MULTICORE DESIGN SIMPLIFIED

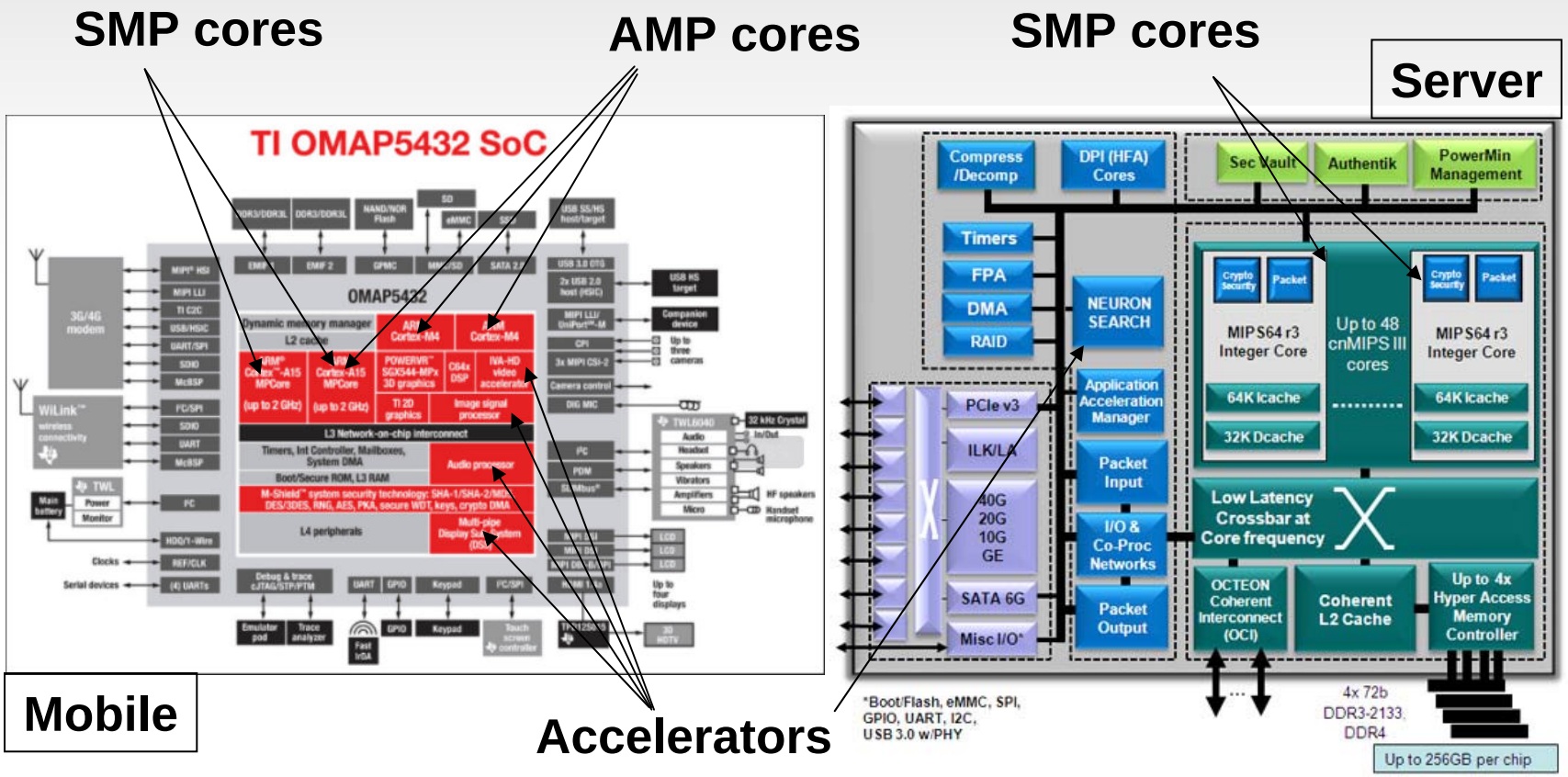
imperas

Parallel Simulation Accelerates Embedded Software Development, Debug and Test

Larry Lapides
Imperas Software Ltd.
larryl@imperas.com

Modern SoCs Have Many Concurrent Processing Elements

MULTICORE DESIGN SIMPLIFIED
imperas



Many simulated cores slow down traditional simulators

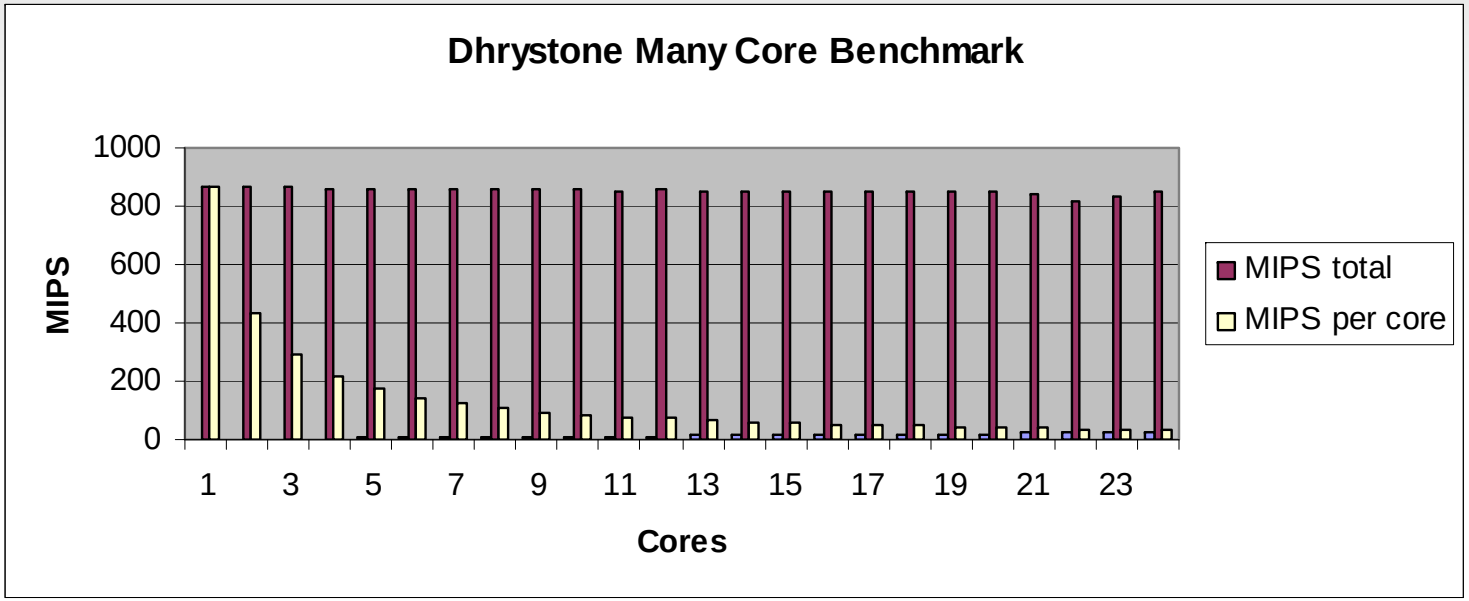
Simulation for Software Development is Becoming a Mainstream Methodology

MULTICORE DESIGN SIMPLIFIED
imperas

- Determinism, i.e. repeatable simulation results from the same simulation conditions
- Controllability, observability
- Ease of use
- Early availability, before silicon

- Virtual platforms are not replacing testing of software on hardware – whether actual hardware, development board, FPGA prototype – but complementing and supplementing traditional testing techniques

Multiprocessor Target Simulations



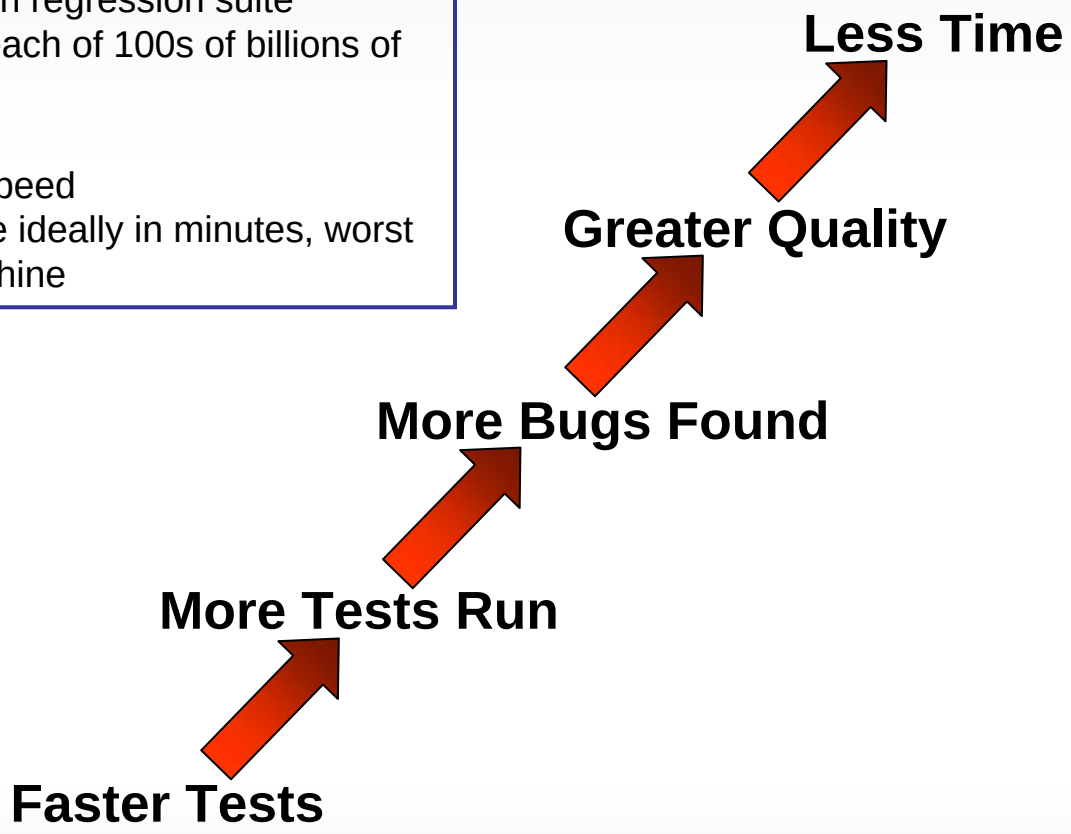
- 1-24 cores in platform, no communication, running same application
- Imperas simulation scales exceptionally well, as total simulation throughput stays constant
- However – each core gets only a pro-rated share of the overall simulation
  - With 10 cores simulating, Dhrystone runs at only 85 MIPS (10% of the total simulation speed) on each simulated core

Software Quality is Directly Proportional to Test Speed

MULTICORE DESIGN SIMPLIFIED
imperas

Customer Test Requirements

- Software Test Suites
  - Automotive: 1000s of tests in regression suite
  - Networking: 100s of tests, each of 100s of billions of instructions
- Test run time requirements
  - Tests should run near real speed
  - Test suite needs to complete ideally in minutes, worst case hours, on a single machine



The Software Simulation Performance Challenge

MULTICORE DESIGN SIMPLIFIED
imperas

- How to provide a simulation solution that scales well with the inclusion of many cores and hardware accelerator blocks in the virtual platform
- Reducing the per-core throughput is not an acceptable solution

- Needs to handle Symmetric MultiProcessor (SMP) architectures
- Needs to handle Asymmetric MultiProcessor (AMP) architectures
- Needs to handle hardware accelerator IP blocks

Parallel Simulation is the Obvious Answer, But History Shows Mixed Results

MULTICORE DESIGN SIMPLIFIED
imperas

- One problem that needed to be solved was simulation of networks of devices
  - Not multicore processors
  - Not so many multicore PCs, however, networks of PCs (compute farms) were available
  - Networks of devices are usually loosely coupled, with minimal communication between cores
  - Distributed simulation, splitting the simulation over a network of host machines, has provided a solution for this specific problem
- As multicore processors moved into the semiconductor mainstream, and multicore PCs moved into the compute mainstream, parallel simulation on a single host has been tried
  - As before, this is successful for loosely coupled or AMP architectures
  - For tightly coupled or SMP architectures, the cost of synchronization of the parallel simulation processes, in order to maintain determinism, cancelled out the benefits gained from parallelization

How Just-In-Time (JIT) Code Morphing Simulators Work

MULTICORE DESIGN SIMPLIFIED
imperas

- Code translation from target processor (e.g. ARM, MIPS, …) to host x86
- Quantum based simulation
  - All processors simulate within a quantum, then peripheral models are executed, i.e.
    1) CPU0 simulates N instructions
    2) CPU1 simulates N instructions
    3) CPU2 simulates N instructions
    4) CPU3 simulates N instructions
    5) Peripheral activities are simulated
    6) Time is then advanced to the next quantum and processor simulation starts again
  - Code translations are stored in a code "dictionary" to speed simulation during the quantum
  - Each quantum is typically 500 – 10,000 instructions
    - Quantum too small: slower simulation
    - Quantum too large: too many interrupts per quantum
  - Interrupts will cause the simulation to stop during the quantum so that the interrupt code can be executed

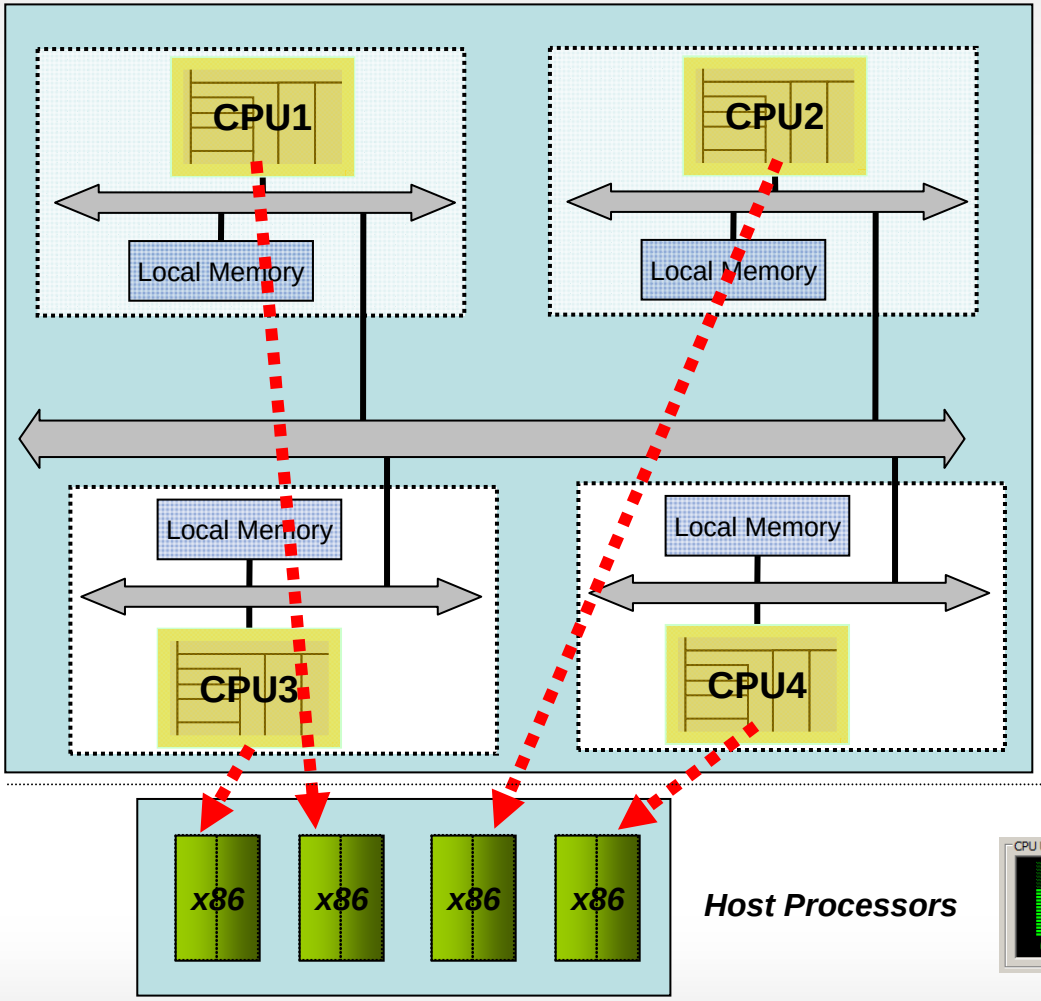
Parallel Simulation Concept

MULTICORE DESIGN SIMPLIFIED
imperas

- Assign simulation threads for each processor model within a quantum to a x86 core on the host
- To maintain determinism, need to identify points where synchronization is needed
  - Even within a quantum
- Imperas: invented a new synchronization algorithm
- MultiProcessor target on MultiProcessor host (MPonMP) technology
- QuantumLeap is the product name

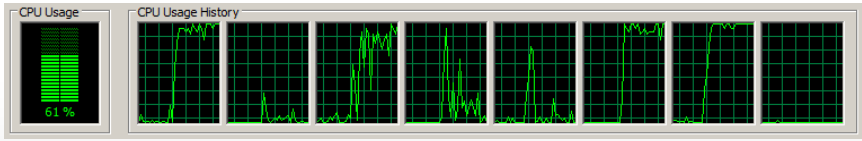
Imperas QuantumLeap (AMP System)

MULTICORE DESIGN SIMPLIFIED
imperas



Simulation

Host Processors



QuantumLeap Overview

MULTICORE DESIGN SIMPLIFIED
imperas

- Applicable for SMP and AMP system architectures, and for virtual platforms with hardware accelerator IP blocks
- Both uniform and non-uniform memory architecure (NUMA) configurations are supported
  - Memory can be shared or private to cores in any way
  - Sharing can be at any granularity, down to a single byte
- Works with both traditional memory locking schemes, based on test-and-set idioms, and scalable locking schemes, based on speculate/commit or load/store exclusive idioms
- Only minimal changes needed by model developers
  - For example, the Imperas Open Virtual Platforms (OVP) ARM processor model family of ~45 different core variants is implemented by approximately 100,000 lines of C source code; fewer than 10 lines of changes were required to enable QuantumLeap usage
  - Now the same model is used for both single thread and parallel simulation
- QuantumLeap parallel simulations are deterministic

QuantumLeap Features & Limitations

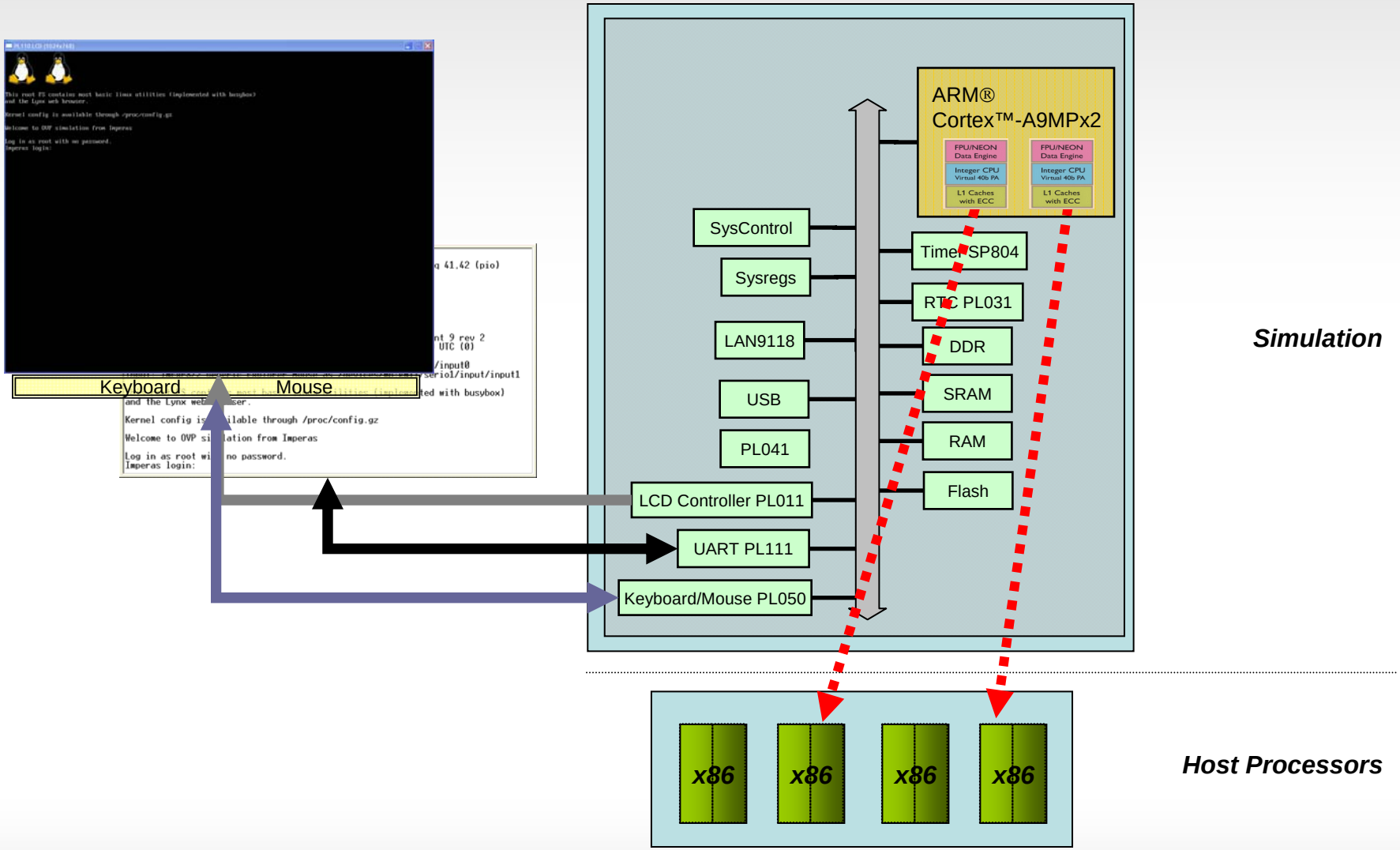
MULTICORE DESIGN SIMPLIFIED
imperas

- Requires no changes to user source code
- Works as normal with debuggers, IDE, Imperas software development, debug and test tools, …
- Scales well with increasing number of host x86 cores

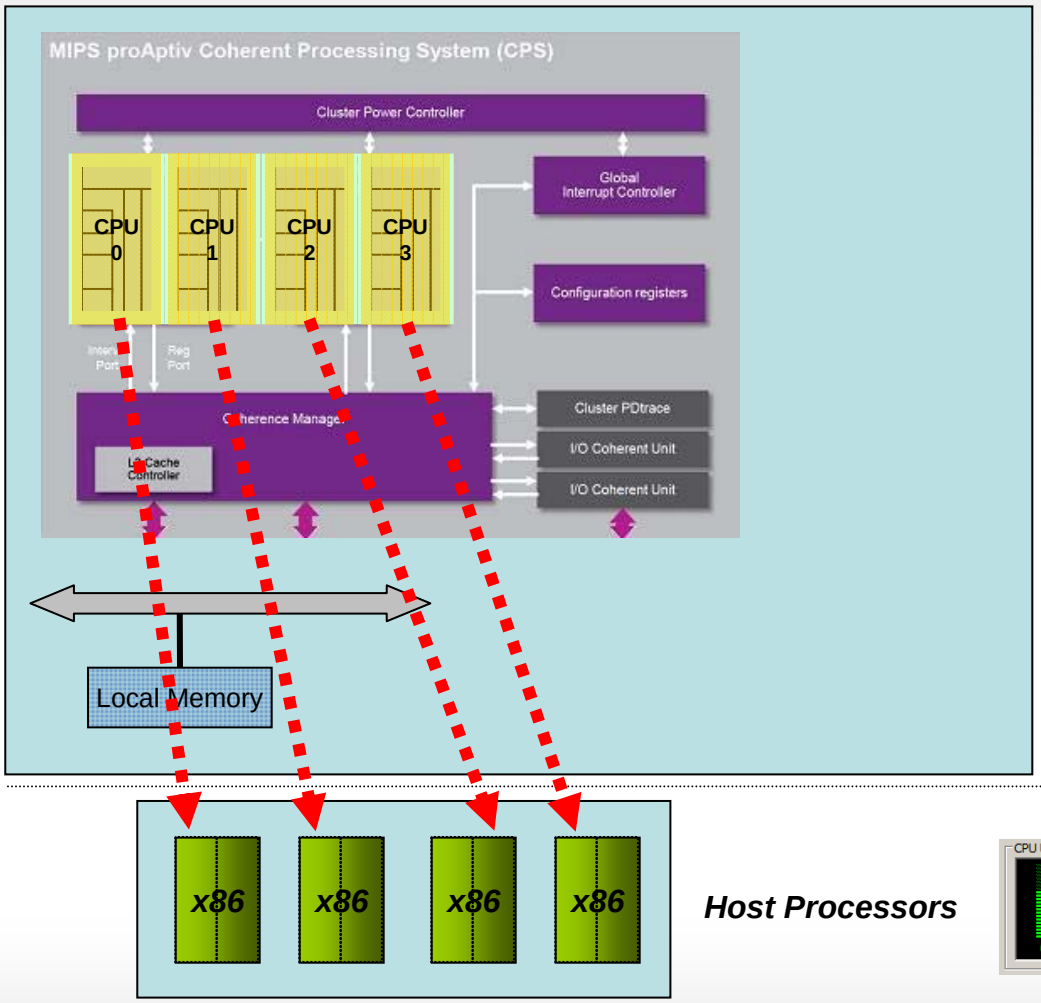
- Limitations
  - SystemC: only works for SMP architectures
    - ARM's big.LITTLE architecture will work also
  - All shared data access must be controlled by synchronizing instructions to achieve determinism

Imperas QuantumLeap (SMP System)
ARM Cortex-A9MPx2 Linux SMP Kernel

MULTICORE DESIGN SIMPLIFIED
imperas

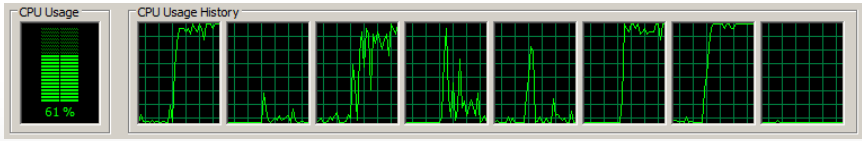


Imperas QuantumLeap (SMP System)
MIPS proAptiv 4 Core Configuration



Simulation

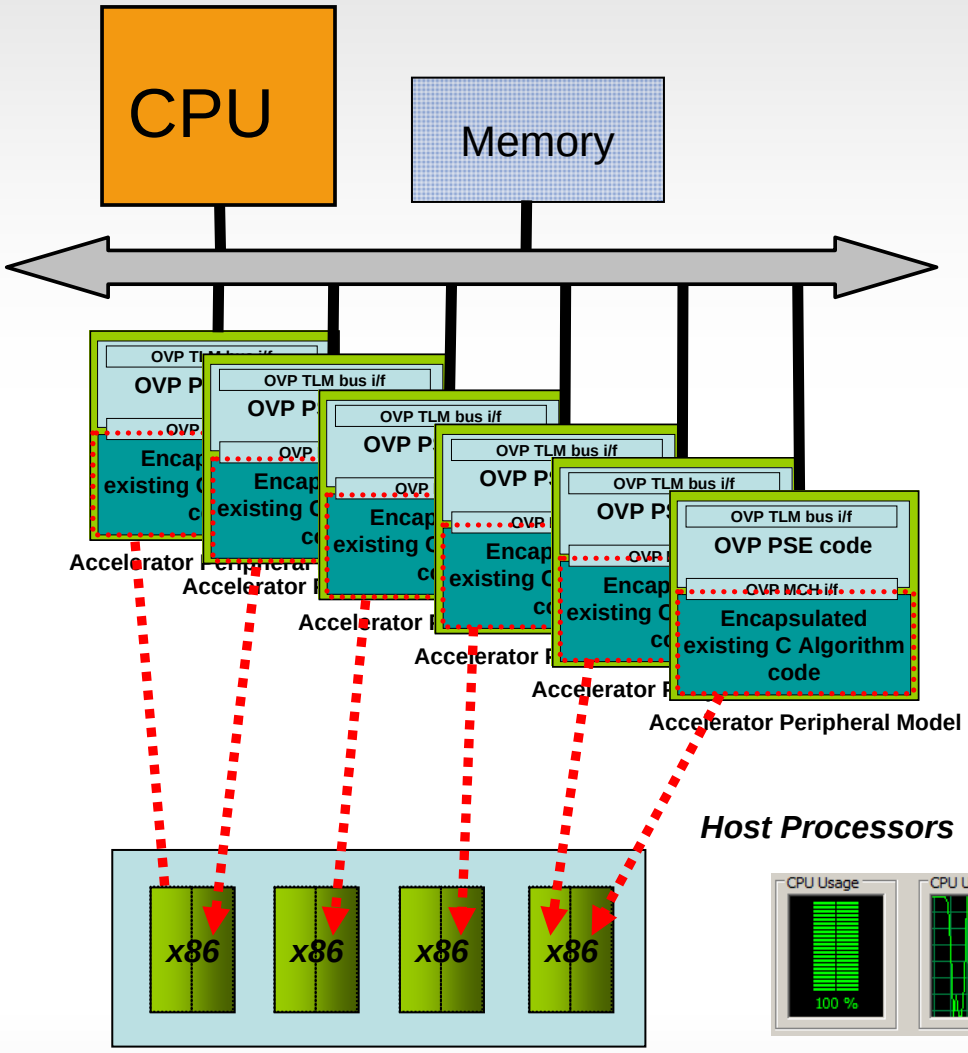
Host Processors

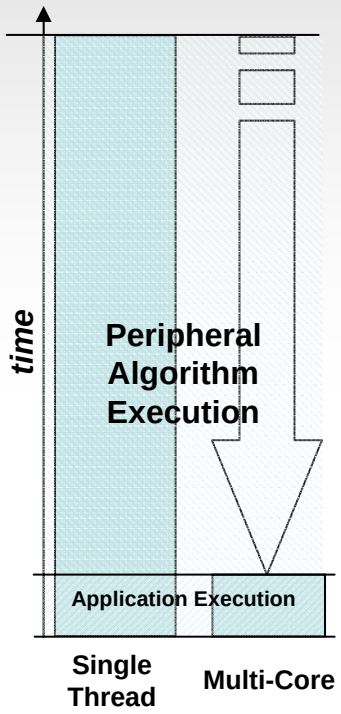


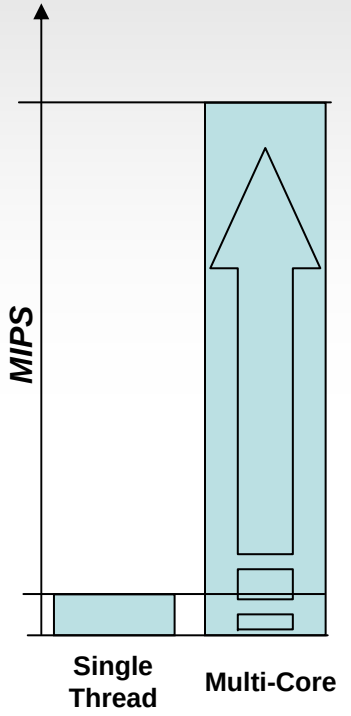
Algorithm Acceleration

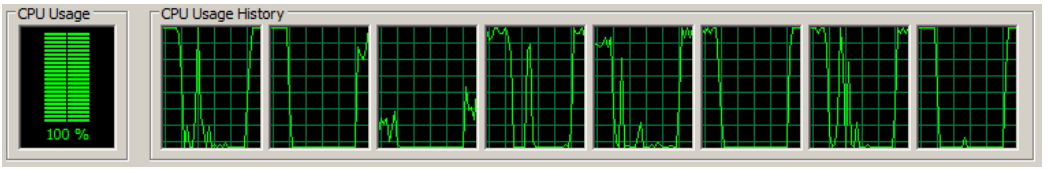
Peripheral Model Parallel Execution

MULTICORE DESIGN SIMPLIFIED
imperas









QuantumLeap Results

MULTICORE DESIGN SIMPLIFIED
imperas

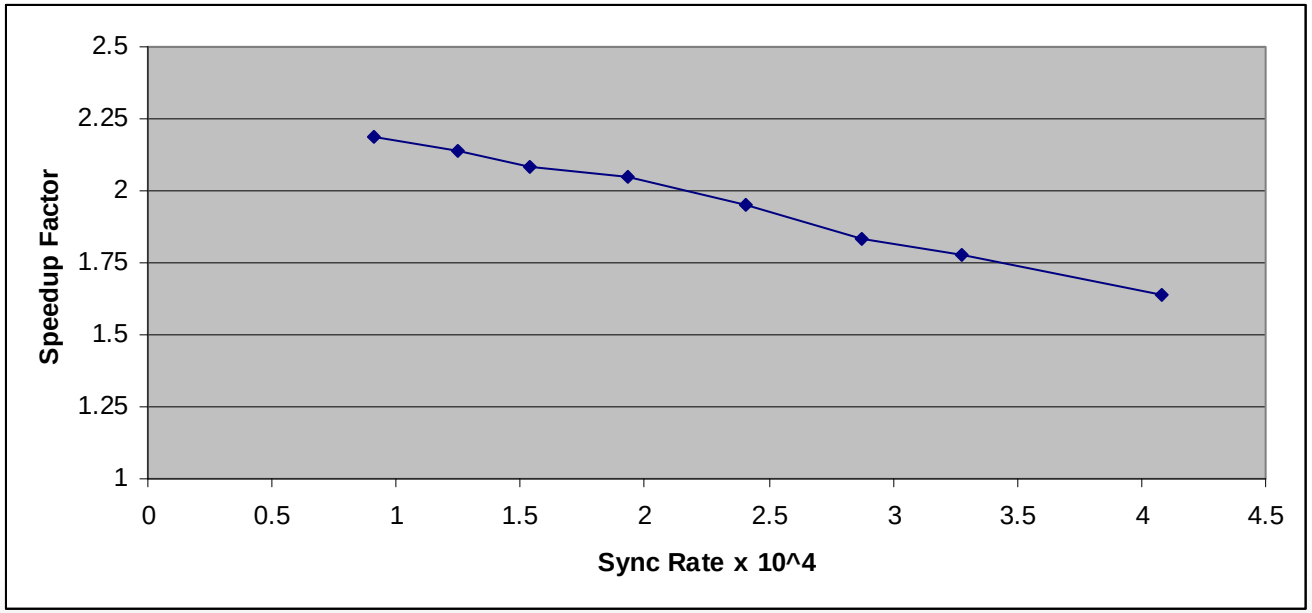
- Bare metal tightly coupled application: ARM parallel prime search (from DS-5 package)
  - Performance increase ranges from 1.6x to 2.2x depending upon starting point (affects synchronization frequency)
  - ARM Cortex-A57MPx4
- SMP Linux boot
  - Negligible speed up from QuantumLeap: SMP Linux has minimal parallelization
  - ARM Cortex-A57MPx4
- Applications running on Linux
  - Over a range of applications, speedup is > 2.5x
  - ARM Cortex-A57MPx4
- AMP applications
  - Speedup ~3.5x for 4 processor virtual platform running on 4 core host PC
  - MIPS proAptiv

- Host: 3.5Ghz QuadCore Intel i7-4770K

Parallel Simulation Speedup is Affected by Synchronization Frequency

MULTICORE DESIGN SIMPLIFIED
imperas

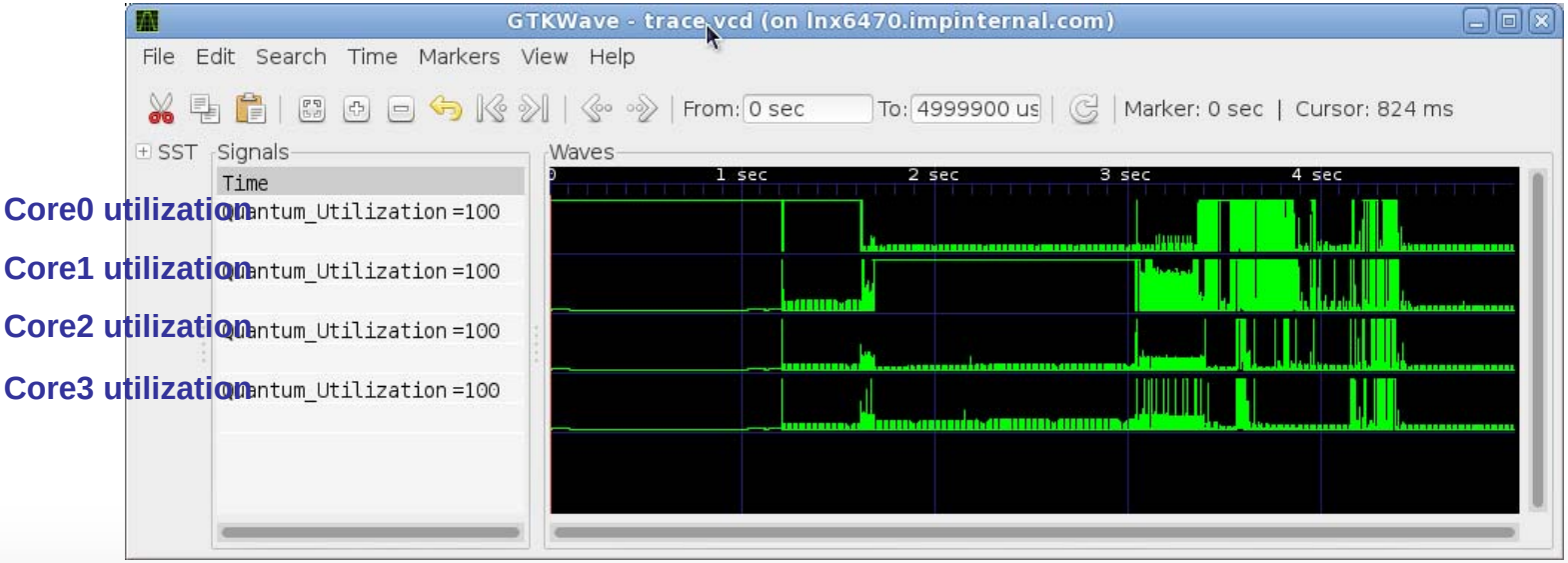
- Run tightly coupled parallelized application to find prime numbers
- Different starting points provide different synchronization frequencies, since as the starting point increases the distance between primes increases (frequency decreases)
- As expected, the speedup due to parallel simulation decreases due to increased sync frequency



Parallel Simulation Speedup is Limited by the Parallel Nature of the Software

MULTICORE DESIGN SIMPLIFIED
imperas

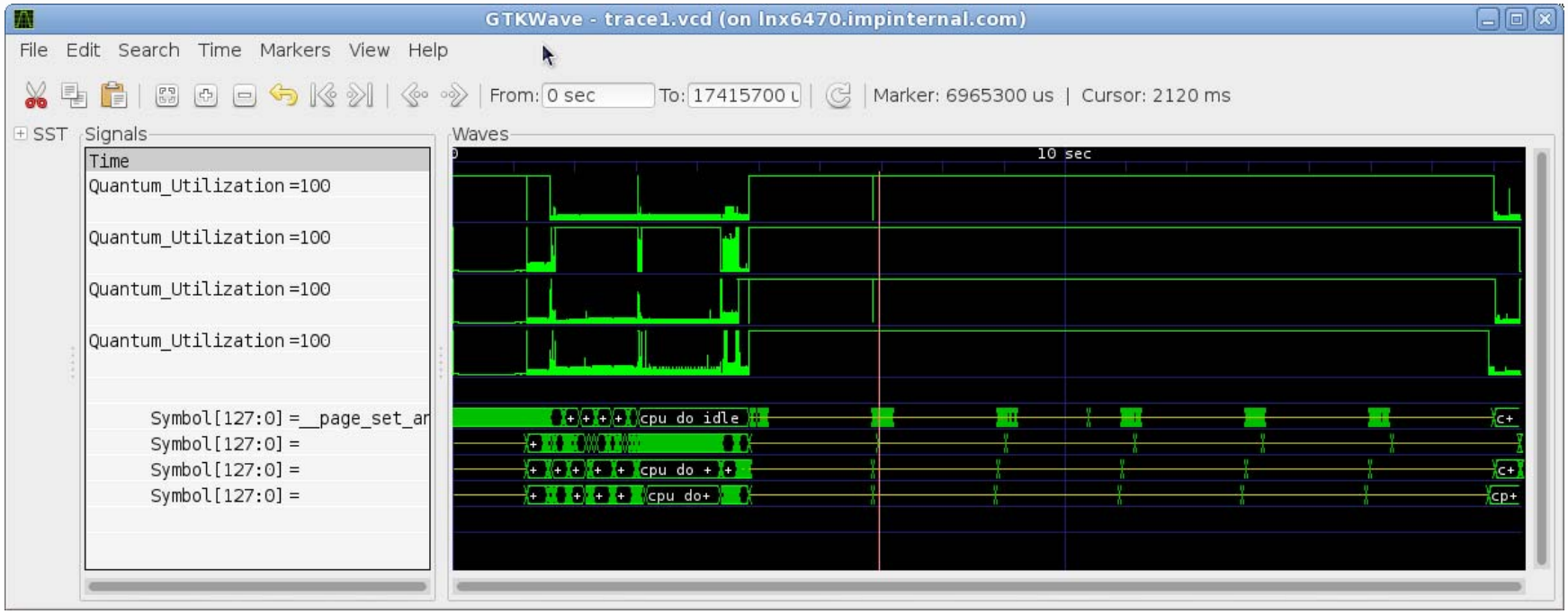
- Example: SMP Linux boot on ARM Cortex-A57MPx4
- SMP Linux is a SMP application, so we expect that it should be accelerated by parallel simulation
- However, there is no speedup observed!
- Actually, SMP Linux boots in a very serial manner



Parallel Simulation Speedup for Applications Running on Linux

MULTICORE DESIGN SIMPLIFIED
imperas

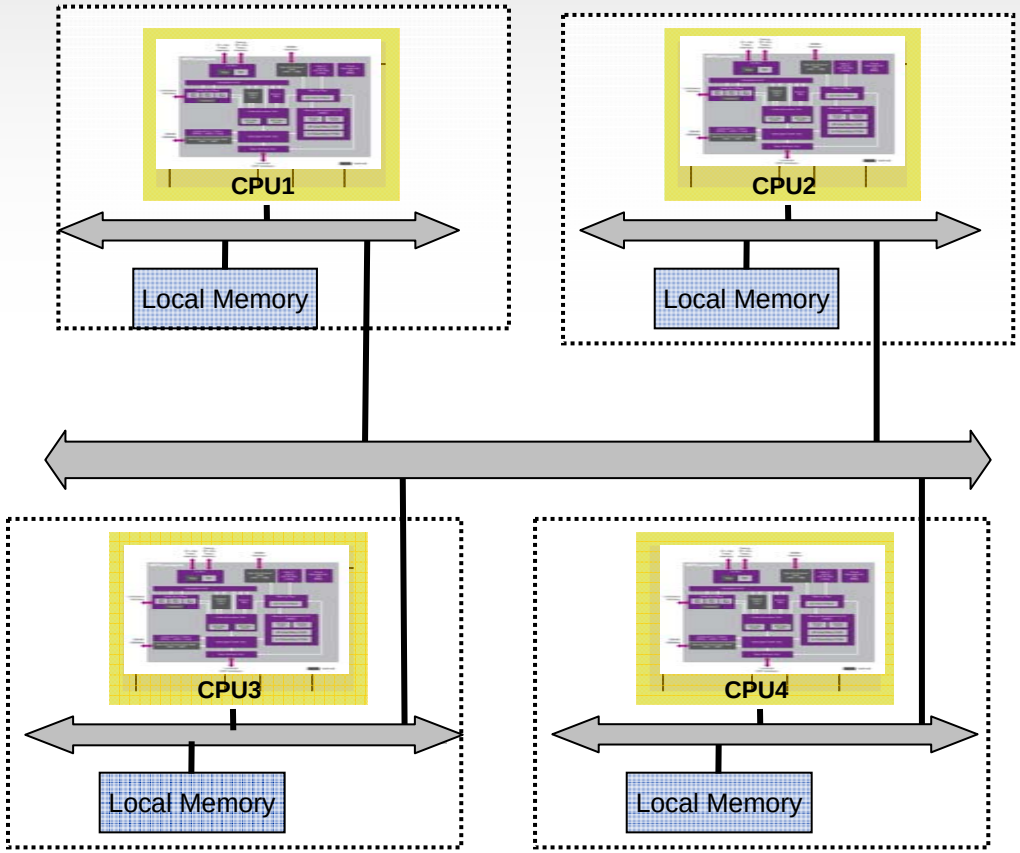
- Run Linux application benchmark on virtual platform: 4 different applications running simultaneously
- Linux allocates each application to a different processor in the virtual platform
- Trace shows Linux boot followed by applications running



- When Linux boot time is removed from data, QuantumLeap speedup is > 2.5x

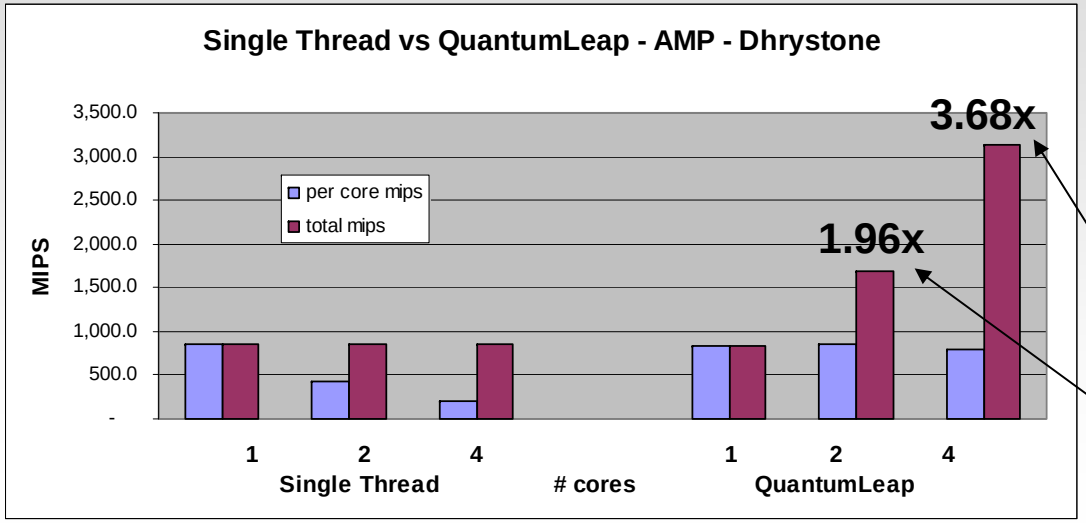
AMP - 4 Processor Platform
MIPS proAptiv

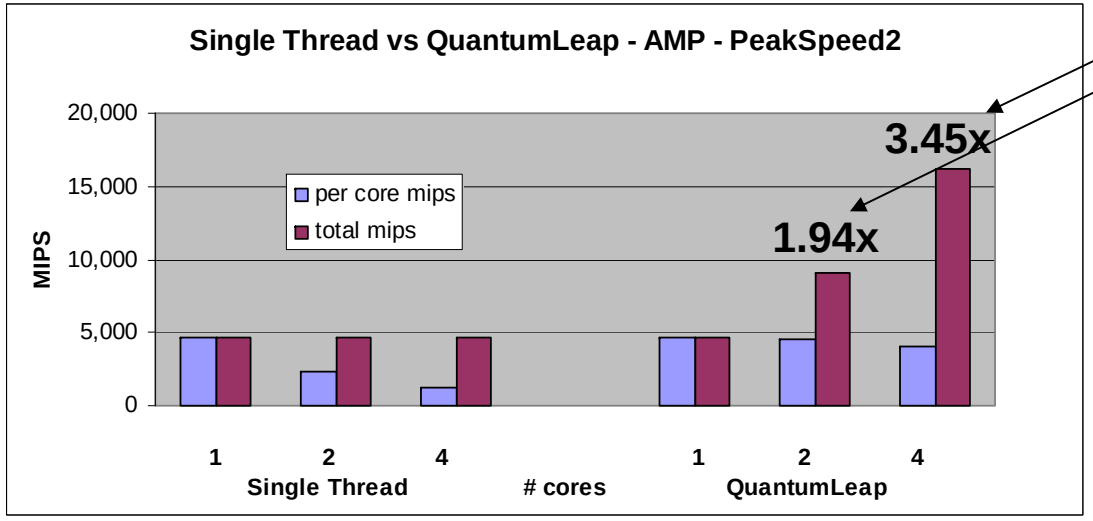
MULTICORE DESIGN SIMPLIFIED
imperas



- 4 independent cores
- Bare metal, each with own local program and data
- Minimal communication

QuantumLeap AMP Speedup



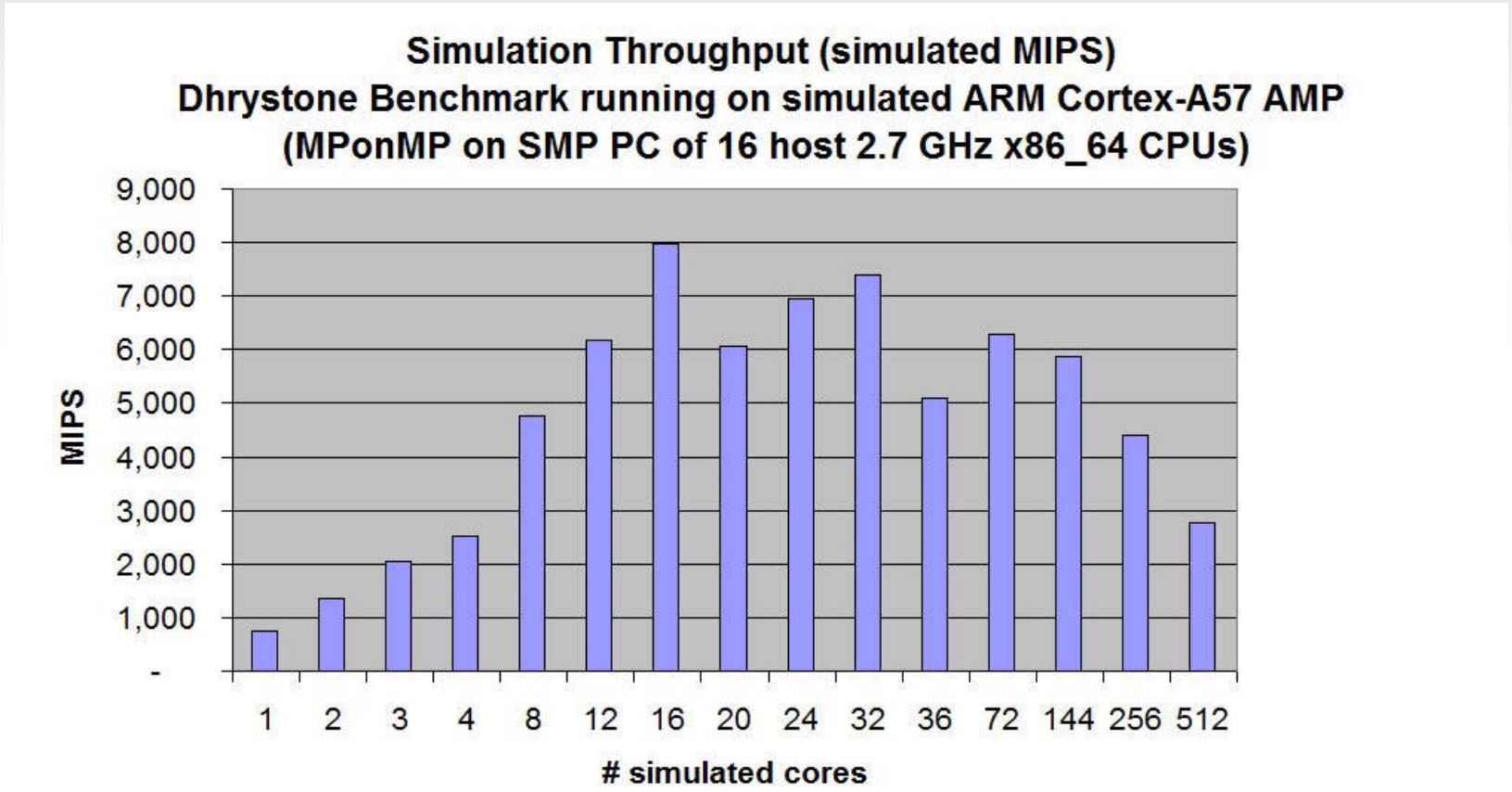


Significant gains if host cores available

- 1-4 cores in platform, no communication, running same application
- QuantumLeap speeds up AMP simulation performance significantly
- 3.4GHz quad core i7-3770 host machine, Linux OS

QuantumLeap Results on 16-Core Host

MULTICORE DESIGN SIMPLIFIED
imperas



- More work is needed, however, initial experiments show that QuantumLeap scales to work on larger numbers of target processors, and larger numbers of host processors

Summary

MULTICORE DESIGN SIMPLIFIED
imperas

- Single threaded virtual platform simulation has performance problems as the number of processors in the target virtual platform increases
- A new parallel simulation algorithm shows excellent performance improvements while maintaining simulation determinism
  - Tightly coupled bare metal application speedup of ~2x for 4 processor target on 4 core host
  - Linux application speedup of >2.5x for 4 processor target on 4 core host
  - AMP application speedup of ~3.5x for 4 processor target on 4 core host
- The new algorithm appears to scale well with both increasing target processors and increasing host processors