An Evaluation of the Multiscalar Paradigm

Scott E. Breach

Computer Sciences Department
University of Wisconsin-Madison

Scott E. Breach	breach@cs.wisc.edu
Andreas I. Moshovos	moshovos@cs.wisc.edu
T.N. Vijaykumar	vijay@cs.wisc.edu
Gurindar S. Sohi	sohi@cs.wisc.edu

Sequential Programs + ILP

- Superscalar - State of the Art
  - ✔ Centralized Scheduling
  - ✔ 4-way Issue
  - ✔ Out-Of-Order
  - ✔ Range of Instruction Window <100
- Multiscalar - Novel Alternative
  - ✔ Decentralized Scheduling
  - ✔ Scaleable Issue
  - ✔ Out-Of-Order
  - ✔ Range of Instruction Window >>100

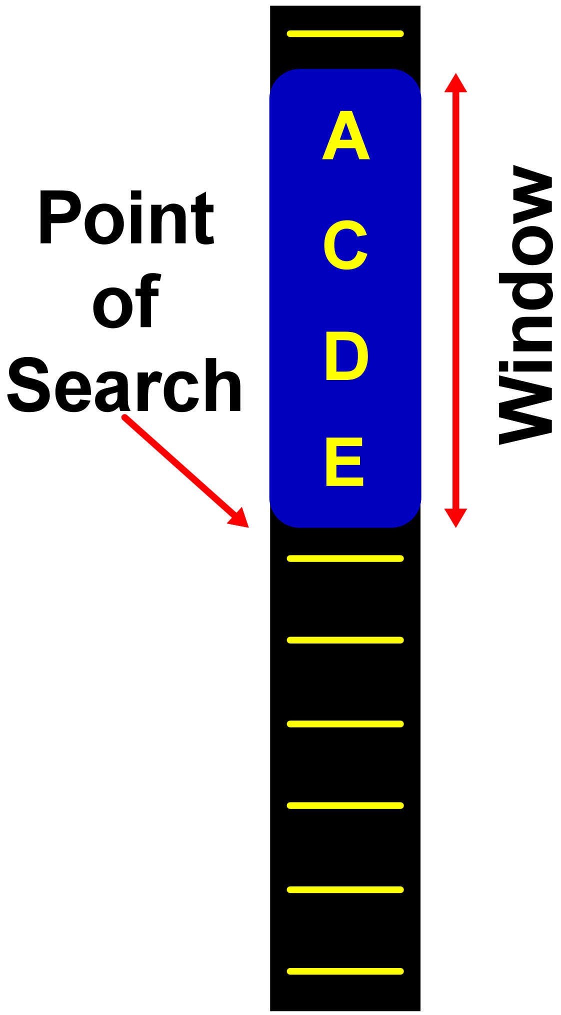
Promising Future Paradigm

- Comparing...
  - ☞Superscalar 16-way
  - ☞Multiscalar 8-unit 2-way
- SPEC95 INT
  - ✚25-50% Improvement
- SPEC95 FP
  - ✚50-200% Improvement

Talk Outline

- Building a Window for ILP
- Methodology/Configuration
- Performance Comparison
  - ☞ Superscalar/Multiscalar
  - ☞ SPEC95 INT/FP
  - ☞ Insight/Analysis
- Future Directions

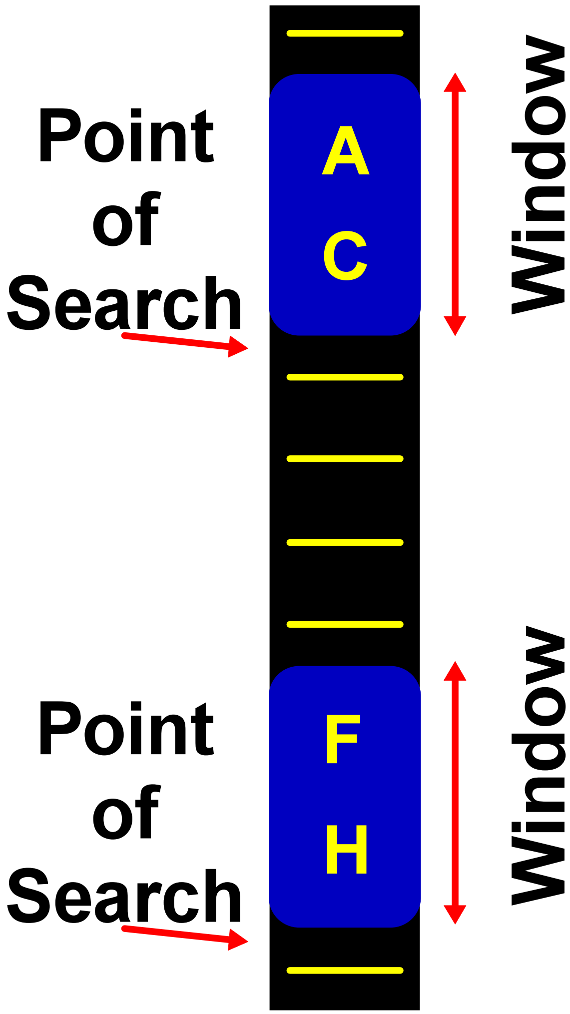
Building Superscalar Window



- Single Window
  - ✔ Large # Instructions
  - ✔ Concentrated Wide Issue
- Centralized Scheduling

Harder to Clock Fast

Building Multiscalar Window



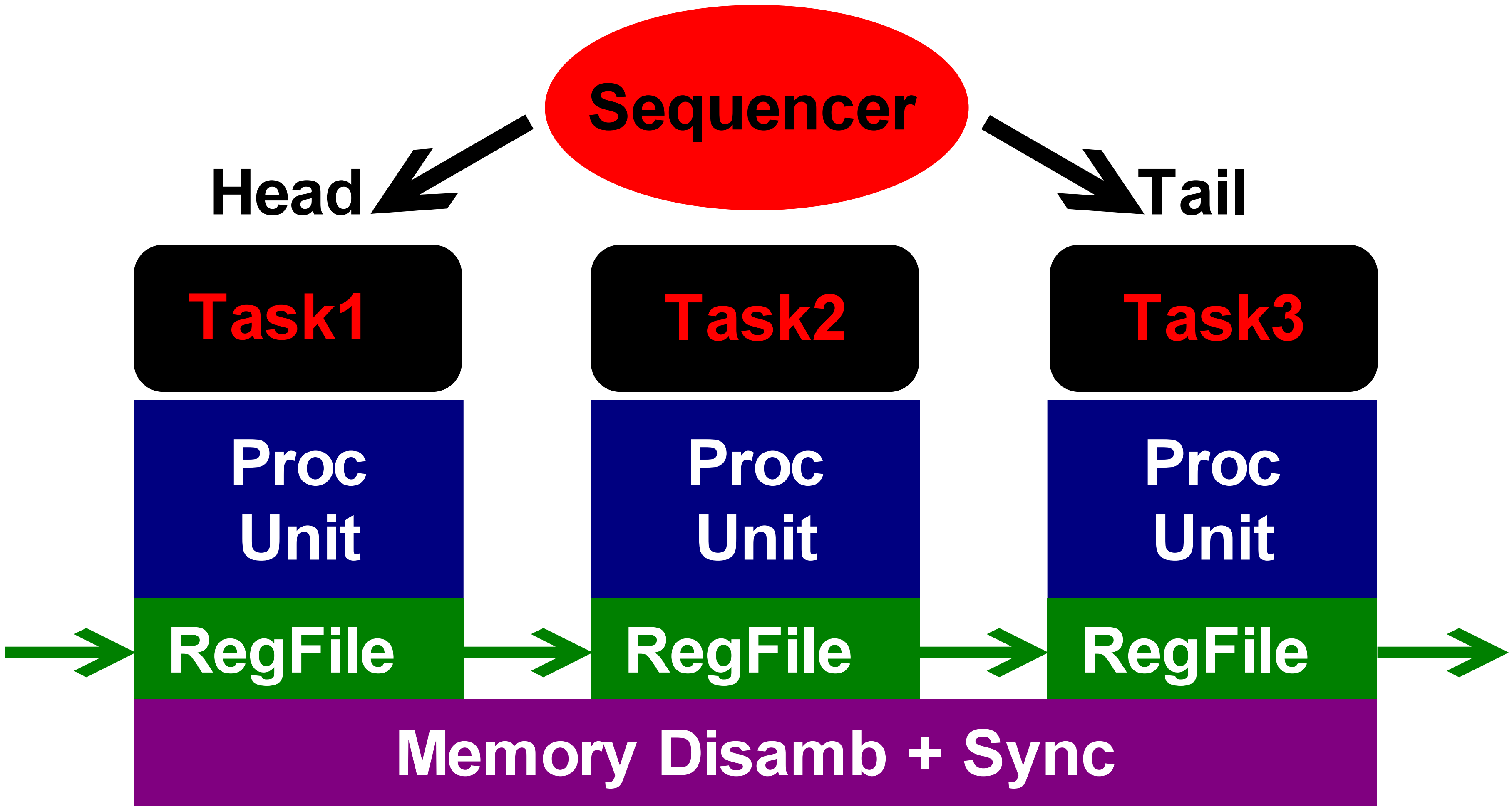
- Multiple Windows (or Tasks)
  - ✔ Small # Instructions
  - ✔ Distributed Narrow Issue
- Decentralized Scheduling

Easier to Clock Fast

Engineering Window

- Superscalar Window Challenge
  - ✘ Clock Speed
  - ✘ Design Time
  - ✘ Validation
- Multiscalar Window Advantage
  - ☞Think Large, Build Small
    - ✔Clock Speed
  - ☞Replicate/Reuse
    - ✔Design Time
    - ✔Validation

Multiscalar Big Picture



Talk Outline

- ~~Building a Window for ILP~~
- Methodology/Configuration
- Performance Comparison
  - Superscalar/Multiscalar
  - SPEC95 INT/FP
  - Insight/Analysis
- Future Directions

Methodology

- Compiler
  - ✔ Modified GCC 2.7.2
  - ✔ Highest Level of Optimization

- Hardware Simulator
  - ✔ Instruction-Driven
  - ✔ Cycle-Level

Configuration - Scheduling

- Superscalar
  - ✔ 128 In-Flight Instructions
  - ✔ Out-Of-Order, 16-way Issue
- Multiscalar
  - ✔ 16 In-Flight Instructions Per Unit
  - ✔ Out-Of-Order, 8-unit 2-way Issue

Configuration - Func Units

- Functional Units
  - ✔ Type+Number
    - ◆ 16 Add/Sub/Logic/Addr
    - ◆ 8 Mult/Div
    - ◆ 8 Load/Store
    - ◆ 8 Float
    - ◆ 8 Branch
  - ✔ Resources
    - ◆ Superscalar - Concentrated
    - ◆ Multiscalar - Evenly Distributed

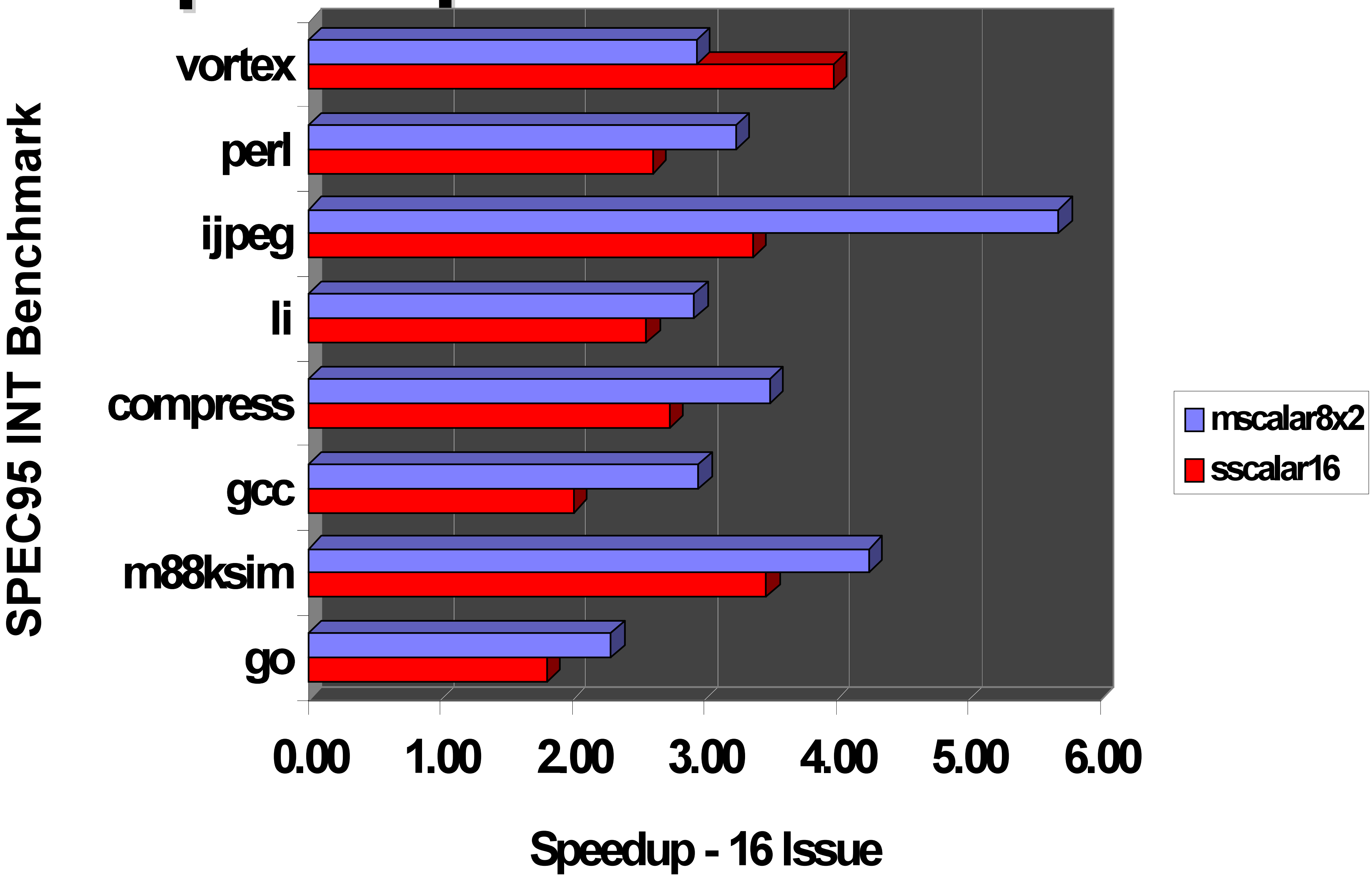
Configuration - Memory

- Inst Memory
  - ✔32K 1st Level, 1 Cycle, Non-Blocking
  - ✔Infinite 2nd Level, 12 Cycle
  - ✔16/32 Words Per Cycle
  - ✔Upto 16 Outstanding Misses
- Data Memory
  - ✔32K 1st Level, 2 Cycle, Non-Blocking
  - ✔Infinite 2nd Level, 12 Cycle
  - ✔8 Words Per Cycle
  - ✔Upto 64 Outstanding Misses

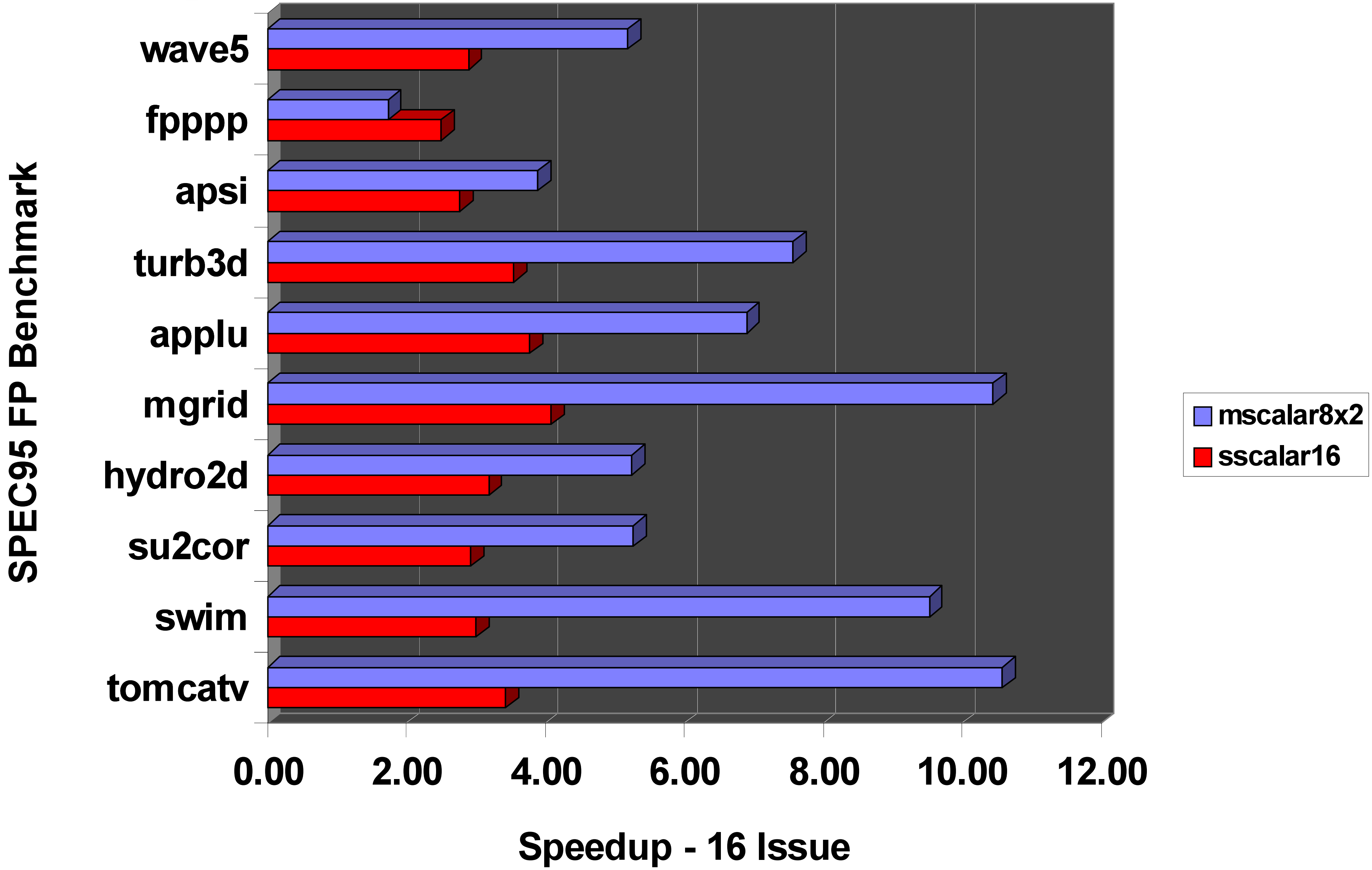
Talk Outline

- ~~Building a Window for ILP~~
- ~~Methodology/Configuration~~
- Performance Comparison
  - ☞ Superscalar/Multiscalar
  - ☞ SPEC95 INT/FP
  - ☞ Insight/Analysis
- Future Directions

Speedup - SPEC95 INT



Speedup - SPEC95 FP



Performance Summary

- SPEC95 INT
  - ✚ 25-50% Improvement
  - ☞ Clock Advantage Seems Quite Important
- SPEC95 FP
  - ✚ 50-200% Improvement
  - ☞ Clock Advantage Seems Less Important

Nature of Experiment

- ✘ Superscalar: Idealistic 16-Issue Design
- ✔ Multiscalar: Realistic 16-Issue Design

Compiler Window Factors

- If Tasks Too Small
  - Register Dependences
    - Wait Overhead
    - Aggravate Critical Paths
- If Tasks Too Big
  - Memory Dependences
    - Squash Overhead
    - Buffer Overflow

Hardware Window Factors

- Communication Delay
- Load Imbalance
- Pipeline Fill/Drain
- Misspeculation Penalty

Bottom Line on Window...

- If Same as Superscalar - SPEC95 INT
  - ☞Difficult to Sustain Same Raw IPC
  - ☞With Clock Advantage Better Speedup

- If Better than Superscalar - SPEC95 FP
  - ☞Possible to Sustain Better Raw IPC
  - ☞With Clock Advantage Even Better Speedup

Talk Outline

- ~~Building a Window for ILP~~
- ~~Methodology/Configuration~~
- ~~Performance Comparison~~
  - ~~Superscalar/Multiscalar~~
  - ~~SPEC95 INT/FP~~
  - ~~Insight/Analysis~~
- Future Directions

Performance Issues

- Compiler Uses Heuristics
  - Avg Task Size =16.0 Inst SPEC95 INT
  - Avg Task Size = 68.9 Inst SPEC95 FP
- Hardware Alloc/Dealloc Policy
  - Strictly Sequential
  - Load Imbalance = As Much As 25%
- Highly Dependent Code

Future Directions

- Compiler - Augment Heuristics with Profiling
  - More Flexible Tasks
  - More Aggressive Scheduling
- Hardware - Remove Strict Alloc/Dealloc Policy
  - Decouple
    - Spec/Arch State Update
    - Resource Alloc/Dealloc
- Hardware - Ease Effect of Dependences
  - Data Value Speculation

Results Appear Promising...

Multiscalar Information

http://www.cs.wisc.edu/~mscalar/

Scott E. Breach	breach@cs.wisc.edu
Andreas I. Moshovos	moshovos@cs.wisc.edu
T.N. Vijaykumar	vijay@cs.wisc.edu
Gurindar S. Sohi	sohi@cs.wisc.edu