Measuring inter-task interferences in the NGMP

Francisco J. Cazorla[1]

Mikel Fernandez[1]

Roberto Gioiosa[1]

Eduardo Quiñones[1]

Marco Zulianello

Luca Fossati

[1] Computer Architecture/Operating System Interface Group (www.bsc.es/caos)

Francisco.cazorla@bsc,es roberto.gioiosa@bsc.es

ESA Workshop on Avionics Data, Control and Software Systems (ADCSS)
25-27 October 2011, ESTEC
Multi-Core Processors for Space Applications

BSC

- Spanish national research center (www.bsc.es)
  - +300 people (>80% are researchers)
- Areas of research:
  - Life Sciences
  - Earth Sciences
  - Computer Sciences
    - Research on HPC systems, desktop-like systems, …
    - Research on real-time systems → CAOS group

The CAOS group

- Computer Architecture/Operating System (CAOS) group
  - www.bsc.es/caos
  - 17 people
  - Projects
    - Industrial: ESA, Sun Microsystems, IBM
    - Public Funded
      - FP6
      - FP7: MERASA, PROARTIS, parMERASA
  - Group leader: Francisco J. Cazorla

Agenda

- **Introduction**
  - **Motivation**
  - **Objective of the activity**
- Infrastructure
- Experiments and results
- Conclusions and Future work

Introduction

- Critical Real-time Embedded Systems (CRTESs) or hard real-time systems are in everyday life



- Some of the main requirements of hard real-time systems
  - Functional correctness (like any other computing system)
  - **Timing correctness**

Requirements

- Increasingly higher functional value to keep competitive edge
- CRTEs require **more computational power**
  - More and more functions required
  - Functions are becoming more complex
  - Examples:
    - Automotive: (5x-10x) driver assistance in steer-by-wire, brake-by-wire, etc
    - Aerospace: (>4x) Unmanned Aerial Vehicles
    - Space: computational-intensive value-added on-board functions
- Within bounded development and production costs

Achieving High Performance

- Such required performance could be achieved by designing complex single-core processors
  - Longer pipelines
  - Out of order execution

Multi-core processors are considered the solution for some of these problems!

  - Too complex due to their non deterministic run-time behaviors
  - Timing anomalies
  - High energy requirements of such complex processors don't satisfy CRTE low-power constraints

Multi-cores for Hard Real-Time Systems

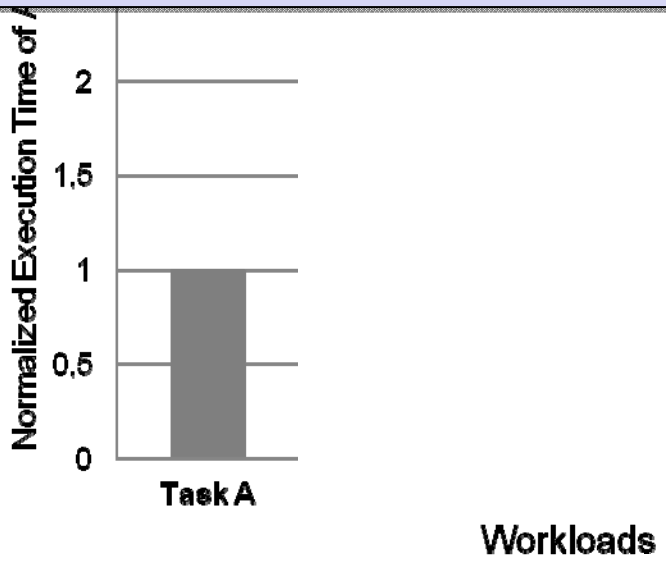
- Pros:
  - Better performance per watt than single-core processors
  - Maintain simple core design
  - Enable co-hosting mixed-criticality applications
    - Hardware utilization is maximized, while cost, size, weight and power requirements are reduced.
- Cons:
  - Harder to time analyze w.r.t. single-core chips
    - It is hard to provide a safe and tight WCET estimation in multi-cores
    - Because of inter-task interferences!

Inter-task interferences

- Appear when several tasks that share a hardware resource want to access to it at the same time, so an arbitration stage is required

- The Execution time, and hence the WCET, of a task in a

WCET depends on the workload!!!



State of the art (hardware proposals)

- Several proposals developed to ease the computation of WCET estimates of tasks running on multicore architectures (MERASA, ACROSS, GENESYS, PRET, TTA, PREDATOR ...)
  - Either by isolating interactions between tasks or
  - Or by upper-bounding the maximum interaction between tasks (MERASA)
  - NPI activity between BSC and ESA.
    - Title: Architectural solutions for the timing predictability of next-generation multi-core processors
    - Objective: Creating hardware support for taking inter-task interferences into account when computing WCET estimations for the NGMP (simulator)
    - People: Vivek Sabinneni, Francisco J. Cazorla, Eduardo Quiñones, Luca Fossati, Marco Zulianello

State of the art (hardware proposals)

- Current MT architectures do not implement those hw features
  - It will take several years to be implemented
  - Industry cannot benefit nowadays from that proposals
  - **COTS multicore processors have to be used instead**

Using COTS multicore processors

- Static analysis have several problems when used in industrial-size applications [1]
  - Hardware analysability, Computational tractability , Information gathering
- Measurement-based Timing Analysis (MBTA) approaches, or hybrid approaches, have emerged
  - MBTA for single-threaded architectures
    - WCET estimation = longest observed execution time (LOET) x safety margin

[1] *"On the Industrial Fitness of WCET Analysis". Mezzeti, Vardanega. WCET Worskshop 2011.*

Using COTS multicore processors

- In multicores the effect of inter-task interferences affect the computation of the safety margin
  - WCET estimation = longest observed execution time (LOET) x safety margin x **margin for inter-task interferences**

Objective of this project

- Define and develop a *benchmark suite*
  - Representative of reference ESA applications,
  - Suitable to exercise the new NGMP multicore processor
  - Capable to generate different inter-task interference scenarios that may arise in the NGMP processor, by stressing different hw shared resources.
- The ultimate goal of the benchmark suite is to provide a methodology to measure the real-time capabilities of the NGMP

Agenda

- Introduction
- **Infrastructure**
  - **Target environment and metrics**
  - **Reference benchmarks**
- Experiments and results
- Conclusions and Future work

Target applications and metrics

- The target workloads comprise both
  - Hard real-time applications or *control applications*
  - non-hard real-time applications or *payload applications*
- Metrics
  - Hard Real-Time Applications:
    - The sensitivity (jitter) of the HRT applications to the execution environment which include the other HRT and NRT applications.
    - Understanding and quantifying the impact of shared resources
  - Non-Hard Real-Time Applications:
    - The performance of the NRT applications
    - How much performance can be obtained by NHRT tasks without affecting (much) HRT apps?

Developing representative benchmarks

- Hard-real time applications
  - Micro benchmarks
    - Put high load on a single resource (L1, L2, cpu)
    - Used to measure the highest interference an application can suffer
  - Standard benchmarks: EEMBC, CoreMark
  - Mimicking applications
    - Applications that mimic main characteristic of some selected reference apps.
      - Instruction mix, memory access frequency, ..
- Non-hard real-time applications
  - Standard benchmarks: ParSec

The board: ML510

- NGMP with 4 cores
- Private per-core resources
  - Core, Data and instruction caches
- Shared resources
  - The bus to the L2, L2, and the memory bandwidth (memory controller)
  - I/O resources are also shared but are not considered in this project
- DDR2 interface runs at 140 MHz
- NGMP frequency: 70 MHz

The NGMP

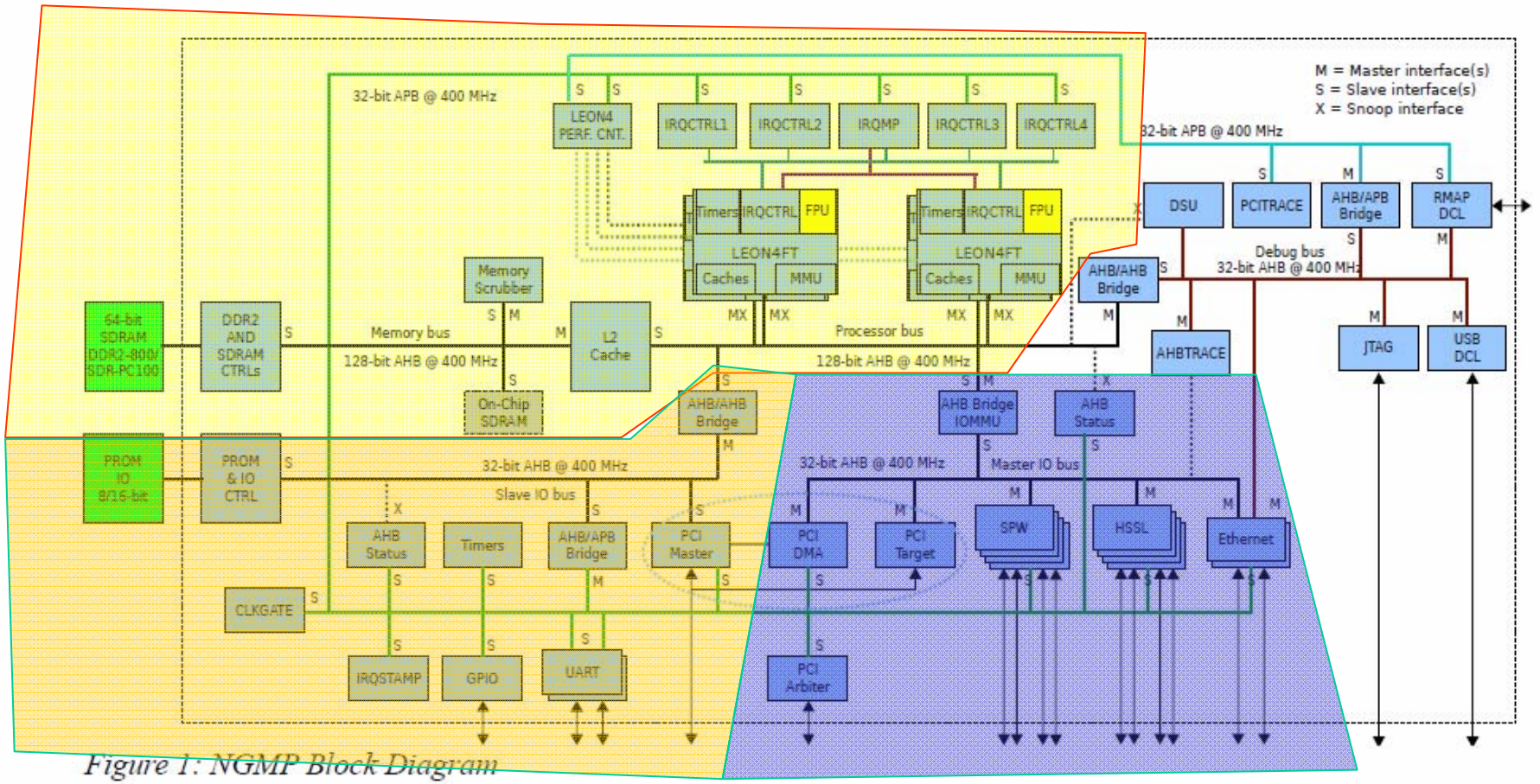


Figure 1: NGMP Block Diagram

http://microelectronics.esa.int/ngmp/LEON4-NGMP-DRAFT-1-6-changebars.pdf

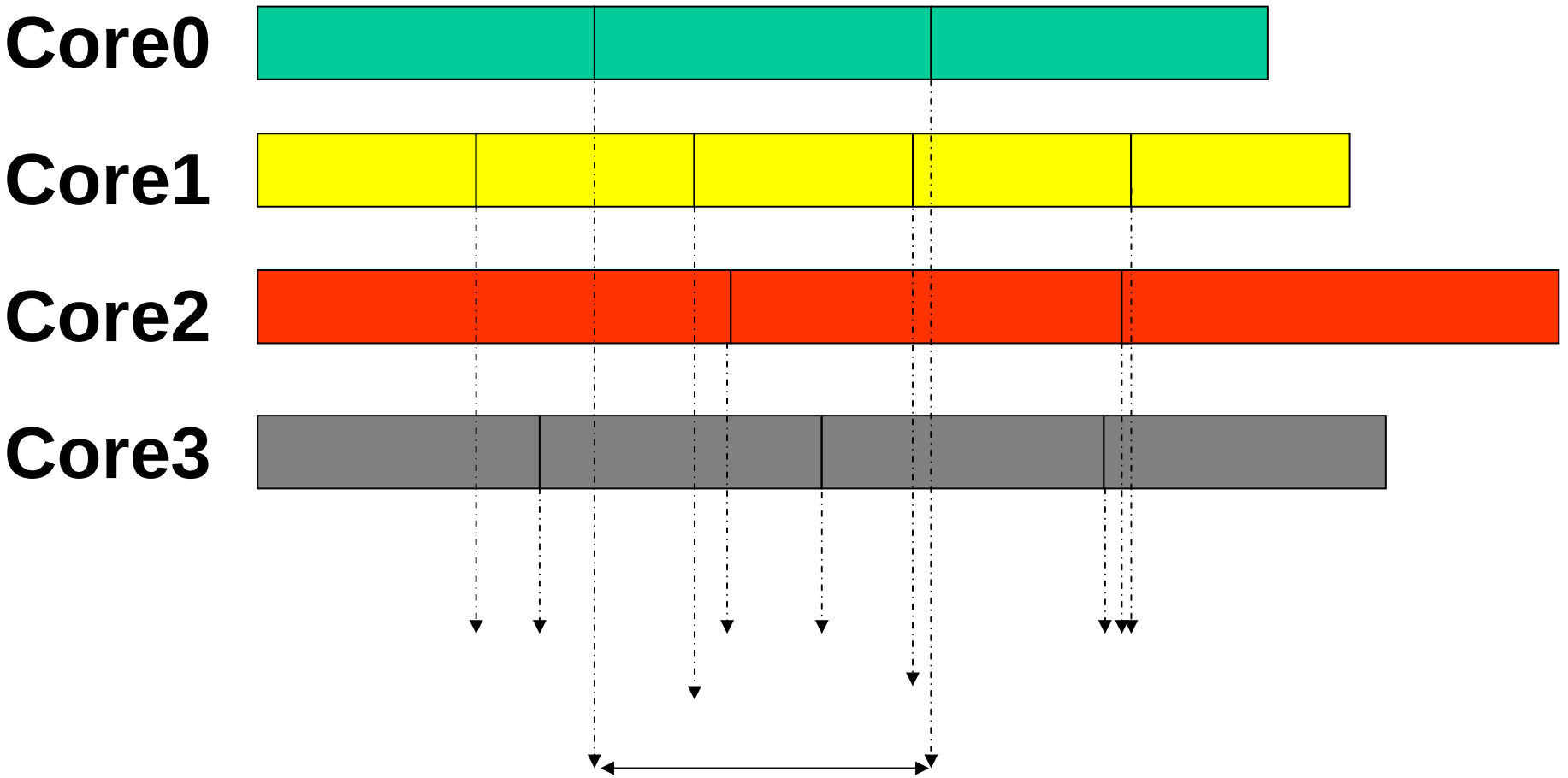
Agenda

- Introduction
- Infrastructure
- **Experiments and results**
  - **Microbenchmark-only experiments**
  - **Results**
- Conclusions and Future work

Experiments

- We will run a different ‘load’ in the desired cores
  - A load is any program that can be encapsulated into an executable with a calling script
- Performance counters are polled
  - This enables measuring inter-task interferences

Experiments (conceptually)



Execution time of Task 0 when running with a constant load on C1, C2 and C3

By comparing T0 Exec. Time w.r.t its run in isolation → inter-task interferences

Microbenchmark only experiments

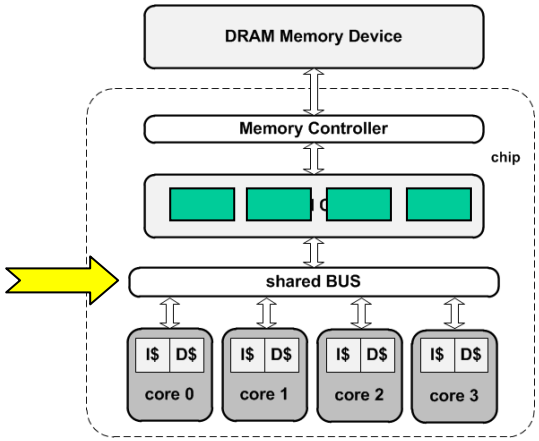
- We will focus on microbenchmarks
  - Will bound the maximum variation tasks may suffer due to inter task interferences.
- In all cases, as reference execution time we have the execution time of each benchmark when it runs in isolation
- Run different sets of microbenchmarks and compute the execution time variation of each of them
  - quadruples: (L2 L1 ADD MULT), …
  - Pairs: (L1, L1), …

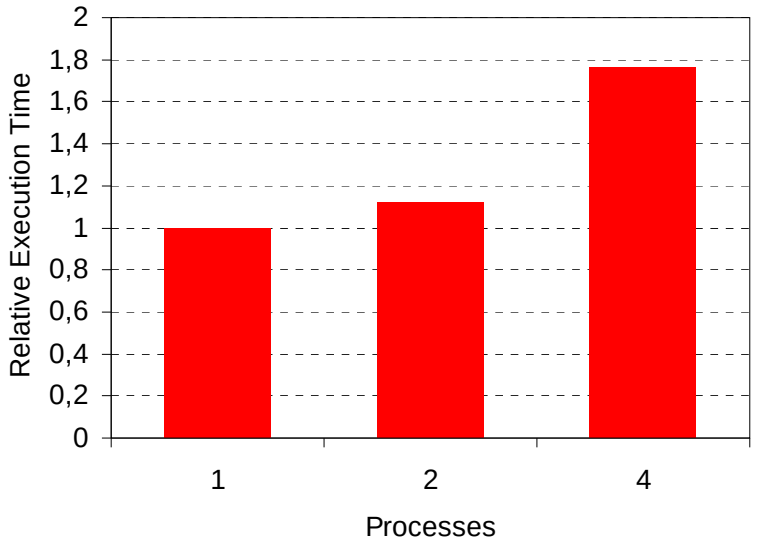
Measuring the inter-task interferences

- We designed several experiments showing the effect of inter-task interferences on:
  - AMBA Bus that connects core to L2
  - Memory bandwidth
  - Memory bandwidth + L2 cache
- Importance of task scheduling

Results: Amba bus

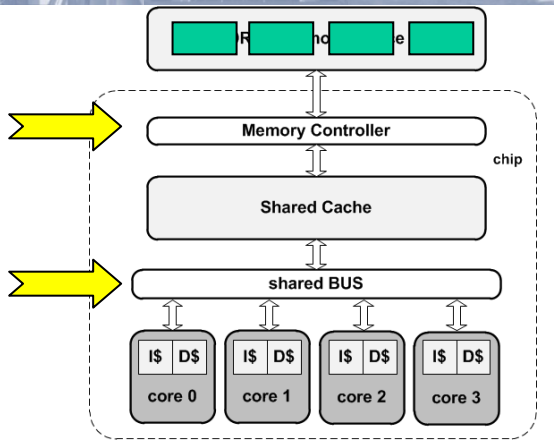
- AMBA Bus that connects core to L2
  - 4 copies of $L2_{40KB}$ (less than 1/ 4)
  - Each copy always misses in DL1 and hits in L2
  - N copies → interaction in Amba bus
- Conclusions.
  - Reference case: Single-Threaded execution time
  - The worst delay due to sharing the AMBA Bus
    - 10% for 2 tasks
    - 75% for 4 tasks





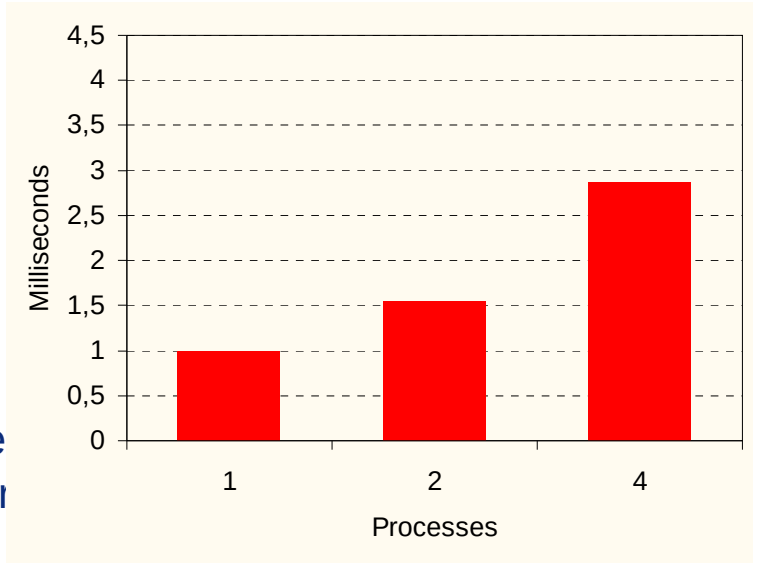
Results: memory bandwidth

- Memory bandwidth
  - 4 copies of $L2_{miss}$ (memory)
  - All accesses in each copy always miss in L2
  - N copies → interaction in the memory controller & the memory BW **(and also in the AMBA bus)**



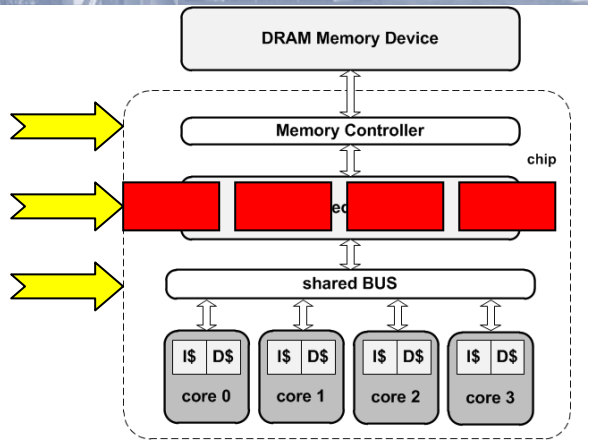
- Conclusions*.
  - Reference case (Single-Threaded)
  - The worst delay due to sharing memory bandwidth
    - 50% for 2 tasks
    - 2.9x for 4 tasks

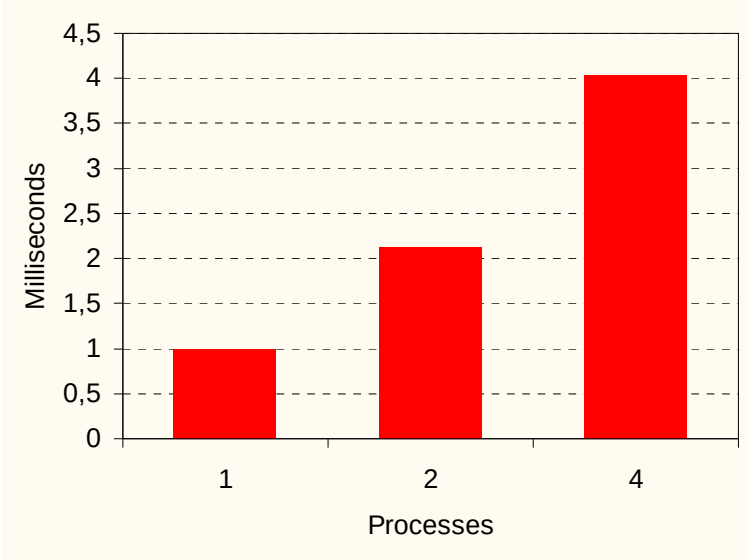
*(In the FPGA implementation of the NGMP the ratio core_frequency/memory frequency is lower than in reality)



Results: L2 + memory

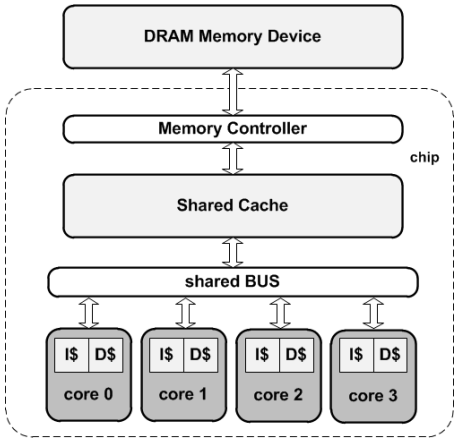
- Memory bandwidth + L2 cache
  - 4 copies of $L2_{FULL}$
  - Each copy will hit in L2 many times
  - N copies → interaction in L2 and memory bandwidth and memory controller (also in the AMBA bus)
  - Reference case (Single-Threaded)
- Conclusions
  - The worst delay due to sharing memory bandwidth and L2
    - 2x for 2 tasks and
    - 4x for 4 tasks

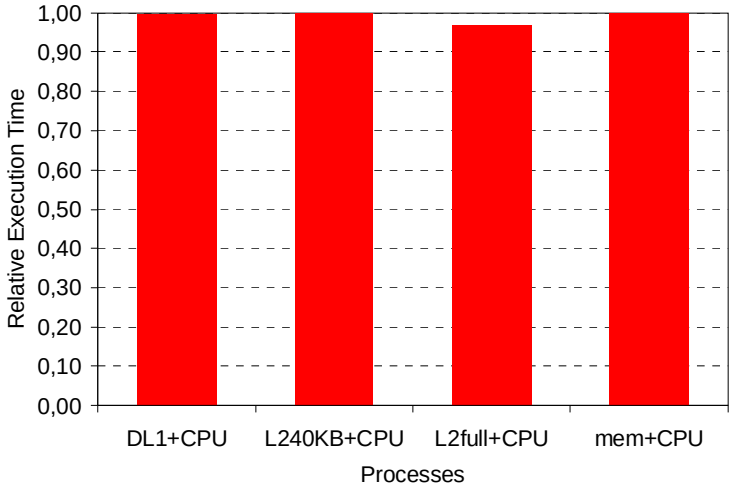




Importance of task scheduling

- We run tasks (micro benchmarks) with ‘complementary’ resource requirements
  - Data cache and CPU
  - L2 and CPU
  - Memory and CPU
- Conclusions:
  - Reference case: Single-Threaded execution time of the first thread
  - No slowdown when the first thread runs with CPU as second thread
  - CPU is not affected either





Agenda

- Introduction
- Infrastructure
- Experiments and results
- **Conclusions and Future work**

Conclusions

- We have presented initial results about the effect of inter-task interferences on time predictability for the NGMP.
- Worst observed behaviors are the following
  - AMBA bus effect. Up to 75% for 4 cores
  - Memory bandwidth effect*. Up to 2.9x for 4 cores
  - L2 cache + memory BW effect*. Up to 4x for 4 cores
- Task scheduling will play a key role reducing actual inter-task interferences-effects from the worst-observed behaviors
  - We can obtain 0% performance degradation if we schedule non-interfering tasks.

Future Work

- We have focused on microbenchmarks running on Linux
- Microbenchmarks
  - Bound the maximum variation tasks may suffer due to inter task interferences.
- Next steps
  - Applications to obtain more realistic results
    - EEMBC, CoreMark and mimicking benchmarks as control apps.
    - ParSec as background applications.
  - RTEMs

Measuring inter-task interferences in the NGMP

Francisco J. Cazorla[1]

Mikel Fernandez[1]

Roberto Gioiosa[1]

Eduardo Quiñones[1]

Marco Zulianello

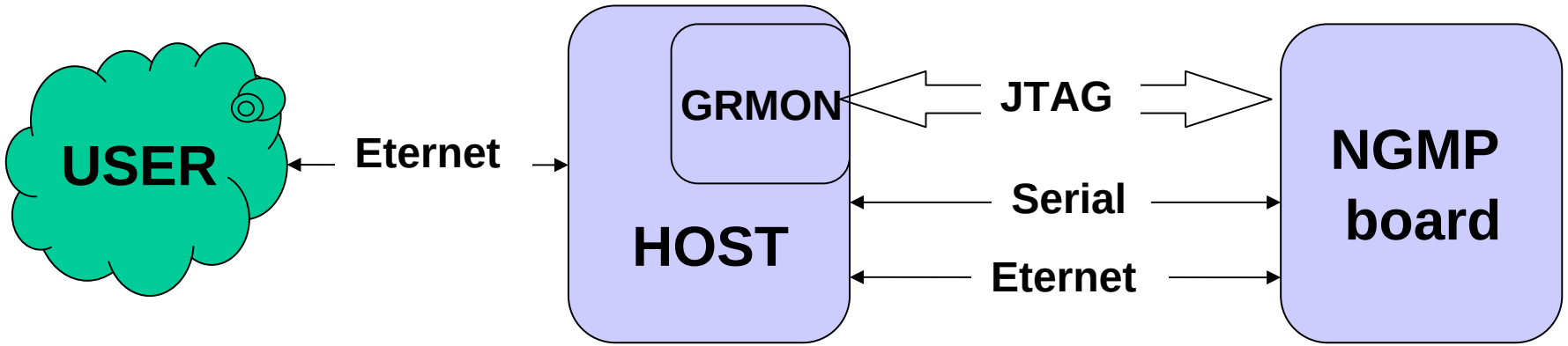
Luca Fossati

Acknowledgements

Jiri Gailser, Per Danielsson, and Jan Andersson for their technical support with ML510

The execution infrastructure

- We want to develop a set of scripts that allows anyone to
  - Connect to the host machine
  - Run experiments on NGMP
  - Collect results



System setup

- Host machine
  - Linux desktop
    - Compiling, linking toolchains
    - GRMon
  - Connected to NGMP board
    - JTAG (debug), preferred
    - Serial (standard)
    - Ethernet (standard, debug)
- NGMP board
  - Software
    - Linux 2.6
    - RTEMS
    - Linux 3.0
  - Connected to Host
    - debug to GRMon
    - Standard interface (serial, Ethernet)