SimNow™: Fast Platform Simulation Purely In Software

Robert Bedichek

Introduction

AMD

- Simulators come in many flavors.
- Some simulators are micro-architectural simulators, and some are instruction-level simulators.
- Some simulators emulate only the CPU model, and others emulate the whole PC platform.
- Instruction-level simulators are usually written in one of three different ways
  - Conventional emulation
  - Threaded-code simulation
  - Dynamic translation

What is the SimNow™ Simulator?

- SimNow™ is a fast and configurable x86 and AMD64 dynamically-translating instruction-level platform simulator. With SimNow™, users connect complex software models to form a full PC platform emulation environment.
- We use SimNow™ to emulate AMD Athlon™ 64 and AMD Opteron™ multiprocessor systems that run commercial operating systems and applications. Specifically, AMD and its partners use SimNow™ for:
  - BIOS and Device Driver development
  - Generating state snapshots and event traces for input to detailed timing simulators, to support processor architecture development
  - Prototyping software-visible architecture changes
  - Non-intrusive and deterministic measurement and testing of software at the instruction-execution level
  - Modeling of future platform tradeoffs for correctness and performance analysis

Presentation Map

AMD

- Overview
- Comparison with Other Simulators
- How it Works
- Demonstration
- Requirements/Goals/Uses
- Status
- Conclusion

SimNow™ vs. Earlier Simulators

AMD

- SimNow™ is much faster than any other x86 simulator of which we are aware
  - Its speed comes from using dynamic translation and in not attempting to model fine timing detail
- SimNow™ models the entire PC platform.
  - SimNow™ models specific chipsets and functionality of those chipsets; enough to allow unmodified commercially-available BIOS's and OS's to boot and run correctly.
- SimNow™ is configurable, and can emulate about a dozen different AMD Athlon™64 and AMD Opteron™ platforms.
- SimNow™ is entirely a user application
  - SimNow™ does not require any special drivers to be installed on the host machine

Simulator Performance

AMD

- Approximate Slowdowns:
  - Rev 1.x (Circa 2002) of SimNow™: 1000:1
  - Typical threaded-code simulator: 100:1
  - Rev 2.1 (August, 2003) SimNow™: 50:1
  - Current SimNow™: 10:1
  - Typical Virtual Machine: 1.5:1
- Simulator performance is critical – it determines how many architectural tradeoffs we can make in a fixed number of months. A fast simulator enables the user to do tasks in simulation that were infeasible with a slow simulator.

Why Is The New SimNow™ So Much Faster Than Earlier Versions?

AMD

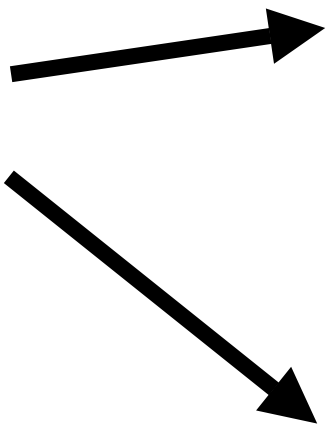
- Previous versions of SimNow™ CPU models emulated every instruction in C++ code.
- New versions of SimNow™ CPU models translate simulated x86 instructions to sequences of host x86 instructions
  - These are called “translations” and are cached and executed
  - Decoding phase is like conventional simulators
  - Code generation phase is new
- Higher performance because translations are executed many times, so decode/code-gen time is amortized
- Many features can be added without slowing down the simulator when these features are off
  - Conventional simulators slow down even when features are off, from all the testing that winds up in the main loop

Translation Caching Example

Cached Translation
In Host Memory

Guest x86 code

```
...
Add  eax,ebx
Mov (eax),ecx
...
```



Load host eax from memory home for guest eax value

Load host ebx from memory home for guest ebx value

Add eax,ebx

Load host ecx from memory home for guest ecx value

Compute host pointer from eax (I.e., simulate TLB)

Mov ecx to memory

Store eax in host memory home for guest eax value

SimNow™ Demo!

- We will boot an unmodified commercially available BIOS and Operating System

SimNow™ Internals -- Analyzers

AMD

- Key feature: user-written analyzers
  - Small sequences of stylized C code compiled to x86 byte stream
  - Dynamically linked to the simulator to gather statistics and generate exceptions and perhaps even modify instruction semantics
  - All tracing and breakpoints can be written by the user via analyzers
  - Analyzers can be added to various points in the simulator
    - At instruction decode time
    - At exception processing time
    - At instruction execution time

Device Models

- While performance of a full-system simulator is driven almost entirely by simulated processor performance, a full-system simulator requires complex device models in order to function.
- We spend more effort on the device models than the high performance processor model.
- SimNow™ models numerous devices, including a NIC, a RAID controller, various southbridges and northbridges
  - but this is a tiny fraction of the hardware devices in the market.
- SimNow™ also models the machine check architecture and power states of AMD processors.

General Simulator Requirements

AMD

- As high AMD64 performance as is practical
- Allow users to add I/O models and fragments of analysis code
- Deterministic execution
  - This is critical to its usefulness as a software measurement tool
- Rich scripting and debugger interfaces
- Ability to model MP's and complex I/O bridges
- Testability, in particular, we should be able to compare execution signatures with other simulators
- Ability to be tightly connected with a timing-accurate microsimulator

Non-goals

AMD

- Portability to platforms other than those based on AMD64 processors
  - Long mode simulation requires an AMD Opteron™ or Athlon™ 64 host running a 64-bit OS
- Portability to OS's other than Linux-64 and Windows-64
  - And we won't support all versions of Linux and Windows
- Timing accuracy (we use a microsimulator for that)
- Complete I/O models (takes too long, not necessary)
  - Some I/O models only model 10-20% of the whole I/O chip
  - We model what BIOS's, OS's and applications need, not everything in the specifications/RTL

Microsimulation On The Cheap

AMD

- We do microsampling, a combination of SimNow™ and a timing-accurate processor model (simx). Here's how it works:
  - SimNow™ runs the whole workload, i.e., the OS + benchmark
  - Simx runs for, say, 1m instructions every billion instructions of workload. So it executes a small fraction of the total stream.
  - Research results (see the "Smarts" paper in 2003 ISCA) show that with this technique one can get performance results that are within 1.5 percent of the result obtained when one runs the microsimulator on the entire workload.

Current Status

AMD

- SimNow™ as of August, 2004:
  - Boots and runs Linux-64, Linux-32, Windows® 2000, Windows® XP, Windows64, Solaris (32-bit version), and DOS
  - Runs unmodified Phoenix and Award BIOS's for AMD Athlon™ 64 and AMD Opteron™ platforms
  - Runs SpecJBB with simulated 1P, 2P, 4P, and 8P configurations
  - Runs 64-bit Linux builds of SPECint2000 and SPECfp2000
  - Runs SYSmark® and Winstone® benchmarks
  - Can generate trace files in several formats
  - Provides an interface to several commercial debuggers

Conclusion

- We have built a new kind of CPU model (new, that is, to the x86 space)
  - It is about 100 times faster than the model it replaced
  - Total system performance improvement varies considerably with workload
- Our goal is to produce the fastest, most flexible, most reliable, and most useful x86 simulator in the industry. We think we've achieved this.
- SimNow™ is available to AMD partners.

Backup

AMD

-

Slowdown – Key Simulator Metric

- B(X) means B runs on X
- Time(Y) means time to do Y
- Say we run a benchmark B on simulator S, which is running on host H, i.e., B(S(H))
- $T_{simulator}$ = Time(B(S(H)))
- T_{host} = Time(B(H))
- Slowdown is $T_{simulator} : T_{host}$

SimNow™ Internals -- Translations

AMD

- Register - register operations are just copied, 1:1
- Memory operations call into 12 to 14 instruction software TLB lookup sequence that translates the simulated virtual memory address to a pointer on the host machine
- Complex instructions translate to a call sequence – heavy lifting is done in C++ code, just like a conventional simulator
- Unit of translation is the basic block, or shorter
- Most translations are chained to their successor translations, reducing the overhead to dispatch in most cases
- Hand-coded assembly for the other cases, e.g., to find the successor of indirect jumps and returns

Developing Device Models

- SimNow™ device models are all written to a common interface, and compile to DLL's on Windows, or .so on Linux
- SimNow™ discovers these DLL's or .so's at runtime, and has no predetermined knowledge about device models
- Therefore, device models can be written by a user without knowledge of SimNow™ internals, as long as the common interface is documented and understood
- We have created a SimNow™ device model SDK. With this SDK, a user can create a SimNow™ device model without needing the source code for the rest of SimNow™
- We encourage users to develop device models as they need them.