Using Simulation to Develop and Maintain a System of Connected Devices

Massimiliano De Otto
Technical Account Manager
THE CHALLENGES OF DEVELOPING CONNECTED ELECTRONIC SYSTEMS
Mobile Networks Update - Lessons Learned

- Mobile Networks can be extremely complex:
  - Huge number of deployed devices: > 1,000,000,000
  - Device and Infrastructure are heterogeneous
  - Vendor Customization

- Complexity must be managed to be successful
  - Roles and responsibilities have to be defined
  - Role activities have to be decoupled

- Development cycles are shorter and shorter

- Cost of change must be low
  - Introducing a change should have a low impact,
  - No disruption of Service
  - Must be scalable

- How can I manage change, test, validation, What if...?
What Is Wind River Simics?

Wind River Simics is a full system simulator used by software developers to simulate the hardware of large and complex electronic systems.

- Simulate any size of target system.
- Run unmodified target binaries.

Simics allows you to break the rules of embedded product development.
Complete Virtualization

- User program
- Target operating system & Middleware
- Simulated target hardware
- Simics
- Host operating system
- Host hardware

- All software: arbitrary & unmodified. Same as on a real system.
- Arbitrary; ARM, x86, x86-64, SparcV8, SparcV9, MIPS32, MIPS64, PPC32, PPC64, H8, SH, C64, ...
- Linux, Windows
- 32-bit and 64-bit PCs

Host hardware

Host operating system

Simics

Target operating system & Middleware

User program
Simulating the Electronic System

Run your system software on your desktop

Complete production software

Identical build tools chain

The software can’t tell the difference

Runs binaries from real target

Simulated (virtual) hardware

The software can’t tell the difference

Runs binaries from real target

Simics

CPU

CPU

RAM

ROM

Bus

PCI

I²C

Disk

Disk Ctrl

Flash

Network

User Intf device

A/D

Java VM

DB

Android

Operating system

Drivers

Hardware abstraction layer

Boot firmware

User program

Run your system software on your desktop

Complete production software

Identical build tools chain

The software can’t tell the difference

Runs binaries from real target

Simulated (virtual) hardware

CPU

CPU

RAM

ROM

Bus

PCI

I²C

Disk

Disk Ctrl

Flash

Network

User Intf device

A/D

Java VM

DB

Android

Operating system

Drivers

Hardware abstraction layer

Boot firmware

User program
Simics Can Model Any System or Networked Systems

- Simulate multiple networked devices
- Mixed architectures, mixed software
- Multi-core, multi-processor, multi-board
- Control all boards as a single entity
  - Breakpoint stops all
  - Save and restore system state of all boards

Ethernet, wifi, serial, etc
COOL SIMICS FEATURES
System-Level Features

Checkpoint and restore

Multicore, Processor, Board

Real-World Connections

Repeatabile fault injection on any system component

Scripting

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

Mixed endianness, word sizes, heterogeneity
Simulation provides cool debugging features

Synchronous stop for entire system

Determinism and repeatability

Reverse execution

Unlimited and powerful breakpoints

break -x 0x0000->0x1F00
break-io uart0
break-exception int13

Trace anything

Insight into all devices
Repeatability and Reverse Debugging

- Repeat any run trivially
  - No need to rerun and hope for bug to reoccur

- Stop and go back in time
  - No rerunning program from start
  - Breakpoints and watchpoints backward in time
  - Investigate exactly what happened this time

This control and reliable repeatability is very powerful for parallel code.
A Simulated Networked Target System

- Three target machines running any Android stack,
- A management machine to push updates,
- All interconnected in a single network, with external world connectivity,
- All contained inside a single simulation session
Network Fault Injection

- Causing faults at the 10.10.0.22 machine’s network interface
  - Simics Ethernet probe attached to the network interface
  - Fault injection module randomly dropping 20% of packets or corrupt packets randomly
- See how Software react
Simics Fault Injection

- **Simics target control**
  - Access any part of target
  - Change any part of target state or configuration
  - Scripting for precise targeting and replay of faults
  - No permanent damage to target
- **Purpose**
  - Test system-level fault handling
  - Test fault low-level fault detection
  - Model failing hardware

- **Example fault actions:**
  - *Dropping packets on networks*
  - Injecting bad checksums
  - Network partitioning
  - Delaying hardware replies
  - Transient memory corruption
  - Permanent memory corruption
  - Transient register corruption
  - Permanent register corruption
  - Data bus transmission errors
  - Address bus transmission errors
  - Triggering spurious interrupts
  - Permanent subsystem failures (disconnecting from system)
  - Processor crash
  - Temperature sensor reporting overheating
  - Error reporting registers flagged
WIND RIVER