Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Executing Parallel Real-time Software on Multi- and Manycores in a Timely Manner

Presenters:

Francisco J. Cazorla, Head of the Computer Architecture/ Operating System group at BSC
Leonidas Kosmidis, Senior Researcher at BSC

Contributors:

Eduardo Quinones, Jaume Abella,
Carles Hernandez, Enrico Mezzetti,
Senior Researchers at BSC

11th ESA ADCSS2017

ESTEC– October 19th, 2017

Motivation in a Nutshell: More complex HW and SW

- Facts
  - End of federated architectures and arise of Integrated architectures
    - E.g., IMA (avionics), AUTOSAR (automotive), IMA-SP (space)
    - Increased reliability, reduced SWaP costs
  - Increased overall system's value provided by software (electronics)
    - More and more critical functionalities provided by software

Motivation in a Nutshell: More complex HW and SW

Facts

- End of federated architectures and arise of Integrated architectures
  - E.g., IMA (avionics), AUTOSAR (automotive), IMA-SP (space)
  - Increased reliability, reduced SWaP costs
- Increased overall system's value provided by software (electronics)
  - More and more critical functionalities provided by software

Consequences

- Software's increasing performance needs
  - E.g. 100x according to arm in automotive (from 2016 to 2024)
- Use of more aggressive HW designs: multi- and many-cores (MMCs)

Motivation in a Nutshell: More complex HW and SW

- Facts
  - End of federated architectures and arise of Integrated architectures
    - E.g., IMA (avionics), AUTOSAR (automotive), IMA-SP (space)
    - Increased reliability, reduced SWaP costs
  - Increased overall system's value provided by software (electronics)
    - More and more critical functionalities provided by software

- Consequences
  - Software's increasing performance needs
    - E.g. 100x according to arm in automotive (from 2016 to 2024)
  - Use of more aggressive HW designs: multi- and many-cores (MMCs)

- Needs
  - Improved WCET analysis and timing V&V in general
  - Exploiting parallel hardware: programming models and accelerators

Outline

1. Multi-core and Many-core (MMC) contention
    - Concept and proposed solutions

2. A probabilistic angle to WCET estimation
    - Concept and maturity of existing tools

3. The way ahead: accelerators and parallel programming models in critical systems
    - Challenges and our work in this domain

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MULTICORE AND MANYCORE (MMC) CONTENTION

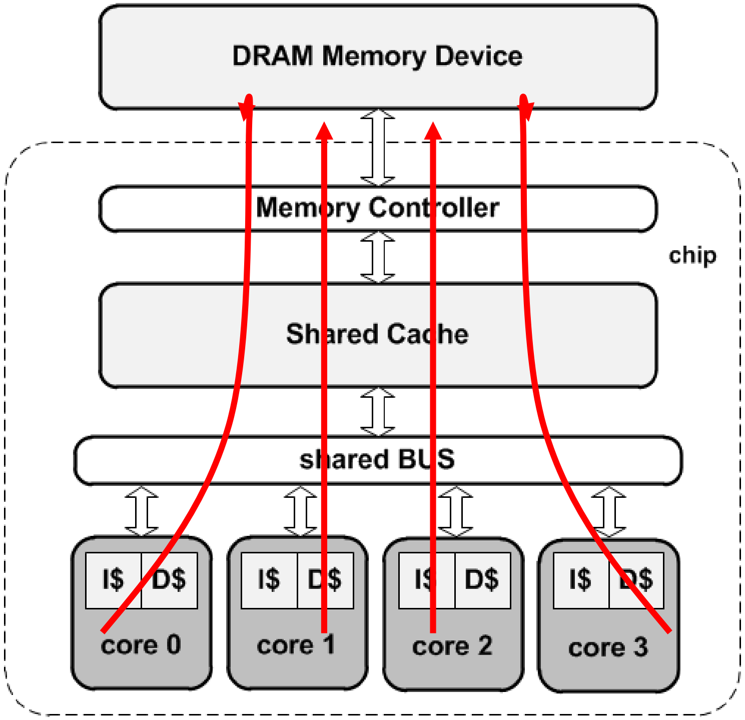
Unió Europea
Fons europeu
de desenvolupament regional

Generalitat de Catalunya
Departament d'Empresa
i Coneixement

AGAUR
Agència
de Gestió
d'Ajuts
Universitaris
i de Recerca

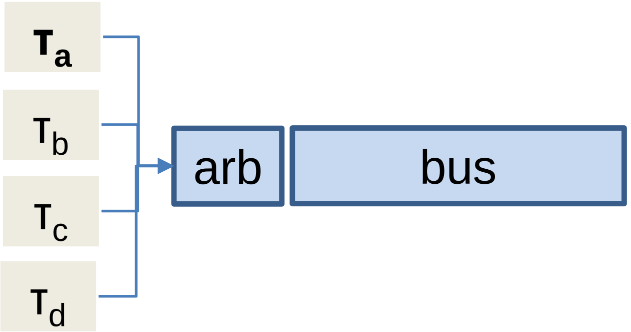
MMCs

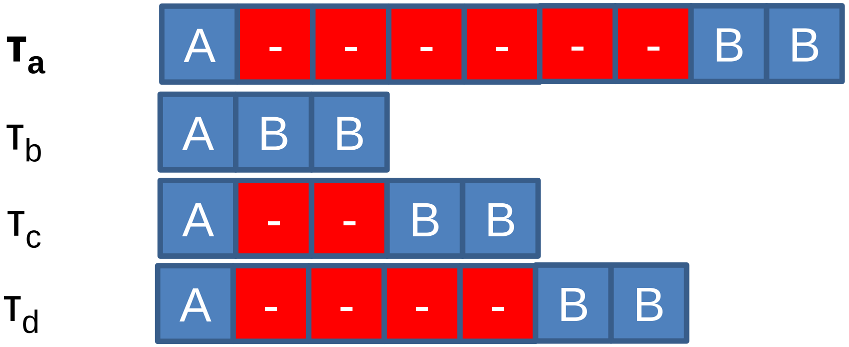
- I love them (high resource usage)
- I hate them (contention → low predictability)
  - The sole presence of a task in a core affects the execution time of other tasks in other cores



Measurement-Based Timing Analysis

- Conservative approach for contention analysis
  - Every access of the task under analysis suffers the worst contention
    - Each requests waits for all other N-1 requests



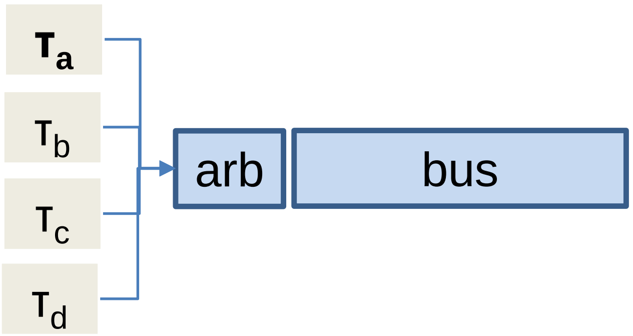


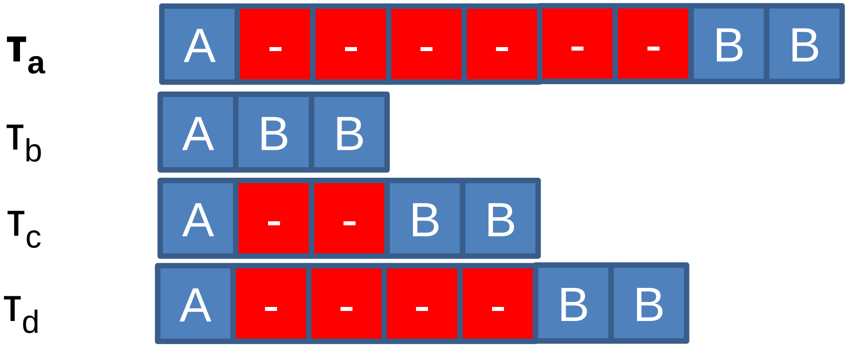
τ_a	A	-	-	-	-	-	-	B	B
τ_b	A	B	B						
τ_c	A	-	-	B	B				
τ_d	A	-	-	-	-	B	B		

Measurement-Based Timing Analysis

Conservative approach for contention analysis

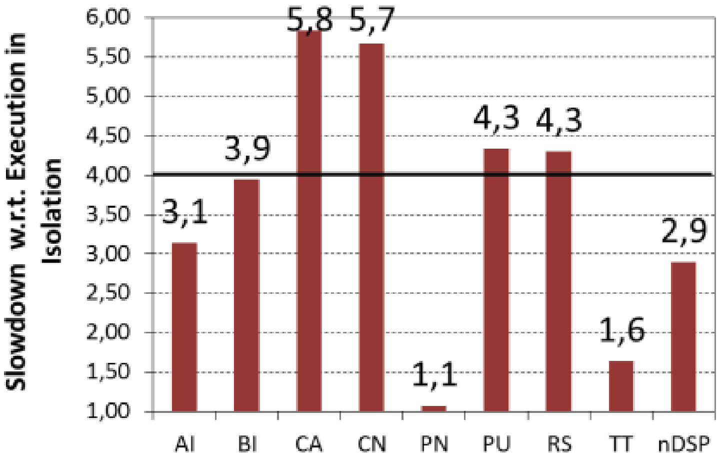
- Every access of the task under analysis suffers the worst contention
  - Each requests waits for all other N-1 requests

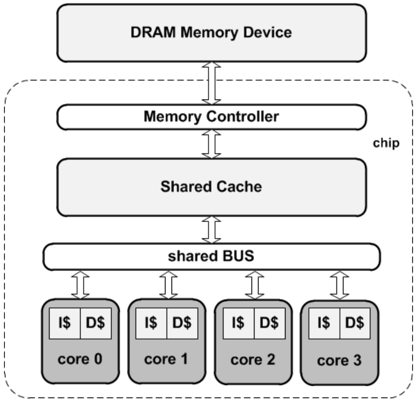




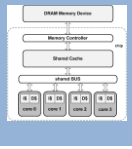
T_a	A	-	-	-	-	-	-	B	B
T_b	A	B	B						
T_c	A	-	-	B	B				
T_d	A	-	-	-	-	B	B		

Pessimistic Results

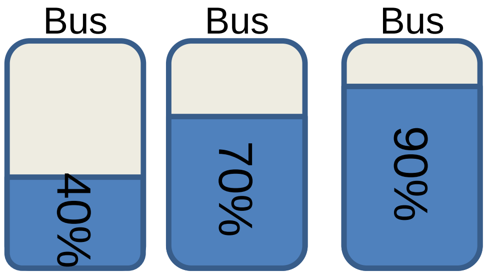




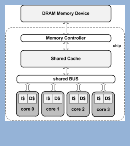
Contention modelling: approach 1



- Micro-benchmark (µb)
  - Simple benchmarks putting desired load on shared resources
  - Bus, memory bandwidth, cache

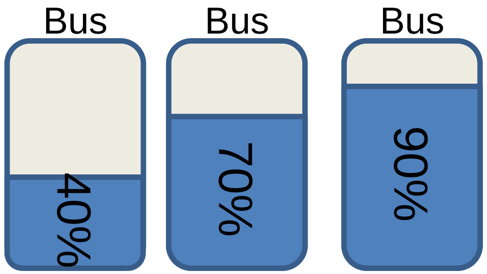


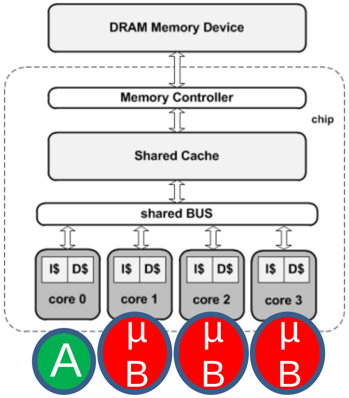
Contention modelling: approach 1



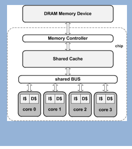
- Micro-benchmark (μb)
  - Simple benchmarks putting desired load on shared resources
  - Bus, memory bandwidth, cache

- Approach
  - Pre-characterize the expected load at operation (load-op)
  - Run each application against $\mu\text{b}_{\text{load-op}}$

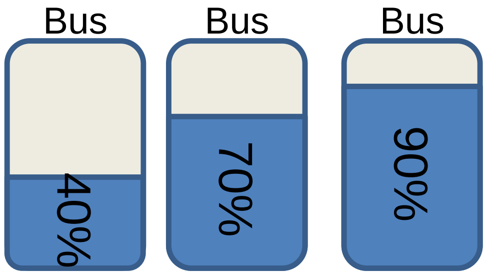




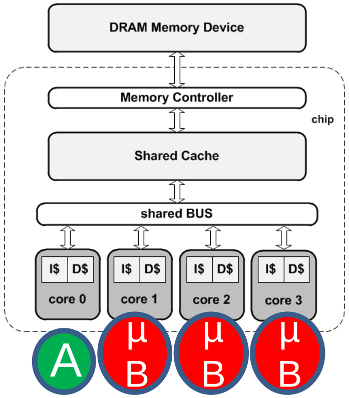
Contention modelling: approach 1



- Micro-benchmark (µb)
  - Simple benchmarks putting desired load on shared resources
  - Bus, memory bandwidth, cache



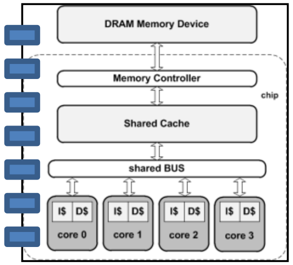
- Approach
  - Pre-characterize the expected load at operation (load-op)
  - Run each application against $\mu b_{\text{load-op}}$



- Benefits
  - Tighter WCET estimates than the conservative approach
  - Still measurement based → no static modelling
  - Characterization in isolation
    - No need to run target applications simultaneously
    - WCET analysis starts independently for each application before integration

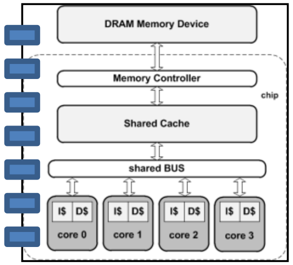
Contention modelling: approach 2

- Hardware monitors (Perf. Monitoring Counters)
  - Provide information about ‘events’ on resource usage



Contention modelling: approach 2

- Hardware monitors (Perf. Monitoring Counters)
  - Provide information about ‘events’ on resource usage
- Approach
  - Derive the access latency to the resources with μb

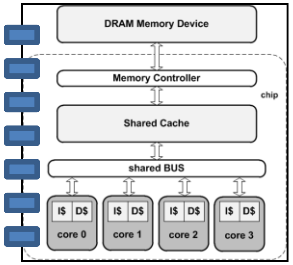


μB μB

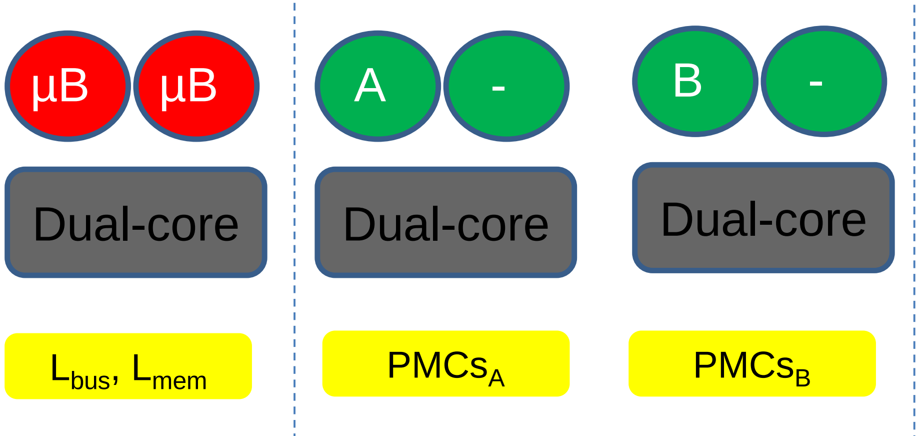
Dual-core

L_{bus}, L_{mem}

Contention modelling: approach 2



- Hardware monitors (Perf. Monitoring Counters)
  - Provide information about ‘events’ on resource usage
- Approach
  - Derive the access latency to the resources with µb
  - Run each application in isolation and read PMCs



µB µB

Dual-core

L_{bus}, L_{mem}

A -

Dual-core

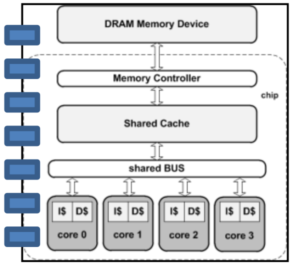
$PMCs_A$

B -

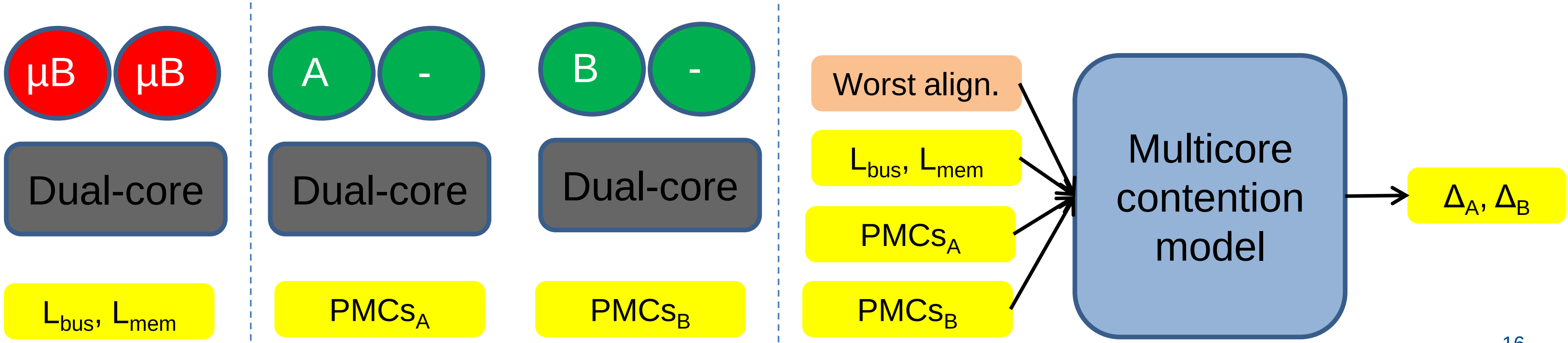
Dual-core

$PMCs_B$

Contention modelling: approach 2



- Hardware monitors (Perf. Monitoring Counters)
  - Provide information about ‘events’ on resource usage
- Approach
  - Derive the access latency to the resources with µb
  - Run each application in isolation and read PMCs
  - A Multicore Contention Model combines PMCs and derives the contention tasks cause each other (Δ_A, Δ_B)



µB µB

Dual-core

L_{bus}, L_{mem}

A -

Dual-core

$PMCs_A$

B -

Dual-core

$PMCs_B$

Worst align.

L_{bus}, L_{mem}

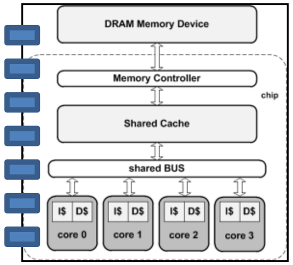
$PMCs_A$

$PMCs_B$

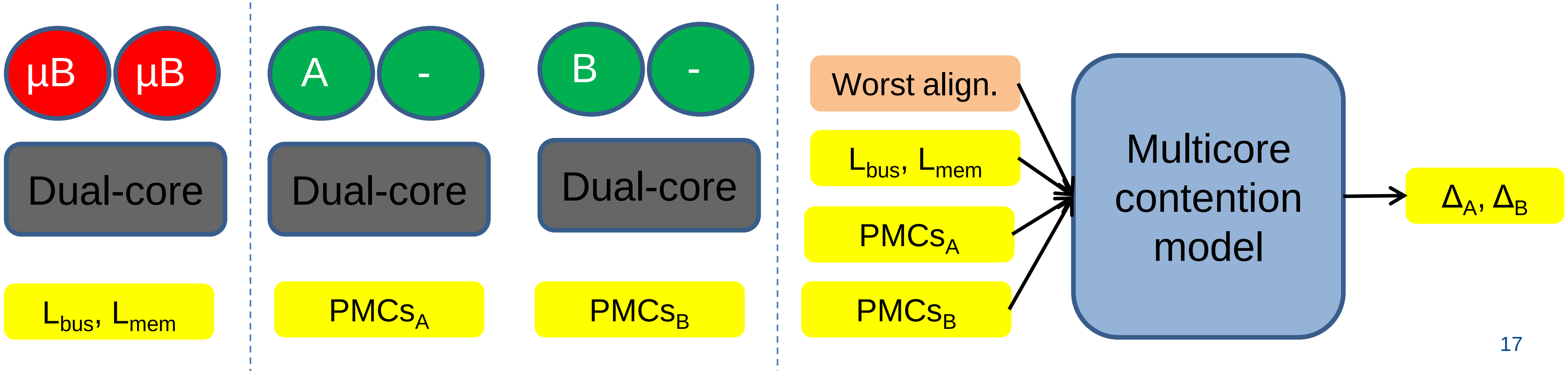
Multicore contention model

Δ_A, Δ_B

Contention modelling: approach 2



- Hardware monitors (Perf. Monitoring Counters)
  - Provide information about ‘events’ on resource usage
- Approach
  - Derive the access latency to the resources with µb
  - Run each application in isolation and read PMCs
  - A Multicore Contention Model combines PMCs and derives the contention tasks cause each other (Δ_A, Δ_B)
- Difference w.r.t. Approach 1
  - Factors in the **worst possible alignment of requests**



µB µB

Dual-core

L_{bus}, L_{mem}

A -

Dual-core

$PMCs_A$

B -

Dual-core

$PMCs_B$

Worst align.

L_{bus}, L_{mem}

$PMCs_A$

$PMCs_B$

Multicore contention model

Δ_A, Δ_B

What mµbt can offer you?

"If you have a problem, if no one else can help, and if you can find them, maybe you can hire: The A-Team."

What mµbt can offer you?

"If you have a problem, if no one else can help, and if you can find them, maybe you can hire: The A-Team."

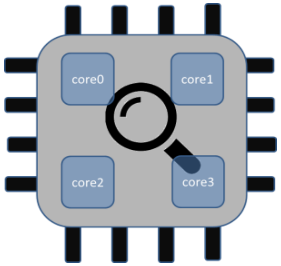
"If you have a problem with multicore timing analysis, you can find us: MµBT team"

BSC's multi-core microbenchmark technology (**mµBT**) and performance analysis experience

francisco.cazorla@bsc.es

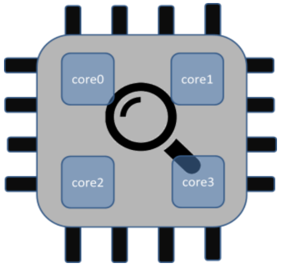
What mµbt can offer you?

- Analysis of new multicore platforms
  - Determining the time predictability characteristics of different processors (Worst-case perf. analysis)

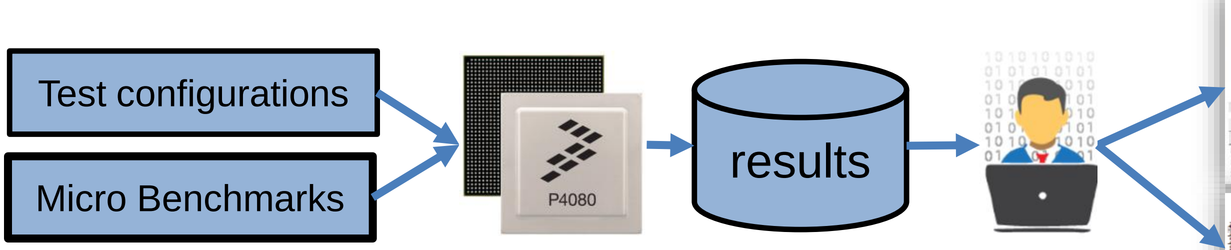


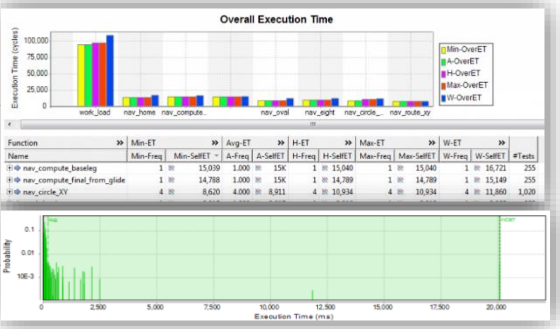
What mµbt can offer you?

- Analysis of new multicore platforms
  - Determining the time predictability characteristics of different processors (Worst-case perf. analysis)



- Evidence gathering for timing V&V on multicore



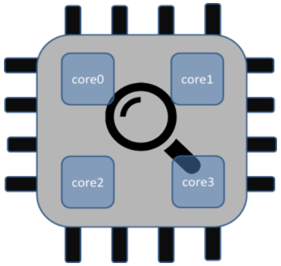


Function	Min-ET		Avg-ET		H-ET		Max-ET		W-ET		
Name	Min-Freq	Min-SelfET	A-Freq	A-SelfET	H-Freq	H-SelfET	Max-Freq	Max-SelfET	W-Freq	W-SelfET	#Tests
nav_compute_baseleg	1	15,039	1.000	15K	1	15,040	1	15,040	1	16,721	255
nav_compute_final_from_glide	1	14,788	1.000	15K	1	14,789	1	14,789	1	15,149	255
nav_circle_XY	4	8,620	4.000	8,911	4	10,934	4	10,934	4	11,860	1,020

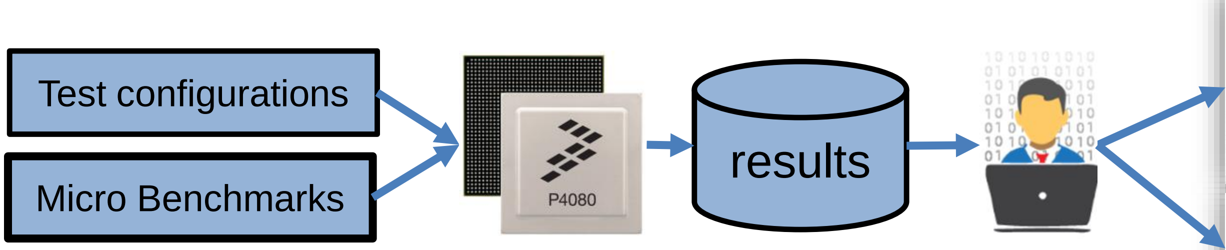
RAPITA Systems Ltd
A DANLAW Company

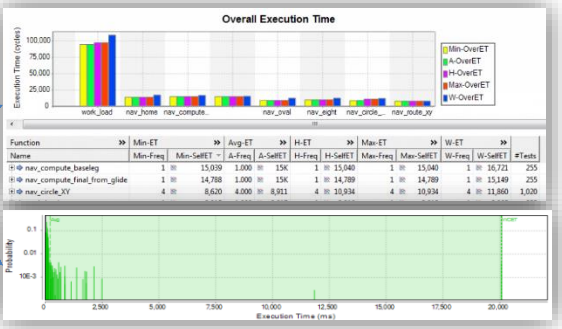
What mµbt can offer you?

- Analysis of new multicore platforms
  - Determining the time predictability characteristics of different processors (Worst-case perf. analysis)



- Evidence gathering for timing V&V on multicore





Function	Min-ET		Avg-ET		H-ET		Max-ET		W-ET		
Name	Min-Freq	Min-SelfET	A-Freq	A-SelfET	H-Freq	H-SelfET	Max-Freq	Max-SelfET	W-Freq	W-SelfET	#Tests
nav_compute_baseleg	1	15,039	1.000	15K	1	15,040	1	15,040	1	16,721	255
nav_compute_final_from_glide	1	14,788	1.000	15K	1	14,789	1	14,789	1	15,149	255
nav_circle_XY	4	8,620	4.000	8,911	4	10,934	4	10,934	4	11,860	1,020

RAPITA Systems Ltd
A DANLAW Company

- Task scheduling for multicores
  - How task can be scheduled factoring in contention?
- Target platforms

Freescale
- T2010, T2040, T2080
- P4020, P4080, …

ARM boards
- Zynq7000
- UltraScale+

LEON
- LEON3
- LEON4

AURIX
- TC277 family

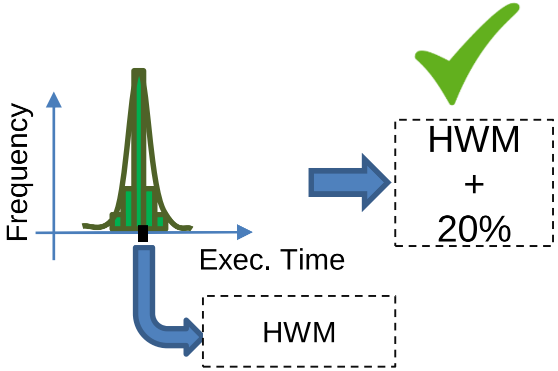
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

A PROBABILISTIC ANGLE TO WCET ESTIMATION

PROXIMA

From deterministic to probabilistic view of WCET

- Execution time of a program

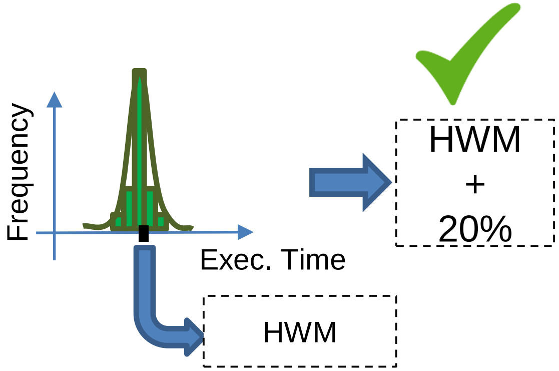


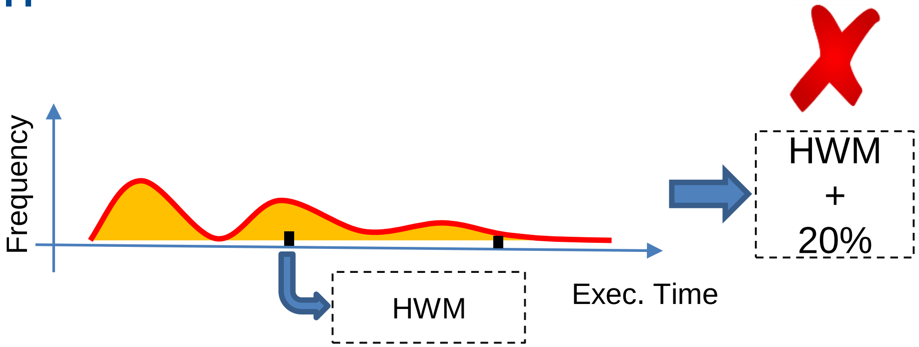
HWM
+
20%

HWM

From deterministic to probabilistic view of WCET

- Execution time of a program

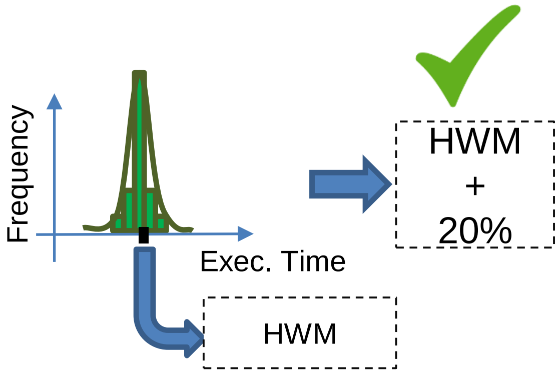


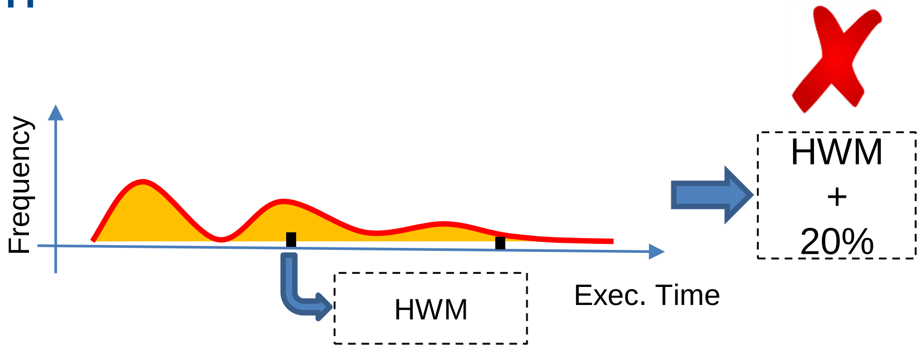


  – Statistical Analysis fits the nature of observed ET distributions

From deterministic to probabilistic view of WCET

- Execution time of a program





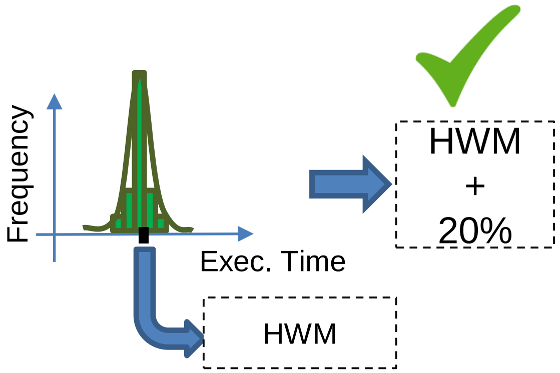
  – Statistical Analysis fits the nature of observed ET distributions

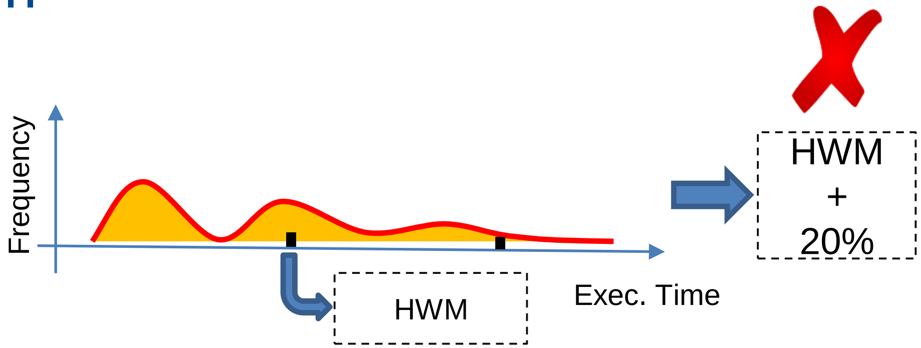
- Renounce the absolute worst case

  – Control resource-interference (mμbt-based approach)

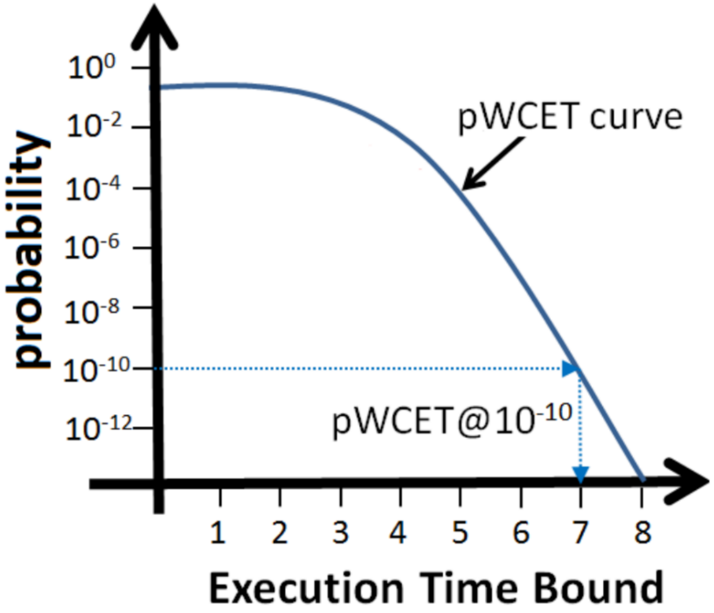
From deterministic to probabilistic view of WCET

- Execution time of a program





  - Statistical Analysis fits the nature of observed ET distributions
- Renounce the absolute worst case
  - Control resource-interference (mµbt-based approach)
- Attach probability to events
  - Those below 10^{-x} per hour → irrelevant
  - instead of… if something can happen then assume it happens

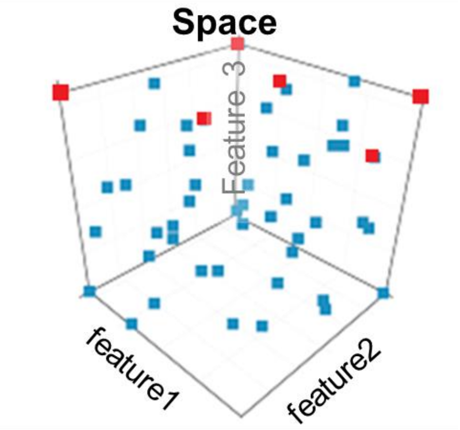


Role of randomization

- Phases:
  - Analysis phase: carry out test campaigns & derive WCET estitmates
  - Operation phase: Actual use of the system
- Confidence on testing:
  - Ensure that the worst-case conditions exercised or approximated
  - **The user**
    - **Can only follow what happens at a high level**
    - **Can’t follow Low-level events (cache placement, bus occupancy, floating-point operation duration)**

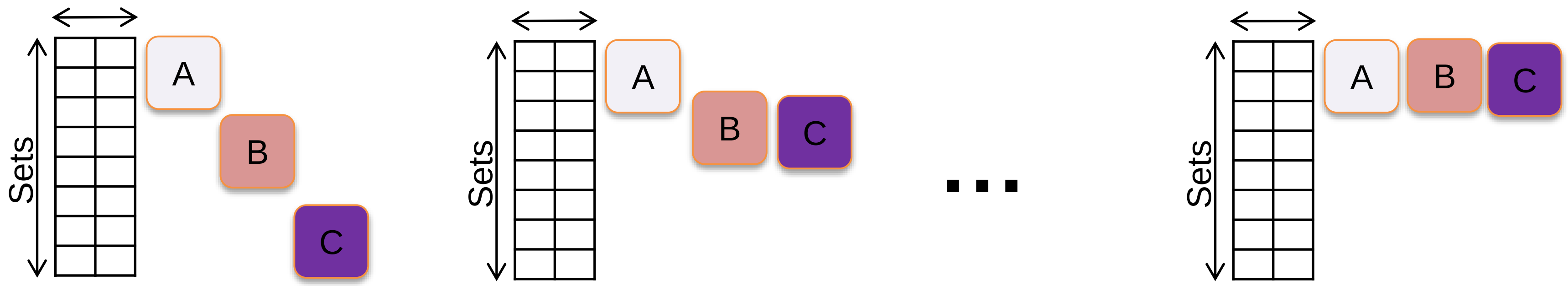
Role of randomization

- Phases:
  - Analysis phase: carry out test campaigns & derive WCET estitmates
  - Operation phase: Actual use of the system
- Confidence on testing:
  - Ensure that the worst-case conditions exercised or approximated
  - **The user**
    - **Can only follow what happens at a high level**
    - **Can't follow Low-level events (cache placement, bus occupancy, floating-point operation duration)**
- Idea:
  - Randomize low-level events
  - Their worst-behaviour (or set of behaviours) will have a probability of appearance
    - Just carry out enough tests!!!
  - Randomization implemented at HW and SW



Cache randomization

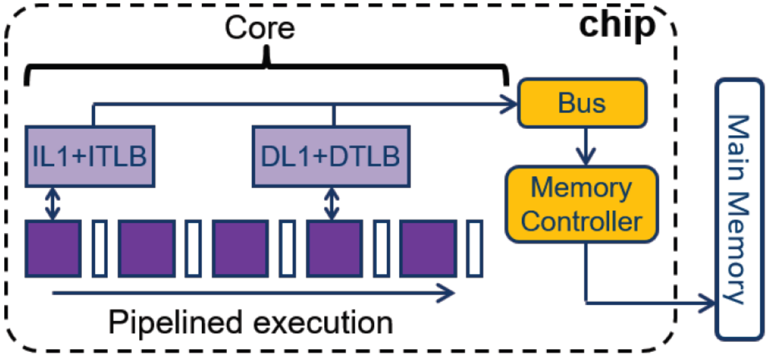
- Example cache placement:

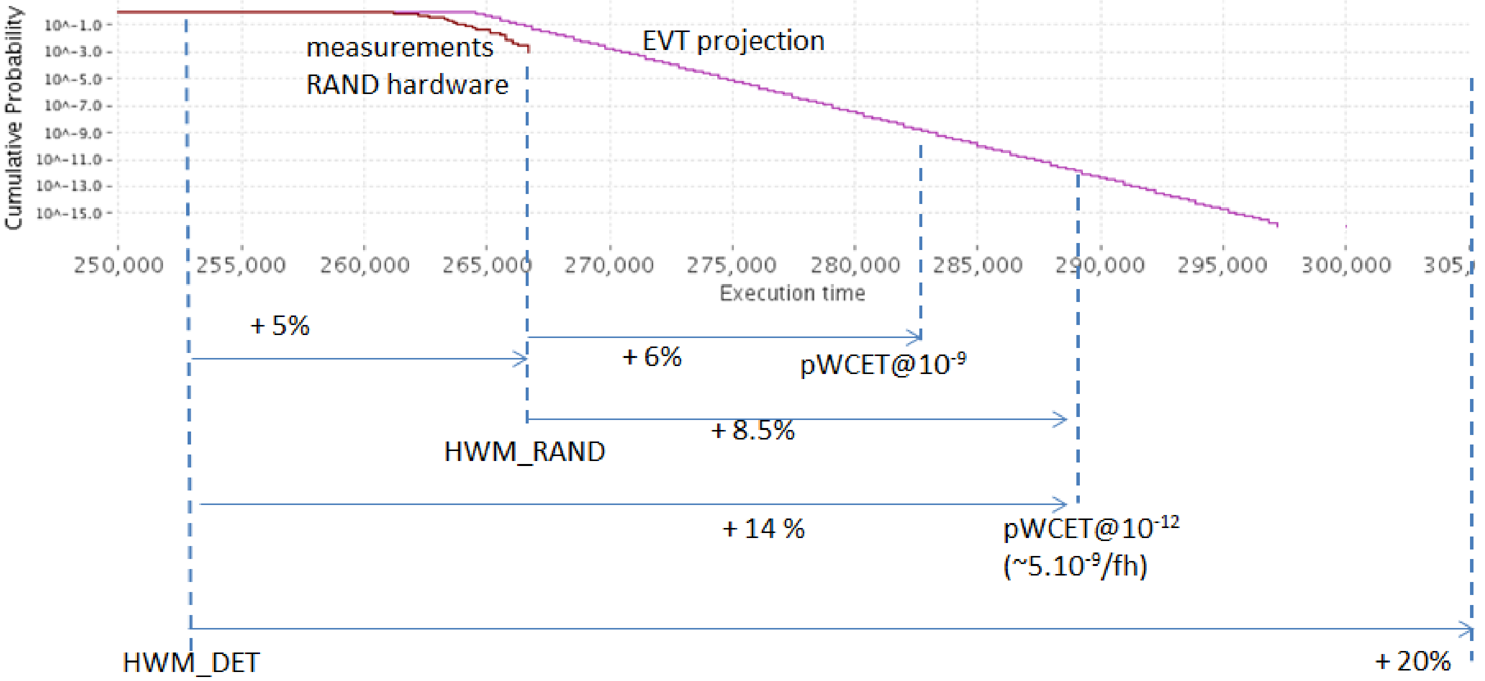


- Deterministic system
  - How does the user get confident that experiments capture bad (worst) mappings?
  - Memory mapping varies across runs, but not in a random manner
- Randomized systems
  - Make N runs
  - We can derive
    - the probability of the observed mappings @ operation
    - the probability of unobserved mappings

Airbus IMA application. Hardware Randomization

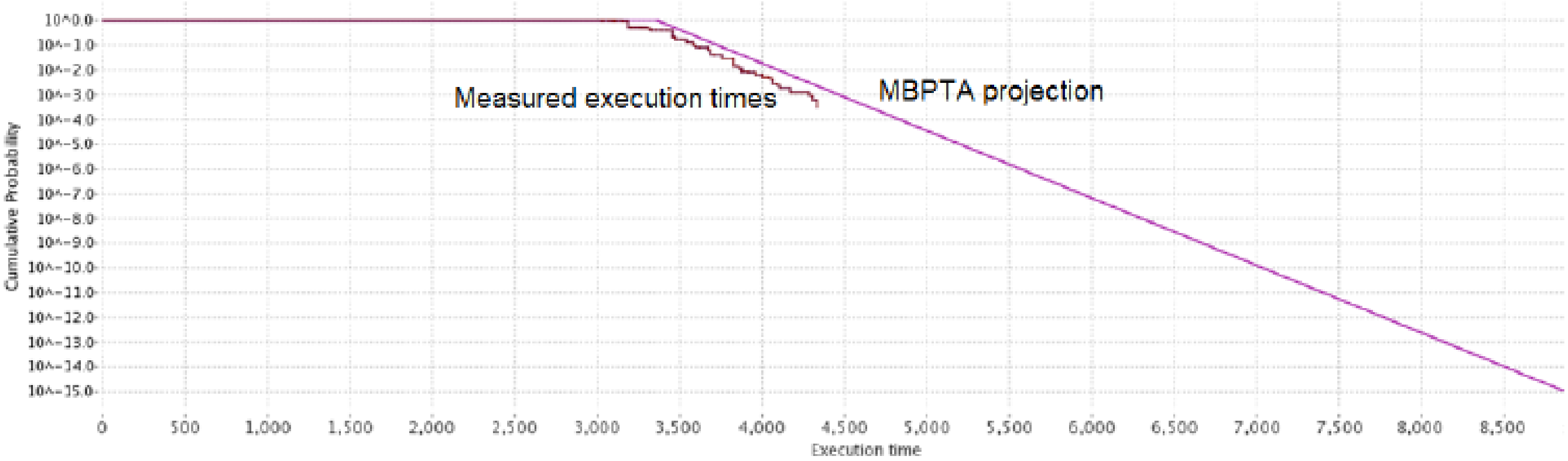
- PikeOS A653 person. & hypervisor
- MBPTA **HW randomized FPGA**
  - 4-core LEON3 on FPGA
  - IL1/DL1 caches (6KB 4-way)
  - Random repl.: iL1,dL1,iTLB,dTLB
  - Random placement IL1 and DL1





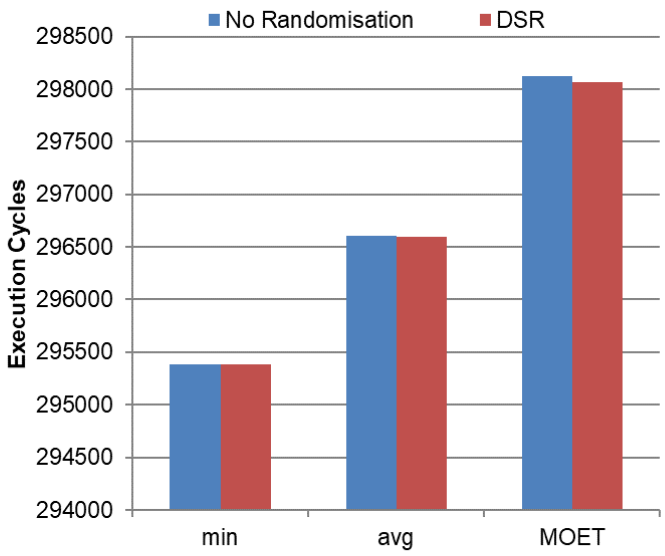
ESA application. Hardware Randomization

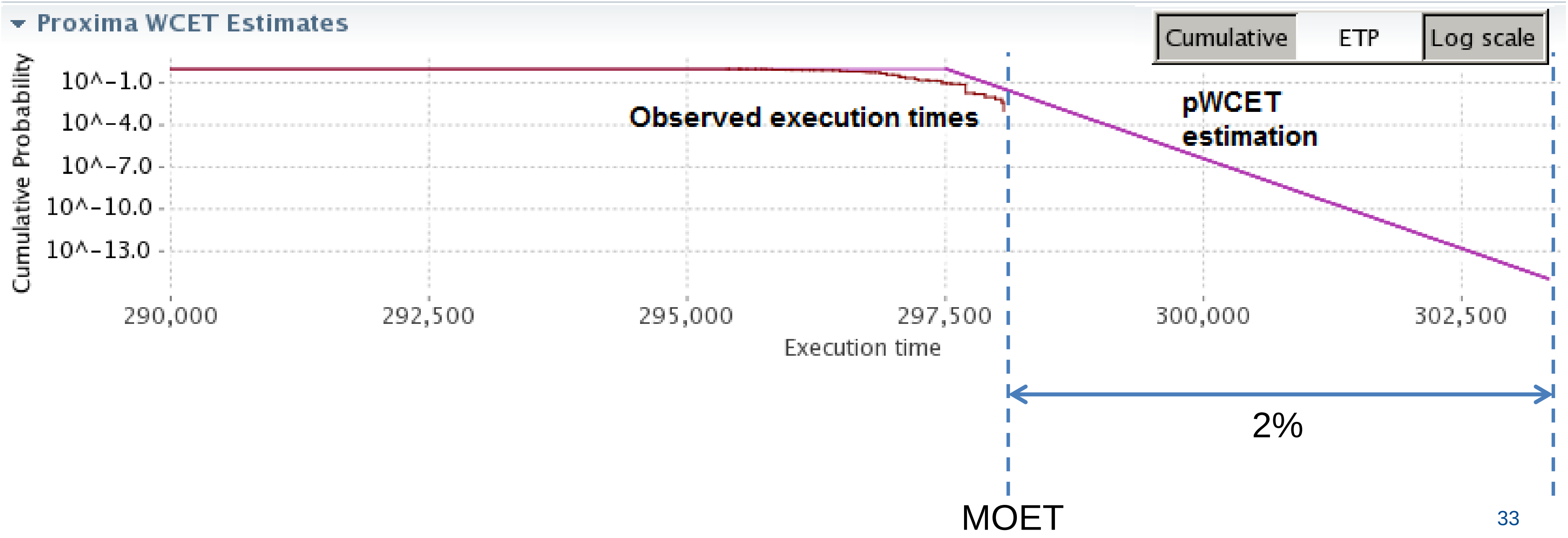
- Thrust Vector Control by the European Space Agency (ESA)
  - Fixed priority scheduler with 3 periodic tasks
  - Automatically generated C code from high-level model of the closed-loop system
  - Run bare-metal
  - Sensor data acquisition, actuator control in X-axis and in Y-axis



ADS application. SWRand

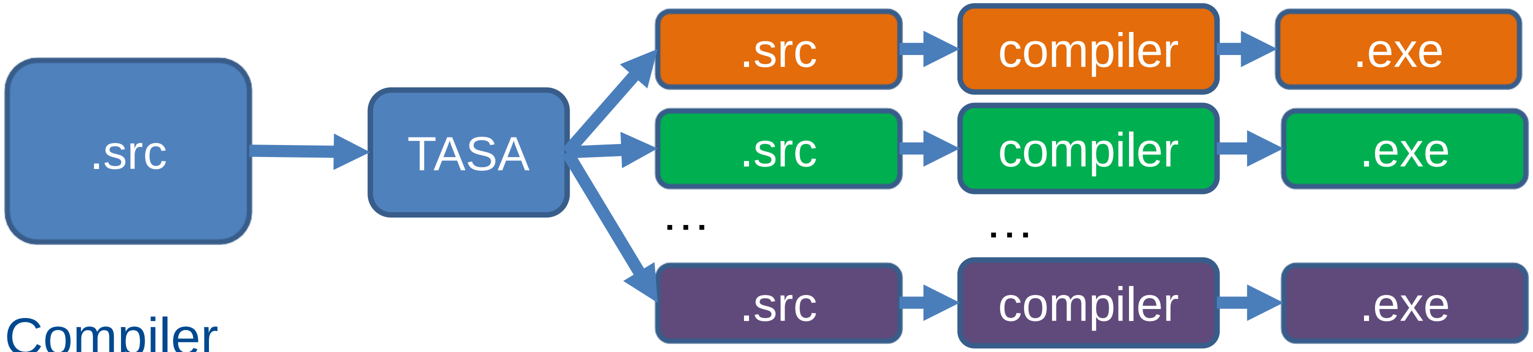
- High-critical control application that controls Mirror Displacements
- SWRAnd
  - <2% on number of instructions
  - No impact on execution time





What we can offer you?

- SWRand technology
  - Source-to-source compiler that generates functionally equivalent source versions of the input code with random memory layout

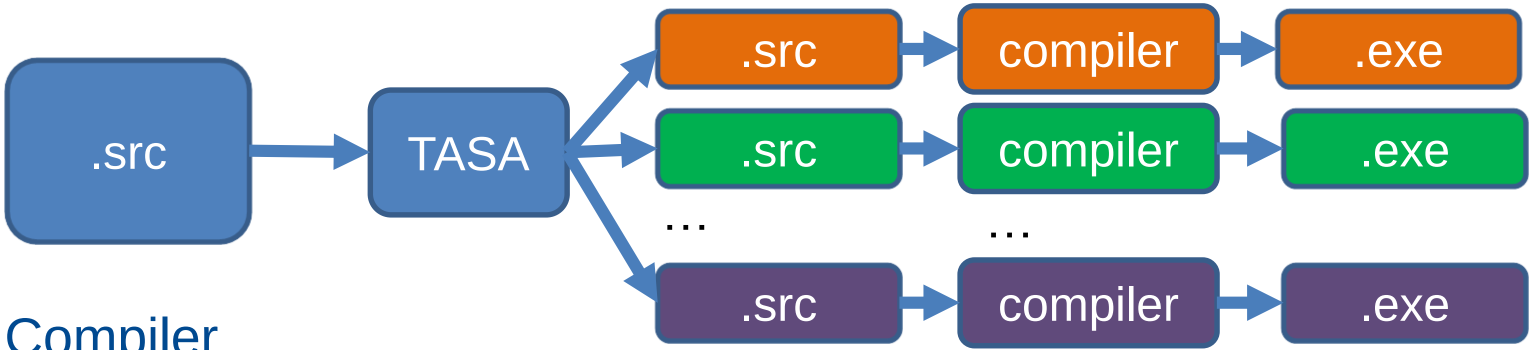


  - Compiler
    - Support to implement memory layout randomization features in your preferred compiler, linker and OS

What we can offer you?

SWRand technology

- Source-to-source compiler that generates functionally equivalent source versions of the input code with random memory layout

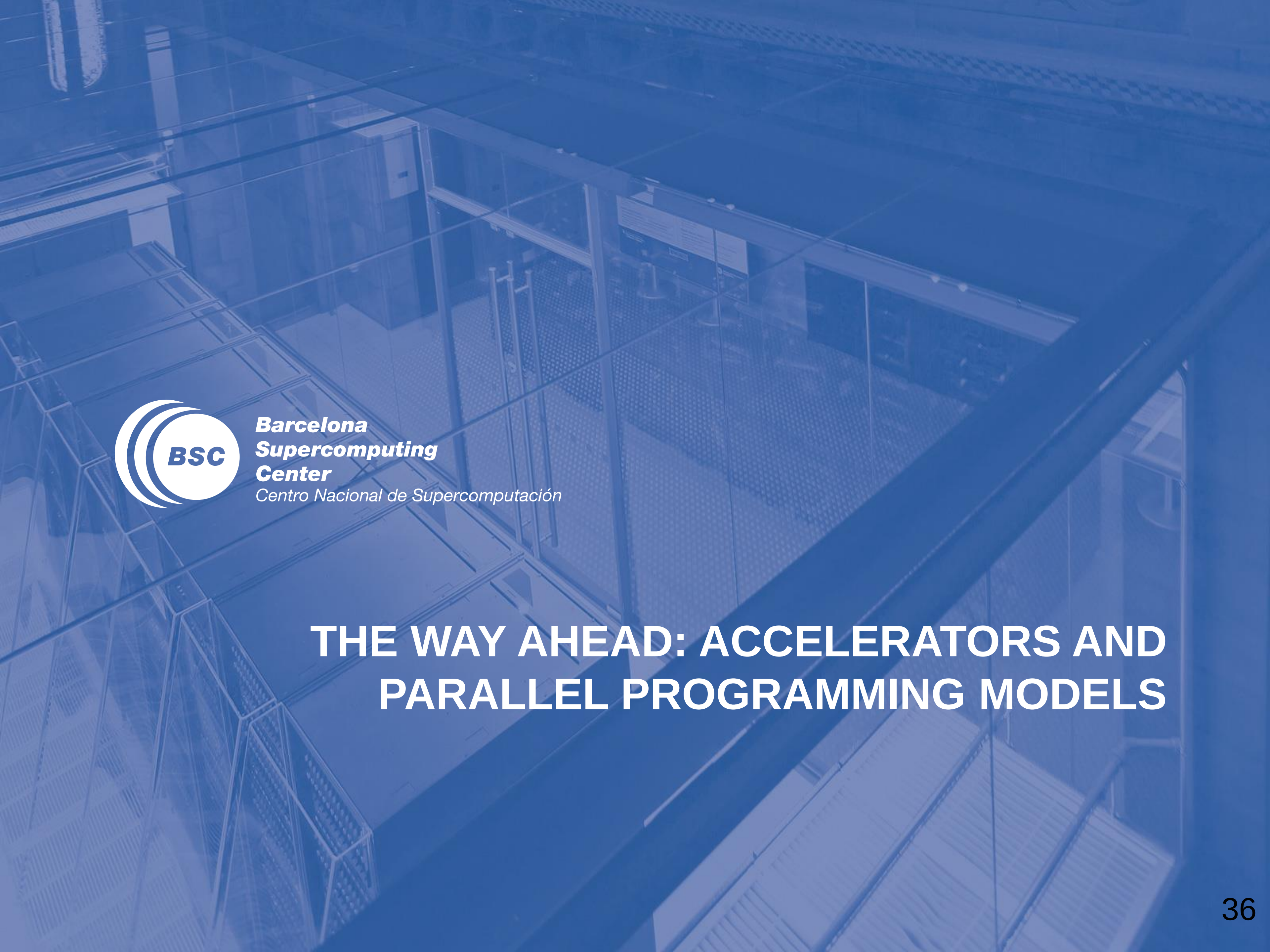


- Compiler
  - Support to implement memory layout randomization features in your preferred compiler, linker and OS

HWRand technology

- LEON3+ platform with randomization features implemented
- http://www.gaisler.com/index.php/products/processors/leon3

COBHAM



Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

THE WAY AHEAD: ACCELERATORS AND PARALLEL PROGRAMMING MODELS

Why accelerators?

- Critical systems get increasingly complex
  - New Advanced Driving Assistance Systems (ADAS) in automotive
  - Autonomous Guidance, Navigation and Control (GNC) in space
- Current µ-controllers cannot provide required performance
  - Safety standards (ISO 26262, ECSS): strict timing, reliability, safety

Why accelerators?

- Critical systems get increasingly complex
  - New Advanced Driving Assistance Systems (ADAS) in automotive
  - Autonomous Guidance, Navigation and Control (GNC) in space
- Current µ-controllers cannot provide required performance
  - Safety standards (ISO 26262, ECSS): strict timing, reliability, safety
- Embedded Heterogeneous Architectures are the solution
  - High Performance
  - Low Power
  - So far only used in non-critical markets, e.g. mobile phones

Why accelerators?

- Critical systems get increasingly complex
  - New Advanced Driving Assistance Systems (ADAS) in automotive
  - Autonomous Guidance, Navigation and Control (GNC) in space
- Current µ-controllers cannot provide required performance
  - Safety standards (ISO 26262, ECSS): strict timing, reliability, safety
- Embedded Heterogeneous Architectures are the solution
  - High Performance
  - Low Power
  - So far only used in non-critical markets, e.g. mobile phones
- Many potential options for the automotive market:
  - NVIDIA PTX
  - Qualcomm recently acquired NXP/Freescale
  - Other Embedded CPUs/GPUs: ARM, Imagination Technologies
  - FPGAs: Xilinx, Intel (acquired Altera)
  - Many-cores (Kalray MPPA, Texas Instruments Keystone etc)

The beginning of a new era in critical systems: Challenges

- Several options for embedded heterogeneous platforms
- …but no certification/qualification yet
  - High-Cost and Effort
  - Only justified by customer interest and high-volumes

The beginning of a new era in critical systems: Challenges

- Several options for embedded heterogeneous platforms
- …but no certification/qualification yet
  - High-Cost and Effort
  - Only justified by customer interest and high-volumes
- Industry: Which heterogeneous platform is better for my computationally intensive system?
  - No Open representative ADAS or avionics applications and benchmarks
  - EEMBC’s ADASMARK Benchmark still under development
  - Several industrial and academic works, all targeting specific hardware, closed-source developments

The beginning of a new era in critical systems: Challenges

- Several options for embedded heterogeneous platforms
- …but no certification/qualification yet
  - High-Cost and Effort
  - Only justified by customer interest and high-volumes
- Industry: Which heterogeneous platform is better for my computationally intensive system?
  - No Open representative ADAS or avionics applications and benchmarks
  - EEMBC's ADASMARK Benchmark still under development
  - Several industrial and academic works, all targeting specific hardware, closed-source developments
- We need open representative accelerator applications and benchmarking studies:
  - Trompouki et al, An Open Benchmark Implementation for Multi-CPU Multi-GPU Pedestrian Detection in Automotive Systems, ICCAD 2017

Accelerators in critical systems: Challenges (2)

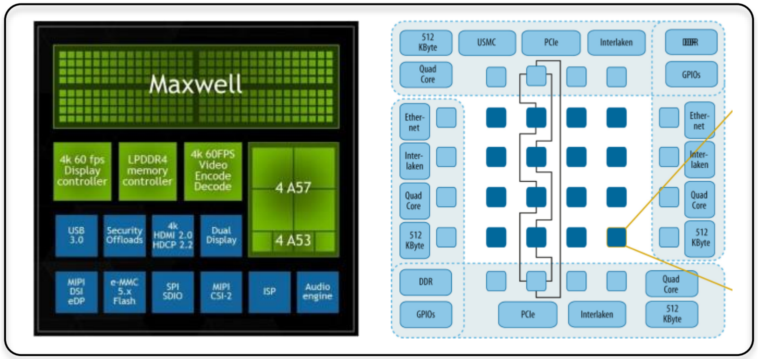
- Timing is essential for critical real-time systems
- Almost no studies so far in the embedded domain
  - Still we struggle with the WCET of CPUs
  - Highly parallel → highly unpredictable
  - Optimised for throughput, not latency
  - Few works in the desktop domain
    - Not the same: Fundamental architectural differences → Low-Power
- Potential Solutions:
  - Custom Designs
  - Software only at programming and runtime level

Accelerators in critical systems: Challenges (3)

- Programmability
- Different programming model compared to traditional CPU programming

Conventional Models

Parallel Programming Models



- Are the existing tools
  - suitable for critical systems? Certification, libraries …
  - efficient to use? Extract the available performance from the hardware?
  - enabling programmer’s productivity?
  - re-usable from platform to platform? Functional and Performance portability?

Parallel Programming Models

- Based on the **principle** that developers specify *what the application does and not how it is done*
  - Parallel computation is not fully controlled by the programmer, but by run-time mechanisms
- **Improves productivity** in terms of providing better programmability, portability and performance … ☺
- … at the expense of **complicating deriving safety guarantees** ☹

- Our research focuses on **OpenMP and GPU parallel programming models**

Why OpenMP?

- **Mature language** constantly reviewed and augmented (last release Nov 2015)
- **Programmability**
  - Support for fine-grain data- and task-parallelism very convenient to develop real-time embedded systems
  - Allows incremental parallelization
- **Performance** and **efficiency**
  - Similar to other models (e.g. TBB, CUDA, OpenCL and MPI)
  - Features an advanced accelerator model for heterogeneous computing
- **Portability**
  - Supported by many chip and compiler vendors (Intel, IBM, ARM, TI, Kalray, Gaisler)

OpenMP and Safety

- **OpenMP enables guaranteeing safety** requirements
  - Time predictability
    - Reasoning about the timing behaviour of the parallel execution
      - Worst-case response time analysis by means of schedulability analysis techniques
      - Dynamic and static resource allocation approaches supported
  - Safety and correctness
    - Ensuring that the correct operation in response to its inputs
    - Support **reliability** and **resiliency** mechanisms in terms of
      - Compiler analysis techniques for checking parallel programming correctness, avoiding deadlock and race conditions scenarios
      - Error handling methodologies

What we can offer you:

OpenMP:

- Development framework (compiler and run-time) supporting different multi-core and many-core parallel platforms
  - Kalray MPPA, TI Keystone, NGMP+ RTEMS SMP (under the contract "*Parallel Programming Models for Space Systems*" ESA Contract No. 4000114391/15/NL/Cbi/GM)
- Research to include OpenMP within Ada is being conducted
- Definition of a new OpenMP specification group to support safety requirements within the OpenMP language committee
  - Join Us!

What we can offer you:

OpenMP:

- Development framework (compiler and run-time) supporting different multi-core and many-core parallel platforms
  - Kalray MPPA, TI Keystone, NGMP+ RTEMS SMP (under the contract "*Parallel Programming Models for Space Systems*" ESA Contract No. 4000114391/15/NL/Cbi/GM)
- Research to include OpenMP within Ada is being conducted
- Definition of a new OpenMP specification group to support safety requirements within the OpenMP language committee
  - Join Us!

Embedded GPUs/FPGAs:

- Unified programming (compiler+runtime) between different accelerators: CUDA, OpenCL, Graphics and Compute APIs
- Embedded GPU Benchmarking for critical domains

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

CONCLUSIONS

Conclusions

1. Multi-core and Many-core (MMC) contention
   - Concept and proposed solutions

2. A Probabilistic Angle to WCET estimation
   - Concept and maturity of existing tools

3. The Way Ahead: accelerators and parallel programming models in critical systems
   - Challenges and our work in this domain

BSC

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Executing Parallel Real-time Software on Multi- and Manycores in a Timely Manner

Presenters: **Francisco J. Cazorla,** Head of the Computer Architecture/ Operating System group at BSC
Leonidas Kosmidis, Senior Researcher at BSC

Contributors: **Eduardo Quinones, Jaume Abella, Carles Hernandez, Enrico Mezzetti,** Senior Researchers at BSC

11th ESA ADCSS2017

ESTEC– October 19th, 2017