A Practical Methodology to Tightly Upperbound Contention in COTS Multicores

Gabriel Fernández
Francisco J. Cazorla
Enrico Mezzetti
Jaume Abella
(BSC)

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080."

SAFURE

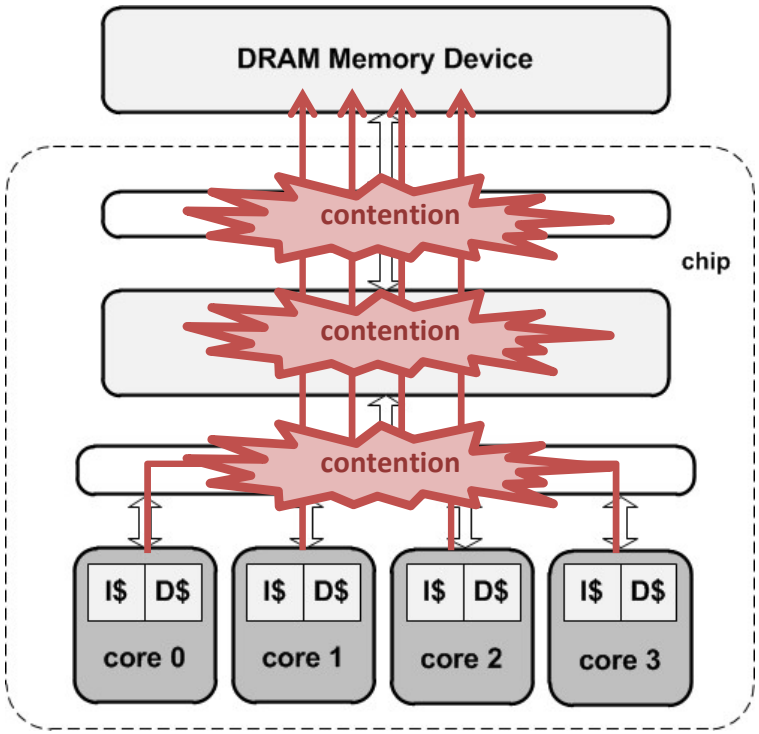
SAFety and secURity by dEsign for interconnected mixed-critical cyber-physical systems

Motivation

- Real-time systems need to **guarantee timing correctness**
  - Timing analysis: Providing a Worst Case Execution Time (WCET)
- **Multicore systems** constitute the future computing platform for real-time systems
  - WCET must hold in multicores
- Real-time industry resorts to **COTS processors** to obtain performance at an affordable cost

Contention in COTS Multicores

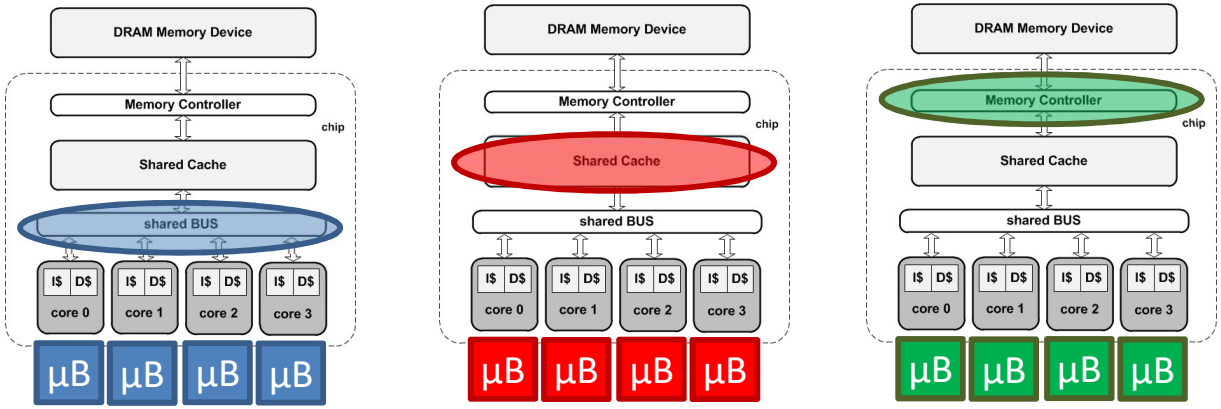
- Multicores challenge timing analysis due to the **impact of contention**
  - WCET must be preserved at all times to prevent resource clogging
  - But resource sharing challenges WCET estimates



- ***Challenge: How to account for the impact of contention in WCET estimates and guarantee that they hold during operation?***

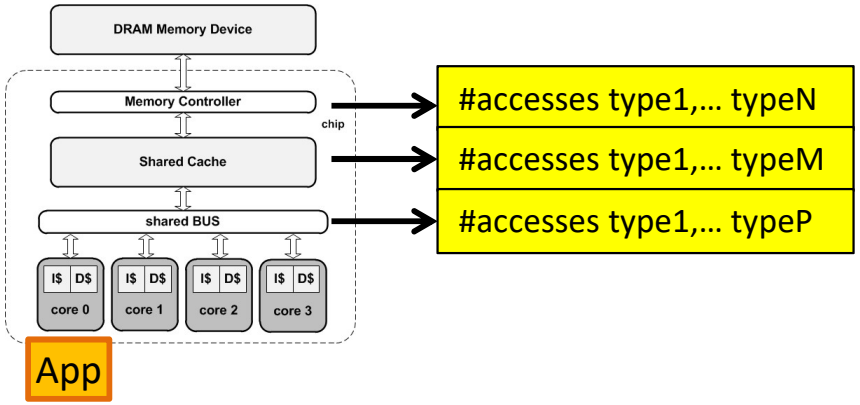
Practical Approach

STEP 1: Maximum impact of contention per access per resource



Stressing benchmarks (µB)
Specialized assembly programs using a resource intensively

STEP 2: Access counts to each shared resource for APP analyzed



STEP 3: Estimate maximum contention

APP1 #accesses **X** Worst impact | Worst impact | Worst impact

Fully time composable bound:
Pessimistic but valid under any workload

APP1 #accesses
APP2 #accesses
X Worst impact | Worst impact | Worst impact

Partially time composable bound:
Tighter but valid under limited load

Practical Approach: STEP 1

- **Quantify the access delay** to each shared resource
  - Use stressing benchmarks (μB) stressing those resources
  - Measure accesses and execution time with Performance Monitoring Counters (PMCs)

- Architecture **specifications must be understood** to design μB
  - Continuous access to ad-hoc arrays with specific sizes, access strides, delay between accesses,…

- Technology **based on measurements**
  - Needs adaptation to target architecture
  - …but it is portable to many (all?) architectures

Practical Approach: STEP 2

- Application characterization
  - **Quantify (upper-bound) the number of accesses** of the application under study to shared resources
- **Application is run in isolation**
  - Compatible with unit testing (no need to have full system integrated)
  - Contention can be estimated at early design stages
- Quantification **based on measurements**
  - Relies on PMCs

Practical Approach: STEP 3

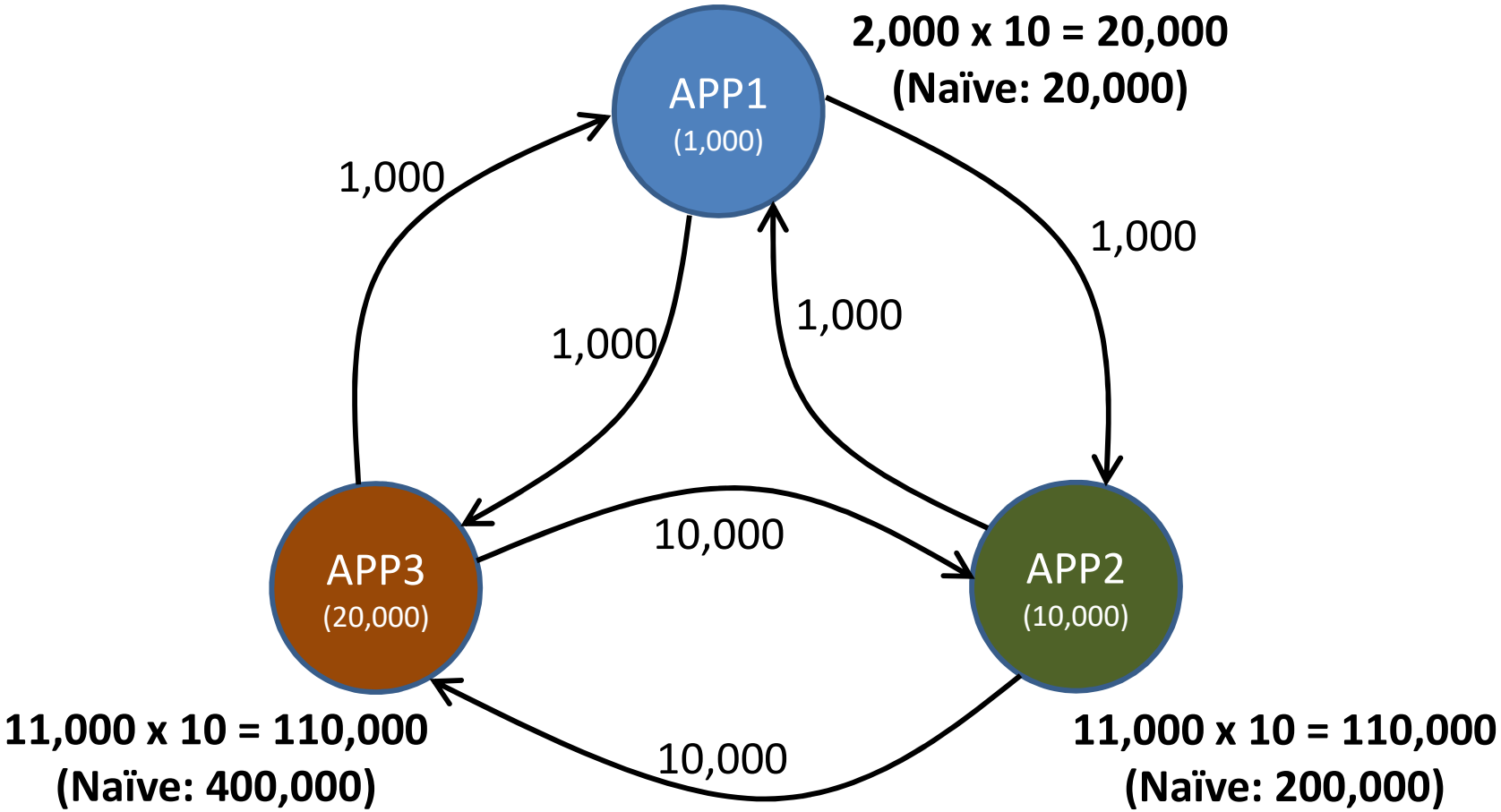
- Bounding contention
  - **Analytic (simple) models based on measurements** from STEP 1 and STEP 2
- **Accuracy and tightness strictly dependent on the PMC infrastructure**
  - Specific events monitored, separation of events per type, counters for events or stall cycles,…
- Contention estimation
  - **For *any* workload**: all accesses assumed to experience maximum contention
    - Pessimistic but fully time composable
  - **For specific loads**: maximum contention produced by specific load (e.g. leveraging info about contender applications)
    - Tighter but partially time composable

Time Composability

Fully time composable bounds	Partially time composable bounds
No assumption made on contenders, no need to know anything about contenders	**Access information** about contenders (or load types) needs to be leveraged
Provides worst contention ever (**likely pessimistic**)	Provides bounds to specific degrees of contention (**much tighter**)
Bounds hold valid in any scenario, so **no need for monitoring**	Bounds only valid under specific access counts, so **runtime monitoring needed** (if access counts exceeded, RTOS may have to intervene)

Time Composability

- Illustrative example

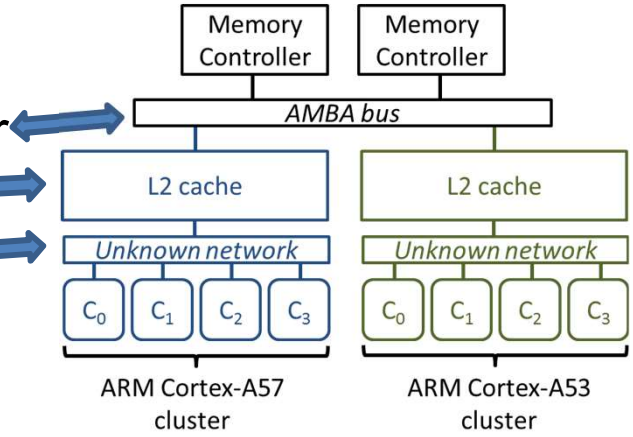


Integration on Real Platforms

- **DragonBoard 810**
  - Failed integration
- **Juno development board**
  - Some degree of success
- **Infineon AURIX TC27x**
  - Successful integration
  - Evaluation on an automotive use case
- Integration on a commercial toolset (**RapiTime**)
- *Also integrated on Gaisler LEON4, and ongoing on NXP T2080 and Xilinx Zynq (not presented today)*

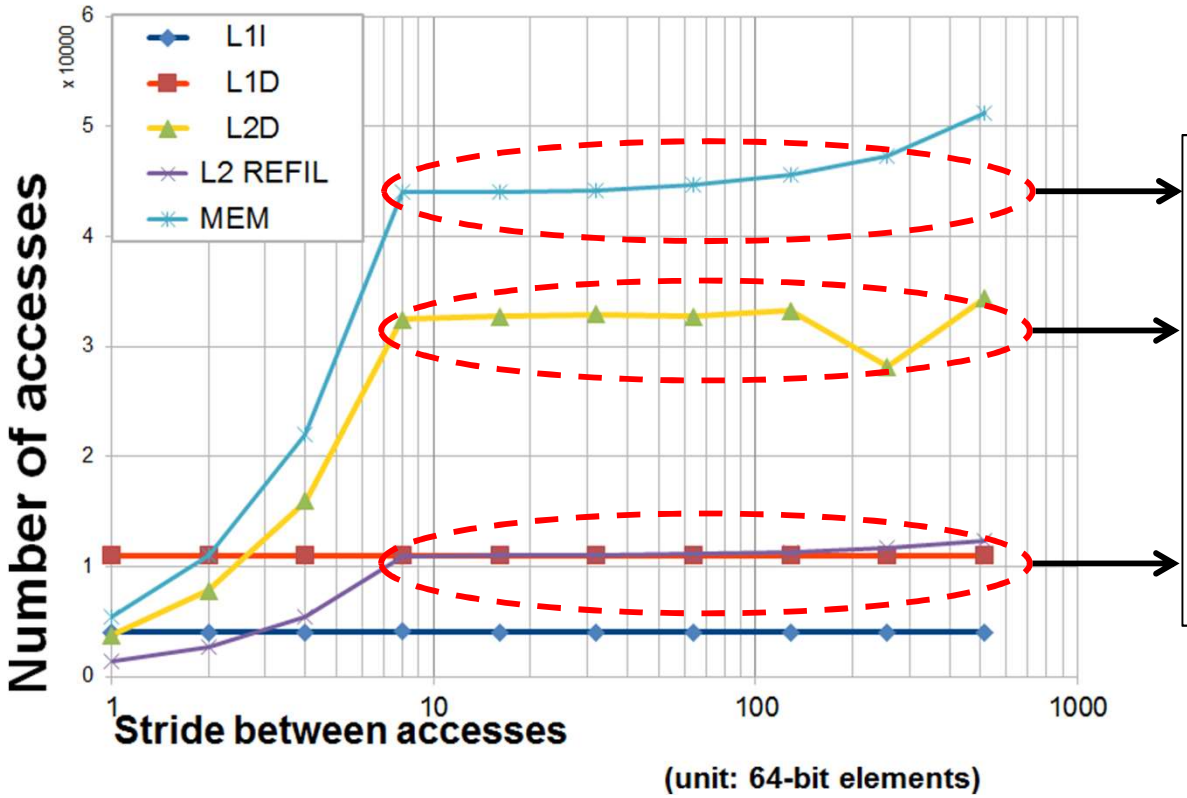
DragonBoard

- SnapDragon 810 processor (ARM big.LITTLE)
  - Implements a A57 **4-core high-performance** cluster and a A53 **4-core low-power** cluster
  - On-chip shared resources
    - Access to shared memory controller
    - L2 cache space
    - Access to shared L2
  - PMC support limited
- Methodology
  - Microbenchmarks with diverse behavior
    - “Zero” off-core accesses
    - Sustained L2 accesses (hits)
    - Sustained memory accesses (L2 misses)



DragonBoard

- **Inability to disable prefetcher**
  - Command in documentation simply fails
  - The same command on other boards works successfully



Once stride is large enough (each access to a different cache line), we would expect:

1) All L1D accesses are misses
2) Each L1D access produces **1 L2D** and **1 MEM** Access

Reality: each L1D miss causes **3 L2D** accesses and **4 MEM** accesses. Prefetcher produces uncontrolled activity!!

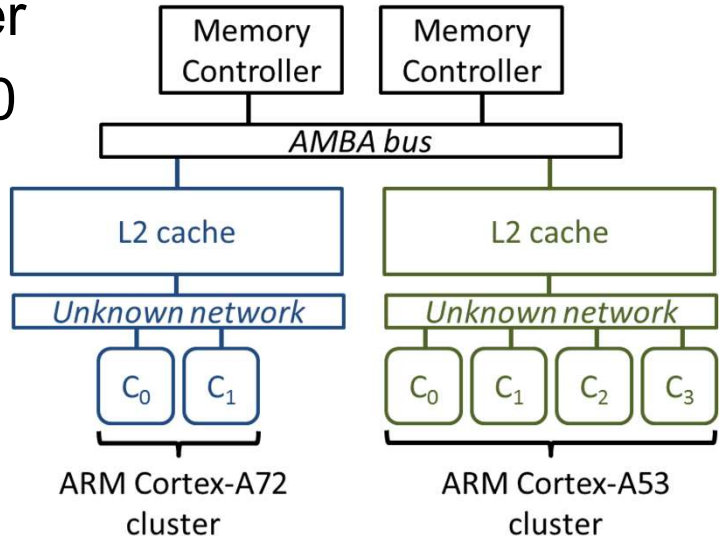
DragonBoard

- Different providers
  - Clusters designed by **ARM**
  - Integrated on SnapDragon 810 by **QUALCOMM**
  - QUALCOMM applied **modifications unmatched in ARM's documentation**
  - Lack of detailed documentation on components beyond ARM clusters

- Conclusion
  - **Platform not usable for critical real-time systems**
  - Details on usability and characteristics need to be released to enable its use

Juno Development Board

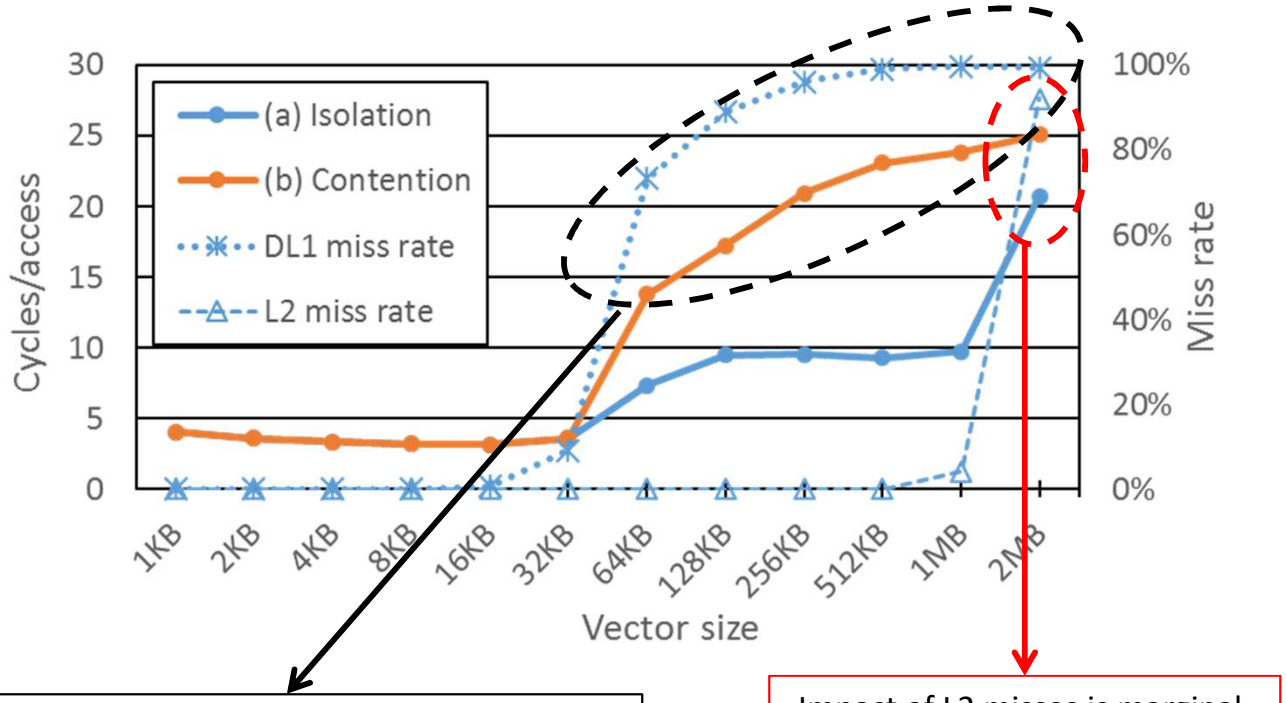
- ARM big.LITTLE processor
  - Implements a A72 **2-core high-performance** cluster and a A53 **4-core low-power** cluster
  - Very similar to SnapDragon 810
  - PMC support limited



- Main differences with DragonBoard
  - **Full platform developed by a single provider** (ARM)
  - Although PMCs and documentation are limited, they are usable and accurate

Juno Development Board

- Platform characterization successful
  - Information still limited due to limited PMCs available



Impact of contention grows linearly with DL1 miss rate, so with the number of L2 accesses

Latency grows from ~10 to ~25 cycles due to contention

Impact of L2 misses is marginal. Execution time dominated by contention in the access to L2

Work done in collaboration with Sylvain Girbal (TRT)

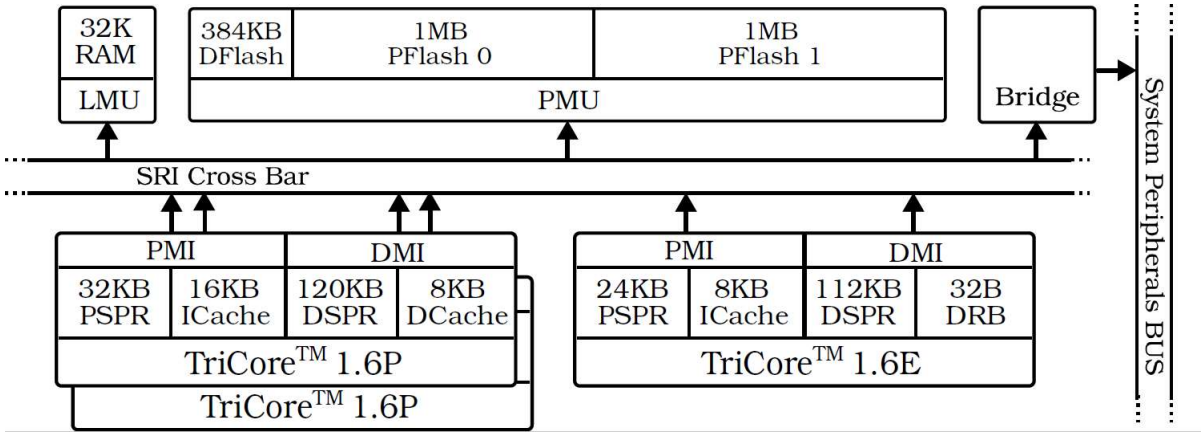
Juno Development Board

- **Single provider**
  - Documentation reliable
  - µB allow stressing shared resources and expose impact of contention
  - Still, lack of details and scarce PMCs limit the confidence had on whether scenarios observed are truly the worst ones

- Conclusion
  - **Platform usable** for critical real-time systems, **but some uncertainty remains**
  - Details on usability and characteristics need to be released to increase confidence on results

Infineon AURIX TC27x

- Infineon's **multicore for safety-critical automotive systems**
  - Implements 2 TC1.6P (high-performance) cores and 1 TC1.6E (low power) core
  - Crossbar interconnect
  - Shared RAM and Flash memories
  - Local scratchpads also reachable remotely
  - PMCs allow counting events and stall cycles for some (not all) relevant events



Infineon AURIX TC27x

- Platform characterization successful
  - Limited PMCs require **some pessimistic assumptions** on access counts
- Contention model developed successfully
  - **ILP model to upper bound contention**
  - Builds upon PMC measurements only
  - Compatible with fully and partially time composable estimates

$$\Delta^{cont}_{b\to a} = [\Delta cs^{co}_a] + [\Delta cs^{da}_a] = \left[n^{pf0,co}_{b\to a} \times l^{pf0,co} + n^{pf1,co}_{b\to a} \times l^{pf1,co} + n^{lmu,co}_{b\to a} \times l^{lmu,co}\right] + \left[n^{dfl,da}_{b\to a} \times l^{dfl,da} + n^{pf0,da}_{b\to a} \times l^{pf0,da} + n^{pf1,da}_{b\to a} \times l^{pf1,da} + n^{lmu,da}_{b\to a} \times l^{lmu,da}\right]$$

$$n^{dfl,da}_{b\to a} = min(n^{dfl,da}_a, n^{dfl,da}_b)$$

$$n^{pf0,co}_{b\to a} \le min(n^{pf0,co}_a + n^{pf0,da}_a, n^{pf0,co}_b)$$

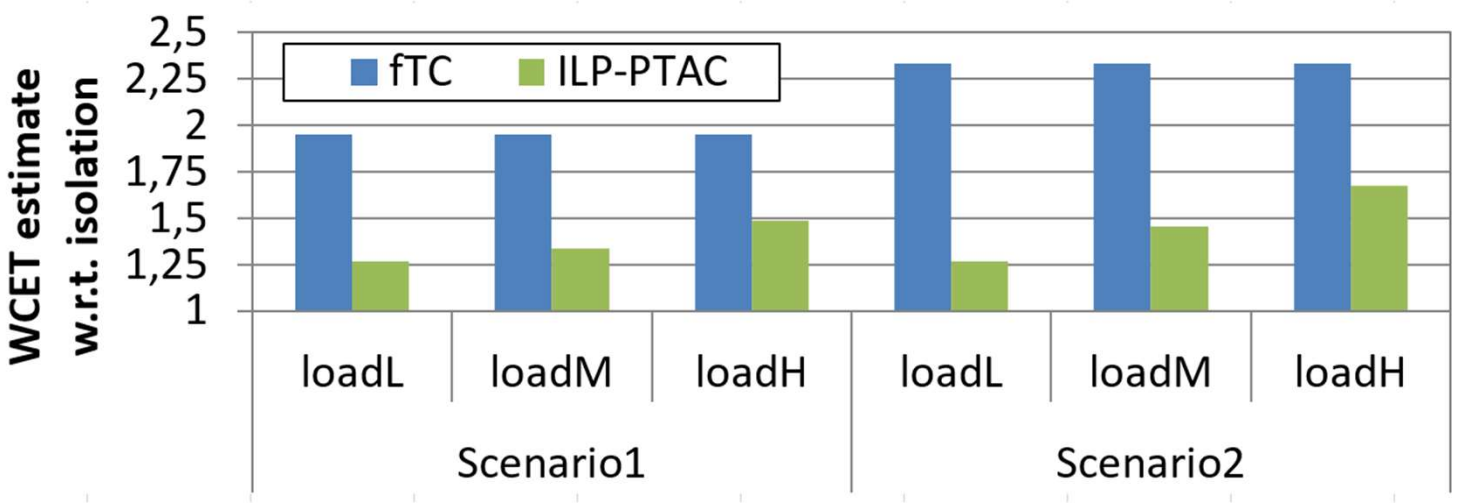
$$n^{pf0,da}_{b\to a} \le min(n^{pf0,co}_a + n^{pf0,da}_a, n^{pf0,da}_b)$$

$$n^{pf0,co}_{b\to a} + n^{pf0,da}_{b\to a} \le n^{pf0,co}_a + n^{pf0,da}_a$$

$$n^{pf1,co}_{b\to a} \le min(n^{pf1,co}_a + n^{pf1,da}_a, n^{pf1,co}_b)$$

Infineon AURIX TC27x

- Benchmark resembling a Cruise Control System
  - Code/data deployed in two ways (scenarios 1 and 2) to be particularly sensitive to contention
  - Results show effectiveness of partially time-composable ILP model against relatively low (loadL), medium (loadM) and high (loadH) contention



- Preliminary evaluation on a representative function of the **automotive use case** of Magneti-Marelli shows that execution time can increase up to **7.2% only** due to multicore contention
  - **Work done in collaboration with Magneti-Marelli**

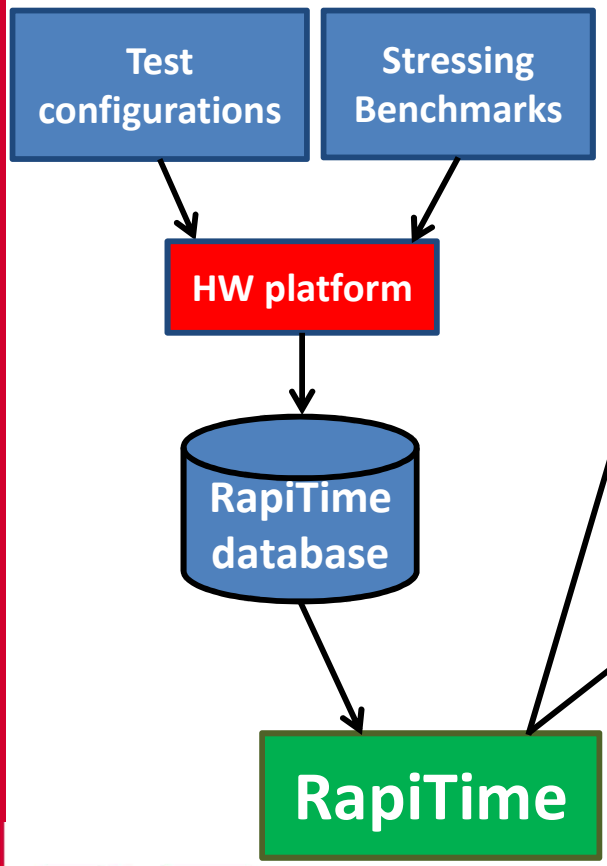
Infineon AURIX TC27x

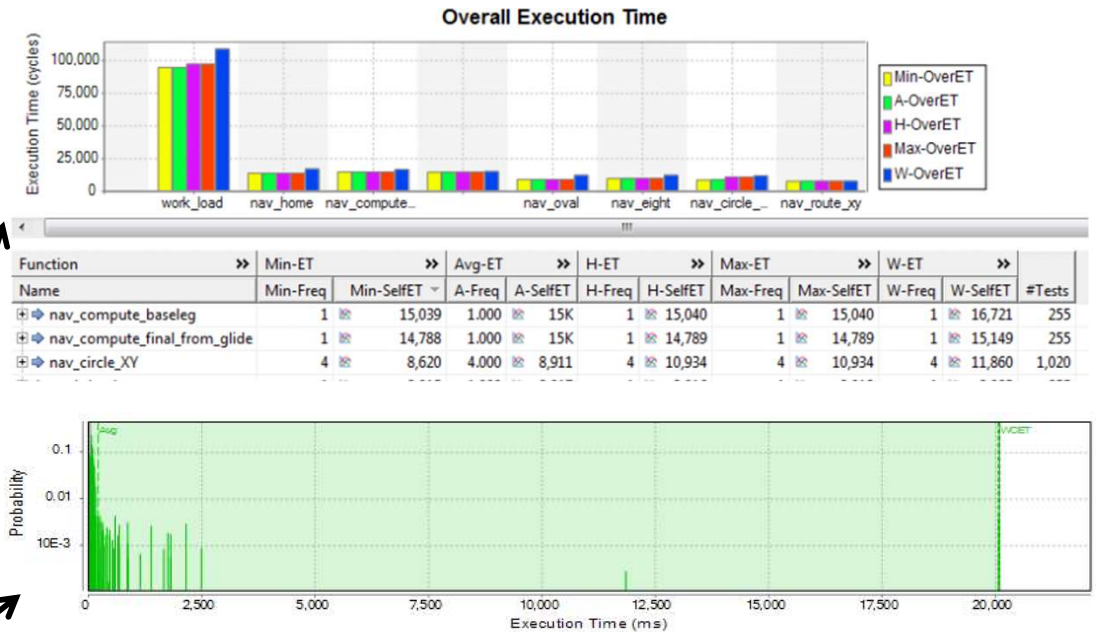
- **Platform intended for safety-related systems**
  - Documentation reliable
  - µB allow stressing shared resources and expose impact of contention
  - More details on PMCs
    - Still, some missing details led to potentially pessimistic assumptions

- Conclusion
  - **Platform suitable to model multicore contention**
  - Methodology integrated and applied successfully
  - Now evaluating larger software units with Magneti-Marelli

RapiTime Integration

- Powerful toolset for WCET estimation based on measurements
- We provide an extension towards multicores





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

Work ongoing in collaboration with Rapita Systems Ltd.

RAPITA SYSTEMS LTD

RapiTime Integration

- Our methodology needs to be integrated on a per-platform basis
  - **µB need being tuned for each platform** type
    - Minor modifications if target platform has the same architecture as the original one (e.g. from T2080 to P4080)
  - Measurements provided in standard RapiTime format, so toolset is fully portable

- **Currently being integrated for diverse platforms**
  - Infineon TC27x, TC29x (automotive)
  - NXP T2080 (avionics, railway)
  - Xilinx Zynq (multidomain)

Summary

- Multicore contention needs to be accounted for reliably and tightly
- Our **measurement-based methodology** enables modelling it
  - Results prove reliability and tightness
  - **Successful integration** on diverse platforms
  - Being integrated on further platforms
  - **Successful evaluation on industrial use cases**
- Methodology already being integrated on a **commercial tool set**
  - Proves high degree of maturity

SAFURE Grant Agreement No. 644080

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080."

"This work was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0025. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government."

If you need further information, please contact the coordinator:

TECHNIKON Forschungs- und Planungsgesellschaft mbH
Burgplatz 3a, 9500 Villach, AUSTRIA
Tel: +43 4242 233 55 Fax: +43 4242 233 55 77
E-Mail: coordination@safure.eu

The integration of our technology in a commercial toolset is beyond SAFURE objectives, so it is co-funded by the European Regional Development Fund (ERDF) of the European Union in the framework of the ERDF (FEDER) program in Catalonia 2014-2020 under the grant SDESI (2016 PROD 00115).

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