FeS₂: A Full-system Execution-driven Simulator for x86 Naveen Neelakantam[†], Colin Blundell[‡], Joe Devietti[§], Milo M. K. Martin[‡] and Craig Zilles[†]

[†]University of Illinois at Urbana-Champaign [‡]University of Pennsylvania [§]University of Washington

Motivation

Need for an x86 simulator

W

- Many tools only exist on or are more mature on x86 · e.g. Pin, Harmony JVM
- Many interesting apps are only available as x86 binaries · e.g. Word, Photoshop

Need for full-system simulation

- Modern applications are multi-threaded
- · Communicate via O/S mechanisms such as signals
- · O/S thread scheduling can affect program behavior
- Modeling O/S behavior requires simulating supervisor code i.e. full-system

Need for accurate timing model

 Architectural features have first-order performance effect · e.g. caches, branch prediction, superscalar, out-of-order

Simulation Methodology

Timing-first simulation[1]

- · Separation of *functional* and *timing* concerns
- Timing simulator (FeS₂) calculates execution times
 - Includes a *mostly* correct functional model
 - · Runs ahead of functional simulator
- · Functional simulator detects functional deviations · Trails timing simulator and verifies architected state

Leverage Simics[2] for functional model

 Production guality x86 full-system simulator · Provides golden standard functional model

How it works

- FeS₂ verifies its state when it commits an x86 op · Compares with Simics state
- · If mismatch, reload state from Simics



Modular design (C++ object-oriented code) · Separate classes for branch predictor, scheduler, etc

Leverages PTLsim[3] for µop decode and execute

• x86 op \Rightarrow uop decoder from PTLsim · uop functional model also from PTLsim

Open Source (GPL)

Configurable out-of-order timing model

- Superscalar decode (default 3 x86 ops)
- Superscalar rename/execute/commit (default 4 μops)
- D-cache (currently 32KB L1, 512KB L2)
- Branch predictors (currently 8Kb gshare, 32-entry RAS)





Simulating an Application

Leverage Simics scripting capabilities

- Simics script can trigger at various events
- Magic instruction (special x86 no-op)
 - Breakpoints

yes

🕈 no

done

· Script transitions into and out of timing-first mode

Simulate entire application or samples

- · Sampling preferable in general
- · Timing simulation is relatively slow

Releases

Complete uniprocessor model in initial release Available at http://fes2.cs.uiuc.edu

Multiprocessor model is already finished

- · Leverages the GEMS' Ruby[4] memory timing model
- To be included in a future release

start timing event (magic instruction, breakpoint)



(magic instruction, breakpoint, icount)

W

Time

References

- 1. Carl J. Mauer et al. Full system timing-first simulation. In SIGMETRICS '02. pp. 108-116. June 2002
- 2. Virtutech Simics: http://www.simics.com
- 3. Matt Yourst. PTLsim: A cycle accurate full system x86-64 microarchitectural simulator. In ISPASS '07. April 2007
- 4. Milo M.K. Marin et al. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. Computer Architecture News. September 2005



no

reload state