MULTICORE DESIGN SIMPLIFIED

Embedded Software Verification Using Virtual Platforms

Duncan Graham

DAC Virtual Platform Workshop

29 July 2009

Embedded Software Verification Using Virtual Platforms

- Embedded software/systems issues: complexity
- Key software development tasks
  - OS
  - Drivers
  - Applications
- Virtual platform infrastructure: necessary but not sufficient
- Requirements for software development

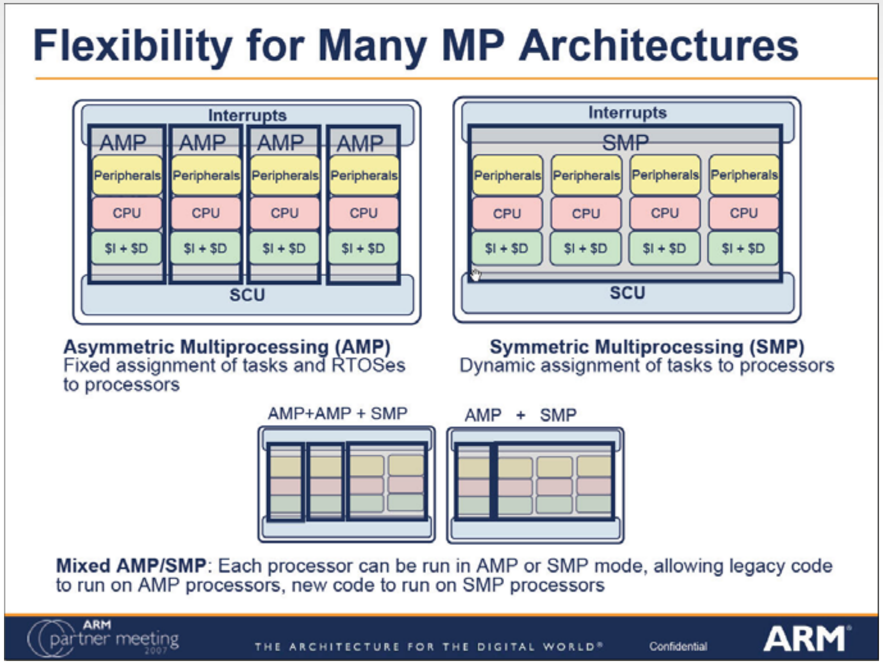
Embedded Software Verification Using Virtual Platforms

MULTICORE DESIGN SIMPLIFIED imperas

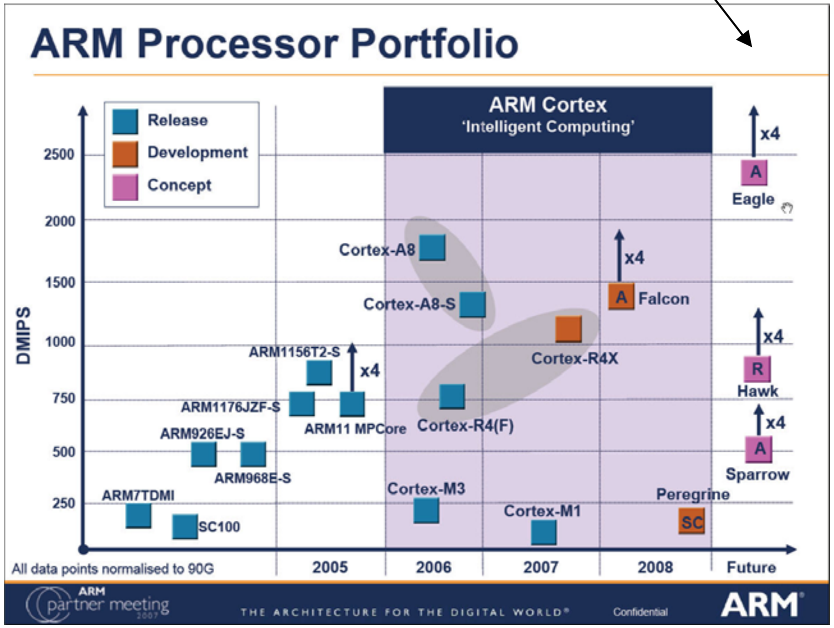
- Embedded software/systems issues: complexity
- Key software development tasks
  - OS
  - Drivers
  - Applications
- Virtual platform infrastructure: necessary but not sufficient
- Requirements for software development

The Future of SoCs is Multicore

MULTICORE DESIGN SIMPLIFIED imperas

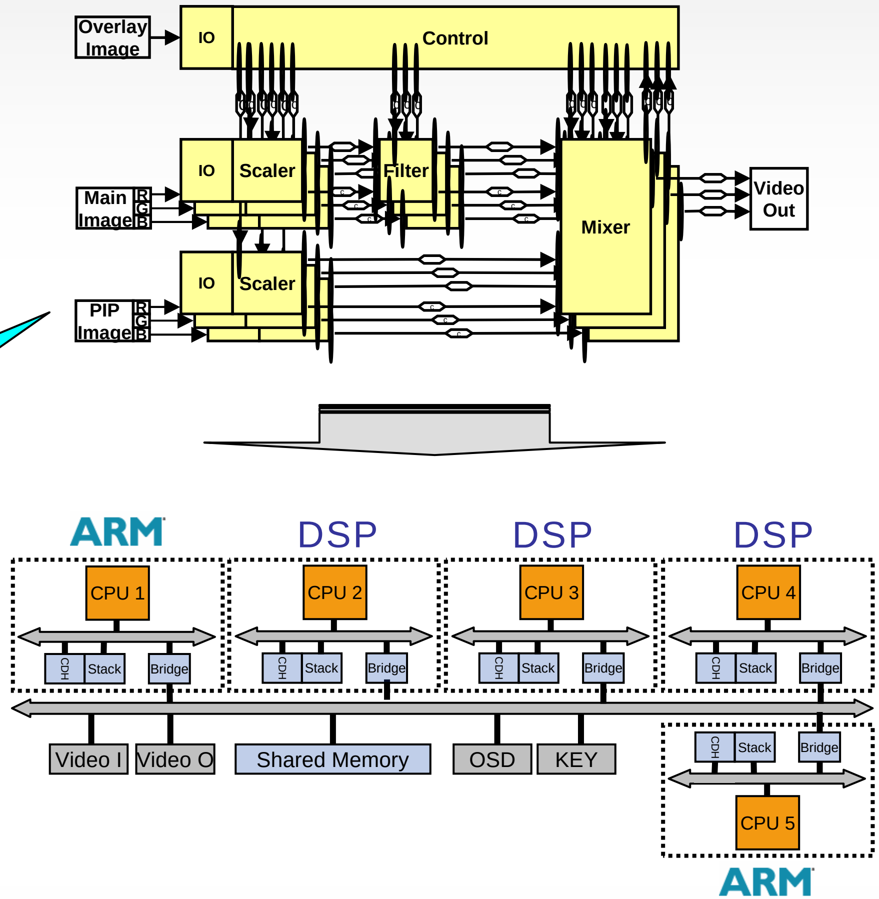


Future development work is all MP



ARM roadmap slides from ARM DevCon 2008

But, To ARM's Dismay, the Future (and Present) is Heterogeneous

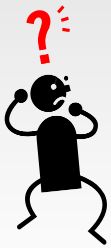


Picture in picture video

Multicore SoC Programming

Challenge: Parallel Software on Parallel Hardware

MULTICORE DESIGN SIMPLIFIED imperas



1. Which programming model and methodology do I adopt for my customers…?

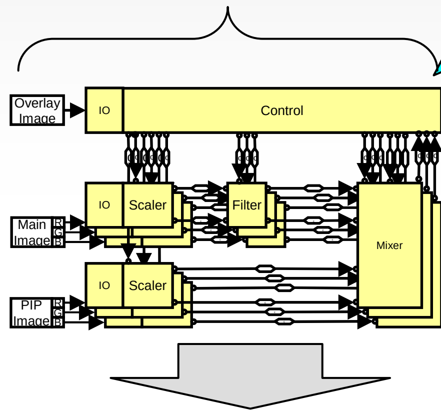
4. How do I develop and verify a range of reference applications for an MPSoC?

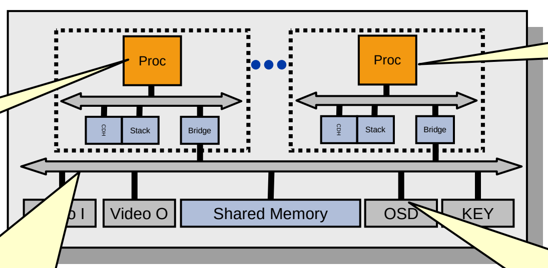
5. Can I automate the software verification?

2. How do I parallelize the application?

3. How do I build a programming environment?

6. How do I deliver the software and programming environment to customers?





How do I model my custom CPU?

Which CPU do I choose?

Is this the right bus architecture?

Which hardware IP do I need to develop?

Hardware / Software Complexity Growth

MULTICORE DESIGN SIMPLIFIED
imperas

- From 1990 to 2000 chip gate capacity increased about 100x
- But design complexity increased by 100,000x
- This resulted in RTL functional verification taking about 70% of design resources
- New technologies, new methodologies
  - Constrained random generation
  - Coverage driven verification
  - Dynamic assertion checking
  - Better debugging tools

- Software code for embedded systems is doubling annually
- Amount of concurrency is doubling every 18 months
- Complexity of software driven by concurrency, shared resources
- New technologies, methodologies needed for functional and performance verification of embedded software
  - Simulation (virtual platforms)
  - ???
- Conclusion: borrow from the progress made in RTL functional verification

Embedded Software Verification Using Virtual Platforms

MULTICORE DESIGN SIMPLIFIED
imperas

- Embedded software/systems issues: complexity
- Key software development tasks
  - OS
  - Drivers
  - Applications
- Virtual platform infrastructure: necessary but not sufficient
- Requirements for software development

Embedded Software Tasks: Hardware-Related Software

MULTICORE DESIGN SIMPLIFIED
imperas

Boot OS | **Static Compiled Drivers** — **Dynamic Compiled Drivers (LKMs)** | **Bare Metal Applications**

- Which OS?
- New version?
- New port?
- SMP?
- AMP?
- Performance bugs

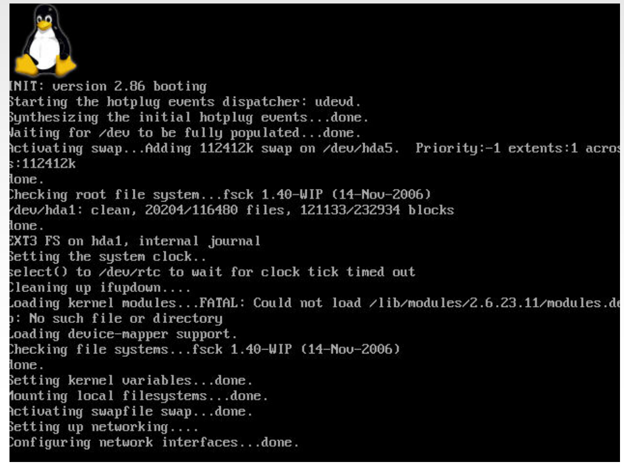
- New OS?
- New driver?
- How to verify functionality?
- How to debug core + peripheral?

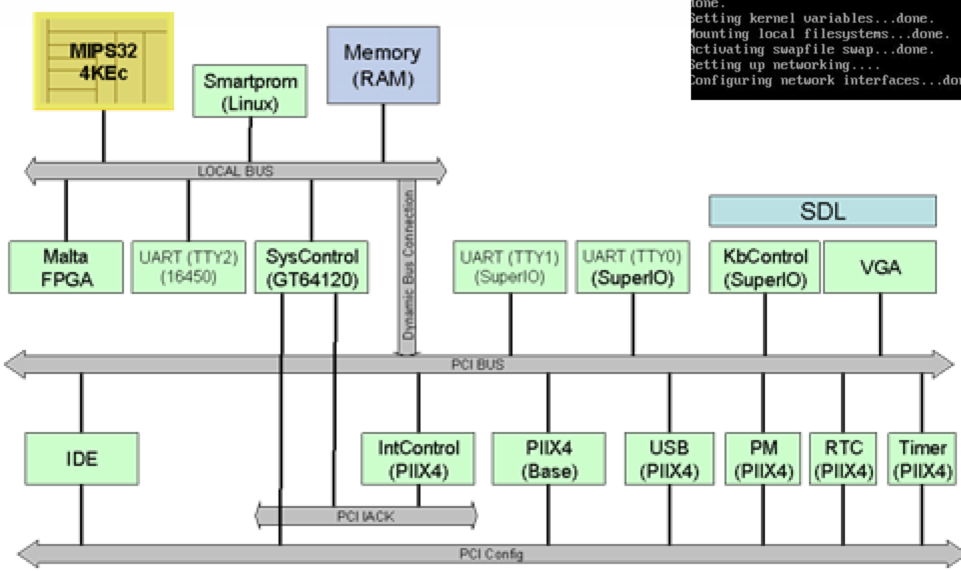
- Multicore?
- Shared memory?
- Functional bugs
- Performance bugs

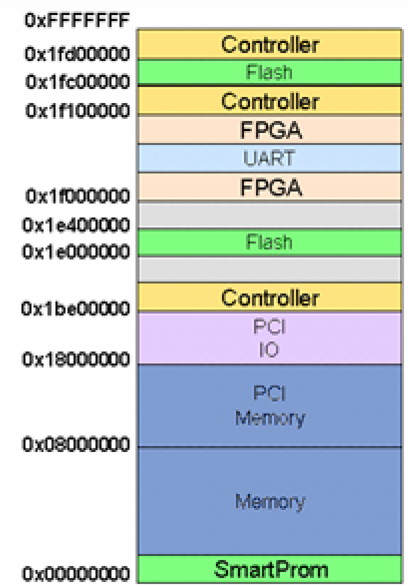
OS Issues

MULTICORE DESIGN SIMPLIFIED
imperas

- How to …
  - Port OS to new platform
  - Port new version of OS
  - Go from single core to multithread/multicore OS
  - Analyze performance issues





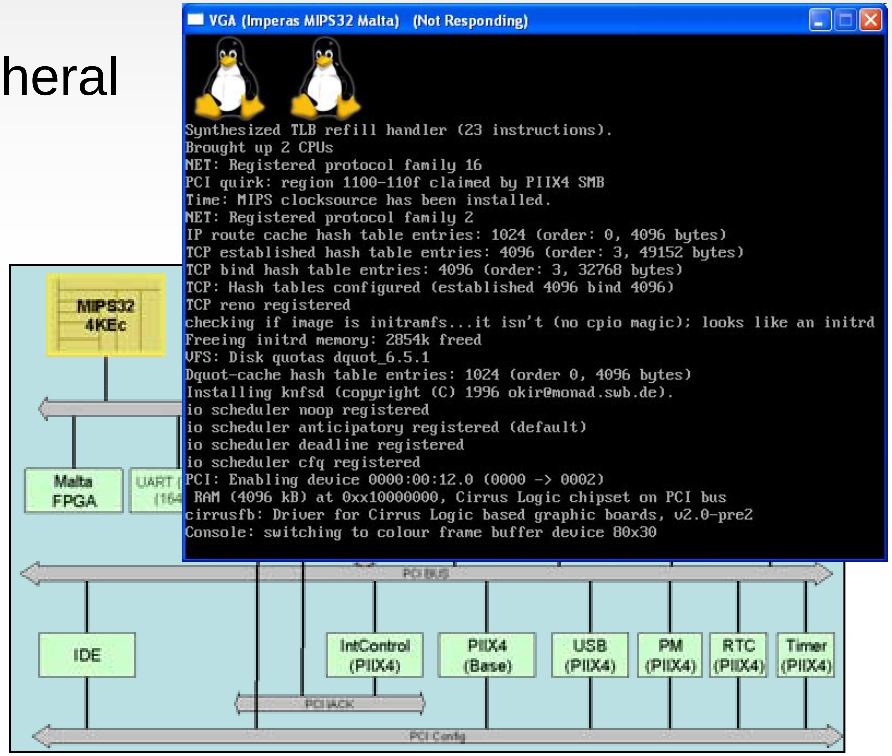


Boot OS | **Compiled Drivers** — **LKM Drivers** | **Bare Metal Applications**

Static & Dynamic Driver Issues

MULTICORE DESIGN SIMPLIFIED imperas

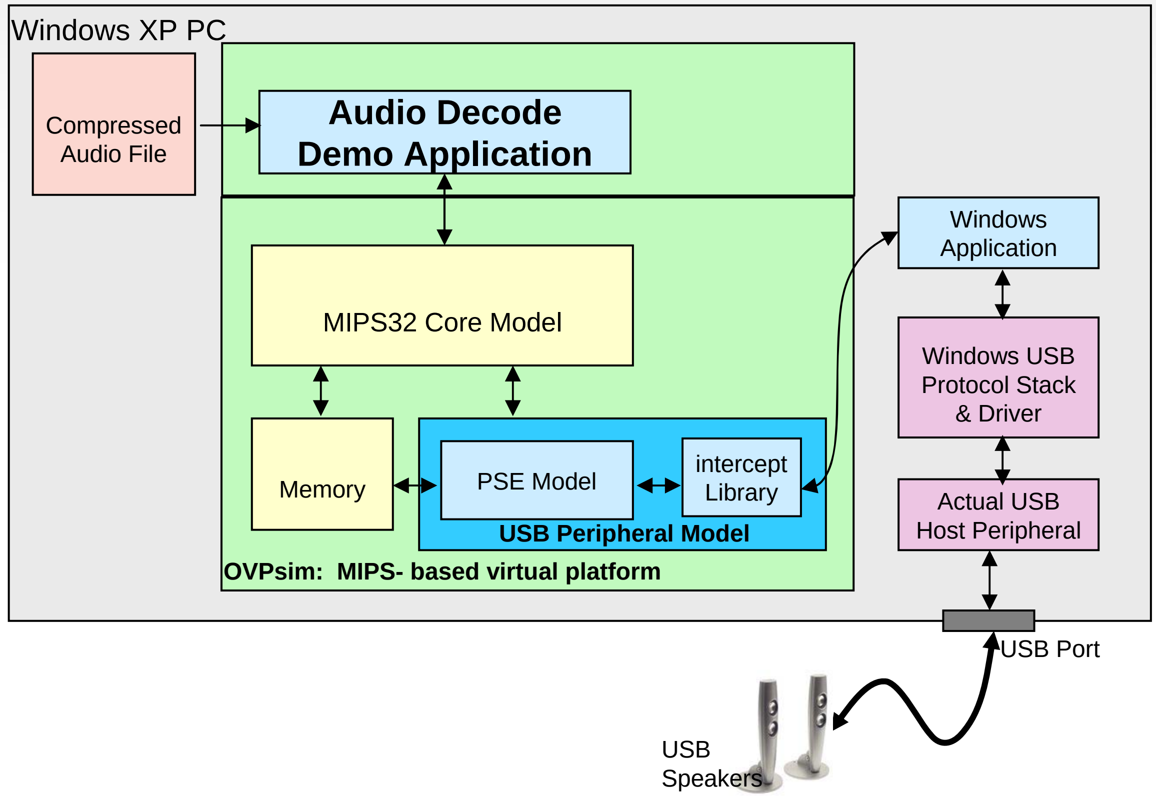
- How to …
  - Port driver to new OS
  - Develop driver for new peripheral
  - Verify driver functionality
  - Debug core + peripheral simultaneously



Boot OS	Compiled Drivers — LKM Drivers	Bare Metal Applications

Issues With Bare Metal Applications

- How to …
  - Analyze shared memory
  - Develop on multicore platforms
  - Find functional bugs
  - Find performance bugs

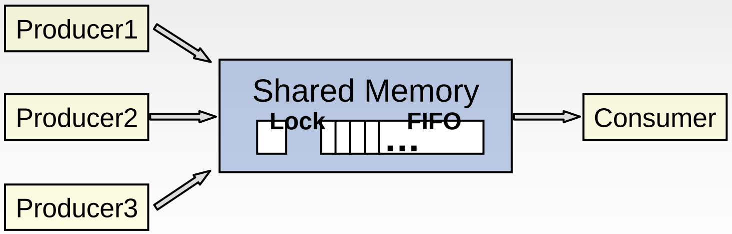


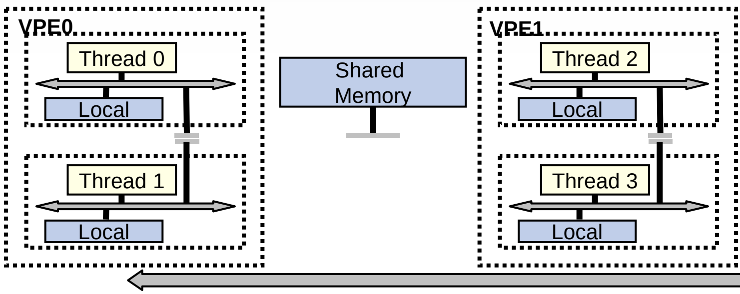
Boot OS | Compiled Drivers | LKM Drivers | **Bare Metal Applications**

Complex Platforms (AMP) Bring It All Together

MULTICORE DESIGN SIMPLIFIED imperas

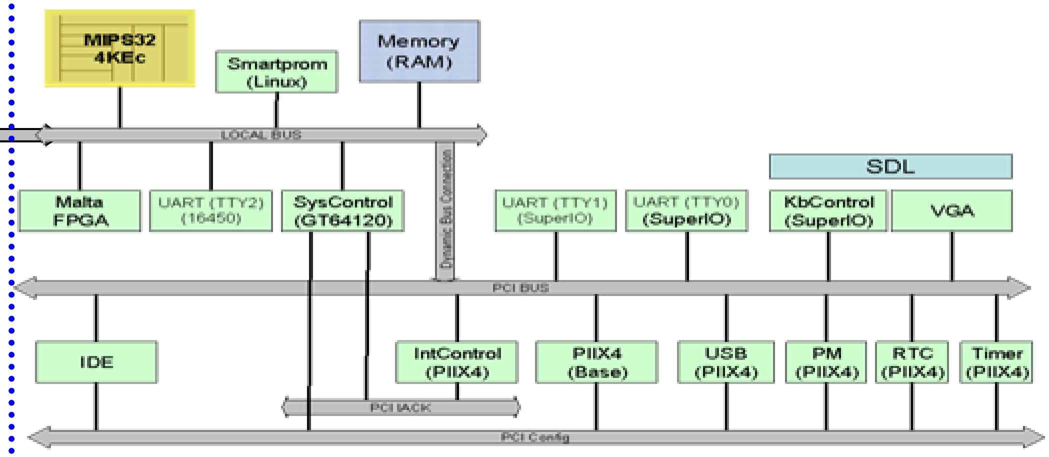
Data Processing





System Management

MP SoC users may only use 1 or 2 cores for Linux, the remainder for data processing applications



Boot OS

Compiled Drivers

LKM Drivers

Bare Metal Applications

Embedded Software Verification Using Virtual Platforms

MULTICORE DESIGN SIMPLIFIED imperas

- Embedded software/systems issues: complexity
- Key software development tasks
  - OS
  - Drivers
  - Applications
- Virtual platform infrastructure: necessary but not sufficient
- Requirements for software development

Simulation is Necessary

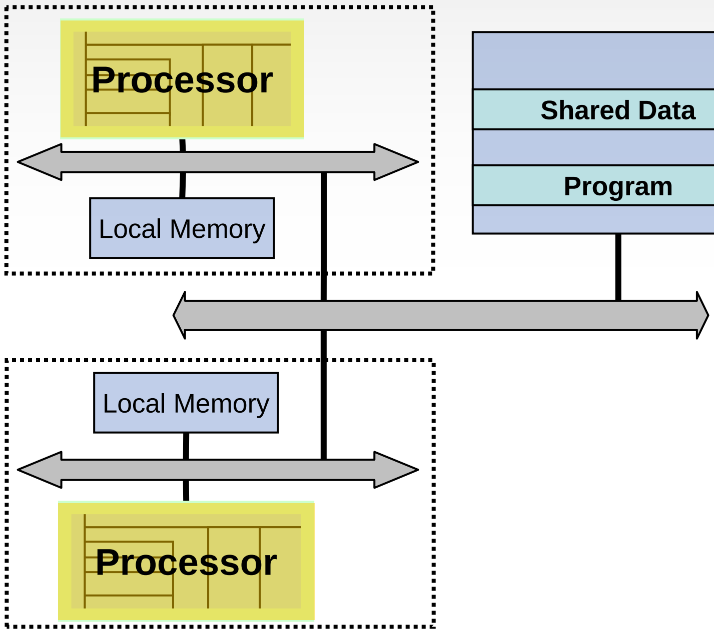
- Unimaginable to build a SoC today without simulation of the hardware
- Similarly, embedded software needs simulation as part of the development flow
  - Earlier start to development
  - Easier to find/fix bugs

- Virtual platform needs
  - Open source models developed in non-proprietary language
  - Easy to develop new models
  - Fast simulation

Open Virtual Platforms (OVP)

- **Modeling APIs**
  - Publishing of C OVP APIs for Processor, Peripheral, and Platform modeling
  - Documentation & header files
- **Open source library of models**
  - C source of models written to C OVP APIs
  - Processor models of ARM, ARC, MIPS, OpenRisc OR1K, x86, …
  - Peripheral models of standard embedded devices
  - Example embedded platforms in C, C++, SystemC, TLM-2.0
    - Including full platforms that boot operating systems like Linux, Nucleus
- **OVP reference simulator, free for non-commercial use**
  - Runs processor models fast, 500 MIPS typical
  - Interfaces to GDB via RSP/socket
  - MP capable, scalable and very efficient
  - Can encapsulate existing processor models (ISS)
  - Callable with C/C++/SystemC/TLM-2.0 wrapper
- **www.OVPworld.org**

OVPsim multicore2
2 Processor Platform (2x MIPS32)

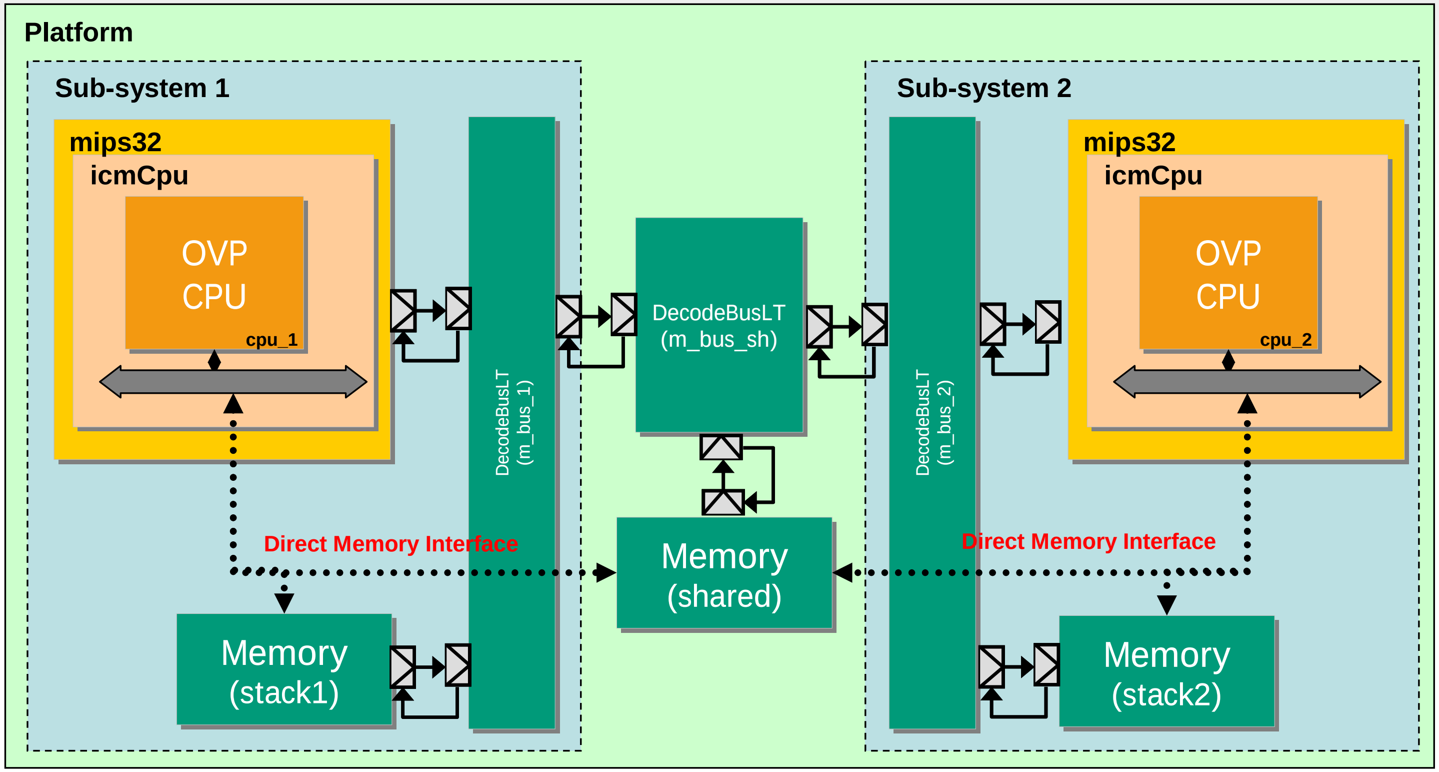


- Application is Fibonacci series generated on one and read by second processor from shared memory
- Local memory
  - heap and stack
- Shared memory
  - program and data

- Easy to create platform and use
  - C:\> platform .exe application .elf
  - Loads application into shared memory and runs it
  - On 3GHz PC runs up to 500 MIPS

OVP SystemC TLM 2.0

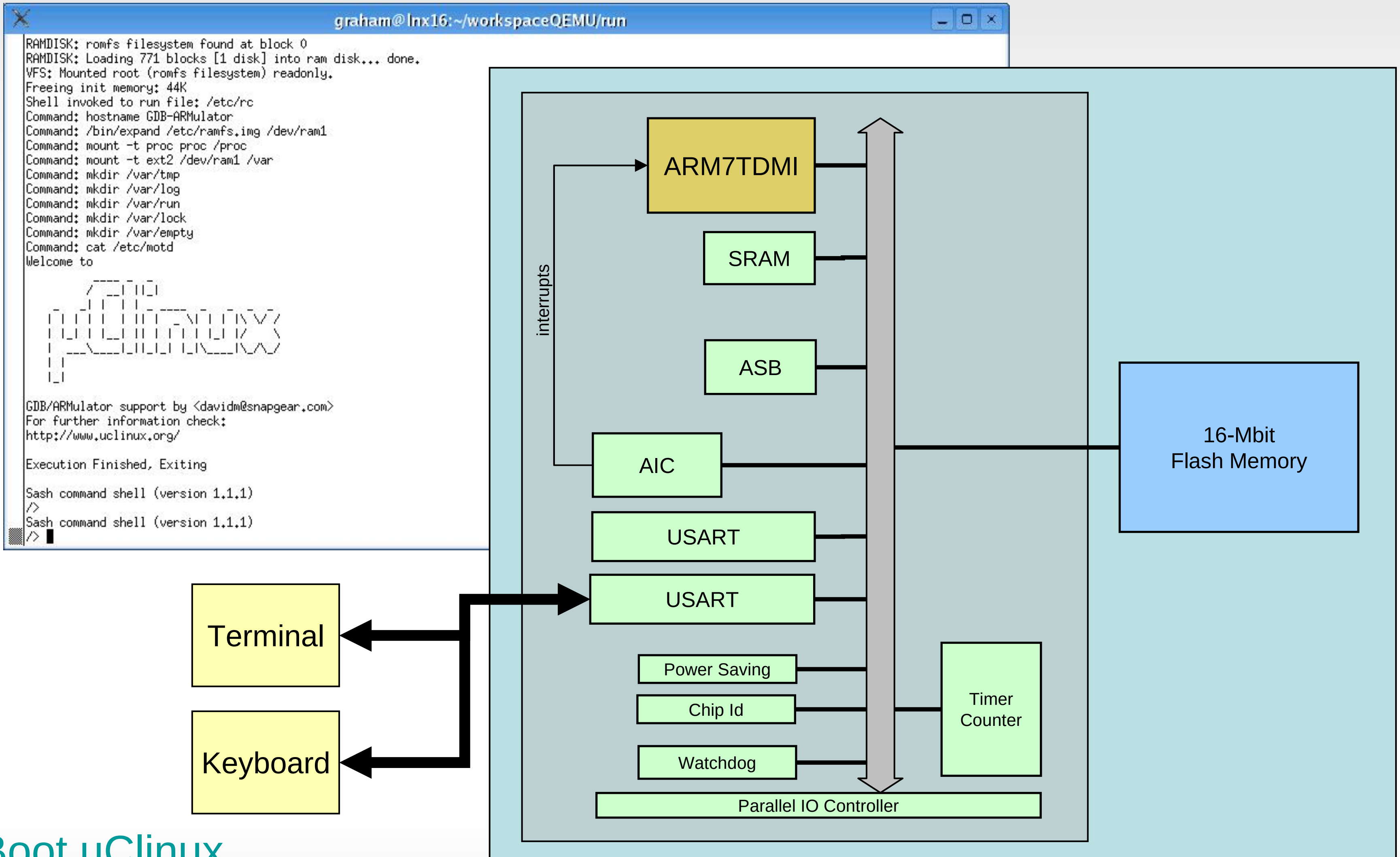
Demo OVPsim_multicore2_tlm2.0



- Local memory for heap and stack
- Shared memory for program and data
- Application is Fibonacci series generated on one processor, and read by second from shared memory
- Very simple to use and runs very fast

OVPsim booting uClinux on ARM Atmel AT91sam7

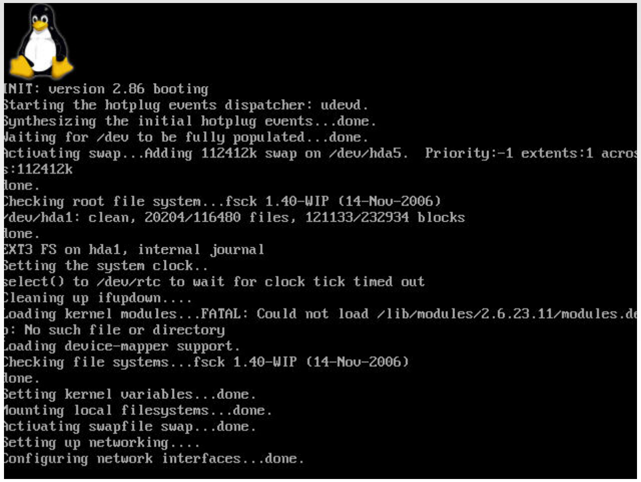
MULTICORE DESIGN SIMPLIFIED
imperas

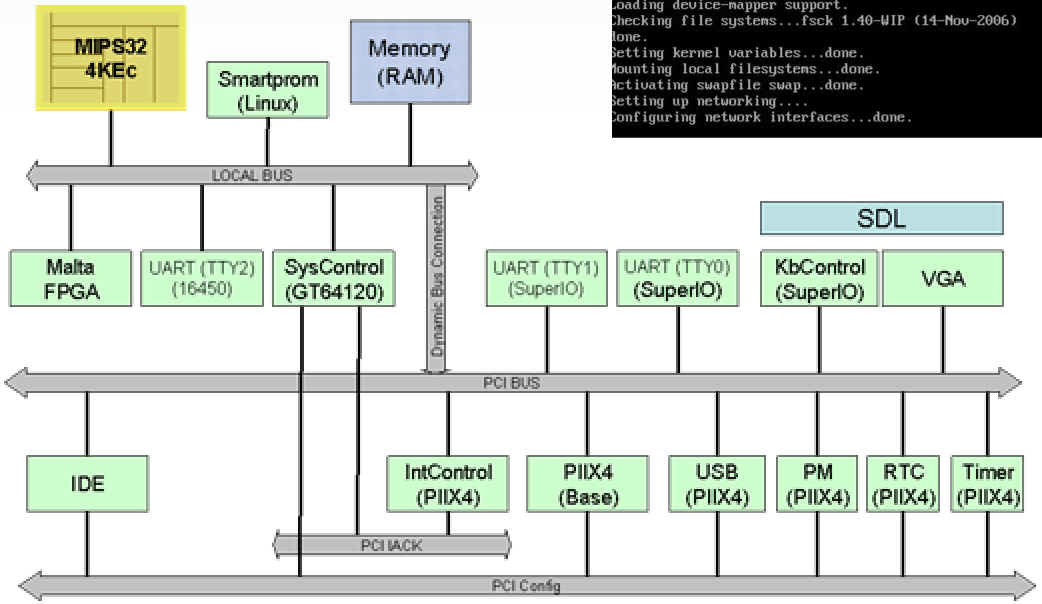


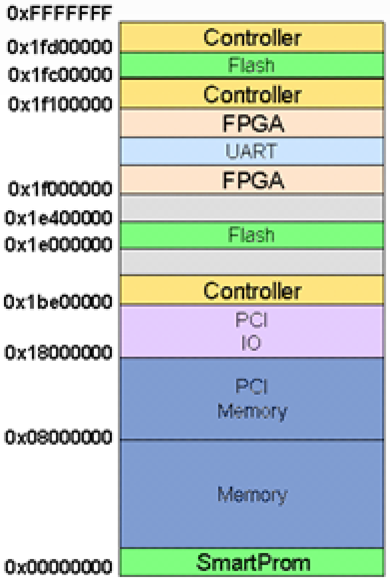
Boot uClinux

OVPsim MIPS Linux platform

MULTICORE DESIGN SIMPLIFIED
imperas

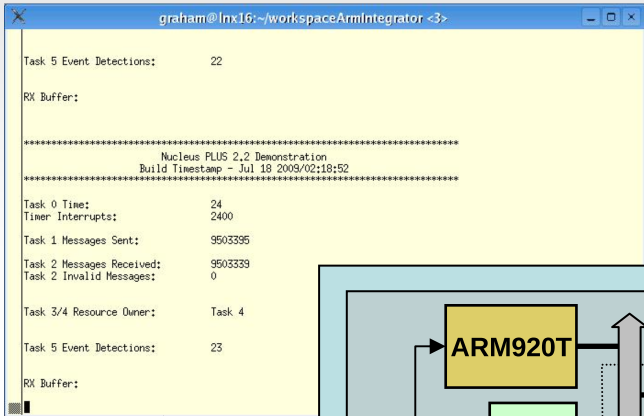




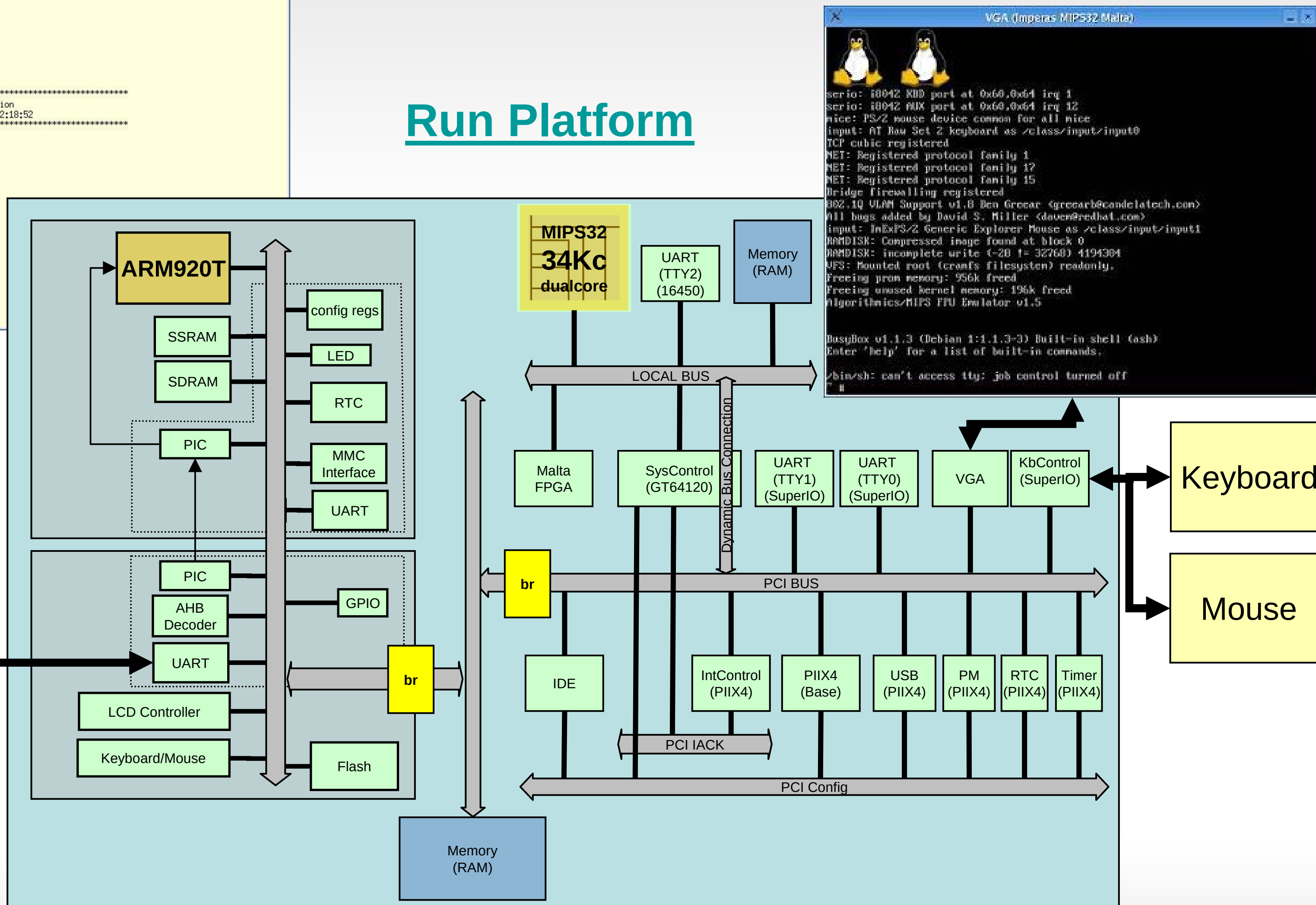


- Boot Linux on Windows PC…
- Easy to run: platform .exe vmlinux

OVPsim Heterogeneous Platform
ARM Nucleus / MIPS SMP Linux



Run Platform



telnet localhost 9999

Case Study 1: Automotive Electronics

MULTICORE DESIGN SIMPLIFIED
imperas

- Key need is running more simulations
- Platform is simple microcontroller based on ARM7
- Simulation speed of the vendor simulator was too slow for complete regression runs in reasonable amount of time
- Used Open Virtual Platforms to achieve 50x simulation speed improvement

- Simulation speed is enough

Embedded Software Verification Using Virtual Platforms

MULTICORE DESIGN SIMPLIFIED imperas

- Embedded software/systems issues: complexity
- Key software development tasks
  - OS
  - Drivers
  - Applications
- Virtual platform infrastructure: necessary but not sufficient
- Requirements for software development

Case Study 2:
Home Entertainment System

MULTICORE DESIGN SIMPLIFIED
imperas

- Original SoC
  - 1 general purpose core
  - Running Linux 2.6.n
  - 12 proprietary DSPs
  - 3 DDR1 memory banks
    - 1 dedicated to general purpose core
- Cost Reduction SoC
  - 1 general purpose core
  - Running Linux 2.6.n+m
  - 2nd general purpose core for housekeeping
  - 12 proprietary DSPs
  - 2 DDR2 memory banks
    - Not dedicated to general purpose core

- Cost reduction SoC has hardware fully verified, uses existing software
- Chip brought up in lab, fully functional, but …
- Runs at half the speed in about 30% of the operating scenarios
- Insufficient diagnostics on chip for debug
- Simulation platform built, but by itself not able to debug problem

Simulation Is Necessary, But Not Sufficient

MULTICORE DESIGN SIMPLIFIED
imperas

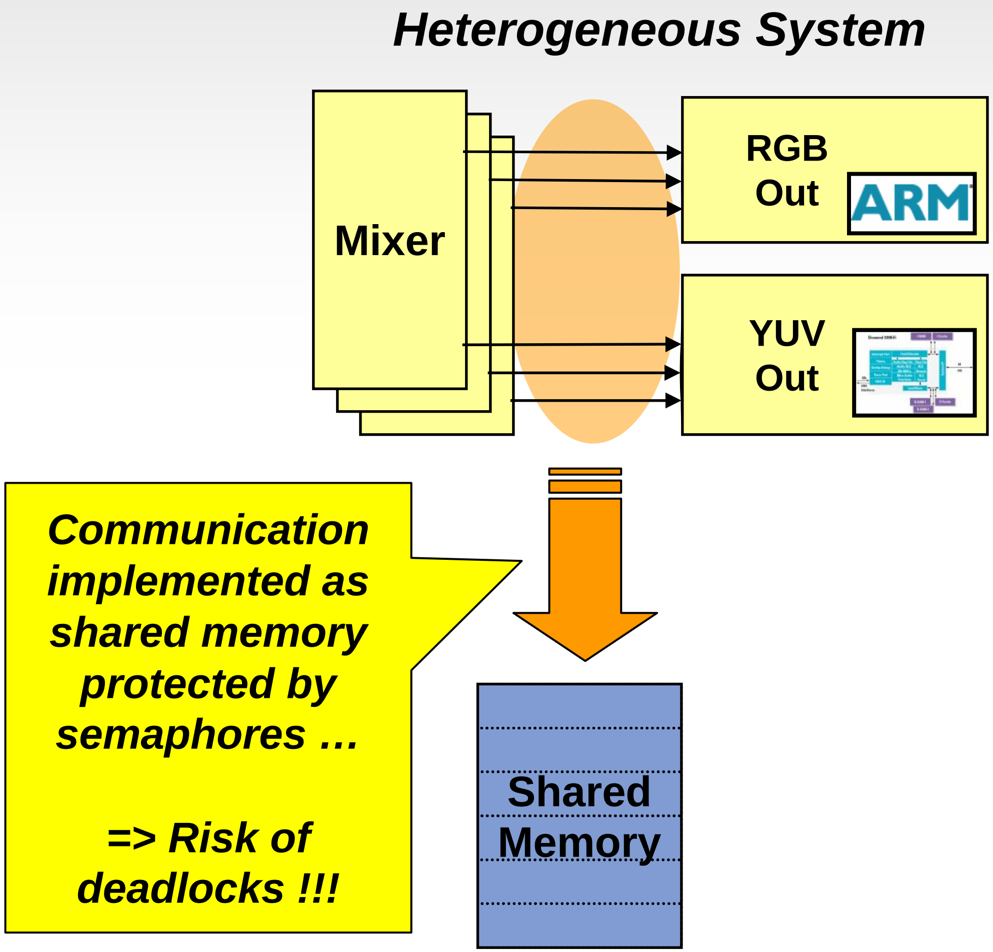
- Simulation is only the infrastructure
- Just like on the hardware side, with more complexity in the software, new tools and methodology are needed
- Need to find functional bugs and performance bugs
- Need new metrics

New Tools/Methodologies

- Non-intrusive: cannot have “heisenbugs”
- Work on heterogeneous, AMP platforms
- Address functional and performance bugs
  - Deadlock, race, stall, …
  - Cache thrashing, performance bottlenecks, …
- Provide introspection of the complete platform
  - Processors, peripherals, memory
- Metrics
  - Code coverage
  - Functional coverage
  - ???

Dynamic Assertion Checking

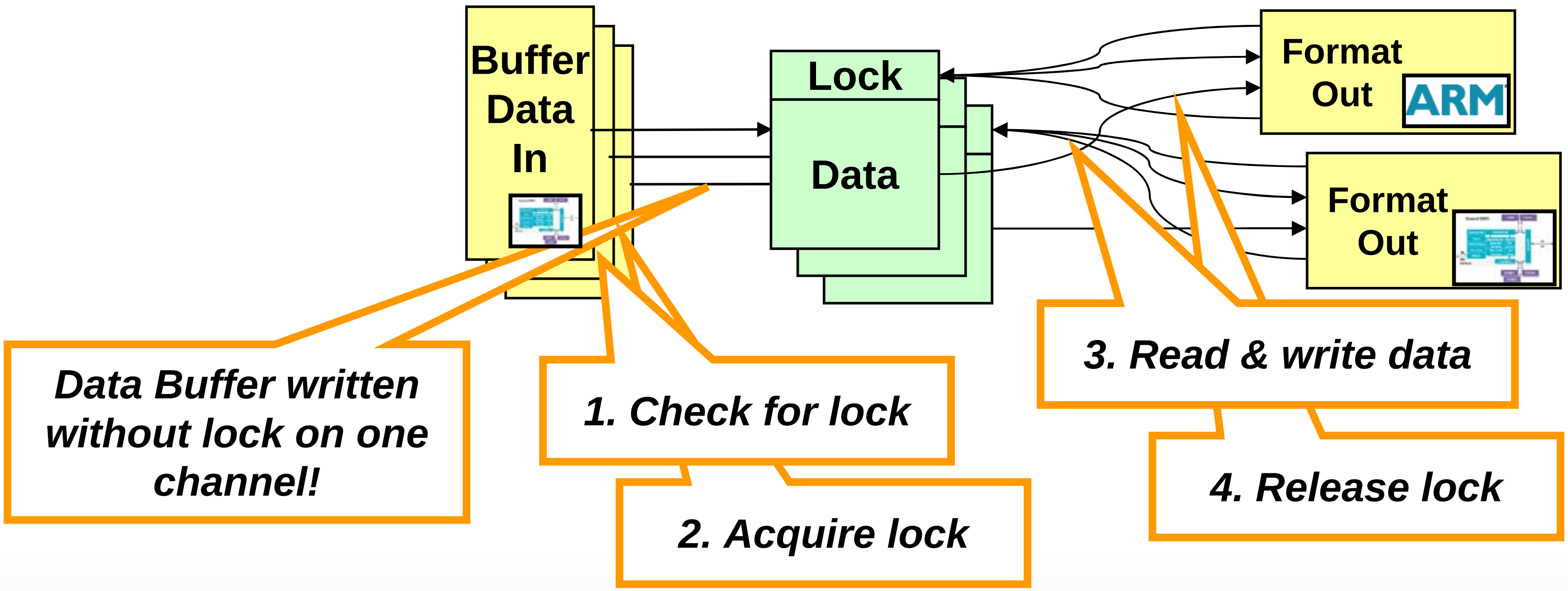
MULTICORE DESIGN SIMPLIFIED imperas



- Control Simulation
  - Processors and peripherals
- Force application into potential error state
- Create worst case conditions quickly

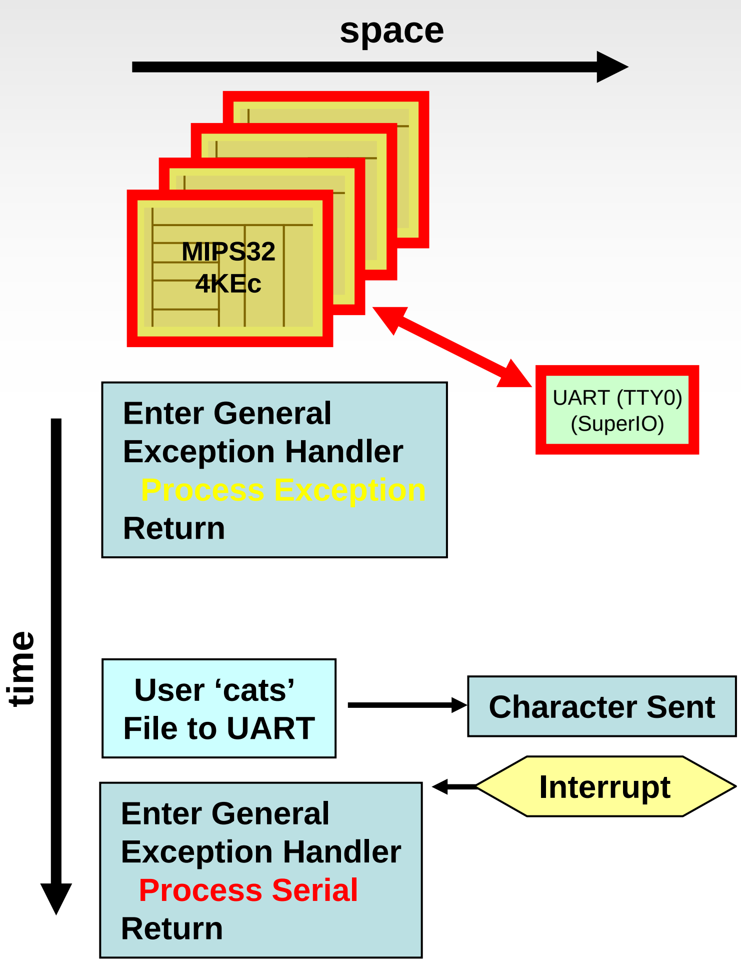
User Defined Dynamic Assertions

- Create a specific user assertion
  - ***Buffer is not accessed without a lock***
- Complex sequence of operations
- Over multiple processor cores



Platform Introspection for Debugging Driver Software

MULTICORE DESIGN SIMPLIFIED
imperas



- Debug the software running on processor(s) with all platform executing
- Introspection enables view of other platform components, including all registers
- Ability to switch from code on different processors onto code inside behavioral models
- Standard gdb features

MULTICORE DESIGN SIMPLIFIED

imperas

Conclusions

- Embedded software development is hard, and getting harder very quickly
- Need to recognize the real issues – the development tasks that need to be successful – and address those needs
- Virtual platforms (models and simulation) are necessary, but …
- New tools and methodologies, including metrics, are required