pt. functional units | 3 | 4 x 1 |
| # of load/store ports | 8 (one per bank) | 4 x 1 |
| BTB size | 2048 entries | 4 x 512 entries |
| Return stack size | 32 entries | 4 x 8 entries |
| Instruction issue queue size | 128 entries | 4 x 8 entries |
| I cache | 32 KB, 2-way S. A. | 4 x 8 KB, 2-way S. A. |
| D cache | 32 KB, 2-way S. A. | 4 x 8 KB, 2-way S. A. |
| L1 hit time | 2 cycles (4 ns) | 1 cycle (2 ns) |
| L1 cache interleaving | 8 banks | N/A |
| Unified L2 cache | 256 KB, 2-way S. A. | 256 KB, 2-way S. A. |
| L2 hit time / L1 penalty | 4 cycles (8 ns) | 5 cycles (10 ns) |
| Memory latency / L2 penalty | 50 cycles (100 ns) | 50 cycles (100 ns) |

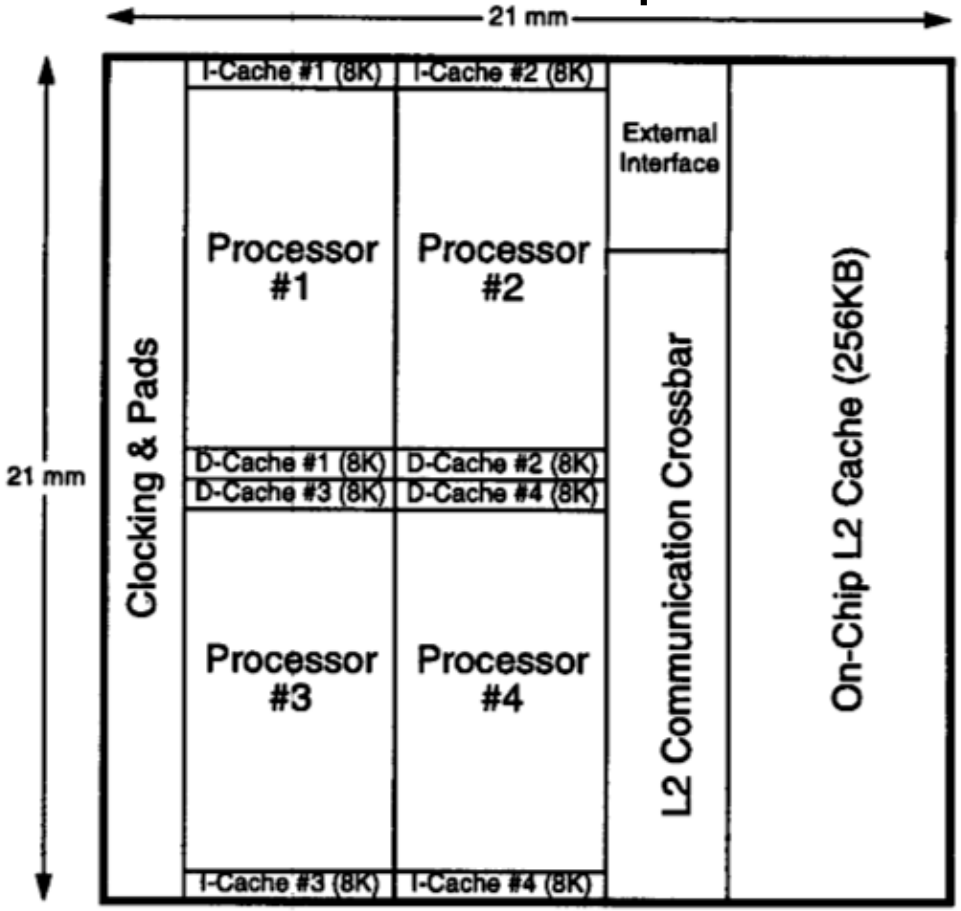
6-issue design scaled from 4-issue MIPS R10000;
2-issue design based on Alpha 21064

# Floorplans: 6-issue SS vs. 4 x 2 CMP

Extrapolate sizes based on a 4-issue MIPS R10000 processor




6-issue superscalar




4 x 2-issue CMP

**4 x 2 CMP differences**

- —simpler issue logic
- —much simpler renaming logic
- —1/4 size branch prediction buffer
- —more execution units

# Integer Benchmarks

- **eqntott**: translates logic equations into truth tables
  - —manually parallelized bit vector comparison routine (90% time)
- **compress**: compresses and uncompresses files in memory
  - —unmodified on both SS and SMT architectures
- **m88ksim**: simulates Motorola 88000 CPU
  - —manually parallelized into 3 threads using SUIF compiler
    - threads simulate different instructions in different phases
      parallelization analogous to the h/w pipelining it simulates
- **MPSim**: simulates a bus-based multiprocessor
  - —manually assign parts of model hierarchy to different threads
  - —4 threads: one for each simulated CPU

The case for a single-chip multiprocessor, Olukotun et al., ASPLOS-VII, 1996.

# FP and Multiprogramming Benchmarks

**Floating point applications (all parallelized with SUIF)**

- **applu**: solves parabolic/eliptic PDEs
- **apsi**: computes temp, wind, velocity, and distrib. of pollutants
- **swim**: shallow water model with 1k x 1k grid
- **tomcatv**: generates mesh using Thompson solver

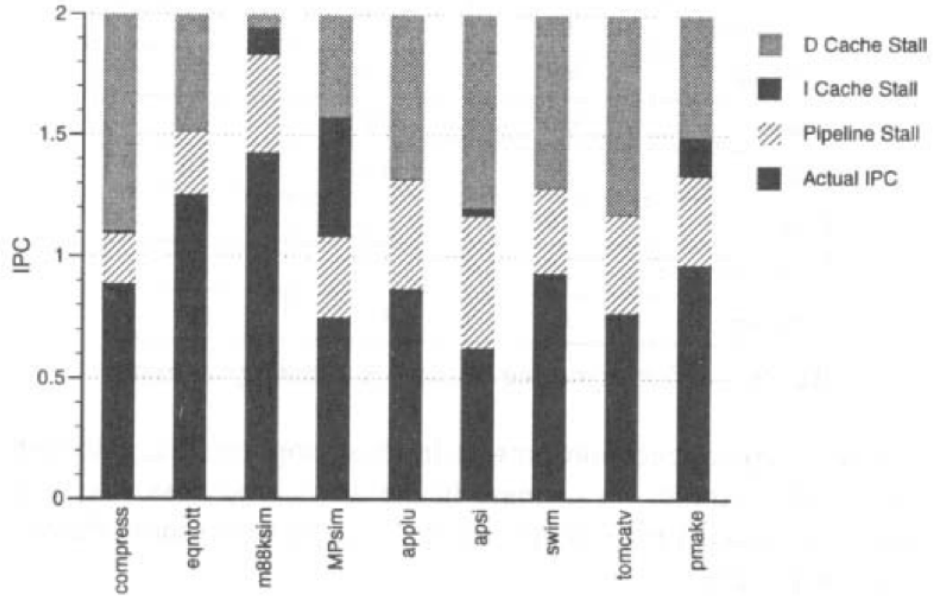
**Multiprogramming application**

- **pmake**: performs parallel make of gnuchess (C compiler)
  - —same application simulated on both architectures
  - —OS exploits extra PEs in MP architecture to run parallel compiles

The case for a single-chip multiprocessor, Olukotun et al., ASPLOS-VII, 1996.

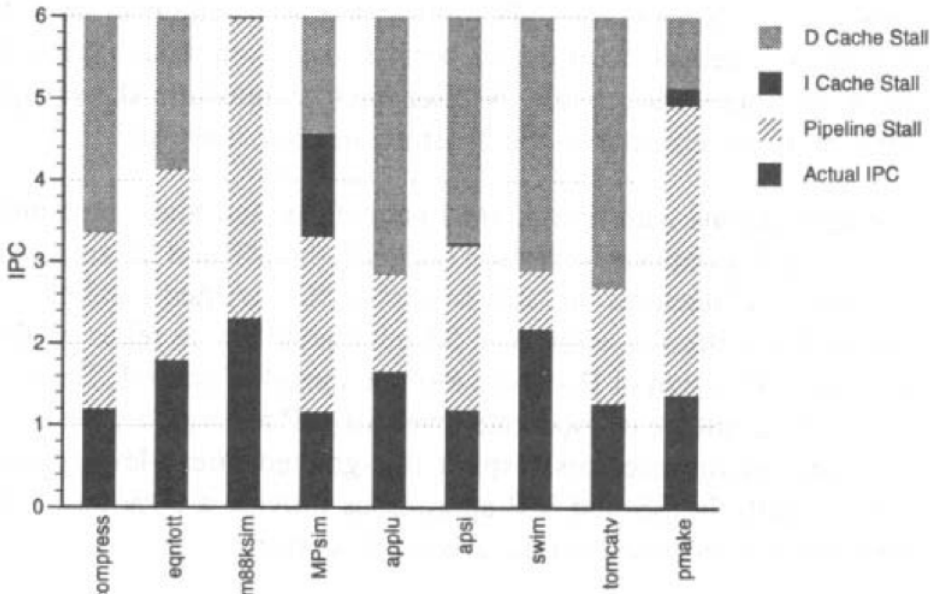
# IPC Breakdown of Superscalar PEs

## 2-issue




The case for a single-chip multiprocessor, Olukotun et al., ASPLOS-VII, 1996.

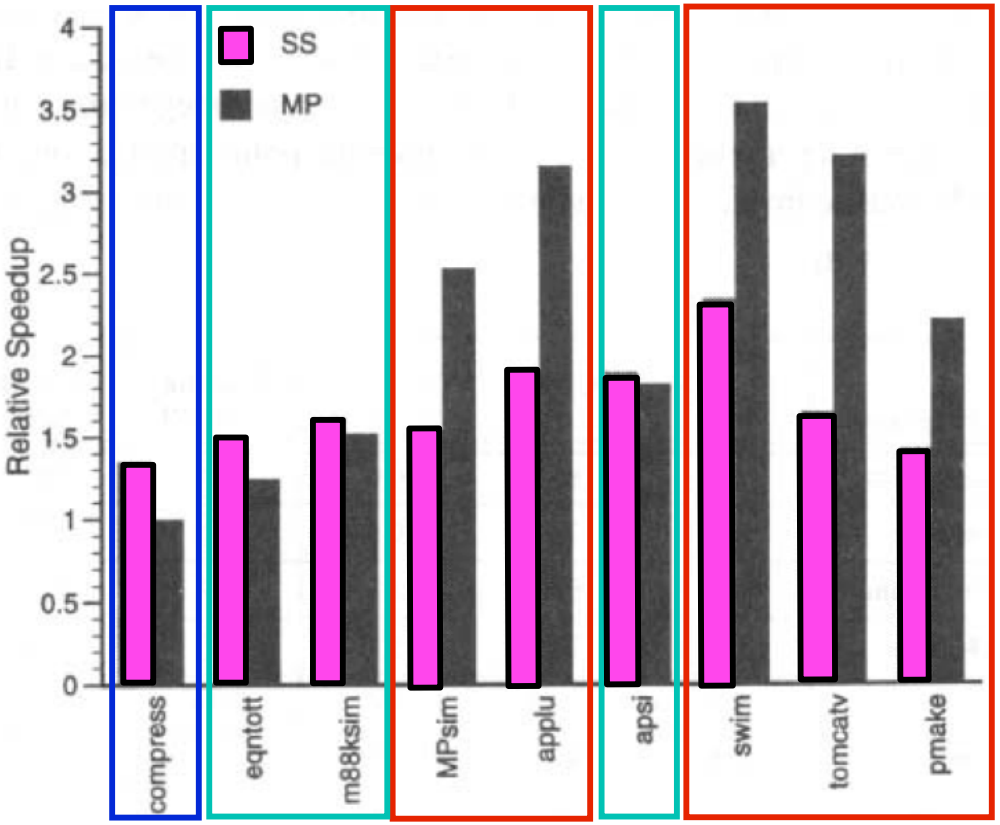
## 6-issue




- **6-issue vs. 2-issue**
  - performance increases 1.6x tomcatv
  - performance increases 2.4x swim

- **2-issue**
  - much of IPC loss due to dcache stalls
- **6-issue**
  - limited ILP increases pipeline stalls
  - fewer icache stalls with larger icache
  - FP appl have significant ILP, but dcache stalls consume > 50% IPC

# Performance: 4 x 2 CMP vs. 6-issue SS

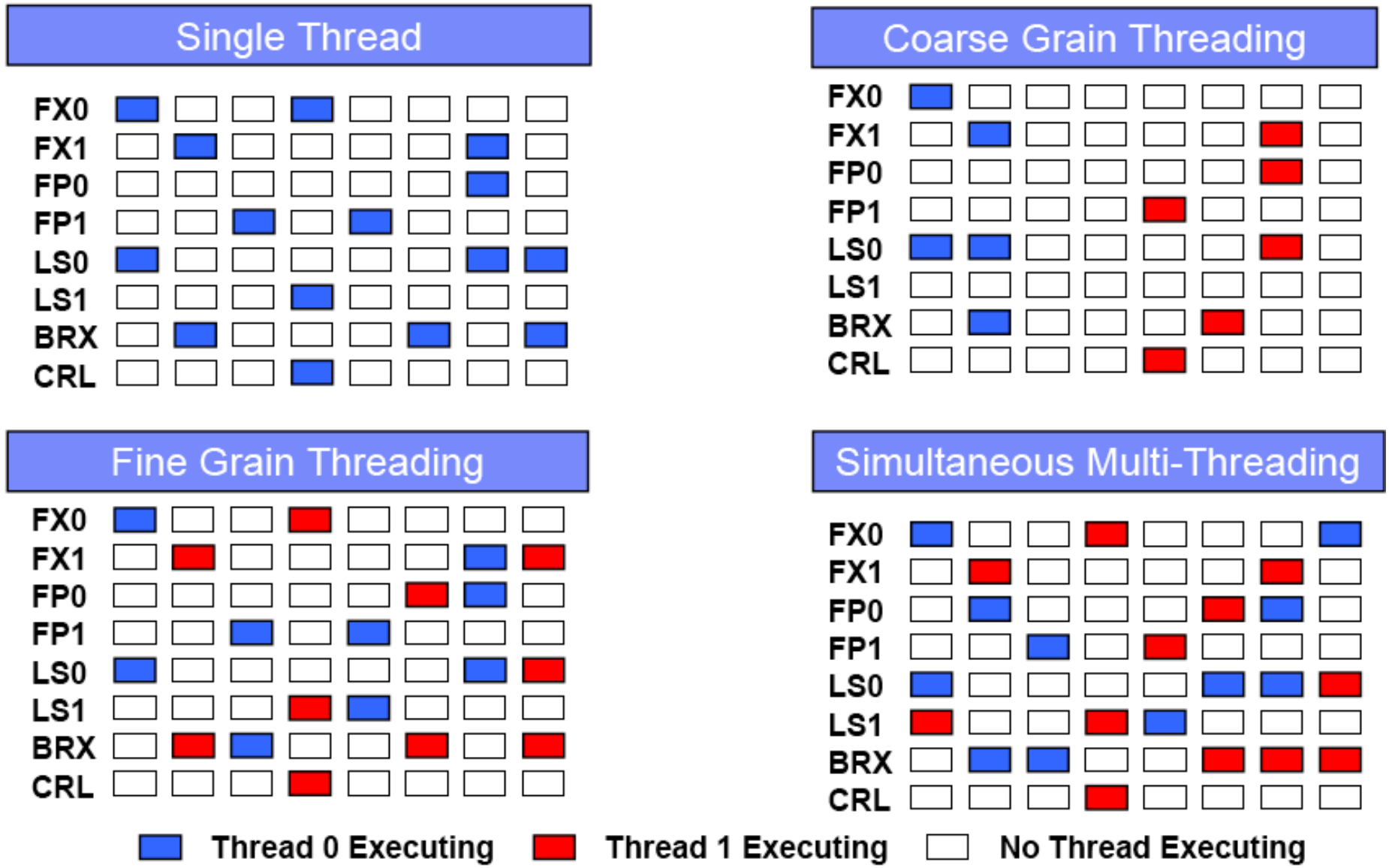



- **Non-parallelizable codes**
  - —wide superscalar architecture performs up to 30% better
- **Codes with fine-grain thread-level parallelism, high communication**
  - —wide superscalar architecture is at most 10% better w/ same clock frequency
  - —expect that simpler CPUs in CMP would support higher clock rates eliminating this advantage

- **Codes with coarse-grain thread-level parallelism and multiprogramming workloads**
  - —CMP performs 50-100% better than wide superscalar
  - —SS less able to dynamically extract parallelism from 128-wide instruction window

# Simultaneous Multithreading: Maximizing On-chip Parallelism

# High-level View of Threading Alternatives




From Hot Chips 15, August 2003
*POWER5: IBM's Next Generation POWER Microprocessor*, Ron Kalla (IBM)

# Simultaneous Multithreading

**Permit several independent threads to issue to multiple functional units each cycle**

- **Objective: increase processor utilization despite …**
  - —long memory latencies
  - —limited available parallelism per thread
- **Design parameters**
  - —number of threads
  - —number of functional units
  - —issue width per thread
  - —binding between thread and units
    - – static
    - – completely dynamic
- **Advantages: combines …**
  - —multiple instruction issue from superscalar architectures
  - —latency hiding ability of multi-threaded architectures

# Goals of SMT Simulations

**Evaluate alternatives**

- **Wide superscalars**
- **Traditional multi-threaded processors**
- **Small-scale multiple issue multiprocessors**

# Alternatives Studied

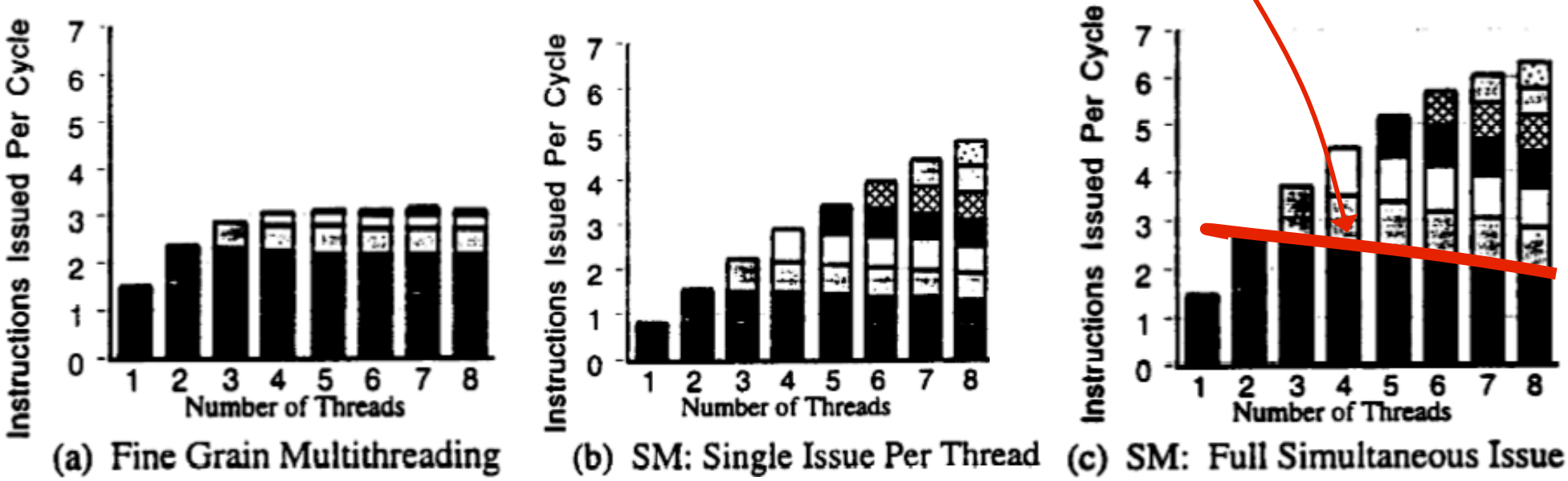
- **Baseline**
  - —fine-grain multi-threading (not SMT; like Tera MTA, Sun Niagara)
    - only one thread issues instructions each cycle
    - can use entire issue width, but admits horizontal waste
    - attacks vertical waste by
      - hiding memory latency
      - hiding functional unit latencies
- **SMT**
  - —SM: full simultaneous issue
    - 8 threads compete for each issue slot each cycle
  - —SM: single issue, SM: dual issue, SM: four issue
    - limit # instructions each thread can issue in a cycle
  - —SM: limited connection
    - connect each h/w context to 1 of each type functional unit
    - each unit can only receive instructions from its attached threads
    - key points
      - partitioning of functional units < dynamic than other models
      - functional units still shared among threads (for high utilization)

# Instruction Scheduling

- Limited support for dynamic execution
  - —8-instruction per thread issue window
    - issues instructions out of order to functional units
  - —dependence-free instructions are issued in-order to scheduling window by each thread
- Use Multiflow trace compiler for good static scheduling
  - —trace scheduling

# SMT Performance: 3 Design Points

slowdown due to contention for other system resources (e.g. cache, TLB, branch predictor)

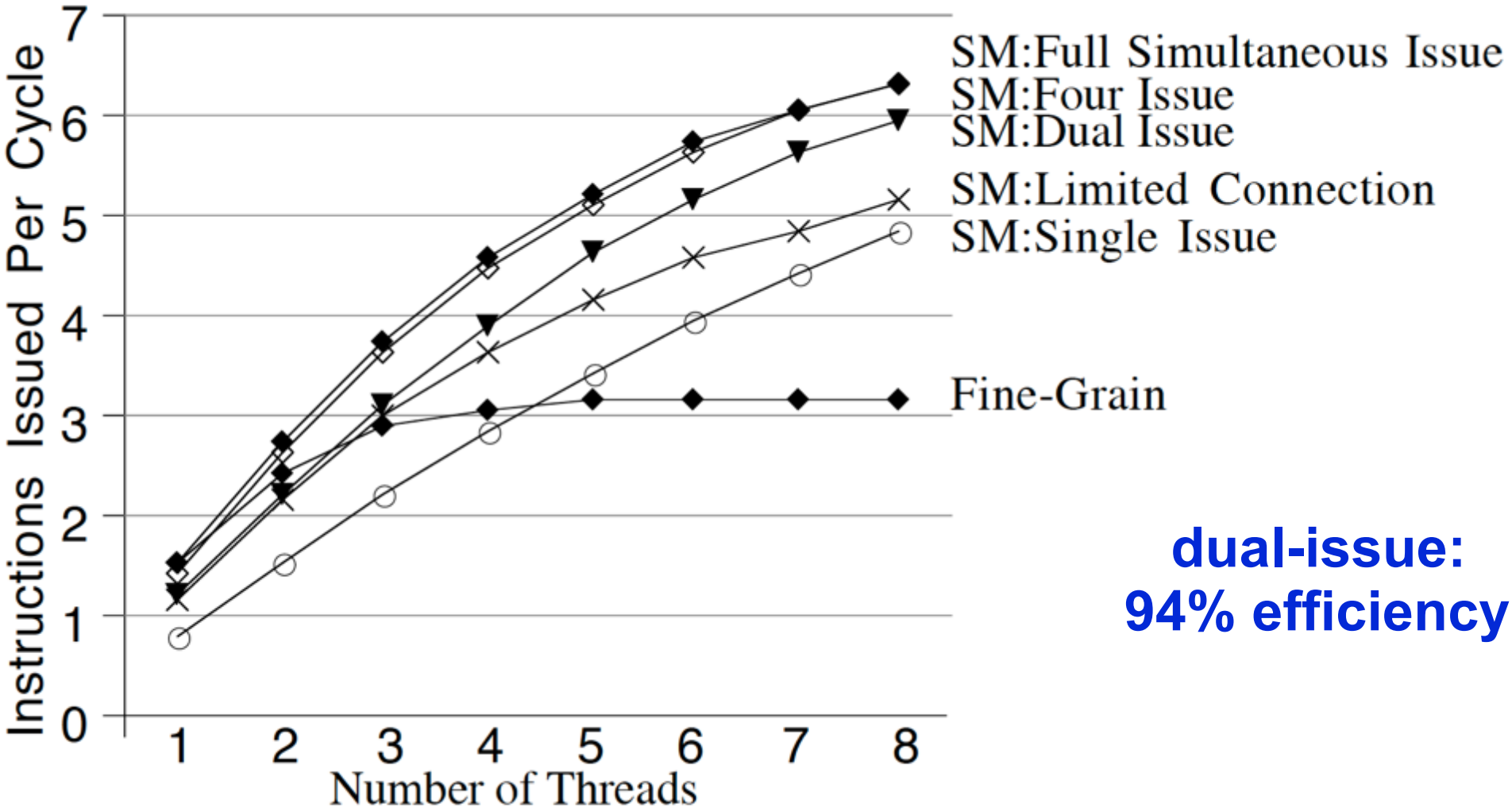



- **Segments of each bar show IPC contribution by each thread**
- **Highest priority thread at the bottom**
- **What do the graphs show?**
- **Claim: fine-grain multithreading only beneficial for 2-5 threads**

**Figure credits: Simultaneous multithreading: maximizing on-chip parallelism, Tullsen et. al. ISCA, 1995.**

# SMT: Comparison of All Models

Figure credit: Simultaneous multithreading: maximizing on-chip parallelism, Tullsen et. al. ISCA, 1995.




**dual-issue:**
**94% efficiency**

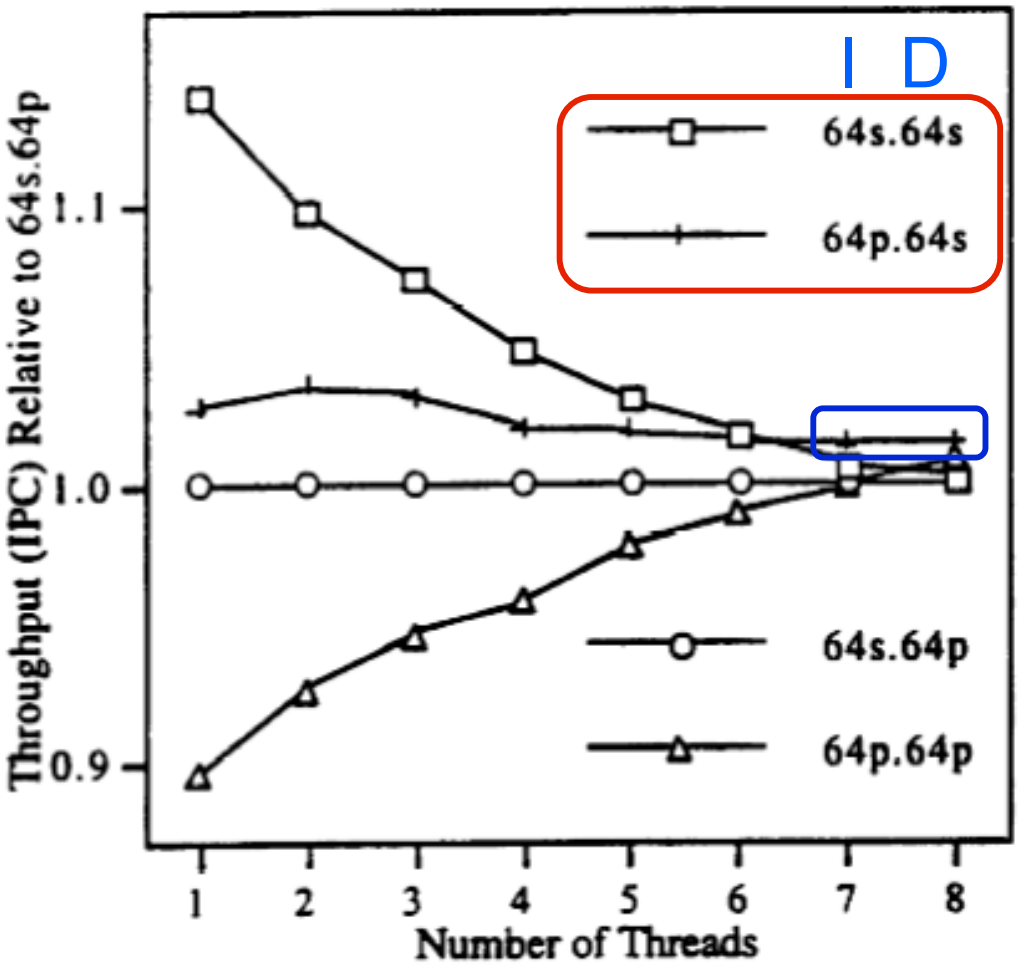
**Q: Why does IPC of fine-grain multi-threading plateau?**

# Cache Issues for SMT

- Decrease in locality from single-threaded model
- Waste due to Icache misses
  - —1% @ 1 thread
  - —14% @ 8 threads
- Waste due to Dcache misses
  - —12% @ 1 thread
  - —18% @ 8 threads
- Waste due to DTLB misses
  - —1% @ 1 thread
  - —6% @ 8 threads
  - —approaches
    - increase shared DTLB from 64 to 96 entries: 1% waste @ 8 threads
    - private DTLB entries: 24 each yields waste of 2%

# Cache Design for SMT

**Private vs. shared L1 cache per thread?**




- **Shared Dcache optimizes for small # threads**
- **Shared Dcache outperforms over all # threads**
- **Private Icaches help @ 7-8 threads**

for performance, which is the best design point?

**Notation: Icache.Dcache (KB)**

# SMT vs. CMP

**Key difference: way resources are partitioned and scheduled**

- —CMP statically partitions resources
- —SMT enables partitioning to change every cycle

| Purpose of Test | Common Elements | Specific Configuration | Throughput (instructions/cycle) |
|---|---|---|---|
| Unlimited FUs: equal total issue bandwidth, equal number of register sets (processors or threads) | **Test A**: FUs = 32<br>Issue bw = 8<br>Reg sets = 8 | SM: 8 thread, 8-issue | 6.64 |
| | | MP: 8 1-issue procs | 5.13 |
| | **Test B**: FUs = 16<br>Issue bw = 4<br>Reg sets = 4 | SM: 4 thread, 4-issue | 3.40 |
| | | MP: 4 1-issue procs | 2.77 |
| | **Test C**: FUs = 16<br>Issue bw = 8<br>Reg sets = 4 | SM: 4 thread, 8-issue | 4.15 |
| | | MP: 4 2-issue procs | 3.44 |
| Unlimited FUs: Test A, but limit SM to 10 FUs | **Test D**:<br>Issue bw = 8<br>Reg sets = 8 | SM: 8 thread, 8 issue, 10 FU | 6.36 |
| | | MP: 8 1-issue procs, 32 FU | 5.13 |
| Unequal Issue BW: MP has up to four times the total issue bandwidth | **Test E**:<br>FUs = 32<br>Reg sets = 8 | SM: 8 thread, 8-issue | 6.64 |
| | | MP: 8 4-issue procs | 6.35 |
| | **Test F**:<br>FUs = 16<br>Reg sets = 4 | SM: 4 thread, 8-issue | 4.15 |
| | | MP: 4 4-issue procs | 3.72 |
| FU Utilization: equal FUs, equal issue bw, unequal reg sets | **Test G**:<br>FUs = 8<br>Issue BW = 8 | SM: 8 thread, 8-issue | 5.30 |
| | | MP: 2 4-issue procs | 1.94 |

# SMT Claims

- SMT faster by 24%
  - —single SMT processor with 10 functional units
  - —conventional 8-PE single-issue MP with 32 functional units
- Study caveats
  - —architecture not built
  - —results may be optimistic in two respects
    - – number of pipeline stages required for instruction issue
      - Alpha uses 2 pipeline stages for issue
      - if SMT requires more issue stages, increase mispred penalty
    - – data cache access time for a shared cache
  - —this study should serve as an upper bound of SMT performance

# SMT Experience with IBM's Power5

- Most workloads benefited from SMT: gains -11% to +41%
- Main cause of performance degradation: load/store activity
- Lower values of SMT gain came from
  - —repeated invalidation and reloading of L2 lines by 2 programs
  - —higher rejection of loads/stores as a result
  - —higher number of loads satisfied from L3 and memory
    - data footprints wouldn't fit into L2 cache
- SMT cost/benefit assessment
  - —24% chip area for SMT support yields 20.5% average SMT gain
- From LLNL: Using Power5 SMT on ASCI Purple
  - —use no more than 1 MPI task per CPU
  - —use 2nd virtual CPU for use by auxiliary threads or daemons
  - —performance characteristics vary
    - sPPM and UMT2K realize a 20-22% performance gain (says IBM)

# Discussion Questions

- What questions do these papers leave unanswered?
- What are the implications of SMT for the memory subsystem?
- Interactions between the two techniques?
  - —synergies
  - —interference
- How should the techniques be combined?
  - —what is the right balance point between the techniques?
    - how much superscalar execution per h/w thread?
    - how many simultaneous multithreads per core?
    - how many cores?
  - —how should one determine this?

# References


- **Simultaneous multithreading: maximizing on-chip parallelism**, Dean Tullsen, Susan Eggers, and Henry Levy, In 25 Years of ISCA, 1995.
- **The case for a single-chip multiprocessor**, K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K. Chang, ASPLOS-VII, 1996.
- **Characterization of simultaneous multithreading (SMT) efficiency in POWER5**, H. M. Mathis, A. E. Mericas, J. D. McCalpin, R. J. Eickemeyer, and S. R. Kunkel, IBM Journal of Research and Development, 49(4/5), 2005.
- **Purple: Fifth Generation ASC Platform,** https://asc.llnl.gov/computing_resources/purple/
- **Lecture Notes: Out-of-Order Processors.** Rajeev Balasubramonian October 13, 2007. http://www.eng.utah.edu/~cs6810/ooo.pdf