Simultaneous Multithreading

Motivation

Hardware Multithreading

Superscalar Execution
Superscalar Execution

Superscalar Execution with Multithreading

Thread 1
Thread 2
Thread 3

Horizontal waste
Vertical waste
Simultaneous Multithreading

Issue Slots

Thread 1
Thread 2
Thread 3
Thread 4
Thread 5

The Potential for SMT

Number of Threads

Simultaneous Multithreading

Fine-Grain Multithreading

Conventional Superscalar

Goals

We had three primary goals for this architecture:

1. Minimize the architectural impact on conventional superscalar design.
3. Achieve significant throughput gains with many threads.

A Conventional Superscalar Architecture

- Fetch up to 8 instructions per cycle
- Out-of-order, speculative execution
- Issue 3 floating point, 6 integer instructions per cycle

- Fetch Unit
- PC
- Instruction Cache
- Decode
- Register Renaming
- Integer instruction queue
- Integer units
- Integer reg's
- Data Cache
- Floating point instruction queue
- fp units
- fp reg's
- Int/id-store units
An SMT Architecture

- Fetch up to 8 instructions per cycle
- Out-of-order, speculative execution
- Issue 3 floating point, 6 integer instructions per cycle

Instruction Cache

- Fetch unit
- PC

Instruction Queue

- Floating point instruction queue (fp units)
- Integer instruction queue (int units)

Register Renaming

Decode

Data Cache

- Fp reg's
- Int reg's
- Int/Id-store units

Bottlenecks of the Baseline Architecture

- Instruction queue full conditions (12-21% of cycles)
  - Lack of parallelism in the queue.
- Fetch throughput (4.2 instructions per cycle when queue not full)

Performance of the Naïve Design

Improving Fetch Throughput

- The fetch unit in an SMT architecture has two distinct advantages over a conventional architecture.
  - Can fetch from multiple threads at once.
  - Can choose which threads to fetch.
Improved Fetch Performance

- Fetching from 2 threads/cycle achieved most of the performance from multiple-thread fetch.
- Fetching from the thread(s) which have the fewest unissued instructions in-flight significantly increases parallelism and throughput.

This SMT Architecture, then:

- Borrows heavily from conventional superscalar design.
- Minimizes the impact on single-thread performance
- Achieves significant throughput gains over the superscalar (2.5X, up to 5.4 IPC).

Improved Performance

![Graph showing improved performance](image)

Multithreading Models

- Coarse-grain – switch contexts (typically several cycles) on long-latency event.
  - MIT Alewife
- Fine-grain – switch contexts every cycle.
  - HEP, Tera
- Simultaneous Multithreading
  - Compaq 21464, Intel Pentium 4, Power 5
- Pros, cons, issues, …?
A New Multithreading Model

- Combines SMT and coarse-grain multithreading.

HEP Multithreading

Tera Multithreading

Cool Tera Features

- Full/empty bits on memory
- Randomized memory (why??)
- No bypassing
- Explicit-dependence lookahead
- LIW
- No caches
- High-bandwidth network

- Which of these are related to multithreading?
Summary

- Why simultaneous multithreading?
- Long-term solution?
- When won’t it work?
- What next?