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

Mikel Fernandez[†], David Morales[†], Leonidas Kosmidis[†], Alen Bardizbanyan[*], Ian Broster[‡], Carles Hernandez[†], Eduardo Quinones[†], Jaume Abella[†], Francisco Cazorla[†,§], Paulo Machado[¶], Luca Fossati[¶]
[†]Barcelona Supercomputing Center (BSC) [‡]Rapita Systems LTD
[§]Spanish National Research Council (IIIA-CSIC) [¶]European Space Agency [*]Cobham Gaisler

***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.**

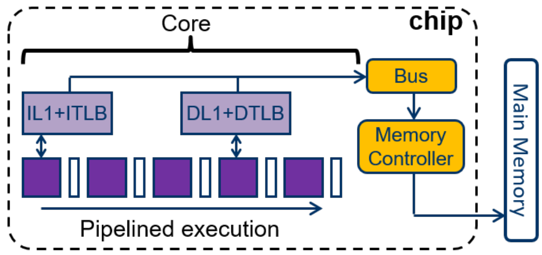

Fig. 1. Reference Architecture.

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 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 performing, to the best of our knowledge, the first evaluation of MBPTA with a Space case study executed on a FPGA platform 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 *jittery 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

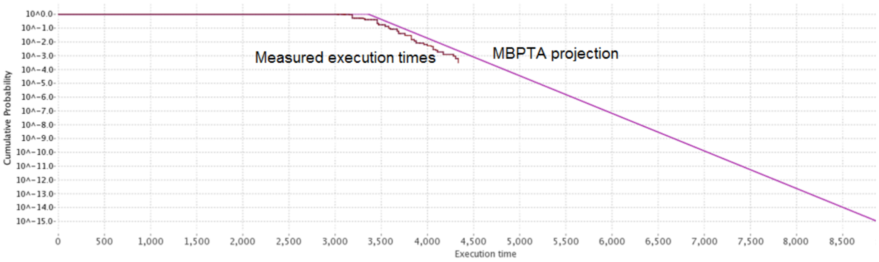

Fig. 2. pWCET estimates obtained with MBPTA for TVCA

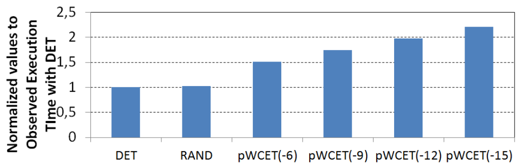

Fig. 3. MBPTA vs. DET observed execution times

gives the end user probabilistic guarantees that all potential mappings (and their impact on timing) are properly covered. For that purpose we implemented random replacement for IL1, DL1, ITLB and DTLB; and random modulo placement [5] for IL1 and DL1. In both cases we build on a pseudo-random number generator that has been shown to provide enough randomization for MBPTA [2].

FPU. FDIV/FSQRT operations take variable latency depending on the values operated. With MBTA this requires the user to control in the analysis test experiments the latency of those operations providing evidence that the distribution observed upperbounds that at operation. Since, this is infeasible in general, for MBPTA we changed the FPU so that during the analysis phase, both operations exhibit a fixed latency that matches their highest latency. The net result is that their jitterless timing behavior at analysis time upperbounds that during operation, releasing the end user from having to control the impact on execution time of the particular values operated.

III. Evaluation

We use a hard-real time Thrust Vector Control Application (TVCA) developed by the European Space Agency. The application comprises C code, automatically generated from a high-level model of the closed-loop control system, as it is the case in many critical real-time applications across domains such as in the avionics and automotive. TVCA, that runs unmodified on bare-metal, implements a fixed priority scheduler with 3 periodic tasks: sensor data acquisition, actuator control in x-axis and actuator control in y-axis. We execute TVCA 3,000 times to collect execution times which satisfied the convergence criteria defined in the MBPTA process. We flush caches, reset the FPGA and reload the executable across executions to have the same conditions for each execution. We also set a new seed for each experiment after the binary has been reloaded. Further we make per-path analysis taking the maximum across paths.

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.

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.

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 HiPEAC 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

[1] J. Abella et al. WCET analysis methods: Pitfalls and challenges on their trustworthiness. In *SIES*, 2015.

[2] I. Agirre et al. IEC-61508 SIL 3 compliant pseudo-random number generators for probabilistic timing analysis. In *DSD*, 2015.

[3] L. Cucu-Grosjean et al. Measurement-based probabilistic timing analysis for multi-path programs. In *ECRTS*, 2012.

[4] Z. Stephenson et al. Supporting industrial use of probabilistic timing analysis with explicit argumentation. In *INDIN*, 2013.

[5] C. Hernandez et al. Random modulo: a new processor cache design for real-time critical systems. In *DAC*, 2016.

[6] L. Kosmidis et al. A cache design for probabilistically analysable real-time systems. In *DATE*, 2013.

[7] Rapita Systems Ltd. Rapita verification suite. http://www.rapitasystems.com/products/rvs. Accessed Jan 2015.

[8] http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53. *Leon3 Processor*. Cobham Gaisler.

[9] F. Wartel et al. Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study. In *SIES*, 2013.

[10] F. Wartel et al. Timing analysis of an avionics case study on complex hardware/software platforms. In *DATE*, 2015.

[11] R. Wilhelm et al. The worst-case execution-time problem overview of methods and survey of tools. *ACM TECS*, 7:1–53, May 2008.