PROXIMA

Improving measurement-based timing analysis through randomisation and probabilistic Analysis

Francisco J. Cazorla (francisco.cazorla@bsc.es)

Director of the CAOS research group at BSC and
researcher at the Spanish National Research Council (IIIA-CSIC)

Coordinator of the FP7 IP Project PROXIMA

EMC^2 Workshop. September 28th 2016.
Paris, France
www.proxima-project.eu

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

Who we are

Probabilistic real-time control of mixed-criticality multicore and manycore systems (PROXIMA)

Coordinator: Francisco J. Cazorla (BSC)

Oct 2013 – Sep 2016

Integrated Project (IP), total budget : 6,793,991 Euro

Barcelona Supercomputing Center
Rapita Systems Limited
Sysgo S.A.S
Universita Degli Studi Di Padova
INRIA
Cobham Gaisler
Airbus Operations SAS
University of York
Airbus Defence&Space
IKERLAN S.COOP
Infineon Technologies UK Ltd

PROXIMA Consortium

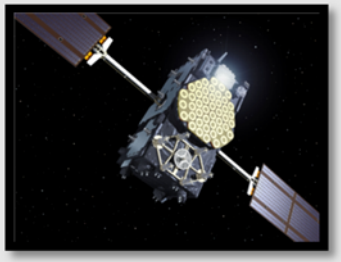
Critical Real Time Embedded Systems (CRTES)

Does the Software work?

Functional correctness	Timing correctness
Software performs its task	Software fits its assigned time budget

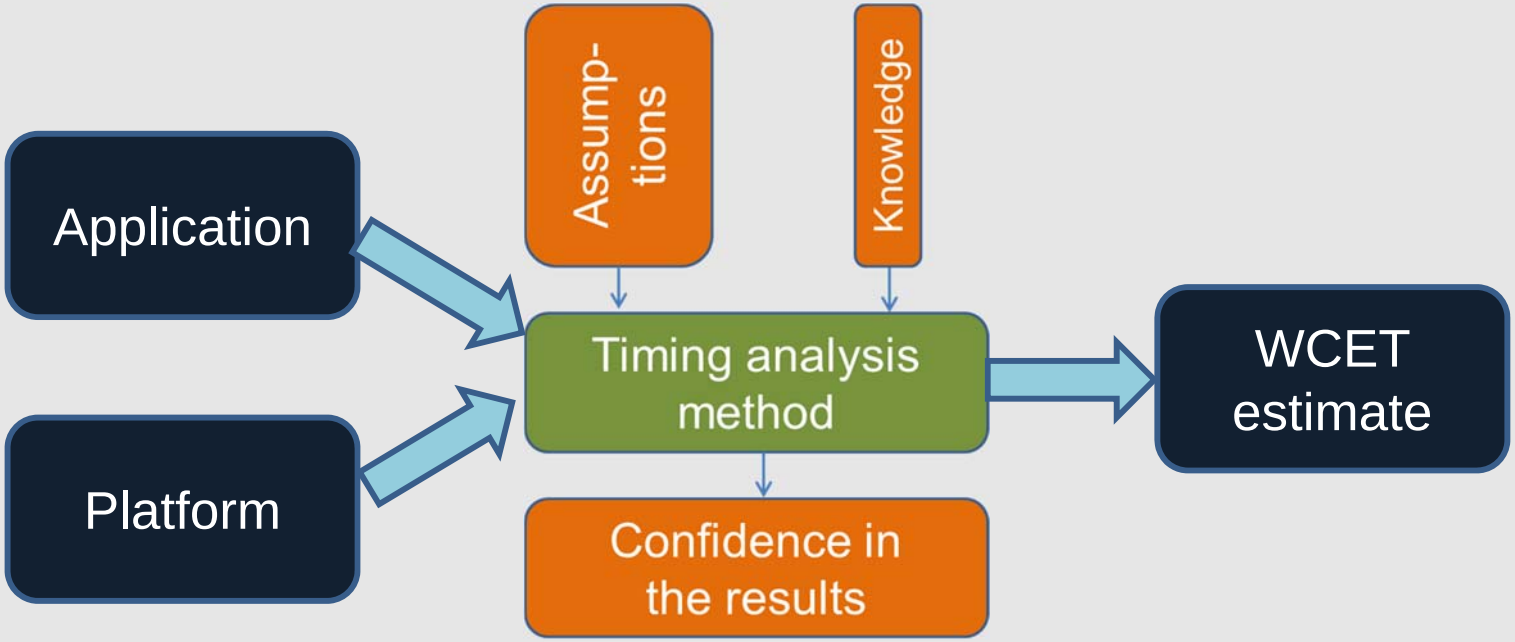
Provide evidence about the timing (and functional) correctnes of the system against safety standards






Timing verification

- Obtaining tight WCET estimates is complex
  - Several methods derived in last decades
  - Rely on assumptions and inputs on the HW/SW
  - For each domain/system preserving those assumptions is hard
  - No method is fully trustworthy on all accounts



Challenges

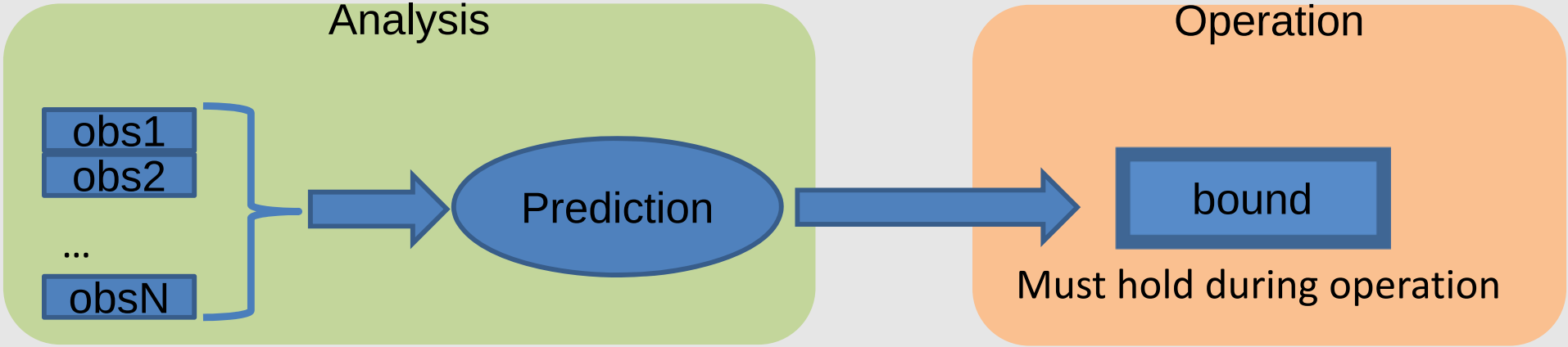
- **For end users**
  - Industrial users have to derive WCET estimates
    - With the domain-specific degree of trustworthiness
    - Strict cost and effort constraints
    - Keep a high benefit/cost ratio
- **For PROXIMA**
  - Varying end-user requirements on timing verification
    - Different per platform, domain and criticality level
      - Overheads that timing analysis “is allowed to create” vary (cost, instrum.)
      - Evidence/confidence required vary as well
      - Relates to Timing Bands
  - **Building an one-size-fits-all solution is not realistic for us**
  - PROXIMA provides several solutions w/ different requirements
    - Instrumentation at the Unit-of-Analysis (e.g. function)
    - Instrumentation at basic-block level

Measurement-Based Timing Analysis

- Main focus of PROXIMA
- Measurements is the dominant timing analysis approach across different market segments
  - Automotive
  - Railway
  - Space
  - …
- “Measurements are unsafe”?
  - Measurements are used for highest-criticality software (e.g. DAL-A in avionics)

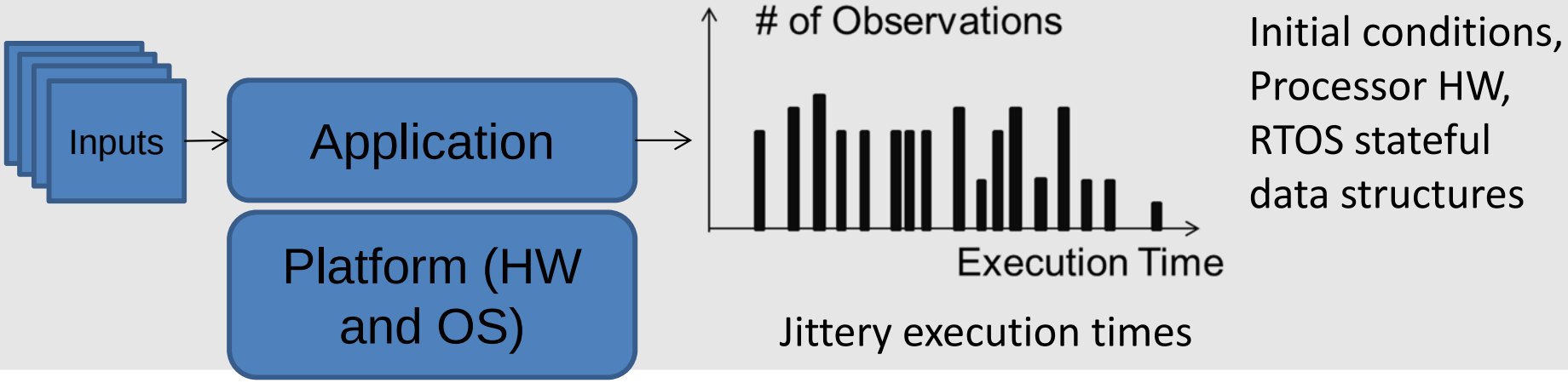
Measurement-Based Timing Analysis

- Analysis phase
  - Collect measurements to derive a WCET estimate that holds valid during system operation
- Operation phase
  - Actual use of the system (under assumption it stays within its performance profile)



The “fallacy” of Deterministic Systems

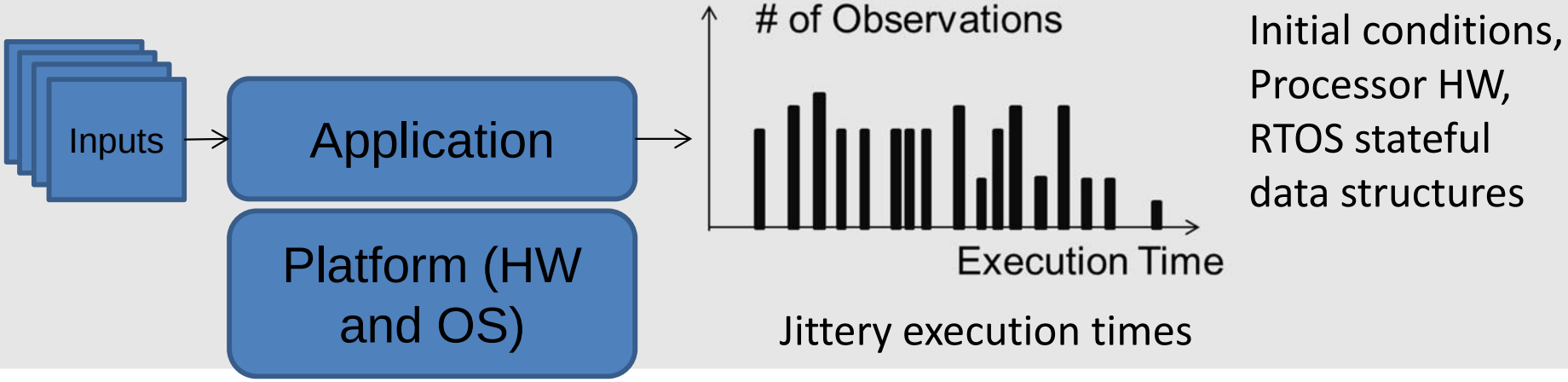
- Deterministic systems
  - Everything is repeatable
  - Do the same thing twice and it’s exactly the same
- But this is not really true for all aspects of computation
  - Run the same thing multiple times, get **variations** in ordering, timing, interactions between components
- HW/SW complexity grows breaking deterministic models



SETV aka SJ

- Sources of
  - Execution Time Variability (SETV) or
  - Jitter (SJ)

  Any platform element in the platform that cause execution time of a program to vary
- The value that each SETV gets for a given experiment defines the **execution conditions** for that experiment
  - Systems are complex to understand, **the user can only follow what happens at a high level**

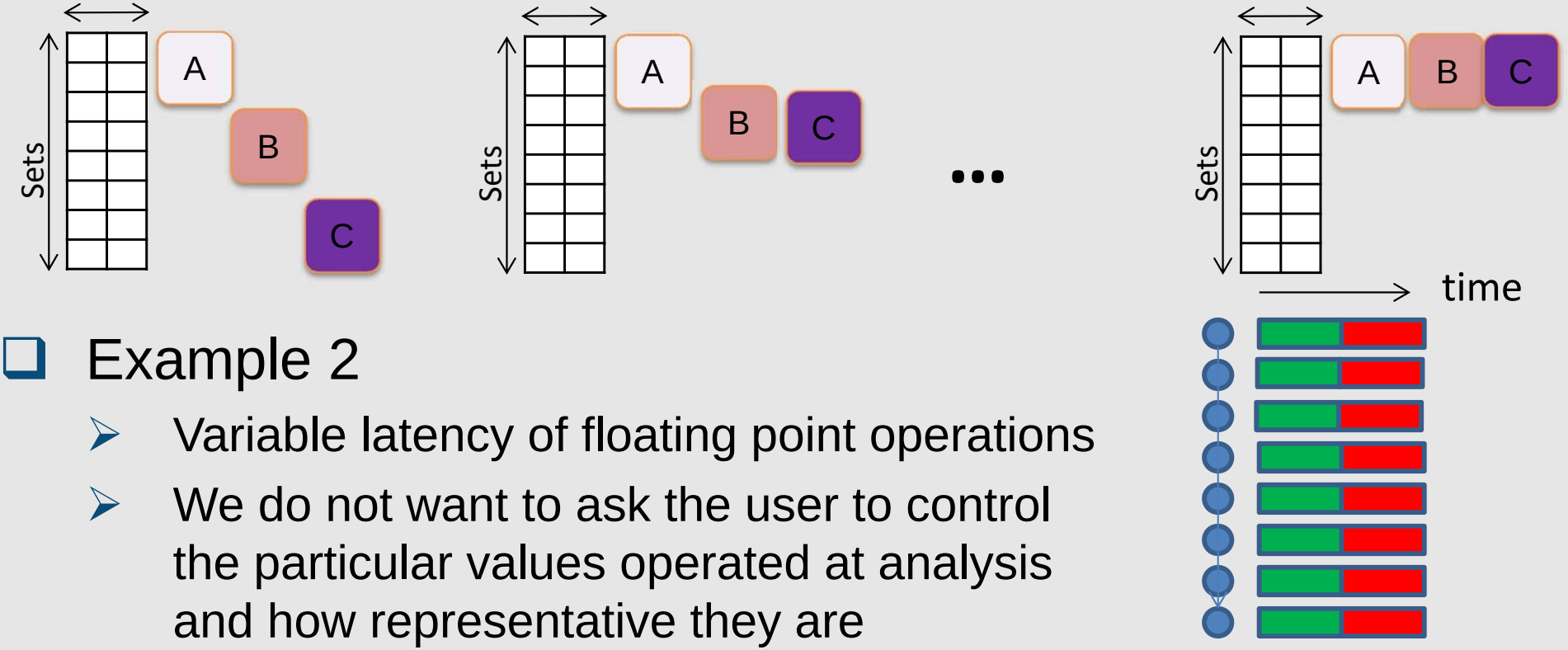


High-level and low-level SETV

- **High-level SETV**: the user has some control on them
  - Input vectors impact on execution paths
    - Metrics to measure coverage (SC, DC, MCDC)
    - Tools to determine coverage (e.g. RapiCover from Rapita Systems)
    - Which path was traversed, to make claims on path coverage
- The use of complex high-performance hardware creates other **low-level SETV**
  - The user lacks means to measure the coverage of low-level SETV
    - Often insufficient support from the HW

Examples of low-level SETV

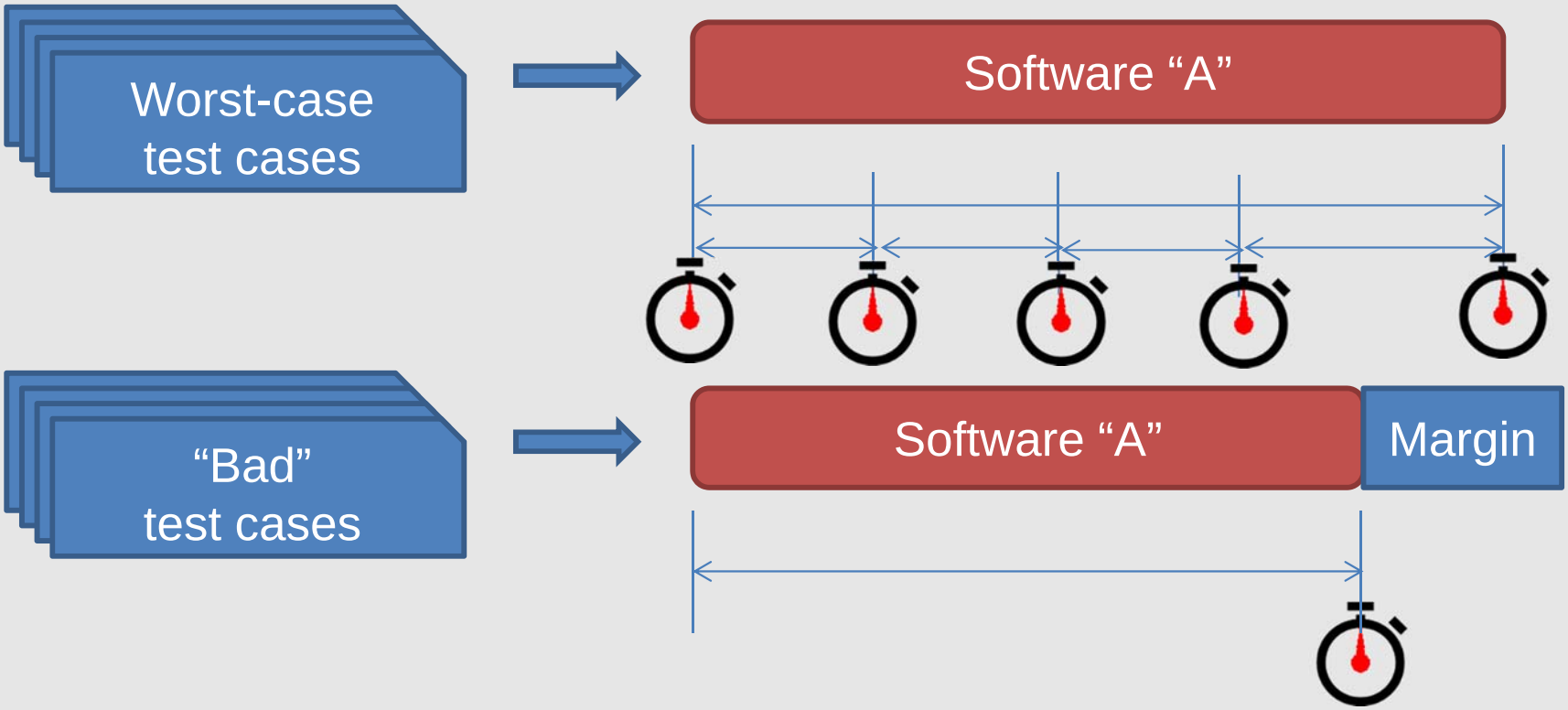
- Example 1
  - The mapping program objects (functions) →
    - How software objects are assigned to memory →
    - How they are placed in cache → conflicts suffered →
    - Execution-time effects



- Example 2
  - Variable latency of floating point operations
  - We do not want to ask the user to control the particular values operated at analysis and how representative they are

State of practice of deterministic systems

- Test cases: from worst to good test cases



- Level of conservative margin is unclear (20%, 50%, 200%)
  - No scientific basis, only expert judgement
  - It works in practice when user has “sufficient” HW/SW knowledge

State of practice of deterministic systems

- Confidence: ensure that the worst-case conditions have been exercised or closely approximated
  - Effort involved
  - Diluted in the overall testing campaign
- Desired properties:
  - Accurate and cost-effective timing analysis
  - Representative testing
  - Constraints:
    - The user can ensure that test inputs and test conditions exercise each component adequately
    - All important SoJ have been observed wwithout adding conditions that are infeasible in practice

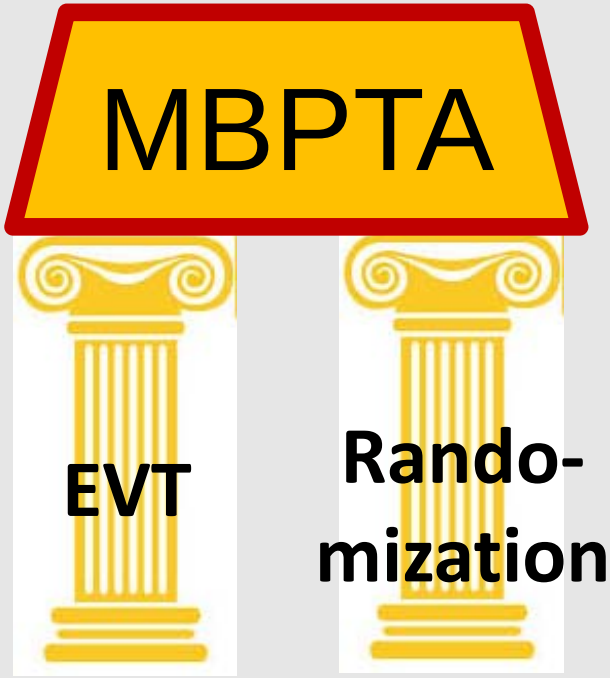
Measurement-Based Timing Analysis

- Goal
  - Based on system **analysis-time** measurements
  - Derive WCET estimates that hold at **system operation**
- How can the user (without dealing with system internals)
  - Be sure that he captures in the measurements taken at analysis time those events impacting execution time?
  - How to provide convincing evidence?
- Problems
  - User to control the execution conditions exercising bad scenarios
  - In systems with several jittery factors the user has to architect experiments to make events to happen on the same run
  - The user has no means to determine the number of runs to do

PROXIMA measurement-based approach

- PROXIMA MBPTA focuses on
  - Change platform behaviour so that
    - low-level SETV are “handled” by the platform w/o user intervention

      or
    - user has means to control SETV in a cheap/fast way
  - Increasing confidence on derived bounds
- Approach
  1. Probabilistic Timing Analysis (Extreme Value Theory)
  2. Time Randomization
     - Functional behavior stays unchanged
  - **Both are required**

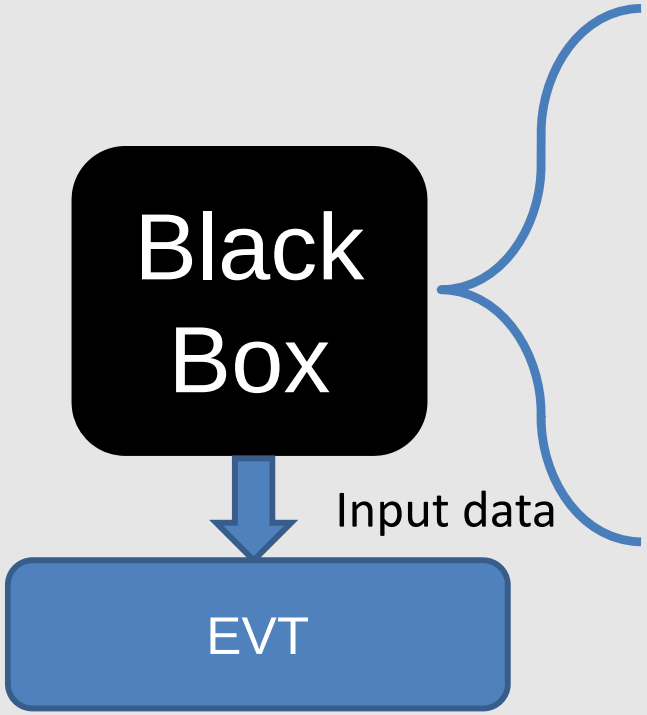


Statistical analysis (EVT)

- Building block of MBPTA, **but it is <u>not</u> MBPTA**
- Used to consider probabilities associated with extreme (and thus rare) events
  - Models the behavior of maxima/minima in the tail of a distribution
- Successfully applied in fields such as hydrology/insurance
- We are interested in EVT to predict – under precise hypotheses – extreme (worst-case) execution time of a software program executing on a processor

Extreme Value Theory /3

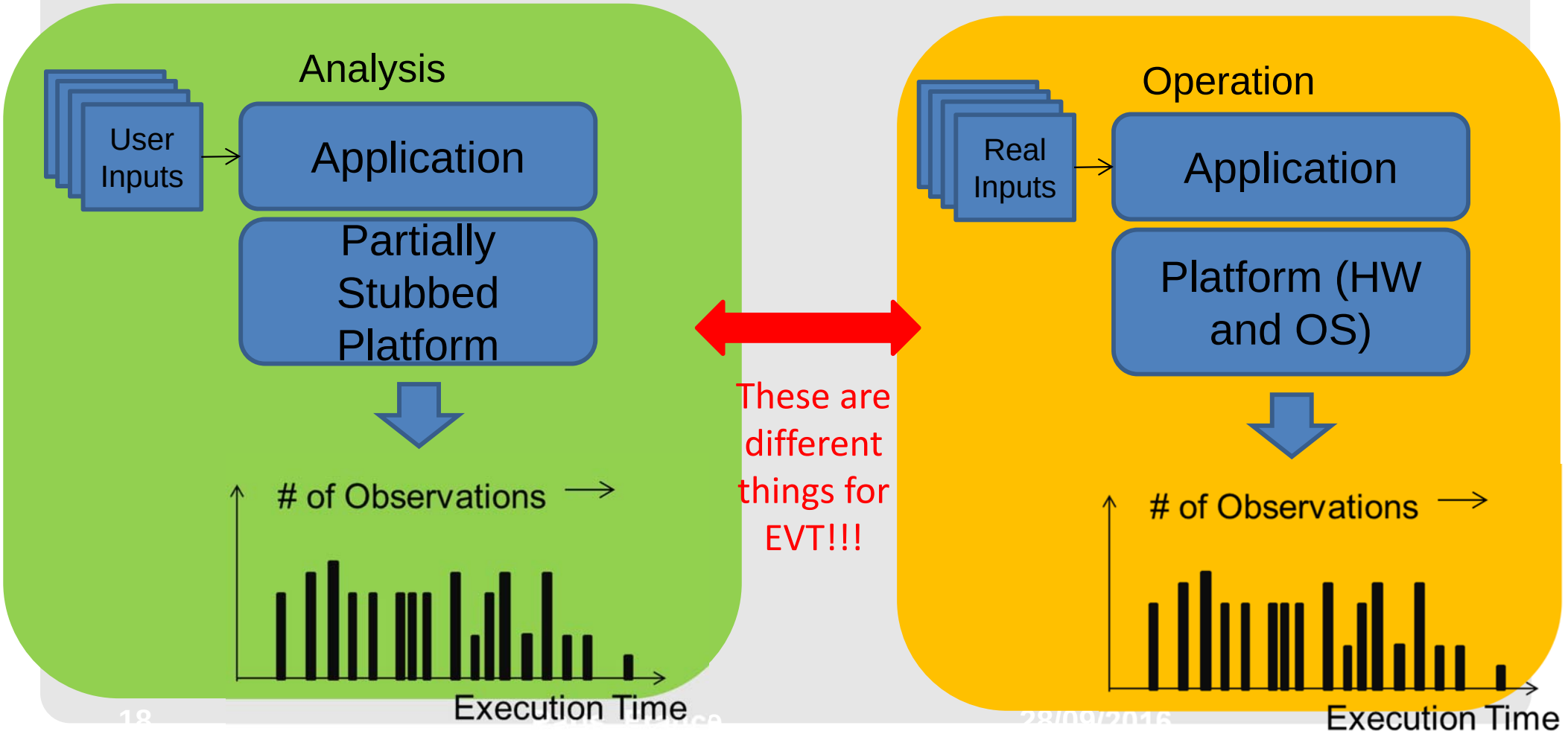
- Considers the system as a black box
- Derives the combined probability of appearance of those events observed (captured)



- Minute every time you check your watch
- Number of times a day my cat makes something crazy
- Level of the sea
- …
- Execution time observations coming from a computer system

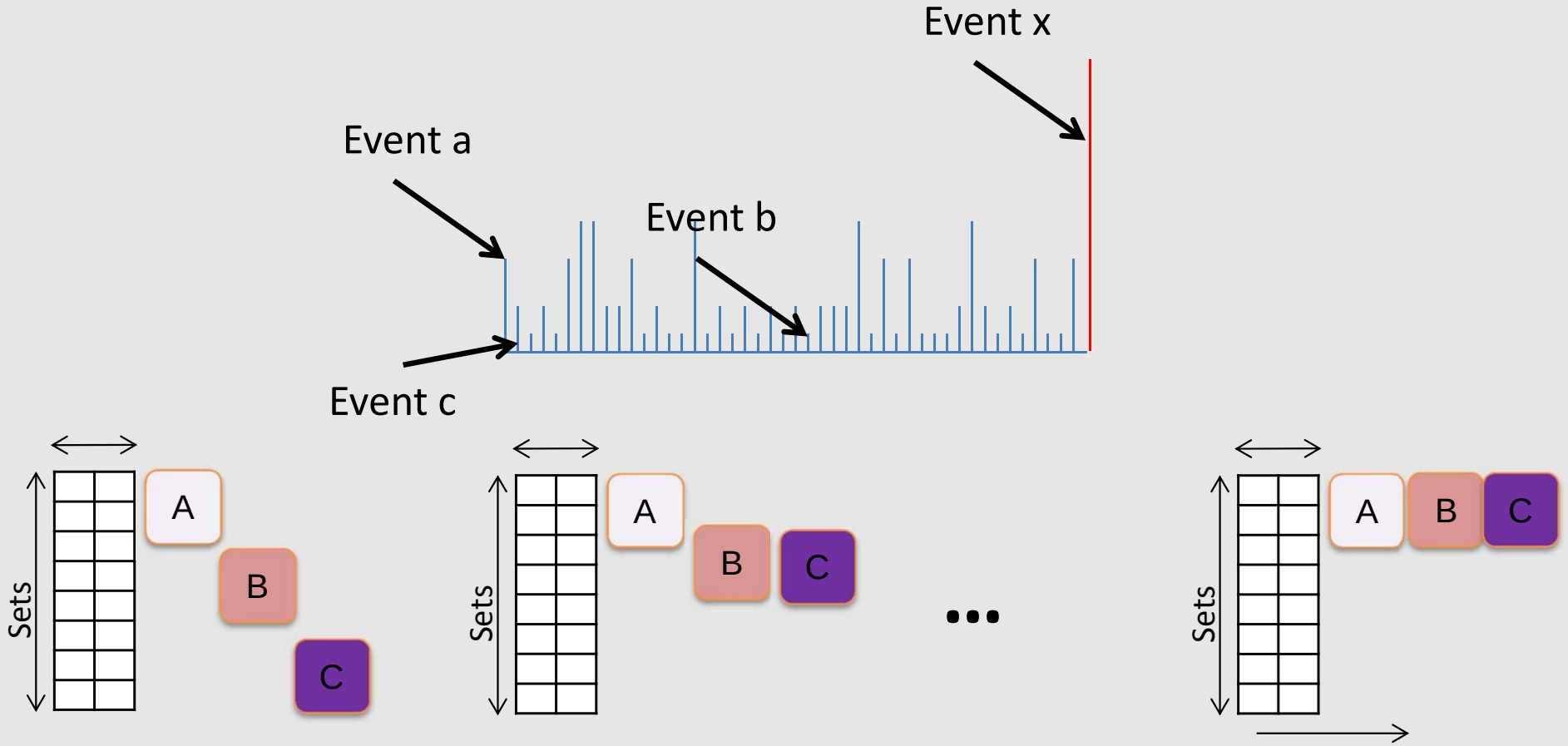
Extreme Value Theory /3

- Derives the <u>combined</u> probability of appearance of those events <u>observed</u> (captured) → "**<u>Observe the same thing</u>**"



Extreme Value Theory /4

- Cannot predict those <u>events that are not observed</u> and whose impact is bigger than that of those observed



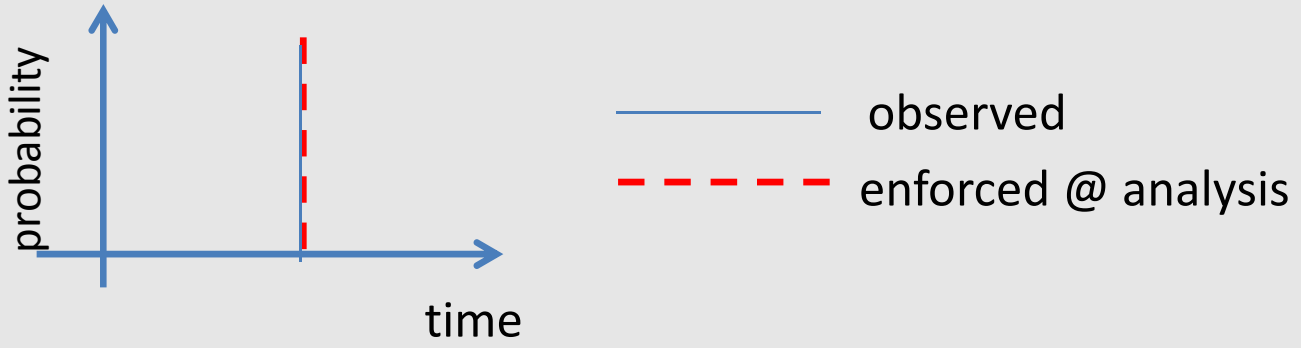
- To solve our problem EVT must be fed with *meaningful* (**representative**) observations

Gaining representativeness /1

MBPTA-compliant platforms

Jitterless resources

- Fixed latency, the same outcome at every occurrence of that event
- E.g., integer adder



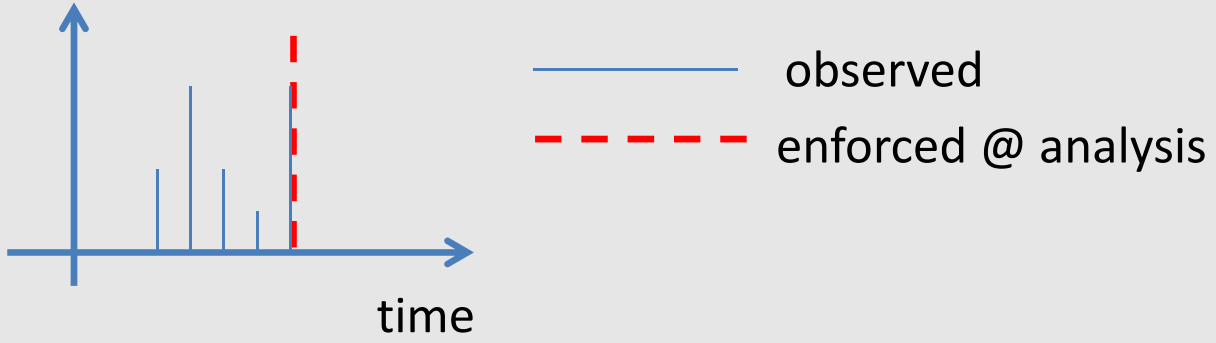
- Analysis – Operation representativeness issue
  - "What you see (at analysis) is what you get (at operation)"

Gaining representativeness /2

MBPTA-compliant platforms

Deterministically upper-bounded resources

- FP multiplication whose latency is 1 or 5 cycles depending on the values operated
- Enforce the FPU to take 5 cycles at analysis time
- During deployment it can take any latency in [1…5]

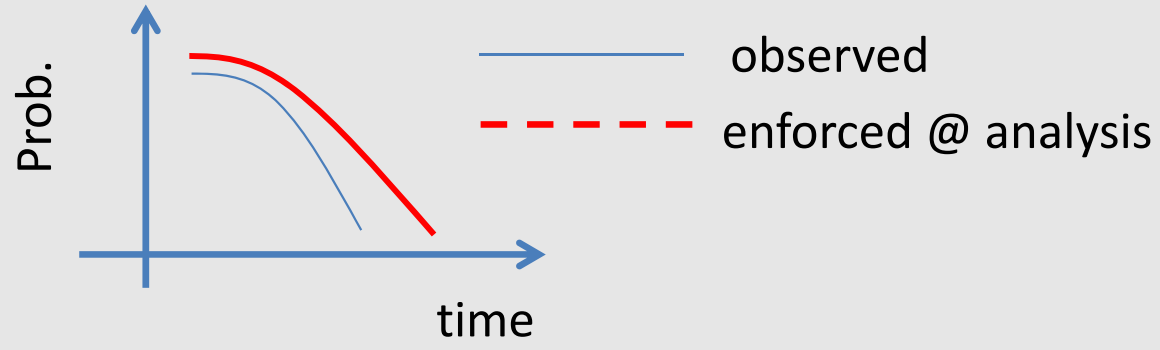


- Analysis – Operation representativeness issue
  - What you see at analysis is worse than what you have at operation

Gaining representativeness /3

MBPTA-compliant platforms

- **Timing randomization**
  - Introduced for hard-to-predict high-jitter resources (e.g. caches)
  - Assuming/enforcing worst latency would be to pessimistic
  - **At hardware level or at software level**
- **Probabilistic upper-bounded resources** (jitter)
  - **Same** probability distribution
    - E.g., cache access with the same hit/miss probabilities
  - **Upper-bounded** distribution
  - E.g., Randomized caches

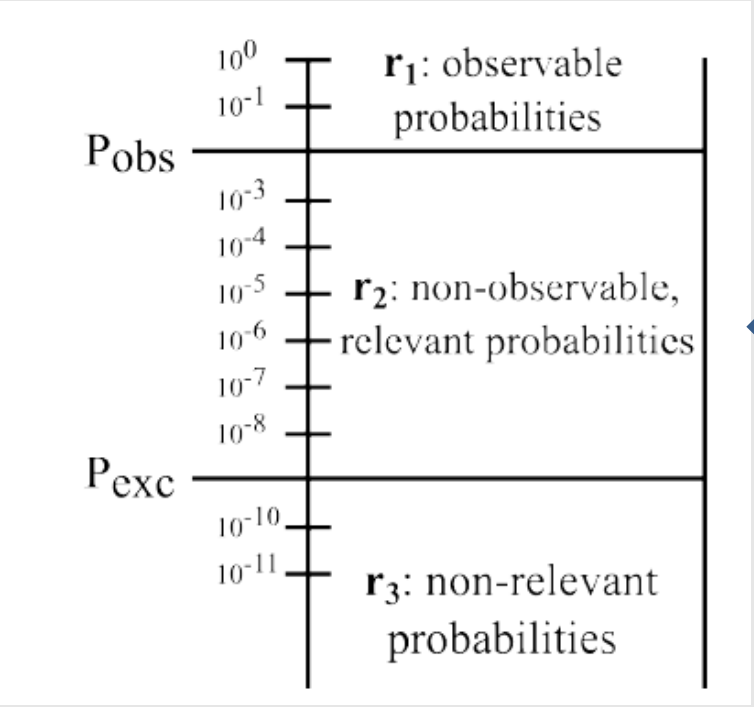


Gaining representativeness /4

MBPTA-compliant platforms

❑ Probabilistically upper-bounded

- ➢ **Number of runs to perform to ensure 'relevant' events are captured in at least one run**
  - Jitterless → 1
  - Deterministic upper-bounded → 1
  - Randomized resources → can be determined



$$P_{obs} = 1 - (1 - P_{eoi})^{R}$$

Probability of an event

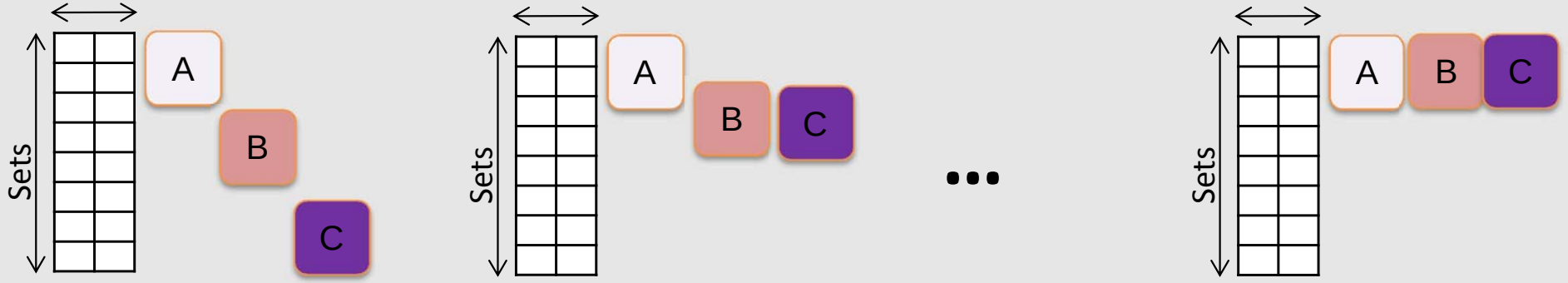
Analysis

Operation

Representativeness?

Example: the cache

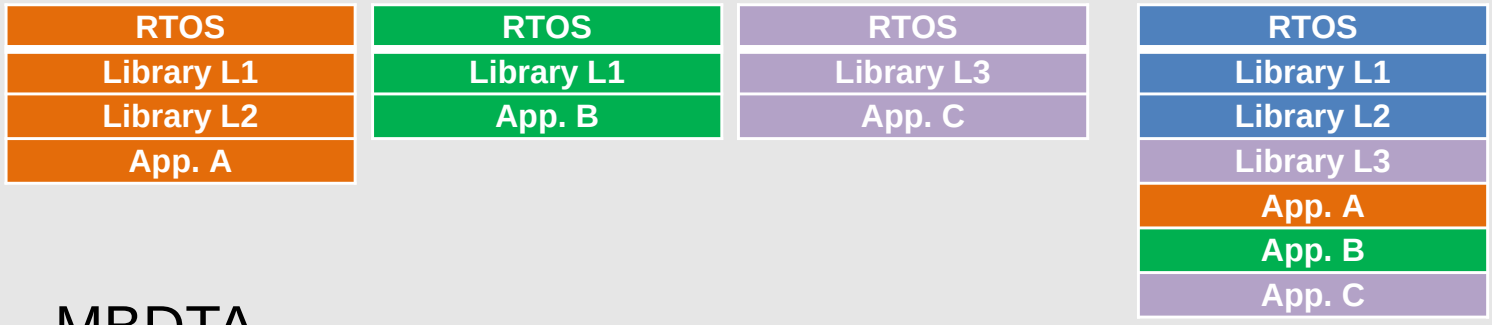
- Memory mapping → cache layouts → execution time



- 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

Facilitate incremental integration

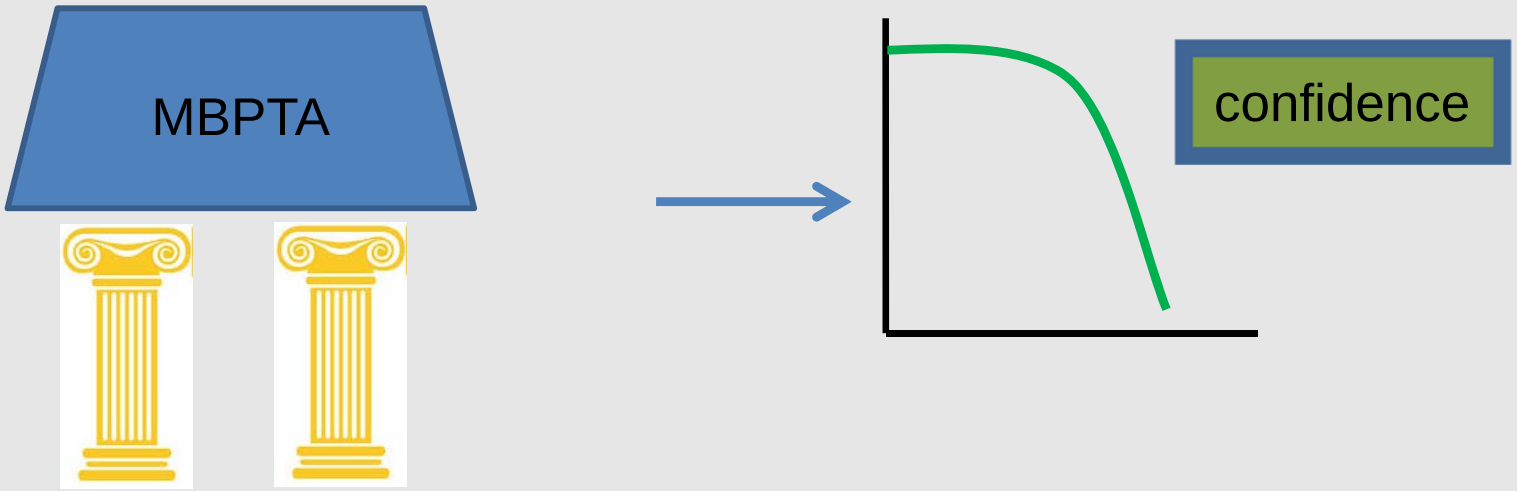
- Incremental HW/SW development and qualification steps
  - Help mastering complexity
  - Trustworthy information on SW timing behavior better obtained as soon as possible to reduce ‘fixes’ costs



  - MBDTA
    - Alignment of objects change for App. A in different integration steps
      - Leading to disruptive changes in the cache behavior
    - Analysis results obtained in isolation do not hold any more
    - After corrections applied to App. A → What impact on App. B and C?
  - MBPTA added value
    - Analysis results are robust against changes in the cache layout as long as a representative number of layouts have been observed

Summary

- MBPTA: Randomization + EVT



EVT

Jitterless resources → 'what you see is what you get'
Randomized resources → probabilistic upper bounding
Worst-latency resources → Deterministic upper bounded resources

MBPTA Basic Application Procedure

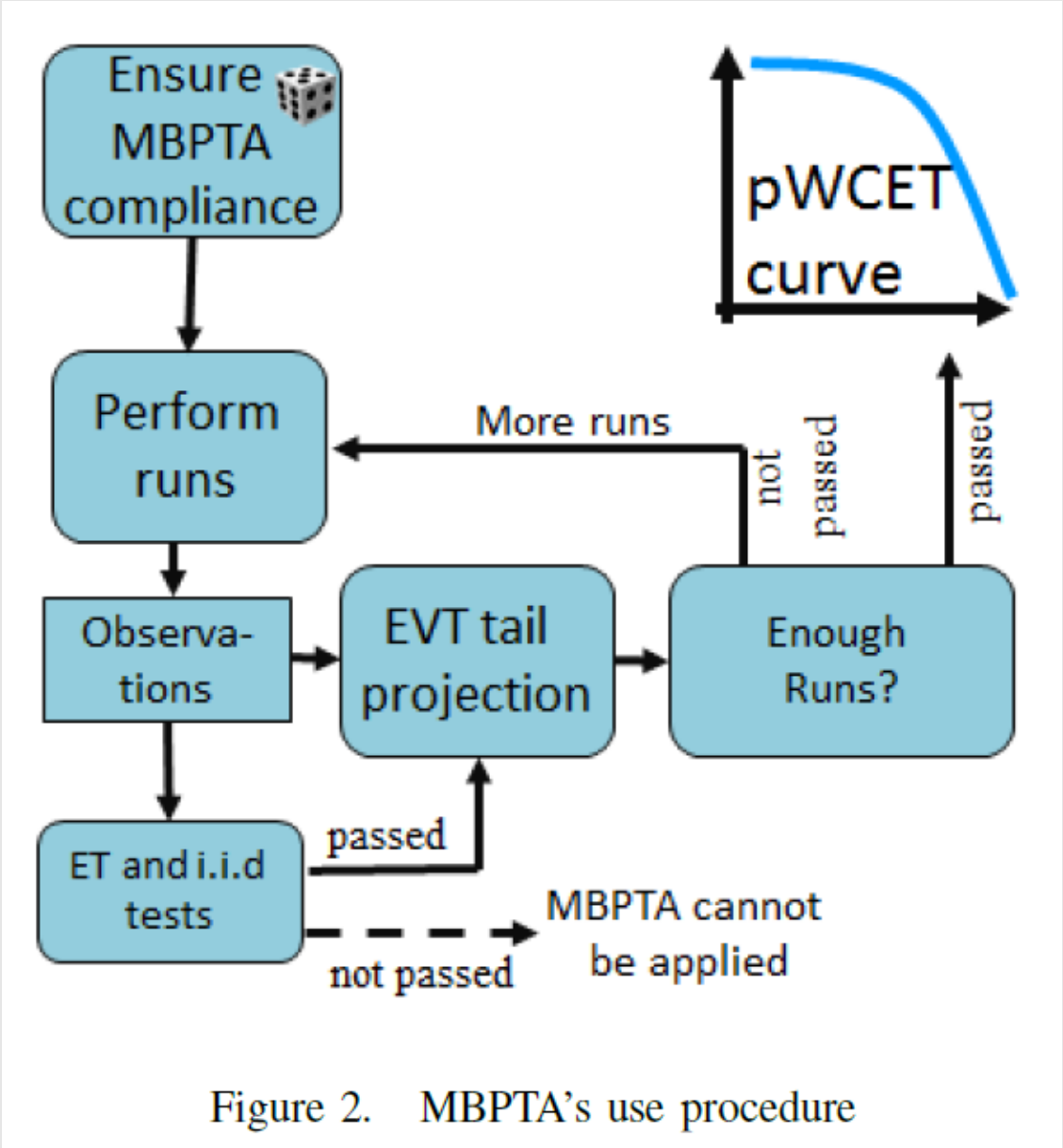


Figure 2. MBPTA's use procedure

EVT projection examples

- Note log scale on the left

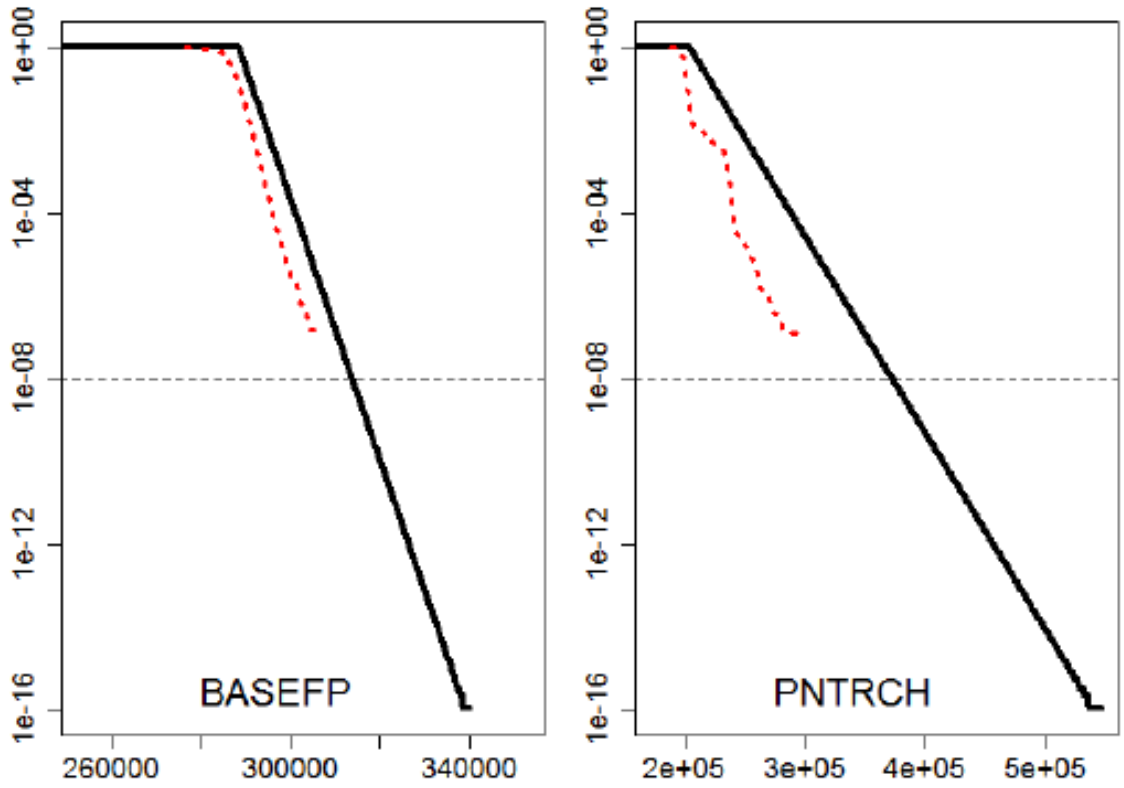
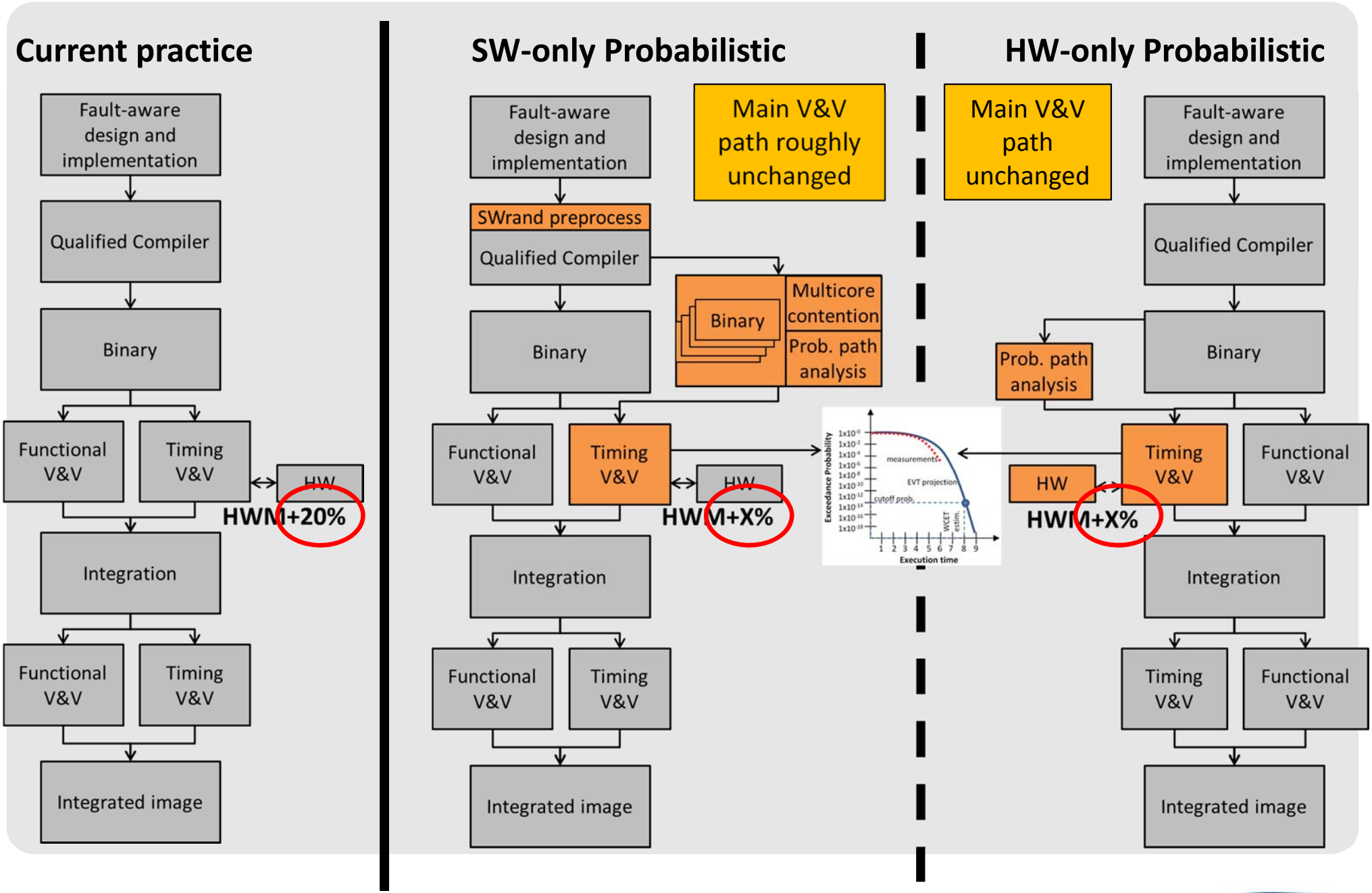
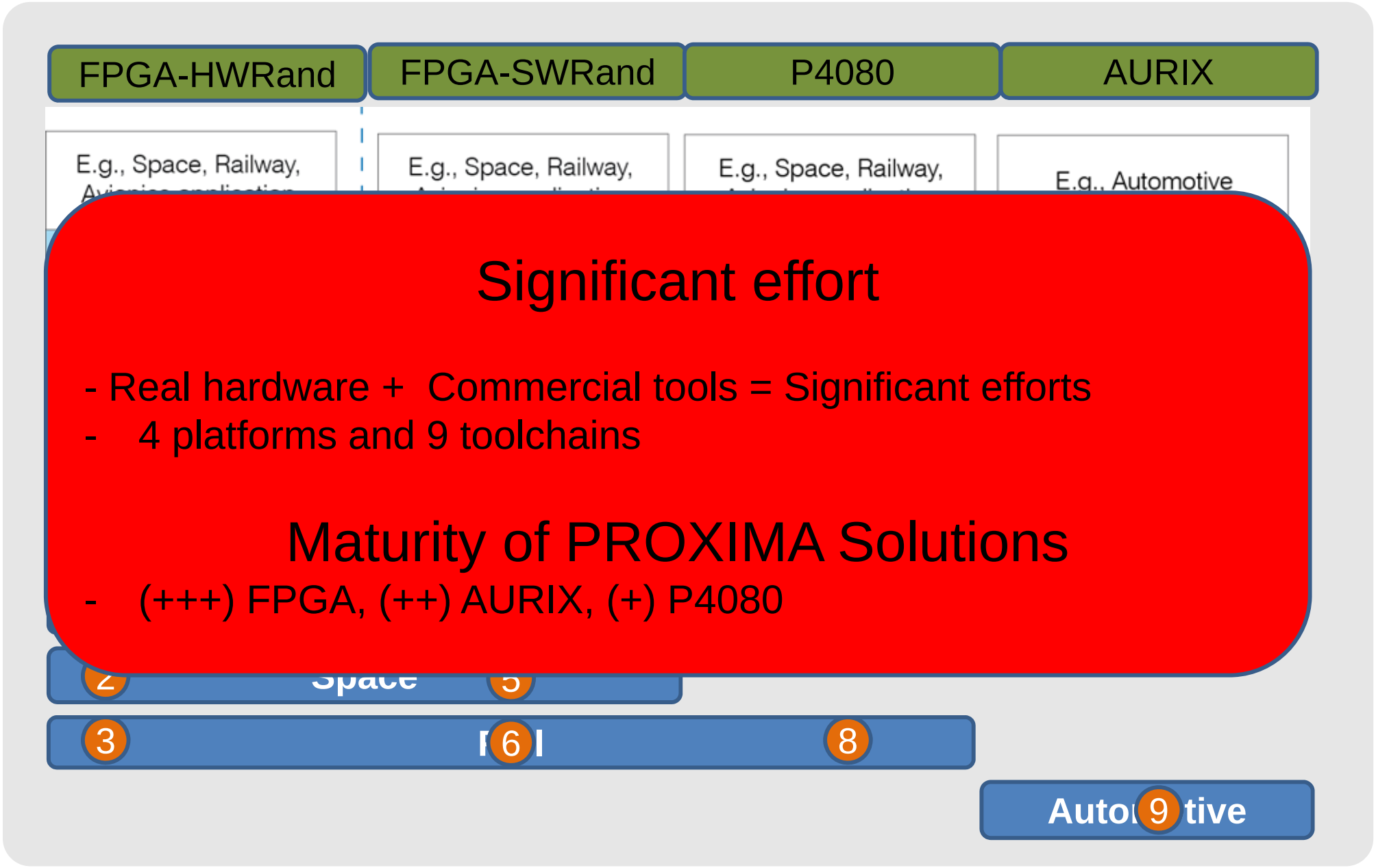


Fig. 13. pWCET distributions (black line) and empirical CCDF (dashed red line) for `basefp` and `pntrch`.

Implications in the V&V process



PROXIMA platforms and toolchains



PROXIMA

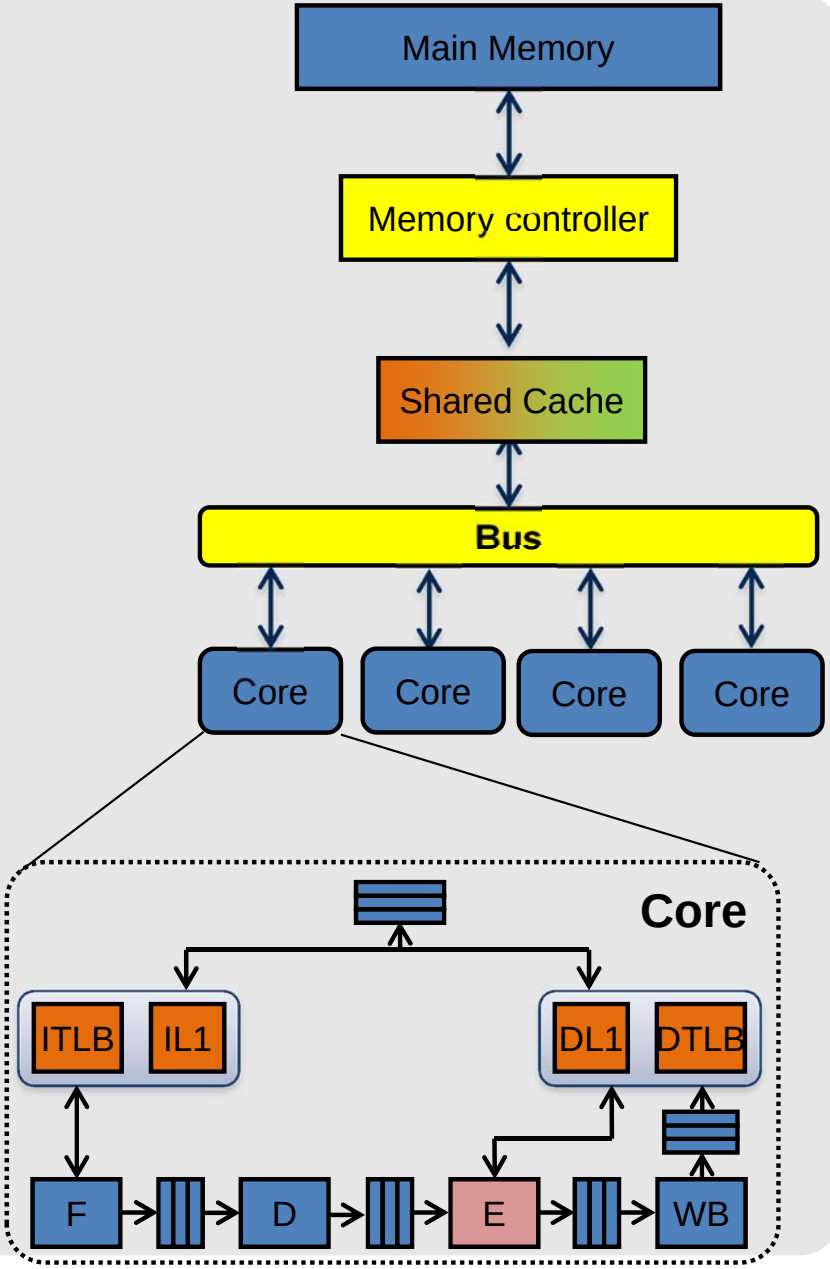
HWRand FPGA platform

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu

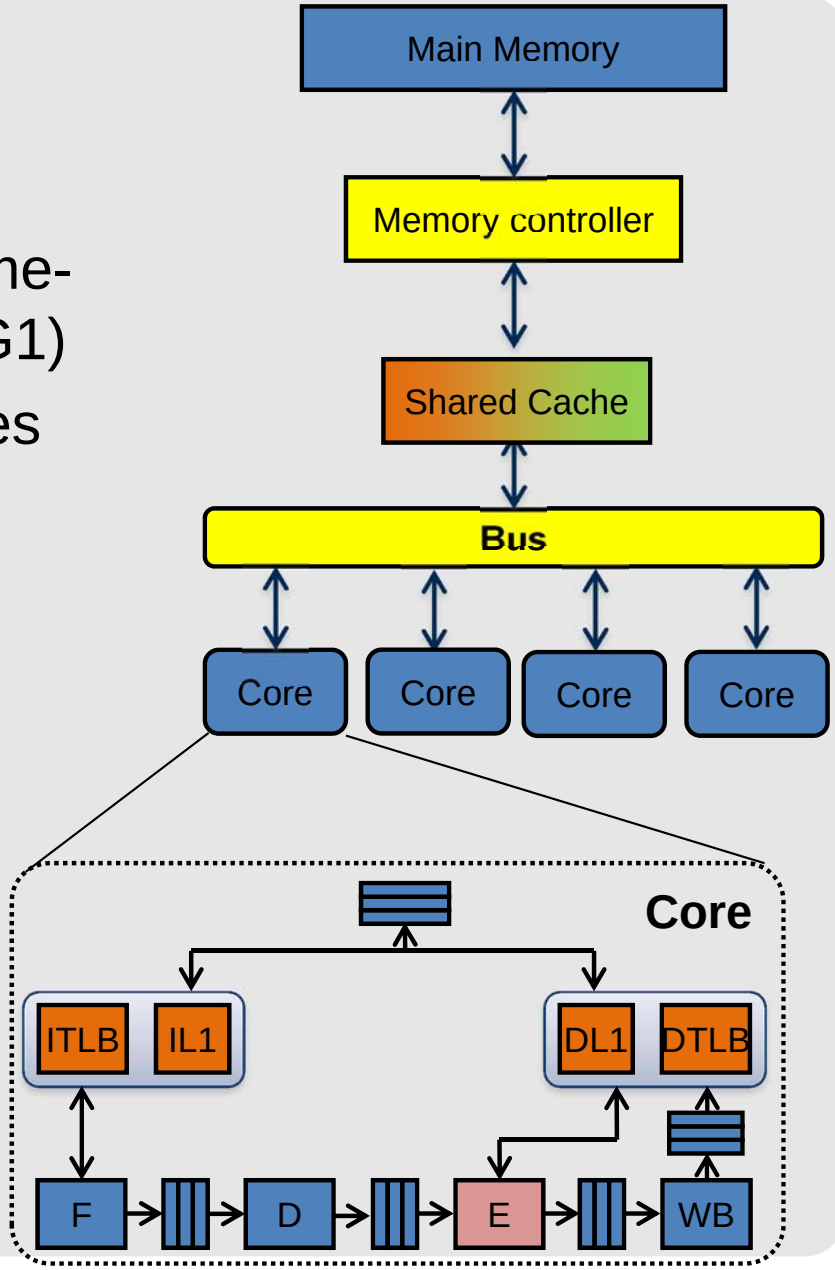
FPU: Upper-bound

- Bound from above
- FPU (FDIV, FSQRT)
  - FDIV latency: 15-18 cycles
    - Enforce always 18
  - FSQRT latency: 23-26 cycles
    - Enforce always 26
  - Limited pessimism
  - End user does not need to control input values operated



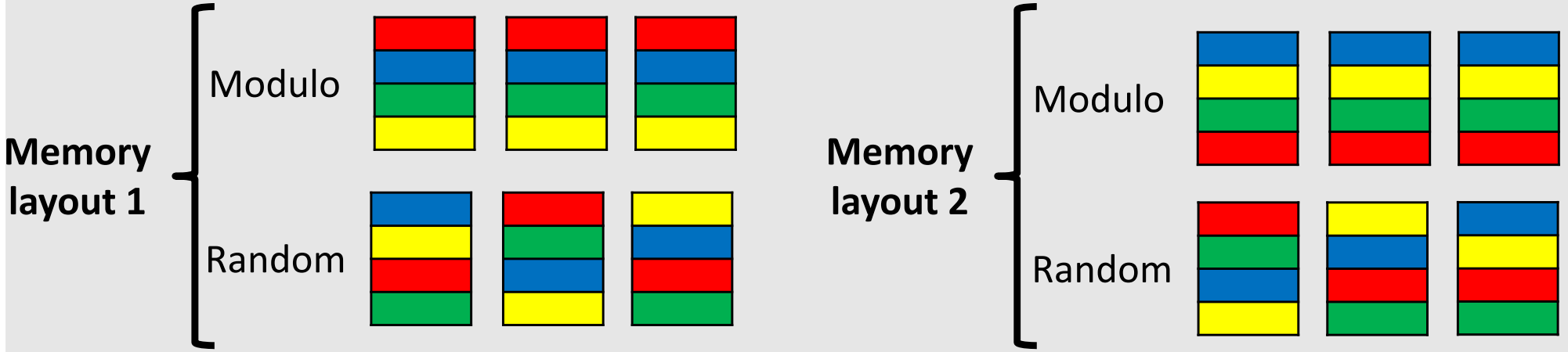
Caches: Randomization

- Randomization
- Cache placement
  - **Goal**: remove the need to control memory placement at analysis (goal G1)
  - Random modulo (IL1, DL1) provides randomization needed with similar performance as modulo
  - Random placement (L2)
- Cache replacement
  - Random replacement (all caches)



Caches: Randomization (2)

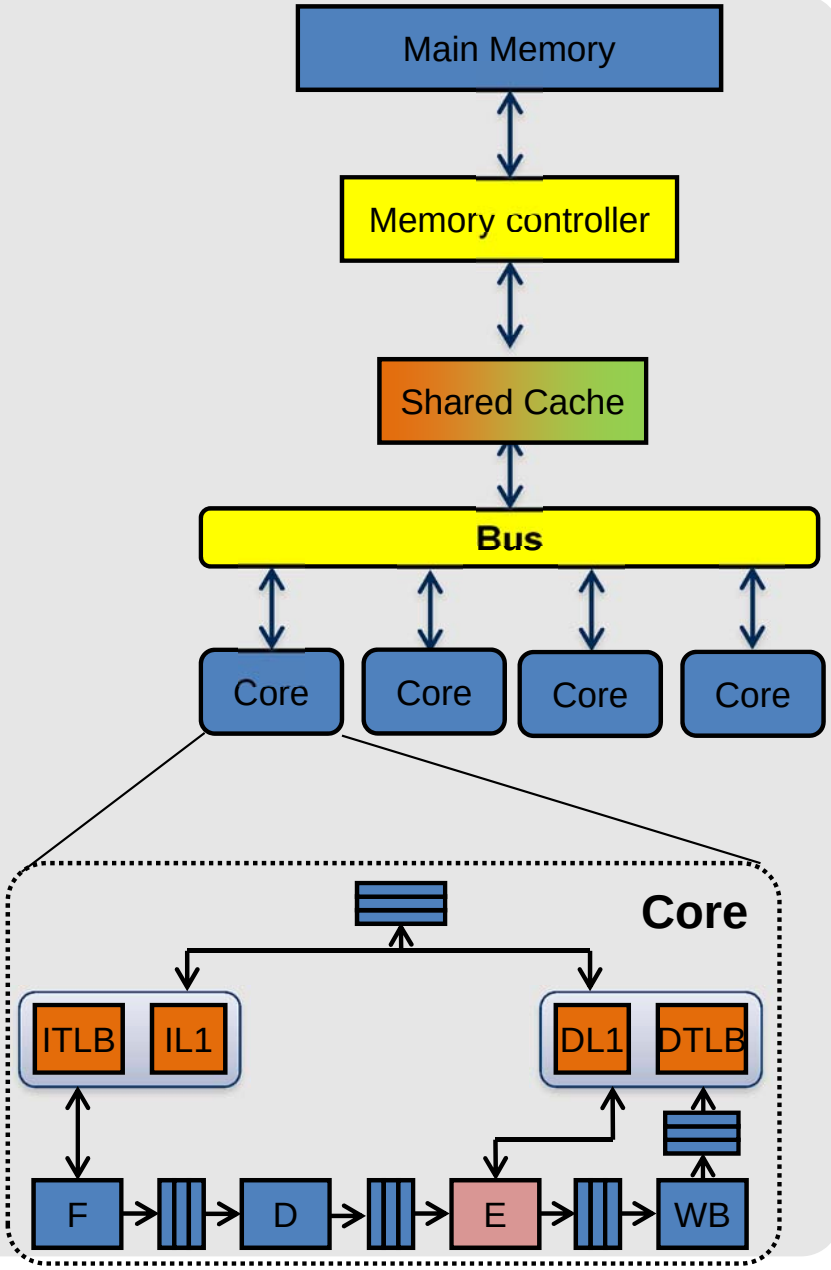
- Random cache placement
  - User released from having to control memory layout
  - Impact of memory layout factored in with randomization
    - An argument can be done on the probability of bad cache layouts to be captured



- Random cache replacement
  - Not mandatory but reduces probability of bad cases

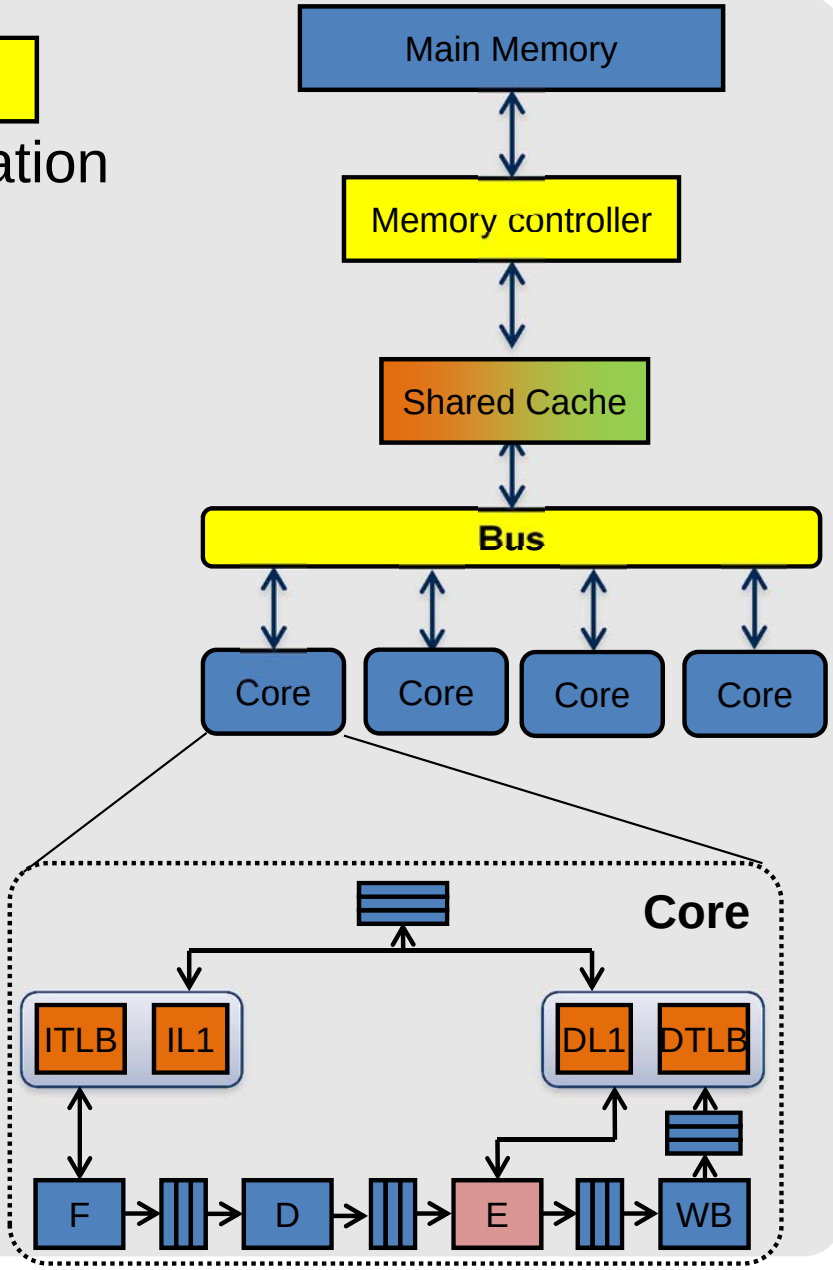
L2

- L2 cache partitioned to keep time composability
- Randomization principle applied to placement and replacement
  - As for first level caches

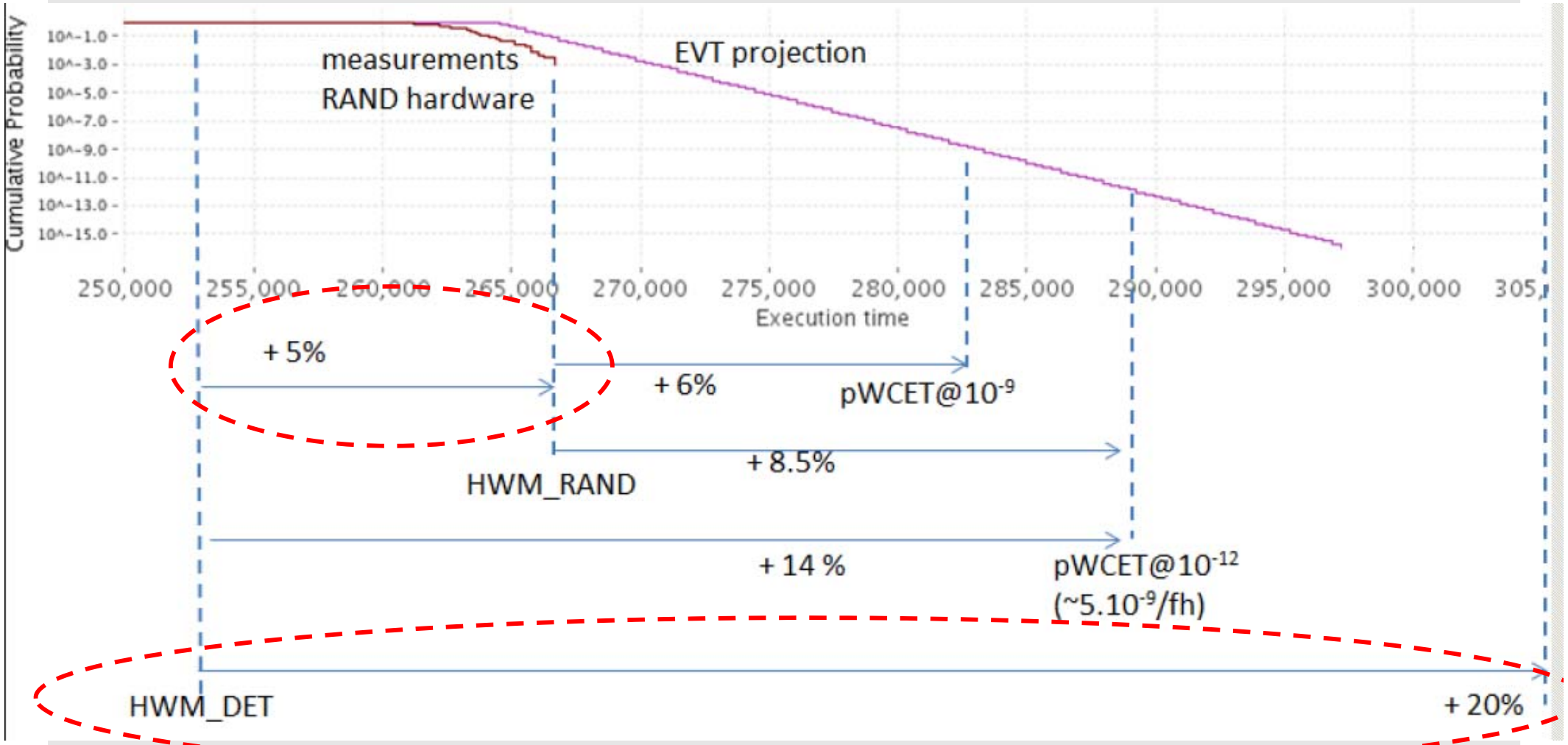


Shared resources: Rand + Upper-bound

- Randomization + upper-bound
  - Number of rounds to wait → arbitration
  - Duration of each round
- Arbitration
  - Randomized
  - User released from controlling time alignment of requests across cores
- Round duration
  - Upper-bounded to attain time composability
  - Bandwidth can be allocated homogeneously or heterogeneously



Results (single core)



PROXIMA

SWRand FPGA platform

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu

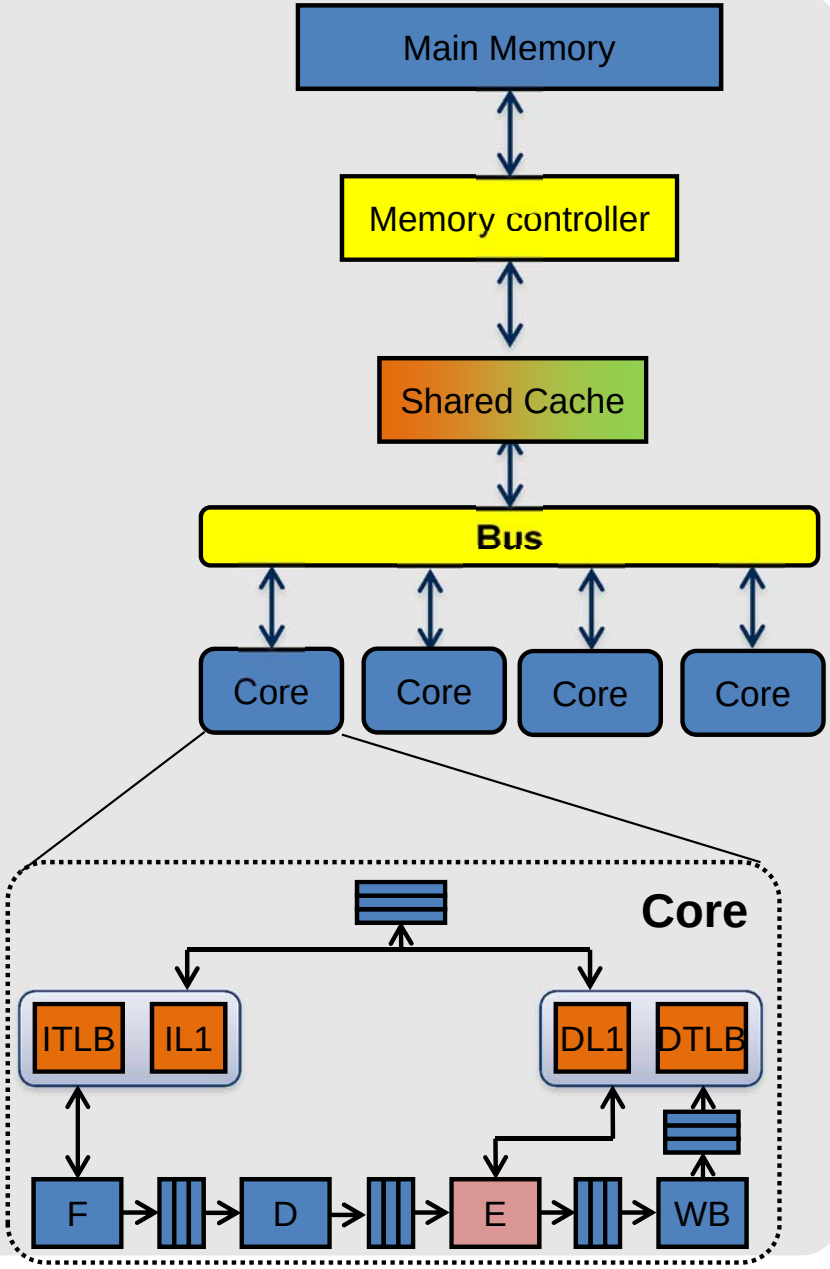
Challenges

- How to achieve the time properties required by MBPTA on COTS processors?

- For the type of COTS we have analysed, there are two main Sources of Jitter (SoJ)
  - Caches → SW randomization
  - FPU → padding
  - Multicore contention → MBPTA multicore analysis (VICI analysis, pronounced as ‘VC’)

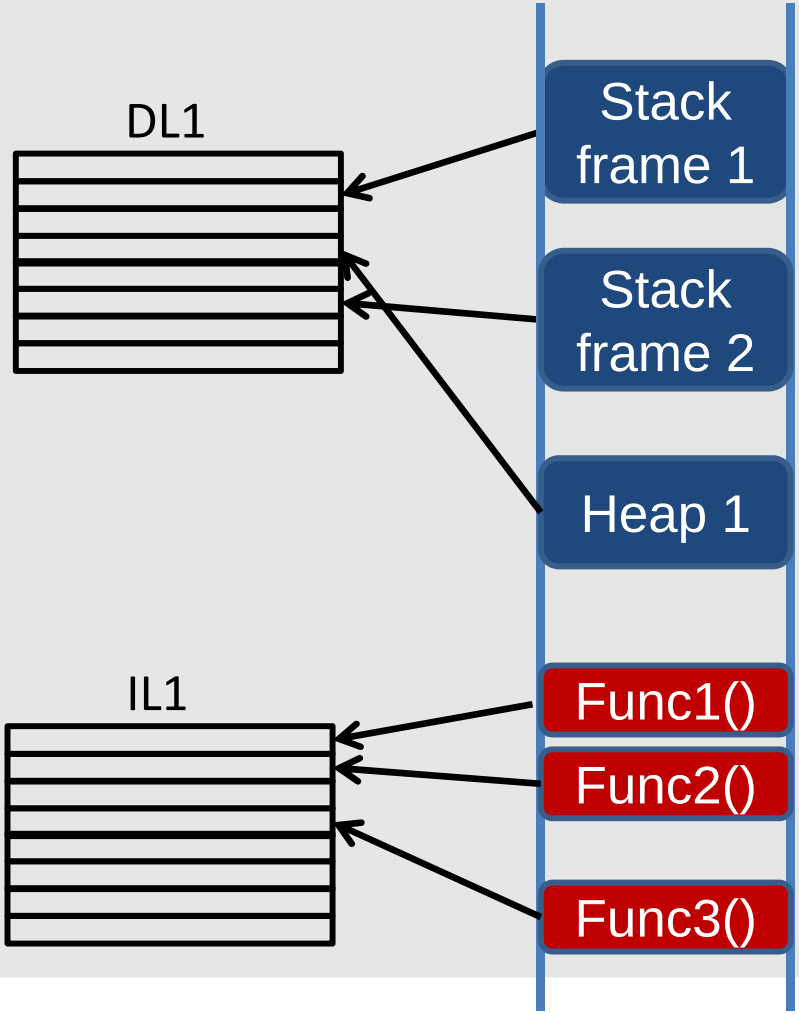
Caches: Randomization

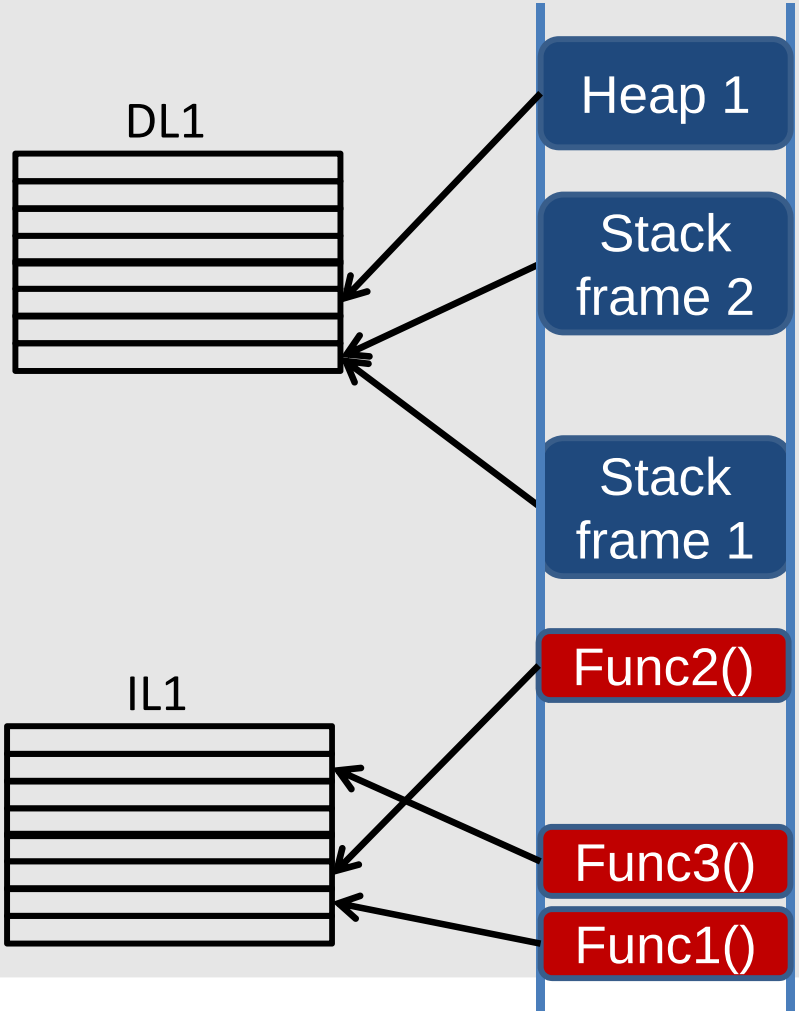
- Randomization 
- Cache placement
  - Randomize by SW location in memory of code and data
    - Random memory locations lead to random placement in IL1, DL1, L2
  - TLBs fully associative so no placement
- Cache replacement
  - Indirectly has some randomization due to random placement
- L2 contention 
  - Cache partition across cores



Caches: Randomization (2)

- Place objects (functions, stack, global data, etc.) in **random memory locations**
  - **Emulate RP but at SW level**



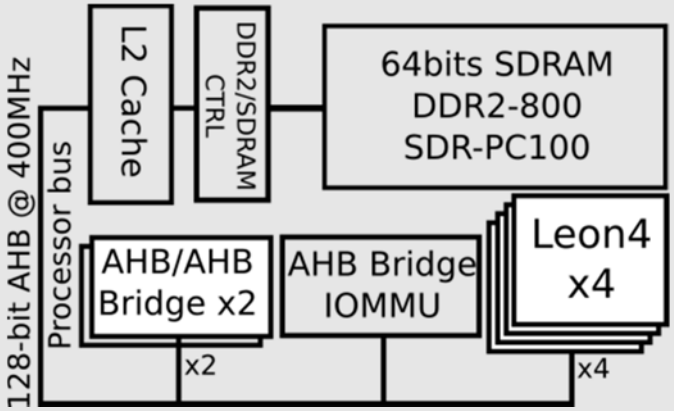


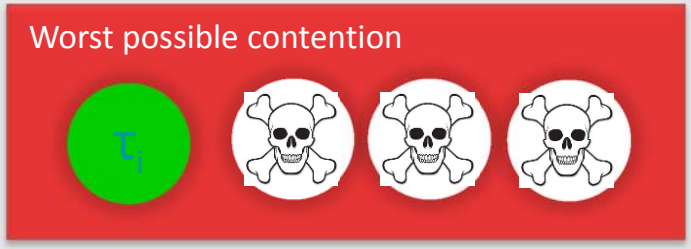
Chip level jitter

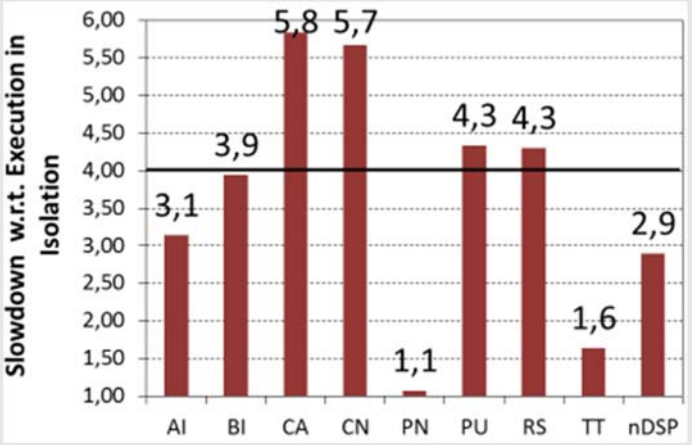
(deriving deterministic bounds to multicore contention)

Extremes of the spectrum of solutions

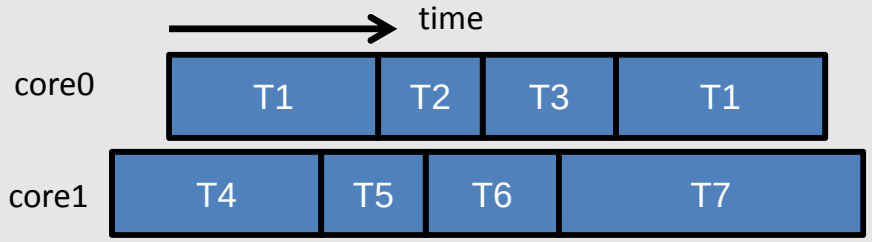
- full time composability







- No time composability or combined WCET estimation
  - Make a combined timing analysis of the tasks in the workload
    - Do not assume any contention, work with the actual contention



  - Any change of the tasks requires reanalyzing all tasks
  - Only shown to work on simulation environments

Partially Time Composable Bounds

- Goal:
  - Can we trade some time composability to tighten WCET?
  - Yes we can
- However… we do not want to ‘lose’ composability in a uncontrolled way
  - We do not want to reanalyze the whole workload when a task changes!
  - Time composability
    - From all or nothing metric, to a metric with degrees
    - Partial Time Composability
- we do not want to disrupt the measurement-based approach of MBPTA

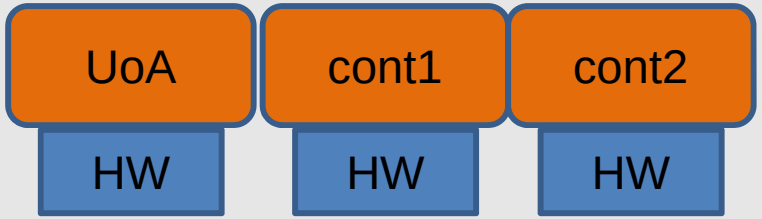
Initial Results

- Intuition:
  - S → abstracts the resource usage of the task under analysis, τ_A
  - T → abstracts the resource usage of contender tasks, $c(\tau_A)$
- Goal:
  - S and T make the WCET derived for τ_A, time composable with respect to a particular usage *U* of the hardware shared resources made by $c(\tau_A)$
  - Rather than with respect to the particular set of co-runners $c(\tau_A)$
- Principle when deriving $WCET_A^U$
  - $WCET_A^U$ determined for a particular utilization U
  - $WCET_A^U$ composable under any workload with resource usage < U
- Hence:
  - "**Forget particular tasks, focus on their resource usage**"
  - WCET = f (S,T) rather than ~~WCET = f(τ_A,$c(\tau_A)$)~~

Contention model

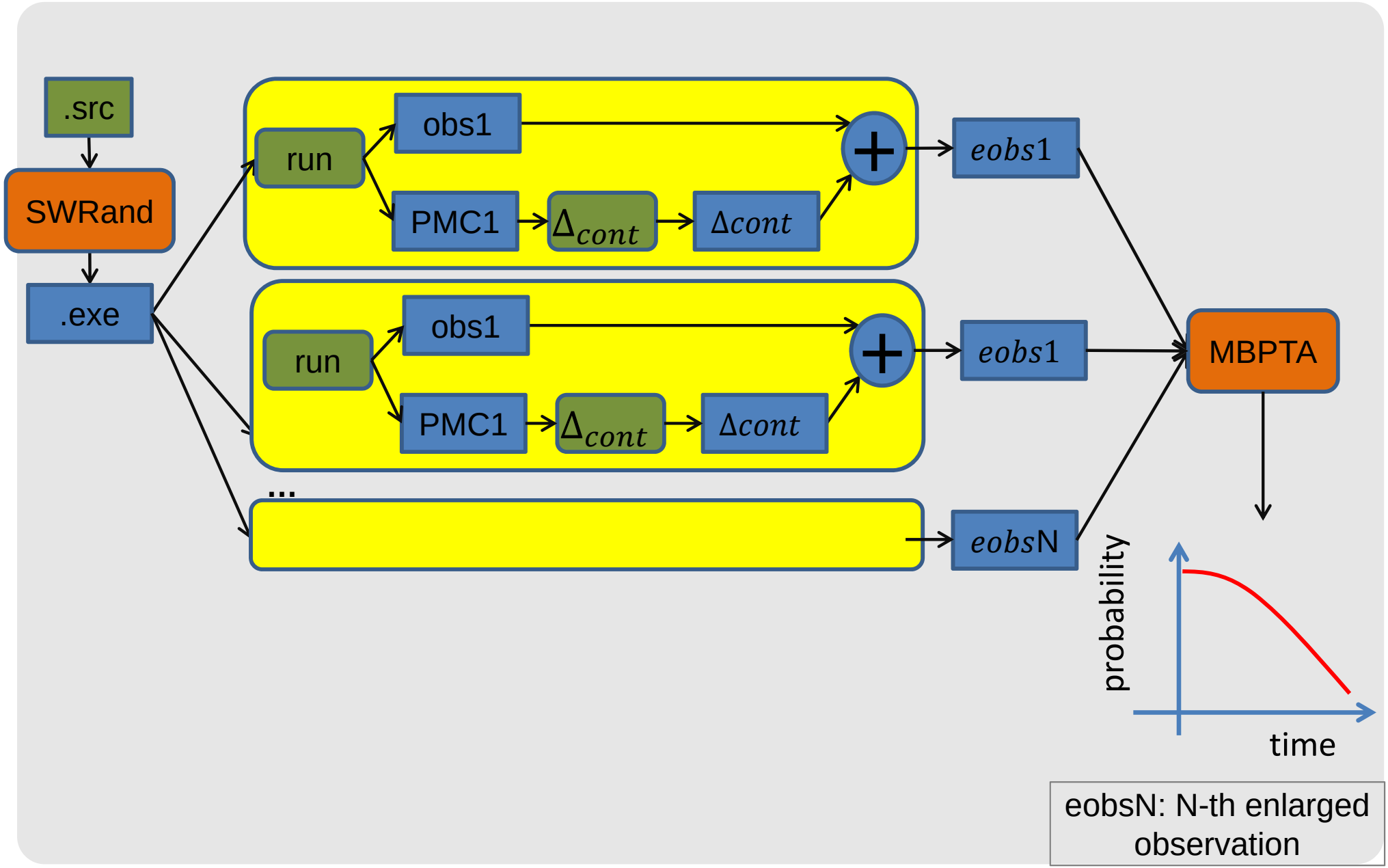
- All the technology based on performing runs in isolation of
  - The task under analysis
  - The contender tasks

  collecting PMCs ($misses, bus acceses,…)
- The model combines those PMC readings and produce contention bounds



- Properties
  - Works with randomized and non-randomized architectures
  - Single core runs → no multicore runs
    - Reduce experiment complexity and time
  - Time composability measure clearly defined
    - Tighther results that fully time composable

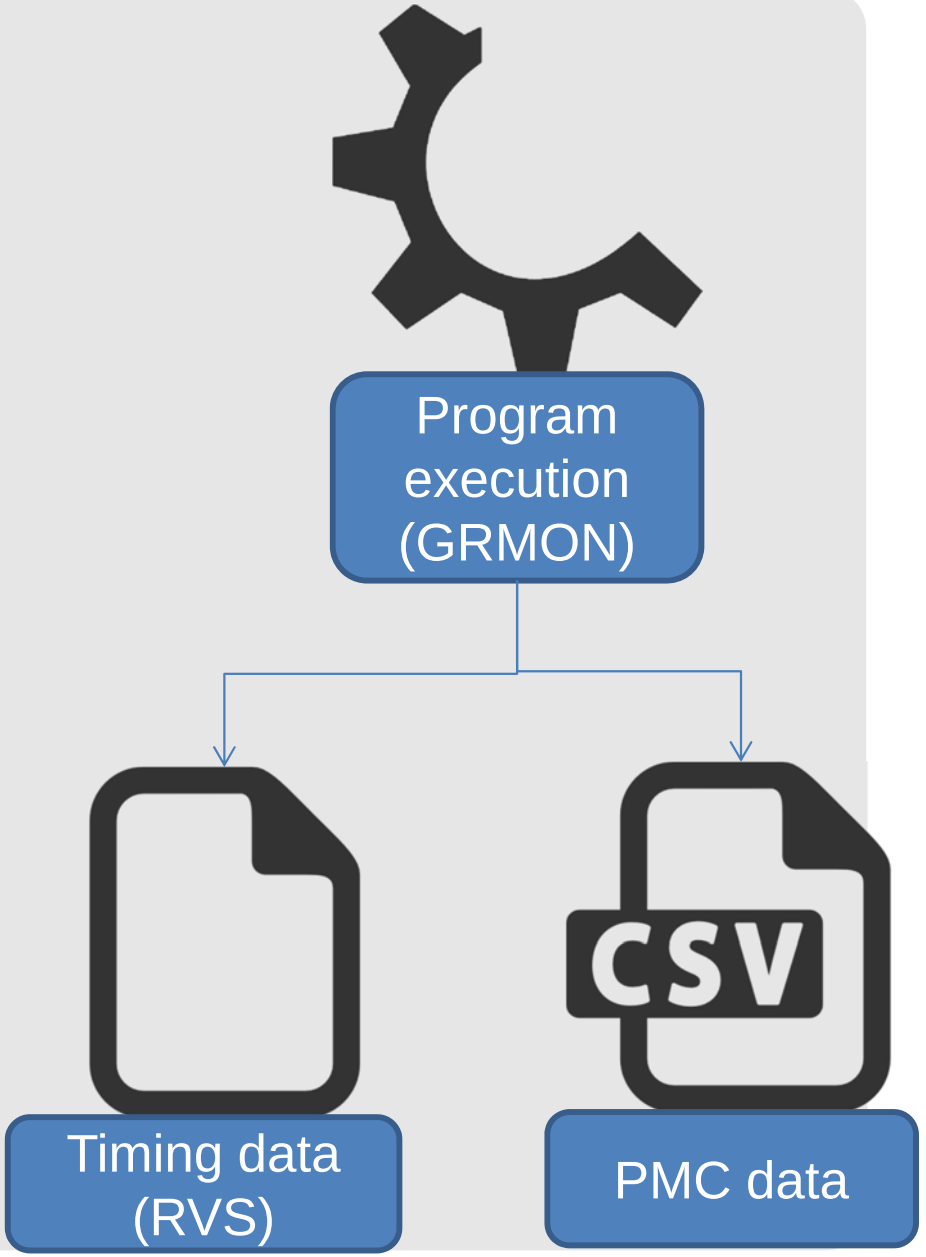
MBPTA for SW randomized single-core



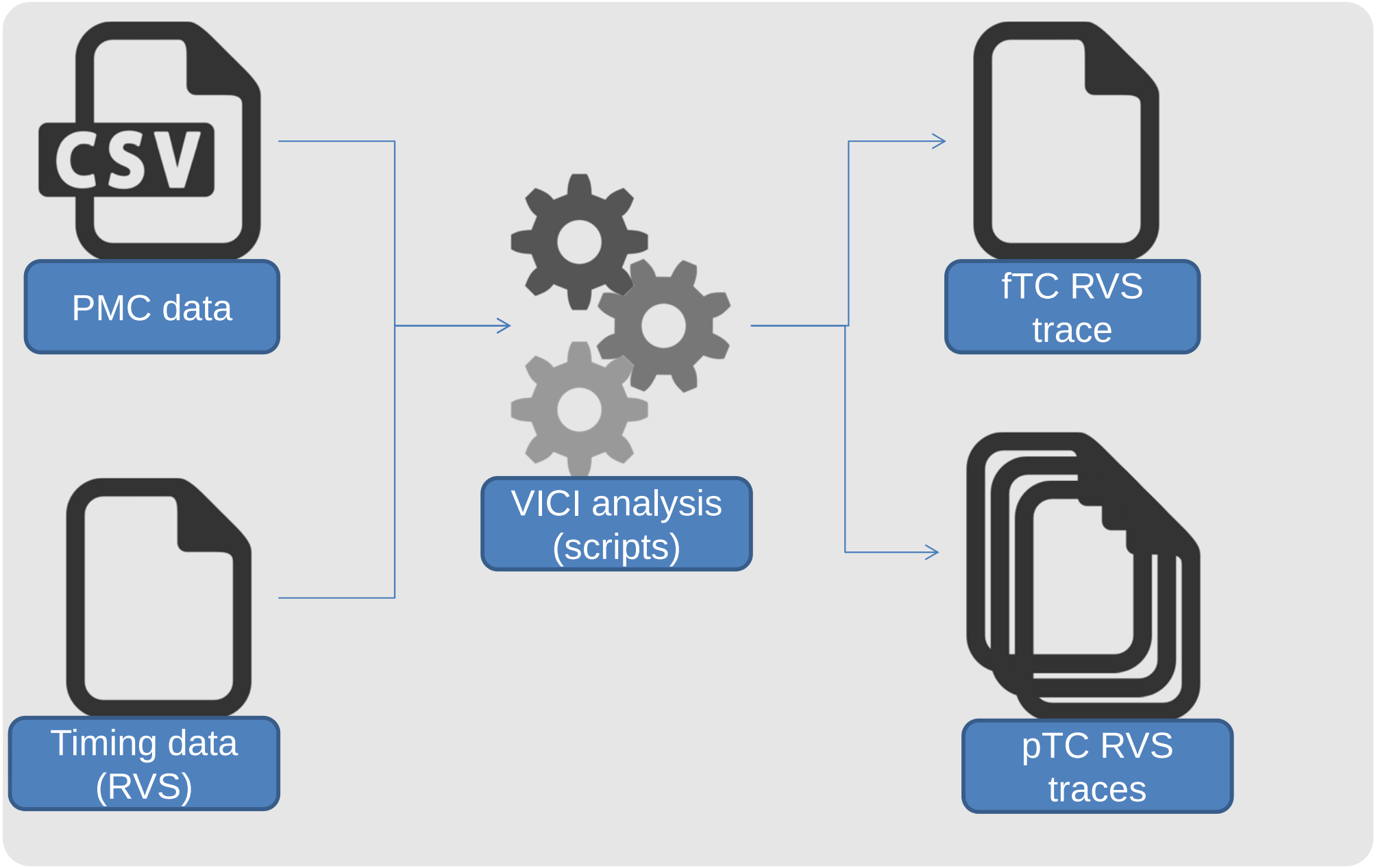
Integration into Rapita's Verification Suite

RVS

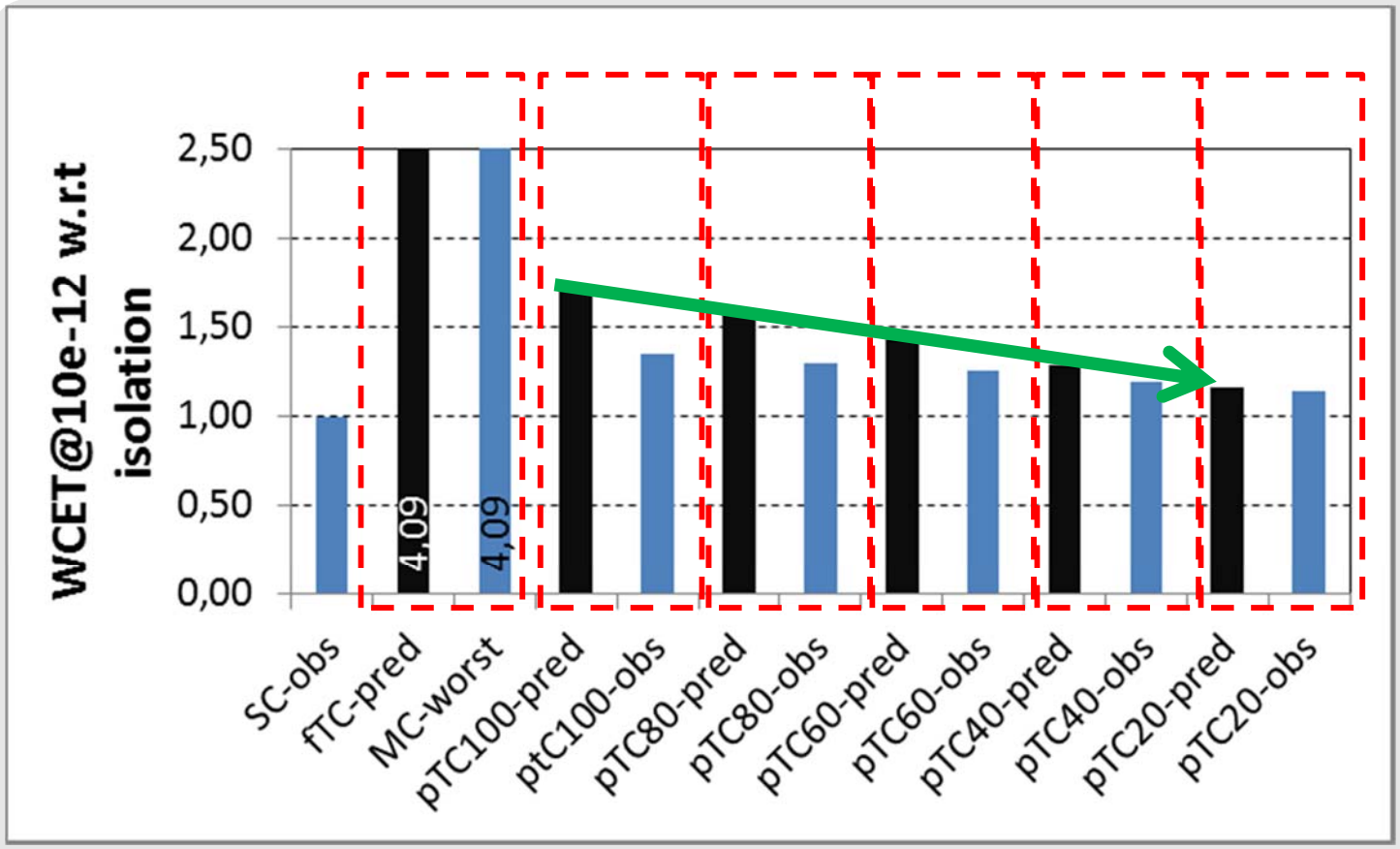
RapiTime



Integration into Rapita's Verification Suite



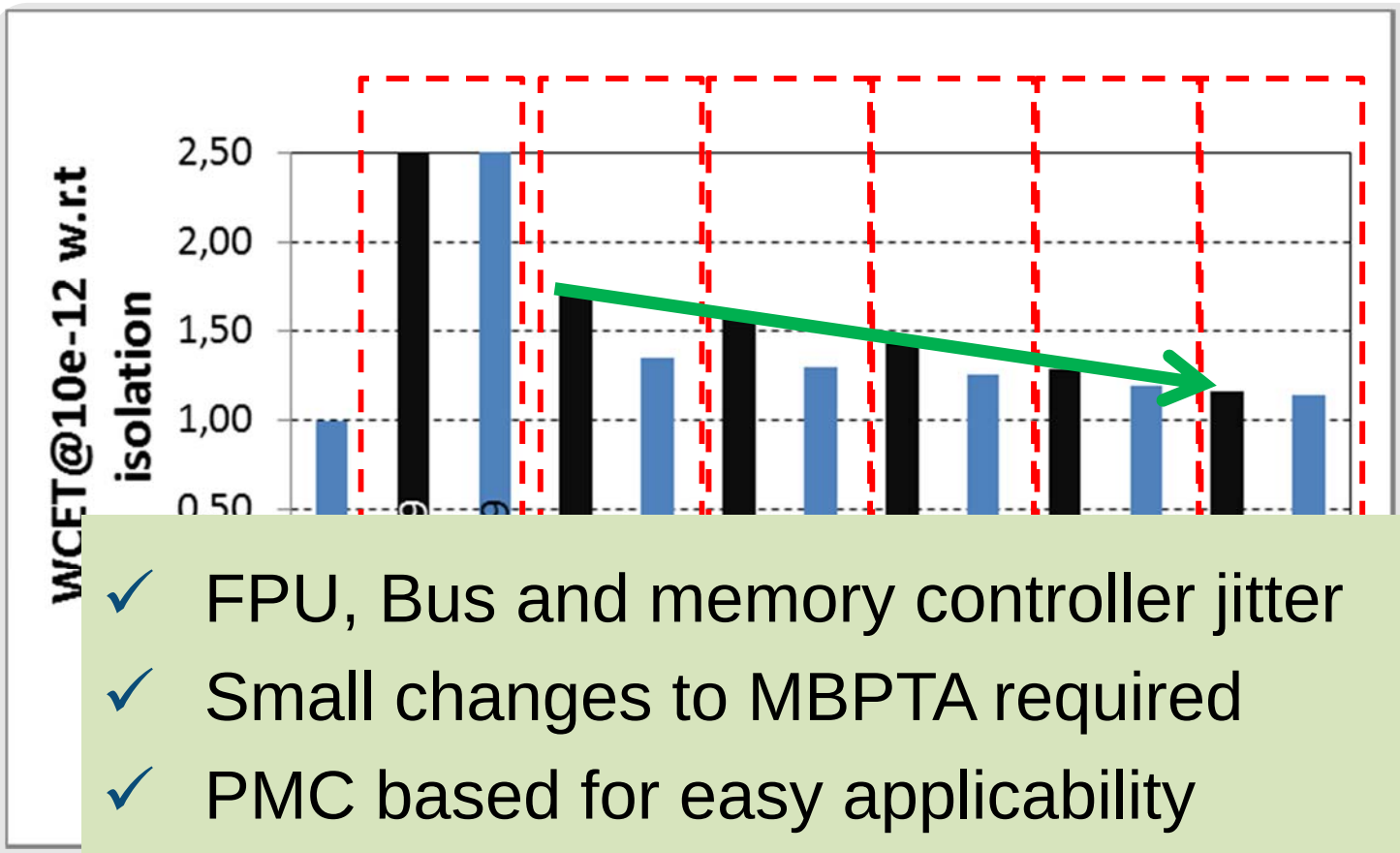
MBPTA projection SWRand multicore. noEPC



- 2 tasks:
  - 1 task,
  - 1 cont.

- ✓ fTC, pTC tight models
- ✓ fTC captures worst case
- ✓ pTC effectively adapts to contenders load on the bus

MBPTA projection SWRand multicore. noEPC

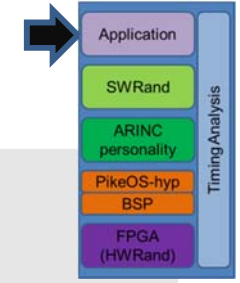


- 2 tasks:
  - 1 task,
  - 1 cont.

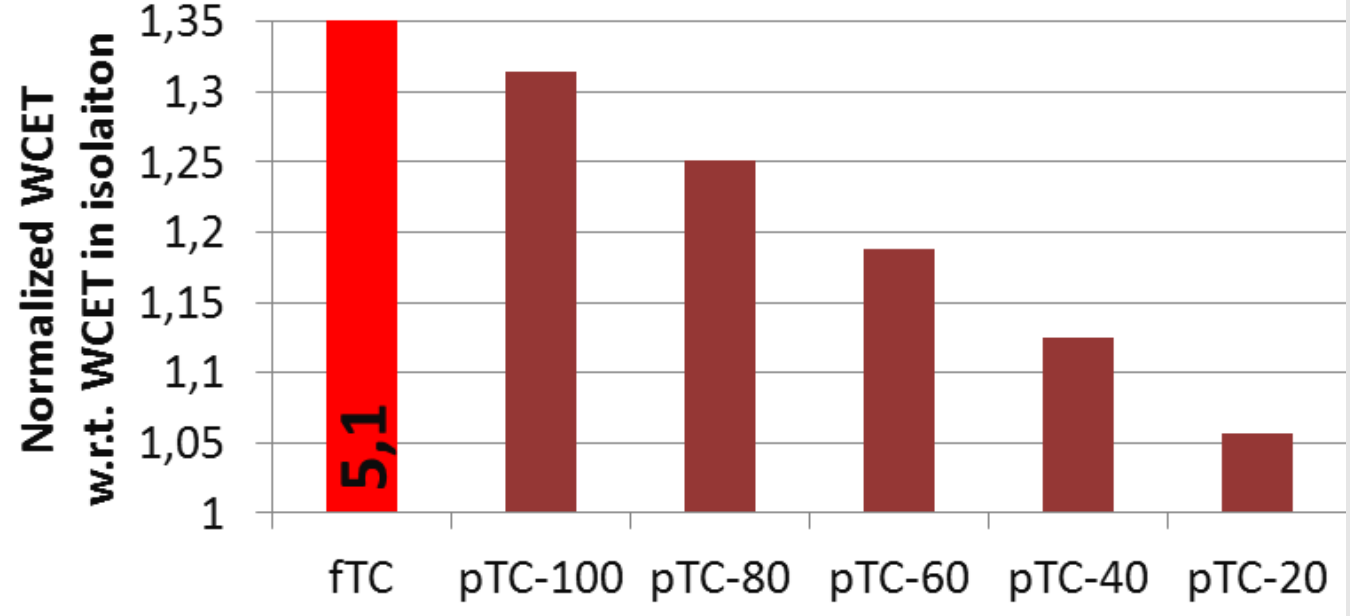
- ✓ FPU, Bus and memory controller jitter
- ✓ Small changes to MBPTA required
- ✓ PMC based for easy applicability
- ✓ Fully TC & partially TC to reduce pessimism

- ✓ fTC, pTC light models
- ✓ fTC captures worst case
- ✓ pTC effectively adapts to contenders load on the bus

Results Railway: FPGA SWRand multicore



- Setup (L2 WriteBack, Round-Robin Bus)
  - WCET normalized to Exection Time in isolation.
  - fTC. Each request of the IKR app. is assumed to contend with a request of the worst type from every other core
  - ptC-100. IKR app. runs against 3 copies of itself.
  - ptc-80. IKR app. runs against 3 copies of an application performing 80% of the accesses the IKR application does



- Tighter estimates than fTC
- Adaptability to contenders load
- Keep some degree of TC

Assumptions

- Assumptions:
  - Run to completion
  - Reliable PMC readings
  - Stressing Kernel methodology derives worst contention latencies
- Other:
  - Task under analysis is randomized
  - Contender tasks are not
    - They can be derived a bound to their access counts
    - Bound is resilient to cache layouts
- Future work:
  - **Probabilistic rather than deterministic bounds**
  - Probabilistic access counts of the contenders bounding impact to contenders cache layouts

PROXIMA

Conclusions

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

Conclusions

- PROXIMA deals with low-level source of jitter
  - Their variability naturally exposed when performing experiments
  - Aims at reducing user intervention and knowledge about low-level sources of jitter
- Randomization and ‘work on worst latency’ approaches used as a means to ensure jitter arises in the measurements
  - Can be applied at Software or Hardware level
- EVT to derive the combined probability of observed events
- Partial results shown
  - Project ends in Sept 30th
  - More information will be provided in the deliverables
    - Tools, Results,
    - …

PROXIMA

Improving measurement-based timing analysis through randomisation and probabilistic Analysis

Francisco J. Cazorla

Director of the CAOS research group at BSC and
researcher at the Spanish National Research Council (IIIA-CSIC)

Coordinator of the FP7 IP Project PROXIMA

EMC^2 Workshop. September 28th 2016.
Paris, France
www.proxima-project.eu

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085