www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MC2: Multicore and Cache Analysis via Deterministic and Probabilistic Jitter Bounding

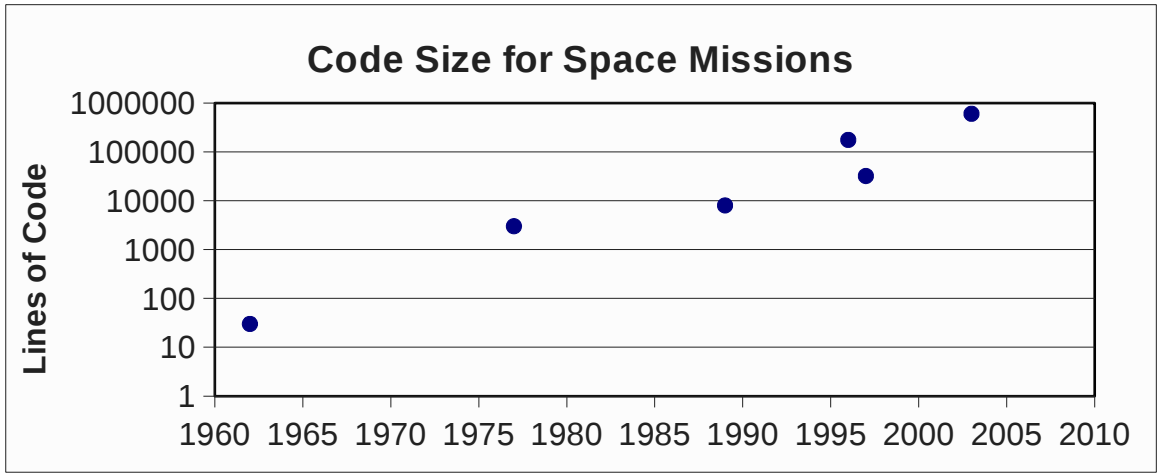
Enrique Díaz[1,2], Mikel Fernández[1], Leonidas Kosmidis[1], Enrico Mezzetti[1], Carles Hernandez[1], Jaume Abella[1], and Francisco J. Cazorla[1,3]

[1]Barcelona Supercomputing Center (BSC), Spain
[2]Universitat Politècnica de Catalunya, Spain
[3]IIIA-CSIC, Spain

22nd International Conference on Reliable Software Technologies
Ada-Europe 2017
Vienna, 15 June 2017

Critical Real-Time Embedded Systems (CRTES)

- CRTES steadily requiring increasing levels of computing power



[1]

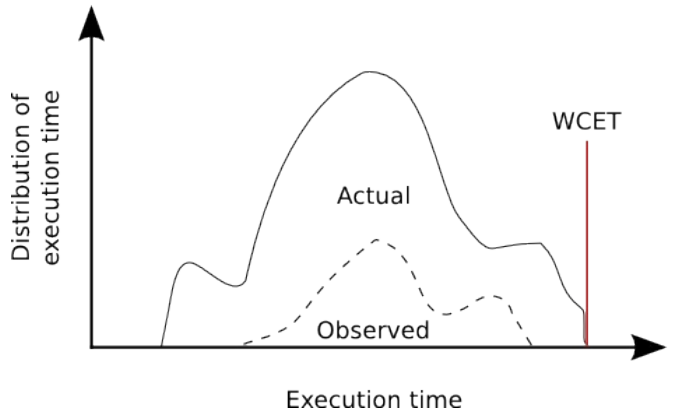
- Delivering high levels of computing power requires using high-performance hardware
  - Caches
  - Multicores

- However, those features challenge timing analysis

[1] Siewiorek, D., & Narasimhan, P. (2006). Fault tolerant architectures for space and avionics. In Workshop on Dependability in Robotics and Autonomous Systems.

Critical Real-Time Embedded Systems (CRTES)

- CRTES require providing evidence about timing correctness of the system against safety standards
  - Time budget must be preserved
  - Need to bound the WCET

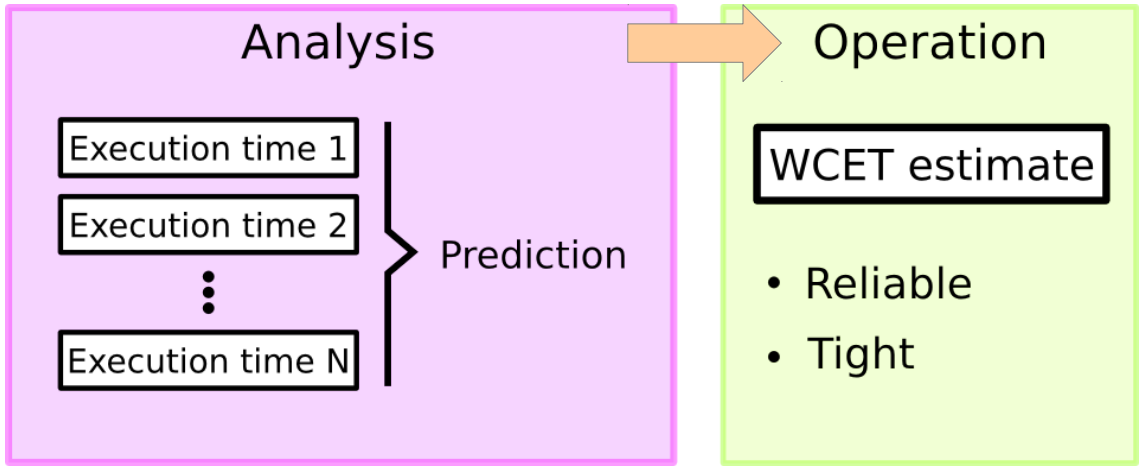


- Obtaining a reliable and tight WCET estimate is complex
  - Several methods
  - Rely on assumptions/inputs of the HW/SW

- Measurement Based Timing Analysis (MBTA)
  - Dominant (i.e. most used) technique in most real-time domains

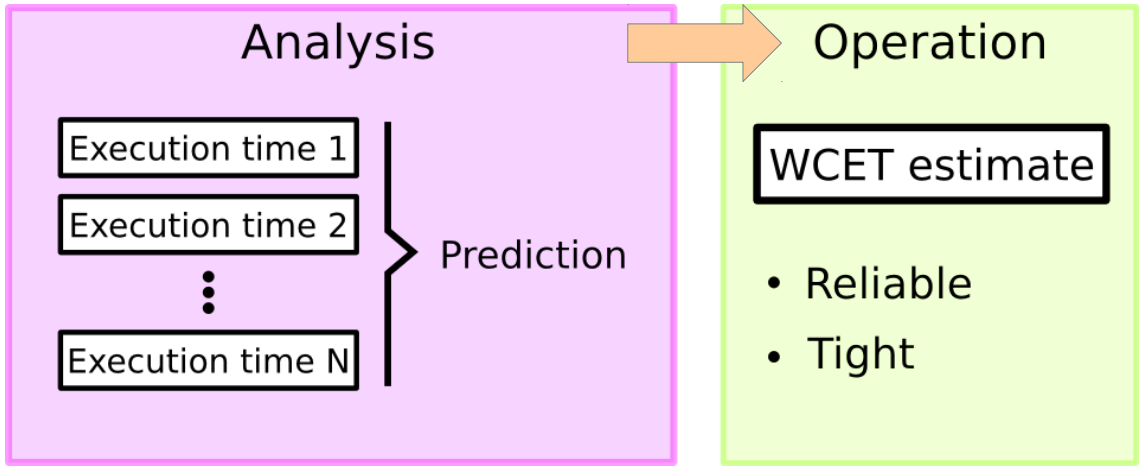
Measurement Based Timing Analysis (MBTA)

Analysis Time - Operation Time



Measurement Based Timing Analysis (MBTA)

- Analysis Time - Operation Time



- Quality of WCET estimates builds on **representativeness**
  - User's ability to relate analysis time and operation time
  - **The end user has to:**
    - Capture worst conditions that can arise at operation
    - That is, capture worst-case behaviors of each jittery resource
      - Cache → worst cache layout
      - Shared resources → worst contention

Agenda

- **Measurement Based Probabilistic Timing Analysis (MBPTA)**
  - **Statistical Analysis**
  - **Jtitter control: Deterministic - Probabilistic upperbounding**
- Dealing with Cache Jitter
- Dealing with Contention Jitter
  - Fully time composable (fTC)
  - Partially time composable (pTC)
- Evaluation
- Conclusions

Measurement-Based **Probabilistic** Timing Analysis (MBPTA)

- MBPTA applies Extreme Value Theory (EVT) on execution time observations to derive probabilistic WCET (pWCET)
  - Analyses the tail of the distribution
  - Predicts the probability of observed events to appear simultaneously (does not capture unobserved events)
    - Assurance of all relevant events observed → representativeness is assumed to come from hardware

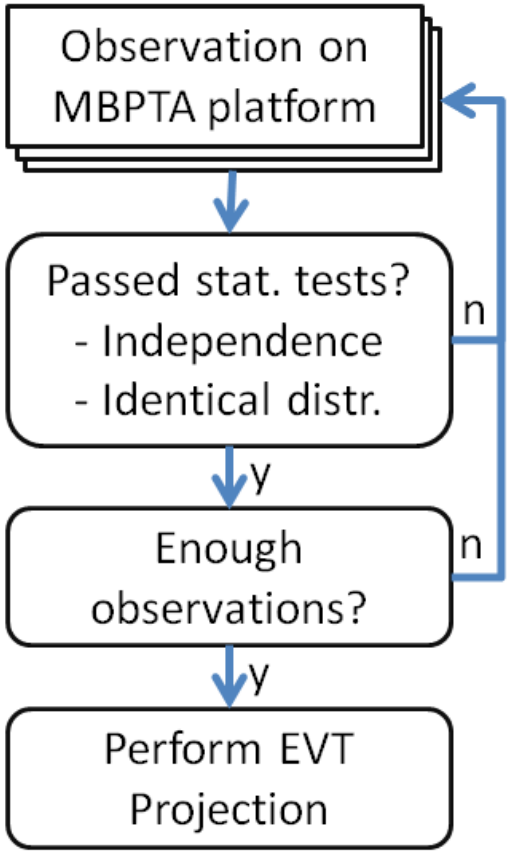
Measurement-Based **Probabilistic** Timing Analysis (MBPTA)

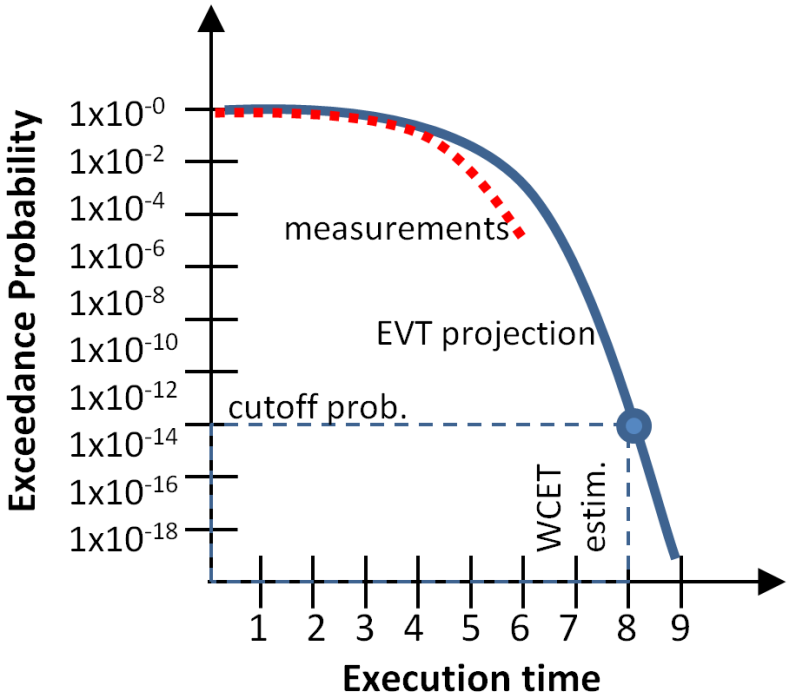
- MBPTA applies Extreme Value Theory (EVT) on execution time observations to derive probabilistic WCET (pWCET)
  - Analyses the tail of the distribution
  - Predicts the probability of observed events to appear simultaneously (does not capture unobserved events)
    - Assurance of all relevant events observed → representativeness is assumed to come from hardware

- **Hardware ensures that variability that can arise at operation emerges naturally from observations taken at analysis**
  - Relieves the user from controlling hardware sources of jitter
  - Increase confidence on WCET estimates

Measurement-Based **Probabilistic** Timing Analysis (MBPTA)

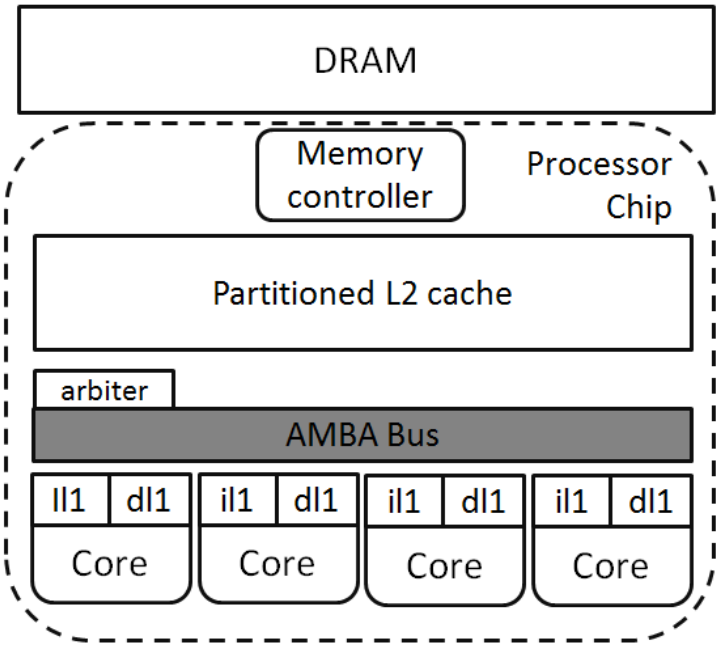
Summary of MBPTA





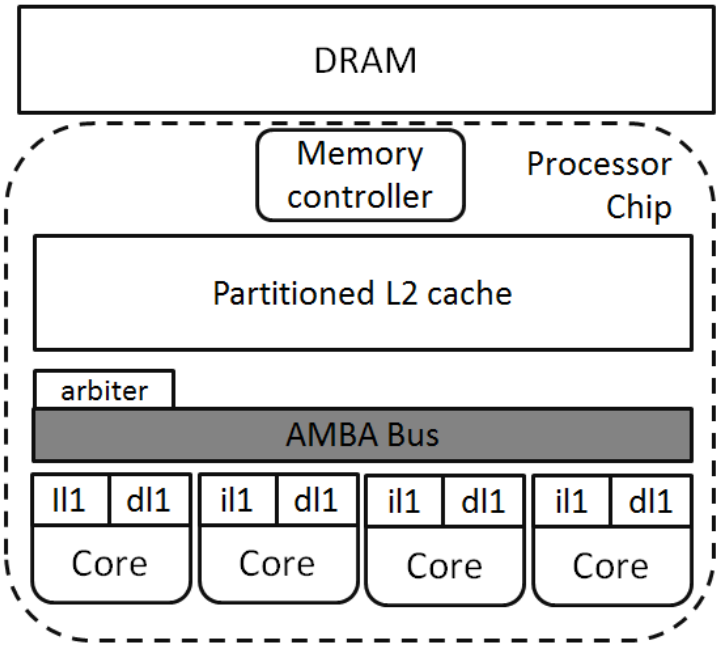
Reference platform

- 4 LEON3 cores
  - 1 executing a time-critical task (Task Under Analysis, TUA)
  - 3 executing non-critical tasks
- AMBA AHB without split requests
- Set of performance monitoring counters (PMCs)



Reference platform

- 4 LEON3 cores
  - 1 executing a time-critical task (Task Under Analysis, TUA)
  - 3 executing non-critical tasks
- AMBA AHB without split requests
- Set of performance monitoring counters (PMCs)



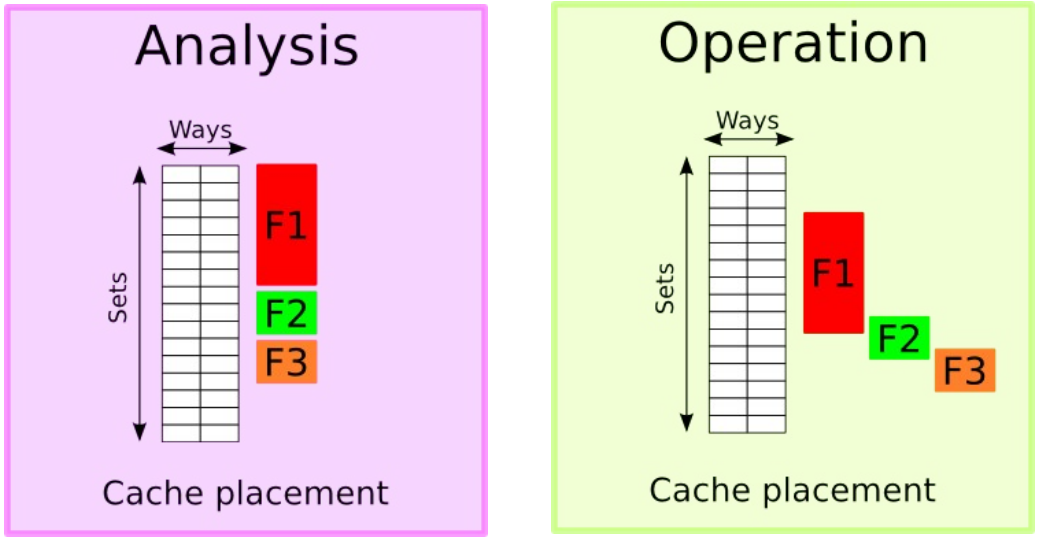
- **Hardware ensures that variability that can arise at operation emerges naturally from observations taken at analysis**
- In our platform:
  - Cache
  - Bus

Agenda

- Measurement Based Probabilistic Timing Analysis
  - Statistical Analysis
  - Deterministic and probabilistic upperbounding
- **Dealing with Cache Jitter**
- Dealing with Contention Jitter
  - Fully time composable (fTC)
  - Partially time composable (pTC)
- Evaluation
- Conclusions

Dealing with cache jitter

Problem

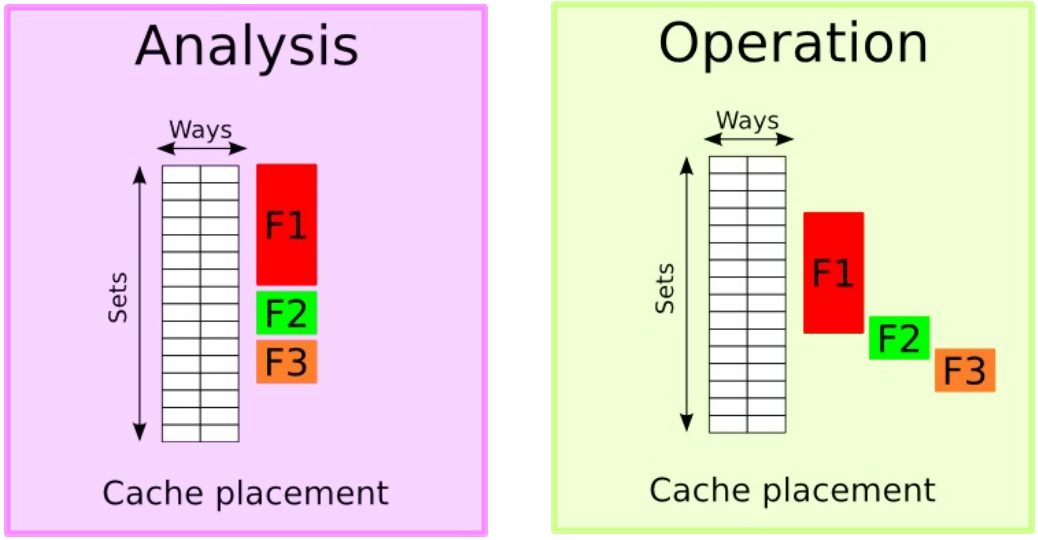


Incremental SW integration

WCET estimates
not valid at operation

Dealing with cache jitter

Problem



Incremental SW integration

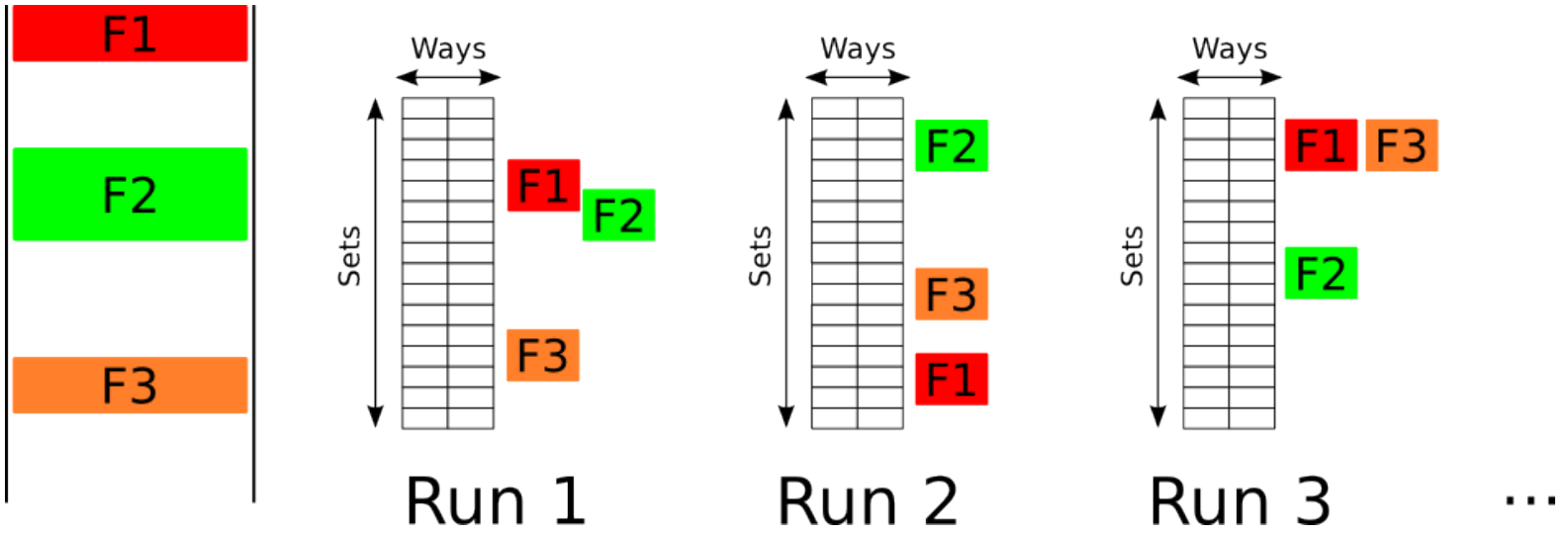
WCET estimates
not valid at operation

Randomization is used to provide MBPTA probabilistic behavior

- Random address to set mapping (random cache layout)
  - The more the runs made, the higher the number of cache layouts explored
- This allows cache jitter to be properly modeled by MBPTA

Dealing with cache jitter

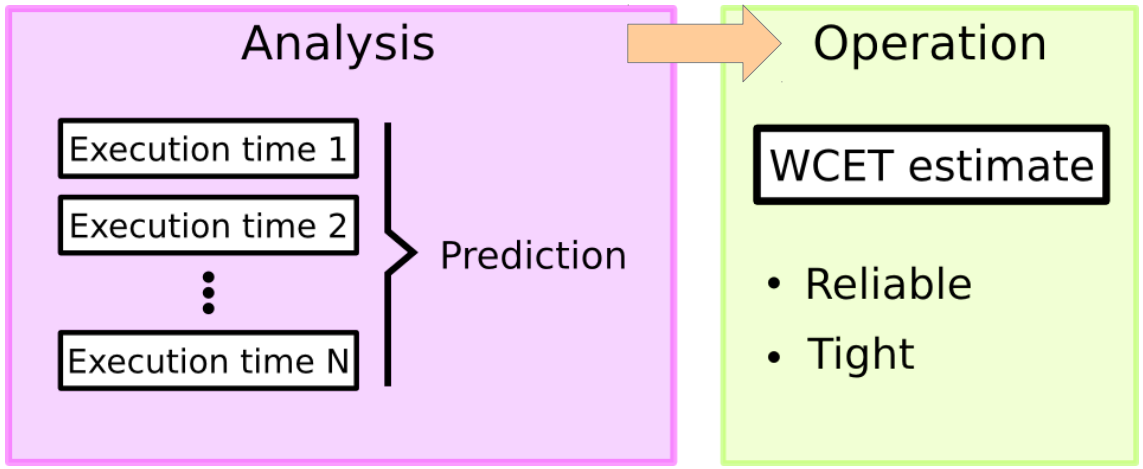
- TASA (Toolchain Agnostic Software rAndomization) + COTS
  - Static variant of software randomization at source-code level
  - Randomizes position in memory of functions, stack frames and global data
  - When loaded into memory the random binary will translate in a random memory mapping, hence, random cache layout



Dealing with cache jitter

Summary

- Cache jitter is captured with observations

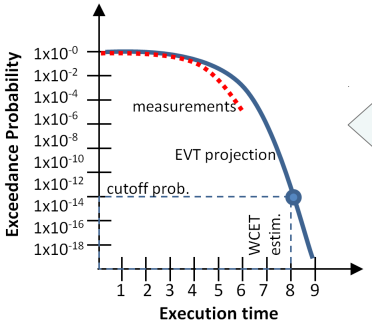


Agenda

- Measurement Based Probabilistic Timing Analysis
  - Statistical Analysis
  - Deterministic and probabilistic upperbounding
- Dealing with Cache Jitter
- **Dealing with Contention Jitter**
  - **Fully time composable (fTC)**
  - **Partially time composable (pTC)**
- Evaluation
- Conclusions

Dealing with contention jitter

- Approach:
  - Enlarge observed execution times with a bound of the maximum contention the task can suffer



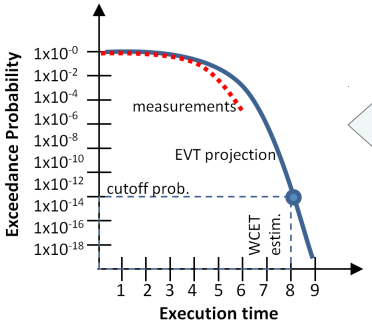
$$eet^i = oet^i + \Delta_{cont}$$

Cache jitter **already factored in**

Multicore jitter **to be factored in**

Dealing with contention jitter

- Approach:
  - Enlarge observed execution times with a bound of the maximum contention the task can suffer



$$eet^i = oet^i + \Delta_{cont}$$

Cache jitter **already factored in**

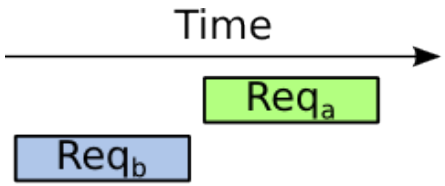
Multicore jitter **to be factored in**

- How?
  - From PMCs collect bus/cache access information of each task when run **in isolation**
  - Combines the PMC information and access latencies to derive the bound
    - No need to run with contender tasks to derive Δ_{cont}

Contention model - fTC

- Derives a WCET estimate that upperbounds slowdown suffered regardless of the load of conteders

- Assumes worst alignment and type of access

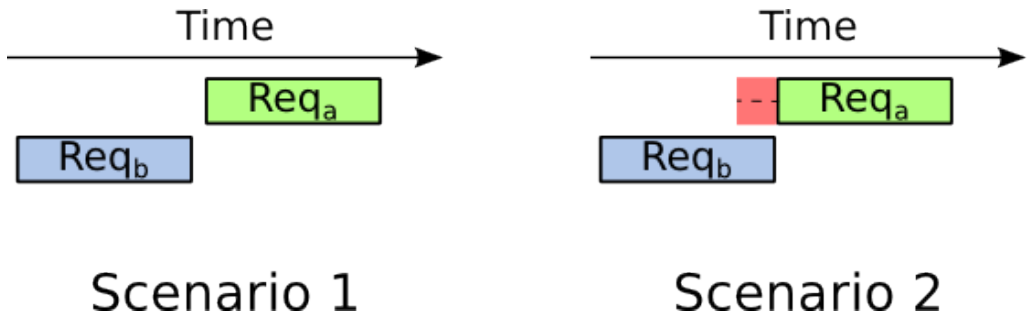


Scenario 1

Contention model - fTC

- Derives a WCET estimate that upperbounds slowdown suffered regardless of the load of conteders

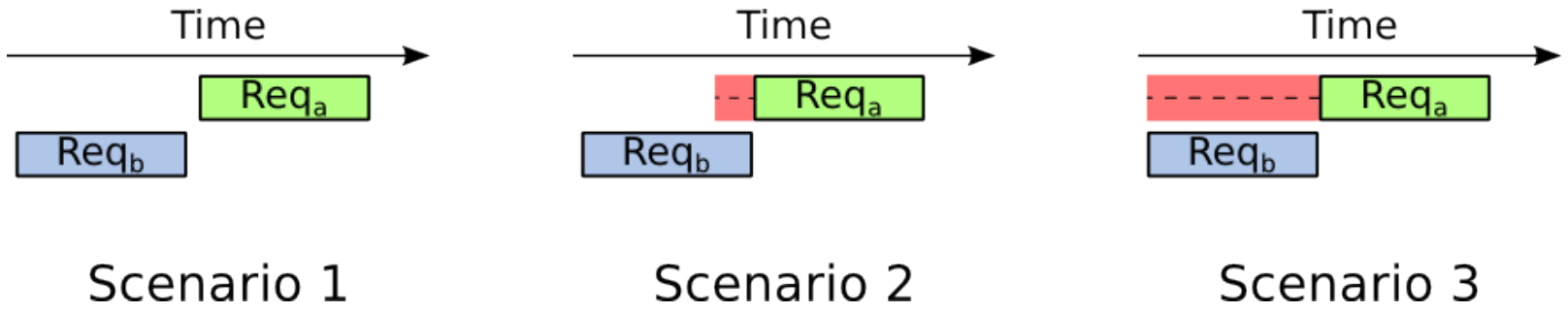
- Assumes worst alignment and type of access



Contention model - fTC

- Derives a WCET estimate that upperbounds slowdown suffered regardless of the load of conteders

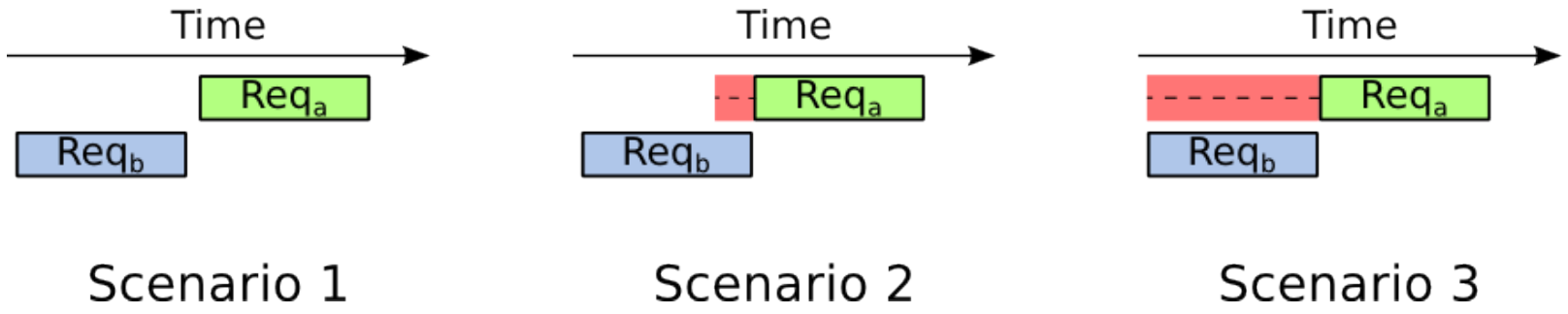
- Assumes worst alignment and type of access



Contention model - fTC

- Derives a WCET estimate that upperbounds slowdown suffered regardless of the load of conteders

- Assumes worst alignment and type of access



Nc = Number of cores (4)

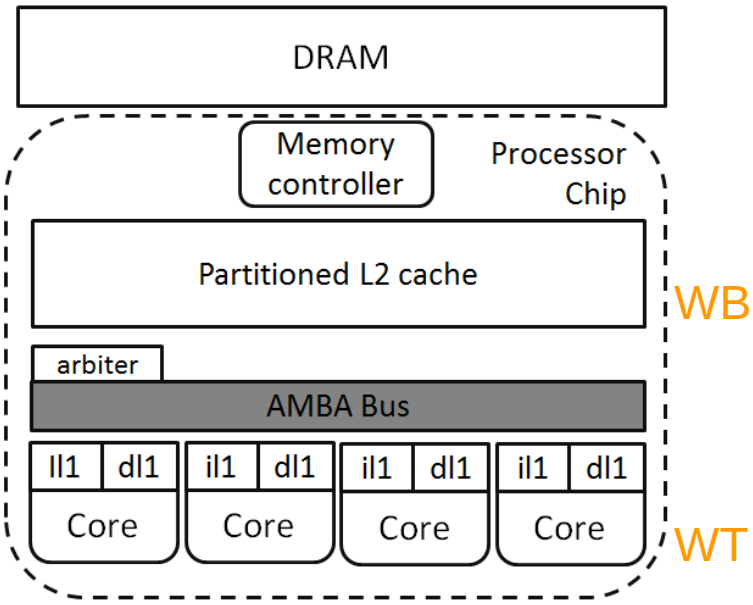
$$eet_a^i = oet_a^i + \Delta_{cont}^{i,fTC} = oet_a^i + \left[n_a^i \times (Nc-1) \times l^{xmd}\right]$$

Contention model - pTC

- Trade time-composability to tighten WCET estimates
  - Incremental integration with small efforts
- pTC tracks contender's information
  - Number of contenders
  - Number of requests of each type

Type mcd		Description
sh	$l^{sh} = 1$	L2 st hit
lh	$l^{lh} = 8$	L2 ld hit in L2
lmc	$l^{lmc} = 28$	L2 ld clean miss
smc	$l^{smc} = 28$	L2 st clean miss
lmd	$l^{lmd} = 31$	L2 ld dirty miss
smd	$l^{smd} = 31$	L2 st dirty miss

L2 cache	hits (n^h)	misses (n^m)	
		dirty (n^{md})	clean (n^{mc})
loads(n^l)	n^{lh}	n^{lmd}	n^{lmc}
stores (n^{st})	n^{sh}	n^{smd}	n^{smc}



Contention model - pTC

- Ideal PMC support scenario (contention impact of τ_b on τ_a)

$$\underbrace{n^{lh} \times l^{lh}} + \underbrace{n^{sh} \times l^{sh}} + \underbrace{\cdots} n^{lmc} \times l^{lmc} + n^{lmd} \times l^{lmd} + n^{smc} \times l^{smc} + n^{smd} \times l^{smd}$$

Contention model - pTC

- Ideal PMC support scenario (contention impact of τ_b on τ_a)

$$\underbrace{n^{lh} \times l^{lh}} + \underbrace{n^{sh} \times l^{sh}} + \underbrace{\dots} + n^{lmc} \times l^{lmc} + n^{lmd} \times l^{lmd} + n^{smc} \times l^{smc} + n^{smd} \times l^{smd}$$

- Reality:

Name	Description
pmc^{icm}	Bus reads caused by *ic* misses
pmc^{dcm}	Bus reads caused by *dc* misses
pmc^{st}	Writes to L2
pmc^{m}	Misses in the L2

L2 cache	hits (n^h)	misses (n^m)	
		dirty (n^{md})	clean (n^{mc})
loads(n^l)	n^{lh}	n^{lmd}	n^{lmc}
stores (n^{st})	n^{sh}	n^{smd}	n^{smc}

$pmc^{icm} + pmc^{dcm}$

pmc^{st}

pmc^{m}

Contention model - pTC

L2 cache	hits (n^h)	misses (n^m)	
		dirty (n^{md})	clean (n^{mc})
loads(n^l)	n^{lh}	n^{lmd}	n^{lmc}
stores (n^{st})	n^{sh}	n^{smd}	n^{smc}

$pmc^{icm} + pmc^{dcm}$ + pmc^{st}

pmc^m

Access latency of each type

\+

$$\hat{n}^{md} = min(pmc^m, pmc^{st})$$

$$\check{n}^{mc} = pmc^m - \hat{n}^{md}$$

Bounding dirty misses

$$\hat{n}^{lh} = min(n^h, n^l)$$

$$\check{n}^{sh} = n^h - \hat{n}^{lh}$$

Bounding load hits

\-

$$n^h = (pmc^{icm} + pmc^{dcm} + pmc^{st}) - pmc^m$$

$$n^l = n^{lh} + n^{lmc} + n^{lmd}$$

Contention model - pTC

- We have
  - Contenders' accesses classified by type
  - TUA accesses
- Pairing with TUA accesses
  - From longest latency, to shortest

$$\hat{c}^{md} = min(n_a^i, \hat{n}_b^{md})$$
$$n_a'^i = max(0, n_a^i - \hat{c}_b^{md})$$

$$\check{c}^{mc} = min(n_a'^i, \check{n}_b^{mc})$$
$$n_a'' = max(0, n_a'^i - \check{c}^{mc})$$

$$\hat{c}^{lh} = min(n_a''^i, \hat{n}_b^{lh})$$
$$n_a''' = max(0, n_a''^i - \hat{c}^{lh})$$

$$\check{c}^{sh} = min(n_a'''^i, \check{n}_b^{sh})$$
$$n_a''''^i = max(0, n_a'''^i - \check{c}^{sh})$$

- Slowdown task τ_b causes on τ_a due to contention

$$\Delta^{i,pTC}_{\tau_b \rightarrow \tau_a} = (\hat{c}^{md} \times l^{md}) + (\check{c}^{mc} \times l^{mc}) + (\hat{c}^{lh} \times l^{lh}) + (\check{c}^{sh} \times l^{sh})$$

Contention model - pTC

- Overall contention τ_a suffers from its three contenders τ_b, τ_c, τ_d

$$\Delta_{cont}^{i,pTC} = \Delta_{\tau_b \rightarrow \tau_a}^{i,pTC} + \Delta_{\tau_c \rightarrow \tau_a}^{i,pTC} + \Delta_{\tau_d \rightarrow \tau_a}^{i,pTC}$$

Contention model - pTC

- Overall contention τ_a suffers from its three contenders τ_b, τ_c, τ_d

$$\Delta_{cont}^{i,pTC} = \Delta_{\tau_b \rightarrow \tau_a}^{i,pTC} + \Delta_{\tau_c \rightarrow \tau_a}^{i,pTC} + \Delta_{\tau_d \rightarrow \tau_a}^{i,pTC}$$

$$eet^i = oet^i + \Delta_{cont}$$

Cache jitter

Multicore jitter

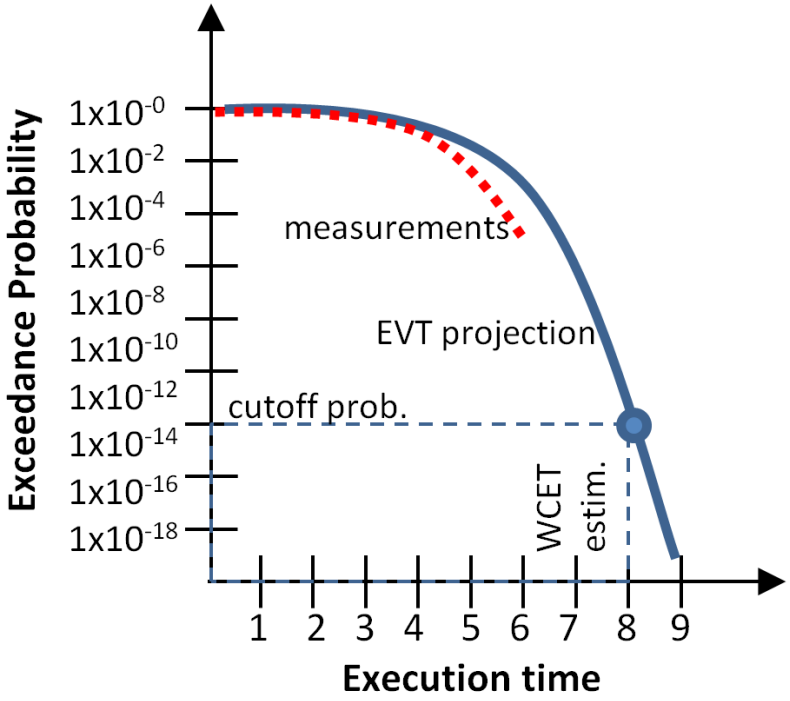
Agenda

- Measurement Based Probabilistic Timing Analysis
  - Statistical Analysis
  - Deterministic and probabilistic upperbounding
- Dealing with Cache Jitter
- Dealing with Contention Jitter
  - Fully time composable (fTC)
  - Partially time composable (pTC)
- **Evaluation**
- Conclusions

Evaluation

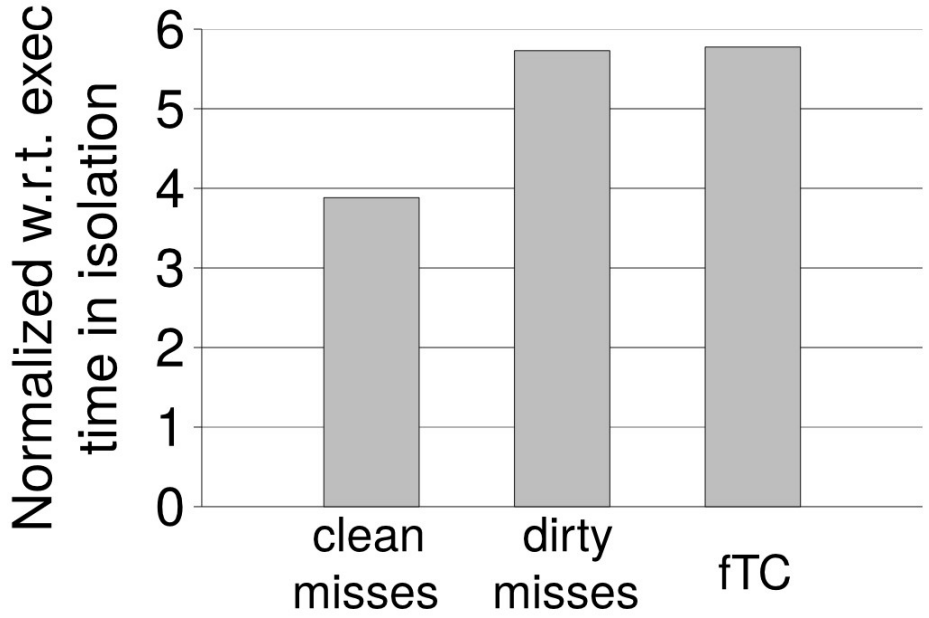
- FPGA implementation
  - 4 LEON3 cores
    - 16KB 4-way L1 (I&D) WT, no WA
    - 128KB 4-way L2 (32KB partitioned) WB

- MBPTA setup
  - 10^{-12} probability threshold
  - 3000 runs
  - Statistical tests, IID



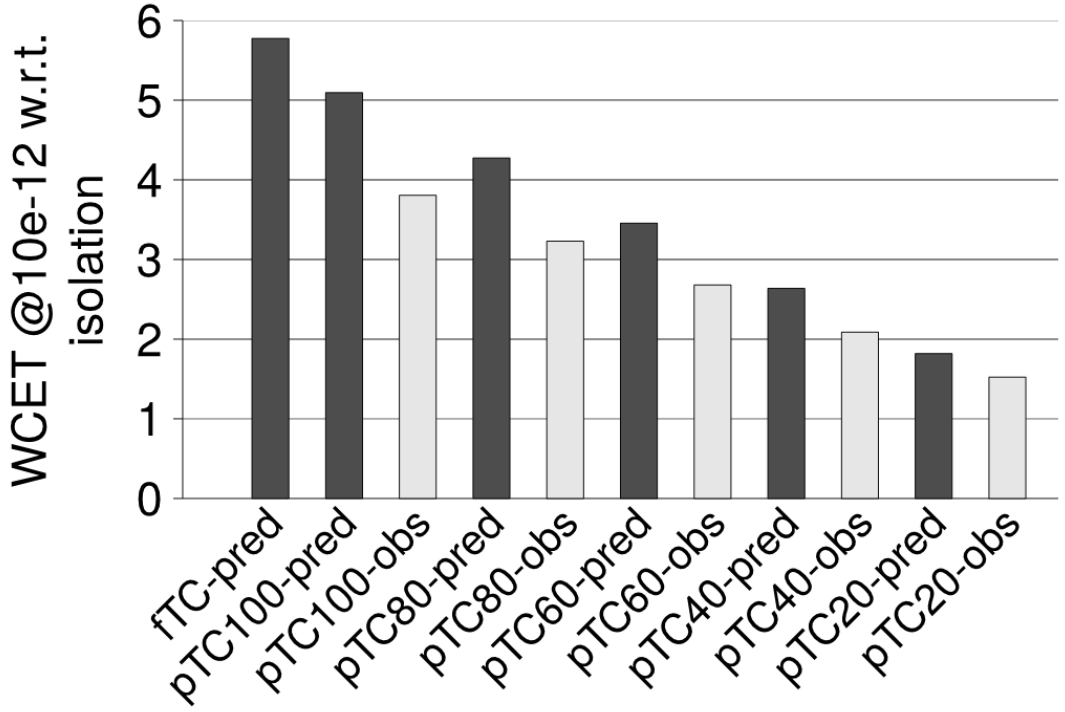
Evaluation

- Assessing fTC accuracy
- TUA:
  - Synthetic application performing uniform accesses to the shared bus (30% of its ET)
- Contender tasks:
  - 3 contenders performing dirty/clean misses



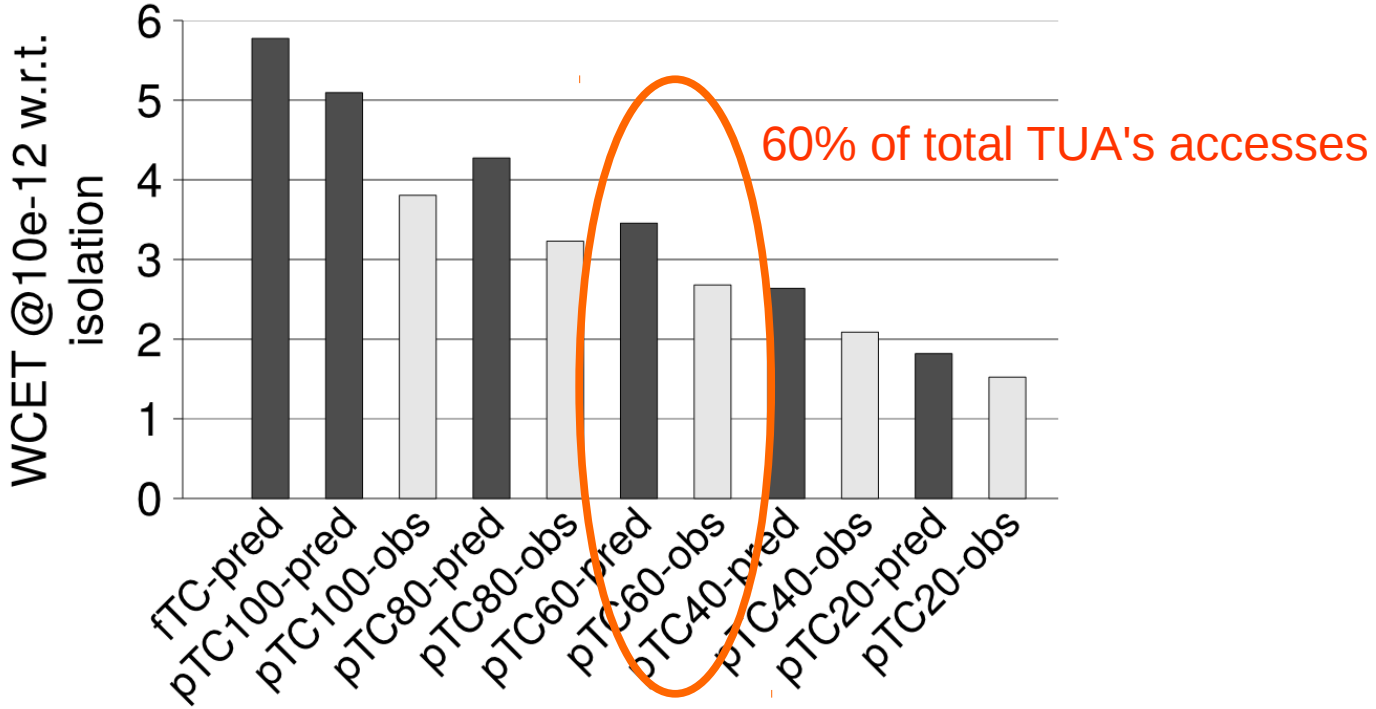
Evaluation

- Assessing pTC accuracy
- TUA:
  - Synthetic application performing uniform accesses to the shared bus (30% of its ET) as TUA
- Contender tasks:
  - 3 contenders performing a percentage of clean misses (w.r.t. TUA's accesses)



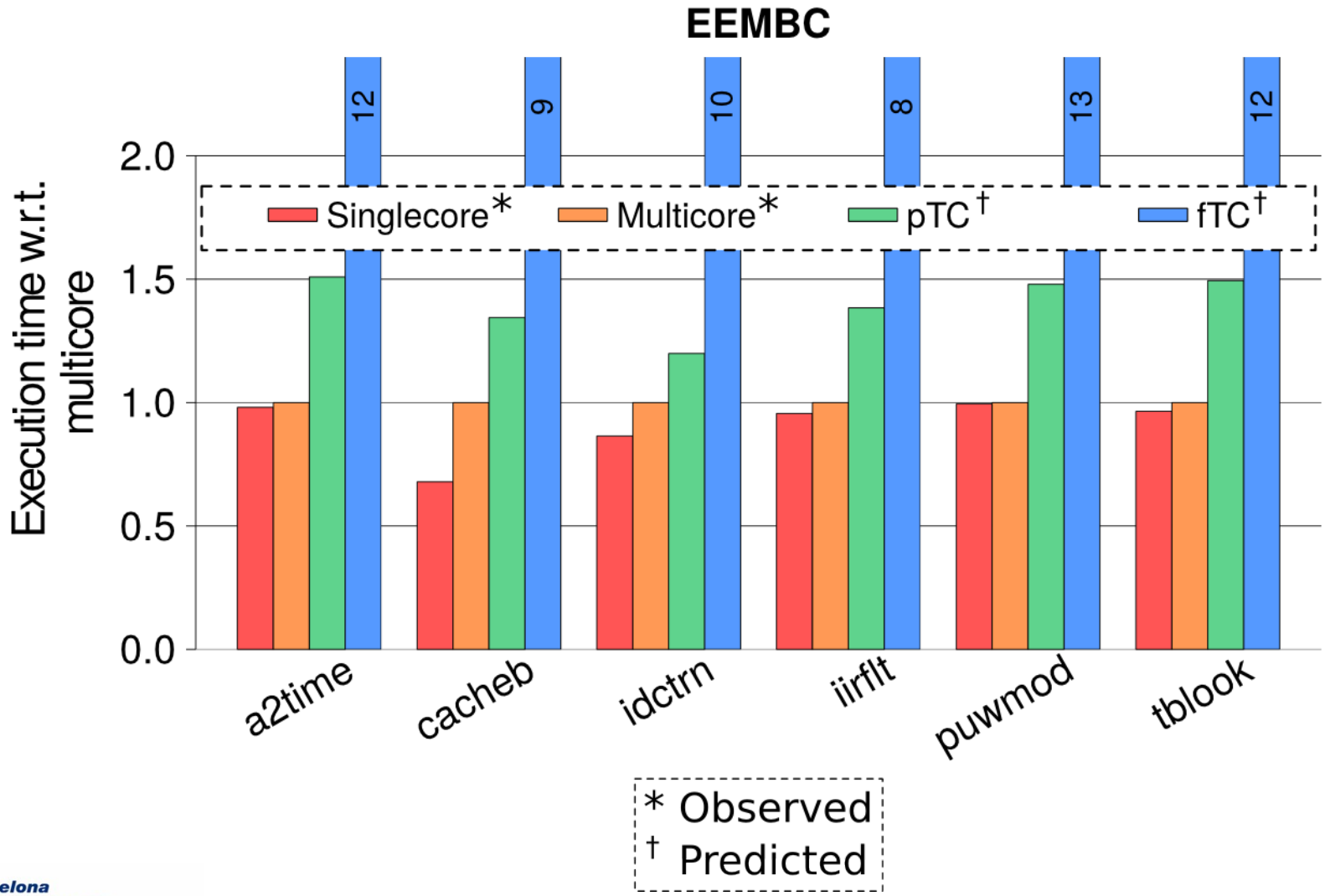
Evaluation

- Assessing pTC accuracy
- TUA:
  - Synthetic application performing uniform accesses to the shared bus (30% of its ET) as TUA
- Contender tasks:
  - 3 contenders performing a percentage of clean misses (w.r.t. TUA's accesses)



Evaluation

- EEMBC as TUA
  - Against 3 copies of themselves



Agenda

- Measurement Based Probabilistic Timing Analysis
  - Statistical Analysis
  - Deterministic and probabilistic upperbounding
- Dealing with Cache Jitter
- Dealing with Contention Jitter
  - Fully time composable (fTC)
  - Partially time composable (pTC)
- Evaluation
- **Conclusions**

Conclusions

- We have proposed a technique for COTS multilevel-cache multicores
  - It relies on MBPTA to derive pWCET estimates

- Our results show how MC2 effectively captures contention and safely upperbounds observed values

- Future work
  - Extend the technique to other platforms
  - Promote specific PMCs to reduce overhead

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MC2: Multicore and Cache Analysis via Deterministic and Probabilistic Jitter Bounding

Enrique Díaz[1,2], Mikel Fernández[1], Leonidas Kosmidis[1], Enrico Mezzetti[1], Carles Hernandez[1], Jaume Abella[1], and Francisco J. Cazorla[1,3]

[1]Barcelona Supercomputing Center (BSC), Spain
[2]Universitat Politècnica de Catalunya, Spain
[3]IIIA-CSIC, Spain

22nd International Conference on Reliable Software Technologies
Ada-Europe 2017
Vienna, 15 June 2017