Virtualized Systems Development with Simics – FSW Pasadena 2010

Chad Margolin,
Simics Technical Account Manager
Chad.margolin@windriver.com
What is Simics?

Simics® - an adaptive virtual platform that enables customers to define, develop and deploy electronics systems more efficiently.
What is a Virtual Platform?

- A piece of software
- Running on a regular PC, server, or workstation
- Functionally identical to the target hardware
- Runs the same software as the physical hardware system
  - Production binary
  - “Test as you fly, fly as you test”
Simics Scales

Examples

» Satellite constellation, telecom network

» Telecom rack, avionics bay, blade server

» C-W SVME183, VPX6-187 GE-IP VG5, …

» PCIe, RapidIO, I²C, Custom FPGA

» PPC440GP, P4080, OMAP, Octeon CN3860

» PPC750, Core 2, e500mc, POWER6, MIPS64
Key Characteristics Of Simics

- Wind River is the provider of Simics: a high-performance, high fidelity, full system simulator
  
  - **High Performance** – fast enough to run *real* software loads (typically 100’s of MIPS, up to multiple GIPS)
  
  - **High Fidelity** – run full production software, including firmware, device drivers, hypervisor, RTOS/OS, application software
  
  - **Full System** – simulate entire systems with complete machines, networks, backplanes, etc, not just processor cores, or SoCs, or boards
  
  - **Development Environment** – provide the most powerful and effective develop, debug and test platform via tools and system control and visibility
  
- The true value of Simics is through enablement of process change: *Virtualized Software Development*
Simulating the Electronic System

The software can’t tell the difference
Runs binaries from real target

User program
Java VM  DB  Middleware

Operating system
Drivers  Hardware abstraction layer  Boot firmware

Simics
CPU  Bus  Disk  Network
CPU  PCI  Disk Ctrl  User Intf device
RAM  I²C  Flash  A/D
ROM  PCI  Bus  Network

Run your system software on your desktop
Complete production software

Identical build tools chain
Simulated (virtual) hardware

© 2010 Wind River. All Rights Reserved.
Three Steps of Debugging

1. Provoking errors
   - Forcing the system to a state where things break

2. Reproducing errors
   - Recreating a provoked error repeatedly and reliably

3. Isolating the source of the errors
   - Investigating the program flow and data
   - Depends on success in reproduction for efficiency
   - Identifying the line of code causing the problem

Virtual hardware helps with all three steps
Repeatability and Reverse Debugging

- Repeat any run trivially
  - No need to rerun and hope for bug to reoccur
- Stop & go back in time
  - Instead of rerunning program from start
  - Breakpoints & watchpoints backwards in time
  - Investigate exactly what happened this time
- This control and reliable repeatability is very powerful for parallel code!

On hardware, only some runs reproduce an error

With repeatability all runs reproduce an error

With reverse execution there is no need to iterate
Multi-* & Debug

- Limited visibility into hardware
  - Single debug port, multiple processors
  - High speed, concurrent execution

- Timing-sensitive chaotic behavior
  - Small changes in timing alters system behavior radically
  - Hardware variations impact software behavior

- Lack of determinism
  - Rerunning a program gives different results
  - Hard to reproduce bugs

- Heisenbugs
  - Inserting probes to trace behavior alters behavior
  - Bugs hide when they are being debugged

- System keeps running even if one core stopped
Traditional Hardware/Software PLC – Simplified

Define

- HW Spec.
- Architecture Design
- Final Rev HW

Develop

- Multiple Design Spins

Hardware Development

- Integration & Test

Software Development

- Product Spec.
- SW Spec.
- Firmware:
  - Boot code
  - Diagnostics
  - Drivers
  - BSP

Deploy

- Customers
- Partners
- Sales
- Support
- Marketing

© 2010 Wind River. All Rights Reserved.
VSD-based PLC

Define
- HW Spec.
- Architecture Design
- Multiple Design Spins
- Final Rev HW
- Architectural Investigation
- Product Spec.
- SW Spec.
- Firmware:
  - Boot code
  - Diagnostics
  - Drivers
- Application Development

Develop
- Progressive Integration & Test
- To Ecosystem & Customers

Deploy
- Customers
- Partners
- Sales
- Support
- Marketing

Customers
- Documents
- Demos
- Training
- Support

© 2010 Wind River. All Rights Reserved.
Simics 4.4 Overview

- First Customer Shipment (FCS): May 19, 2010
- General Availability (GA): May 31, 2010
- Simics 4.4 Highlights
  - Simics Analyzer for debugging, analyzing and verifying target software applications
  - Simics Extension Builder for extension plug-ins, integration to enterprise tools and additional work flows. Integrations with 3rd party simulators (ISS).
  - Mix C, C++, SystemC and DML models
  - Eclipse DML Editor improvements
  - Performance improvements
  - Freescale QorIQ P1/P2 Virtual Platform
  - Intel Core i7 Virtual Platform (Nehalem / Tylersburg)
Simics 4.4 Product Family

Simics Analyzer, WR Hypervisor Add-in
Simics Analyzer, VxWorks Add-in
Simics Analyzer, FSL Hypervisor Add-in
Simics Analyzer, QNX Add-in
Simics Analyzer, Linux Add-in
Simics Analyzer, OSE Add-in

Simics Ethernet Networking
Simics Accelerator
Simics Hindsight
Simics Model Builder
Simics Extension Builder
Simics Virtual Platforms
Simics Custom Virtual Platforms

© 2010 Wind River. All Rights Reserved.
Simics Analyzer

Software debug & analyze
- Graphical time line view
  - What runs where and when
  - Eclipse view
  - Run time execution details
  - Software context – processes and functions
- Full system process list
  - OS Awareness information
  - List of machines, CPUs, OSs, processes & status
  - Eclipse view

For heterogeneous systems
- Full system solution
- Particularly good for multicore systems
- Mixed architectures
  - Power PC, IA, ARM, MIPS, etc
- Mixed OSs
  - VxWorks, Wind River Linux, Wind River Hypervisor, Linux, OSE, QNX, FSL Hypervisor
Simics Analyzer (cont)

Software verification
- Code Coverage
  - Non instrumented code (exact same binaries that goes into the end system)
  - Statement coverage
  - Both C level and asm level coverage data

Available in Simics workflow
- No need to switch to other tool / workflow
- Presented in standard web browser
- Coverage data also available in pure text format for integration purposes
Simics Analyzer – Graphical Time Line View
Simics Analyzer – Code Coverage

Coverage analysis

Binary: ebony-simplex

Source Files | Disassembly | /home/david/simics/4.4/test64-linux/tx00_coverage/generic/scratch/run-1/htm

Covered 4 of a possible 8 lines (50.0%).

```c
#include "tr.h"
static unsigned char table[256];

void add_translation(unsigned char from, unsigned char to)
{
    table[from] = to;
}

unsigned char tr(unsigned char c)
{
    if (table[c])
        return table[c];
    else
        return c;
}
```
System-Level Simulation Benefits

Checkpoints and restore

Multicore, processor, board

Real-world connections

Repeatable fault injection on any system component

Scripting

```
con0.wait-for-string "$n
con0.record-start
con0.input "./ptest.elf 5\n" con0.wait-for-string "."
$r = con0.record-stop
if ($r == "fail.") {
   echo "test failed"
}
```
Simics Software Debugging Benefits

Synchronous stop for entire system

Determinism and repeatability

Reverse execution

Unlimited and powerful breakpoints

Trace anything

Insight into all devices

```
break -x 0x0000->0x1F00
break-io uart0
break-exception int13
```
Thank you

WindRiver Simics