Timing Analysis of an Avionics Case Study on Complex Hardware/Software Platforms

Franck Wartel[ψ], Leonidas Kosmidis[†,*], Adriana Gogonel[⋆], Andrea Baldovin[φ], Zoe Stephenson[Ш],
Benoit Triquet[ψ], Eduardo Quiñones[†], Code Lo[⋆], Enrico Mezzetti[φ], Ian Broster[Ш],
Jaume Abella[†], Liliana Cucu-Grosjean[⋆], Tullio Vardanega[φ], Francisco J. Cazorla[†,‡]

[ψ]Airbus, France, [†]Barcelona Supercomputing Center, Spain, [*]Universitat Politècnica de Catalunya, Spain
[⋆]INRIA, France, [φ]University of Padua, Italy, [Ш]Rapita Systems, UK, [‡]Spanish National Research Council, Spain

***Abstract*—Probabilistic Timing Analysis (PTA) in general and its measurement-based variant called MBPTA in particular have been shown to facilitate the estimation of the worst-case execution time (WCET). MBPTA relies on specific hardware and software support to randomise and/or upper bound a number of sources of execution time variation to drastically reduce the need for user-provided information, thus replacing uncertainty by probabilities.**

MBPTA has been proven effective for specific single-core processor designs. However, particular hardware features and multicores in general challenge MBPTA application in industrial-quality developments. While solutions to those challenges have been proven on benchmarks, they have not been proven yet on real-world applications, whose timing analysis is far more challenging than that of simple benchmarks. This paper discusses the application of MBPTA to a real avionics system in the context of (1) software-only single-core solutions and (2) hardware-only multicore solutions with an ARINC 653 operating system.

I. Introduction

Software run in aircraft on-board processors has experienced an exponential growth during the last couple of decades, nearing 100 MB for commercial airplanes [12], and the trend is expected to continue with the increase of automated functionality and the complexity of existing software base. Similar trends are observed in other application domains, automotive and space among others, where software complexity also grows considerably. In order to respond to the increased computation power demand, more powerful hardware and software platforms need to be deployed, such as multicore processors with complex cache hierarchies. However, many of those systems have different degrees of real-time constraints as they have to produce correct results *in time*.

A number of timing analysis methods have been devised in the past to estimate the worst-case execution time (WCET) of programs so that guarantees can be attained regarding whether programs will complete execution in time. Static deterministic timing analysis (SDTA) and measurement-based deterministic timing analysis (MBDTA) [37] are worth of mention among them. SDTA has been shown to reach its practical limit for industrial-size problems [25], [18]. Exceedingly high cost are incurred to feed SDTA with all the information needed to obtain tight WCET estimates, if possible at all. Large pessimism arises from lack of knowledge. Such lack of knowledge translates into lower guaranteed performance per system unit, and thus a need for more hardware units. Either way, cost grows in pace with the number and complexity of the real-time functions needed. MBDTA instead relies on the expertise of the system designers to devise stressful tests and fix margins to account for the unknown. Unfortunately, although this approach is widely used in industry, no scientific confidence can be had on those margins. Further, as the complexity of software and hardware increases, and as the number of heterogeneous software functions to be integrated rises, the complexity to set affordable yet trustworthy margins becomes unattainable.

Probabilistic timing analysis (PTA) [8], [16], [9] and in particular its measurement-based variant (MBPTA) [9] has recently emerged as a powerful alternative. It relies on upper-bounding and/or randomising the sources of execution time variation (SETV) [7] that exist in the system, to obtain trustworthy and tight WCET estimates with affordable cost for the end user, by drastically reducing the number of SETV to be controlled. This is achieved with little overhead in the average case [35], [19]. PTA delivers a WCET distribution that upper bounds the actual execution time distribution of the program so that for any exceedance probability, the probabilistic WCET (pWCET) provided by PTA is equal or higher than the actual one for the program. The pWCET to be used is the one with an arbitrarily low exceedance probability so that it can be exceeded with a probability well below the failure probability specified by the system safety requirements, as for any other source of failure (e.g., hardware failures).

MBPTA has been successfully applied on an avionics case study run on a single-core processor equipped with the hardware support needed to enable the use of MBPTA [35]. However, two important challenges have not been addressed yet: (i) applying MBPTA to industrial applications running on platforms where MBPTA compliance is attained only by software means [21], and (ii) applying MBPTA to industrial applications running on multicore platforms, where the presence of hardware shared resources challenges the composability of the pWCET bounds, possibly preventing them from holding regardless of the tasks running on the other cores [17]. So far, those two solution candidates have only been studied in the context of reference benchmarks such as EEMBC [29] and Mälardalen [15]. In this paper we address those two challenges and make the following contributions:

1) We show step-by-step how MBPTA can be applied on top of software randomisation for an avionics case study.
2) We compare software-only and hardware-only solutions on a real application.
3) We evaluate a multicore version of the avionics case study, comparing it against its single-core counterpart.

The target application handles data concentration and maintenance of the flight control computers, and builds upon an ARINC 653 Operating System. The experience on the use and adaptation of MBPTA together with software-only and multicore solutions in the context of a real avionics application provides important insights to end users on how

to adopt this technology in their own environment. Further, results show that pWCET estimates obtained are comparable to those obtained with MBDTA, which is common practice in safety-related systems [6], proving that guarantees can be obtained without sacrificing performance if hard-to-predict deterministic SETV are replaced by randomised SETV for which probabilistic predictability can be attained.

The remainder of this paper is organised as follows. Section II presents the Integrated Modular Avionics (IMA) system selected as case study. Section III introduces MBPTA. Section IV describes how software-only solutions are used for the case study and compares them against hardware-only solutions. Section V presents a MBPTA compliant multicore design and evaluates the case study. Section VI introduces some related work. Section VII provides conclusions.

II. Avionics Case Study

The case study in question builds upon the IMA concept, which offers standard interfaces and encapsulated services for the integration of multiple subsystems into the same (shared) platform [33], potentially with mixed criticality levels across different applications. Similar paradigms can be found in other domains. In the automotive domain, for example, the main standard for the development of software components, AUTOSAR [4] bears some analogy in intent with IMA. Integrated architectures support the development of software components supplied from multiple sources with an incremental development and qualification.

A. IMA Concepts

Two different avionics applications have been considered. The first one, which we refer to as APP1, performs data concentration and maintenance of the flight control computers. APP1 is used for the evaluation of software-only and multicore solutions. The second application, APP2, computes an independent estimation of centre of gravity (CG) position, to secure the aircraft from CG excursions. APP2 is used only for the multicore evaluation.

Both applications run on an ARINC 653 Operating System (OS) [3], which offers temporal and spatial partitioning:

- *Temporal partitioning* allows different partitions to execute without affecting one another temporally. CPU time is strictly allocated to partitions. Partitions are the scheduling unit of a static schedule computed offline. The hyper-period of all partitions is known as **MA**jor **F**rame, and it is divided into a number of **MI**nor **F**rames. All MIF have the same duration and period, and contain a number of partitions. When a given partition is active, only processes that belong to it can be scheduled. More than one partition can be allocated to the one and the same application within a given MAF. By running one partition at a time, IMA assumes exclusive per-partition access to all resources, a natural concept for single-core environments. However, this is less obvious for multicores, unless time composability can be attained by construction to guarantee timing independence across parallel partitions.
- *Spatial partitioning* allows different partitions to execute accessing only the memory areas allocated to them. This implies that no access can occur from a partition outside of its memory areas.

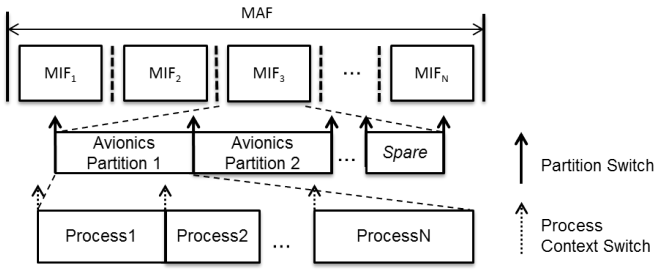

Fig. 1. MAF, MIF, partitions and processes structure in the context of IMA.

B. Case Study

The applications considered in the case study are scheduled as outlined in Figure 1, where APPi notionally corresponds to "Avionics partition i" in the figure. Each application runs a number of processes and each such process consists of a number of functions to acquire data from I/O ports. Data acquired are deserialised, processed, serialised and sent back through physical I/O ports. A number of A653 services may be invoked by these functions to this end. The execution time of those applications is analysed at the level of A653 processes, i.e. end-to-end measurements of periodic processes, as those processes are assigned hard deadlines.

The real applications are hosted on MPC755 processor based boards, where the L1 data cache (DL1) operates in copy-back mode. It has not been possible to use state-of-the-art SDTA for the single-processor version of this platform, within affordable effort for the end user and acceptable pessimism in the WCET estimates. SDTA is challenged by the timing interference caused by the execution of the A653 OS services and OS preemptions, and by the use of a copy-back DL1 cache. For the multicore processor version, the situation is no better, as state-of-the-art SDTA has not reached industrial quality there yet.

Single-core and multicore MBPTA-compliant hardware solutions are not yet implemented in any commercial processor although steps in this direction are currently being made [31]. Thus, all evaluations – included for software-only solutions – have been conducted on top of a processor simulator modelling the MPC755 instruction set architecture and a cycle-accurate timing simulator. The processor simulator is based on SoCLib [24]. Specific architecture details for the single-core and multicore architectures are provided as needed in the following sections. Applications are run on top of a A653 compliant OS prototype that provides zero-disturbance constant-time services [5] as needed by MBPTA. I/O data have been placed in specific memory regions given that I/O devices are not available in the simulation infrastructure.

III. MBPTA

MBPTA [9] has emerged as an industrial-friendly timing analysis technique as it allows deriving trustworthy and tight WCET estimates following a black-box approach on the user side provided that the underlying execution platform allows some properties to be attained [22] and measurements are collected conveniently to capture the upper-tail of the execution time distribution [7]. MBPTA relies on a platform that causes the execution time behaviour to occur with a given probability. With that, MBPTA allows deriving a pWCET

cumulative distribution that upper-bounds the actual execution time distribution of any software program. Such pWCET distribution can be expressed in the form of a complementary cumulative distribution function (CCDF), also known as exceedance function. From the CCDF one can determine the pWCET estimate that is exceeded with a probability below a given exceedance threshold (e.g., 10^{-15} per run).

MBPTA rests on a platform where the different SETV are either upper-bounded or time-randomised [7] at analysis time so that the execution times observed there are either deterministically or probabilistically higher than those that may occur during operation. MBPTA constructs the pWCET distribution by collecting a number of observations (end-to-end execution time measurements) and using *Extreme Value Theory* (EVT) [23], [9] to obtain a pWCET distribution. EVT provides continuous distributions that allow obtaining pWCET estimates for arbitrarily low exceedance probabilities.

A. MBPTA Requirements

EVT, which is used by MBPTA, requires that the observed execution times can be described by independent and identically distributed (i.i.d.) random variables according to [9]. As explained in [7], platforms used with MBPTA must allow attaining i.i.d. properties by construction upper-bounding and/or randomising SETV. Recent work shows that those constraints may be relaxed in some cases [34]. In particular, input-dependent SETV with relatively low impact on the WCET are typically upper-bounded like, for instance, the initial instruction cache state, which is invalidated before each measurement, and the input-value-dependent functional units, which are enforced to operate at their highest latency at analysis time. Other SETV with larger impact on the execution time are randomised like, for instance, placement of the different memory objects in cache. Placement randomisation has been achieved through different means: (i) using hardware random placement solutions [19], [20], and (ii) randomising with software-only means the placement of the memory objects across runs [21]. Hardware random placement has already been shown to be effective in the context of an industrial case study [35]. However, software-only solutions are more complex to apply as they need to integrate the application with a runtime layer that is able to place the code of functions and their stack frames at random locations during execution. This has been proven doable on benchmarks. One of the main challenges faced in this paper is the use of this technology on a real avionics case study. Randomisation has also been used in the context of multicores to arbitrate the access to shared resources such as a shared bus [17]. As explained in [7], MBPTA requires observing at analysis time the same or higher deterministic or probabilistic latencies than at deployment for all SETV. In the particular case of a shared bus, this implies that, the arbitration policy in use must be non-work-conserving at analysis time. In other words, it must always arbitrate across all potential contenders (e.g., all cores attached to the bus) regardless of whether they have pending requests or not, as this is the worst case at deployment. This caters for time-composability by construction, as needed by MBPTA.

Finally, MBPTA also relies on architectures free of timing anomalies, as this allows effectively upper-bounding the different SETV [7], [35].

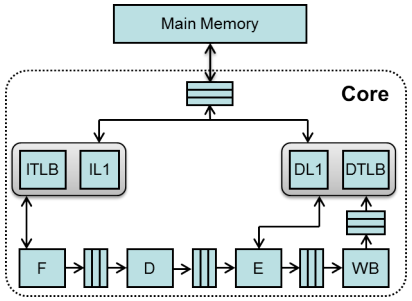

Fig. 2. Block diagram of the single-core processor architecture.

In summary, MBPTA requirements have been proven to be attainable for some single-core processor designs by software means. Hardware means have shown to enable MBPTA compliance on top of a broader range of processor designs including single-cores and multicores. However, solely hardware-only single-core solutions have been proven suitable for industrial applications. In the remainder of this paper we report on the use of software-only solutions for single-cores and of hardware-only solutions for multicores against real-world industrial systems.

IV. Software-only Solutions

This section evaluates the case study in a single-core processor where MBPTA compliance is attained by software-only means, and compares it against hardware-only solutions.

A. Experimental Setup

Figure 2 depicts the processor architecture. It includes a pipeline with in-order issue and finalisation, fixed-latency units (all of them 1 cycle although this is irrelevant in this study) and it is free of timing anomalies by construction [36]. It includes instruction and data first level caches (IL1 and DL1 respectively) and translation look-aside buffers (ITLB and DTLB respectively). IL1 and DL1 are 32KB 8-way 32B/line caches with modulo placement and least recently used (LRU) replacement policies. DL1 is copy-back. ITLB and DTLB have 8 entries for 1KB pages and are fully-associative (LRU replacement). Thus, the cache system is the only component breaking MBPTA compliance, which is solved by using software-only randomisation of the data and code objects [21]. For comparison purposes we also consider hardware-only solutions where random placement and random replacement are used instead for modulo and LRU respectively [19]. The hit and miss latency is 1 and 100 cycles respectively.

B. Use of Software Randomisation

The software randomisation of the location of memory objects is performed by a compiler and run-time system called Stabilizer [10]. The compiler pass of Stabilizer has been developed within LLVM [1] as described in [21]. In this particular experiment, the OS calls were stubbed into the application code to ease the integration of the prototype compiler on top of the simulator. This simplification does not really deflect from the real system, as the OS for use was designed (and proven) to have fully time-composable (constant-time, zero interference) behaviour on the target processor [5].

The MBPTA method involves the following steps.

TABLE I
RESULTS FOR THE I.I.D. TESTS.

SW rand	Indep	i.d.	Both tests passed ?
Code	0.14	0.83	Ok
Stack	0.21	0.74	Ok
Code+Stack	0.17	0.90	Ok

TABLE II
MINIMUM NUMBER OF RUNS (*MNR*) AND TIME REQUIRED TO COLLECT DATA AND APPLY MBPTA.

	MNR	time
SW-only	550	9 min
HW-only	600	10 min

1) Execution Time Observations: The experiments presented here consider end-to-end execution time measurements of APP1. Each of its processes (see Figure 1) calls functions whose body and stack frame are randomly located in memory. No global data exist, so that their location does not need to be randomised. In order to preserve i.i.d. execution-time behaviour across MAFs, the location of functions and stack frames changes at the beginning of each MAF, which forces the cache to be flushed. Hence, cache conflicts among functions (instruction cache conflicts) and stack frames (data cache conflicts) randomly change across MAFs.

2) Fulfilling the i.i.d. properties: I.i.d. properties must be warranted for EVT to be used. We have verified that they hold with the tests described in [9] when using code randomisation only, stack randomisation only, and both of them simultaneously. For the standard 0.05 threshold value, the independence test is passed if the test delivers a value below 1.96, and the i.d. test passes if the returned value is above 0.05. The test results are reported in Table I: all tests were passed successfully, which means that software randomisation together with cache flushing satisfies the MBPTA requirements. If we tried to test the used architecture without software randomisation, all execution times would fall very near to one another, thus describing a degenerate function that fails at least the independence test and cannot be used with EVT.

3) CRPS Test: Minimum Number of Runs Required: Determining the minimum number of observation runs required for MBPTA to converge is paramount for industrial applications as it is a significant cost factor for development. Table II summarises the minimum number of runs (labelled as *MNR*) and the time (in minutes) required to collect them and perform the timing analysing of APP1 when using SW-only or HW-only randomisation techniques. While the application of MBPTA takes negligible time (always less than 10 seconds), collecting measurements is a more time consuming process. Given that each measurement takes less than 1 second, the duration in both cases is below 10 minutes.

The minimum number of runs in the case of SW-only solutions is very similar to that of HW-only solutions and in the same order as in other applications of MBPTA [9], [35] (see Table II).

4) EVT projection: Figure 3 shows the pWCET estimates for an exceedance threshold of 10^{-15} per run (thus acceptable even for the highest integrity level in avionics [35]) comparing HW-only and SW-only randomisation. For the sake of comparison, we have included the highest observed execution time (not WCET) in the baseline architecture with no randomisation (*No rand*), and stacked them on top of what would be the

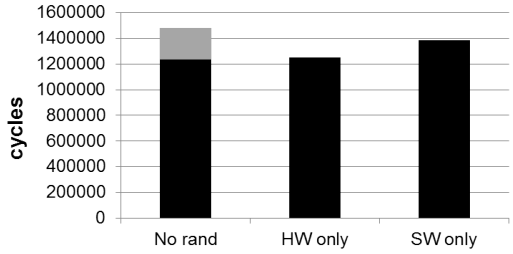

Fig. 3. pWCET estimates of applying HW-only and SW-only randomisation compared to actual execution time with no randomisation.

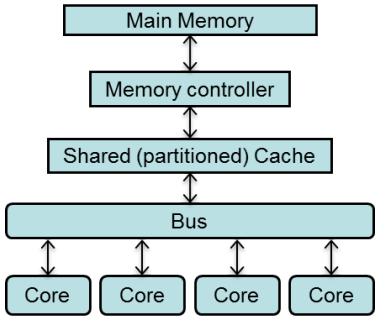

Fig. 4. Block diagram of the multicore processor architecture; the core design is the same as for the single-core case.

WCET estimate if a 20% margin was added on top of the execution time, in line with industrial practice [35], [6]. As shown, MBPTA together with SW-only solutions delivers a pWCET estimate only 12.2% larger than the highest execution time observed without randomisation. Moreover, SW-only solutions provide tighter estimates than the case where a 20% margin is added, while also providing higher confidence as scientific methods support the pWCET estimation process. When compared against HW-only solutions, SW-only solutions deliver 10.9% higher pWCET estimates, showing that, while HW-only solutions are more efficient, SW-only solutions are competitive and less intrusive as hardware remains unchanged.

Average Performance. Although less relevant than pWCET estimates, we have also evaluated average performance and observed that HW-only and SW-only randomisation increase average execution time by 0.7% and 12.0% w.r.t. no randomisation. Note that conventional placement and replacement policies have been devised with average performance in mind, thus being quite often superior than random ones.

V. MULTICORE

This section introduces a MBPTA compliant multicore design and evaluates the case study in the context HW-only solutions suited for multicore processors.

A. Experimental Setup

The MBPTA compliant multicore design is depicted in Figure 4. The same core design as for the single-core setup for SW-only solutions has been considered with some differences: caches implement random placement and replacement [19], [20], DL1 is write-through, so all store instructions are forwarded to the second level unified cache (UL2). UL2 is copy-back to reduce the number of memory accesses. In-flight memory requests have been limited to 16 per core and the store buffer is limited to 4 entries. UL2 is a 256KB 8-way 32B/line cache with 4 cycles hit latency. Placement

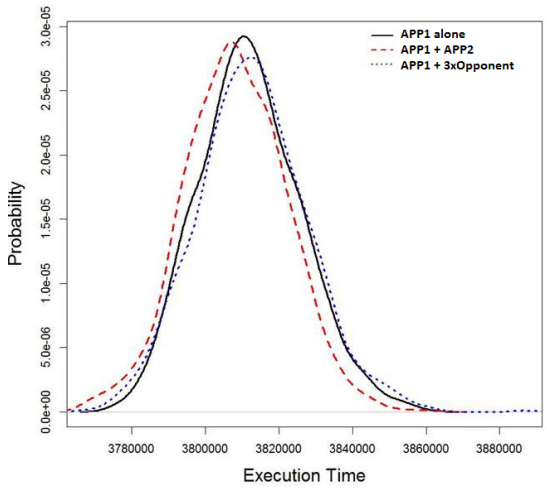

Fig. 5. Probability distribution function for the execution time of APP1 in 3 different workloads.

and replacement policies are analogous to those of DL1 and IL1. UL2 assumes homogeneous cache partitioning (way partitioning) across cores [27]. A 2-cycle latency shared bus is used with random-permutations arbitration policy, which has been proven to be the most efficient MBPTA compliant policy [17]. Analogously, we use the same arbitration policy for the memory controller to arbitrate requests across cores. In this study, we have focused on a close-page, interleaved-bank, DDR2-800E memory setup as in [17], where inter-request latency is 27 cycles and requests are served in 16 cycles once access is granted.

B. Multicore Workloads

A MBPTA compliant processor design provides time-composability by construction, so that an application can be analysed in isolation and its results stay valid upon composition with other applications. In order to corroborate this claim, we have analysed APP1 in our 4-core multicore in 3 different setups: (1) alone, (2) APP1+APP2 and 2 idle cores, and (3) APP1 and 3 copies of a program constantly accessing memory (operating as an *opponent*), thus creating high contention in the bus and the memory controller. Figure 5 shows the probability distribution function of the execution time of APP1 in the 3 workloads for 1,000 runs of each. As shown, the distribution does not change noticeably and variations can be caused just because of random events. We used the Kolmogorov-Smirnov identical distribution test [14] (the same used by MBPTA) to statistically corroborate this conclusion. Thus, we can conclude that any workload is acceptable to obtain measurements. In our case, we use workload (2) as it allows collecting measurements for APP1 and APP2 simultaneously.

We also run those workloads on a deterministic architecture with modulo placement and LRU replacement caches, partitioned UL2, and round-robin bus and memory controller arbitration (work-conserving, so only arbitrating across pending requests). The execution time variation observed between workloads (1) and (3) ranges with a ratio of 3.5x.

C. Multicore Evaluation

As for the single-core case, we have verified that i.i.d. properties are attained with the corresponding tests for both applications. The number of runs needed for each application was up to 750, thus requiring a maximum of 13 minutes to obtain pWCET estimates on a real platform. Those measurements have been used to obtain pWCET estimates for the same exceedance threshold as before, 10^{-15} per run.

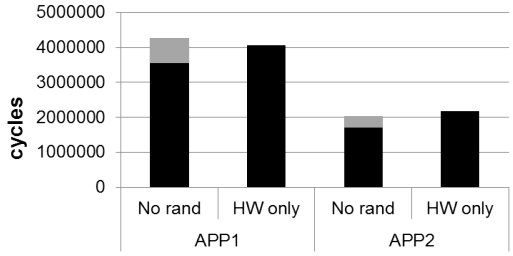

Fig. 6. pWCET estimates of applying HW-only randomisation compared to actual execution time with no randomisation.

Figure 6 shows the pWCET estimates for APP1 and APP2. For the sake of comparison, we have also included the actual execution time in a non-time-randomised architecture (*No rand*) where APP1 and APP2 run together with 2 instances of the *opponent*, and stacked on top what would be the WCET estimate if a 20% margin was added on top of that execution time. As shown, MBPTA on top of the multicore delivers pWCET estimates 15% and 28% higher than the actual execution time with no randomisation for APP1 and APP2 respectively. Those values are very close to the typical 20% margin. Notably however, the 20% margin in this case should account not only for different memory placements of objects that can affect cache behaviour in the non-randomised setup, but also for higher interference on shared hardware resources competed for in parallel by the individual cores. APP1 and APP2 do not run each against 3 instances of the *opponent*. While creating very high contention, that *opponent* cannot be asserted to cause the worst possible contention of shared hardware resources. Overall, MBPTA on top of multicores delivers similar pWCET estimates to those where a 20% margin is added providing higher confidence due to the scientific arguments behind MBPTA.

Average Performance. Average execution time for APP1 and APP2 on the MBPTA compliant multicore is 7.4% and 4.2% higher than that on top of the non-randomised and non-time-composable multicore respectively. As explained before, this is expected as random policies are not as efficient in terms of average performance as conventional ones.

VI. Related Work

Our work falls in the domain of probabilistic timing analysis (PTA). Systems with at least one parameter described by a probability distribution need some kind of PTA. The work on PTA can be classified into two main groups: (1) schedulability analysis and (2) the determination of probability distributions for the parameters. We omit the first group as our work falls into the second. In this second group, a plethora of techniques has been devised providing probability distributions for the probabilistic execution time and WCET [13], [16], [9], [8], [2] in general or for component-based systems [28]. Also probabilistic arrivals have been studied recently [11], [26].

Our work concerns the use of probability distributions for probabilistic worst-case execution time. This work presents the utilisation of the MBPTA method originally presented in [9] on an industrial case study. While MBPTA has already

been proven on a case study [35], this was only done for HW-only solutions in single-core processors. Our paper proves the scalability of the approach in more challenging scenarios such as those where MBPTA properties can only be attained by software means and in multicore processors where time composability across tasks running simultaneously challenge time analysability.

The MBPTA approach has been studied in the context of safety avionics standards (DO-178B) [32], and a certification argument for the use of the techniques in airborne software was produced [30]. That certification work continues in the PROXIMA project [31].

VII. Conclusions

In this paper we have shown that SW-only solutions to attain MBPTA compliance can be successfully applied on IMA-based applications in single-core environments delivering competitive WCET estimates close to what can be achieved with HW-only solutions and close to the actual execution time (not WCET) on non-randomised platforms. We also show that those avionics applications can be analysed on top of MBPTA compliant multicores with the same (low) degree of complexity as in single-core processors and also delivering tight WCET estimates. In both cases MBPTA has been directly applied on legacy code and with little information requirements from the user side, as needed by industry. Our results also show that good average performance is still attained despite of the fact that randomisation is typically against average performance in comparison with conventional architectures.

The evaluation of this case study also reveals the importance of time-composability in the context of multicores. If time-composability is attained, as in the case of the MBPTA compliant multicore, WCET estimates can be obtained in isolation enabling incremental qualification as needed by industry.

Finally, SW-only solutions for multicores have not been developed yet, but considerable effort is currently being devoted to filling that gap, which will allow using PTA with low cost for COTS processors.

All in all, we have shown that MBPTA scales to the challenge of SW-only solutions and multicore designs for some real applications.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROARTIS Project (grant agreement 249100) and the PROXIMA Project (grant agreement 611085). This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant TIN2012-34557 and the HiPEAC Network of Excellence. Leonidas Kosmidis is funded by the Spanish Ministry of Education under the FPU grant AP2010-4208.

References

[1] LLVM. http://dragonegg.llvm.org/.

[2] S. Altmeyer and R.I. Davis. On the correctness, optimality and precision of static probabilistic timing analysis. In *DATE*, 2014.

[3] ARINC. Avionics Application Software Standard Interface: ARINC Specification 653P1-3. Aeronautical Radio. www.arinc.com.

[4] AUTOSAR. AUTomotive Open System ARchitecture, 2012. http://www.autosar.org.

[5] A. Baldovin, E. Mezzetti, and T. Vardanega. A time-composable operating system. In *WCET workshop*, 2012.

[6] M. Buhlmann. Keynote: High integration ECU in automotive environment. In *ECRTS*, 2014.

[7] F.J. Cazorla, T. Vardanega, E. Quinones, and J. Abella. Upper-bounding program execution time with extreme value theory. In *WCET workshop*, 2013.

[8] F.J. Cazorla et al. Proartis: Probabilistically analyzable real-time system. *ACM Transactions on Embedded Computing Systems*, 2012.

[9] 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 *ECRTS*, 2012.

[10] C. Curtsinger and E.D. Berger. Stabilizer: Enforcing predictable and analyzable performance. Technical report, UMass, CS TR 2011-43, 2011.

[11] F. Dewan and N. Fisher. Efficient admission control for enforcing arbitrary real-time demand-curve interfaces. In *RTSS*, 2012.

[12] G. Edelin. Embedded systems at thales: the artemis challenges for an industrial group. In *ARTIST Summer School in Europe*, 2009.

[13] S. Edgar and A. Burns. Statistical analysis of WCET for scheduling. In *RTSS*, 2001.

[14] W. Feller. *An introduction to Probability Theory and Its Applications*. John Willer and Sons, 1996.

[15] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper. The Mälardalen WCET benchmarks – past, present and future. In *WCET Workshop*, 2010.

[16] J. Hansen, S Hissam, and G. A. Moreno. Statistical-based WCET estimation and validation. In *WCET Workshop*, 2009.

[17] J. Jalle, L. Kosmidis, J. Abella, E. Quiñones, and F.J. Cazorla. Bus designs for time-probabilistic multicore processors. In *DATE*, 2014.

[18] R. Kirner and P. Puschner. Obstacles in Worst-Case execution time analysis. In *ISORC*, 2008.

[19] L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. A cache design for probabilistic real-time systems. In *DATE*, 2013.

[20] L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. Multi-level unified caches for probabilistically time analysable real-time systems. In *RTSS*, 2013.

[21] L. Kosmidis, C. Curtsinger, E. Quinones, J. Abella, E. Berger, and F.J. Cazorla. Probabilistic timing analysis on conventional cache designs. In *DATE*, 2013.

[22] L. Kosmidis, E. Quinones, J. Abella, T. Vardanega, I. Broster, and F.J. Cazorla. Probabilistic timing analysis and its impact on processor architecture. In *DSD*, 2014.

[23] Samuel Kotz and Saralees Nadarajah. *Extreme value distributions: theory and applications*. World Scientific, 2000.

[24] LiP6. SoCLib. www.soclib.fr/trac/dev.

[25] E. Mezzetti and T. Vardanega. On the Industrial Fitness of WCET Analysis. In *WCET Workshop*, 2011.

[26] M. Neukirchner, T. Michaels, P. Axer, S. Quinton, and R. Ernst. Monitoring arbitrary activation patterns in real-time systems. In *RTSS*, 2012.

[27] M. Paolieri, E. Quinones, F.J. Cazorla, G. Bernat, and M. Valero. Hardware support for WCET analysis of hard real-time multicore systems. In *ISCA*, 2009.

[28] R. Perrone, R. Macedo, G. Lima, and V. Lima. An approach for estimating execution time probability distributions of component-based real-time systems. *J. UCS*, 15(11):2142–2165, 2009.

[29] Jason Poovey. *Characterization of the EEMBC Benchmark Suite*. North Carolina State University, 2007.

[30] PROARTIS. D2.3 certification arguments guidelines. http://www.proartis-project.eu/system/files/D2.3%20Certification%20Arguments%20Guidelines%20(Initial%20Draft).pdf.

[31] PROXIMA. *EU-FP7 Project: www.proxima-project.eu*.

[32] RTCA and EUROCAE. *DO-178B / ED-12B, Software Considerations in Airborne Systems and Equipment Certification*, 1992.

[33] SAE International. Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. *ARP4761*, 2001.

[34] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart. On the sustainability of the extreme value theory for WCET estimation. In *WCET Workshop*, 2014.

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

[36] I. Wenzel, R. Kirner, P. Puschner, and B. Rieder. Principles of timing anomalies in superscalar processors. In *ICQS*, 2005.

[37] Wilhelm R. et al. The worst-case execution-time problem overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems*, 7:1–53, May 2008.