# RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors

Vijay S. Pai
Parthasarathy Ranganathan
*Sarita V. Adve*

Department of Electrical and Computer Engineering
Rice University
Houston, TX

# Outline

- Overview of RSIM
- Motivation for RSIM
- Architectures modeled
- Interface
- Statistics
- Internals
- Experience
- Summary and Future Work

# Overview of RSIM

- RSIM: **R**ice **S**imulator for **I**LP **M**ultiprocessors
- For uniprocessor and multiprocessor architecture education and research
- Detailed execution-driven simulator
  - State-of-the-art processor pipeline
  - Aggressive memory system
  - Shared-memory multiprocessor
- NOT trace-driven unlike many uniprocessor simulators
- NOT direct-execution based unlike many multiprocessor simulators

# Motivation for RSIM (1 of 3)

- Most multiprocessor studies use *direct-execution* simulation
  - Application runs directly on host
  - Calls simulator on memory instructions
  - Models simple processors with blocking reads
- Direct execution fast and accurate for *old* sytems
- BUT current systems use processors with aggressive ILP
  - Out-of-order, speculation, non-blocking reads
- Approximations used with direct-execution
  - Simple processors $\approx$ ILP (?)
  - Simple + faster CPU $\approx$ ILP (?)
  - Simple + faster CPU + faster L1 cache $\approx$ ILP (?)
- *Are simple approximations to ILP valid?*

# Motivation for RSIM (2 of 3)

- Effectiveness of simple simulation models in HPCA'97
- Results for two models next

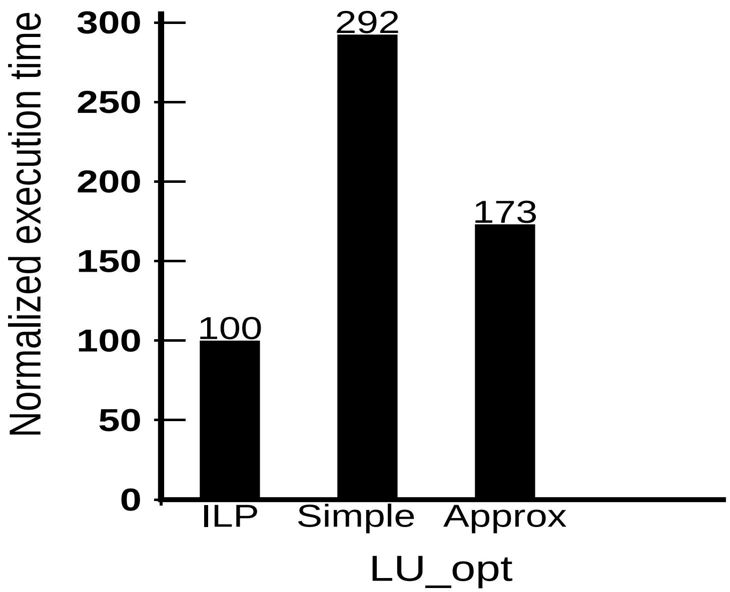
  *Simple*

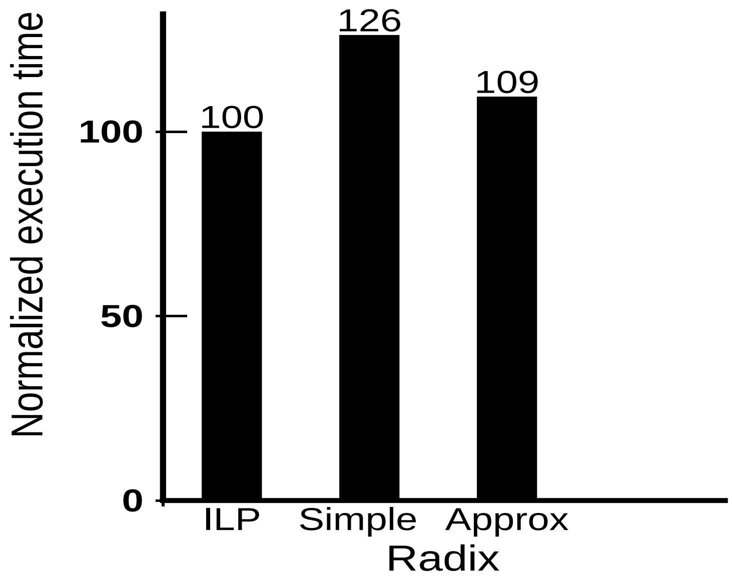
  - Most common simulation model
  - Same clock speed as ILP system

  *Best approximate*

  - Simple processor
  - Clock speed = 4xILP clock speed
  - L1 cache hits also sped up
  - Rest of memory system unchanged

# Motivation for RSIM (3 of 3)




Most common model: 26% to 192% error

Best model: -8% to 73% error

# Outline

- Overview of RSIM
- Motivation for RSIM

⇒ Architectures modeled
  - Processor Microarchitecture
  - Cache and Memory Sub-System
  - Multiprocessor System
- Interface
- Statistics
- Internals
- Experience
- Summary and Future Work

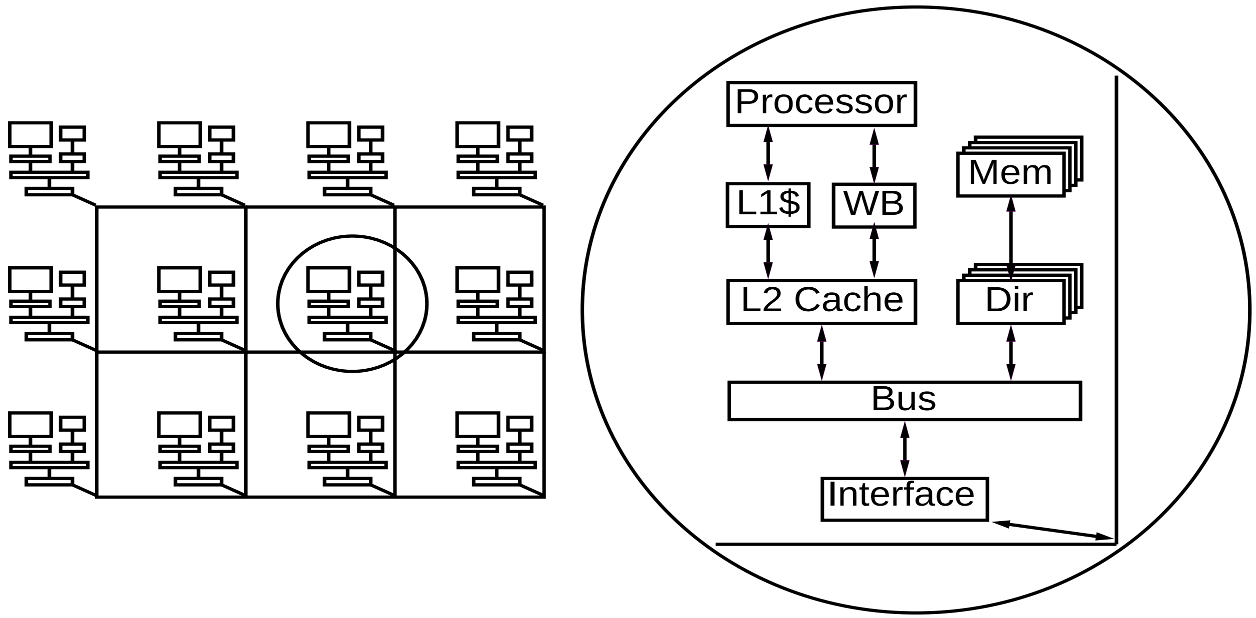
# Processor Microarchitecture

- Combines features from many current processors
  - Closest to MIPS R10000
- Key Features
  - Multiple instruction issue (superscalar)
  - Out-of-order scheduling (option for in-order)
  - Register renaming
  - Dynamic branch speculation
  - Limited predicated instructions
  - Non-blocking loads and stores
  - Software-controlled prefetching
  - Multiple implementations of various consistency models
- Most parameteres user-configurable

## Cache and Memory Sub-system

- Two-level data cache hierarchy
  - L1 cache multiported, writethrough or writeback
  - L2 cache pipelined and writeback
  - Coalescing write buffer between writethrough L1 and L2
  - Lockup-free caches, outstanding requests in MSHRs
- Main memory interleaved
- Memory accessed through pipelined split-transaction bus
- All contention modeled
- Instruction cache not modeled (100% instruction hit rate)
- No virtual memory
- Most parameters user-configurable

# Multiprocessor System (1 of 2)




- Hardware cache-coherent NUMA shared-memory system
- Invalidation-based directory protocol
- 2D mesh, wormhole routed network
- Network interface on memory bus

# Multiprocessor System (2 of 2)

- Multiple memory consistency models supported
  - Sequential consistency
  - Processor consistency
  - Release consistency
- Multiple implementations of each consistency model
  - Straightforward
  - Hardware prefetching from instruction window
  - Speculative load execution

# Outline

- Overview of RSIM
- Motivation for RSIM
- Architectures modeled

⇒ Interface

- Statistics
- Internals
- Experience
- Summary and Future Work

# RSIM Interface

- RSIM instruction set resembles SPARC V9
  - Currently no delay slots

    Use SPARC V9 gcc modified to remove delay slots
  - No single precision floating point
  - No 64 bit integer operations
  - Rudimentary OS features to run applications
- Compiled SPARC executables predecoded before running through RSIM
- Predecoders to convert other RISC ISA's to RSIM format should be straightforward

# Statistics

- Large number of statistics
  - Overall execution statistics
  - Processor statistics
  - Memory system and network statistics
- Most metrics include
  - Average value
  - Standard deviation
  - Distribution histograms
- Graphical diplays using splot

## Overall Execution Statistics

- Total execution time
- Instructions per cycle (IPC)
- Execution time divided into various categories
  - Busy time
  - ALU stalls and FPU stalls
  - Data reads and data writes

    L1 hit, L2 hit, Local memory hit, Remote memory hit
  - Exceptions
  - Branches
  - Synchronization
  - Other user-defined classes

# Processor Statistics

- Usage of various functional units
- Branch prediction behavior
- Occupancy of active list
- Performance of instruction fetching

# Memory System and Network Statistics

- Hit and miss rate
  - Cold, conflict/capacity, coherence misses
- Average latency of various classes of memory operations
- MSHR occupancy
- Prefetch effectiveness
- Network traffic, usage of various buffers

# RSIM Internals (1 of 2)

- Faithfully models
  - Processor pipelines
  - Cache and memory system
  - Network
  - Contention at all resources
- Coded in C and C++
- Simulation methodology
  - Cycle-by-cycle simulation of processor and cache
  - Event-driven simulation of memory and network

    Memory system and network code from RPPT

    Event-driven simulation engine uses YACSIM

# RSIM Internals (2 of 2)

- Platforms supported
  - SUN UltraSPARC, SparcStations, SGI Power Challenge
  - Expect easy portability across many UNIX platforms

    Only needs System V getcontext/setcontext

- Cost of detail is increased simulation time
  - Uniprocessor simulations

    10000 – 15000 instructions/second on Ultra 1/140

  - Mulitprocessor simulations

    Speed depends on communication in application

    7x to 10x slower than our direct-execution simulator

# Experience – Research

- Evaluation of ILP-motivated implementations of memory consistency models [ASPLOS'96]
- Overall impact of ILP on multiprocessor performance and simulation methodology [HPCA'97]
- Interaction of prefetching with ILP in multiprocessors [ISCA'97]
- New techniques to improve the performance of sequential consistency [In Progress]
- Producer-initiated communication with ILP processors [In Progress]

# Experience – Education

- Senior-level and first-year graduate architecture courses
- Assignments
  - Identify performance bottlenecks in ILP uniprocessors
  - Impact of various processor configuration parameters
- Semester-long projects

  Examples from senior class

  - Comparison of in-order and out-of-order scheduling
  - Various branch prediction strategies
  - Tradeoffs between performance and die area
  - Benefits of high bandwidth DRAM modules
  - Stream buffers with out-of-order processors

# RSIM Summary

- Detailed execution-driven simulator for state-of-the-art uniprocessors and shared-memory multiprocessors
  - NOT trace-driven
  - NOT direct-execution based
- Provides needed realistic experimental environment for architecture education and research

# Future Work

- Delay slots (will allow general compiler)
- Alternative networks
- Alternative coherence protocols
- More aggressive branch prediction
- Instruction cache support
- Performance visualization support
- Simulator performance improvements
  - Parallelization
  - Sampling
  - Selective analytical modeling
- Public distribution – end of Spring '97