Probabilistic Timing Analysis on Time-Randomized Platforms for the Space Domain

Mikel Fernandez§, David Morales¹, Leonidas Kosmidis³, Alen Bardizbanyan*, Ian Broster³, Carles Hernandez³, Eduardo Quinones¹, Jaume Abella¹, Francisco Cazorla¹, Paulo Machado*, Luca Fossati³

Abstract—Timing Verification is a fundamental step in real-time embedded systems, with measurement-based timing analysis (MBTA) being the most common approach used to that end. We present a Space case study on a real platform that has been modified to support a probabilistic variant of MBTA called MBPTA. Our platform provides the properties required by MBPTA with the predicted WCET estimates with MBPTA being competitive to those with current MBTA practice while providing more solid evidence on their correctness for certification.

I. INTRODUCTION

The use of increasingly complex hardware (e.g. processors comprising caches) across all real-time domains challenges current timing analysis approaches and complicates deriving reliable and tight WCET estimates [1]. This confronts industry with a dilemma of enjoying high computing performance, which in the space domain allows more autonomous and ambitious space missions, increasing competitive edge; but at the cost of incurring higher risk of delivering less reliable WCET estimates, creating potential safety risks.

We focus on MBTA that has a considerable presence in current industrial practice in domains such as automotive and space due to its good benefit/cost ratio [11]. In particular we focus on Measurement-Based Probabilistic Timing Analysis (MBPTA), a variant of MBTA. MBPTA [3][9][10] derives a probabilistic WCET (or pWCET) distribution that describes the highest probability (e.g. $10^{-15}$) at which one instance of the program may exceed the corresponding execution time bound. This probability is set in accordance with the corresponding safety standard [4]. MBPTA aims at reducing the control to the end user has to exercise on the conditions of the experiments performed during the analysis phase so as to ensure that the worst-case execution conditions that can occur during system operation are properly covered. Exercising such control incurs massive effort for the original MBTA, especially in the presence of cache memories.

MBPTA combines probabilistic timing analysis and the injection of randomization in the timing behavior of certain hardware resources. On the one hand, randomization ensures that, if enough runs are performed, the impact of all platform events with a relevant probability are captured in the test measurements. On the other hand, probabilistic analysis captures the impact that events observed in different test runs appear in the same run at operation (with the corresponding increase in execution time) without requiring the end user to construct a test case forcing all those events to arise in a single run.

MBPTA academic literature covers, among others, foundational aspects, probabilistic analysis and the impact of randomization. On the industrial side, some works assess MBPTA with avionics case studies [9], [10] on simulation environments, limiting the evidence on their applicability to real industrial setups. This paper helps covering this gap by implementing a version of the LEON3 [8] processor which was modified so it is MBPTA compliant. We perform the timing analysis evaluation with a commercial timing analysis tool [7] that has been properly enhanced to support probabilistic analysis.

II. HARDWARE-RANDOMIZED PLATFORM

Background: MBPTA aims at providing guarantees that the execution time observations at analysis time capture application’s worst-case behavior during operation. MBPTA controls the jitter caused by hardware resources. Jitterless resources are naturally compliant with MBPTA since their impact in execution time is constant, so analysis-time measurements already capture their behavior during operation. This is the case, for instance, of the integer arithmetic unit given that all types of integer operation have fixed latency. For jitter resources MBPTA applies two solutions: forcing resources to work on their worst latency at analysis time or randomizing their timing behavior. The former solution, makes that the worst impact that the resource may have on execution time arises at analysis, hence making them MBPTA compliant. This is the case of some FPU operations, whose latency depend on the values operated. Meanwhile, randomization makes latencies of a resource to have a probabilistic behavior and thus, allows MBPTA to reliably upper-bound its impact by collecting enough number of measurements so that its worst-case behavior can be predicted with probabilistic means. This is the case of time-randomized caches [6].

Platform: We focus on a 4-core LEON3 [8] with 7-stage pipelined cores comprising first level instruction (IL1) and data (DL1) caches, with the DL1 implementing write-through no write allocate policies; and a bus that propagates DL1 and IL1 misses to the DRAM shared memory controller, see Figure 1. In our implementation IL1 and DL1 are 16KB 4-way set-associative caches. TLBs comprise 64 entries.

Our goal in this board is to control cache jitter and FPU jitter. To make the baseline platform MBPTA-compliant the following hardware modifications have been performed.

Cache Modifications. The memory layout of code/data determines the cache sets where they are placed with large impact on program’s execution time. Random placement releases the user from controlling the memory placement of programs in memory at analysis so that its effect upperbounds that during operation. With random placement, program’s data/code are mapped to random sets in each run, regardless of the memory positions in which data/code are allocated. Hence, simply making enough runs and applying MBPTA
This approach is used for its cost/benefit ratio but for this served for the non-randomized architecture [1]. The use of by an engineering factor (e.g. 50%) the highest value ob-

scale. We observe that the prediction, straight line, tightly upper-bounds the observed values. If both tests are passed, enabling MBPTA.

as these values are largely above 0.05 both tests are passed, enabling MBPTA.

level and the task frequency of execution.

Average performance. The observed average execution times for DET and RAND architectures (first two bars) show that there is not noticeable difference. Hence, our hardware changes did not affect the average performance of TVCA.

Conclusions. MBPTA results are competitive w.r.t. MBTA while providing more confidence than just increasing the high watermark execution time by an engineering factor to cover the uncertainty of factors like cache placement.

Fulfilling the i.i.d properties. MBPTA requires the execution times to have certain statistical properties to be independent and identically distributed. We test independence with the Ljung-Box test and a 5% significance level (a typical value for this type of tests). For identical distribution we use the two-sample Kolmogorov-Smirnov test also with a 5% significance level. This means that i.i.d. is rejected only if the value for any of the tests is lower than 0.05. We obtained 0.83 and 0.45 for each test respectively. As these values are largely above 0.05 both tests are passed, enabling MBPTA.

pWCET estimates. The X-axis in Figure 2 shows the execution time while the Y-axis shows probabilities in logarithmic scale. We observe that the prediction, straight line, tightly upper-bounds the observed values.

We compare our approach with an industrial practice based on MBTA applied to the baseline non-randomized, i.e deterministic (DET) platform. This approach consists in increasing by an engineering factor (e.g. 50%) the highest value observed for the non-randomized architecture [1]. The use of this approach is used for its cost/benefit ratio but for this

method to be used with sufficient confidence requires ensuring that the worst-case conditions have been exercised or closely approximated (e.g. the worst cache placement of objects). In Figure 3 we observe that pWCET estimates are within the same order of magnitude than the actual execution times, starting with an increase of 50% for a cutoff probability of 10^-6. Naturally, as we decrease the cutoff probability, i.e. the probability that once instance of the program overruns its budget, the pWCET estimate increases to reduce the overrun probability. The particular cutoff probability is to be chosen based on the applicable domain standard, the task criticality level and the task frequency of execution.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community’s FP7 [FP7/2007-2013] under the PROXIMA Project (www.proxima-project.eu), grant agreement no 611085. This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2015-65316-P and the HiPPEC Network of Excellence. Jaume Abella has been partially supported by the Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717. Carles Hernandez is jointly funded by the Spanish Ministry of Economy and Competitiveness and FEDER funds through grant TIN2014-60404-JIN.

REFERENCES