# Full-System Timing-First Simulation

Carl J. Mauer

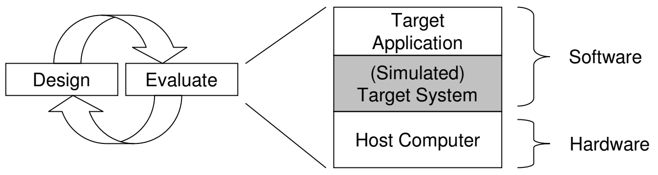
Mark D. Hill and David A. Wood

Computer Sciences Department

University of Wisconsin—Madison

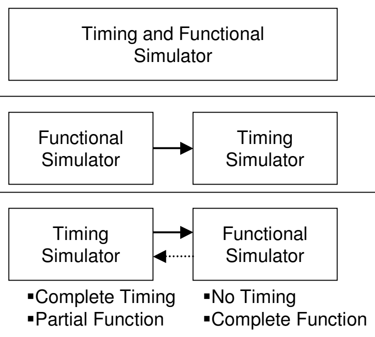
## The Problem

- Design of future computer systems uses simulation



- Increasing functionality and timing demands from:
  – Micro-architecture complexity
  – Multiprocessing
  – Complex workloads
- How can simulators be structured to manage complexity?

## Solutions



Integrated (SimOS)
- Complex

Functional-First (Trace-driven)
- Timing feedback

Timing-First (This paper)
+ Models multiprocessor timing
+ Decouples complexity
+ Leverages existing simulators

- Separates the complexity of functional correctness from timing accuracy

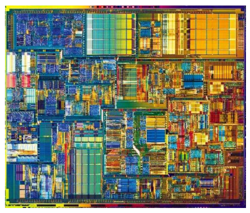
* Timing-directed discussed in paper

## Outline

- Overview
- Introduction
- Timing-First Simulation
- Evaluation
- Conclusion

## Challenges to Timing Simulation

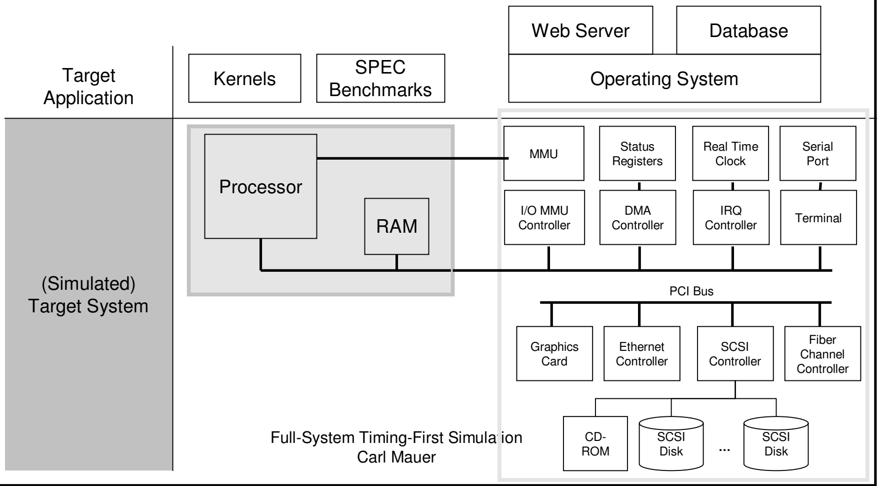
- Execution driven simulation is getting harder
- Increasing micro-architecture complexity
  - Multiple “in-flight” instructions
  - Speculative execution
  - Out-of-order execution
- Increasing use of multiprocessing
  - Chip multiprocessors
  - Hardware multi-threading



Full-System Timing-First Simulation
Carl Mauer

## Challenges to Functional Simulation

- Commercial workloads have high functional fidelity demands



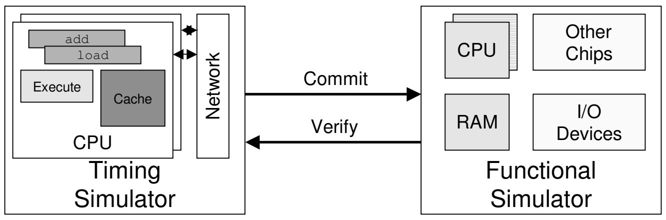
Full-System Timing-First Simulation
Carl Mauer

## Outline

- Overview
- Introduction
- Timing-First Simulation
- Evaluation
- Conclusion

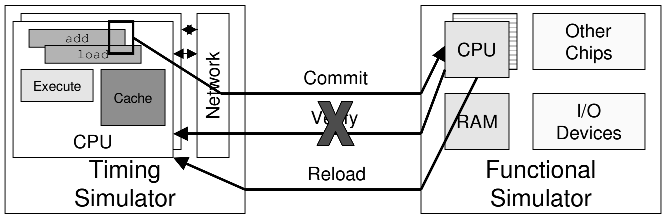
## Timing-First Simulation

- Timing Simulator
  – does functional execution of user and privileged operations
  – does detailed multiprocessor timing simulation
  – does NOT implement full instruction set or any devices
- Functional Simulator
  – does full-system multiprocessor simulation
  – does NOT model detailed micro-architectural timing



## Timing-First Operation

- As instruction retires, step CPU in functional simulator
- Verify instruction’s execution
- Reload state if timing simulator *deviates* from functional
  - Very uncommon case
  - Example: Instructions with unidentified side-effects



## Benefits of Timing-First

- Support detailed multi-processor timing models
- Avoid re-implementing full-system simulator (hard!)
- Faster development, same performance
  - 99% correct is (much!) easier than 100%
  - Functional simulation is (much!) faster than timing simulation
  - Independent correctness checker helps development

- Questions:
  - How much error is introduced with this approach?
  - Are there simulation performance penalties?

## Outline

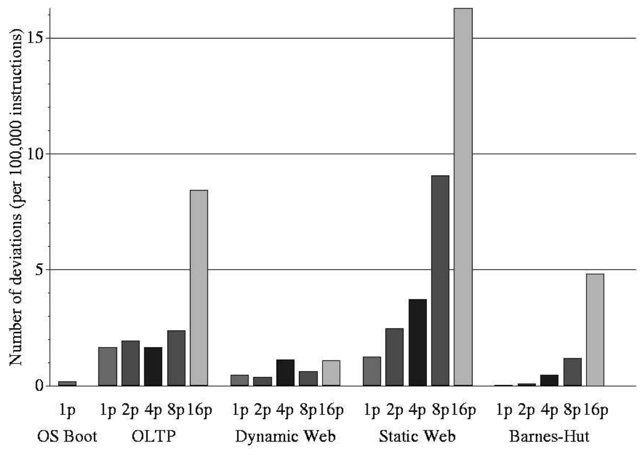
- Overview
- Introduction
- Timing-First Simulation
- Evaluation
- Conclusion

## Evaluation

- Our implementation, TFsim uses:
  - Functional Simulator: Virtutech Simics
  - Timing simulator: Implemented in less than one-person year
- Evaluated using OS intensive commercial workloads
  - OS Boot: > 1 billion instructions of Solaris 8 startup
  - OLTP: Database workload with 1 GB of data
  - Dynamic Web: Apache serving message board, using code and data similar to `slashdot.org`
  - Static Web: Apache web server serving static web pages
  - Barnes-Hut: Scientific SPLASH-2 benchmark

## Measured Deviations

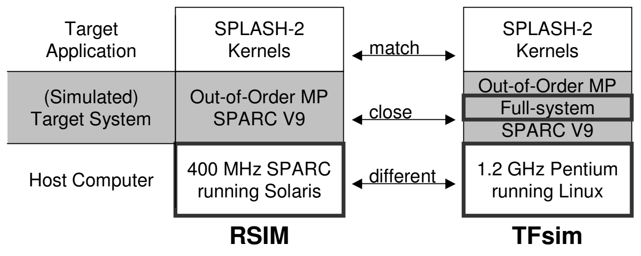
- Less than 2 deviations per 10,000 instructions (0.02%)



## Analysis of Results

- Runs full-system workloads!
- Timing performance impact of deviations
  - Worst case: less than 3% performance error
  - Average case: less than 0.3% performance error
- 'Overhead' of redundant execution
  - 18% on average for uniprocessors
  - 18% (2 processors) up to 36% (16 processors)

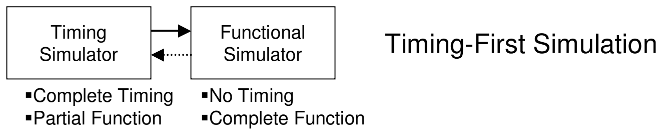
## Performance Comparison



- Absolute simulation performance comparison
  - In kilo-instructions committed per second (KIPS)
  - RSIM Scaled: 107 KIPS
  - Uniprocessor TFsim: 119 KIPS

## Conclusions

- Execution-driven simulators are increasingly complex
- How to manage complexity?
- Our answer:



  - Introduces relatively little performance error (worst case: 3%)
  - Has low-overhead (18% uniprocessor average)
  - Rapid development time

## Questions?

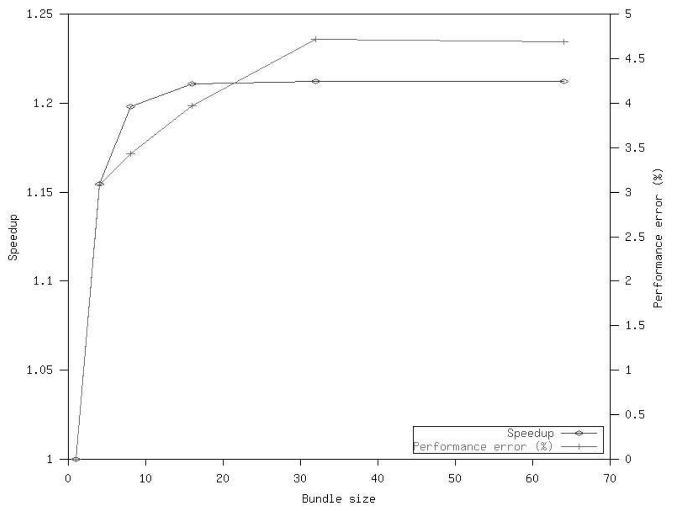
Full-System Timing-First Simulation
Carl Mauer

## Related Work

| Name | Dynamic Full System | Out-of-Order | Multi-processor | Decoupled |
|---|---|---|---|---|
| Multiscalar Simulator [6] | | Yes | | Yes |
| FastSim [24] | | Yes | | Yes |
| SimpleScalar 3.0 [2] | | Yes | | Yes |
| MASE [18] | | Yes | | Yes |
| RSIM [15] | | Yes | Yes | |
| SimpleMP [22] | | Yes | Yes | Yes |
| ASIM [14] | | Yes | Yes | Yes |
| g88 [4] | Yes | | | |
| gsim [19] | Yes | | Yes | |
| Talisman [5] | Yes | | Yes | |
| Simics [20] | Yes | | Yes | |
| PharmSim [7] | Yes | Yes | Yes | |
| SimOS [23] | Yes | Yes | Yes | |
| TFsim [This paper] | Yes | Yes | Yes | Yes |

Full-System Timing-First Simulation
Carl Mauer

## Bundled Retires



Full-System Timing-First Simulation
Carl Mauer