Why Tools Matter

Todd Austin
University of Michigan

ISPASS Conference, April 2008

A long time ago, in a place far away…
My Experience: SimpleScalar Tool Set

- Computer system design and analysis infrastructure
  - Uniprocessor models
  - User-mode input/output
  - Supports many ISAs
  - Portable to host platforms
  - Deployed widely in academia and industry for more than a decade

- Available with source and docs from www.SimpleScalar.com

Obligatory Shot of Simulator Running DOOM
Reasons Against Building & Releasing Tools

• Don’t want to loose your “competitive edge”

• Often, tools development is not viewed as “research”

• User support and code maintenance is very demanding, couldn’t your time be spent better?

• Broken tools could be harmful to the state-of-the-art

• Vast majority of user feedback is negative

Goals of This Talk

Encourage aspiring tools hackers to build and release their tools.

Give you my best advice for undertaking such a challenge.

Suggest some promising (simulator) tool directions to pursue.
Why Tools Matter -- to the Community

The altruistic viewpoint...

- Accelerate the pace of research
  - Good tools enable researchers

- Accelerate the pace of tool development
  - Through enabling infrastructure development
  - Through the growth of a community of developers

Why Tools Matter -- to Me

The “narcissistic” viewpoint...

- Amplify your contribution to the research community
  - Take credit for other people’s research!

- Gain public recognition

Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
All Rights Reserved. This version of SimpleScalar is licensed for academic
non-commercial use. No portion of this work may be used by any commercial
entity, or for any commercial purpose, without the prior written permission
of SimpleScalar, LLC (info@simplescalar.com).*
Advice for Building Successful Tools

Based on my experiences with SimpleScalar.

Plus, observations from other tool developers.

But don’t forget… you’ll need a lot of luck, too!

And now… six pieces of advice.

Advice: Let Research Drive the Outcome

• Tools serve research demands
  – Let your research agenda define these demands

• Great infrastructure enables research
  – Build the infrastructure and you will get there first!
  – E.g., SimpleScalar was first to recognize the positive benefits of mis-speculation [PAID’95]

• It will sharpen the focus and usability of the infrastructure
Advice: Be Innovative

- Don’t try to only improve engineering, re-imagine the design
- You will stay interested in the project longer
- It will become easier to publish your work

Innovation: Execution-Driven Simulation

- Trace-based simulation
  - Simulator reads instruction “trace”
  - Simple to implement, no functional component needed
- Execution-driven simulation
  - Simulator “runs” the program, generating complete execution
  - More difficult to implement, requires instruction function and I/O handling
  - Approach has many advantages, e.g., access to values, misspeculation
Innovation: External I/O Tracing (EIO)

• Addresses difficulties of creating reproducible experiments
  – 1000x compression over industry-standard branch tracing
  – Re-creates of all program data values (unlike branch tracing)
• External I/O events traced
  – System calls, external interrupts
  – All events time-stamped with processor cycle count

Innovation: GPV – Graphical Pipeline Viewer
Innovation: MASE Microarchitecture Simulator

- Performance simulation model more detailed than sim-outorder
- Many simulation advances
  - x86 target and uArch support
  - Improves validation support
  - Oracle for “perfect” studies
  - Speculative state management facilities simplify speculation
  - Callback interface permits sophisticated memory simulation

Advice: Build a Platform

- Build a tool for building tools
  - “Separate mechanism and policy”
- Assume users will utilize components a-la-carte
- Strive to make the researchers code only glue that binds components
Warning: Flexibility vs. Barrier-to-Entry

• A single file describes all aspects of the architecture
  – Used to generate decoders, dependency analyzers, functional components, disassemblers, appendices, etc.
  – Machine definition + 10 line main == functional simulator

• Optimizing performance has similar effects

Advice: Make Usability a Priority

• Make it work out of the box
  – Include everything the user needs
  – Lots of pre-built simulators
  – Cross-compilers
  – Workload
  – Documentation and tutorials

• Open source is a must-have!

• Build portable code

• Be your first and best customer
Usability: SimpleScalar’s Abstraction “Knob”

- **Sim-Fast**
  - 420 lines
  - no timing
  - 8+ MIPS

- **Sim-Safe**
  - 350 lines
  - no timing
  - w/ checks

- **Sim-Profile**
  - 900 lines
  - no timing
  - lot of stats

- **Sim-Cache**
  - ~1000 lines
  - functional
  - cache stats

- **Sim-Bpred**
  - 3900 lines
  - performance
  - OoO issue
  - branch pred.
  - mis-speculation
  - ALUs
  - cache, TLB
  - 500+ KIPS

- **Sim-Outorder**
  - 900 lines
  - no timing
  - w/ checks

Performance

Detail
Advice: Work at Gaining Market Share

• Be a champion
  – Centralize decisions and integration
  – End result will be more coherent, focused

• Strive for quality, but avoid perfection
  – “Be prepared to throw one away.”
  – Build a regression test suite
  – Incorporate debugging features

• Listen to users
  – Understand how the current infrastructure limits their work
  – Track users with an easy-to-google name

SimpleScalar Timeline

SimpleScalar Citations (as per Google Scholar)
Advice: Don’t Ignore Business Considerations

- Use a license that meets your needs
  - Users often confuse “open” with “free”
  - See www.OpenSource.org for licenses

- Secure university rights early
  - Before the lawyers smell the money!

- Be aware of intellectual property issues
  - More difficult to accept contributions
  - Get trademarks, web addresses, and copyrights ASAP

Tomorrow’s Tools

- First, look at some of today’s successes:
  - SimpleScalar – uniprocessor uArch
  - Trimaran – VLIW architectures
  - Wattch – power modeling
  - HotSpot – thermal modeling
  - CACTI – cache modeling
  - GEMS, M5, RSIM – multiprocessor simulation
  - PIN – binary instrumentation

- Their recipe for success?

- Some research arenas that need more tools…
  - Reliability, Security, Application-Specific Processors, GPUs, Sensor Networking, Virtualization, Nano-technologies, Non-traditional computation models
Summary

• The “Why’s” far outweigh the “Why Not’s”

• Six tips for undertaking tool development and release
  – Let Research Drive the Outcome
  – Be Innovative
  – Build a Platform
  – Make Usability a Priority
  – Work at Gaining Market Share
  – Don’t Ignore Business Considerations

Acknowledgements

• Doug Burger, University of Texas, Austin
• Scott Breach, AMD
• Steve Bennett, Intel
• Eric Larson, Seattle University
• Chris Weaver, Intel
• Guri Sohi, University of Wisconsin
• Butler Lampson, Microsoft Research (quotes)