Randomized Caches Can Be Pretty Useful to Hard Real-Time Systems

Enrico Mezzetti[1], Marco Ziccardi[1], Tullio Vardanega[1], Jaume Abella[2], Eduardo Quiñones[2], and Francisco J. Cazorla[2,3]

1 **University of Padova**
{emezzett,mziccard,tullio.vardanega}@math.unipd.it
2 **Barcelona Supercomputing Center**
{jaume.abella,eduardo.quiñones,francisco.cazorla}@bsc.es
3 **Spanish National Research Council (IIIA-CSIC)**

Abstract

Cache randomization per se, and its viability for probabilistic timing analysis (PTA) of critical real-time systems, are receiving increasingly close attention from the scientific community and the industrial practitioners. In fact, the very notion of introducing randomness and probabilities in time-critical systems has caused strenuous debates owing to the apparent clash that this idea has with the strictly deterministic view traditionally held for those systems. A paper recently appeared in LITES [17] provides a critical analysis of the weaknesses and risks entailed in using randomized caches in hard real-time systems. In order to provide the interested reader with a fuller, balanced appreciation of the subject matter, a critical analysis of the benefits brought about by that innovation should be provided also. This short paper addresses that need by revisiting the array of issues addressed in the cited work, in the light of the latest advances to the relevant state of the art. Accordingly, we show that the potential benefits of randomized caches do offset their limitations, causing them to be – when used in conjunction with PTA – a serious competitor to conventional designs.

2012 ACM Subject Classification Computer systems organization~Real-time system architecture; Theory of computation~Probabilistic computation; Computer systems organization~Special purpose systems;
Keywords and phrases Real-time systems, probabilistic WCET, randomized caches
Digital Object Identifier 10.4230/LITES.xxx.yyy.p
Received Date of submission. **Accepted** Date of acceptance. **Published** Date of publishing.

Editor LITES section area editor

1 Introduction

Timing analysis techniques aim at deriving upper bounds to the worst-case execution time (WCET) of a software program or parts of it. Timing analysis techniques are generally classified as either static or measurement-based [19]. Static WCET analysis techniques try to compute a provably trustworthy WCET bound for a given program from an abstract model of a processor and the source or executable of the program. Measurement-based analysis aims to contain the intrinsic pessimism of static techniques, by measuring the execution of small program fragments (typically basic blocks) run on the target hardware, and then combining them into WCET values using structural information on the program under analysis.

The recent advent of approaches based on probabilistic arguments [5, 6] suggests that the family of timing analysis techniques can also be usefully classified in relation to their using either deterministic (DTA) or probabilistic reasoning (PTA). Four distinct approaches can then be singled out in that respect: static and measurement-based deterministic timing analysis (SDTA and MBDTA), on the one end of the spectrum; static and measurement-based probabilistic timing analysis (SPTA and MBPTA), on the other [1].

As a precondition to computing sound and tight bounds to worst-case execution time, DTA and PTA alike postulate predictability in the timing behavior of the system of interest, from both the processor and the application perspective (whether top-down or bottom-up). Yet they do so with different flavors. Deterministic approaches require or assume execution platforms whose hardware and software components can be accurately abstracted into a fully deterministic model. Conversely, probabilistic methods allow combining deterministic and randomized timing behavior in processor resources [15], to produce WCET estimates that are guaranteed to be exceeded only with a given (arbitrarily low) probability of occurrence.

Cache memories indeed are one of the processor resources whose timing is randomized in the context of use of PTA [12, 13]. This notwithstanding, PTA should *not* be understood as a timing analysis method *for* randomized caches [1]. Conversely, time randomization (applied to cache memories, but not only) should be regarded as an enabling factor to PTA.

In a recent work, Reineke [17] has critically reviewed the impact of randomized caches on timing analysis [12,13], raising an array of serious concerns and concluding that randomized caches are harmful for hard real-time systems. While the author's concerns are valid and need to be addressed, the conclusions arrived at are negatively slanted by failing to equitably consider the benefits that randomized caches do bring about.

In this short paper we review the observations and conclusions made in [17], extending them in a way that: (1) the advantages of using randomized caches in hard real-time systems are captured and understood; (2) recent results, appeared subsequently to Reineke's cited work, are recalled to mitigate some of the author's concerns; and (3) finer-grained reasoning compounded by the fresher perspective provided by the point above, is presented for the original conclusions that we deemed too general.

Overall, our work provides substantive arguments to contend that randomized caches used together with PTA should be regarded as competitive alternatives to conventional caches used together with DTA. This is especially true in scenarios where set-associative caches are used in the target processor (which is very common indeed) and timing analysis is performed with measurement-based approaches: there, randomized caches together with MBPTA increase the user's ability to compute trustworthy and tight WCET estimates with less effort.

2 Recap of main arguments against the adoption of randomized caches

The arguments made against the adoption of randomized caches in hard real-time systems in [17] follow a rather schematic flow. The random replacement and placement policies are in fact evaluated for efficiency and precision against their deterministic counterparts – Least Recently Used (LRU) replacement and modulo placement – with respect to both static (SDTA and SPTA) and measurement-based (MBDTA and MBPTA) WCET analysis techniques. The reasoning uses both fully associative and set-associative caches.

The relevant observations and conclusions are recalled below. In the interest of brevity, in the following narrative we do not use verbatim quotes from [17], but rather our own interpretations of the author's observations and conclusions.

2.1 Random replacement in fully associative caches and SPTA

The first part of the argument considers fully associative caches and develops a comparison between a predictable deterministic replacement policy (LRU) and an instance of random replacement

(Evict on Miss, EoM)[1]. After commenting on the correctness of the formula bounding the hit probability (originally proposed in [20] and recently improved in [9]), the superiority of LRU is claimed, building on the following two main observations:

O1 - LRU dominates EoM on reuse distance. LRU always guarantees better hit probabilities than EoM when the analysis exploits reuse distances[2].

O2 - Current SPTA cannot exploit stack distance. When LRU replacement is used in place of Random replacement, current cache analyses – used with SDTA – can benefit from bounding stack distances[3] that can be arbitrarily lower, hence more accurate and less pessimistic, than reuse distances.

The conclusion drawn in this regard is that:

C1 - LRU replacement is always preferable to EoM. With simple, state-of-the-art analysis methods, LRU replacement is preferable to Random replacement in S*TA.

2.2 Random replacement in fully associative caches and MBPTA

The mirror argument against EoM in MBPTA builds on the restrictive assumptions made in the early (still immature) formulation of it presented in [8]. Namely, a variant of the method limited to single-path programs, with known worst-case initial state and no input dependence.

The conclusion here is that:

C2 - Deterministic replacement enables more efficient MBPTA. (Under restrictive assumptions) a single execution with LRU is sufficient to exhibit the worst-case behavior against hundreds in MBPTA with Random replacement, which makes deterministic replacement far more efficient for MBPTA than Random replacement.

2.3 Random placement in set-associative caches and SPTA

In contrast to fully associative caches, set-associative caches rely on mapping memory blocks to cache sets. In discussing placement policies, [17] compares the classic deterministic modulo placement policy (MOD) with random placement (RAND). The author argues the superiority of MOD over RAND on the observation that the formula for the computation of hit probabilities suffers from weaknesses similar to those noted in the replacement case, in that RAND analysis fails to account for existing dependencies.

A counterexample is presented to sustain that:

---

[1] Note that, from the PTA perspective, LRU exhibits a *degenerate* hit probability, which is either 0 or 1, but nothing in between.

[2] The notion of reuse distance captures the number of *cache accesses* that occur in-between two consecutive accesses to the same address. For instance, in the sequence of accesses a, b, c, b, a, the reuse distance for the second occurrence of access a is 3.

[3] The notion of stack distance captutes the number of *unique addresses accessed* in-between two consecutive accesses to the same address. For instance, in the sequence of accesses a, b, c, b, a, the stack distance for the second occurrence of access a is 2.

O3 - The independence assumption fails. With RAND, hit probabilities depend on prior accesses to the same set, which breaks the independence hypothesis[4] and compels the user to always assume zero probability of hit.

And therefore:

C3 - Random placement nullifies SPTA. Random replacement cannot be adopted in so far as SPTA cannot handle conditional probabilities.

2.4 Random placement in set-associative caches and MBPTA

Random placement is claimed inappropriate for MB*TA approaches as caches may exhibit extremely poor performance for unfortunate, yet very rare, mapping (conflict) conditions. With MB*TA, therefore, one should either capture those cases in the measurement observations, thus incurring possibly massive overestimation, or ignore (and miss) them, thus becoming unsound. Moreover, those extreme cases, being so far apart, may even cause the statistical test on identical distribution to fail [10]. In any such case, MBPTA will fail.

The firm conclusion here is that:

C4 - Random placement is not adequate for MBPTA. Random placement causes MBPTA to fail.

3 Arguments in favor of randomized caches

In the following we first systematically analyze each claim made in [17] and recalled in section 2, and then provide specific arguments in support of randomized caches and their use in probabilistic timing analysis approaches.

Revisiting C1 (O1+O2)

It is indeed correct to say that, if we limit ourselves to the reuse distance metric, current state-of-the-art SPTA approaches cannot do better than LRU. Observation O2 is therefore accurate, in the sense that current SPTA does not exploit stack distances. Yet, no intrinsic trait of SPTA allows making final claims on this matter, as further improvements, such as e.g. those outlined in [3, 4], may well arrive at mitigating that limitation.

As noted in [17], current SPTA approaches cannot do better with information on reuse distances considered in isolation. However, having information on individual reuse distances implies also having global knowledge on all reuse distances. Interestingly, the hit probability of an access may benefit from the information on the reuse distances of all the accesses that have occurred within its reuse distance. As observed in [4], full knowledge on the reuse distances of all accesses in a sequence (as assumed by SDTA) allows information to be derived on the cache contention. The cache contention $con(e_l, T)$ of an access e_l in a sequence T is conceptually defined in [4] as the number of accesses within the reuse distance of e_l that have been assigned a non-zero hit probability. All accesses in $con(e_l, T)$ are assumed to use their own separate location in memory (see [4] for a formal definition of con). Using this information, a more precise lower bound to

[4] Independence in this context refers to whether the outcome of one cache access depends or not on the output of previous accesses. For instance, the fact that a previous access is a miss leads to an eviction, which may decrease the hit probabilities of subsequent accesses.

the hit probability of an access is provided in [4], captured by Equation 1, where k is the reuse distance and N is the associativity of the cache. It is important to observe that $con(e_l, T) \leq k$.

$$P(hit_{random}(e_l)) = \begin{cases} 0 & con(e_l, T) \geq N \\ \left(\frac{N-1}{N}\right)^k & \text{otherwise} \end{cases} \quad (1)$$

Table 1 reports an example taken from [4] where, building on contention information, random replacement is shown to outperform LRU, even when the latter relies on stack distance.

Table 1 Comparison of random and LRU hit probabilities for all accesses in the sequence $a, b, c, d, f, a, b, c, d, f$ when reuse and stack distances of all accesses are known (same information) in a cache with associativity 4

	a	b	c	d	f	a	b	c	d	f
stack distance	∞	∞	∞	∞	∞	4	4	4	4	4
$P(hit_{LRU})$	0	0	0	0	0	0	0	0	0	0
$P(hit_{random})$	0	0	0	0	0	$\left(\frac{3}{4}\right)^4$	$\left(\frac{3}{4}\right)^4$	$\left(\frac{3}{4}\right)^4$	0	$\left(\frac{3}{4}\right)^4$

From the example in Table 1 one can conclude that LRU together with SDTA does not always outperform random replacement together with SPTA.

Conclusion C1 was sustainable at that time of its writing. Yet, more recent advances [3, 4] show that it does not hold in the general case. In fact, it is overly difficult to determine whether one approach does always outperform the other, as this may largely depend on the particular system and its domain of application. Hence, no absolute claim can soundly be made against either approaches on the basis of the current state of the art.

Revisiting C2

The claim that random replacement is not efficient with MB*TA rests on the assumptions made in the initial formulation of MBPTA [8]. Considering those assumptions, conclusion C2 in [17] maintains that a single measurement would be sufficient to hit the WCET if deterministic replacement is used, instead of several hundreds with MBPTA.

LRU has been shown to provide better average performance than random replacement [12, 18]. However, systematic pathological cases may well occur with LRU (cf. [16, 18]), which cause it to behave much worse than random replacement. An example case follows. Consider a loop traversing a vector whose size is $N + M$ cache lines where N is the cache size expressed in lines. Under LRU, if $M = 0$ the vector fully fits in cache and all accesses (except cold misses) are hits. This is the best case and no other replacement policy can do better. If $M > 0$ instead, and each cache lines holds a single element of the vector, *all* accesses are misses[5]. Consider now vector $V[]$, with $N = 2$, $M = 1$, and a sequence of accesses $V[1]$, $V[2]$, $V[3]$ in a loop, where $V[i]$ stands for the i^{th} position in V. At any iteration after the first one, when all accesses always miss, $V[1]$ misses and replaces $V[2]$, $V[2]$ misses and replaces $V[3]$, $V[3]$ does so with $V[1]$. In that case, there is no way to hit any line because they are *systematically* evicted before they can be reused. We refer to this phenomenon as pathological LRU behaviour. Thus, if $M > 0$ any other replacement policy cannot do worse than LRU. For instance, under random replacement there is always a

[5] If each cache line contained D distinct data items, then there would be 1 miss and $D - 1$ hits in that case, due to spatial locality. In the rest of this discussion we ignore hits due to spatial locality since they are identical for all examples and replacement policies.

non-zero probability of hit in cache even if $M > 0$. Of course, as M increases the hit probability decreases, but with a gentle slope and not abruptly, as shown in [18].

Random replacement can indeed incur pathological cases too. Yet, it would do so with a ridiculously low probability and not systematically, which arguably is an even more important trait. Consequently, random replacement leads to increases in execution time with (rapidly) decreasing probabilities. The execution time distribution for high values thus is smooth and the slope of its tail steep, such that lower probabilities of exceedance can be reached with small increases in the WCET threshold.

For LRU instead, we have just seen that whether the scenario is pathological or not, depends on correlation between the footprint of the program data and the cache size, in other words on the application use of the cache space. Small changes in that correlation can indeed create abrupt performance variations [18].

It can thus be contended that, regardless of the 1:100 ratio in the number of runs that LRU may need in comparison with random replacement, the determination of the most convenient replacement policy is so entirely application dependent that no valid general answer can be preordained.

Furthermore, conclusion C2 rests on an assumption that needs careful consideration for a complete view of the problem space to be had.

In point of fact, C2 may only hold – if at all – for fully-associative caches. The distinctive feature of fully-associative caches is that they allow cache analysis to focus exclusively on the replacement policy, disregarding the effects of placement. If set-associative or direct-mapped caches were used instead in the very example considered in C2, a single measurement observation could only provide a WCET bound solely valid for the specific memory layout the program had in the observed run. A memory layout is specific in many respects as it affects execution timing not only because of code placement, but also as an effect of data placement and alignment in memory and on the stack (ignoring here the use of dynamic data allocation). Hence, when the layout changes (even only for small displacements), arising in consequence of re-linking or incremental software integration, the previously computed WCET becomes invalid.

MBPTA cures this problem by leveraging random placement, which allows the analysis to abstract from consideration of all possible memory placements [12, 13].

A very distinctive trait of MBPTA, as opposed to to MBDTA, is its ability to take great benefit from the combination of upper-bounding and randomization in the timing behavior of processor resources, which allows it to implicitly capture a wider scope of execution conditions than MBDTA [7]. With that base, MBPTA yields a WCET value associated with a given confidence, quantified as the probability of exceedance. MBDTA, instead, is designed to only provide a single answer to the timing analysis question, which may consequently be only either true or false. Still, the way to arrive to it in practice is often more based on "engineering judgment", for coverage of unobserved execution conditions, than on scientific evidence of sort. Conversely, by upper-bounding and randomizing the sources of execution time variation in the processor, MBPTA replaces uncertainty by scientifically quantified probabilities.

Path coverage issues should also be considered here. While it is overly difficult with MBDTA as yet to determine how sufficient path coverage can be obtained and how different path traversals may affect cache behavior, MBPTA leverages the character of randomization, which attenuates causal effects. This trait intrinsically leads to less abrupt performance variations [16, 18], which eases the problem of identifying worst-case paths, thereby reducing the hardness of the path coverage problem [14]. Arguably therefore, using random replacement can be deemed to increase the benefits of randomization without increasing the number of runs needed for MBPTA.

In fact, randomization is a veritable enabler for MBPTA, which may equally well be applied to

other processor resources than for replacement policies in caches. The application of randomization to buses, for example, has been recently presented in [11].

Revisiting C3 (O3)

Conclusion O3 is correct for current SPTA approaches. The valid point in O3 is that, when random placement is adopted, SPTA cannot assign hit probabilities greater than zero to individual accesses, in the face of residual dependence between hit probabilities. It should be noted however that existing SPTA techniques [4] have solved a similar problem for random replacement, where independence across accesses is not guaranteed either. The cited work has proven that, in the case of random replacement, dependencies can be studied locally – with low complexity – to obtain tight lower bounds for hit probabilities. Similarly, given the example presented in [17], where the sequence of 10 accesses *{a, b, a, b, a, b, a, b, a, b}* is considered, one should be able to determine that the probability of having up to ten misses in that sequence – starting from an empty cache state – is exactly $\frac{1}{s}$, where s is the number of cache sets, and up to two misses otherwise.

It is therefore fair to say that there is nothing intrinsic that prevents a technique as young as SPTA from being able to solve the problem addressed by observation O3.

For the sake of comparison, it is also interesting to notice that deterministic placement has its share of weaknesses with respect to analysability. As we already noted in fact, the result of SDTA strictly and solely applies to the memory layout considered in the analysis. Any change in that, irrespective of how small, invalidates the result. Moreover, the particular memory location of objects may make accesses a and b in the previous example map to the same cache set and thus be always misses. Similarly, if the information provided to SDTA does not allow identifying whether a and b do indeed map to the same cache set, SDTA must then conservatively assume that they do, and consider all accesses as misses. It is therefore evident that SDTA may draw conclusions that are as bad as those of state-of-the-art SPTA, because of either a bad memory placement or lack of precise information.

Revisiting C4

Conclusion C4 builds on the observation that low-probability events may occur that lead to very high execution times. A low-probability event (possibly occurring, say, once in $1,000,000$ runs) is very unlikely to be observed within the few hundreds of runs typically required by MBPTA. Hence, the author of [17] maintains that MBPTA would be fooled by its very reliance on possibly very low probability of occurrence, as caused by, for instance, random placement.

At the time of Reineke's writing, conclusion C4 was a certainly valid claim. The Heart-of-Gold (HoG) technique [2], which appeared slightly later, addresses the observations that gave rise to C4. The HoG work specifically delves into the internal reasoning of MBPTA to identify why a C4-like scenario can occur and what processor resources can lead to it. In particular, the HoG authors show that the most challenging events are those whose probability of occurrence ($P_{extreme}$) is low enough for the probability of not observing any of them (P_{not_seen}) in the MBPTA runs, to be above the exceedance probability threshold set by the relevant system requirements. In the HoG work – and in much of the current PTA literature –, the assumption is made that such a threshold is related to the acceptable failure rate as sanctioned by the applicable safety-related standards for critical system components.

The work in [2] contains a detailed analysis of the processor resources that MBPTA wants randomized. The HoG method allows determining $P_{extreme}$ for the randomized resources of interest, thus identifying risky scenarios. Random placement is shown to be liable to such risks. The HoG solution to this problem is to either increase $P_{extreme}$ or the number of runs until

P_{not_seen} drops below the limit threshold. In that manner, the risky events are captured by MBPTA and the resulting WCET estimates can be regarded as trustworthy.

It can therefore be contended that conclusion C4 in [17] holds true solely for the original MBPTA method [8], but not for any MBPTA method enriched with [2].

Interestingly, a similar argument to C4 can be used to unveil a nice property of MBPTA, also highlighting a serious issue with deterministic placement policies and SDTA approaches. If an extreme pattern of memory accesses may occur that causes the cache to exhibit a pathological behavior leading to very high WCET, SDTA approaches cannot disregard that possibility, regardless of how rare it can be. With MBPTA and random placement, instead, each pathological behavior is considered only with respect to its probability of occurrence. In this light, in contrast with C4, we can conclude that random placement can in fact mitigate – as opposed to exacerbate – the effects of pathological worst-case behaviors. From an industrial perspective, the inherent robustness of MBPTA and randomized caches to pathological behavior on specific memory access patterns arguably is one of the strongest arguments in favour of probabilistic techniques.

4 Conclusions

We thoroughly enjoy the attention given to time-randomized architectures and probabilistic timing analysis in the scientific community for the assessment of what consequences they may bring to the timing analysis of hard real-time systems. We are equally interested in any critique that can be moved to those techniques as part of that scrutiny.

The work presented in [17] raised numerous issues on the adoption of randomized caches and PTA approaches. For the benefit of the interested community, this short paper places those considerations into the bigger and more complete picture where they rightly belong. To that end, we provide some additional observations and present recent research results, which collectively allow us to make the following contentions:

1. In the situations where set-associative caches and measurement-based timing analysis can be soundly used – whether by virtue of scientific authority or of social consensus among the domain stakeholders –, randomized caches together with MBPTA increase the ability to obtain trustworthy and tight WCET estimates while reducing the burden on the user.
2. In the other situations it is unclear whether deterministic analyses on top of conventional caches or probabilistic analyses on top of randomized caches are the best choice, in that the results obtained are highly dependent on the particular characteristics of the system of interest.

It can therefore be maintained that the advantages brought about by randomized caches as discussed in this paper make them a serious candidate for use with hard real-time systems in places where conventional caches exhibit limits that undermine their reputation.

Acknowledgements This work has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement 611085 (PROXIMA, `www.proxima-project.eu`). This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2012-34557, the HiPEAC Network of Excellence and COST Action IC1202: Timing Analysis On Code-Level (TACLe). Jaume Abella has been partially supported by the Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717. The authors also wish to acknowledge Benoît Triquet and Franck Wartel from Airbus France, for their useful comments on an earlier version of this paper.

References

1 Jaume Abella, Damien Hardy, Isabelle Puaut, Eduardo Quinones, and Francisco J. Cazorla. On the comparison of deterministic and probabilistic WCET estimation techniques. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2014.

2 Jaume Abella, Eduardo Quinones, Franck Wartel, Tullio Vardanega, and Francisco J. Cazorla. Heart of gold: Making the improbable happen to extend coverage in probabilistic timing analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2014.

3 Sebastian Altmeyer, Liliana Cucu-Grosjean, Robert I. Davis, and Benjamin Lesage. Progress on static probabilistic timing analysis for systems with random cache replacement policies. In *Proceedings Real-Time Scheduling Open Problems Seminar (RTSOPS'14)*, 2014.

4 Sebastian Altmeyer and Robert I. Davis. On the correctness, optimality and precision of static probabilistic timing analysis. Technical report, Technical Report YCS-2013-487, University of York, 2014. Available from http://www.cs.york.ac.uk/ftpdir/reports/2013/YCS/487/YCS-2013-487.pdf.

5 Guillem Bernat, Antoine Colin, and Stefan M. Petters. WCET analysis of probabilistic hard real-time system. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), Austin, Texas, USA, December 3-5, 2002*.

6 F.J. Cazorla, E. Quinones, T. Vardanega, L. Cucu-Grosjean, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim. PROARTIS: probabilistically analysable real-time systems. *ACM Transactions on Embedded Computing Systems, Special issue on Probabilistic Embedded Computing*, 2012.

7 F.J. Cazorla, T. Vardanega, E. Quinones, and Jaume Abella. Upper-bounding program execution time with extreme value theory. In *International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2013.

8 L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F.J. Cazorla. Measurement-based probabilistic timing analysis for multi-path programs. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

9 Robert I. Davis, Luca Santinelli, Sebastian Altmeyer, Claire Maiza, and Liliana Cucu-Grosjean. Analysis of probabilistic cache related pre-emption delays. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2013.

10 William Feller. *An Introduction to Probability Theory and Its Applications*. John Willer and Sons, 1996.

11 J. Jalle, L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. Bus designs for time-probabilistic multicore processors. In *Design, Automation and Test in Europe Conference (DATE)*, 2014.

12 L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. A cache design for probabilistically analysable real-time systems. In *Design, Automation and Test in Europe Conference (DATE)*, 2013.

13 L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. Multi-level unified caches for probabilistically time analysable real-time systems. In *IEEE Real-Time Systems Symposium (RTSS)*, 2013.

14 Leonidas Kosmidis, Jaume Abella, Franck Wartel, Eduardo Quinones, Antoine Colin, and Francisco J. Cazorla. PUB: Path upper-bounding for measurement-based probabilistic timing analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2014.

15 Leonidas Kosmidis, Eduardo Quiñones, Jaume Abella, Tullio Vardanega, Ian Broster, and Francisco J. Cazorla. Measurement-based probabilistic timing analysis and its impact on processor architecture. In *17th Euromicro Conference on Digital System Design, DSD*, 2014.

16 Eduardo Quinones, Emery D. Berger, Guillem Bernat, and Francisco J. Cazorla. Using Randomized Caches in Probabilistic Real-Time Systems. In *22nd Euromicro Conference on Real-Time Systems (ECRTS)*, 2009.

17 Jan Reineke. Randomized caches considered harmful in hard real-time systems. *Leibniz Transactions on Embedded Systems*, 1(1), 2014.

18 Mladen Slijepcevic, Leonidas Kosmidis, Jaume Abella, Eduardo Quinones, and Francisco J. Cazorla. DTM: Degraded test mode for fault-aware probabilistic timing analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2013.

19 R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G.Bernat, C. Ferdinand, R.Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, G. Staschulat, and P. Stenström. The worst-case execution time problem: overview of methods and survey of tools. *Trans. on Embedded Computing Systems*, 7(3):1–53, 2008.

20 Shuchang Zhou. An efficient simulation algorithm for cache of random replacement policy. In *the 7th IFIP international conference on Network and parallel computing (NPC2010)*, pages 144–154, 2010.