



VIRTUALIZED SOFTWARE DEVELOPMENT

JUNE 2007

INFO@VIRTUTECH.COM

WWW.VIRTUTECH.COM



SIMICS

INTRODUCTION

There is a similar story embedded in any presentation to an electronic system engineering organization:

Presenter: Moore's Law!

Audience: Oh no, what shall we do?

Presenter: Not to worry. We have a new approach that will save you!

Audience: What a relief.

Moore's law, the exponential growth in the number of transistors per integrated circuit, has drastically shifted the system development process from a focus on hardware to an increase in software complexity. This change has required a re-tooling of the software development process to make testing and debugging much more efficient and economical. This white paper describes the approach to software development and testing using Virtutech's virtualized software development platform, Simics.

Moore's Law increases software implementation

Moore's Law dictates that more and more of a system is most economically implemented as software. Since this is an economic imperative, those who resist the shift will fall behind competitors who are able to deliver cheaper solutions. However, this also means that most of the cost and most of the schedule risk for system development depends on the software component.

Oh no, what shall we do for hardware?

One of the most significant problems with developing software for any sort of esoteric hardware (almost anything other than a PC or a workstation) is that the software has to be tested on something that approximates the production hardware on which the software will ship. This means developers must test on actual hardware if it is available, or at the very least, use some approximation of the final hardware. However, both approaches are highly insufficient because:

- systems are now increasingly complex, making it impossible to build a sufficiently accurate approximation before real hardware is available

- engineering productivity is too low when tracking down subtle problems on unwieldy hardware
- engineering is wasted solving problems that only exist on the temporary hardware

Not to worry, just simulate it!

The solution to delivering hardware-like development accuracy with better-than-hardware performance requires a new approach. Virtualized software development centers on the development and use of a software model of the final system hardware. This software model of the target system is so accurate that the unchanged binary code can be run on any hardware (e.g. a Dell desktop PC) at speeds sufficient enough for software developers and testers to use it for debugging their software. Since the execution runs on an off-the-shelf PC, this approach keeps getting better as the host microprocessors get faster—also observing Moore's Law! Virtualized software development becomes a more attractive solution as the system becomes more complex.

What a relief. Higher performance and lower cost

Virtutech is the leader in providing virtualized software development solutions to electronic system companies, from small enterprises to Fortune 100 corporations in networking, aerospace, defense, storage, computing, security and other application areas. The move to virtualized software development is driven by two underlying themes: full-system simulation costs significantly less than real hardware and virtualized software development performs significantly higher than real hardware.

THE INCREASING COMPLEXITY OF ELECTRONIC SYSTEMS

The increasing complexity of electronic systems is more suited to implementation in software rather than designing hardware especially for a specific purpose.

The modern cell-phone provides a good example. Initially, digital cell-phones were implemented with about half a dozen custom-designed integrated circuits. But as silicon got faster, there was no advantage to running the algorithms faster since the data rates for voice calls were fixed by the GSM, CDMA and other standards. The functionality moved into processors so that today a digital cell-phone (aside from the radio) consists primarily of a single

Why processors become more attractive

The reason that microprocessors become more attractive as an implementation approach over time is a subtle consequence of Moore's Law. As semiconductor processes continue their relentless advance, processors of all types get faster and cheaper. But then so does everything else implemented on silicon. So why does this favor processors?

Most functions implemented on silicon have an externally-determined performance. As the implementation increases in speed it is not possible for the performance of the function to increase as well since that is determined externally. Instead, the duty cycle of the silicon implementation decreases so that it is only used intermittently and thus, in some sense, wasted. This is not true for processor-based implementations of a function since multiple functions can be merged onto fewer and fewer processors. Only "as fast as possible" functionality can really take advantage of the increased performance of the silicon substrate in areas like graphics chips, top-end routers, video-game consoles and so on, and even then only for the performance-critical parts.

The result is that increasing semiconductor performance drives more and more of a system into software.

chip containing a digital signal processor (DSP) and a general purpose microprocessor, or perhaps even just a single processor with a DSP-rich instruction set. The bulk of the functionality of a cell-phone has moved into software.

However, there is an additional trend increasing the complexity of electronic systems. Where systems were once stand-alone, they now communicate with other systems. This makes developing and testing these systems significantly more complex. Checking that a PDA functions correctly is only the first step; checking that it functions correctly as part of a smart-phone, when real-time packets are arriving over the radio, requires analysis at a higher level. Most of the problems that arise with systems that have already shipped are due to unforeseen or rare interactions between the system and its environment, or internally between different parts of a complex multi-processor system.

The combination of these two trends towards increased software content and greater environmental interaction translates into increasingly complex software. As a result, the cost and risk of system development is becoming further concentrated in the software. Current approaches to software development are inadequate for developing this type of software load.

WHAT TO RUN ON?

Software developers frequently face the same problem. Their daily work consists of editing code, compiling it and then debugging it by running it on some approximation of the final hardware. If the code they are writing is destined to run on the same type of machine that is on their desktop, typically a PC, then they can simply run it on the PC and use a PC-based debugging environment. If the code they are writing is destined to run on something more unusual (e.g., a 3G base-station containing multiple PowerPC processors), they cannot test and debug as simply as they would for PC-based software. Therefore, software developers have a problem if their code is going into devices such as a cell-phone, set-top box, router, satellite, or an avionics module in a plane. Developers will run into an even larger problem if their code is going into any piece of hardware that hasn't yet finished being designed and built.

Traditionally, the two typical choices available to software developers are to run the code on their PCs—despite the complexity—or to use real hardware.

Running the target code on a PC involves building some sort of test scaffolding that provides enough of the behavior of the real hardware that the software has a chance of running. This requires a leap of faith – will software written on an Intel-based PC running on test scaffolding behave like the same code compiled with a different compiler, running on a MIPS processor inside a real router? The phase when a developer discovers whether this bet pays off is known as system integration. If the bet doesn't pay, the resulting re-work delays the shipment of the entire system.

The alternative approach is to use the real hardware by:

- buying real hardware if it exists
- using approximate real hardware if it exists—usually at least the processor and some peripherals are available, or
- making a mockup with field-programmable gate-arrays (FPGAs)

Both the leap-of-faith and the real hardware approaches fall short. Host-based development on a PC is too inaccurate to generate reliable code on schedule. The hardware approach is far too costly and unwieldy and sometimes infeasible, as the system might not yet exist or is prohibitively expensive to use for testing purposes.

Sometimes a third approach is attempted: using hardware-software co-verification tools. These are certainly accurate enough, but they are orders of magnitude too slow for accurate testing. If it takes a weekend to boot the operating system once, what do software developers do for their daily edit-compile-debug cycle? At best, co-simulation can be used for developing very low-level code such as a BIOS.

Simics from Virtutech offers another option that combines the speed and convenience of running on an engineer's desktop PC with the accuracy of running on the real hardware. This approach is called virtualized software development. A model developed for use in a virtualized software development environment is so accurate the software cannot tell the difference; additionally, the model is fast enough that the software and test engineers are productive. This approach is based on running the binary object code unchanged, and thus can be used for everything other than the very final phases of system integration.

VIRTUALIZED SOFTWARE DEVELOPMENT COSTS FAR LESS THAN HARDWARE

Speedup

Simulation speeds up for two reasons: the host hardware gets faster and simulation algorithms improve. Unfortunately, target hardware also gets faster since its processors also benefit from improvements in semiconductor technology. Thus the most appropriate measure is the *slowdown*, how much slower the virtualized hardware runs in comparison with the real hardware.

In the 1980s, the slowdown on a normal mix of benchmarks, not corner-cases selected to make the simulation look especially good or bad, would run with a slowdown of 150-200X. That is, it took about 200 instructions on the host machine to simulate each one on the target. The underlying approach in those days was static interpretation, instruction by instruction.

In the 1990s, the slowdown was 30-100X. The underlying approach had switched to threaded code, replacing the code with a series of equivalent procedure calls.

At present the slowdown is 5-15X using JIT (just-in-time) compiler technology. Many target instruction sequences are translated into native host instruction sequences which are then executed directly by the hardware.

In the future, these advances are expected to continue with speed-ups dropping close to 1 under some circumstances. Indeed, on idle loops, spin locks and device polling, it is already possible to simulate with many fewer instructions simulated than to actually execute on the hardware, a slowdown of less than 1.

The complexity of the software debug and test environments makes them expensive to construct. Large electronic system companies have enormous rooms containing acres of racks of equipment, while smaller companies either require increasing amounts of already scarce time at inter-operability laboratories or accept lower quality code if there is insufficient time.

Large electronic system companies budget tens to hundreds of millions of dollars for these test environments. Virtualized software development can save a significant portion of the test lab hardware and result in enormous savings, since virtual platforms can run on any hardware, including a commodity-level PC with significantly lower capability than the system being simulated and a significantly lower price point. As off-the-shelf PCs become faster, it becomes much more competitive to use them for simulations of non-commodity hardware. Additionally, replicating whole virtual test racks costs only a slight fraction of the prohibitively expensive cost of replicating real hardware test racks, meaning a virtual test rack can be used by every engineer in the company. This economical scalability contributes to significantly reduced capital requirements and operational expenses.

VIRTUALIZED SOFTWARE DEVELOPMENT PERFORMS BETTER THAN HARDWARE

Virtualized software development is advantageous in cases where the real hardware is not available or the design has not been completed. With virtualized software development, software development can start as much as a year earlier, which can be an entire product cycle of consumer electronics such as cell-phones or PDAs.

Even when hardware is available, virtualized software development has many advantages. Quite simply, it is a much more programmer-friendly debugging environment.

The three main areas of superiority are:

- checkpoint/restart
- determinism
- non-intrusiveness

Simics can checkpoint an execution (for example after an initial boot), which allows subsequent runs to restart from the checkpoint. The target software cannot detect any variation in its world and simply continues executing at the next tick of the clock after a restart. Real hardware can typically only be re-booted from a system reset, and even then there is the possibility of many state changes.

Simics is deterministic, meaning that re-running the same execution will produce the same result. Even when real-world timing occurs, such as keyboard input, it is possible to re-run the simulation and capture the keystrokes and the keystroke timing, so that the simulation will produce an identical output. Real hardware is not like this. It is notoriously difficult to reproduce subtle timing bugs. On a multi-processor system it is almost impossible to repeat an identical run, but Simics is deterministic even with multi-processor systems *and* even when the simulation is distributed across multiple host computers.

Simics is non-intrusive. When the simulation stops at a breakpoint, the software cannot tell—not even the operating system kernel or the interrupt routine for the timer. Simics halts all the hardware devices including timers, disks, and cache memories, allowing anything to be inspected without causing unwanted side-effects. Stopping real hardware for inspection is impossible since basic operating system services may need to keep running if the execution will continue eventually. Even when this is not the case, halting the execution has side-effects, leading to the notorious phenomenon known as “heisenbugs”—bugs that vanish when they are observed but re-appear if the system is run normally.

SUMMARY

Virtutech’s Simics provides a virtualized software development environment for debugging and testing software to run on non-standard hardware, which includes any system other than a PC or workstation. Simics provides both

fidelity and performance, running software unchanged with real-world workloads.

Simics is unique. Simics can:

- model hardware so accurately that production binaries can run, including real-time operating systems, device-drivers, and protocol stacks
- run object code without access to source code, making it possible to use Simics in regression testing, field-engineering and support
- run lightly-loaded target systems faster than real time because it optimizes idle loops when waiting for hardware events
- run heavily-loaded target systems at speeds approaching real-time
- scale to systems containing hundreds of processors and thousands of devices distributed across multiple host machines.

With peak single processor performance exceeding four billion virtual instructions per second under optimal conditions, Simics is the world's highest performance virtualized software development solution. Simics is the only system that provides deterministic distributed simulation with checkpoint and restart, making it not only more affordable than developing on real hardware but much more productive as well.



CONTACT INFORMATION

Corporate Headquarters	Virtutech, Inc. 1740 Technology Drive #460 San Jose, CA 95110 Tel: 408- 392-9150 Fax: 408- 608-0430
European Sales and Support Office	Virtutech AB Drottningholmsvägen 14, 3tr SE-112 42 STOCKHOLM SWEDEN tel +46 8 690 07 20 fax +46 8 690 07 29
Internet	info@virtutech.com http://www.virtutech.com

© Copyright 2005, 2007 Virtutech, Inc. All Rights Reserved.

TRADEMARKS. Virtutech, the Virtutech logo, and Simics are trademarks or registered trademarks of Virtutech, Inc. and/or its subsidiaries, in the United States and/or other countries.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. VIRTUTECH MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Virtutech, Inc., 1740 Technology Drive #460, San Jose, CA 95110, USA