Software Verification for Low Power, Safety Critical Systems
Agenda

- Software Verification for Systems
- Low Power, Safety Critical
Agenda

- Software Verification for Systems
  - Many processors with lots of software
  - Software Verification requirements
  - Embedded Software Development Issues
  - Adopting Simulation (Virtual Platforms)
  - Advanced tools available with simulators

- Low Power, Safety Critical
Modern SoCs Have Many Concurrent Processing Elements

Renesas R-Car H2: Automotive infotainment and ADAS
SW Verification Requirements

- **Hardware Dependent Software (HDS)**
  - Most complex foundation layer
    - Drivers, hypervisors, assembly libraries, operating system
  - Buried problems often appear elsewhere in a system, leading to misdirected analysis
  - Ripe for corner case type issues
  - Post development bugs hardest to fix
  - Testing needs to be platform centric not application centric

- **Modern SoC SW verification is complex**
  - SMP/AMP multicore interaction
  - Shared memory & devices
  - Extensive accelerators, peripherals
  - Externally authored, complex libraries
  - Complex SW/HW interaction (e.g. power)
    - How to estimate power consumption and test power management strategies over a wide range of conditions?
Embedded Software Development Issues
(in no specific order)

- Schedule
- Quality
- Functionality
- Timing, power constraints
- Security / safety
- Predictability of the software engineering task: management accuracy on software resource and schedule requirements is +/- 50%
- Unknown / unmeasurable delivery risk
Hardware-Based Software Development

- Has timing/cycle accuracy
- JTAG-based debug, trace
- Traditional development board or hardware emulator based testing
  - Late to arrive
  - Limited physical system availability
  - Emulators are too slow to run enough system scenarios
  - Limited external test access (controllability)
  - Limited internal visibility
    - How to observe power consumption?

- To get around these limitations, software is modified
  - printf
  - Debug versions of OS kernels
  - Instrumentation for specific analytical tools, e.g. code coverage, profiling
  - Even using different tool chains and libraries

- Modified software may not have the same behavior as clean source code
Virtual Platforms Complement Hardware-Based Software Development

- Current methodology employs testing on hardware
  - Proven methodology
  - Has limitations
  - We are at the breaking point

- Virtual platform based methodology delivers controllability, visibility, repeatability, automation

**Application Layer:** Customer Differentiation

**Middleware:** TCP/IP, DHCP, LCD, ...

**OS:** Linux, FreeRTOS, µC/OS-III, ThreadX, ...

**Drivers:** USB, SPI, ethernet, ...

Virtual platforms – software simulation – provide a complementary technology to the current hardware-based methodology.
Advantages of Virtual Platform Based Software Development
(Instruction Accurate Simulation)

- Earlier system availability
- Easy access for entire team
- Runs actual binaries, e.g. runs ARM executables on x86 host
- Fast, enables quick turnaround and comprehensive testing
- Full controllability of platform both from external ports and internal nodes
  - Corner cases can be tested
  - Errors can be made to happen
- Full visibility into platform: if an error occurs, it will be observed by the test environment
- Easy to replicate platform and test environment to support automated continuous integration (CI) and test automation / regression testing on compute farms
Building the Virtual Platform

- The virtual platform is a set of models that reflects the hardware on which the software will execute
  - Could be 1 SoC, multiple SoCs, board, system; no physical limitations
  - Functionally accurate, such that the software does not know that it is not running on the hardware
- Models are typically written in C or SystemC
- Models for individual components – interrupt controller, UART, ethernet, … – are connected just like in the hardware
- Peripheral components can be connected to the real world by using the host workstation resources: keyboard, mouse, screen, ethernet, USB, …
Imperas Tool Architecture

Application Software & Operating System

Virtual Platform
- Peripheral
- Memory

Simulation Engine

SlipStreamer API

CPU HELPER

OS HELPER

VAP TOOLS

Multiprocessor / Multicore Debugger

Eclipse IDE

TESTBENCH

Output Data
Trace Profile Coverage Schedule...

© 2016 Imperas Software Ltd (www.imperas.com)
Running Software on a Virtual Platform
BareMetal Platform

- Quickly execute any program on a ‘default’ BareMetal platform
- Simply run the executable with the name of the application object
Virtual Platform Booting Linux Kernel on ARM Cortex-A57 / Cortex-A53 big.LITTLE

Imperas OVP ARM MultiCluster model (big.LITTLE)

Interrupt Distributor - GIC-400/GIC-500 (GICv2/GICv3)

ARM Cortex-A57

ARM Cortex-A53

Cache Coherent I/F CCI-400/CCI-500

Memory

UART0_0

simulate
ostrace
gtrace
And many many tools can be used with the simulation

- Trace
- Coverage
- Profile
- Memory analysis
- Dynamic (Temporal) Assertions
- osAware: context, schedule, task, ...

- All at the flick of a switch…
- All non-intrusively…
- All without requiring any changes to software…
  - Runs with production binaries

=> Use of simulation (virtual platforms) is the modern way to develop and verify embedded software
And…

- When it comes to Continuous Integration
- When it comes to test automation
- When it comes to regression testing

=> Virtual platforms become essential for embedded software development
Agenda

- Software Verification for Systems
- Low Power, Safety Critical
Collaborations

Pontifical Catholic University of Rio Grande do Sul, (PUCRS) Porto Alegre Brazil.

University of Leicester, Leicester, UK.

Institute for Information Technology, Oldenburg, Germany.

This work was partially funded by the European Union’s Horizon2020 research and innovation programme under the grant agreement No 687902. More information at http://www.safepower-project.eu/
Working on Verification of Software

- Timing Analysis
- Power Analysis
- Fault Simulation
Working on Verification of Software

- Timing Analysis
  - using Instruction Accurate simulation / virtual platforms
    => software performance

- Power Analysis
  - using Instruction Accurate simulation / virtual platforms
    => software affected power

- Fault Simulation
  - using Instruction Accurate simulation / virtual platforms for software verification and compliance to standards such as ISO 26262
    => software fault tolerance
CPU Timing Estimator

- CPU Characterization Data for each CPU variant
  - Limitation: In-order processors
- Timing Estimator loaded onto CPU instance as binary intercept library
  - No edits/changes needed in CPU model binary, or platform, or other models
- Controlled by simulation command line arguments
- Takes account of instructions, branches, sequences, memory
Accuracy Evaluation vs Board

- Example:
  - Cortex-M4F running FreeRTOS
  - WCET and other benchmarks

- Worst case error <13%
- Average error across all benchmarks run <5%
- Most errors are <8%
- Simulation speeds with timing up to 500Mips
- Caveat: performance and accuracy are application dependent
Working on Verification of Software

- Timing Analysis
- Power Analysis
- Fault Simulation
Eu SafePower project

- Explorative project to enable the development of low power mixed-criticality systems through the provision of a reference architecture, platforms and tools to facilitate the development, testing, and validation
SafePower
Xilinx Zynq Virtual Platform

Harness

Zc702/zc706

Zynq

Zynq Processing Sub-system

CPU0 (ARM Cortex-A9MPx2)

uartps

slcr

Ethernet MAC

devcfg

Flash Controller

Memory

Smartloader

Zynq Programmable Logic

MicroBlaze

Memory

Interchangeable Zynq Programmable Logic Modules

MicroBlaze

Memory
SafePower:
Xilinx Zynq zc706 Virtual Platform
fentISS XtratuM Hypervisor on ARM Cortex-A9MPx2

Shown running fentISS Xtratum ‘Hello World’ example application
SafePower Dynamic Frequency Voltage Scaling (DVFS) Analysis

Application, Operating System, Firmware

System

Power Model

Understand

Runs on

Design-Time Opt.

Tracing
SafePower
Power Traces Analysis

- Application, Operating System, Firmware
- Power Model
- Analyze/Debug

Design-Time Opt.
Tracing
Runs on

Graph showing various metrics including:
- voltage_VCC_PINT
- frequency_CPU
- utilization_CPU_0
- utilization_Memory_Write_CPU_2.6219
- power_Total

SafePower: Power Monitoring in SW

Application, Operating System, Firmware

Power Consumption

Register Interface

Power Model

Power Sensor

Design-Time Opt.

Analyze/Debug

Tracing

voltage_VCC_PINT 1.0
frequency_CPU 6.67E8
utilization_CPU_0 1.0
utilization_Memory_Write_CPU_2.8219
power_Total 0.6987
Working on Verification of Software

- Timing Analysis
- Power Analysis
- Fault Simulation
Fault Simulation is needed

- Compliance with standards requires products to demonstrate reliability and tolerance to fault injections
  - Eg: automotive ISO26262 requires this
- Fault Simulation proved in the HW world it was a good way to show hardware had been manufactured fault free
- Fault Simulation can be used with virtual platforms to show that software is reliable under the presence of faults
Fault Simulation Overview

4 stages

1. Fault Free Simulation Run
2. Hardware & Software Introspection & Fault List Creation
3. Simulation Runs With Faults Injected
4. Golden Data

Detection Data

Analysis & Reporting
- \([A, B, C] = \{\text{Cortex-A9MPx1, X2, X4}\}\)
- 8,000 faults generated for each processor configuration
- 16 applications (Rhodinia HPC benchmarks) running under Linux 3.13
Summary

- Virtual platforms – software simulation – provide a complementary technology to hardware-based testing of software
- Virtual platforms can run full production binaries including hypervisors and hardware virtualization to assist in verification of safety critical systems
- Besides functional correctness, timing properties and power consumption are gaining importance
- Virtual platforms become essential with adoption of Continuous Integration methods for embedded systems software

- Virtual platforms for software verification help you
  - Achieve higher quality software
  - Reduce development schedules
  - Increase software project predictability
  - Reduce delivery risk
Thank you

For more information: info@imperas.com
www.imperas.com
www.OVPworld.org