Simultaneous Multithreaded Processors

Susan Eggers, Hank Levy
Aaron Eakin, Luke McDowell, Dimitriy Portnov,
Josh Redstone, Steve Swanson, Mike Swift

University of Washington

Jack Lo (Transmeta), Dean Tullsen (UC San Diego),
Brian Dewey & Manu Thambi (Microsoft), Sujay Parekh (IBM Yorktown)
Luiz Barroso, Joel Emer, Kourosh Gharachorloo, Rebecca Stamm (Compaq)

The Executive Summary

Simultaneous Multithreading (SMT) processor

- a multithreaded out-of-order superscalar
- two key features
  - multiple threads can issue instructions to multiple functional units in every cycle
  - threads dynamically share almost all hardware resources each cycle

Consequence:

- speedups over superscalars of 2X - 4X
- with only a 10% increase in hardware cost

Successful commercialization (Compaq, Intel, XStream Logic, etc.)

The Problem We Were Solving

Lots of hardware in current microprocessors to exploit ILP

- multiple instruction issue with multiple functional units
- out-of-order execution
- lock-up-free caches
- branch prediction
- ...

Precious little ILP in programs to exploit

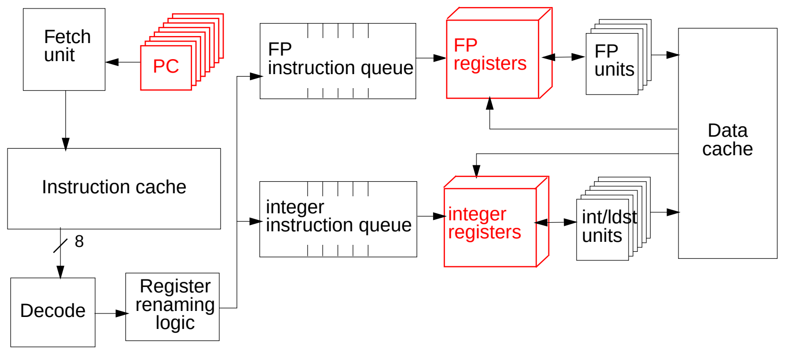
- IPC of about 1.5 on an in-order superscalar, 2.2 on an out-or-order superscalar

An SMT Architecture

Achieved 3 original design goals:

1. Achieve significant throughput gains with many threads
   - SMT outperforms superscalars by 2-4X
2. Minimize the performance impact on a single thread executing alone
   - < 1.5% slowdown
3. Minimize the architectural impact on a conventional out-of-order superscalar design
   - 10% increase in chip area

Implementing SMT (hardware viewpoint)



From OOO Superscalar to SMT

Extra pipeline stages for accessing **thread-shared register files**

- 8 threads * 32 registers + renaming registers

SMT instruction fetcher

- fetch from the two highest throughput threads (called ICOUNT)

Small items

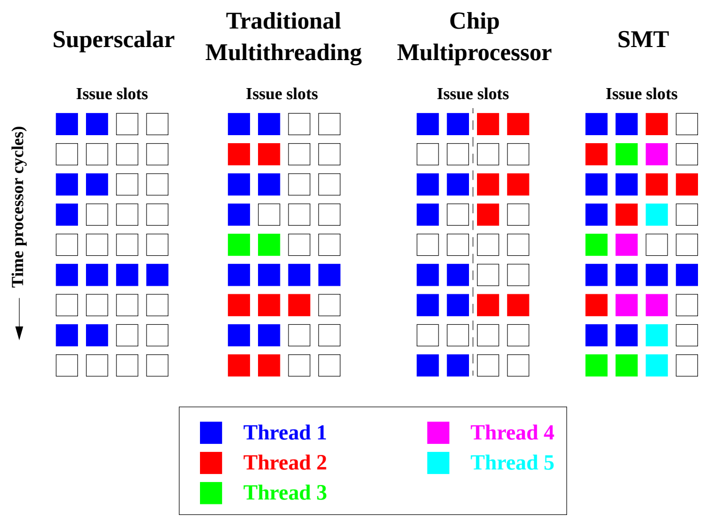
- per-thread program counters
- per-thread return stacks
- per-thread bookkeeping for instruction retirement, trap and instruction dispatch queue flush
- thread identifiers, e.g., with BTB entries, TLB entries

How SMT Increases IPC

SMT converts thread-level parallelism (TLP) to instruction-level parallelism (ILP)

- fetches & executes multiple threads simultaneously (every cycle)
  - many threads contribute to IPC
- all threads share hardware resources dynamically
  - flexible multi-thread utilization
  - single thread has access to whole machine

Comparison of Issue Capabilities



SMT Research to Date

Concept & Potential of Simultaneous Multithreading: ISCA ‘95
& ISCA 25th Anniversary Anthology

SMT’s Microarchitecture: ISCA ‘96

Performance Comparisons:

- Multiprogramming workload: IEEE Micro ‘97
- Parallel programs: TOCS ‘97
- Commercial databases: ISCA ‘98

Tuning Compiler Optimizations for SMT: Micro ‘97 & IJPP ‘99

Cheap SMT Synchronization: HPCA ‘98

Software-directed Register Deallocation: TPDS ‘99

Analysis of OS Behavior on SMT: ASPLOS ‘00