Towards Limiting the Impact of Timing Anomalies in Complex Real-Time Processors

Pedro Benedicte
Universitat Politecnica de Catalunya
Barcelona Supercomputing Center
pbenedic@bsc.es

Jaume Abella
Barcelona Supercomputing Center (BSC)
jaume.abella@bsc.es

Carles Hernandez
BSC
carles.hernandez@bsc.es

Enrico Mezzetti
BSC
enrico.mezzetti@bsc.es

Francisco J. Cazorla
BSC and IIIA-CSIC
francisco.cazorla@bsc.es

ABSTRACT

Timing verification of embedded critical real-time systems is hindered by complex designs. Timing anomalies, deeply analyzed in static timing analysis, require specific solutions to bound their impact. For the first time, we study the concept and impact of timing anomalies in measurement-based timing analysis, the most used in industry, showing that they require to be considered and handled differently. In addition, we analyze anomalies in the context of Measurement-Based Probabilistic Timing Analysis, which simplifies quantifying their impact.

CCS CONCEPTS

• **Computer systems organization** → *Embedded systems*; *Real-time system specification*; *Real-time system architecture*;

KEYWORDS

Timing Ananomalies, WCET, embedded critical systems

ACM Reference Format:
Pedro Benedicte, Jaume Abella, Carles Hernandez, Enrico Mezzetti, and Francisco J. Cazorla. 2019. Towards Limiting the Impact of Timing Anomalies in Complex Real-Time Processors. In *Proceedings of ACM Asia South Pacific Design Automation Conference (ASP-DAC 2019),* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3287624.3287655

1 INTRODUCTION

Driven by the current trend towards autonomous systems (e.g. in cars, satellites, drones, etc.), the demand for new software-based, performance-hungry, complex functionalities is steadily increasing. For example, autonomous driving cars are expected to provide the same performance as 25 high-end multicore desktops [4]. As a result, embedded critical real-time systems have rapidly evolved from simple micro-controllers to high-performance complex processors such as those found, for instance, in NVIDIA DrivePX [12]. The other side of the coin is that providing evidence on software's correct timing behavior, as demanded by safety standards, on those complex processors is a much more onerous and challenging task, especially when considering the derivation of bounds to applications' Worst-case Execution Time (WCET). As a major source of complexity in timing verification, modern processors are generally prone to *timing anomalies* [11, 16, 19], a well-known phenomenon causing that local worst-cases (e.g. an access *A* incurring a cache miss) are not guaranteed to lead to the global worst-case (i.e. a cache hit on *A* results in a longer execution time).

Despite the obstacles to their analysis are still far from being solved, complex processors have been embraced by the embedded real-time market as their de facto baseline hardware computing solution. In the automotive domain, where car manufacturers have started facing the performance requirements brought by software implementing high-end autonomous functions, complex processors are not going to be relegated to the execution of low-integrity functionalities. On the contrary, they will be executing software affecting safety goals with ASIL C/D (ASIL D is the highest integrity level in ISO-26262 in the automotive domain). An analogous trend is observed in other embedded real-time critical domains, such as avionics and space. Hence, dealing with timing anomalies is mandatory to enable and support the analysis of complex computing platforms. This applies to both Static Timing Analysis (STA), for which timing anomalies have been deeply analyzed [16][19], and Measurement-Based Timing Analysis (MBTA) [18], for which instead timing anomalies have been totally neglected so far.

In this work we promote a change of perspective, analyzing timing anomalies in the context of MBTA. This is particularly relevant for the automotive domain, where MBTA is the most widely used timing analysis technique. In particular, our contributions are:

(1) We analyze timing anomalies in MBTA, concluding that anomalies cannot be handled as they are for STA. In particular, we observe that the challenges they bring to MBTA are not related to analysis (i.e. model) assumptions, but rather to the generic difficulty for the user to exercise sufficient control of all factors with bearing on timing during program runs in the analysis phase. Based on this observation, we tackle, for the first time, the challenge posed by timing anomalies on MBTA of high-performance processor designs, by analyzing their impact on WCET estimates.
(2) With emphasis on the probabilistic variant of MBTA, called MBPTA [1], we show how the use of those hardware features that can potentially cause timing anomalies can still be safely enabled in critical real-time systems. To that end, we leverage the use of time-randomized hardware designs

ASP-DAC 2019, Jan. 2019, Tokyo, Japan

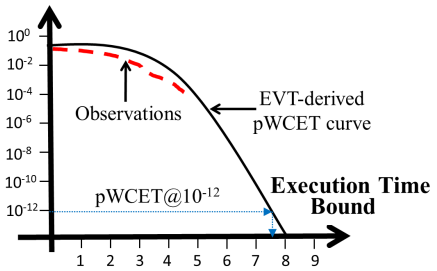

Figure 1: Example of a pWCET curve

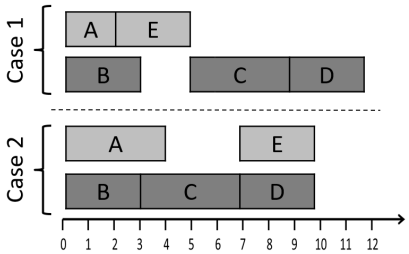

Figure 2: Example of timing anomaly

(e.g. random placement and replacement caches, random arbitration in shared resources) that make the occurrence of timing anomalies probabilistic.

(3) We instantiate the aforementioned key principles on a particular example of a type of timing anomaly as it may happen on a commercially available time-randomized processor design.

2 BACKGROUND ON TIMING ANALYSIS

2.1 Measurement-Based Timing Analysis

Across several real-time domains, industry mostly resorts to measurement-based (deterministic) timing analysis (MBDTA or MBTA) as the main strategy for timing analysis. The confidence on the derived WCET estimates strongly relates to the knowledge of the software and hardware had by the end user, as well as his/her ability to exercise sufficient control on the experiments, to generate suitable input vectors to trigger the WCET during the analysis phase. However, such degree of control can only be reasonably exercised on simple processor designs. On more complex hardware designs, attaching a degree of confidence to the WCET is much more difficult: the WCET estimate is often calculated as the high watermark plus some safety margin to account for the unknown, whose confidence is hard – if at all possible – to assess.

As the complexity of hardware and software increases, the degree of control that end users need to exercise in the estimation of the WCET becomes less obvious to attain, and uncertainty to size safety margins grows. MBPTA has been specifically designed to addresses these challenges. The use of MBPTA is also fueled by the increased execution time variability that programs suffer when run on complex hardware. In fact, this variability results in execution time distributions with arbitrary variance and shape, motivating the use of statistical techniques to derive time bounds.

MBPTA [1] relies on hardware/software platforms with some specific timing properties that allow reducing the degree of control that end users need to exercise to collect execution times at analysis. To that end, MBPTA postulates that the hardware sources of execution time variation are conveniently randomized and/or upper-bounded, often with some hardware support, so that their behavior naturally emerges in the measurements collected during the test campaign (i.e. system analysis/design phase). Randomization must be preserved during system operation phase for representativeness reasons.

MBPTA, which has already been successfully applied to industrial case studies [3], uses extreme value theory (EVT) [9] to produce an execution time distribution that upper-bounds the execution time of the task under analysis during operation based on a sample of execution time measurements. In the so-called MBPTA-compliant processor designs, each execution time occurs with a probability, and the underlying (analyzed) distribution upper-bounds that exhibited during operation. For each particular execution time value, MBPTA delivers the probability above which this same value cannot be exceeded by the execution time of a single program run. This distribution is referred to as probabilistic WCET (pWCET for short), see Figure 1. For instance, an exceedance probability of 10^{-15} per run indicates that the particular pWCET estimate cannot be exceeded with a probability above 10^{-15}. Whether the pWCET can be exceeded with a probability below 10^{-15} or it cannot be exceeded at all remains unknown. Such remaining (exceedance) probability relates to the residual risk in software verification, which is intrinsic to the software verification process, as no method allows proving with absolute confidence that software will not fail during operation [6].

2.2 Timing Anomalies from STA Standpoint

A high-level definition of timing anomalies, from the perspective of timing analysis, is given in [16] as those cases where a local worst-case does not lead to the global worst-case. An illustrative timing anomaly is shown in Figure 2 where instructions in each row execute serially due to data dependencies (i.e. E consumes some data produced by A), and C and E use the same resource. We see that by experiencing a longer latency for A (local effect) the overall execution time decreases (global effect).

For STA, timing anomalies jeopardize the formal reliability of the approach. STA is in fact forced to resort to some form of abstraction to be able to model all possible inputs and hardware states, see top part of Figure 3. Abstractions in turn introduce nondeterminism in the model, where an abstract state can have multiple successor states. Since modeling all possible transitions between abstract states rapidly becomes computationally intractable, STA approaches typically discard those states that are unlikely to lead to the global worst-case behavior. In particular, the underlying assumption in STA approaches is that timing can be safely analyzed at the level of single execution blocks as a function of an initial state and a given input, so that local worst-cases transitions are always assumed to lead to the global worst-case timing. Timing anomalies clearly spoil this assumption, forcing STA to take the appropriate countermeasures.

A relevant STA timing anomaly classification looks into the effects on timing on the analysis scope, distinguishing between bounded and unboundable timing anomalies [19]. Anomalies are classified as *k-bounded* if their effect can be factored in by adding a conservative (possibly overly-pessimistic) constant to the computed WCET bound. Instead, to account for unboundable timing anomalies, leading to the so-called *domino effect*, STA is forced to consider all possible states and transitions, which is evidently untenable. As shown in Figure 3, STA focus is on determining whether anomalies can happen on a given platform, and also, on how they can be efficiently bounded.

3 TIMING ANOMALIES IN MBTA

MBTA approaches have not considered timing anomalies as a concern so far, since the usual standpoint from which practitioners are used to consider timing anomalies is specifically that of STA. In our opinion, instead, some form of timing anomalies are to be considered relevant even in the scope of MBTA.

It is worth noting that, in the context of end-to-end program analysis, timing anomalies can happen due to hardware resources having variable latencies. Considering a sequence of instructions with fixed input data, accessing a fixed set of hardware resources, the sequence can lead to different execution times only in response to different initial processor states, inducing a variable response time (jitter) of a subset of the involved hardware components. While it is true that anomalies do not explicitly break any MBTA assumption, their potentially erratic impact on timing may jeopardize the fundamental conditions for measurement-based methods. For this reason, anomalies in MBTA need to be understood from the perspective of the *representativeness* of analysis-time observations.

Representativeness covers whether the measurements performed in the test campaigns during the *analysis phase* capture the worst-case relevant effects that can arise during system operation. Timing anomalies have the potential to distort the correspondence between analysis and operation conditions.

Observation 1. *With MBTA, dealing with timing anomalies builds on the ability to argue whether they have been triggered or not when running a program; whether they can actually occur during system operation; and whether their observed impact during analysis tests upperbounds the impact they may incur during operation.*

Figure 3, compares different ways in which timing anomalies – referred to as TA in the figure – can be attacked under the STA and MBTA paradigms. Similarly to STA, analyzing a system that can be proved to be timing-anomaly free would be the most favorable scenario. Unfortunately, assessing the lack of timing anomalies is not realistically affordable in the general case, and can only be possibly achieved with highly-specialized hardware designs [10].

Interestingly, modeling timing anomalies is not a challenge for MBTA. The challenge instead is to trigger anomalies during analysis tests and to size their impact, as they can manifest during operation. Theoretically, users are required to design experiments that capture the potential increase in execution time they entail.

Dealing with anomalies poses a challenge analogous to the one faced by end users to trigger specific low-level hardware interactions, since the user lacks explicit control knobs over them. To conclude the parallelism with STA, from the MBTA perspective,

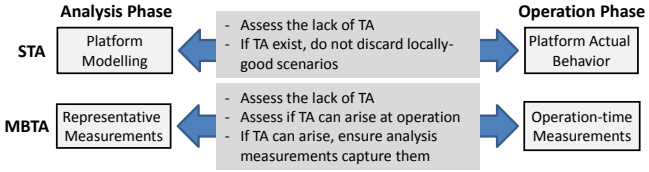

Figure 3: STA and MBTA management of Timing Anomalies

a relevant classification of timing anomalies does not focus on whether an anomaly has k-bounded versus domino effects but rather on *controllability*, which refers to whether or not an MBTA user is able to trigger them in a controlled way. Next, we look into timing anomalies from the perspective of MBTA, in the specific context of MBPTA-compliant hardware.

4 TIMING ANOMALIES IN MBPTA

Timing anomalies in time-deterministic processors may potentially occur systematically. This is not an issue for the way STA deals with anomalies as the relevant aspect is whether an anomaly can either happen (and it is always assumed to) or not. Under MBTA, instead, the frequency at which an anomaly occur has a variable impact on the execution time, which in turn could challenge the reliability of the WCET estimate. This concern is partially cured under MBPTA. In fact, for MBPTA, and in particular time-randomized hardware, certain timing events exhibit a random behavior, which can potentially break systematic patterns and allow building probabilistic arguments on the appearance of timing anomalies.

Observation 2. *Time-randomized processors (e.g., implementing caches [8] and buses [7] with time randomization properties), used in combination with MBPTA, trigger a number of random timing events that potentially break systematic timing behavior.*

As a result, the occurrence of some timing anomalies becomes inherently probabilistic. Moreover, the accumulation of timing anomalies occurs with decreasing probabilities. In fact, a given event potentially triggering an anomaly, necessarily repeats (chain of occurrence) with decreasing probabilities so that execution time variations end up being of lower magnitude than those produced by randomized hardware resources themselves. We will consolidate this argument while reasoning on a practical example in Section 5.

However, MBPTA and randomization do not prevent that some other timing anomalies may be systematically triggered, because they depend on non-time-randomized sources of execution time variation. Under some conditions, however, also these anomalies can be conveniently controlled.

4.1 Taxonomy of Timing Anomalies in MBPTA

In MBPTA-compliant processors, some individual sources of jitter (i.e. resources with variable latency) are controlled in a way that they are upper-bounded, whereas others – those with highest impact in WCET estimates – are time-randomized (e.g., cache memories, bus arbitration, etc.). These different sources of jitter have been analyzed in a LEON-like processor, currently assessed for future space missions, as part of the PROXIMA Project [15].

Observation 3. *Constant execution time events cannot trigger any timing anomaly (but can propagate them).*

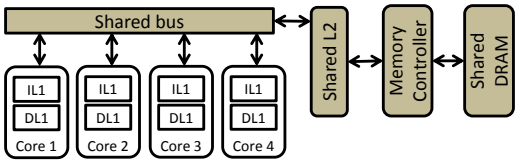

Figure 4: Processor architecture considered in this analysis. IL1, DL1 and L2 stands for first-level instruction and data caches; and L2 cache.

Those resources exhibiting a constant timing behavior exhibit the same behavior at analysis and operation, regardless of any execution condition. They cannot trigger any timing anomaly but cannot compensate nor prevent the effects of other anomalies potentially triggered.

Observation 4. *Random events whose execution time distribution does not change between analysis and operation, will exhibit probabilistically boundable timing anomalies.*

If the response time distribution of the resource remains unaltered between analysis and operation, then the occurrence and impact of timing anomalies can be related to the probability distribution observed at analysis. MBPTA is still responsible for guaranteeing representativeness of the observations.

From the observations above, we introduce a taxonomy of timing-anomaly scenarios for randomized architectures. Each hardware resource can be characterized as potentially triggering:

(1) **No timing anomalies.** Fixed-latency timing events exhibit the same behavior at analysis and during operation. Hence, they trigger no timing anomaly. This classification applies to deterministic resources if the deterministic upper bound is enforced (by hardware means) even during operation.
(2) **Probabilistically-controlled timing anomalies.** Some timing anomalies may be triggered by random events. Thus, they will be observed with a given probability in analysis runs. A rigorous MBPTA application [1] will guarantee that their timing impact is properly captured in the execution time measurements used to derive the pWCET estimate, *as long as their probabilistic behavior is preserved from analysis time to operation.*
(3) **Potentially uncontrolled timing anomalies.** Some timing anomalies may be triggered due to ***latent*** systematic effects, or due to probabilistic events whose probability distribution differs between analysis time and operation. Thus, their timing impact may not be properly captured in the execution time measurements used to feed MBPTA. As a result, the end user needs to account for their effect without explicit support from the hardware or the timing analysis tool, which is a cumbersome task. In general, end users lack the degree of control needed to determine the impact of those anomalies and whether their impact has been properly accounted for, thus decreasing the quality of WCET estimates.

5 DEALING WITH TIMING ANOMALIES

To illustrate how to deal with timing anomalies on an MBPTA-compliant hardware design, we use as example the enhanced LEON3 multicore processor design implemented in an FPGA [2] and commercially available through Cobham Gaisler. The main timing characteristics of such processor are illustrated in Figure 4 and further discussed in Section 6. We identify a number of sources of jitter – and so potential sources to trigger timing anomalies – in the processor design: `FDIV`/`FSQRT` operations, cache memories and randomly arbitrated resources. In the next sections we review the potential timing anomalies that could be triggered in such design, and discuss how jittery resources need to be controlled to avoid harmful (i.e., potentially uncontrolled) timing anomalies.

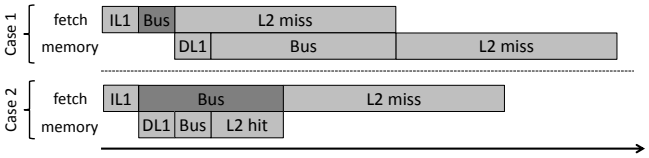

Figure 5: Priority inversion causing a timing anomaly.

5.1 Upperbounding Variable-Latency Units

In our reference processor, `FDIV` and `FSQRT` incur jitter depending on the values operated. Following existing solutions for STA, we force these operations to take their worst latency [13], removing the jitter with minimum impact on average performance.

5.2 Priority Inversion in Cache Access

The requests sent to each cache (IL1, DL1 and L2) are served in arrival order. All requests to IL1 (DL1) are sent from the same pipeline stage, fetch (execution), and hence, the request arrival order matches the program order.

Let us assume instructions i_x and i_y are executed in program order, i.e., $x < y$, then all the requests these instructions can generate to IL1 ($i_{x,IL1}$, $i_{y,IL1}$) and DL1 ($d_{x,DL1}$, $d_{y,DL1}$) will be served in program order. However, this does not apply to L2, since the requests sent to L2 can be generated by instructions in two different stages (fetch and execute), which can generate inversion i.e. the instruction request of the second instruction $i_{y,L2}$ is served by the L2 before the data request of the first instruction $d_{x,L2}$. In case both requests to L2 are generated in the same cycle, $d_{x,L2}$ is prioritized.

When a memory request misses in a private L1, it needs to get access to the bus shared across the 4 cores to reach the L2 cache. Several MBPTA-compliant time-composable arbitration policies have been proposed [7], including round-robin policy. With this policy, we guarantee that, in a 4-core processor, each core will be able to access the shared L2 cache 1 out of every 4 time slots. Hence, the worst latency to reach the L2 cache is 4 time slots minus 1 cycle (if requests can only be sent the first cycle of the slot).

Given this bus behavior, we can see in Figure 5 an example of a timing anomaly that occurs due to priority inversion. MBPTA-compliant time-composable arbitration policies impose accounting for worst-case contention during analysis, regardless of whether the arbitration policy is deterministic (e.g. round robin) or randomized (e.g. random permutations [7]). Either case, this makes that the delay experienced to access the bus may be typically high during analysis (case 2). Then, during operation, for efficiency reasons (e.g. average performance, power, etc.), worst-case contention is not

enforced and requests experience *actual* contention, which will be typically lower, thus increasing the chances of experiencing timing anomalies (case 1).

In general, depending on the observed behaviours at analysis time (AT) and operation time (OT), we have two possible scenarios: (a) Priority inversion happens systematically at AT, or with the same (or higher) probability at AT than during OT. And (b) Priority inversion does not happen at AT, or occurs with lower probability than during OT. Scenario (a) is covered by MBPTA since execution time measurements during analysis account for worse conditions than those during operation [1]. In scenario (b), however, timing anomalies have not been accounted for properly, typically because their occurrence has been prevented (or heavily put down) as a side-effect of the analysis configurations and setups.

In scenario (b), the potential impact on timing that events not captured at AT can have later during OT needs to be understood and gauged. We do so by leveraging on the probabilistic nature of the enhanced LEON3 MBPTA-compliant platform, where random placement and replacement caches are used. In the particular case of the cache-access-inversion timing anomaly, we can reason on the probability of occurrence of its root causes: besides the bus delay, a cache miss in L2 – which has a fixed latency – is causing the eviction of a cache line that is accessed by an instruction sufficiently close in the pipeline. With random caches [8] the probability of an L2 miss to evict the useful cache line referenced by the DL1 miss is bounded by the number of cache sets (S_{L2}) and ways (W_{L2}) in L2, as shown in Equation 1 (P_{evict}). To upperbound the overall timing effect, we use the number of L2 misses potentially triggering the anomaly, which can be in general obtained by exploiting accurate hardware counters (e.g., `L2_MISS_COUNT`) as provided by the standard debug support unit (DSU). The total L2 miss count is a conservative overapproximation as it includes L2 misses that can never cause an anomaly by construction, or that were already triggering timing anomalies at AT. Equation 1 derives an upperbound to overall effects of cache-access-inversion anomaly (Δ) on a given program by collecting the cost of each potential anomaly, which is in turn bounded by the cost of the additional L2 cache miss.

$$\Delta \leq \underbrace{\frac{1}{S_{L2} \cdot W_{L2}}}_{P_{evict}} \times \texttt{L2\_MISS\_COUNT} \times \texttt{L2\_MISS\_LATENCY} \quad (1)$$

In the particular case of our target processor, the L2 cache is a 512KB 4-way 32B/line cache [2]. Moreover, since the L2 cache is partitioned across the 4 cores, each one receives exactly 1 cache way, which means that $P_{evict} \leq \frac{1}{4096 \cdot 1} \approx 0.000244$. The cost of an additional L2 cache miss is 28 cycles in the target platform.

The probability of occurrence of the anomaly (per-se objectively low), rapidly decreases when we consider for example the repeated execution of the same set of instructions within a loop since accesses in the following iterations will likely hit in IL1 and DL1. Looking at the specific anomaly, we also observe that both L2 accesses, instruction and data ones jointly contributing to the anomaly, must be initiated by instructions that can actually suffer from some form of inversion in the pipeline, which is only 7-stages in the LEON. As a result, the effect of an anomaly cannot propagate outside of its pipeline window. Borrowing the terminology used in the scope of static analysis, this anomaly can be classified as *boundable*.

5.3 Priority Inversion due to Initial Cache State

A similar priority inversion could occur due to the initial cache state. While worst-case initial state is enforced during analysis (e.g. empty write-through caches and useless dirty contents in write-back ones), some useful contents may be stored in cache during operation so that some accesses become hits. If those hits lead to timing anomalies, they do it with the same (very low) probability described before. Moreover, by turning misses into hits, execution time is lower and hence, less likely to be close to the WCET.

Equation 1 provides a parametric bound that depends on the number of misses generated by a program. In practice, however, the fact that the probability of occurrence of such anomalies is low and rapidly decreasing (when considering sequences of instructions) makes it arguable whether and to what extent they do actually pose a threat to the trustworthiness of pWCET bounds.

5.4 Managing Arbitration Effects on Requests

As shown in the previous section, the bus and memory controller arbitration policies can also impact timing anomalies. In particular, they determine the latency to serve L2 and memory requests and, indirectly, the order in which requests are served, which can generate a timing anomaly. However, these components process requests in order in the LEON processor, so they cannot produce further anomalies by themselves.

6 QUANTITATIVE ASSESSMENT

We build upon the FPGA implementation of the time-randomized in-order, pipelined LEON3 processor [2]. In particular, the processor features 16KB 4-way 16B/line DL1 and 32B/line IL1 caches. The DL1 is write-through and IL1 read-only. The unified L2 cache is 512KB, 4-way with 32B lines. Latencies are 1 cycle for DL1/IL1 hits, 7 cycles for L2 hits and 28 cycles for memory accesses. DL1 and IL1 implement random modulo [5] and random replacement [8]. L2 cache implements random placement and replacement [8]. For our evaluation we use the EEMBC automotive benchmark suite, which models some real-time automotive functionalities [14]. We obtain pWCET estimates with MBPTA using 1,000 runs per benchmark, which prove to be enough for MBPTA to converge [1].

We have studied the impact that execution time distributions have on pWCET estimates. For that purpose we compute pWCET estimates at an exceedance threshold of 10^{-12} per run with MBPTA [1], although the same observations hold for other values. First, we have computed those pWCET estimates and computed confidence intervals for a significance level of $\alpha = 0.05$, thus meaning that the true pWCET value should fall in that interval with 95% confidence.

Since it is virtually impossible determining whether timing anomalies occurred as well as preventing or enforcing them in a real processor, we perform a statistical assessment of the potential impact of timing anomalies on execution time distributions. In particular, we perform the following experiment: from each execution time trace we produce an additional execution time trace by decreasing each execution time by the expected upper-bound number of timing anomalies (IL1 misses divided by $W_{L2} \cdot S_{L2} = 4,096$) multiplied by

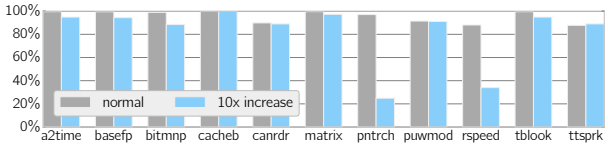

Figure 6: Normalized confidence interval overlap w/wo TA.

the upper bound timing impact of a timing anomaly (28 cycles). The number of timing anomalies is obtained as the quotient of dividing the number of misses by 4,096, and it is increased by 1 with a probability matching the remainder divided by 4,096. Then we compare those execution distributions against the original ones with the Kolmogorov-Smirnov two-sample identical distribution test with a significance level of $\alpha = 0.05$. This test, which returns a P-value in the range [0, 1], indicates that the identical distribution hypothesis cannot be rejected when P-value > 0.05.

In order to compare pWCET estimates with and without timing anomalies, we do so in relative terms, by computing the ratio between the difference of both pWCET estimates and the actual pWCET estimate obtained without timing anomalies. Differences are completely negligible. The highest differences correspond to `a2time` and `tblook` (0.007% and 0.004% respectively), with most of them below 0.001%. We have further compared the confidence intervals to assess how much they overlap. This comparison is depicted in Figure 6 (left bar in each pair), where we can see that the overlap is huge, being always above 88% (95.5% on average) despite having very narrow confidence intervals in some cases. Therefore, pWCET estimates and execution time distributions cannot be proven different. Hence, the effect of timing anomalies on the overall timing is much lower than that already incurred by the variability of finite random samples.

We have repeated the very same analysis in the case where the upper-bounded impact of timing anomalies is further increased by a 10X factor. The difference between pWCET estimates with and without timing anomalies (with an impact increased by 10X) is very small. In particular, the difference is always below 0.1% and below 0.01% in most of the cases. In terms of confidence intervals, Figure 6 (right bar in each pair) shows that they overlap in all cases, so we cannot reject the hypothesis that both execution time distributions lead to the same pWCET estimate. However, we see that the relative overlap in the cases of `pntrch` and `rspeed` is comparatively low (25% and 34% respectively). This effect is produced by the extremely narrow confidence intervals (186 and 79 cycles respectively), so, in practice, pWCET estimates just differ by few tens of cycles.

7 RELATED WORK

Different timing analysis strands deal with timing anomalies in the context of safety-related real-time systems [11, 16, 19]. Among those works, an interesting classification is provided in [19], where processors are broken down into several categories depending on whether they are free of timing anomalies (ideal case), whether they have timing anomalies whose impact can be upper bounded with limited pessimism (good case), or whether they can trigger domino effects that might lead to high execution time impact (challenging case). Existing MBPTA-compliant processors are deemed as free of timing anomalies or have left their consideration for future work. This includes single [8] and multi-core [7, 17].

8 CONCLUSIONS

Timing anomalies can affect the quality of WCET estimates in the increasingly complex hardware used in critical real-time systems. In this paper we provide, for the first time, a definition of timing anomaly for MBTA that differs from that used in STA. We have also made an analysis of how to design hardware and collect measurements to limit – or even remove – the impact of certain timing anomalies for MB(P)TA. With an MBPTA-compliant RTL processor implemented in a FPGA, we assess the influence of timing anomalies on WCET estimates and show that their impact falls within the range of noise w.r.t. the own execution time variability.

ACKNOWLEDGMENT

This work has been partially funded by the Spanish Ministry of Economy (TIN2015-65316-P) and the European Research Council with Horizon 2020 (grant agreement No. 772773). Jaume Abella has been partially supported by Ramon y Cajal grant RYC-2013-14717. Enrico Mezzetti is partially funded by Juan de la Cierva-Incorporación grant IJCI-2016-27396. Pedro Benedicte is funded by the Spanish Ministry of Education under grant FPU15/01394.

REFERENCES

[1] J. Abella et al. 2017. Measurement-Based Worst-Case Execution Time Estimation Using the Coefficient of Variation. *ACM TODAES* (2017).
[2] C. Hernandez et al. 2017. Design and Implementation of a Time Predictable Processor: Evaluation With a Space Case Study. In *ECRTS*.
[3] M. Fernandez et al. 2017. Probabilistic timing analysis on time-randomized platforms for the space domain. In *DATE*.
[4] V. Giorgetta. 2016. Invited talk: Challenges for the Automotive platform of the future. In *Workshop PLATFORMS*.
[5] C. Hernandez et al. 2016. Random Modulo: a New Processor Cache Design for Real-Time Critical Systems. In *DAC*.
[6] International Organization for Standardization. 2009. *ISO/DIS 26262. Road Vehicles – Functional Safety*.
[7] J. Jalle et al. 2014. Bus Designs for Time-Probabilistic Multicore Processors. In *DATE*.
[8] L. Kosmidis et al. 2013. A Cache Design for Probabilistically Analysable Real-time Systems. In *DATE*.
[9] S. Kotz and S. Nadarajah. 2000. *Extreme value distributions: theory and applications*. World Scientific.
[10] S. Law and I. Bate. 2016. Achieving Appropriate Test Coverage for Reliable Measurement-Based Timing Analysis. In *ECRTS*.
[11] T. Lundqvist and P. Stenstrom. 1999. Timing anomalies in dynamically scheduled microprocessors. In *RTSS*.
[12] NVIDIA. 2018. NVIDIA DRIVE PX. Scalable supercomputer for autonomous driving. www.nvidia.com/object/drive-px.html.
[13] M. Paolieri et al. 2009. Hardware Support for WCET Analysis of Hard Real-Time Multicore Systems. In *ISCA*.
[14] J. Poovey. 2007. *Characterization of the EEMBC Benchmark Suite*. North Carolina State University.
[15] PROXIMA. 2014. Probabilistic real-time control of mixed-criticality multicore and manycore systems. http://www.proxima-project.eu/
[16] J. Reineke et al. 2006. A Definition and Classification of Timing Anomalies. In *WCET Workshop*.
[17] F. Wartel et al. 2015. Timing Analysis of an Avionics Case Study on Complex Hardware/Software Platforms. In *DATE*.
[18] I. Wenzel et al. 2005. Measurement-Based Worst-Case Execution Time Analysis. In *SEUS Workshop*.
[19] R. Wilhelm et al. 2009. Memory Hierarchies, Pipelines, and Buses for Future Architectures in Time-Critical Embedded Systems. *IEEE TCAD* (2009).