WIND™

Simplify Your Move to Multi-core with Simulation

Wind River Simics Is a Key Component in the Science of Multi-core Adoption

WHEN IT MATTERS, IT RUNS ON WIND RIVER

EXECUTIVE SUMMARY

It has been almost a decade since Advanced Micro Devices (AMD) proudly proclaimed that multi-core would soon be "the pervasive computing model." It's happening, but slowly. Yes, the benefits of multi-core are real. The performance gains, reduction in power consumption, productivity advantages, and potential for consolidation have all been proven. But the added complexity of multi-core system development has slowed adoption.

Simulation in general offers a way to accelerate the business benefits of multi-core without further complicating life for development teams. Simulation using Wind River® Simics® makes it possible to overcome the complexities of parallelism while actually simplifying life for developers. Further, Simics makes design, debug, and test of multi-core software more efficient than is possible via traditional methods.

This paper shows how simulation solves the problems that have been holding back multi-core adoption, and provides an overview of the capabilities of Simics for multi-core software development.

TABLE OF CONTENTS

WIND

THE MULTI-CORE ADVANTAGE

For users of technology, the appetite for performance is insatiable—particularly in today's era of instant-everything, mobile computing, streaming content, Big Data, and interconnected devices.

Developers have extracted every ounce of performance possible from the device hardware, the operating system, the programming language, and the compiler, but the aggregate performance of a single thread has hit a wall. The most efficient way to wring more performance out of a system is through parallelism—multithreading, multiprocessing, and multi-core systems. Multi-core technology has proven to deliver multiple benefits, including:

- **High performance with easier scaling:** Multi-core processors boost performance by adding processing power with minimal latency, helping to ease the processing burden on performance-intensive use cases such as mobile computing, network packet acceleration, and image, radar, audio, and data processing. And scaling is relatively simple: Just add more processors or cores.
- **Lower power consumption:** In some cases, running multiple cores at a lower frequency yields an almost tenfold increase in bandwidth while the total power consumption is reduced by a factor of four.
- **Consolidation of resources:** Using multiple discrete processors leads to high bill-of-materials cost, power consumption, footprint, and maintenance costs. Multi-core allows these discrete systems to be combined on a single piece of silicon, thus lowering costs.
- **Easier enhancement:** Migration of existing legacy applications to more modern platforms via multi-core provides the ability to augment existing applications with new features.

MULTI-CORE DEVELOPMENT CHALLENGES

While the advantages of multi-core are alluring to business leaders, embedded design teams are sometimes reluctant to make the move. The primary reason: Exploiting the benefits of multi-core requires a new approach to software design, and teams are often unfamiliar with the new generation of development tools and best practices available for multi-core and parallel development.

As a result, many teams end up either reinventing the wheel—at great expense in time and materials—or simply avoiding the adoption of multi-core altogether.

For developers, the biggest challenge of migrating to multi-core is dealing with the resource contention issues that arise. Single-core processors are relatively easy to design with because developers have full access to and control over processor resources, whereas in a multi-core system two or more processes contend for access and control—and that can create issues such as the following:

- **Race conditions:** When two threads or two tasks are working on the same dataset, they both read a variable, modify it, and store it—frequently resulting in the loss of the first update and a race to be second due to lack of synchronization. The solution to race conditions is to protect all shared data with locks, which synchronizes access, ensuring the expected order of events. However, correct locking is difficult to implement. In some cases a lock is missed, resulting in an update from one task being improperly overwritten by another; in other cases updates arrive in the wrong sequence; in still others a "deadlock" occurs, with tasks waiting for each other to release a lock.
- **Timing issues:** It is easy to make inaccurate assumptions about timing based on past experience. For example, a developer might assume that initialization will take a long time, leaving plenty of time for another task to execute—when in fact initialization finishes fast.
- **Bottlenecks:** Race conditions, lock contention, inaccurate assumptions, and memory ordering problems can all cause performance to suffer through frequent interruptions.
- **Debugging issues:** With multi-core processors there is rarely more than a single debug port per chip, so there is limited visibility into hardware. In addition, small changes in timing radically alter system behavior, resulting in a perceived lack of determinism. This means rerunning a program can yield different results, making it hard to reproduce bugs. And the "Heisenbug" phenomenon can occur: Inserting probes to trace behavior can actually alter behavior, making it even harder to find and fix bugs.

SIMULATION ADVANTAGES FOR MULTI-CORE DEVELOPMENT

Simulation can help solve many of the unique challenges of multi-core development. With simulation, developers and researchers use virtual target hardware in place of physical hardware for software development, debugging, testing, system integration, and more. As a result, developers can have ubiquitous access via a simulated system.

With Simics, teams can also simulate systems of virtually any size—from processors and memory to complete boards and devices, racks of boards, and even complete networks and systems-of-systems.

This fast and accurate virtual environment enables embedded development teams to adopt approaches and techniques that are simply not possible on physical hardware. For example, they can view and modify every device, register, or memory location; they can freeze, save, and restore the whole system; and they can run the whole system in reverse to find the source of a bug. Simply put, developers can break the rules of embedded multi-core development.

Using simulation for multi-core delivers the following business-level and technical advantages:

- **Drives quality sooner:** Simulation accelerates every aspect of embedded development—across the entire product lifecycle. Your architecture exploration begins and ends sooner. You explore new silicon choices before the silicon exists. You try out multiple hardware options in days rather than months. You start software development earlier. You debug software before the hardware exists. And you integrate and test faster. For example, with simulation, activities such as board bring-up, system integration, and testing can begin before physical hardware is even available. Simulation can also accelerate prototyping by allowing developers to use virtual prototypes instead of physical prototypes, and it can speed up architectural analysis by enabling teams to run multiple what-if scenarios and try multiple software alternatives before committing to one. It all adds up to a reduction in time-to-market of up to 66%, according to studies and actual results achieved by Wind River customers.

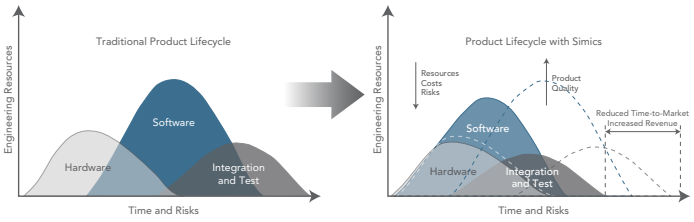

Figure 1: With Simics, teams can compress development and testing schedules

- **Increases flexibility:** Simics works with systems of any size, type, and level of complexity. And you can continue to use the virtual platform even after hardware development is complete.
- **Simplifies provoking, reproducing, and locating flaws:** Simics gives you both repeatability and reverse execution. In essence, it's like having a DVR for testing and debugging. With Simics developers can:
  - **Provoke errors:** Through simulation, development teams can run the complete, real software stack, change systemconfigurations (number of CPUs, CPU speeds, etc.), stop processors, inject faults, and rerun the test. For example, with simulation it is possible to test single core and multi-core setups on a range of frequencies, run the program 20 times on each setup, and determine the percentage of runs triggering race conditions. Simics will show the difference between single core and multi-core in bug triggering frequency—in other words, simulation doesn't hide the bug, so it can be isloated and addressed.
  - **Reproduce errors:** It's always easier to understand a flaw after you have seen the results. With simulation, developers can pause at any time; the software running in the simulated environment simply cannot detect the pause, so developers can inspect the state even at points where real hardware would be unable to stop. Developers can even run the whole simulation backward. Virtual hardware imposes a known parallel execution, so teams have perfect control over timing and communications.
  - **Locate errors:** Instrumentation and debugging with a simulator does not affect the system state. There are no probe effects, so you get no Heisenbugs. Simulation provides superior visibility into system state, with perfect synchronous stop and step of the system and reverse execution to retrace steps leading to an error

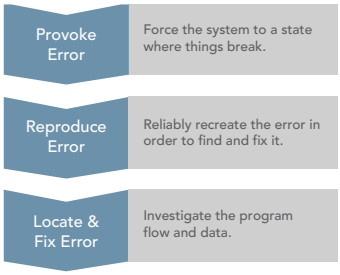

Figure 2: Using Simics to debug multi-core software

- **Proven best practices:** Simics has been in use for multiprocessor and multi-core development for over a decade. The tools have been honed initially in the multiprocessor server world and more recently in the multi-core desktop world, and now are available for use in the embedded world.

DISPELLING THE MYTHS ABOUT SIMULATION FOR MULTI-CORE DEVELOPMENT

Simulation has been widely used in the embedded industry for several years, and has proven its benefits at a broad spectrum of companies—from aerospace and defense contractors such as General Dynamics, which uses simulation to design satellites for NASA; to industrial and medical companies such as AMCC, IBM, Hitachi, and Tektronix; to network equipment makers such as Alcatel-Lucent, Cisco, Ericsson, Huawei, Motorola, Nortel, and many others.

Yet misconceptions persist about simulation and its effectiveness compared to traditional development methods. Here are a few common myths about simulation, along with a few facts about Simics to help set the record straight.

Myth 1: Simulation is not fast enough.

With Simics, you can use your existing server infrastructure to parallelize your testing—that is, you can run the platform under many different situations in parallel so that you can accelerate your work and work more effectively.

Simulation also speeds up the integration process. With simulation, system integration can begin solely on virtual platforms, expanding to a combination of virtual and physical hardware, and finally to fully physical hardware as those models, software, and hardware elements become available. Throughout the process, the same toolchain used on physical hardware can be used. Because integration is performed so much earlier in the development process compared to traditional hardware-based approaches, the "integration" moves from a single high-risk task that begins largely after all hardware is delivered to a much lower-risk activity that progresses in parallel with both software and hardware development. The result is continuous integration, which is far faster and more efficient than traditional methodologies.

In addition, simulation features such as checkpoints and scripting enable developers to reduce the total effort. System reboot, for example, can take up a substantial portion of the time spent in debug during application development. If a checkpoint is taken after boot and before application loading, reboot time can be eliminated—and code that doesn't run is always faster than code that does.

Myth 2: Simulation is not accurate enough.

Simics timing is not cycle-accurate or performance-accurate—and that is a huge advantage for developers and security researchers. Software developers don't want a cycle-accurate simulation because even the hardware isn't cycle-accurate; there is variation from one run to the next. Yet the determinism provided by Simics makes it possible to get exactly the same run as many times consecutively as needed until you've debugged it.

Moreover, Simics allows you to perturb the system state and check for corner cases and edge conditions. For example, if two tasks are running in parallel and one of them is suddenly very slow or excessively fast, it may expose an insufficiency of locking of your data. With Simics you can get answers and address the issues before they become security vulnerabilities.

Myth 3: Simulation doesn't provide debugging capabilities.

In fact, Simics provides debugging capabilities that far exceed those of traditional debugging techniques or virtualization solutions.

For example, you can repeat and isolate bugs quickly with Simics using checkpoints, reverse execution, run-to-run repeatability, hardware inspection, and full system visibility and control—and Simics also includes a powerful source code debugger with a complete graphical user interface (GUI) and intuitive commands that make it easy to script debugging activities and automate debugging.

With Simics, debugging no longer requires a developer to be lucky as well as skilled, experienced, and patient. Simulation trivializes efforts to repeat and isolate the bug, which removes luck from the equation; skill and experience are still necessary.

Of course, some debugging can only be done with real hardware and a JTAG connection, and that is why Wind River makes it easy to extend your debugging capability with Wind River JTAG debugging solutions (see www.windriver.com/products/JTAG-debugging/ for more information).

Myth 4: Simulation doesn't run real code.

Simics actually does run the complete unmodified software stack, so it offers full binary fidelity—which no other simulation solution currently provides.

EXAMPLE: USING SIMICS TO DEVELOP A MULTI-CORE OS

Wind River has used the Simics platform in its own key development projects for years. One early project was the development of a multi-core operating system using one of the first chips targeting this market, the Freescale MPC8641D dual-core processor—but the silicon was not available when Wind River development was slated to start. By using Simics and a virtual model of the Freescale system, the VxWorks® SMP development team was able to start work close to a year ahead of silicon availability—and greatly simplify and accelerate the debugging process.

CONCLUSION

Simulation offers more than "promise" and "potential" for accelerating development timeframes and increasing efficiency, productivity, and collaboration within embedded development teams. It has proven its capabilities and its business value, and its use can help accelerate the adoption of multi-core in enterprises of all types and sizes.

Wind River is the one vendor that can deliver both the industry's most advanced simulation platform—Wind River Simics—and a comprehensive approach to multi-core enablement. With the industry's most complete multi-core software portfolio and expertise, coupled with expert service and support offerings tailored to the needs of multi-core development teams, Wind River has proven to be uniquely capable of leading the industry in the transition to multi-core.

LEARN MORE

For additional information about Wind River Simics, or to schedule a personalized demonstration of the capabilities of Simics, contact Wind River at 800-545-9463. For more information about Wind River Simics, visit www.windriver.com/simics.

WIND

Wind River is a global leader in delivering software for the Internet of Things. The company's technology is found in more than 2 billion devices, backed by world-class professional services and customer support. Wind River delivers the software and expertise that enable the innovation and deployment of safe, secure, and reliable intelligent systems.

© 2013 Wind River Systems, Inc. The Wind River logo is a trademark of Wind River Systems, Inc., and Wind River and VxWorks are registered trademarks of Wind River Systems, Inc. Rev. 09/2013