MULTICORE DESIGN SIMPLIFIED

imperas

# Virtual Platform Based Linux Bring Up Methodology

**DAC Tutorial**
**19 June 2017**

**Simon Davidmann, Imperas**

# Agenda

MULTICORE DESIGN SIMPLIFIED
imperas

- New challenges posed by heterogeneous architectures
- Comparison of hardware-based and Virtual Platform-based methodologies
- Continuous Integration and Virtual Platforms
- Case study:  Linux bring up and testing on Altera Cyclone V SoC FPGA
- Demonstration

# Observations On Embedded Software

MULTICORE DESIGN SIMPLIFIED
imperas

1) As software complexity is increasing exponentially, companies need to adopt better ways to address problems, as ***eventually the existing methods will no longer be sufficient***

2) ***One serious failure changes everything***

3) There is a lesson to be learned from SoC design and verification: ***a structured methodology provides predictable execution and measurable reduction of risk***

⇒ Embedded software development domain needs to adopt a more formalized approach

# Agenda

MULTICORE DESIGN SIMPLIFIED
imperas

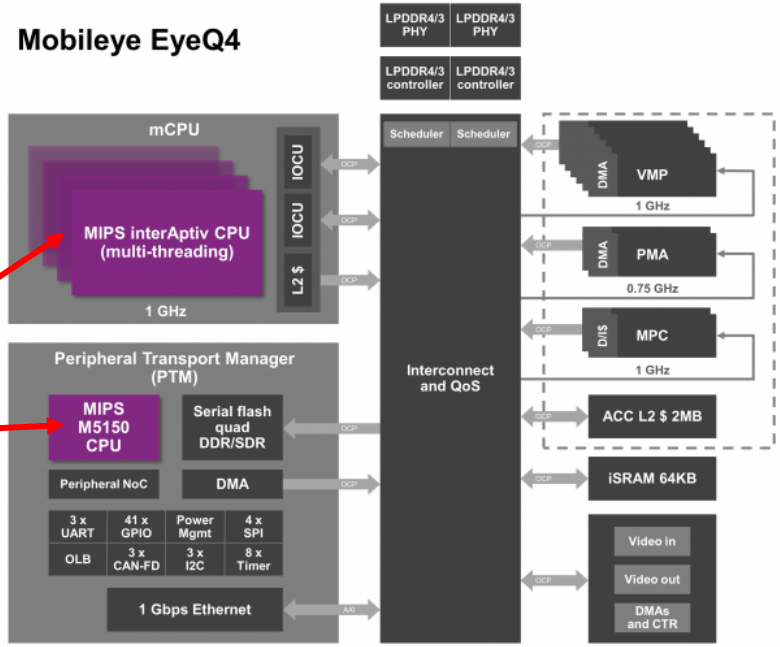
- New challenges posed by heterogeneous architectures
- Comparison of hardware-based and Virtual Platform-based methodologies
- Continuous Integration and Virtual Platforms
- Case study:  Linux bring up and testing on Altera Cyclone V SoC FPGA
- Demonstration

# Heterogeneous SoC Architectures

MULTICORE DESIGN SIMPLIFIED
imperas

- Heterogeneous can have various meanings for SoCs
  - Multiple different processors
  - Multiple computing elements, such as CPU plus GPU
- Why heterogeneous architectures?
  - Optimize the resources on the SoC for different tasks, e.g. application processor plus “minion” processors for power management, communication management, etc.

Mobileye EyeQ4 has
quad core (8 threads) MIPS interAptiv
plus MIPS M5150



# SoC System Architecture

- Historically there was not a common operating system across the processors
- Recently processor IP companies have developed processor configurations that are similar enough to allow a common Linux OS to run
  - Originally this was ARM big.LITTLE, with quad core Cortex-A15 plus quad core Cortex-A7, for power optimization of the application processor
  - The different processors did not run simultaneously; the operating system switched automatically between the quad core processors depending on application load

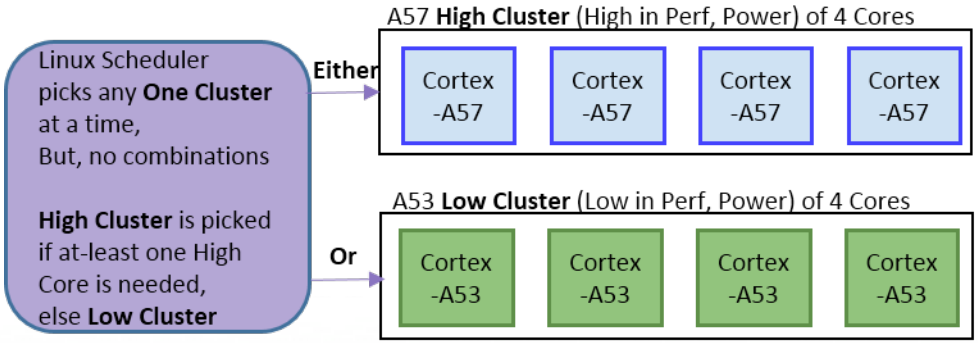
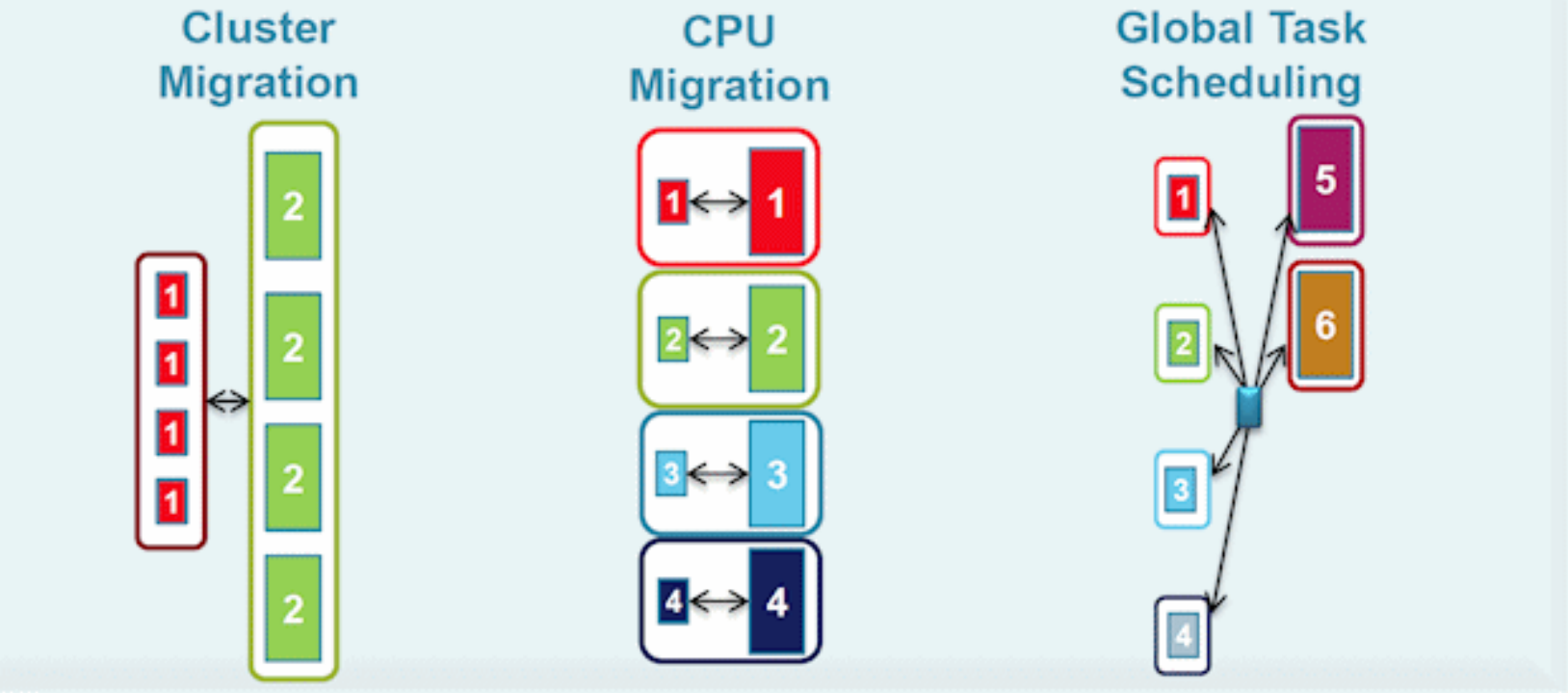



Diagram by Nvidia (https://en.wikipedia.org/wiki/ARM_big.LITTLE)

# ARM big.LITTLE Operating Modes

MULTICORE DESIGN SIMPLIFIED
imperas



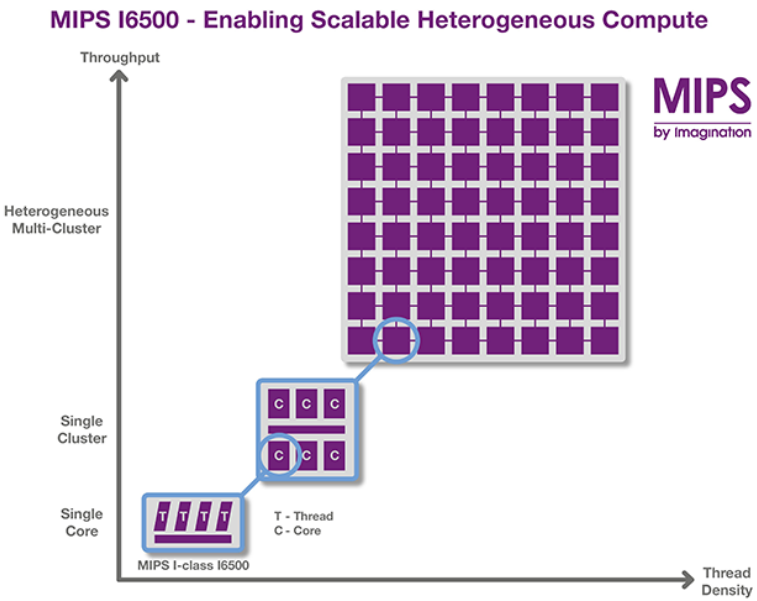

| Cluster Migration | CPU Migration | Global Task Scheduling |
|---|---|---|
| • OS can see one of two processor clusters.<br>• Only one cluster can be active at a time.<br>• Tasks run on either the LITTLE CPU cluster or the big CPU cluster. | • OS scheduler can see four CPU pairs, but only one CPU in each can be active at a time<br>• Tasks run on either the big CPU or the paired LITTLE CPU. | • OS scheduler can see all CPUs and all can be active at any time.<br>• Tasks can run on or move between the LITTLE CPUs and the big CPUs as defined by the scheduler. |

Source: ARM

# MIPS I6500 and ARM DynamIQ Extend Heterogeneous Compute Paradigm

MULTICORE DESIGN SIMPLIFIED
imperas

- Fully configurable architectures now enabled
  - Multiple heterogeneous cores per cluster
  - Multiple heterogeneous clusters per SoC
  - Linux and architecture now supporting > 1000 computing elements




Source: IMG

# Linux Complexity Increases With Full Heterogeneity

MULTICORE DESIGN SIMPLIFIED
imperas

- During boot, Linux needs to probe each core to determine characteristics so that correct hardware routines are installed
- Some Linux modules need to be updated
  - Cache initialization/handling: previously assumed homogeneous cache size, now needs to accommodate potentially different cache sizes for different cores/clusters
  - …
- With increased complexity comes increased porting and bring up issues

# Agenda

MULTICORE DESIGN SIMPLIFIED
imperas

- New challenges posed by heterogeneous architectures
- Comparison of hardware-based and Virtual Platform-based methodologies
- Continuous Integration and Virtual Platforms
- Case study: Linux bring up and testing on Altera Cyclone V SoC FPGA
- Demonstration

# Hardware-Based Software Development

MULTICORE DESIGN SIMPLIFIED
imperas

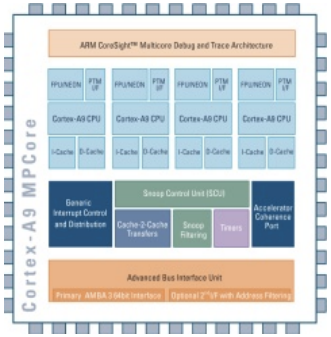
- Has timing/cycle accuracy
- JTAG-based debug, trace
- Traditional breadboard / emulation based testing
  - Limited physical system availability
  - Limited external test access (controllability)
  - Limited internal visibility
- To get around these limitations, software is modified
  - printf
  - Debug versions of OS kernels
  - Instrumentation for specific analytical tools, e.g. code coverage, profiling
  - Modified software may not have the same behavior as clean source code

# Virtual Platform Based Software Development

MULTICORE DESIGN SIMPLIFIED
imperas

- Instruction accurate simulation
  - Runs actual hardware executables
    - e.g. MIPS code on x86 PC
  - Models require only functionality that is needed for software
  - Fast, enables quick turnaround and comprehensive testing
  - Simulation-based development provides visibility, controllability not available from hardware
  - Same or better debugger access than hardware
  - Access for the entire team

SW Development running on virtual platform



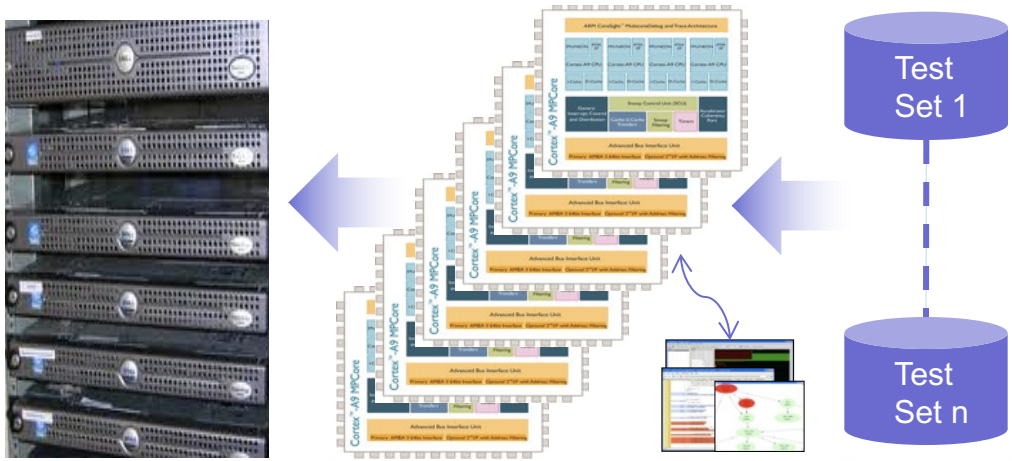

SW virtual platform (w/ OS, etc) running on host

Host Development Machine

# Advantages of Virtual Platform Based Software Development

MULTICORE DESIGN SIMPLIFIED
imperas

- Earlier system availability
- Full controllability of platform both from external ports and internal nodes
  - Can corner cases be tested?
  - Can an error be made to happen?
- Full visibility into platform: if an error occurs, will it be observed by the test environment?
- Easy to replicate platform and test environment to support regression testing on compute farms



# Virtual Platforms Complement Hardware-Based Software Development

MULTICORE DESIGN SIMPLIFIED
imperas

- Current test methodology employs testing on hardware
  - Proven methodology
  - Has limitations
  - We are at the breaking point
- Virtual platform based methodology promises controllability, visibility, repeatability
- Virtual platforms – software simulation – provide a complementary technology to the current methodology

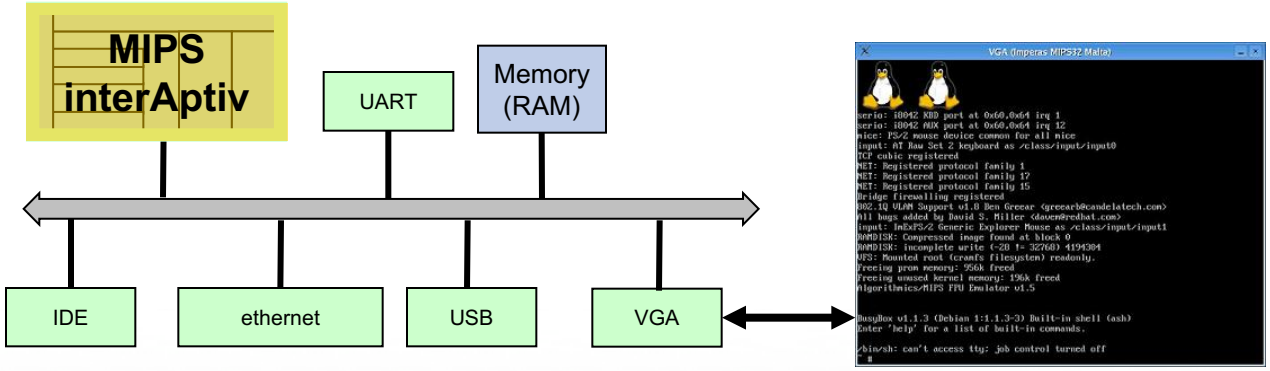
# Building the Virtual Platform

MULTICORE DESIGN SIMPLIFIED
imperas

- 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, <u>such that the software does not know that it is not running on the hardware</u>, i.e. runs the target binaries - unmodified
- 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, …

MIPS Malta Extendable Platform Kit (Linux)

**Run**



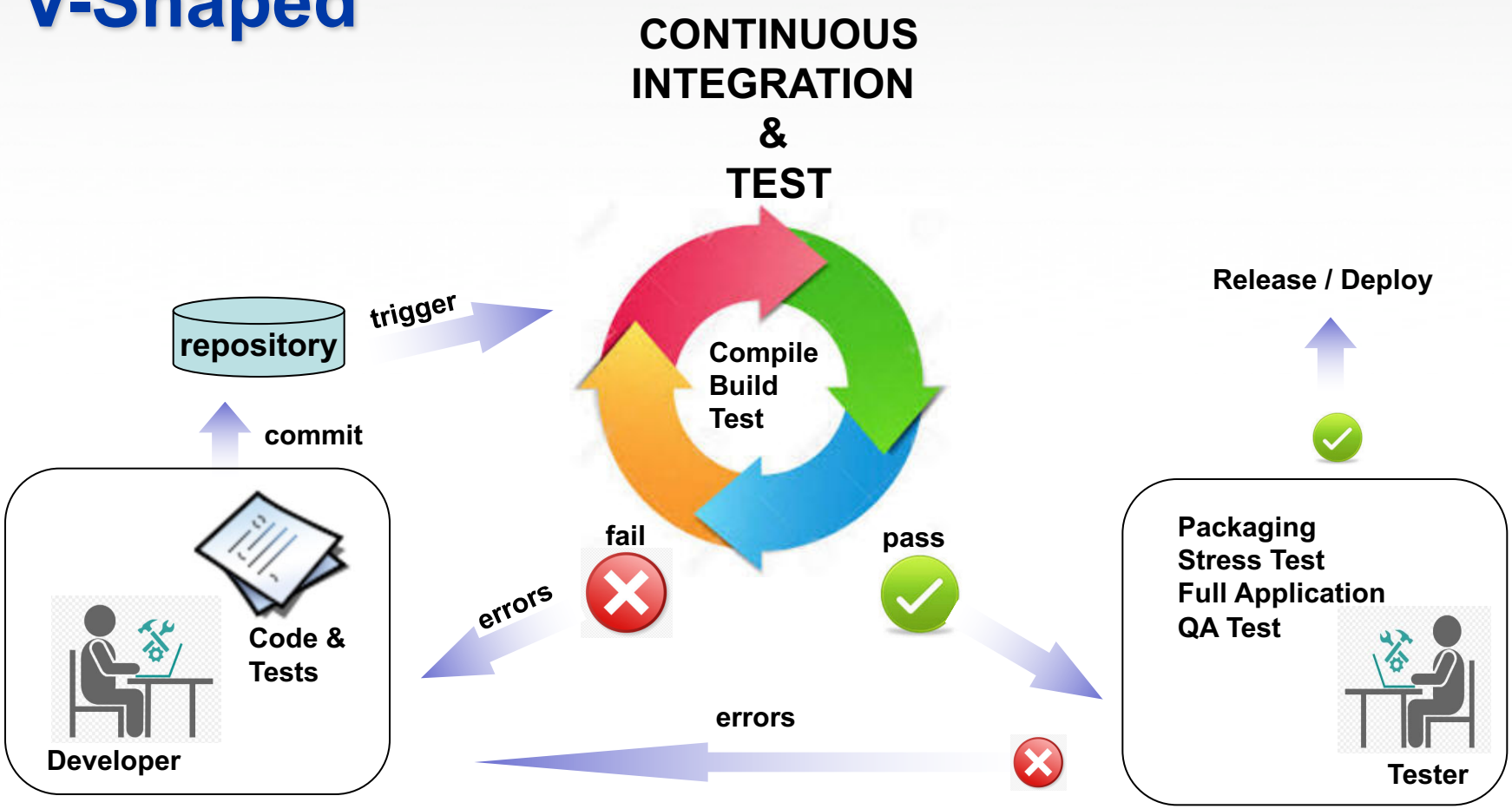
# Agenda

MULTICORE DESIGN SIMPLIFIED
imperas

- New challenges posed by heterogeneous architectures
- Comparison of hardware-based and Virtual Platform-based methodologies
- Continuous Integration and Virtual Platforms
- Case study:  Linux bring up and testing on Altera Cyclone V SoC FPGA
- Demonstration

# Modern Development Methodology: Agile, Not V-Shaped

MULTICORE DESIGN SIMPLIFIED
imperas



# Adopting Continuous Integration & Continuous Test for Embedded requires Simulation

MULTICORE DESIGN SIMPLIFIED
imperas

- Imagine a software build system without access to ‘make’ or ‘ant’
  - they enable effective build automation

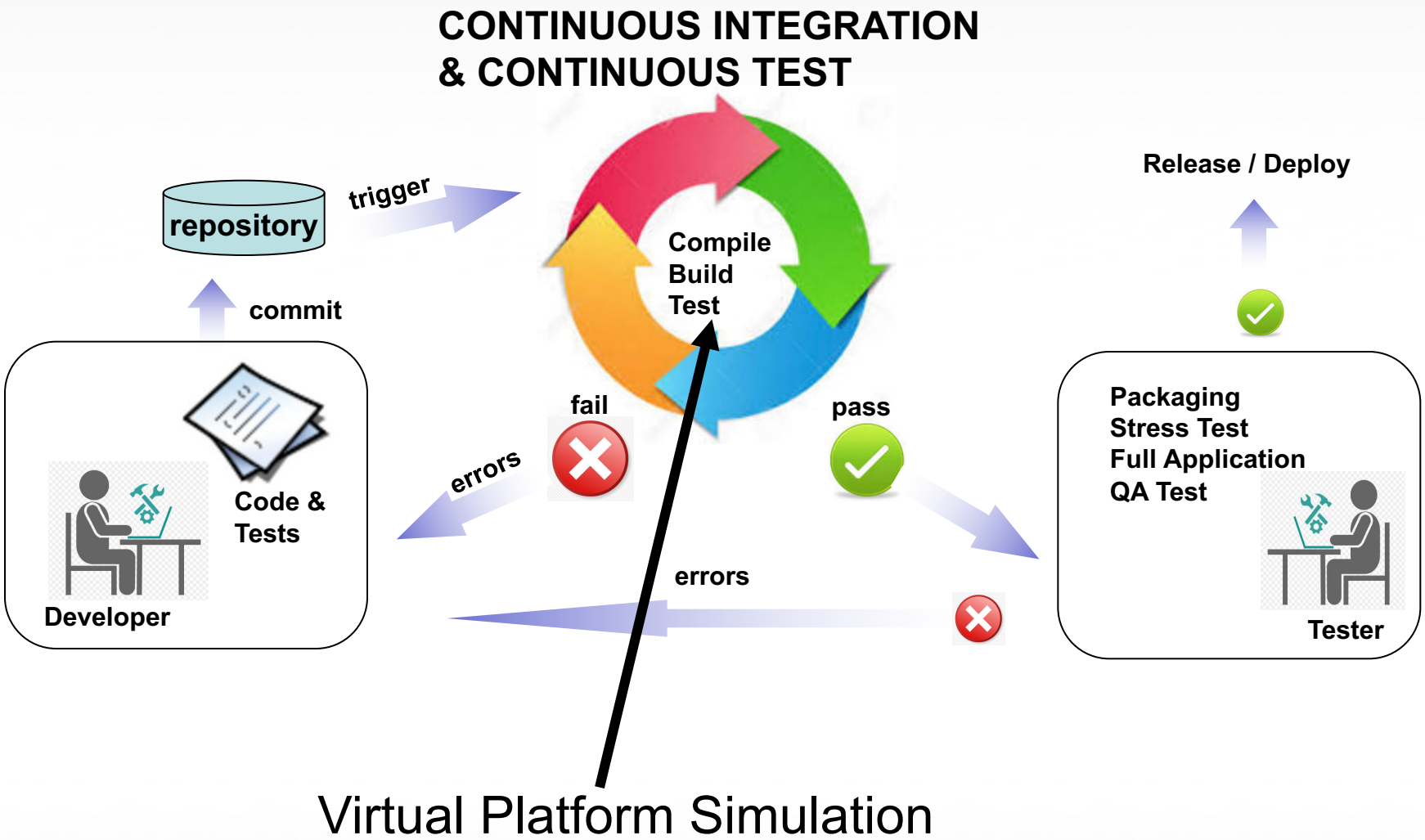
- Simulation enables the effective automation of testing embedded systems as part of a Continuous Integration / Continuous Test (CI/CT) environment
- Simulation enables full automation
  - with no manual intervention

- Use of hardware is just too hard

=> Virtual Platforms (simulation) enable CI / CT for embedded

# Simulation is a key component of embedded CI / CT environment

MULTICORE DESIGN SIMPLIFIED
imperas



# Motivation for Change: Benefits of Continuous Integration

MULTICORE DESIGN SIMPLIFIED
imperas

- Better code structure and quality
  - Frequent code check-in pushes developers to create modular, less complex code
  - Enforces discipline of frequent automated testing
  - Software metrics generated from automated testing and CI (such as metrics for code coverage, code complexity, and feature completeness) focus developers on developing functional, quality code, and help develop momentum in a team
- Easier debug
  - When unit tests fail or a bug emerges, if developers need to revert the codebase to a bug-free state only a small number of changes are lost
- Fewer major integration bugs
  - Immediate feedback on system-wide impact of local changes
  - Integration bugs are detected early and are easy to track down due to small change sets. This saves both time and money over the lifespan of a project.
  - Avoids last-minute chaos at release dates, when everyone tries to check in their slightly incompatible versions
- Constant availability of a "current" build for testing, demo, or release purposes

# Agenda

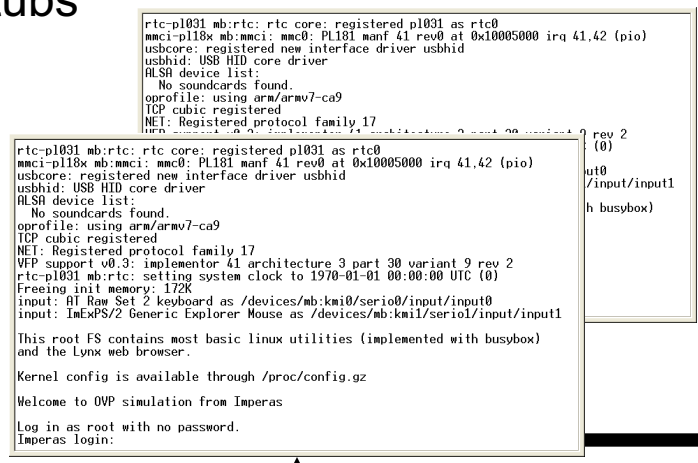
MULTICORE DESIGN SIMPLIFIED
imperas

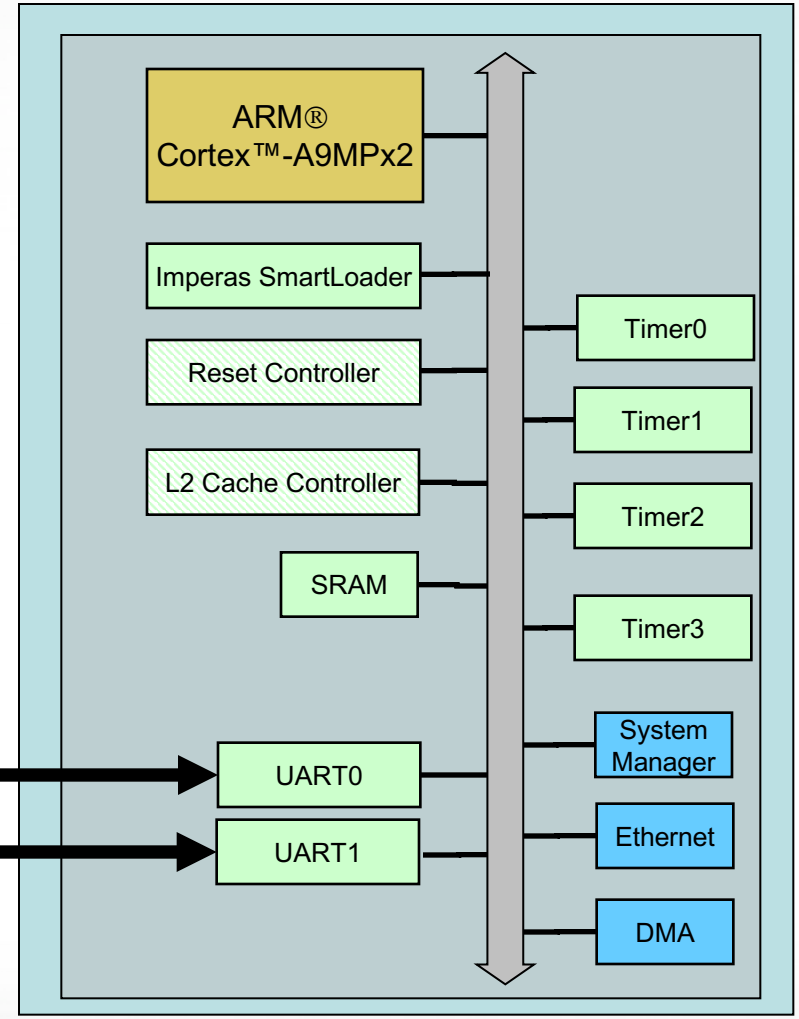
- New challenges posed by heterogeneous architectures
- Comparison of hardware-based and Virtual Platform-based methodologies
- Continuous Integration and Virtual Platforms
- Case study: Linux bring up and testing on Altera Cyclone V SoC FPGA
- Demonstration

# Altera Cyclone V SoC FPGA with ARM Cortex-A9MPx2

MULTICORE DESIGN SIMPLIFIED
imperas

- Green background peripheral models fully functional
- Green diagonal pattern background have only the functionality necessary to boot the operating systems
- Blue background peripheral models just stubs



# Linux Bring Up and Testing on Altera Cyclone V SoC FPGA

MULTICORE DESIGN SIMPLIFIED
imperas

1) Linux boot on single core ARM Cortex-A9 (minimal peripheral models)
2) SMP Linux boot on dual core ARM Cortex-A9 (minimal peripheral models)
3) Add in peripheral models for Cyclone V SoC FPGA

- Need to set up test infrastructure such that Continuous Integration (CI) testing can be performed

# Cyclone V SoC FPGA Virtual Platform

MULTICORE DESIGN SIMPLIFIED
imperas

- Top level virtual platform built using Open Virtual Platforms (OVP, www.OVPworld.org) platform API
- ARM Cortex-A9MPx2 processor core model from the OVP Library
- Peripheral models
  - Some models available in the OVP Library
  - Remaining models of peripheral components developed using OVP APIs
- OVP APIs written for C language
- Simulation engine: Imperas M*SDK

- All OVP processor and peripheral models include both native OVP and native SystemC/TLM2 interfaces, so all the following results could have been achieved using the OSCI SystemC simulator plus Imperas M*SDK product
  - Peripheral models could have been written in SystemC
  - M*SDK tools require OVP processor core models to enable tools

# 1a) Linux Boot on Single Core ARM Cortex-A9

MULTICORE DESIGN SIMPLIFIED
imperas

- Use Linux from Altera:  Altera-3.4
- Use default configurations
- Use default device trees
  - Comment out a few peripherals not yet modeled
- Bug found in Linux kernel preemptive scheduling
  - Running multiple applications under Linux part of standard Imperas bring up testing
  - Linux boots and runs, but does not switch tasks properly

- Approximately 2 weeks engineering effort to build virtual platform able to boot Linux
- Virtual platform boots Linux in under 5 sec on standard PC, Windows or Linux

# 1b) OS-Aware Tools Used to Find the Bug

MULTICORE DESIGN SIMPLIFIED
imperas

- Use OS tracing [task, execve, schedule, context, …] to trace at the OS level, not instruction level
- OS-aware tools debug in hours, once the bug was observed
- Simulation overhead due to OS-aware tools < 10%

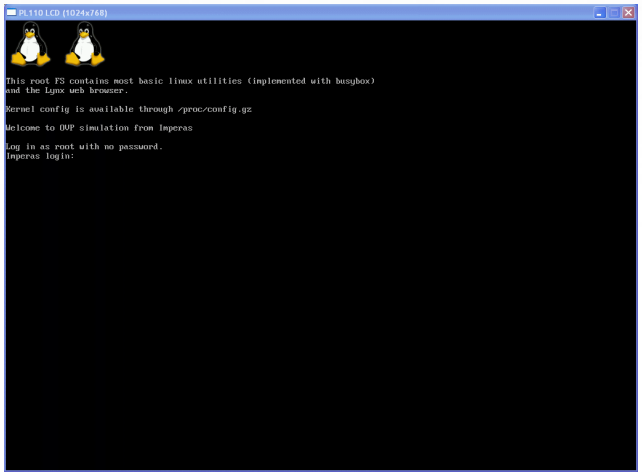
Imperas Multiprocessor Debugger

```
TRC (TASK) 810691893: 'cpu_CPU0': free_pid called for pid=412 ('/sbin/mdev')
TRC (SCHD) 810749566: 'cpu_CPU0': scheduler switched from process 405 ('/bin/sh') to 3 ('ksoftirqd/0')
TRC (SCHD) 810753989: 'cpu_CPU0': scheduler switched from process 3 ('ksoftirqd/0') to 413 ('rcS')
TRC (TASK) 810805210: 'cpu_CPU0': do_execve called for pid=413 ('/bin/hostname')
TRC (EXEC) 810805210: 'cpu_CPU0': do_execve called for pid=413 with filename=/bin/hostname with:
TRC (EXEC) 810805210: 'cpu_CPU0':        argv virt=0x000dd24c "hostname"
TRC (EXEC) 810805210: 'cpu_CPU0':        argv virt=0x000dd264 "Imperas"
TRC (EXEC) 810805210: 'cpu_CPU0':        envp virt=0x7e943fe2 "USER=root"
TRC (EXEC) 810805210: 'cpu_CPU0':        envp virt=0x7e943f9f "HOME=/"
TRC (EXEC) 810805210: 'cpu_CPU0':        envp virt=0x7e943fa6 "TERM=vt102"
TRC (EXEC) 810805210: 'cpu_CPU0':        envp virt=0x7e943fb1 "PATH=/sbin:/usr/sbin:/bin:/usr/bin"
TRC (EXEC) 810805210: 'cpu_CPU0':        envp virt=0x7e943fd4 "SHELL=/bin/sh"
TRC (EXEC) 810805210: 'cpu_CPU0':        envp virt=0x000dd778 "PWD=/"
TRC (TASK) 810822171: 'cpu_CPU0': load_elf_binary('/bin/hostname') called for pid=413
TRC (TASK) 811285592: 'cpu_CPU0': do_exit called for pid=413 ('/bin/hostname')
TRC (SCHD) 811324531: 'cpu_CPU0': scheduler switched from process 413 ('/bin/hostname') to 405 ('/bin/sh')
TRC (TASK) 811326827: 'cpu_CPU0': free_pid called for pid=413 ('/bin/hostname')
TRC (TASK) 811341466: 'cpu_CPU0': do_exit called for pid=405 ('/bin/sh')
TRC (SCHD) 811388059: 'cpu_CPU0': scheduler switched from process 405 ('/bin/sh') to 3 ('ksoftirqd/0')
TRC (SCHD) 811397630: 'cpu_CPU0': scheduler switched from process 3 ('ksoftirqd/0') to 1 ('/init')
TRC (TASK) 811403745: 'cpu_CPU0': free_pid called for pid=405 ('/bin/sh')
TRC (SCHD) 811411242: 'cpu_CPU0': scheduler switched from process 1 ('/init') to 3 ('ksoftirqd/0')
TRC (SCHD) 811413283: 'cpu_CPU0': scheduler switched from process 3 ('ksoftirqd/0') to 414 ('init')
TRC (TASK) 811451482: 'cpu_CPU0': do_execve called for pid=414 ('/sbin/getty')
TRC (EXEC) 811451482: 'cpu_CPU0': do_execve called for pid=414 with filename=/sbin/getty with:
TRC (EXEC) 811451482: 'cpu_CPU0':        argv virt=0x7e8fdb24 "/sbin/getty"
TRC (EXEC) 811451482: 'cpu_CPU0':        argv virt=0x7e8fdb30 "-L"
TRC (EXEC) 811451482: 'cpu_CPU0':        argv virt=0x7e8fdb33 "38400"
TRC (EXEC) 811451482: 'cpu_CPU0':        argv virt=0x7e8fdb39 "ttyS0"
TRC (EXEC) 811451482: 'cpu_CPU0':        envp[] = virt=0x000dd008 <not in TLB>
TRC (TASK) 811469113: 'cpu_CPU0': load_elf_binary('/sbin/getty') called for pid=414
TRC (SCHD) 811480188: 'cpu_CPU0': scheduler switched from process 414 ('/sbin/getty') to 3 ('ksoftirqd/0')
TRC (SCHD) 811482266: 'cpu_CPU0': scheduler switched from process 3 ('ksoftirqd/0') to 1 ('/init')
TRC (SCHD) 811485826: 'cpu_CPU0': scheduler switched from process 1 ('/init') to 414 ('/sbin/getty')
TRC (SCHD) 812001979: 'cpu_CPU0': scheduler switched from process 414 ('/sbin/getty') to 3 ('ksoftirqd/0')
TRC (SCHD) 812006285: 'cpu_CPU0': scheduler switched from process 3 ('ksoftirqd/0') to 0 ('swapper')
TRC (SCHD) 824001960: 'cpu_CPU0': scheduler switched from process 0 ('swapper') to 1 ('/init')
```

# OS-Aware Software Analysis Example: OS Task Tracing

MULTICORE DESIGN SIMPLIFIED
imperas

- ✓ Non-intrusive: no instrumentation or modification of source code
- ✓ Multicore capable




*i*

```
do_execve: pid=19
                filename=/sbin/hotplug
                argv virt=0x804613f0 phys=0x004613f0 "/sbin/hotplug"
                argv virt=0x8045bc34 phys=0x0045bc34 "module"
                envp virt=0x80413500 phys=0x00413500 "HOME=/"
                envp virt=0x804170b4 phys=0x004170b4 "PATH=/sbin:/bin:
                envp virt=0x81150000 phys=0x01150000 "ACTION=add"
                envp virt=0x8115000b phys=0x0115000b "DEVPATH=/mod
                envp virt=0x81150024 phys=0x01150024 "SUBSYSTEM=m
                envp virt=0x81150035 phys=0x01150035 "SEQNUM=13"
do_execve: pid=20
                filename=/sbin/hotplug
                argv virt=0x804613f0 phys=0x004613f0 "/sbin/hotplug"
                argv virt=0x8045bc34 phys=0x0045bc34 "module"
                envp virt=0x80413500 phys=0x00413500 "HOME=/"
                envp virt=0x804170b4 phys=0x004170b4 "PATH=/sbin:/bin:/
                envp virt=0x81150000 phys=0x01150000 "ACTION=add"
                envp virt=0x8115000b phys=0x0115000b "DEVPATH=/modu
                .......
```

**Introspection (Linux OS)**

1) OS-aware tools enable in-depth monitoring and analysis, even before console is available
2) Provides tracing at appropriate levels of abstraction, granularity
   - ~ 1,000,000,000 instructions to boot SMP Linux: instruction tracing to find OS problem would be painfully slow and complicated
   - ~ 700 tasks to boot Linux: task tracing provides better starting point for debugging OS problems during bring up

# OS-Aware Software Analysis Example: OS Scheduler Tracing

MULTICORE DESIGN SIMPLIFIED
imperas

GTKWave - LinuxKernel.vcd

File Edit Search Time Markers View Help

From: 25863744 ns To: 20971773150 Marker: -- | Cursor: 270 ms

SST
capture

Signals
20hal_2127
20hal_2147
20hal_2356
20hal_2359
K11atd_2329
K11atd_2330
K11atd_2331
K11atd_2333
K11atd_2334
K11cron_2342
K20dbus_2354
K20exim4_2368
K20exim4_2369
K20exim4_2371
K20exim4_2373
K20makedev_2376
K20openbsd-inet_2377
K20openbsd-inet_2379

Filter:

Append Insert Replace

Signals
Time
\matrixPthread.e_2302
\matrixPthread.e_2303
\matrixPthread.e_2304
\matrixPthread.e_2305
\matrixPthread.e_2306
\matrixPthread.e_2307
events/0_5
gcc_2297
ifup_989
ifup_1841
init_1
init_745
init_2073
kbd_mode_1620
kblockd/0_35
khelper_6
kjournald_722
klogd_2104
klogd_2105
kpsmoused_716
kseriod_39
ksoftirqd/0_4
kswapd0_71
kthreadd_2
pkill_2403
portmap_1861
portmap_1862
ps_993
rc_748
shutdown_2309
swapon_1670
swapon_1796
swapper_0
sysctl_1785
syslogd_2098
syslogd_2099
vi_2295

Waves
10 sec

- Process creation
- Process deletion
- Context switching
- Non-intrusive
- Multicore capable

# 2a) SMP Linux Boot on Dual Core ARM Cortex-A9

MULTICORE DESIGN SIMPLIFIED
imperas

- Use Linux from Altera:  Altera-3.6
- Use default configurations
- Use default device trees
  - Comment out the peripherals not yet modeled

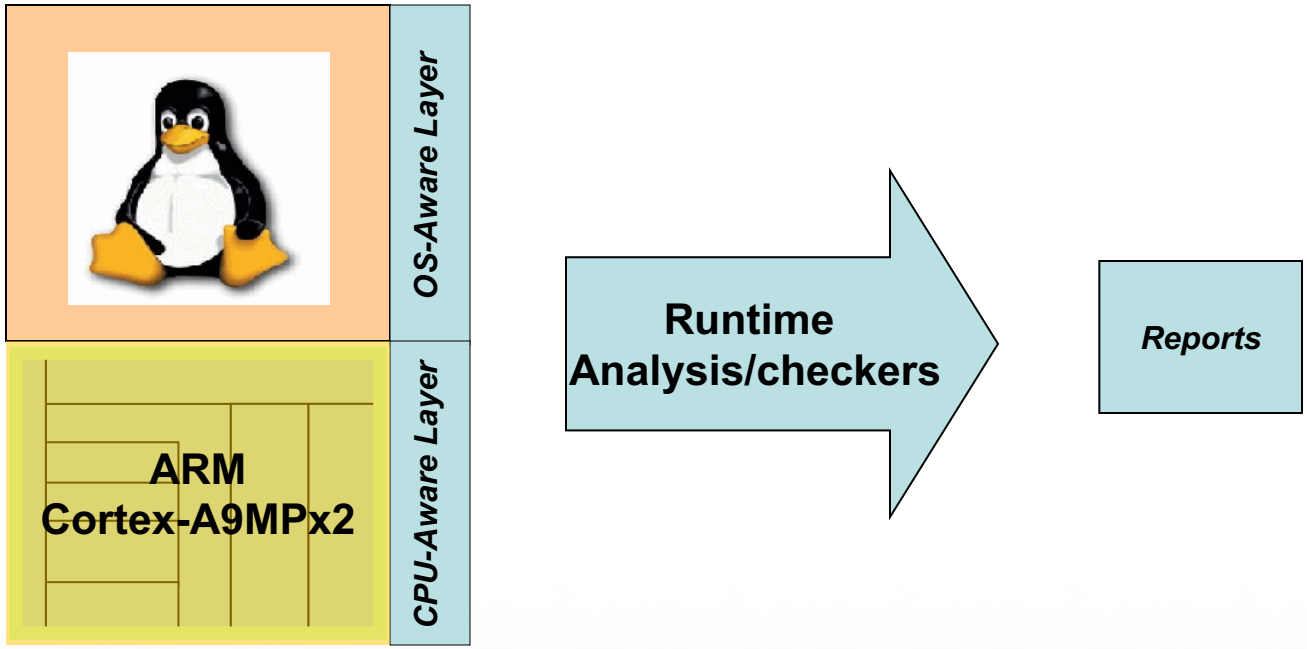
- Bug found in Linux accesses of GIC registers
- Virtual platform debug took 2 days versus 2 weeks on hardware platform (5x improvement)

- Also need to ensure that operating systems do not access forbidden memory segments

# Callbacks on events

MULTICORE DESIGN SIMPLIFIED
imperas

- Non-intrusive trace/callback of
  - Selected changes/events in the hardware of system
  - Selected events in OS/software
- Add C code to monitor and check what has happened – add protocols, rules, assertions



# 2b) Custom Memory Access Monitor Accelerates Platform Debug

MULTICORE DESIGN SIMPLIFIED
imperas

- Memory access monitor is just C code, less than 350 lines, loaded into simulation environment
- When simulation is run, monitor produces warning if memory access rules are violated

```
//
// Define watch areas for memory and peripherals defined in the platform
//
memWatchT amcWatch[] = {
//  name                          watchLow        watchHigh        allowedCPUs
    { "Linux memory",             0,              0x2fffffff,      LINUX_CPU   },
    { "gmac0",                    0xff700000,     0xff700fff,      LINUX_CPU   },
    { "emac0_dma",                0xff701000,     0xff701fff,      LINUX_CPU   },
    { "gmac1",                    0xff702000,     0xff702fff,      LINUX_CPU   },
    { "emac1_dma",                0xff703000,     0xff703fff,      LINUX_CPU   },
    { "uart0",                    0xffc02000,     0xffc02fff,      LINUX_CPU   },
    { "CLKMGR",                   0xffd04000,     0xffd04fff,      LINUX_CPU   },
    { "RSTMGR",                   0xffd05000,     0xffd05fff,      LINUX_CPU   },
    { "SYSMGR",                   0xffd08000,     0xffd08fff,      LINUX_CPU   },
    { "GIC",                      0xfffec000,     0xfffedfff,      LINUX_CPU   },
    { "L2",                       0xfffef000,     0xfffeffff,      LINUX_CPU   },
    { 0 } /* Marks end of list */
};
```

```
Warning (AMPCHK_MWV) LINUX_CPU: AMP write access violation in uart1 area. PA: 0xffc03008 VA: 0xffc03008
Warning (AMPCHK_MWV) LINUX_CPU: AMP write access violation in uart1 area. PA: 0xffc0300c VA: 0xffc0300c
Warning (AMPCHK_MWV) LINUX_CPU: AMP write access violation in uart1 area. PA: 0xffc03010 VA: 0xffc03010
```

# 2b-2) Custom Memory Access Monitor Accelerates Platform Debug (2nd CPU)

MULTICORE DESIGN SIMPLIFIED
imperas

- Memory access monitor is just C code, less than 350 lines, loaded into simulation environment
- When simulation is run, monitor produces warning if memory access rules are violated

```
//
// Define watch areas for memory and peripherals defined in the platform
//
memWatchT amcWatch[] = {
//  name                          watchLow        watchHigh        allowedCPUs
    { "Linux memory",             0,              0x2fffffff,      LINUX_CPU   },
    { "CPU2 memory",              0x30000000,     0x31ffffff,      CPU2_CPU  },
    { "gmac0",                    0xff700000,     0xff700fff,      LINUX_CPU   },
    { "emac0_dma",                0xff701000,     0xff701fff,      LINUX_CPU   },
    { "gmac1",                    0xff702000,     0xff702fff,      LINUX_CPU   },
    { "emac1_dma",                0xff703000,     0xff703fff,      LINUX_CPU   },
    { "uart0",                    0xffc02000,     0xffc02fff,      LINUX_CPU   },
    { "uart1",                    0xffc03000,     0xffc03fff,      CPU2_CPU  },
    { "CLKMGR",                   0xffd04000,     0xffd04fff,      LINUX_CPU   },
    { "RSTMGR",                   0xffd05000,     0xffd05fff,      LINUX_CPU   },
    { "SYSMGR",                   0xffd08000,     0xffd08fff,      LINUX_CPU   },
    { "GIC",                      0xfffec000,     0xfffedfff,      LINUX_CPU   },
    { "L2",                       0xfffef000,     0xfffeffff,      LINUX_CPU   },
    { 0 } /* Marks end of list */
};
```

Warning (AMPCHK_MWV) LINUX_CPU: AMP write access violation in uart1 area. PA: 0xffc03008 VA: 0xffc03008
Warning (AMPCHK_MWV) LINUX_CPU: AMP write access violation in uart1 area. PA: 0xffc0300c VA: 0xffc0300c
Warning (AMPCHK_MWV) LINUX_CPU: AMP write access violation in uart1 area. PA: 0xffc03010 VA: 0xffc03010
**Warning (AMPCHK_MRV) CPU2_CPU: AMP read access violation in Linux memory area. PA: 0x00000020 VA: 0x00000020**

# Agenda

MULTICORE DESIGN SIMPLIFIED
imperas

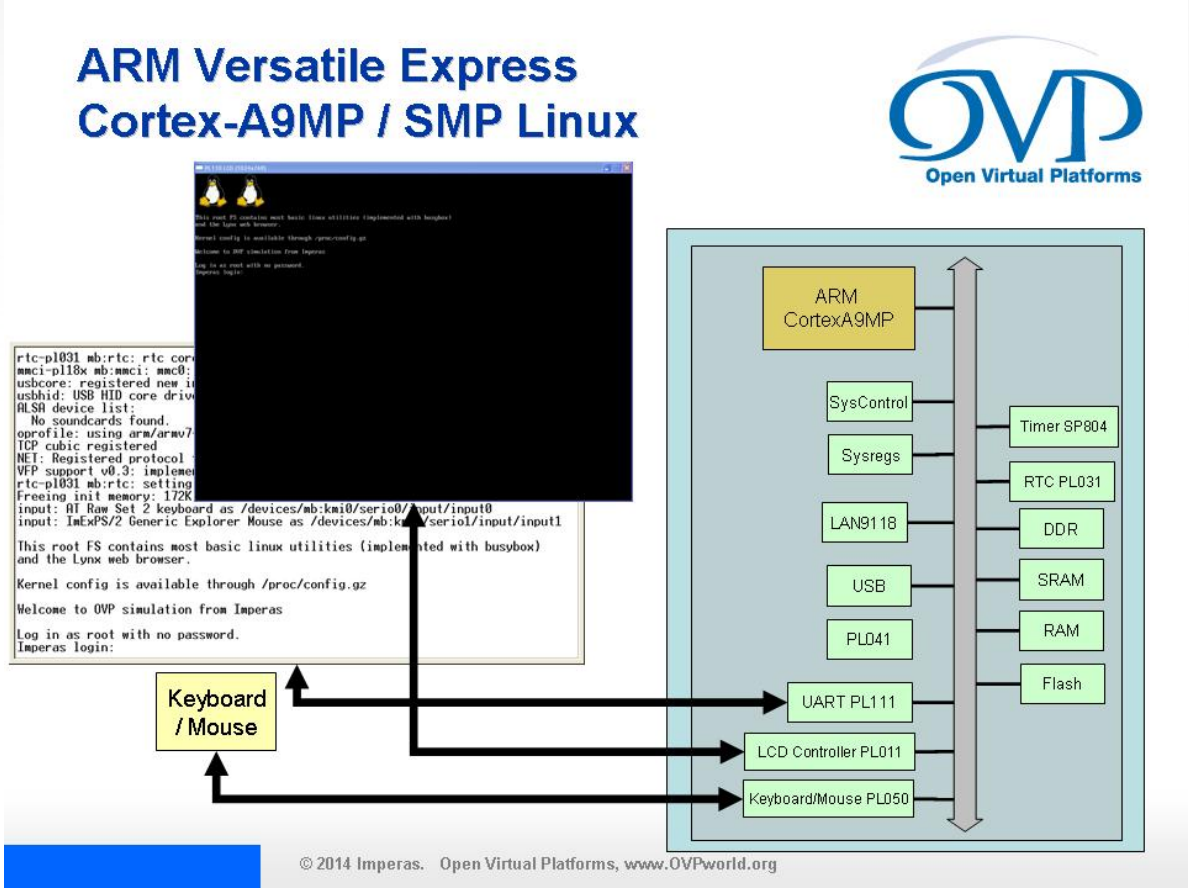
- New challenges posed by heterogeneous architectures
- Comparison of hardware-based and Virtual Platform-based methodologies
- Continuous Integration and Virtual Platforms
- Case study: Linux bring up and testing on Altera Cyclone V SoC FPGA
- Demonstration

# Demonstrations

MULTICORE DESIGN SIMPLIFIED
imperas

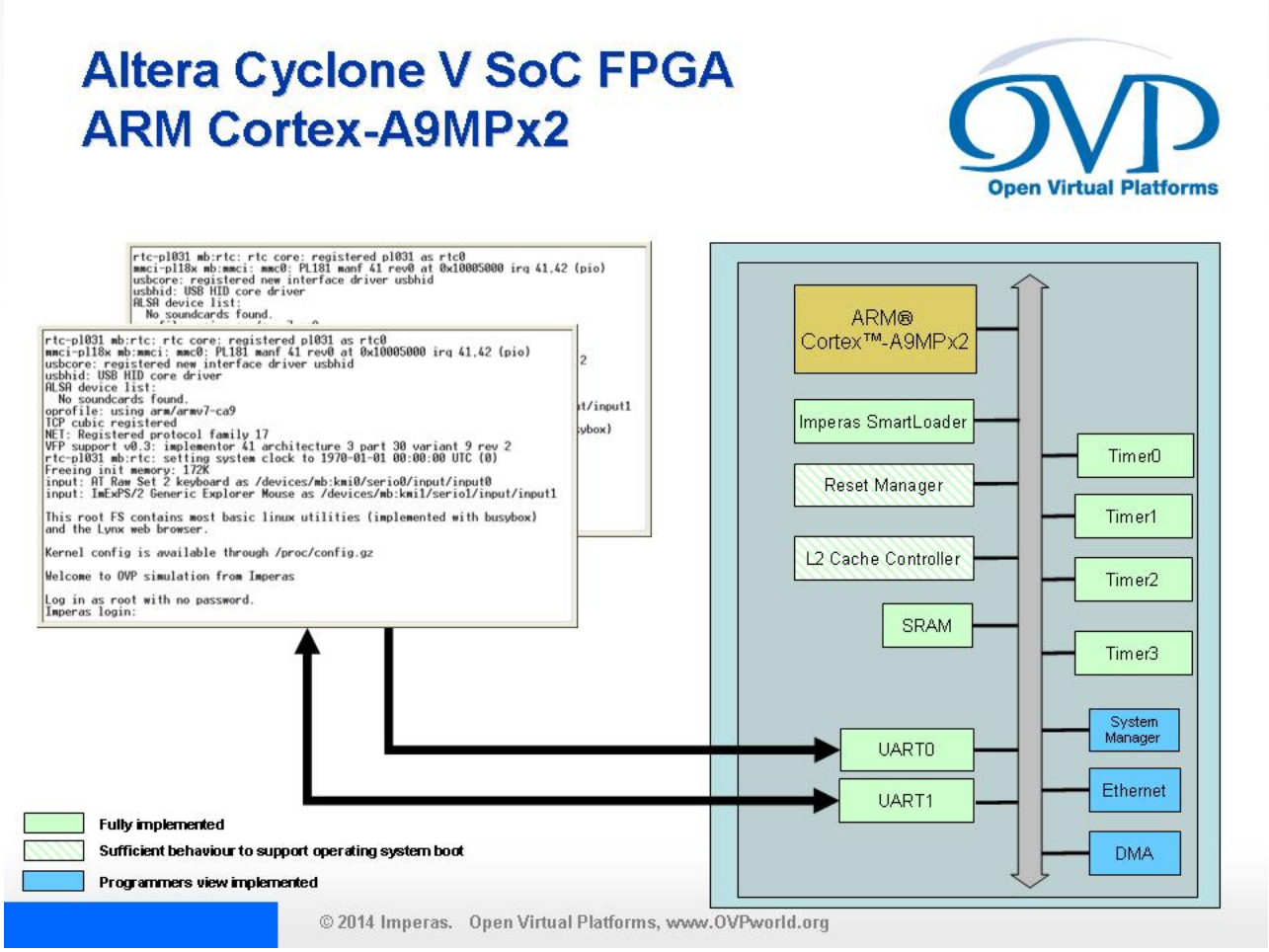
- Linux boot on single ARM Cortex-A9
- Linux boot on Altera Cyclone V
- Linux boot on multicore MIPS I6400
- SMP Linux boot on single core ARM Cortex-A9
  - OS-aware tools
- Memory Monitoring

# Linux boot on single ARM Cortex-A9




Run

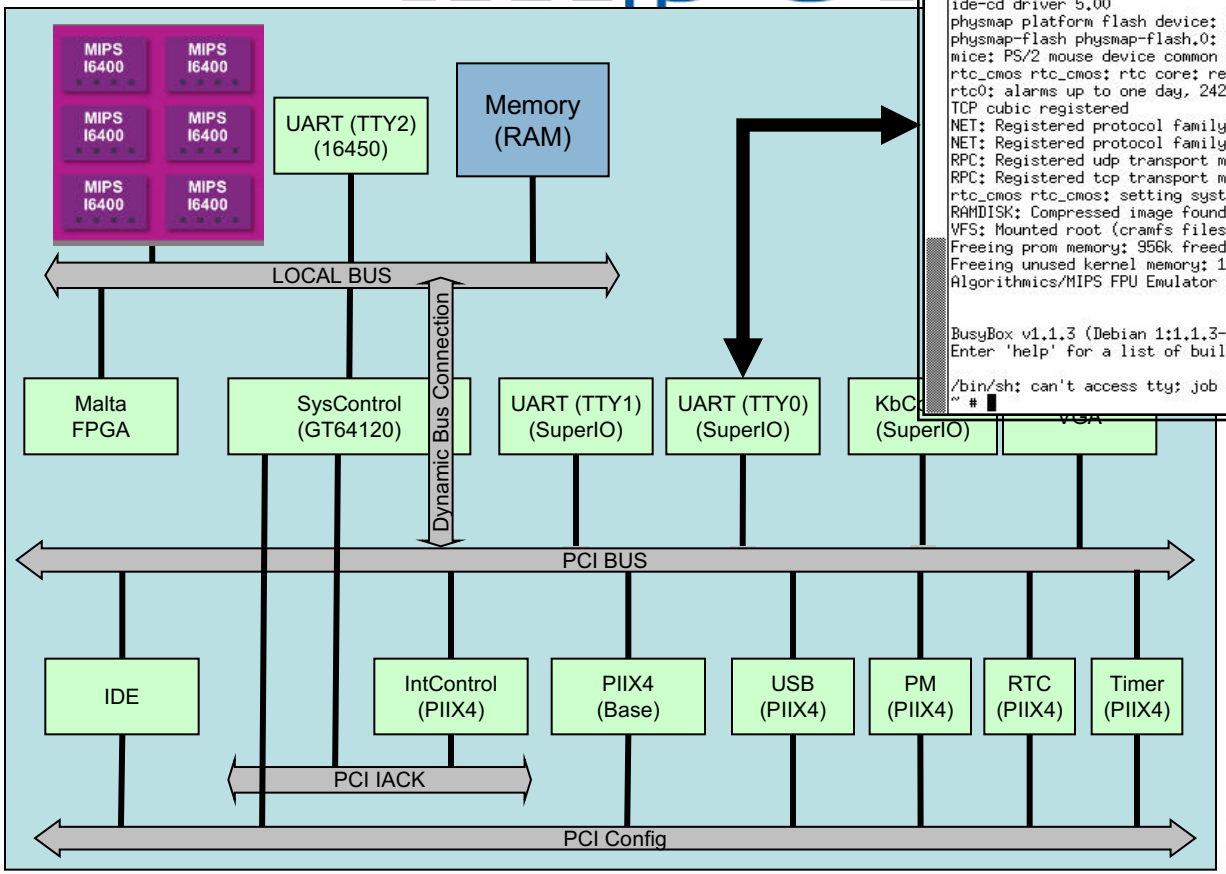
# Linux boot on Altera Cyclone V




Run

# MIPS I6400
# Virtual Platform / Linux

MULTICORE DESIGN SIMPLIFIED
imper



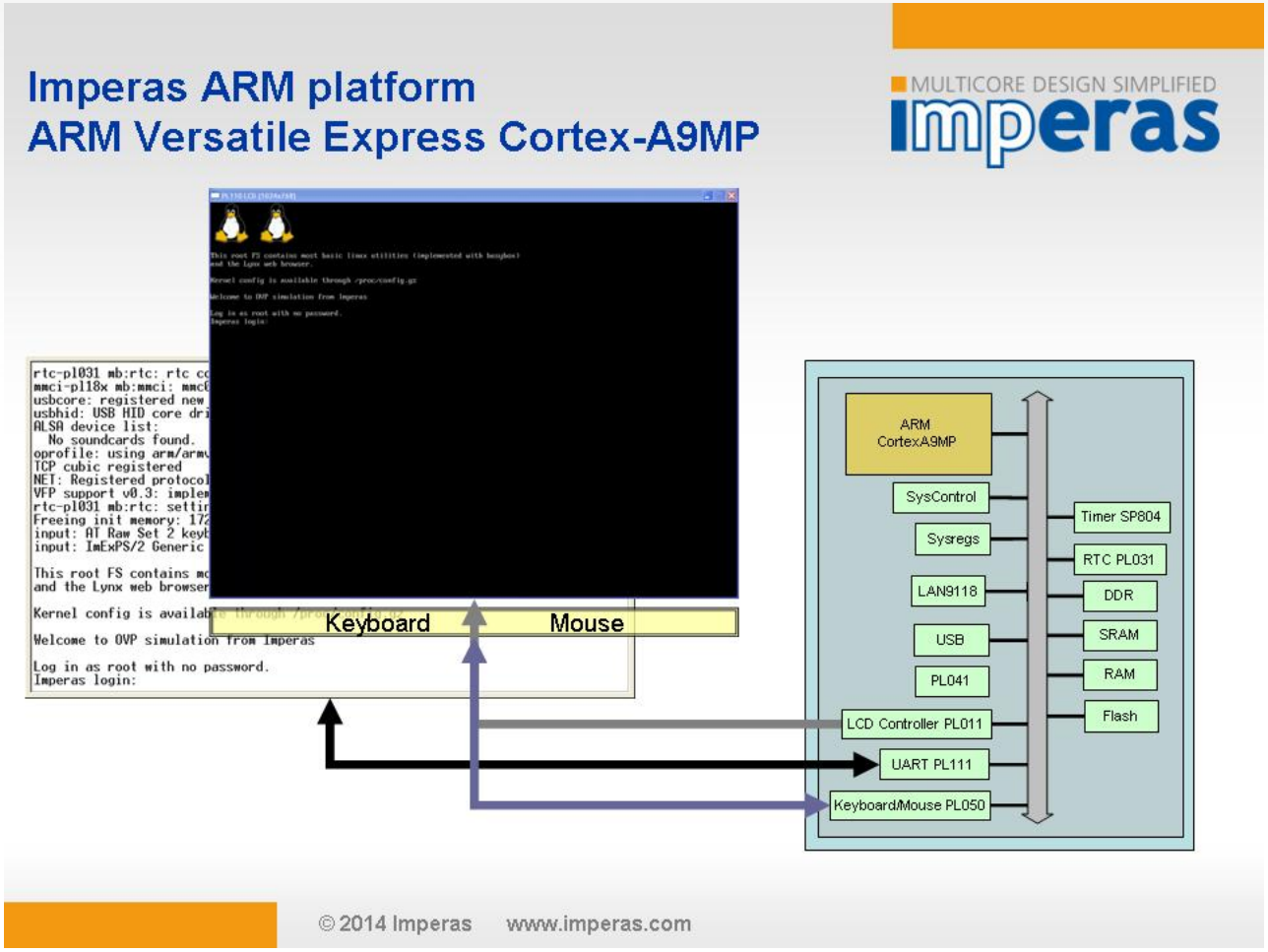

```
piix 0000:00:0a.1: IDE controller (0x8086:0x7111 rev 0x00)
PCI: Enabling device 0000:00:0a.1 (0000 -> 0001)
piix 0000:00:0a.1: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0x1040-0x1047
    ide1: BM-DMA at 0x1048-0x104f
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
ide_generic: please use "probe_mask=0x3f" module parameter for probing all legacy ISA IDE ports
ide-gd driver 1.18
ide-cd driver 5.00
physmap platform flash device: 00400000 at 1e000000
physmap-flash physmap-flash.0: map_probe failed
mice: PS/2 mouse device common for all mice
rtc_cmos rtc_cmos: rtc core: registered rtc_cmos as rtc0
rtc0: alarms up to one day, 242 bytes nvram
TCP cubic registered
NET: Registered protocol family 17
NET: Registered protocol family 15
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
rtc_cmos rtc_cmos: setting system clock to 2008-05-25 12:12:15 UTC (1211717535)
RAMDISK: Compressed image found at block 0
VFS: Mounted root (cramfs filesystem) readonly on device 1:0.
Freeing prom memory: 956k freed
Freeing unused kernel memory: 132k freed
Algorithmics/MIPS FPU Emulator v1.5


BusyBox v1.1.3 (Debian 1:1.1.3-3) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

/bin/sh: can't access tty; job control turned off
~ #
```

run

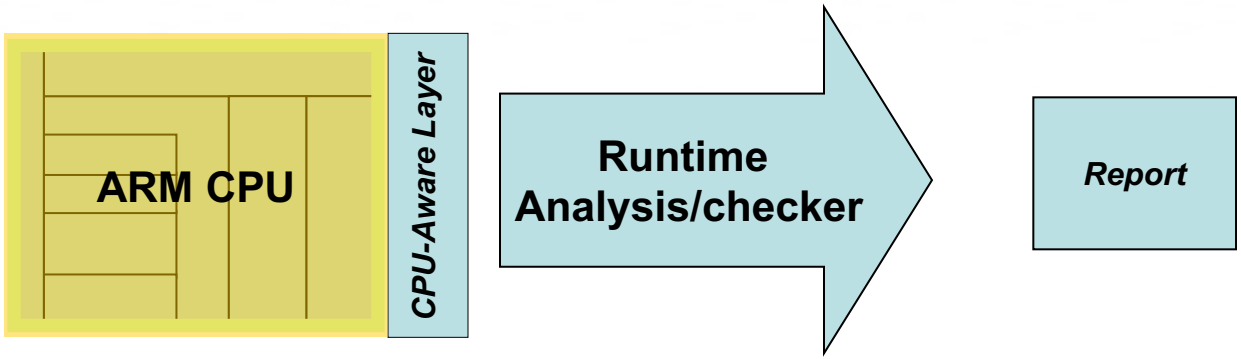
# Linux OS-Aware tools




Run
PrintKernel
TraceTasks

# Memory Monitor

- Non-intrusive trace/callback of
  - Selected changes/events in the hardware of system
  - Selected events in OS/software




Run

# Virtual Platform Adoption*

MULTICORE DESIGN SIMPLIFIED
imperas

- As systems become more complex, organizations are turning to modeling and simulation tools to improve their software development environments
- Virtual platform solutions are being adopted as a mechanism to improve system quality and to accelerate software development and testing
- Engineering teams whose projects align with Agile and Continuous Integration (CI) product development methodologies are more likely to use virtual platform solutions

⇒ If you need to build complex products with high quality in shorter schedules you need to adopt virtual platform based solutions

*** Trends from VDCresearch reports 2014**

# Summary

MULTICORE DESIGN SIMPLIFIED
imperas

- Virtual platforms – software simulation – provide a complementary technology to hardware-based testing of software
- Linux bring up on virtual platforms should be done incrementally
  - Minimizing platform degrees of freedom adds productivity
- OS-aware tools provide additional productivity, efficiency
- Custom tools provide more robust software test environment

MULTICORE DESIGN SIMPLIFIED

imperas

Thank you

www.imperas.com

www.OVPworld.org