

OS Porting & Analysis for Dual Core ARM Cortex-A9 Based Systems

Simon Davidmann

Imperas

31 October 2012

Agenda

- Silicon without software is just sand...
 - Issues in embedded software development
 - OS porting, analysis and bring up
- What is a virtual platform?
 - Building a virtual platform
 - Requirements for a virtual platform development environment
- Case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

Agenda

- Silicon without software is just sand...
 - Issues in embedded software development
 - OS porting, analysis and bring up
- What is a virtual platform?
- Case studies for virtual platform based software development
- Summary, Q&A

Silicon Without Software Is Just Sand



Photo courtesy of Fujitsu



- ✓ Software development is costing more than chip development cost
- ✓ Embedded software is the critical path to system delivery
- ✓ Source code is doubling annually
- ✓ Software complexity is increasing dramatically with multi-core devices, multi-processor systems
- ✓ Products are defined by their software

Issues in Embedded Software Development

- Quality is critical
- Current development methodology breaks with increasing code complexity
- Time to market still counts!
- Management cannot manage the software development process: insufficient metrics
 - *You cannot manage what you cannot measure*

Focus for Today's Presentation: Software Development, Porting, Bring Up

- Current development methodologies use hardware or host development systems
 - Actual hardware
 - Prototypes
 - x86 based development
- These methods lack controllability, visibility, accuracy
 - Controllability: can you test all relevant scenarios?
 - Visibility: if an error occurs, will it be observed by the test environment?
 - Accuracy: will software developed on x86 behave the same on an ARM-based device?
- Virtual platforms – software simulation – provide a complementary technology to the current methodology
 - Instruction accurate simulation promises controllability, visibility, ARM behavior
 - How to deliver on this promise?

Software Failures in Embedded Systems Are Bad!

This Car Runs on Code

February 5, 2010



The avionics system in the F-22 Raptor, the current U.S. Air Force frontline jet fighter, consists of about 1.5 million lines of code. The F-35 Joint Strike Fighter, scheduled for delivery in 2010, will require about 5.7 million lines of code for its onboard systems. And Boeing's new 787 Dreamliner, scheduled to be delivered to customers in 2010, requires about 100 million lines of software code to operate its avionics and other systems.

These are impressive amounts of software code. "In a premium-class automobile recently, "it takes about 100 million lines of software code," says a senior manager of informatics at Technical University, Munich. "The amount of software in cars. All that software executes on microprocessor-based electronic control units throughout the body of your car."

<http://news.discovery.com/autos/toyota-recall-software>

FDA: Software Failures Responsible for 24% Of All Medical Device Recalls

June 20, 2012

Software failures were behind 24 percent of all the medical device recalls in 2011, according to data from the U.S. Food and Drug Administration, which said it is gearing up its labs to spend more time analyzing the quality and security of software-based medical instruments and equipment.

The FDA's Office of Science and Engineering Laboratories (OSEL) released the data in its 2011 Annual Report on June 15, amid reports of a compromise of a Web site used to distribute software updates for hospital respirators. The absence of solid architecture and "principled engineering practices" in software development affects a wide range of medical devices, with potentially life-threatening consequences, the Agency said.

There is growing evidence that software security and integrity is a growing problem in the medical field. In October 2011, for example, security researcher Barnaby Jack demonstrated a remote, wireless attack on an implantable insulin pump from the firm Medtronic.

https://threatpost.com/en_us/blogs/fda-software-failures-responsible-24-all-medical-device-recalls-062012



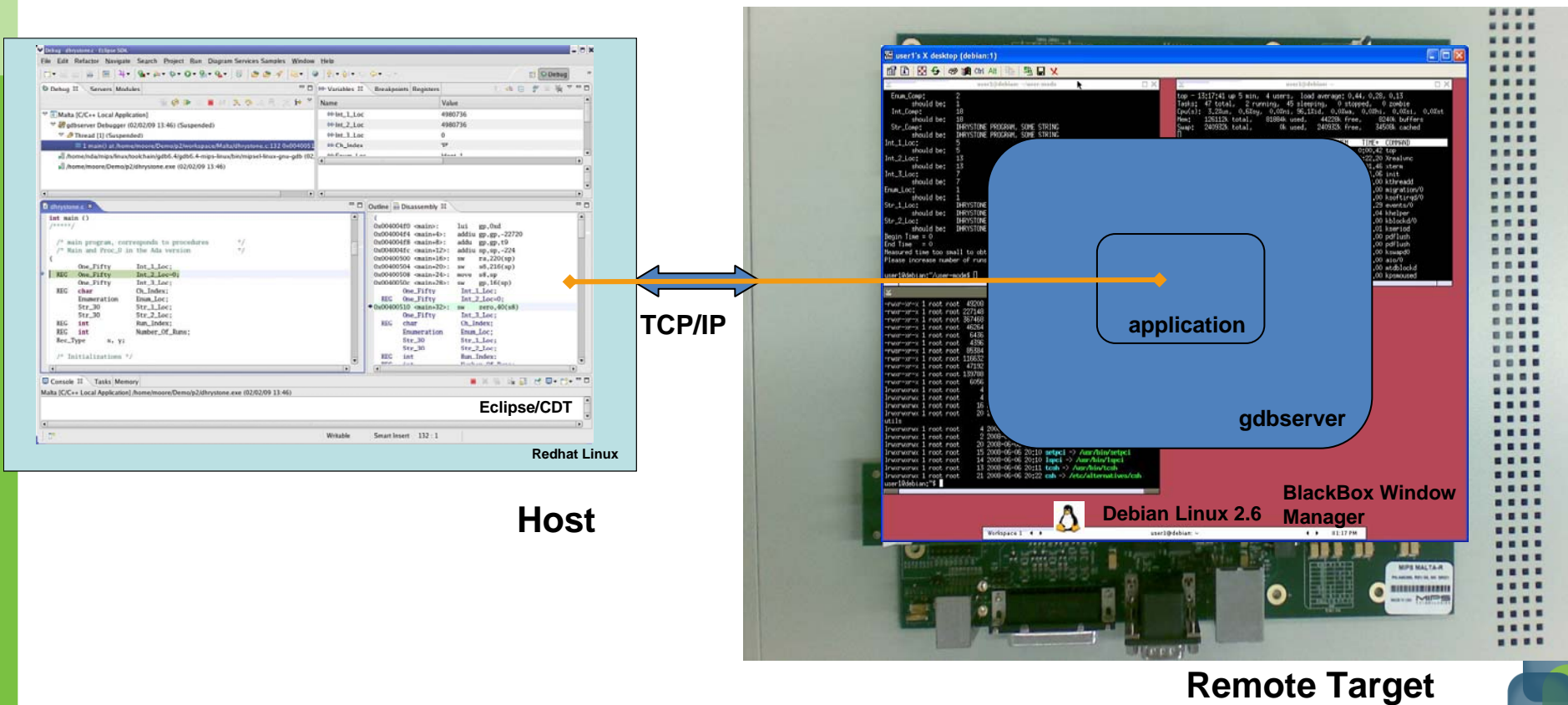
- **Systems are getting more complex**
- **Software failures can be life-threatening**
- **Software failures now include security breaches**

Agenda

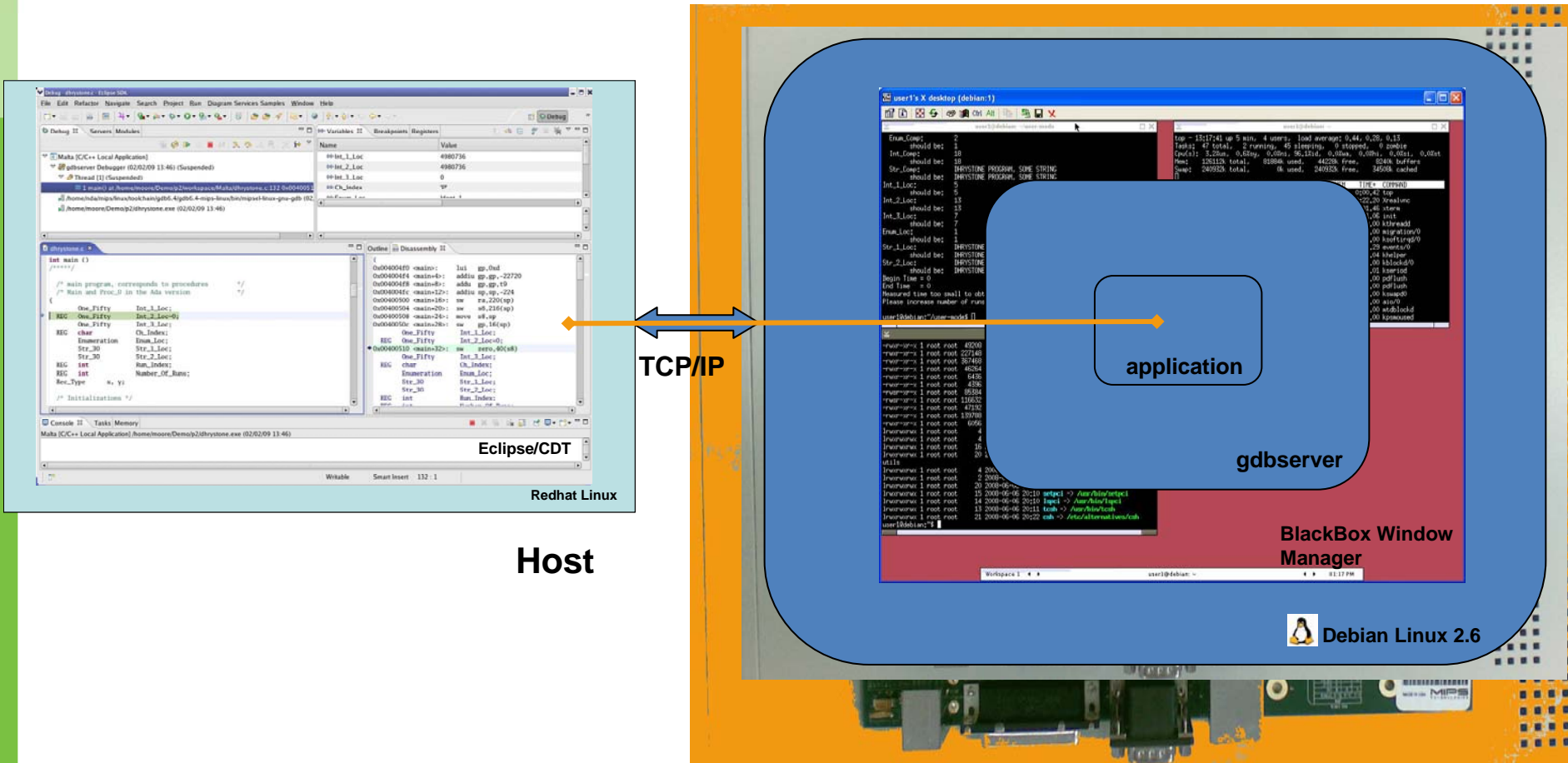
- Silicon without software is just sand...
- What is a virtual platform?
 - Building a virtual platform
 - Requirements for a virtual platform development environment
- Case studies for virtual platform based software development
- Summary, Q&A

Current Methodology, Software Debug on Prototype:

Run gdbserver on target and Eclipse on host to debug application on target



Using a Virtual Platform Provides Exactly the Same Environment (with many of the same limitations)



Virtual Platform as Remote Target

Building the Virtual Platform

- The virtual platform is a set of models that reflects the hardware on which the software will execute
 - Subset / subsystem of a single device
 - Processor chip
 - Board
 - System
- 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, ...
- Models can be cycle accurate, cycle approximate, or instruction accurate, with instruction accurate models providing the highest simulation performance

Instruction Accurate Virtual Platforms Run at 100s of MIPS

- To get the high speed required for real usage, processor hardware is modeled only to the minimum necessary level for *correct or plausible instruction behavior* so that software cannot tell it is not running on real hardware. Other features are approximated or omitted. Some examples:
 - Accurately modeled
 - Most instructions
 - Exceptions
 - Structures, such as TLBs, required to allow OS boot
 - Approximated
 - Tick timers – one “tick” per instruction
 - Random number generators (can affect, for example, TLB replacement algorithms)
 - Omitted
 - Instruction pipelines
 - Speculative execution
 - Write buffers
 - Caches (can be added; not modeled by default)
- General rule – if a feature cannot be modeled with reasonable accuracy, don’t model it at all (no bogus pretence of accuracy)

Open Virtual Platforms Provides the Modeling Infrastructure



- Website community/portal/forum
- Over 7,000 people registered on the website
- Modeling APIs for processor, peripheral, and platform modeling
- Open source library of models (many are Apache 2.0 open source license)
 - Fast Processor Models (100+ by end 2012): **ARM**, MIPS, Renesas, ...
 - Peripheral models: UART, timer, interrupt, ethernet, DMA, I/O, ...
 - Working platforms: Linux, Nucleus, μ C/OS II, bare metal applications, ...
 - OVP and SystemC/TLM2.0 native interfaces for all models

Website Lists all Available Models

Getting Started Documentation Demos & Videos Downloads Forums & Login Partners Contact



Open Virtual Platforms - the source of Fast Processor Models & Platforms

Home ▾ About ▾ Technology ▾ News ▾ Models ▾ Library Resources ▾

Quick Links

- Home
- About
- Technology
- News
- Models
 - Silicon IP Vendors
 - Processor Family Groups
 - Processor Model Variants
 - Processor Model Documentation
 - Processor Model Downloads
 - Getting FLEXIm License Keys
- Library
- Resources

Processor Model Variants

Processor Model Variant: ARM / Classic / ARM1020E

Processor Model Variant: ARM / Classic / ARM1022E

Processor Model Variant: ARM / Classic / ARM1026EJ-S

Processor Model Variant: ARM / Classic / ARM1136J-S

Processor Model Variant: ARM / Classic / ARM1156T2-S

Processor Model Variant: ARM / Classic / ARM1176JZ-S

Processor Model Variant: ARM / Classic / ARM720T

Processor Model Variant: ARM / Classic / ARM7EJ-S

Processor Model Variant: ARM / Classic / ARM7TDMI

Processor Model Variant: ARM / Classic / ARM920T

Processor Model Variant: ARM / Classic / ARM922T

Processor Model Variant: ARM / Classic / ARM926EJ-S

Processor Model Variant: ARM / Classic / ARM940T

Processor Model Variant: ARM / Classic / ARM946E

Processor Model Variant: ARM / Classic / ARM966E

Processor Model Variant: ARM / Classic / ARM968E-S

Processor Model Variant: ARM / Classic / ARMv4

Processor Model Variant: ARM / Classic / ARMv4T

Processor Model Variant: ARM / Classic / ARMv4TxM

In the News

- Imperas paper voted in top 5 at Cadence CDNIive users meeting
- Imperas tools & OVP Fast Processor Models validated with Cadence VSP
- More (30) ----

Press Releases

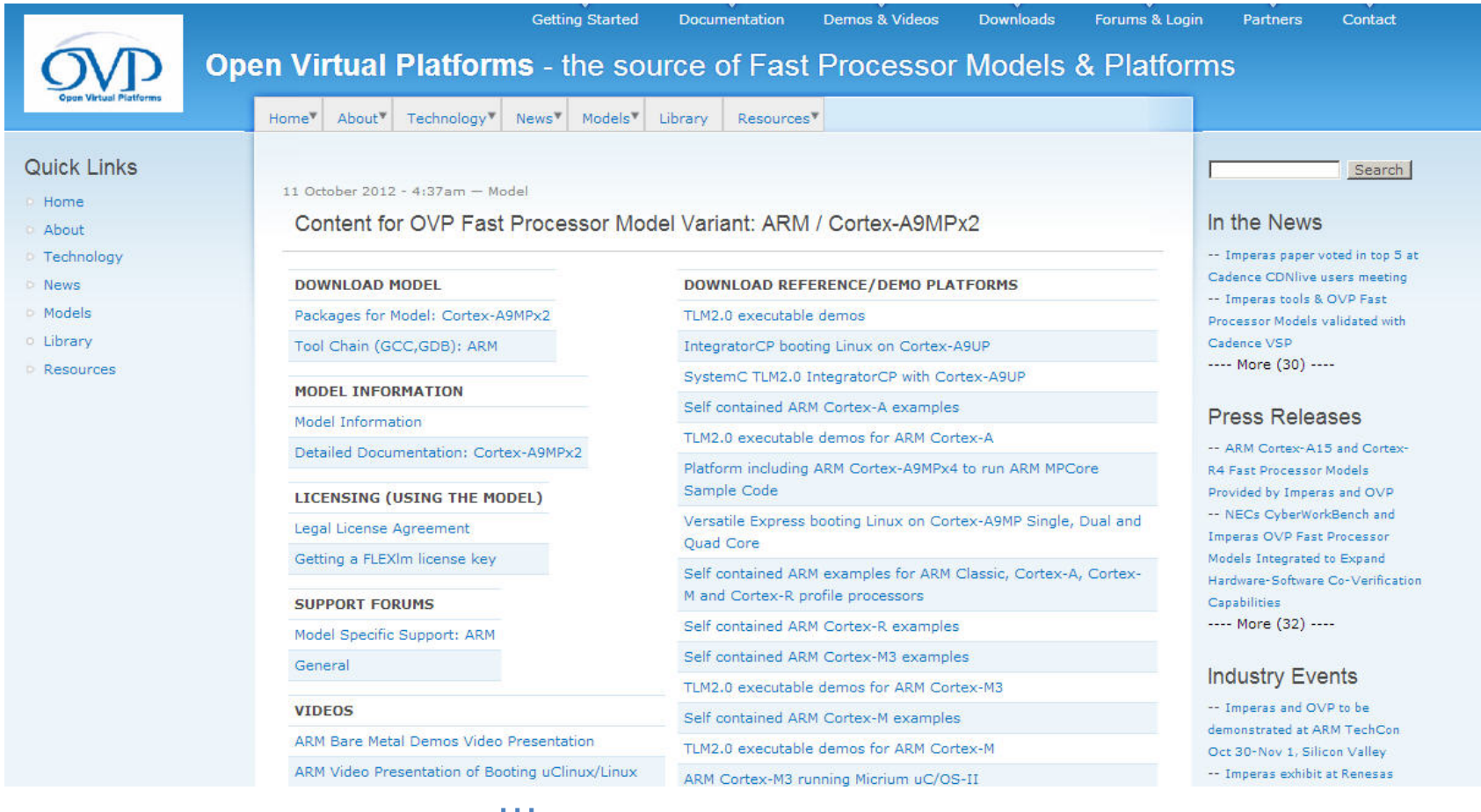
- ARM Cortex-A15 and Cortex-R4 Fast Processor Models Provided by Imperas and OVP
- NECs CyberWorkBench and Imperas OVP Fast Processor Models Integrated to Expand Hardware-Software Co-Verification Capabilities
- More (32) ----

Industry Events

- Imperas and OVP to be demonstrated at ARM TechCon Oct 30-Nov 1, Silicon Valley
- Imperas exhibit at Renesas

Website lists all available models (37 ARM + 17 ARM arch types)

Website Show Model Resources



The screenshot shows the Open Virtual Platforms (OVP) website interface. The main navigation bar includes links for Getting Started, Documentation, Demos & Videos, Downloads, Forums & Login, Partners, and Contact. The page title is "Open Virtual Platforms - the source of Fast Processor Models & Platforms". The breadcrumb trail is Home > About > Technology > News > Models > Library > Resources > . The main content area displays the date "11 October 2012 - 4:37am" and the title "Content for OVP Fast Processor Model Variant: ARM / Cortex-A9MPx2". The page is organized into several sections:

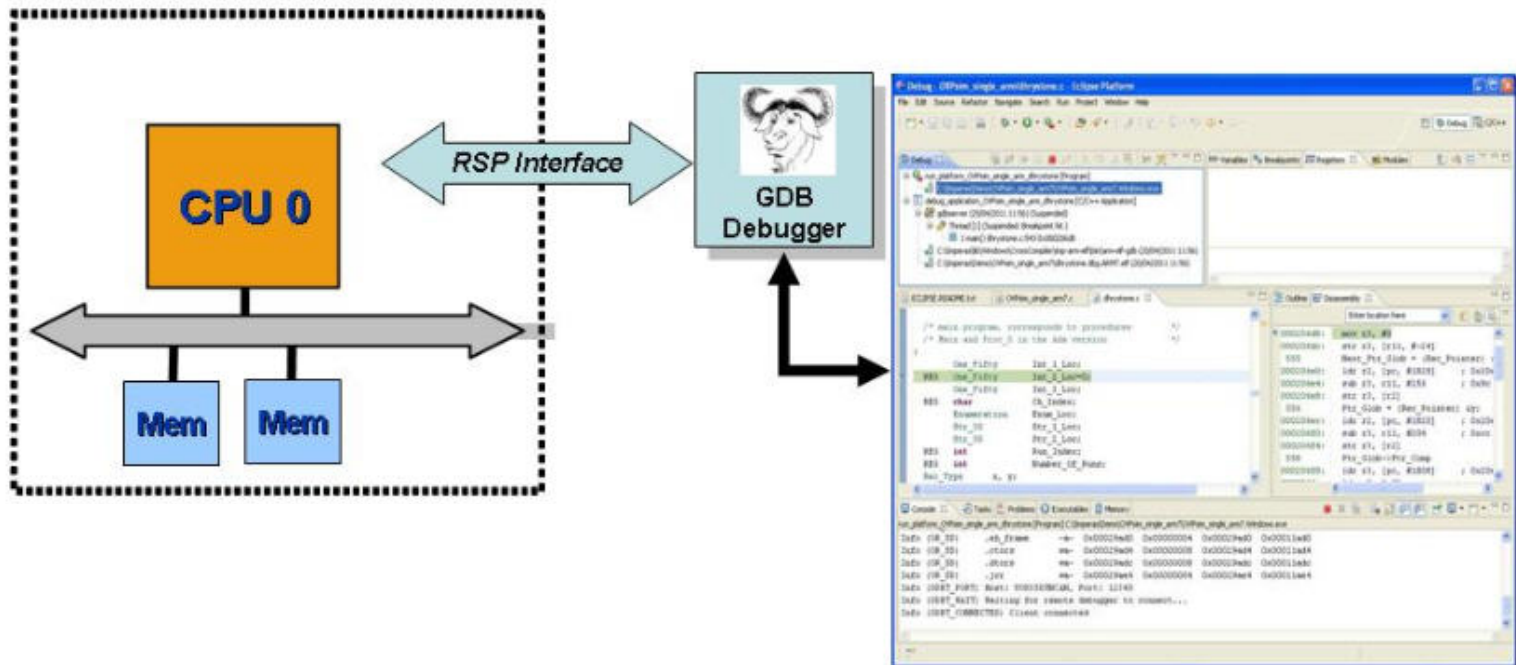
- DOWNLOAD MODEL:** Packages for Model: Cortex-A9MPx2, Tool Chain (GCC,GDB): ARM
- MODEL INFORMATION:** Model Information, Detailed Documentation: Cortex-A9MPx2
- LICENSING (USING THE MODEL):** Legal License Agreement, Getting a FLEXlm license key
- SUPPORT FORUMS:** Model Specific Support: ARM, General
- VIDEOS:** ARM Bare Metal Demos Video Presentation, ARM Video Presentation of Booting uClinux/Linux
- DOWNLOAD REFERENCE/DEMO PLATFORMS:** TLM2.0 executable demos, IntegratorCP booting Linux on Cortex-A9UP, SystemC TLM2.0 IntegratorCP with Cortex-A9UP, Self contained ARM Cortex-A examples, TLM2.0 executable demos for ARM Cortex-A, Platform including ARM Cortex-A9MPx4 to run ARM MPCore Sample Code, Versatile Express booting Linux on Cortex-A9MP Single, Dual and Quad Core, Self contained ARM examples for ARM Classic, Cortex-A, Cortex-M and Cortex-R profile processors, Self contained ARM Cortex-R examples, Self contained ARM Cortex-M3 examples, TLM2.0 executable demos for ARM Cortex-M3, Self contained ARM Cortex-M examples, TLM2.0 executable demos for ARM Cortex-M, ARM Cortex-M3 running Micrium uC/OS-II

On the right side, there is a search bar and sections for "In the News" (listing Imperas paper and Cadence CDNlive users meeting), "Press Releases" (listing ARM Cortex-A15 and Cortex-R4 Fast Processor Models), and "Industry Events" (listing Imperas and OVP demonstration at ARM TechCon).

For each specific model is a 'variant' page listing all relevant content: downloads, videos, presentations, documents

Open Virtual Platforms Provides a very fast simulator

- OVPsim™ simulator (models need the simulator to execute)
 - Runs processor models fast, 100s of mips
 - Interfaces to GDB via RSP
 - Encapsulation in Eclipse IDE for software and platform debug



Virtual Platform Requirements

- ✓ Performance near real time
- ✓ Run target binaries without change
- ✓ Repeatable results
- ✓ Multi-processor debug capability
- ✓ Software verification, analysis, profiling tools

```

Telnet 172.17.1.144
Serial channel initialized.

ARM MPCore Boot and Synchronization Example Code
Built Jun 24 2011 13:02:59

Number of CPUs: 4
Level 1 cache: ON
Level 2 cache: OFF
Snoop Control Unit: ON
Primary GIC: ON
Secondary GIC: ON

iinnffiinnittee && iinnffiinnittee && iinnffiinnittee && iinnffiinnittee &&
ee &&

CORTEXA9> pps

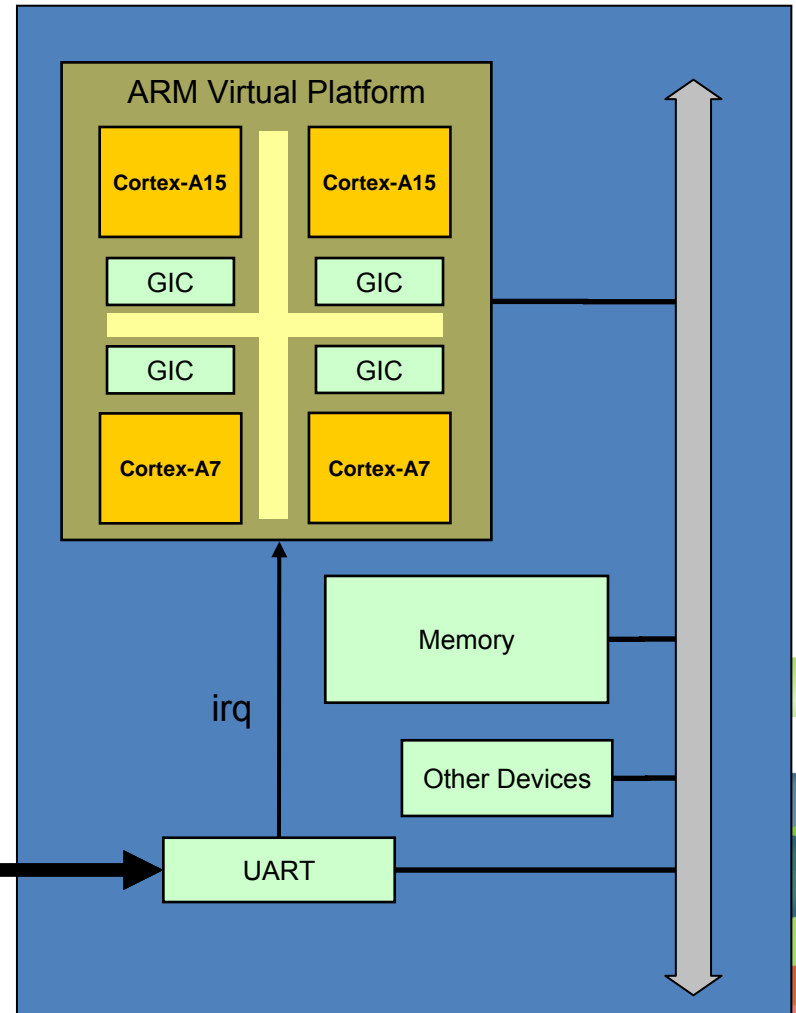
TID  CPUID  SLEEPING  DETACHED  DESCHEDULED  CANCELLED  THREAD
16    0        N          N          N          N          main
0     1        N          V          N          N          infinite
2     2        N          V          N          N          infinite
3     3        N          V          N          N          infinite
1     1        N          V          N          N          infinite
17    queued  N          N          N          N          dummy
18    queued  N          N          N          N          dummy
19    queued  N          N          N          N          dummy

CORTEXA9> pps

TID  CPUID  SLEEPING  DETACHED  DESCHEDULED  CANCELLED  THREAD
0     0        N          V          N          N          infinite
1     1        N          V          N          N          infinite
3     2        N          V          N          N          infinite
16    3        N          N          N          N          main
2     2        N          V          N          N          infinite
17    queued  N          N          N          N          dummy
18    queued  N          N          N          N          dummy
19    queued  N          N          N          N          dummy

CORTEXA9> _
    
```

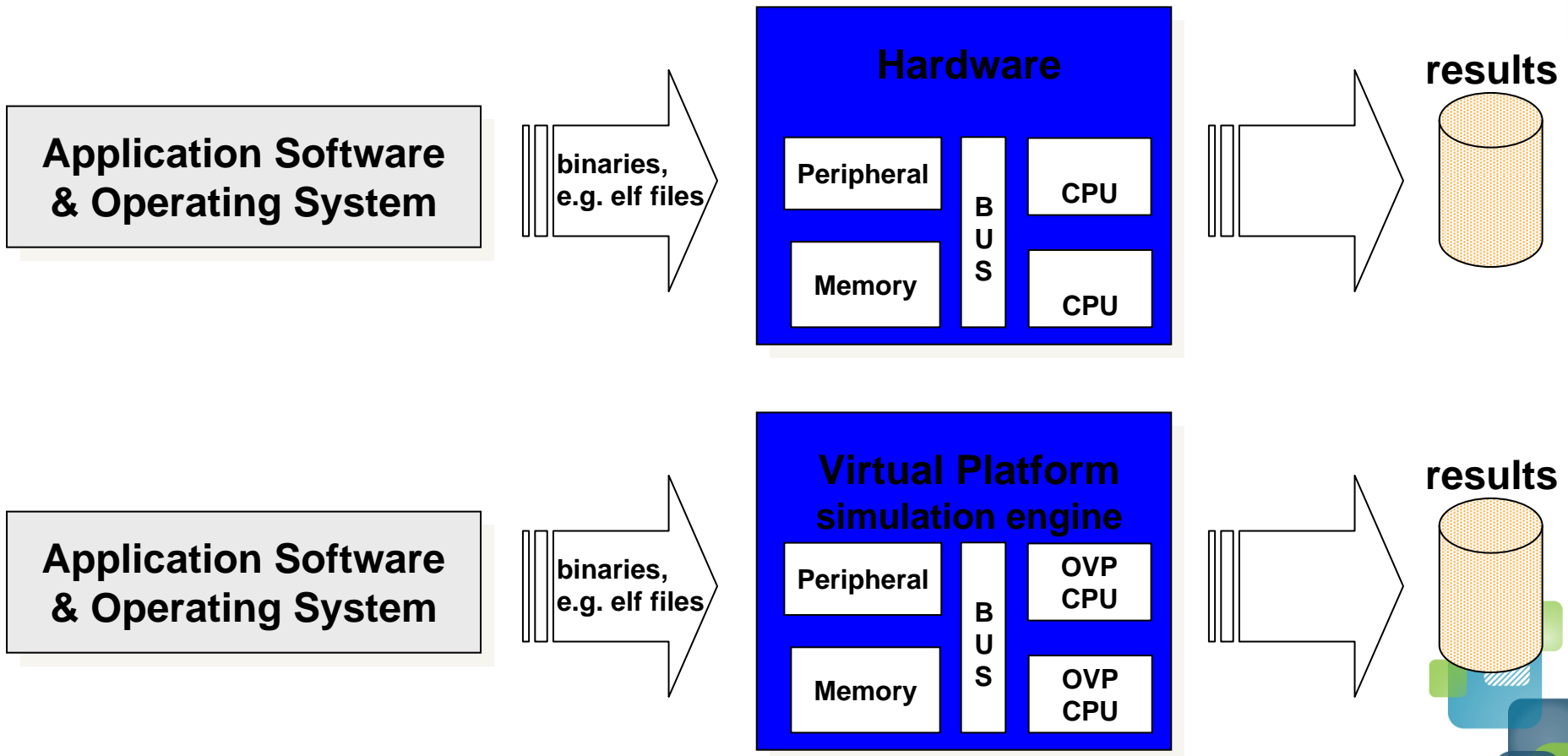
Keyboard



Virtual Platform Requirements Checklist

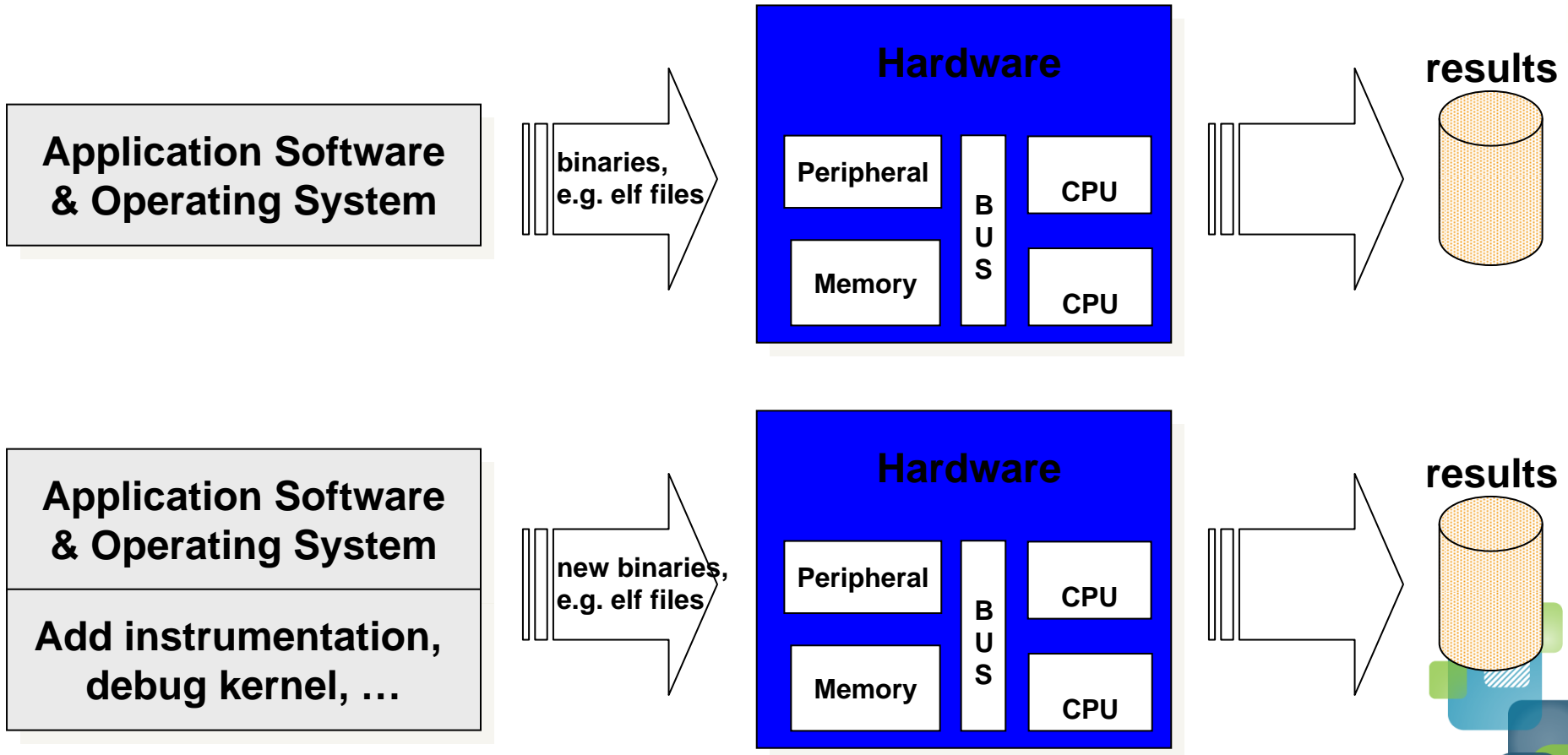
- Performance near real time
- ✓ Instruction accurate virtual platforms run at 100s of MIPS
- Run target binaries without change
- ✓ Use the same tool chain for compiling as for the real hardware
- Repeatable results
- ✓ Simulation is a deterministic process, with repeatable results
- Multi-processor debug capability
 - Whether multiple processors on one device or board or system
- ✓ Available either from virtual platform tool vendor or tool chain (IDE) vendor
- Software verification, analysis and profiling tools
- Tools are needed so the virtual platforms can deliver on the simulation promise of complete controllability, visibility

Virtual Platforms Simulate the Software Running on the Hardware



$$\text{results(HW)} = \text{results(VP)}$$

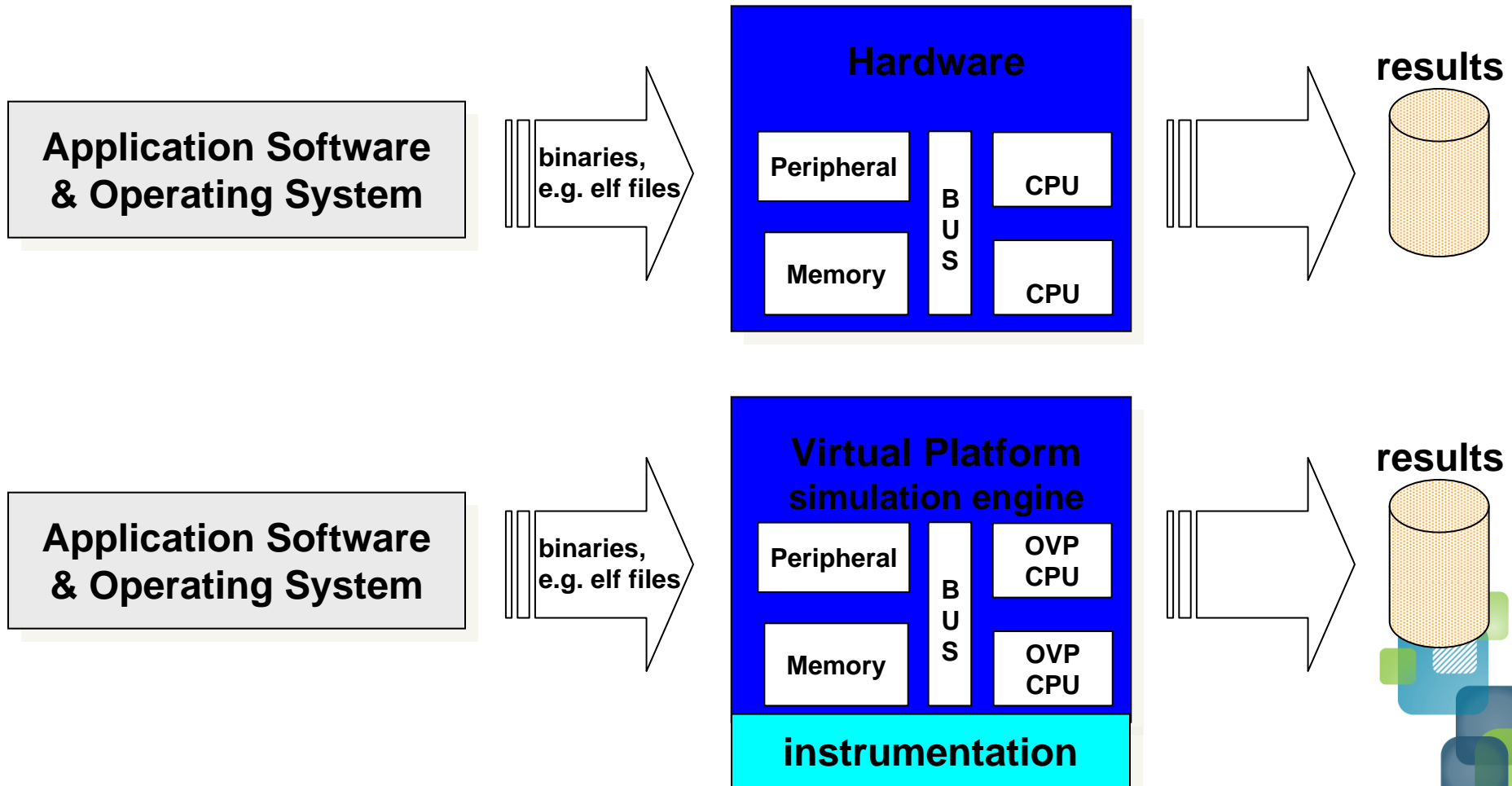
Software Analysis on Hardware (OS tracing, event scheduler analysis, ...)



$$\text{results(HW)} \stackrel{?}{=} \text{results(HW + instrumentation)}$$

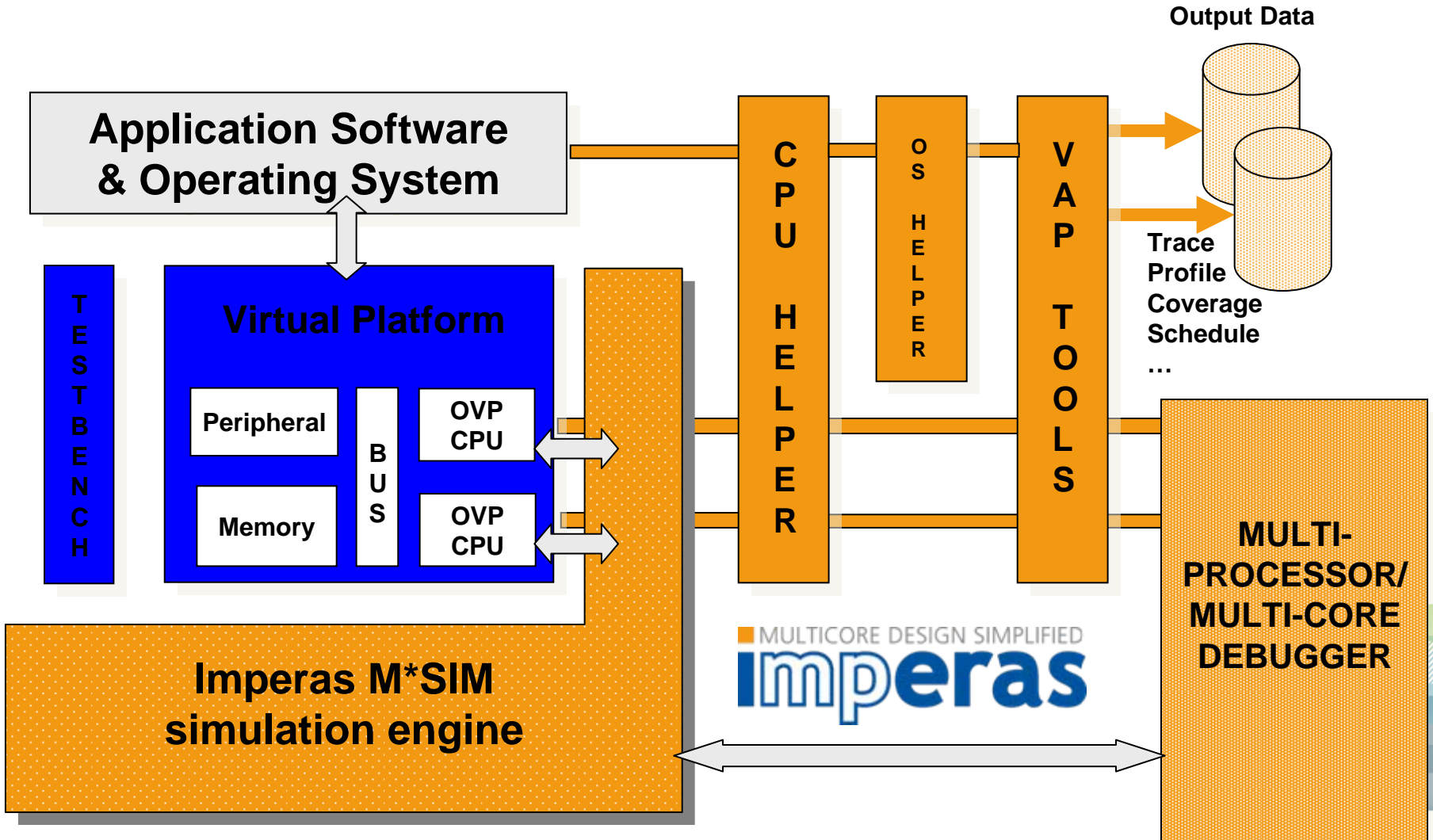
Software Analysis on Virtual Platform can be Non-Intrusive

(code coverage, profiling, tracing, memory analysis, ...)



$$\text{results(HW)} = \text{results(VP + instrumentation)}$$

Virtual Platform with Verification, Analysis and Profiling (VAP) Tools Plus Debugger



Requirements for Verification, Analysis, and Profiling Tools

- Non-intrusive: no modification of application source code
- Minimal overhead: simulations should still run *fast*
- Modular: can run one or more without tools stepping on each other
- Flexible: interactive or scripted use models
- Configurable: adjust for specific platform and focus
- Distributable: need to be shipped with virtual platform as integral part of SDK for specific platform/chip

Agenda

- Silicon without software is just sand...
- What is a virtual platform?
- Case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

Agenda

- Silicon without software is just sand...
- What is a virtual platform?
- Case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

Example 1: SMP Linux / Android on Dual Core ARM Cortex-A9

- Goal: in depth understanding of SMP Linux / Android operation on dual core processor
- Virtual platform: ARM Versatile Express
- Processor model: OVP ARM Cortex-A9 MPx2
- Imperas VAP tools:
 - OS task tracing: start OS analysis and debug at higher level of abstraction than with instruction tracing
 - Requires OS aware capability
 - OS scheduler profiling: process creation / deletion and context switching
 - Requires OS aware capability

ARM Versatile Express Cortex-A9MP / SMP Linux

```

PS-110162 (1024x768)
This root FS contains most basic linux utilities (implemented with busybox)
and the Lynx web browser.

Kernel config is available through /proc/config.gz

Welcome to OVP simulation from Imperas

Log in as root with no password.
Imperas login:

```

```

rtc-pl031 mb:rtc: rtc cor
mmc-pl18x mb:mmci: mmc0:
usbcore: registered new i
usbhid: USB HID core driv
ALSA device list:
No soundcards found.
oprofile: using arm/armv7
TCP cubic registered
NET: Registered protocol
VFP support v0.3: impleme
rtc-pl031 mb:rtc: setting
Freeing init memory: 172K
input: AT Raw Set 2 keyboard as /devices/mb:kmi0/serio0/input/input0
input: ImExPS/2 Generic Explorer Mouse as /devices/mb:kmi1/serio1/input/input1

This root FS contains most basic linux utilities (implemented with busybox)
and the Lynx web browser.

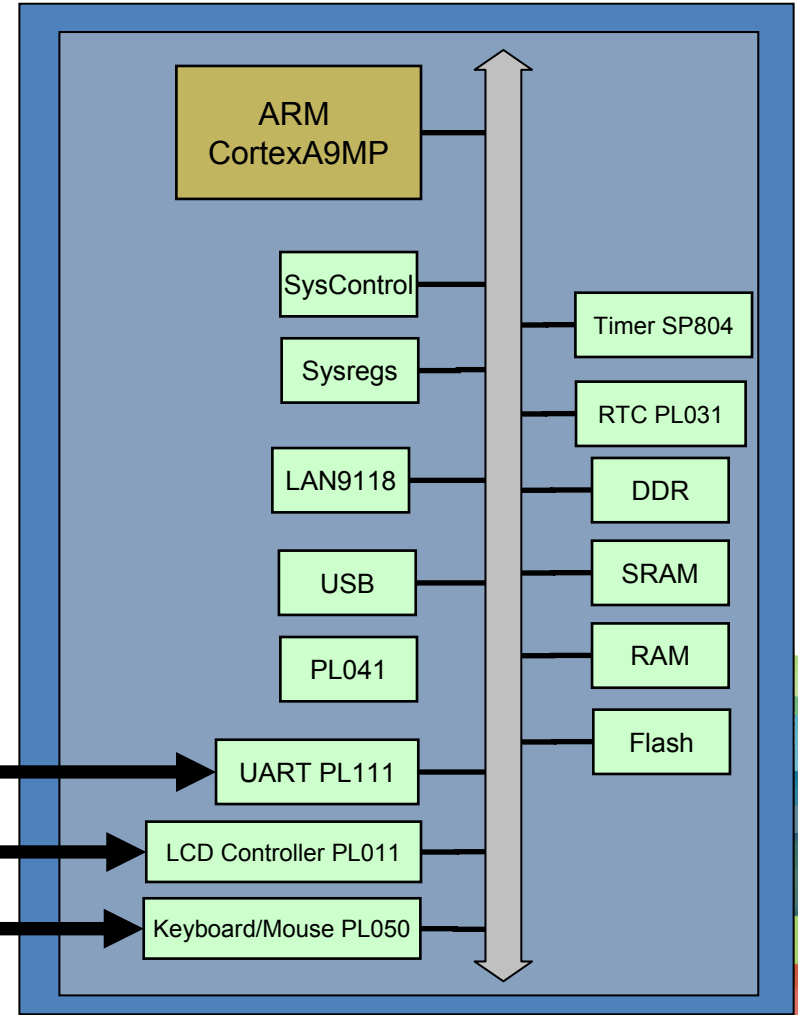
Kernel config is available through /proc/config.gz

Welcome to OVP simulation from Imperas

Log in as root with no password.
Imperas login:

```

Keyboard / Mouse

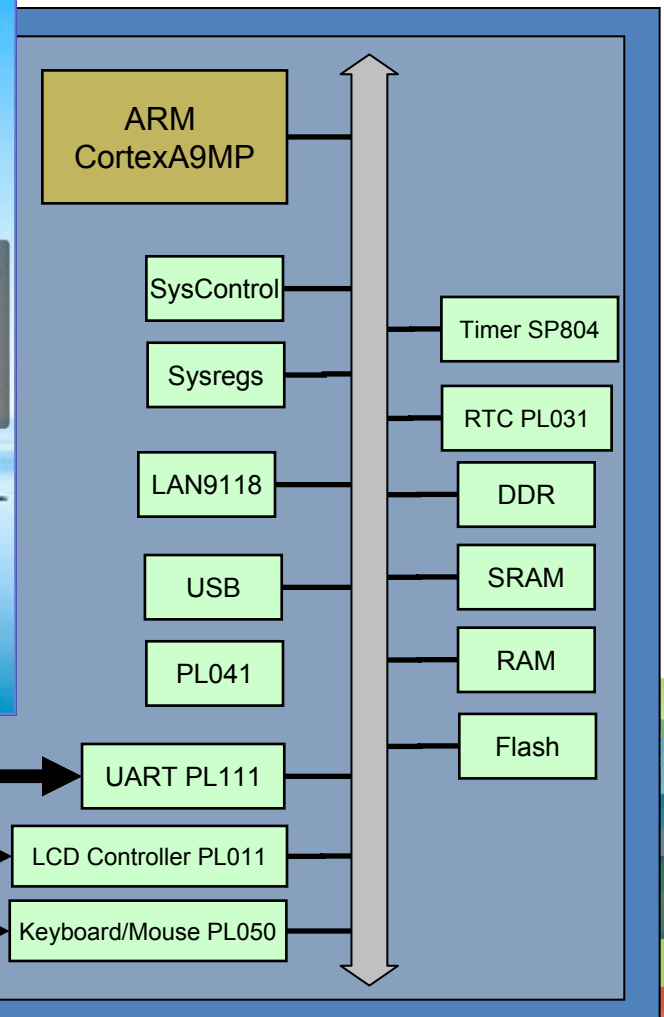


ARM Versatile Express Booting Android

```

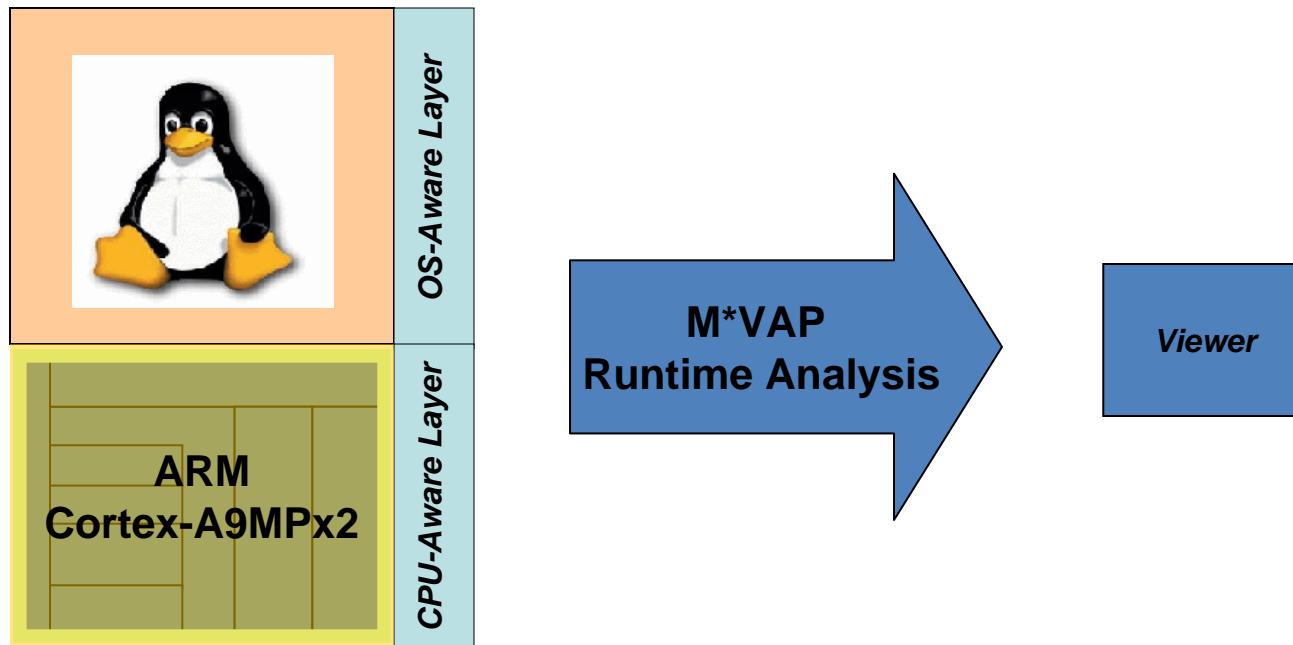
uart0
[ 3.615000] EXT3-fs: barriers not enabled
<6>kjournald starting. Commit interval 5 seconds
[ 3.629000] kjournald starting. Commit interval 5 seconds
<6>EXT3-fs (mmcblk0p2): using internal journal
[ 3.632000] EXT3-fs (mmcblk0p2): using internal journal
<6>EXT3-fs (mmcblk0p2): recovery complete
[ 3.632000] EXT3-fs (mmcblk0p2): recovery complete
<6>EXT3-fs (mmcblk0p2): mounted filesystem with writeback data mode
[ 3.634000] EXT3-fs (mmcblk0p2): mounted filesystem with writeback data mode
<6>EXT3-fs: barriers not enabled
[ 3.639000] EXT3-fs: barriers not enabled
<6>kjournald starting. Commit interval 5 seconds
[ 3.663000] kjournald starting. Commit interval 5 seconds
<6>EXT3-fs (mmcblk0p3): using internal journal
[ 3.665000] EXT3-fs (mmcblk0p3): using internal journal
<6>EXT3-fs (mmcblk0p3): recovery complete
[ 3.665000] EXT3-fs (mmcblk0p3): recovery complete
<6>EXT3-fs (mmcblk0p3): mounted filesystem with writeback data mode
[ 3.667000] EXT3-fs (mmcblk0p3): mounted filesystem with writeback data mode
# <6>warning: 'rild' uses 32-bit capabilities (legacy support in use)
[ 3.831000] warning: 'rild' uses 32-bit capabilities (legacy support in use)

I/Installer< 727>: disconnecting...
E/Installer< 727>: connection failed
D/dalvikcom< 727>: GC_CONCURRENT freed 197K, 48% free 2954K/5639K, external 716K
/
1038K, paused 1ms+2ms
M/PackageManager< 727>: Running ENG build: no pre-dexopt!
  
```



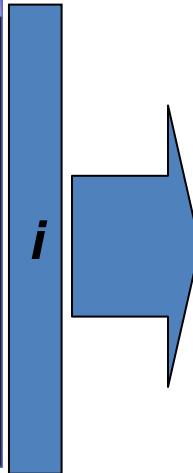
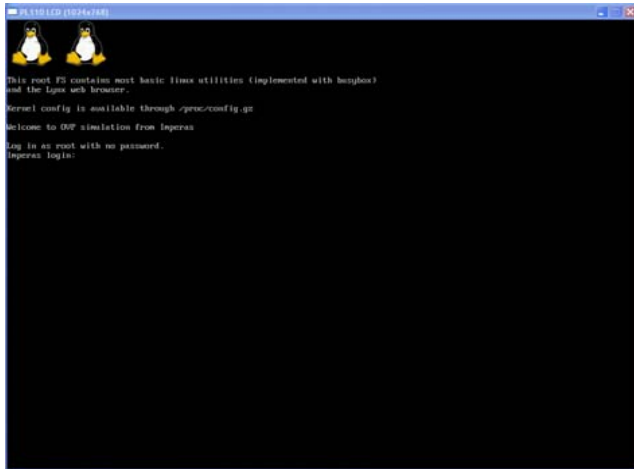
OS-Aware Software Analysis

- Non-intrusive trace of
 - process creation
 - context switch
 - process deletion
- Captures communications between processes



OS-Aware Software Analysis Example: OS Task Tracing

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



Introspection (Linux OS)

```

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"

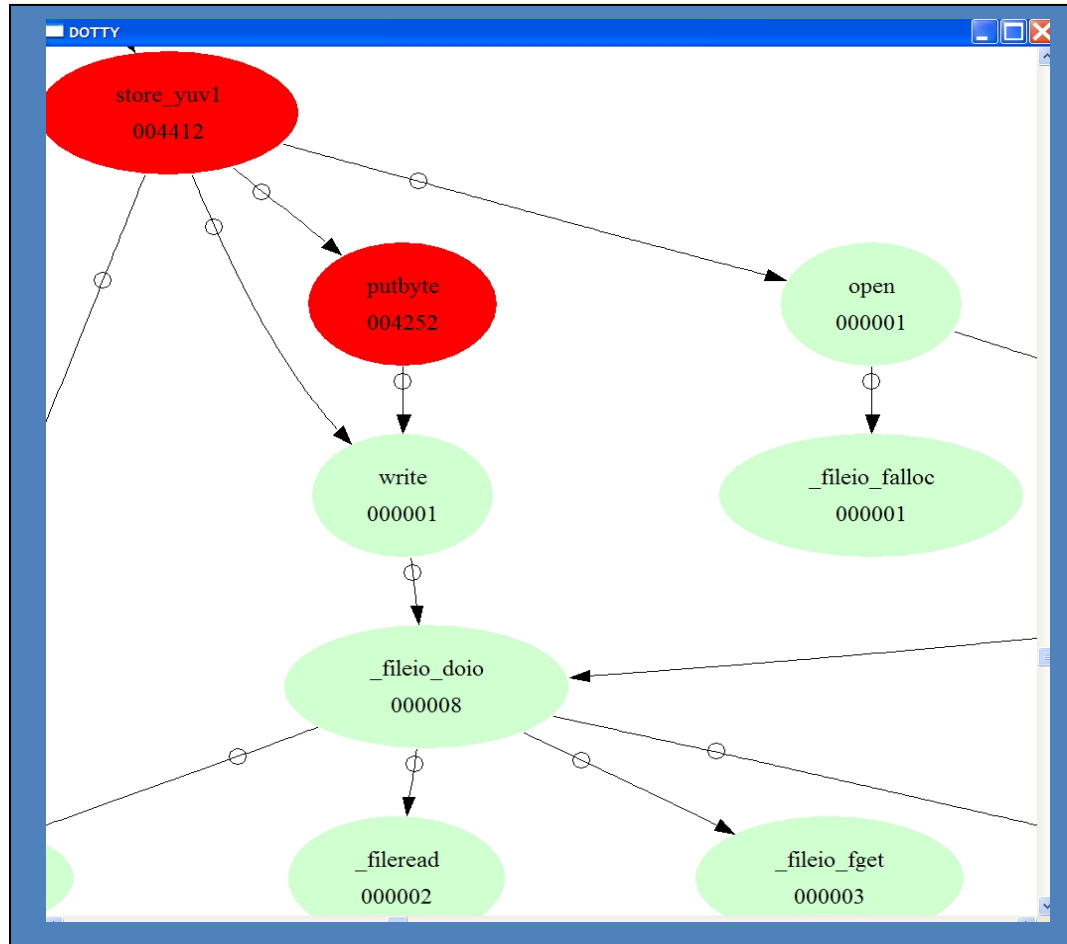
do_execve: pid=20
  filename=/sbin/hotplug
  argv virt=0x804613f0 phys=0x004613f0 "/sbin/hotplug"
  argv virt=0x8045bc34 phys=0x0045bc34 "module"
  envp virt=0x80413500 phys=0x00413500 "HOME=/"
  envp virt=0x804170b4 phys=0x004170b4 "PATH=/sbin:/bin:"
  envp virt=0x81150000 phys=0x01150000 "ACTION=add"
  envp virt=0x8115000b phys=0x0115000b "DEVPATH=/modu
  .....
    
```

- 1) Enables in-depth monitoring and analysis, even before console is available
- 2) Provides tracing at different levels of abstraction, granularity
 - ~ 700,000,000 instructions to boot Linux
 - ~ 700 tasks to boot Linux; i.e. Task Trace provides better ability to debug OS problems, such as bring up

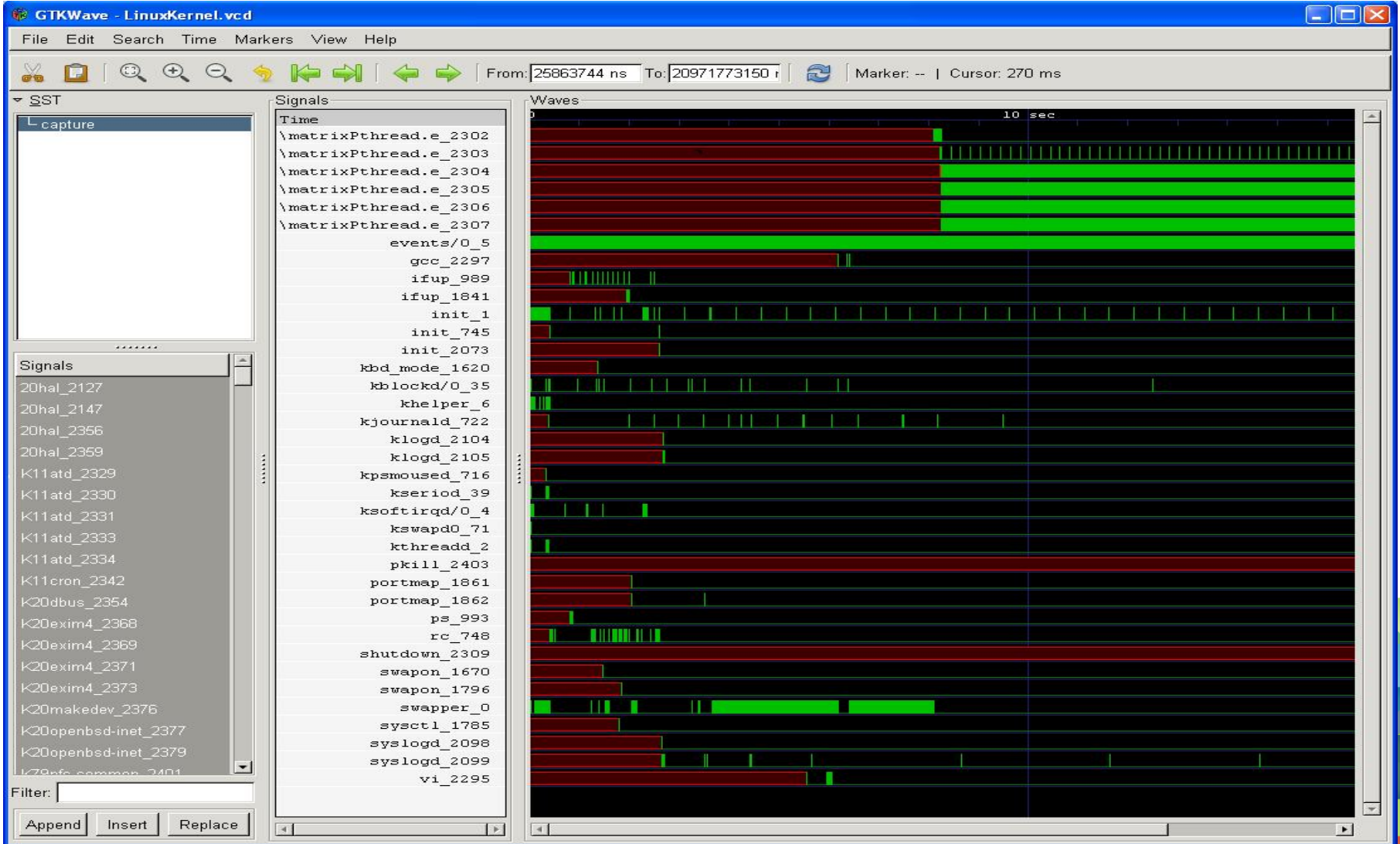
Can be done for any OS

- ✓ Linux
- ✓ Nucleus
- ✓ µC/OS
- ✓ µltron
- ✓ FreeRTOS
- ✓ ...
- ✓ Proprietary

OS Task Call Tree & Profiling (Public Domain Viewer)



OS Scheduler Tracing (Public Domain Viewer)



Case 1: OS-Aware summary

- Modern complex Operating Systems/RTOS runs millions, billions of instructions before ‘interesting’ things happen
- Yes you need to simulate to see what is going on
- BUT – you need more than just instruction trace
- You need advanced OS-aware technologies
 - eg: task trace, scheduler trace, task profile
- AND they must be non-intrusive
 - Especially for multi-core, multi-processor

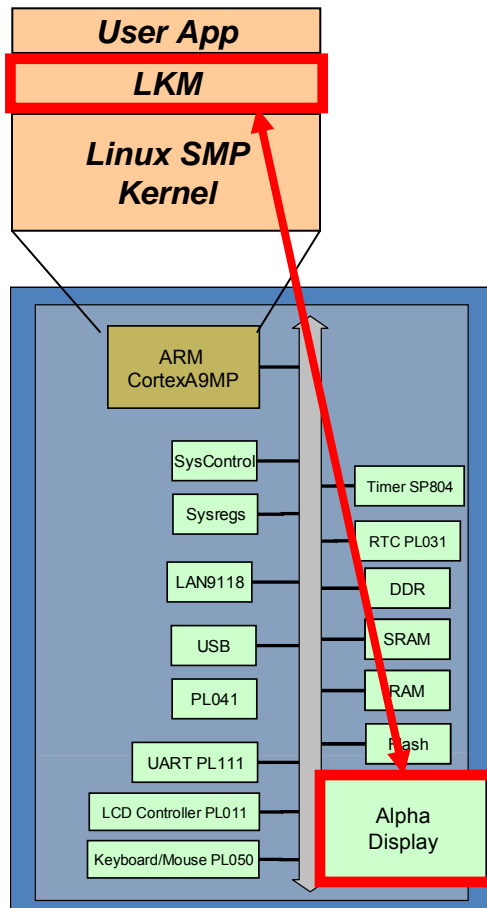
Agenda

- Silicon without software is just sand...
- What is a virtual platform?
- Example case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

Example 2: Analysis & Debug of Loadable Kernel Module (LKM)

- LKMs are dynamic drivers used with Linux
 - Dynamic nature of the driver, plus complex interactions between cores, peripherals, OS and driver make analysis and debug difficult
 - What about timing-dependent bugs in driver with instruction accurate simulation?
 - For multicore devices, need to minimize timing dependencies in software
 - Instruction accurate virtual platforms are complementary to other development methodologies
- Platform: Cortex-A9MPx2
- Imperas Debug + VAP tools:
 - Multiprocessor debug enables simultaneous debug of software on peripherals as well as the processors
 - Set watchpoints, breakpoints on peripherals instead of processors
 - Functional coverage of test scenarios
 - Protocol verification

Abstraction, Operating System Application, Driver (LKM) and Peripheral



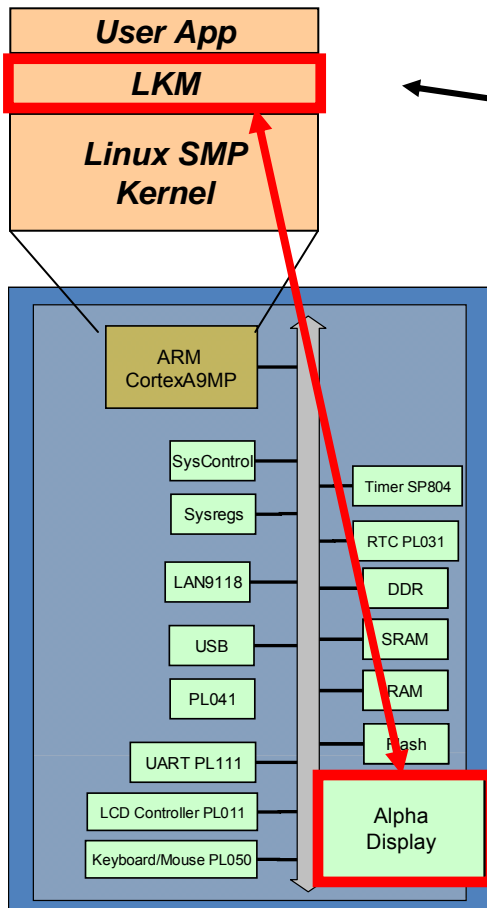
- Add hardware/peripheral to platform
- Needs driver adding to Linux
- LKM sits as part of the Kernel and provides services to the User Apps
- Need to verify interactions with Devices
 - AlphaNumeric Display peripheral added
- Debugging
 - Device driver and Peripheral model
- Challenges related to
 - Different components (CPU+peripheral)
 - Timing – sequencing of events
 - Repeatability

Imperas MP Debugger provides easy development

- Integrated debugger with simulator
 - Full control of running/stopping cores/processors
- Debug of both hardware and software - together
- Spatial, temporal, and abstraction
- Full reproducibility and determinism

- Can see everything in the platform
 - All hardware components
 - All software on all processors
 - [normal software debuggers only ever see the processor and its software – no access to platform components, behavioral subsystems]
 - Be able to single step/control any code – hardware model or software app
- Powerful MP features
 - e.g. place watchpoint on memory – triggers whenever any processor or component accesses
 - Global state, access, views, control, scripting...
 - Control any processor – in SMP cluster or hetero AMP system

Case 2 summary: using complex OSeS requires sophisticated tools



- Imperas MP Debugger provides simultaneous access to hardware and software with OS-awareness
- Sorts out issues with dynamic symbol loading
- Allows very efficient development of software that interacts with hardware
 - Even on top of complex OS

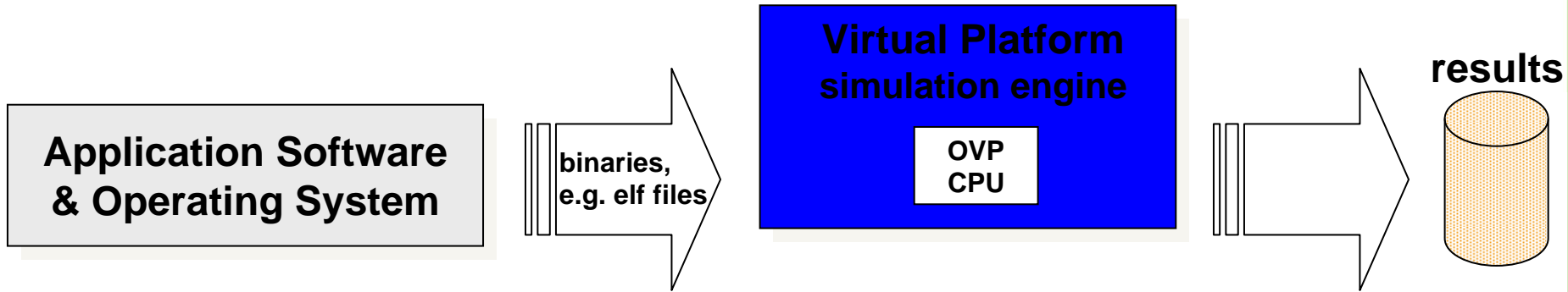
Agenda

- Silicon without software is just sand...
- What is a virtual platform?
- Example case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

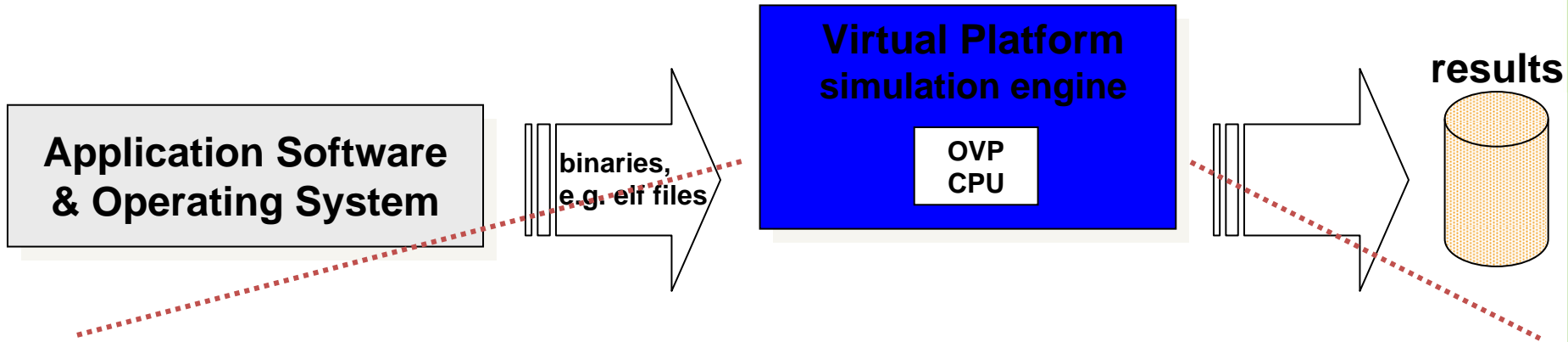
Example 3: In Depth OS Behavior Analysis

- Goal: Use virtual platform visibility to measure the number of instructions from exception entry to return
- Custom tool developed for analyzing exception handler instruction counts
 - Utilizes Imperas VAP Tools infrastructure
 - Interested in callbacks on exceptions and their returns
 - VapHelper provides callbacks on entry and return from exception
 - CpuHelper detects and provides details of exceptions
 - Adds new command to simulation environment to turn on/off tracing
 - Simply reports entries and returns with elapsed instruction counts
 - Could easily be enhanced to provide statistical analysis, report worst case occurrences, provide call stack snapshot at exception, provide OS process information, etc.

Imperas Simulation Infrastructure Enables Tool Definition



Imperas Simulation Infrastructure Enables Tool Definition

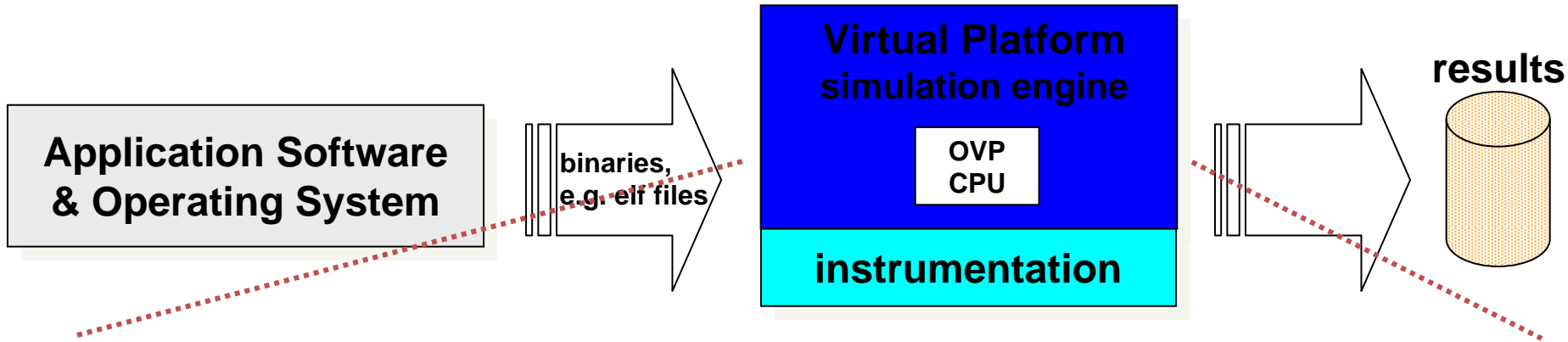


Simulation Infrastructure

OVP Fast Processor Model:
CPU functionality, predefined views, events, actions

Simulation Engine:
Just In Time (JIT) code morphing (binary translation)

Imperas Simulation Infrastructure Enables Tool Definition



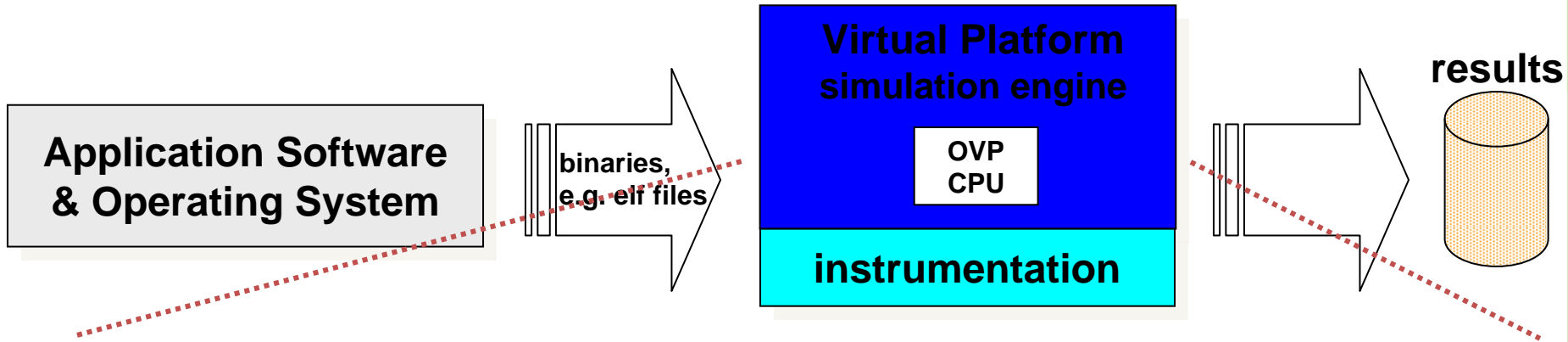
Simulation Infrastructure

CPU and OS Helpers:
CPU and OS specific information

OVP Fast Processor Model:
CPU functionality, predefined views, events, actions

Simulation Engine:
Just In Time (JIT) code morphing (binary translation)

Imperas Simulation Infrastructure Enables Tool Definition



Simulation Infrastructure

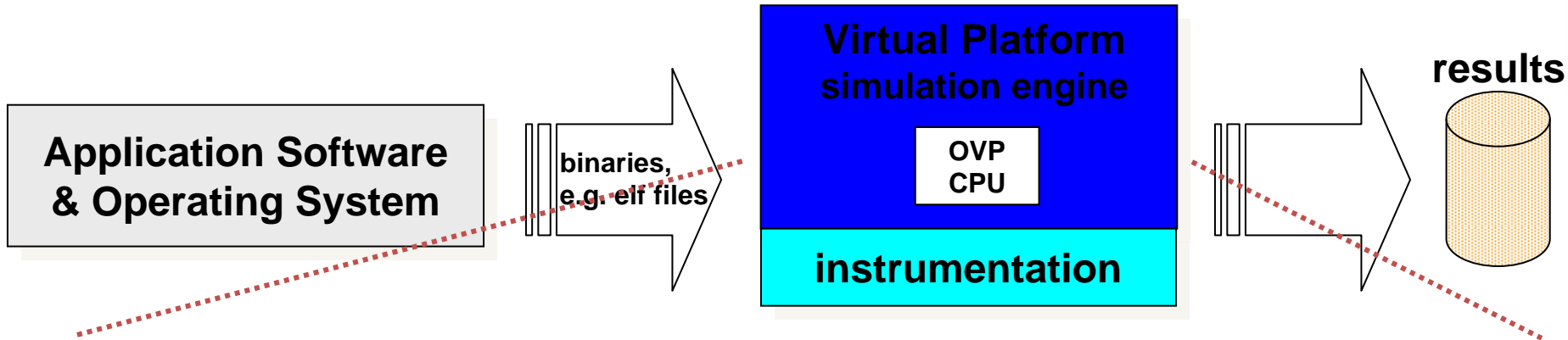
Tool Helper:
API enabling user-definition of software analysis tools

CPU and OS Helpers:
CPU and OS specific information

OVP Fast Processor Model:
CPU functionality, predefined views, events, actions

Simulation Engine:
Just In Time (JIT) code morphing (binary translation)

Imperas Simulation Infrastructure Enables Tool Definition



VAP Tool:
Definition of the tool, written in C (from library or can be user written)

Tool Helper:
API enabling user-definition of software analysis tools

CPU and OS Helpers:
CPU and OS specific information

OVP Fast Processor Model:
CPU functionality, predefined views, events, actions

Simulation Engine:
Just In Time (JIT) code morphing (binary translation)

Simulation Infrastructure

Exception Analysis Tool

Simulator Engine

OVP Fast Processor Model
predefined "exception" event

Instruction 1
Instruction 2
...
Exception
Instruction N
Instruction N+1
...
Instruction N+M
Exception Return
Instruction
...

CPU Helper

- When "exception" event occurs:
 - Determines all the addresses this exception might return to
 - Produces a description string for the event

Tool Helper

- When notified of an "exception" event
 - Determines and saves the current context of the processor
 - Registers intercepts on all possible return addresses
- When exception return address is intercepted
 - Determines if context matches a previously observed exception

Exception Analysis Tool

- Adds a user command to enable/disable exception tracing
- When notified of an exception entry
 - Creates data structure, including instruction count on entry
- When notified of an exception exit
 - Determines elapsed instructions since entry
- Provides report of data collected about the exception event

[Show Exception Analysis Tool](#)

Exception Analysis Tool: Results

- Platform is booting Linux
- Exception analysis tool is used interactively as OS is running
 - Could be used in script
- Reports where exception was taken and returned
- Calculates instructions between exception entry and return

```
C:\WINDOWS\system32\cmd.exe
imperas (mipsle1) > idebug
----- ENTER DEBUG MODE -----
idebug (mipsle1) > ::extrace
idebug (mipsle1) > c
Warning (IDE) /MipsMaltaLinux/PIIX4-IDE: Failed to open file 'mipsle1_hda'
Warning (IDE) /MipsMaltaLinux/PIIX4-IDE: Failed to open file 'mipsle1_hdb'
Warning (IDE) /MipsMaltaLinux/PIIX4-IDE: Failed to open file 'mipsle1_cd'
Warning (DMA_UNSL) /MipsMaltaLinux/PIIX4-IDE: PCI DMA ch:1 (1) 0x14 <= 0x0
Info (MIPS32_IAS_COPO_WRITE) 0x8048d250: write to unsupported COP0 register 21 sel 0
TRC (EXCP_TR) 200904090: 'mipsle1':Exception 1 at 0x8015fe40: excCode_Int Timer (cause=0x40808000)
TRC (EXCP_TR) 200906043: 'mipsle1':Exception 1 at 0x8015fe40 returned. [1953 instrs]
TRC (EXCP_TR) 201904090: 'mipsle1':Exception 2 at 0x804854fc: excCode_Int Timer (cause=0xc0808000)
TRC (EXCP_TR) 201905753: 'mipsle1':Exception 2 at 0x804854fc returned. [1663 instrs]
TRC (EXCP_TR) 202904090: 'mipsle1':Exception 3 at 0x804854f4: excCode_Int Timer (cause=0x40808000)
TRC (EXCP_TR) 202905748: 'mipsle1':Exception 3 at 0x804854f4 returned. [1658 instrs]
TRC (EXCP_TR) 203904090: 'mipsle1':Exception 4 at 0x80100b68: excCode_Int Timer (cause=0x40808000)
TRC (EXCP_TR) 203905748: 'mipsle1':Exception 4 at 0x80100b68 returned. [1658 instrs]
TRC (EXCP_TR) 204904090: 'mipsle1':Exception 5 at 0x80100b6c: excCode_Int Timer (cause=0x40808000)
TRC (EXCP_TR) 204905748: 'mipsle1':Exception 5 at 0x80100b6c returned. [1658 instrs]
TRC (EXCP_TR) 205904090: 'mipsle1':Exception 6 at 0x80100b6c: excCode_Int Timer (cause=0x40808000)
TRC (EXCP_TR) 205905748: 'mipsle1':Exception 6 at 0x80100b6c returned. [1658 instrs]
TRC (EXCP_TR) 206904090: 'mipsle1':Exception 7 at 0x80100b6c: excCode_Int Timer (cause=0xc0808000)
TRC (EXCP_TR) 206905748: 'mipsle1':Exception 7 at 0x80100b6c returned. [1658 instrs]
TRC (EXCP_TR) 207904090: 'mipsle1':Exception 8 at 0x80100b6c: excCode_Int Timer (cause=0x40808000)
TRC (EXCP_TR) 207905748: 'mipsle1':Exception 8 at 0x80100b6c returned. [1658 instrs]
TRC (EXCP_TR) 208904090: 'mipsle1':Exception 9 at 0x80100b6c: excCode_Int Timer (cause=0xc0808000)
TRC (EXCP_TR) 208905748: 'mipsle1':Exception 9 at 0x80100b6c returned. [1658 instrs]
TRC (EXCP_TR) 209904090: 'mipsle1':Exception 10 at 0x80100b6c: excCode_Int Timer (cause=0x40808000)
```

Case 3: Exception Analysis

- With complex operating systems hard to gain visibility
 - Not just for exceptions, but for all operations
- Requires advanced tools with abstractions
 - CPU-aware, OS-aware
- Cannot expect tool vendor to know all types of analysis that is appropriate for your system
- Need ability for user created extensions that are
 - Easy, well documented, efficient, abstraction, fast
 - And of course – non-intrusive
 - And with no requirement for app recompile

Agenda

- Silicon without software is just sand...
- What is a virtual platform?
- Example case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

Example 4: AMP System Analysis

- Goal: in depth understanding of AMP operation on heterogeneous multiprocessor system
- Zynq virtual platform with dual core ARM Cortex-A9 processor plus Xilinx MicroBlaze processor

Heterogeneous Platform

ARM Cortex-A9MPCore + Xilinx MicroBlaze

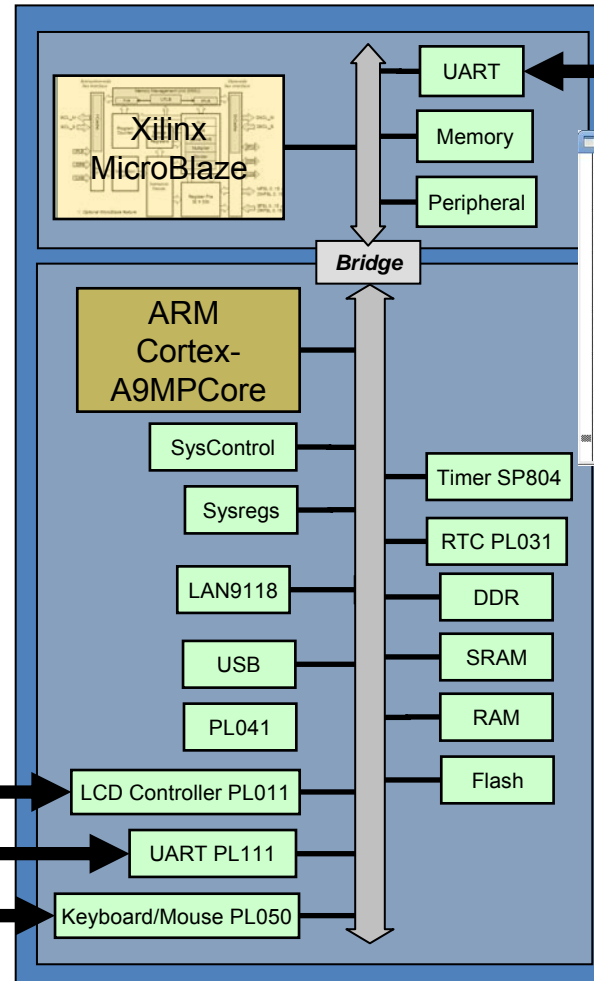
```

MULTILOG (10/26/14)
This root FS contains most basic linux utilities (implemented with busybox)
and the Lynx web browser.
Kernel config is available through /proc/config.gz
Welcome to OVP simulation from Imperas
Log in as root with no password.
Imperas login:
  
```

```

rtc-pl031 mb:rtc: rtc
mmci-pl118 mb:mmci: m
usbcore: registered n
usbhid: USB HID core
ALSA device list:
No soundcards found
oprofile: using arm/a
TCP cubic registered
NET: Registered protocol family 17
VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 2
rtc-pl031 mb:rtc: setting system clock to 1970-01-01 00:00:00 UTC (0)
Freeing init memory: 172K
input: AT Raw Set 2 keyboard as /devices/mb:km0/serio0/input/input0
input: ImExPS/2 Generic Explorer Mouse as /devices/mb:km1/serio1/input/input1
This root FS contains most basic linux utilities (implemented with busybox)
and the lynx web browser.
Kernel config is available through /proc/config.gz
Welcome to OVP simulation from Imperas
Log in as root with no password.
Imperas login:
  
```

Keyboard / Mouse

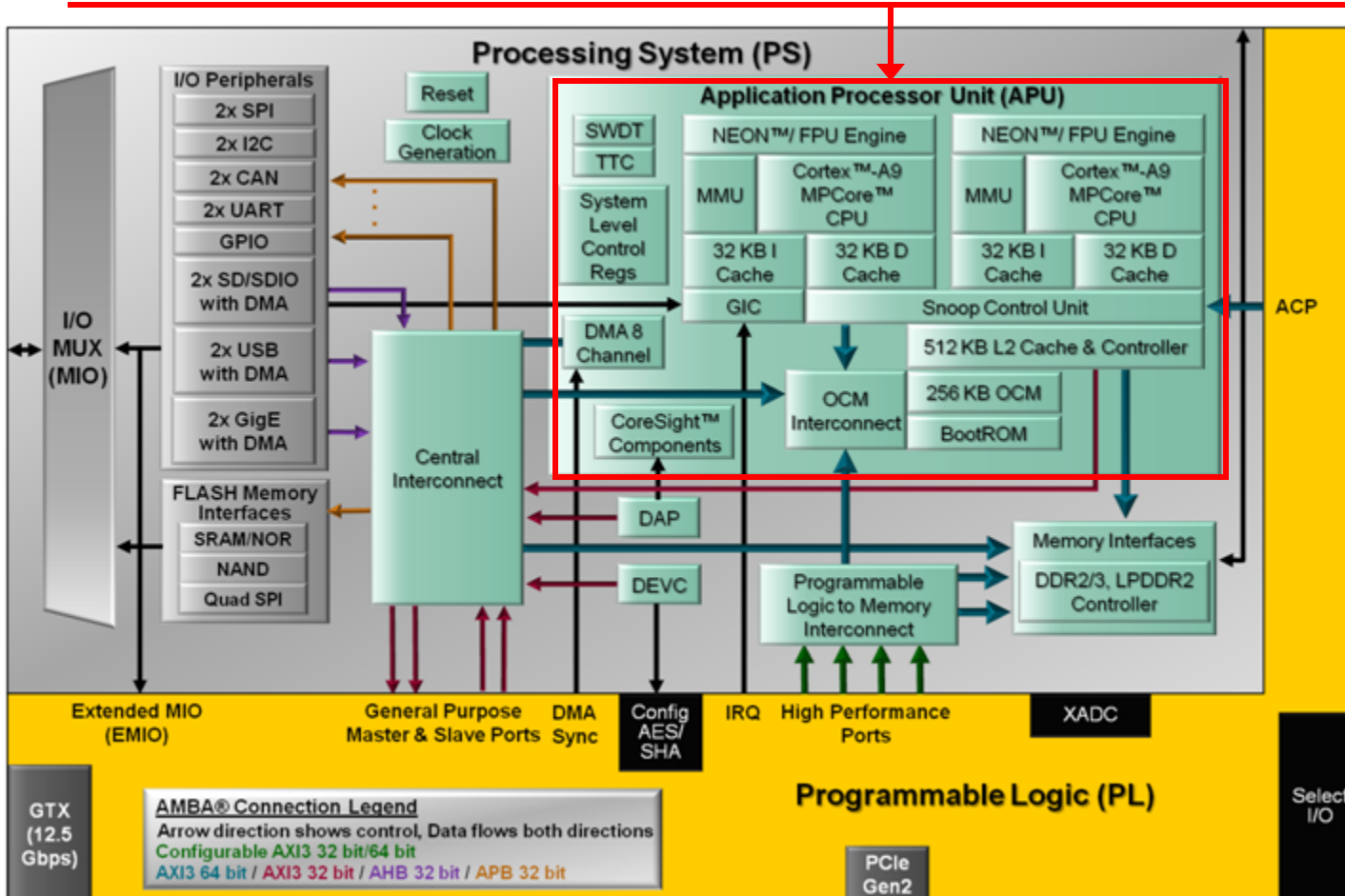


```

Xilinx MicroBlaze Console
DVP MicroBlaze .. sags ... Hello World 9913
DVP MicroBlaze .. sags ... Hello World 9914
DVP MicroBlaze .. sags ... Hello World 9915
DVP MicroBlaze .. sags ... Hello World 9916
DVP MicroBlaze .. sags ... Hello World 9917
DVP MicroBlaze .. sags ... Hello World 9918
DVP MicroBlaze .. sags ... Hello World 9919
DVP MicroBlaze .. sags ... Hello World 9920
DVP MicroBlaze .. sags ... Hello World 9921
DVP MicroBlaze .. sags ... Hello World 9922
DVP MicroBlaze .. sags ... Hello World 9923
DVP MicroBlaze .. sags ... Hello World 9924
DVP MicroBlaze .. sags ... Hello World 9925
DVP MicroBlaze .. sags ... Hello World 9926
DVP MicroBlaze .. sags ... Hello World 9927
DVP MicroBlaze .. sags ... Hello World 9928
DVP MicroBlaze .. sags ... Hello World 9929
DVP MicroBlaze .. sags ... Hello World 9930
DVP MicroBlaze .. sags ... Hello World 9931
DVP MicroBlaze .. sags ... Hello World 9932
DVP MicroBlaze .. sags ... Hello World 9933
DVP MicroBlaze .. sags ... Hello World 9934
DVP MicroBlaze .. sags ... Hello World 9935
DVP MicroBlaze .. sags ... Hello World 9936
  
```

Xilinx Zynq™-7000 EPP System Level Block Diagram

- ✓ Cadence virtual platform includes Imperas OVP Fast Processor Model of ARM Cortex-A9MPx2



- OVP Fast Processor Models enable use of VAP tools
- CPU and OS aware
 - Almost 100+ CPU cores supported
 - OS support: Linux, Nucleus, uCLinux, FreeRTOS, μ C/OS II, eCoS, μ ltron, proprietary, ...
 - Used for hardware-dependent software development
 - Early software development
 - Software testing
 - System analysis
- 25+ M*VAP tools: code coverage, profiling (function, OS events), tracing (instruction, function, event, OS task, OS kernel), memory analysis, ...
- Non-intrusive
 - No instrumentation or modification of application code
 - No change to instruction ordering
- Execute as native host code for minimal overhead
- Can be used interactively or scripted
- Multiple tools can be loaded simultaneously
- User defined tools enabled: fault injection, protocol verification, software behavior analysis, ...
 - Users write tools in C
 - Documented API

Agenda

- Silicon without software is just sand...
- What is a virtual platform?
- Example case studies for virtual platform based software development
 - SMP Linux / Android
 - OS-related software: loadable kernel modules (LKMs) for Linux
 - OS exception analysis
 - AMP system
- Summary, Q&A

Virtual Platform Based Software Development

- Simulation (virtual platforms) enables full visibility, controllability of software
- Tools are needed – more than just simulation – to deliver on the promise of visibility, controllability
- Verification, analysis and profiling tools for virtual platforms provide complementary capability to existing development methodology

Take away: using Virtual Platforms with advanced tools enhances software development in terms of quality, timescales, efforts, and results