PROXIMA: Improving Measurement-Based Timing Analysis through Randomisation and Probabilistic Analysis

Francisco J. Cazorla[1,12], Jaume Abella[1], Jan Andersson[2], Tullio Vardanega[3], Francis Vatrinet[4], Iain Bate[5], Ian Broster[6], Mikel Azkarate-askasua[7], Franck Wartel[13], Liliana Cucu[9], Fabrice Cros[10], Glenn Farrall[11], Adriana Gogonel[9], Andrea Gianarro[2], Benoit Triquet[8], Carles Hernandez[1], Code Lo[9], Cristian Maxim[9], David Morales[1], Eduardo Quinones[1], Enrico Mezzetti[3], Leonidas Kosmidis[1], Irune Aguirre[7], Mikel Fernandez[1], Mladen Slijepcevic[1], Philippa Conmy[6], Walid Talaboulma[9]

[1] Barcelona Supercomputing Center, Spain
[2] Cobham Gaisler, Sweden
[3] University of Padova, Italy
[4] SYSGO, France
[5] University of York, UK
[6] Rapita Systems l.t.d., UK
[7] IK4-Ikerlan, Spain
[8] Airbus SAS, France
[9] INRIA, France
[10] Airbus Defence and Space, France
[11] Infineon, UK
[12] IIIA-CSIC, Spain
[13] While he was at Airbus SAS, France

***Abstract*—The use of increasingly complex hardware and software platforms in response to the ever rising performance demands of modern real-time systems complicates the verification and validation of their timing behaviour, which form a time-and-effort-intensive step of system qualification or certification. In this paper we relate the current state of practice in measurement-based timing analysis, the predominant choice for industrial developers, to the proceedings of the PROXIMA[1] project in that very field. We recall the difficulties that the shift towards more complex computing platforms causes in that regard. Then we discuss the probabilistic approach proposed by PROXIMA to overcome some of those limitations. We present the main principles behind the PROXIMA approach as well as the changes it requires at hardware or software level underneath the application. We also present the current status of the project against its overall goals, and highlight some of the principal confidence-building results achieved so far.**

I. Introduction

Complex high-performance hardware and software components are increasingly used in critical real-time embedded systems (RTES)[2] in match with the rising computational demands of new-generation avionics, automotive, railway and medical RTES.

The verification of the timing behaviour in industrial-quality RTES requires providing evidence that the application will always perform its duties in a timely fashion. This verification involves the use of methods to estimate the worst-case execution time (WCET) of the time-critical application programs, and their completion time once governed by the scheduling decisions made at system level. WCET estimates need to be sufficiently tight (to avoid incurring undue pessimism) and trustworthy enough to earn the level of confidence defined in the applicable safety standards.

Determining a tight and sound Worst-Case Execution Time (WCET) bound of software programs running on modern, high-performance processors is especially challenging [4]. Various WCET analysis techniques exist in the state of the art. The industrial users in PROXIMA all come from measurement-based deterministic timing analysis (MBDTA), which is not surprising owing to its considerable presence in current industrial practice [28]. With MBDTA, the software programs of interest are executed on the target platform to collect a score of execution-time measurements. To achieve minimum soundness, MBDTA requires the user to have control on: (i) the conditions in which the measurement runs are made so that they represent those expected during operation; and (ii) the input and state conditions that may cause the program to incur its worst-case timing behaviour.

The most well-known factors that affect the program's execution time include the input vectors that determine – among other – the control flow path taken by the program in the measurement runs. We call them *high-level level sources of jitter*. The use of complex high-performance hardware creates other *low-level sources of jitter*, which include: the mapping functions that determines how software objects are assigned to memory, as they determine how they are placed in cache, the conflicts that they can suffer, with the consequent execution-time effects; and, in a multicore setting, the way parallel contention on access to shared hardware resources (e.g., a bus) cause access requests to jitter in time.

With MBDTA, the end user must control all input and state conditions with influence on execution-time jitter so as to achieve sufficient *coverage* of its effects for all sources of it in the system (across the whole set of measurement runs made during analysis). While tools exist for high-level sources of jitter that validly aid the user in that endeavour, no such tools exist for low-level sources of jitter. For instance, it is hard to assess whether all potential cache layouts, or a representative subset of them, have been exercised in the measurement runs. Likewise in a multicore setting, determining whether the access requests from a program

[1]Probabilistic real-time control of mixed-criticality multicore systems.
[2]For safety, availability, security, mission or business concerns.

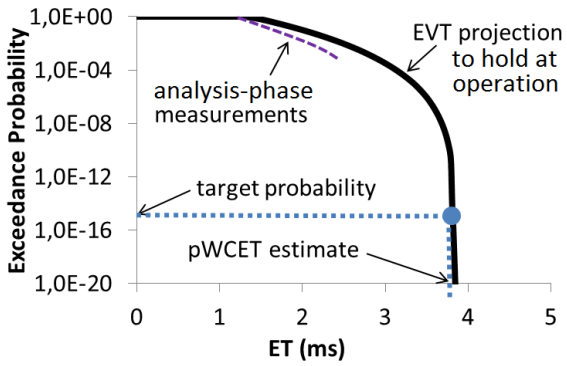

Figure 1. Introduction to MBPTA and its use.

have aligned with those of its parallel contenders in a manner that represents a sufficiently stressful scenario is exceedingly hard. Overall, the lack of control and coverage of low-level sources of execution-time jitter in a processor with high-performance features severely limits the confidence that can be provided on the computed WCET value.

Contribution. The PROXIMA project aims at enabling cost-effective verification of timing analysis – including worst case execution time – for real-world software programs running on complex multicore and manycore platforms. The project vision rests on two main principles. First, selectively introducing randomization in the timing behaviour of selected hardware and software resources or making them work on their worst latency so as to cause the whole spectrum of impact of low-level sources of jitter to be captured in measurement runs. Second, using a measurement-based probabilistic timing analysis (MBPTA) techniques to determine the worst combined impact of all sources of jitter present in the system, thereby quantifying the probability that the execution time of the software program of interest may exceed a given bound. The injection of timing randomisation (and forcing some resources to work on their worst latency) in the execution platform is a fundamental enabler to the guarantee that, across a sufficiently low number of runs, the impact of all existing sources of jitter can be individually covered. This tenet is in contrast with deterministic architectures where, no matter how many measurement runs the user may make, it is hard if at all possible to provide a quantitative guarantee that all the timing phenomena of interest have been observed. Timing randomisation also ensures that the impact that each individual source of jitter has on program execution time follows a probabilistic law, which allows using sound statistical means to determine the number of runs required to observe its manifestation and thus apply probabilistic modelling soundly. In that manner the user is relieved from the heavy control obligations carried by the use of MBDTA.

II. Measurement-Based Deterministic Timing Analysis: State of Practice

Cost/benefit considerations compounded with industrial pragmatism cause MBDTA to be the dominant state of practice. The typical conduct is to capture the "high-water mark" value in execution time across multiple runs of the program of interest and then add an engineering margin to it (e.g., 20%) to compensate for the unknown. For this method to be used with sufficient confidence, substantial effort must be expended to ensure that the worst-case conditions have been exercised or closely approximated, which however is diluted in the overall testing campaign. This has proven to be feasible and viable for simple hardware and simple software.

In the general case however, software programs may have an inordinately large input space, which cannot possibly be exercised in a test campaign searching for worst case. The input vectors used are therefore those intended for functional testing, which may not be fit to incur the longest execution time, falling short on the side of the worst-case path or in the coverage of the low-level sources of jitter. Tool support exists that allows this manual process to be reduced to a series of smaller testing problems that can be more easily managed. For example, a tool may automatically combine the measurements taken from multiple tests on the program of interest, to calculate a high-water mark bound that includes conditions that are not necessarily exercised in a specific test case. Doing that removes the need to drive the program down the worst case path in one single test case. Instead, multiple (smaller and easier) test cases are used to drive each fragment to its own worst case, the results being automatically combined according to the program structure. This approach, which is implemented in commercial tools, has seen broad acceptance in DO-178B/C projects, due to it being incremental on the existing methods, with significant cost saving in terms of testing effort.

As platform complexity increases, the effort to test the longest path within a single test case exceeds practical and cost limitations. Furthermore, low-level factors out of the control of the tester (such as cache state, jitter caused by floating point operations and such like) limit the reliability and confidence of this method.

- For instance, the memory placement of software objects has been deemed a factor of high consequence on execution time in the presence of cache memories as it determines how different addresses compete for cache space. Even if those addresses can be fixed so that inter-task side-effects can be avoided, this is not so for stateful services from the operating system, whose execution time may depend on execution history. Services using different memory locations as a result of past history, would cause different cache access patterns and different execution times to emerge depending on the type and cardinality of tasks included in the test.

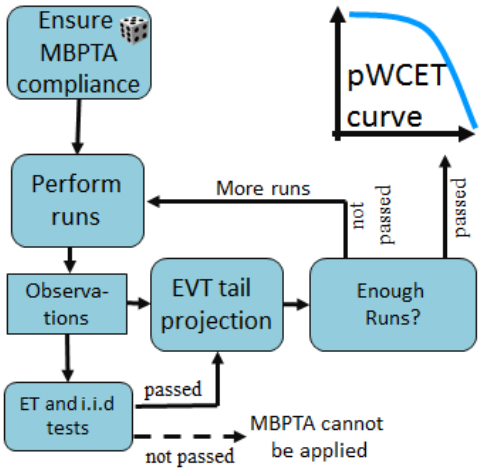

Figure 2. MBPTA's use procedure

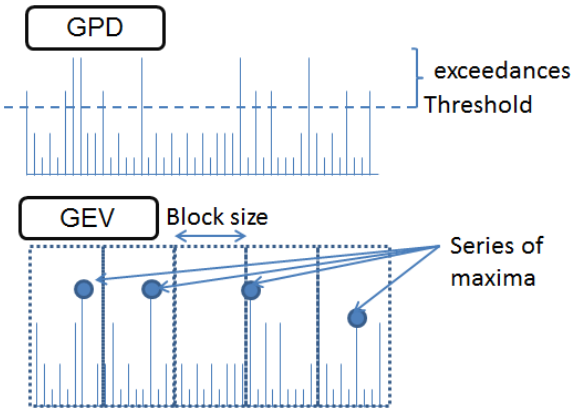

Figure 3. GPD and GEV

- Another example is the floating-point unit. For most architectures the floating-point unit takes a variable latency depending on the particular values operated. This cause that depending on the particular values used in each experiment – which the user can hardly control – the program will suffer a variable impact due to floating point operations.

Getting accurate and cost-effective timing analysis ultimately comes down to a question of representative testing, which means selecting a suitable level of detail for components (ideally as large as possible) so that the user can ensure that test inputs and test conditions exercise each component adequately. This should provide confidence that all important sources of jitter have been observed without introducing additional conditions that are infeasible in practice. Then those results need to be combined in a representative way.

III. Probabilistic Analysis and Randomisation

PROXIMA's MBPTA derives probabilistic WCET (pWCET) estimates in the presence of high-performance hardware. pWCET distributions express the maximum probability with which *one instance* of the program can exceed a given execution time bound. For instance, in Figure 1 we observe that the probability that one instance of the program to run longer than 4 ms is smaller than 10^{-14}. The particular execution-time value chosen is the one whose exceedance probability is deemed sufficiently low in relation to the integrity level of the functionality being analysed, dependent on the corresponding safety standard. For instance, the execution rate of a program could be used to determine the exceedance threshold per instance such that the set of program's executions occurring in an hour can be shown not to incur more exceedances than a given threshold (e.g., 10^{-9}). Interestingly, at the moment safety-related standards and guidance documents do not support the concept of probabilities associated with software. However it is not unusual for acceptable practice for certification to change over time, e.g. the adoption of fixed priority scheduling in aircraft engines [7], [16] which contrasts with the earlier practice of static scheduling. When the need arises for new technology features, the certification authorities often develop their position on issues to be considered and the way the technology might be used, e.g. for caches [25] and multicore processors [26].

For the application of MBPTA, we differentiate between two moments in the lifetime of the system: the *analysis phase*, when verification of the timing behaviour takes place; and the *operation phase* when the system becomes operational. *The goal of MBPTA is to compute the pWCET function of the program of interest with execution-time measurements taken at analysis time that are guaranteed to represent the operating conditions that may occur during operation*. This requires MBPTA to have good control of all low-level sources of jitter during analysis. The values of all sources of execution-time jitter for that experiment are referred to as *execution conditions* for an experiment. MBPTA therefore requires that the execution conditions under which measurements are collected during analysis lead to equal or worst timing behaviour than determined by the conditions that can arise during operation [9].

A. Extreme Value Theory

MBPTA uses Extreme Value Theory [14], [22] (EVT) to build a pWCET distribution based on a sample with a limited number of observations collected during the analysis phase (e.g., in the order of thousands of execution time measurements). Below we provide a brief and informal description of the foundations of EVT: for more formal descriptions of EVT, see [14], [22].

EVT is used to study the probabilities associated with the occurrence of extreme (and thus rare) events. That is, EVT is used to model the behaviour of maxima/minima in the tail of the probability distribution of those events. EVT has been

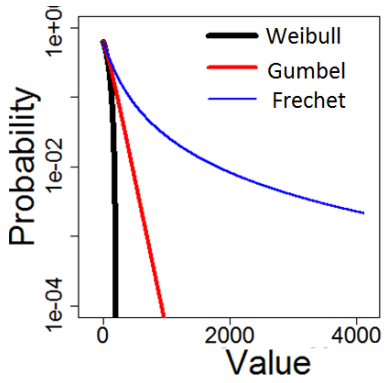

Figure 4. Gumbel, Frechet and Weibul

successfully applied in a number of fields, e.g. hydrology and insurance. EVT has two main results. First, for the distribution of excesses over a threshold (see the top part of Figure 3), EVT shows that the limiting distribution is a generalized Pareto distribution (GPD). Second, – under certain conditions – the distribution of the standardized maximum of a series (see the bottom part of Figure 3) converges to one of the Gumbel, Frechet, or Weibull distributions, Figure 4. All those three distributions are described within the generalized extreme value (GEV) distribution.

EVT requires that the data being analysed can be modelled with independent and identically distributed random variables [14][8]. This can be assessed using specialised statistical tests. Interestingly, some authors have shown that independence across observations is not strictly needed as long as maxima are independent or the dependence across maxima is weak [10], [24]. However, in the rest of this paper we build upon independent data since it is a by-product of MBPTA-compliant platforms.

The Exponential Test is also used to confirm that the maximum of the series converges to Gumbel's exponential tail, which is a good fit the WCET problem since the execution times of a program are finite but its maximum is unknown [11].

MBPTA applies EVT to derive pWCET estimates of a program running on a computing platform that has characteristics which ensure the emergence of sufficient randomness [11]. There is a fundamental challenge in applying EVT to solve the pWCET problem: EVT treats the system as a black box so that the projection it produces from the data it is fed, solely holds for exactly that system. This requires the user to ensure that the observation data obtained from the system incurred during analysis have an upper-bounding relation to those the system may produce during operation. Simple-minded application of EVT to analysis-time observations that do not warrant the above condition would fail to provide sound results for the operation-time behaviour of the system. Another way to appreciate the significance of this problem is to note that EVT should be understood as a method that predicts the worst combined effects of phenomena individually observed during analysis but not to predict the occurrence of those never observed.

As a precondition to sound use of EVT, MBPTA requires that the sources of execution-time jitter phenomena observed during analysis have sufficient (upper-bounding) representativeness of their manifestation during operation. If this condition is warranted, then feeding these observations to EVT produces an approximation of the tail of the distribution of the worst-case timing behaviour that the program may exhibit in the operational life of the system. Next we discuss how this can be achieved using MBTA-compliant processors.

B. Meeting MBPTA requirements

To meet the above-described MBPTA requirements, MBPTA-compliant processors are modified in two ways: randomisation is injected in the timing behaviour of hardware resources whose jitter is high (e.g. caches and buses) [9] so that the probability of their worst-case behaviour can be captured in analysis-time measurement runs [12][3]; other processor resources with small jitter are instead set to work at their highest latency during analysis. As a result the corresponding measurements at analysis upper bound the execution time distribution for that resource that may manifest during operation time [21]. The low-to-high boundary for the above discrimination is processor and application dependent.

It is worth noting that the goal of MBPTA compliance, i.e. randomizing the timing of some (jittery) resources and making the remaining work in their worst latency, is not providing independent and identically distributed (i.i.d.) execution times. Instead, the goal is to help providing an argument that i) analysis time observations capture the execution time impact of jittery resources; ii) that impact upperbounds the one that can occur during operation. The fact that i.i.d. execution times are obtained is a by-product of MBPTA compliance.

C. Application Process

Figure 2 outlines the MBPTA procedure. First, the user has to provide confidence that the execution platform is rendered MBPTA-compliant by hardware or software means, so that the sources of execution-time jitter can be deemed controlled. Second, the user gathers execution-time observations from measurement runs of the software program of interest (Unit of Analysis). Third, this body of data is processed with the Block Maxima Method [14] to derive a distribution of maxima, which is fed to a checker that determines whether the samples that compose it are independent and identically distributed (as EVT requires). The checker also assesses whether the data follows the Gumbel distribution. Once all the tests are passed, the EVT process for tail extension is applied, which determines the parameters of the Gumbel

distribution that best fit the given distribution of maxima. Finally, we assess whether the collected measurements are sufficient to ensure statistical representativeness. Should this not be the case, more measurements would have to be collected and the process would be repeated until this condition is satisfied. Experience suggests that the number of runs required is in the order of thousands.

IV. Current Status and Future steps

In its quest for high test-readiness level, PROXIMA has strived to advance the maturity of all individual elements required for correct functioning of MBPTA. We now describe the situation for each such element.

A. MBPTA Improvements

In terms of the MBPTA procedure per se, PROXIMA is advancing on the following fronts.

- *Path Coverage*: MBPTA provides pWCET estimates that are valid only for the paths that have been exercised by the input vectors provided by the user. However, complexity and cost considerations often restrain the user from providing adequate path coverage. It is also worth noting here that MBPTA technology in general does not build on the 'probability' of each path to occur. It is in fact quite complex, if at all possible, to determine for a particular path the probability with which it will be executed during operation [9] and to guarantee that the path frequency observed at analysis time, for a concocted set of input vectors, matches exactly that probability. With MBPTA, if a path can be executed at operation, its impact is conservatively factored in the pWCET estimate
  The Extended Path Coverage (EPC) [29] technique relieves the user from this stringent coverage requirement only relying on a set of measurements for each basic block (already requested in DO-178C [23] for DAL-A functions). EPC builds on the concept of *probabilistic path-independence* to characterize the probabilistic impact of unobserved paths on the set of observed execution times. The computed impact is then used to synthetically extend the set of observations to obtain the equivalent effect of full path coverage, while incurring a small amount of overestimation in comparison with standard MBPTA. Tool support for EPC has been fully implemented for an FPGA processor that has been developed in PROXIMA, seeking MBPTA-compliance by design. After a first positive evaluation on top of a cycle-accurate simulator reported in [29], EPC support is now being evaluated on the actual FPGA.
- *Dependent data*: The basic versions of EVT require independence conditions on the set of execution time measurements. However, this requirement may not hold if successive values are dependent in time, the distribution changes gradually over time, or the distribution changes periodically.
  By using appropriate methods for modelling block maxima for dependent data and threshold exceedances for dependent data, in the context of Generalized Extreme Value (GEV) and Generalized Pareto Distribution (GPD) estimators respectively, EVT can be used with dependent data. In this respect, by building two separate and independent estimation methods (therefore free of common-mode errors) we provide the validation arguments for the obtained pWCET estimate. The claim is that if the pWCET estimates obtained by the two methods are sufficiently close, then the obtained pWCET is indeed valid with respect to set of execution-time measurements provided.
- *Multicore analysis*: Resources shared by multiple cores are subject to interference resulting from parallel competing accesses or modifications. Inter-core interference constitutes an important part of the execution conditions that need to be controlled. Maximising the interferences observed at analysis time is extremely complex, as interfering co-runners would be required to conflict with each and every requests of the Unit of Analysis.

  1) Our first multicore MBPTA analysis variant relies on observations made with controlled co-runners that constitute a multi-variate model that relates the sources of inter-core interferences to a given impact. The inter-core interference model can then be used to derive an inflation factor that upper-bounds the possible impact of inter-core interference, including the worst case, on the Unit of Analysis in the actual system. The use of a MBPTA-compliant platform, offering randomised arbitration policies and isolation between the shared state space used by different cores, guarantees representativeness of each observed and predicted interference scenarios. The provision of the interferences-generating contenders during analysis is lifted from the end user.
  2) Our second variant, called partially time composable bound, builds on two elements. First, performance monitoring counters (PMCs) when running each task in isolation. These include execution time, bus accesses and memory accesses. And second, worst-latencies that a request from a given task can cause on a contender task. This is derived by deploying a set of specialized application kernels (or resource stressing kernels [15]). With these two pieces of information the model derives the worst contention that a task can suffer from another contender task without the need of simultaneously running them.

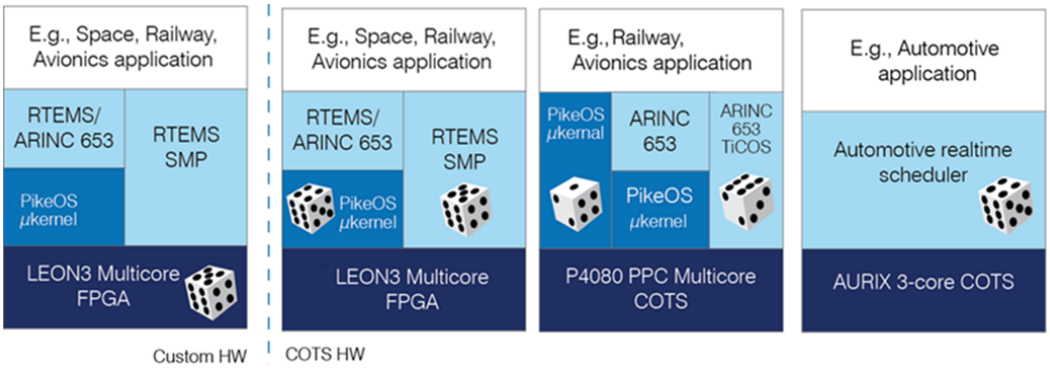

Figure 5. PROXIMA platforms.

- *Tool support* RVS is a framework of tools for on-target verification for embedded, real-time software. RVS includes RapiTime, a measurement based software timing performance tool. RapiTime automates the instrumentation of software for measurements, and processes data to identify WCET hotspots and potential WCET paths based on the actual observed measurements. RVS has been adapted for PROXIMA by the development of new instrumentation routines for gathering data from PROXIMA platforms, by incorporating the PROXIMA pWCET calculations into the standard workflow, and support for data post processing such as via EPC. In addition, RVS viewer has been adapted to include visualisation of the pWCET graphs.

B. MBPTA-enabling hardware

In PROXIMA, we have implemented a MBPTA-compliant 4-core processor FPGA prototype, starting from an RTL LEON3 processor description [27] enhanced with shared L2 cache, improved tracing support, and a per-core floating point unit (FPU). To achieve MBPTA compliance the following hardware modifications have been applied: all floating point operations can be selectively set to work at their worst-case latency; L1 data and instruction caches, TLBs implement random placement and replacement [18]. The same holds for the L2 cache, which is also partitioned across cores to be free from inter-core conflicts. Those modifications increase the FPGA resources consumption by a mere 2%. The next planned step is to implement random arbitration for the on-chip bus [17].

In PROXIMA we also address manycore processors. To this end, we have built a performance simulator that models an exploratory clustered manycore. Efficient networks-on-chip (NoCs) have been designed for tree-based intra-cluster communications and crossbar-based inter-cluster communications. The manycore processor simulator will be further extended to make it entirely MBPTA-compliant, thus dealing with memory and I/O access.

C. MBPTA-enabling software

PROXIMA also seeks to enable MBPTA compliance on top of Commercial Off-The-Self (COTS) processors. To this end, we have extended prior software-only randomisation solutions [19], [20] to attain MBPTA compliance on top of caches that implement modulo placement and LRU replacement.

Software-only randomisation solutions randomise the placement of objects in memory so that the resulting cache conflicts also take a random nature, independent of the actual location of the objects in memory. In PROXIMA we have devised two method to achieve this goal: one which operates on a single executable and causes it to take random placements; another, which builds multiple executables which differ in memory placement. The former is called *Dynamic Software Randomisation* (DSR); the latter *Static Software Randomisation* (SSR). DSR performs randomisation at runtime by placing objects dynamically in random locations with support from a combination of a compiler pass and a runtime library. SSR achieves the same effect in an entirely static manner by randomising the position of objects in the source code (and so in the binary), thus also leading to random memory locations. Whereas DSR randomises execution time across runs, SSR does so across binaries. Hence, the timing analysis process and interpretation of the results changes across techniques, although this is transparent to the user.

Currently, PROXIMA supports SSR on COTS LEON3 and AURIX TriCore processors, and DSR on the COTS LEON3 processor. DSR will also be readied for a P4080 processor.

D. Real-Time Operating System (RTOS)

RTOSes can provide useful help in meeting the MBPTA requirements. The solution embraced within PROXIMA builds on the concept of *time-composable* RTOS, which guarantees to only cause an additive contribution to the execution time of the application program, without causing it to incur additional jitter effects caused by history or data

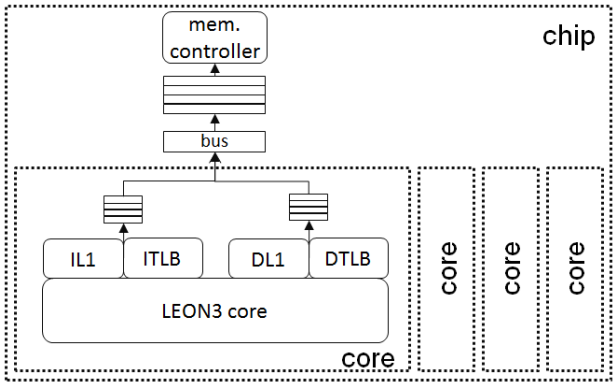

Figure 6. Reference architecture.

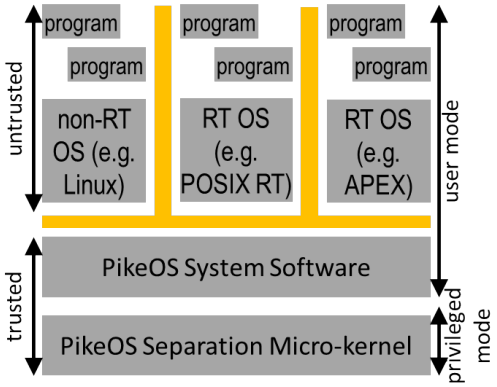

Figure 7. PikeOS framework.

dependence owing to RTOS interference action or services. We seek constant (or at least near-constant) timing behaviour for RTOS services, and prevent the RTOS from interacting with the inner state of shared hardware resources. The resulting time composability is achieved at the cost of a modest performance loss.

The RTOS-level solutions pioneered in PROARTIS [1] have been brought to maturity within SYSGO's PikeOS [2], which supports para-virtualization of multiple guest OSes with guarantees of isolation (see Figure 7). Time composability improvements have also been implemented on the guest OSes used in the PROXIMA case studies, namely ARINC-653, and native PikeOS.

For research purposes at lower Technology Readiness Level (TRL), we produced a time-composable version of more generic RTOSes, such as RTEMS in its recently-released multiprocessor variant, for use with the FPGA, a multicore port of TiCOS, an ARINC-653 compliant RTOS developed in PROARTIS, for use with the P4080, and Erika Enterprise[3], for use with the AURIX.

All of the PROXIMA RTOSes have also been modified to meet the instrumentation requirements entailed by the MBPTA technology, and to support the application of SW-only time randomization techniques.

E. Certification

PROXIMA addresses critical RTES, therefore, it must comply with the safety standards of the target critical industries. The PROXIMA team is developing a cross-domain argumentation considering the commonalities of the different safety certification standards considered in the project: Railway (EN-5012x), Automotive (ISO-26262), Avionics (DO-178) and Space (ECSS-Q-ST-x0C).

Within this cross-domain approach, the mathematical foundations of MPBTA are being demonstrated with rigour and authority sufficient to withstand independent review, together with the required answers of the representativeness question (including for the example, the quality and quantity of the input data to be collected during analysis) and the implications of the requiring hardware and software modifications. The certification approach in PROXIMA also undergoes domain-specific evaluations. In the case of the railway domain, for example, an early safety concept has been elaborated considering the use of MBPTA on a mixed-criticality and multicore scenario [5]. This concept, which also presents a first version of the mathematical foundation of the MPBTA process and the design of a hardware implementation of an on-chip Pseudo Random Number Generator (PRNG) to feed timing randomization [6], has been positively assessed by an external certification authority addressing SIL-4 integrity level within IEC-61508/EN-5012x standards.

[3]Erika Enterprise RTOS, http://erika.tuxfamily.org/drupal/.

V. PROXIMA Platforms and Early Results

PROXIMA seeks to increase the TRL of the MBPTA technology with respect to the status it had at the end of PROARTIS – the predecessor project of PROXIMA.

Execution Platforms. PROXIMA has developed two classes of execution platforms, depending on whether timing randomisation is achieved by hardware or software means. The former class includes custom processors; the latter COTS.

- Hardware-enabled randomisation. In this group, we find the LEON3 [27] based FPGA processor mentioned in section IV, in which randomization is injected through hardware modifications, as shown in the left part of Figure 5. In the said FPGA, the placement and replacement policies for all caches are time randomised, as well as the arbitration on access to hardware shared resources (such as the bus). Some other resources, such as the floating-point unit, which has a jittery response time dependent on the input values, can be set to always respond with worst latency.
- COTS with software-enabled randomisation. This group includes all platforms in which the hardware

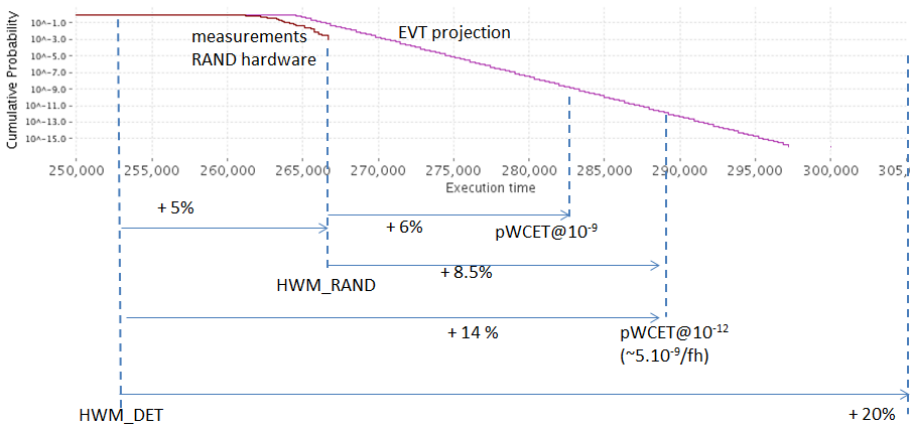

Figure 8. Results obtained and comparison with current practice.

is given cannot be customised to support MBPTA. For those processors, we achieve MBPTA compliance by injecting randomisation via software, with ad-hoc software randomization technology. In this group we find the same LEON-3 [27] based FPGA platform mentioned above, except that no modification has been applied to it, to keep equivalent to a COTS version. This variant enables us to compare the effectiveness of hardware-enabled and software-enabled solutions applied to one and the same processor. In PROXIMA we also use two further COTS platforms, with different degrees of hardware resource sharing: a FreeScale P4080-based board and an AURIX TC7XX-based board.

In all of the execution platforms, RVS is used to gather the execution-times measurement observations. RVS automatically instruments the source code so that measurements can be taken at specific points of execution. The source code is compiled and executed on the target, capturing the data to a series of time trace files. Further other information such as performance-monitoring counter or addresses traces are extracted thanks to manual modification of the application associated to dedicated custom exploitation scripting (not yet integrated in an unified tool suite). The trace data are filtered and used for Extended Path Coverage, followed by processing through the MBPTA calculation programs to create the pWCET curves. The data can then be viewed within the RVS viewer in the form of a series of graphs

On top of these platforms, a set of case studies in the avionics, space, railway and automotive domains are being run to assess the benefits of the PROXIMA approach.

Early Results. The selected application is Weight and Balance Back-up Computation part of the flight control system and in charge of computation of estimations of centre of gravity and weight of the aircraft.

We considered the FPGA platform comprising 4 LEON3 [27] processors, see Figure 6. Each core comprises a set-associative data and instruction caches as well as fully-associative data and instruction TLBs. The request from the instruction path and the data path are sorted in private buses before they are send to the shared bus and the memory controller. The caches implement random placement [18] and random replacement, the latter of which is deployed in many architectures. Fully-associative TLBs implement random replacement. Note that the effect of buffers has been shown MBPTA compliant [13]. For the purpose of this preliminary experiment, we show results for a single-core. This allows us to asses the impact of randomisation in the different caches and TLBs.

The results presented in this paper are related to the execution of this IMA application on top of PikeOS A653 personality and hypervisor running on the MBPTA compliant HW randomized FPGA one a single core with no opponents.

We execute 1,000 times the function under analysis on the target platform. This value ensures the results are representative, see last step in Figure 2. Execution time measurements are captured through specific instrumentation using GPIOs and off the shelf TraceBox hardware provided by Rapita partner. The traces are then processed by a modified RVS toolsuite implementing MBPTA.

We start by checking that the distribution of maximums obtained from the original population 1,000 execution time measurements passes the independence and identical distribution tests. We consider the commonly accepted significance value $\alpha = 0.05$ for both the two-sample Kolmogorov-Smirnov (KS) [14] test for identical distribution and the runs-test [8] for independence. To our best knowledge there is no proof for a precise significance value relevant to the WCET estimation problem. We also pass the Exponentiality Test (ET), which confirms that distribution of maximums converge to a Gumbel.

Figure 8 shows the EVT projection obtained with MBPTA and a comparison in terms of maximum-observed execution time, also known as high-water mark (HWM), and WCET

estimate with respect to a deterministic architecture. In Figure 8 we see the distribution of the execution time measurement collected on the reference randomised platform and the EVT projection obtained from that. The vertical dashed lines from left to right show: the HWM for the deterministic counter part of our reference architecture, the HWM for the randomised reference architecture, the pWCET value for an exceedance probability of 10^{-9} and 10^{-12}. The last vertical line shows the result of increasing by 20% the HWM observed for the non-randomised architecture.

- *HWM*. HWM is only 5% worse for the randomised architecture than for the deterministic one. This value is in the range obtained for average performance results that in general show that time-randomised architectures provide around 10% less performance than their time-deterministic counterparts.
- *WCET estimate*. In terms of WCET, the most important metric in RTES, we show that for exceedance probabilities at 10^{-9} and 10^{-12} the pWCET estimate provided by MBPTA are only 6% and 8.5% higher than the HWM of the randomised architecture. These results show that the pWCET curve computed by MBPTA slants towards the observed values, which provides good tightness.
  The pWCET bound corresponding to that exceedance threshold is 14% higher than the HWM observed on the deterministic architecture, hence better (because tighter and thus less pessimistic) than the WCET value computed with the current MBDTA techniques, which use $HWM+20\%$. Moreover, while MBPTA has a solid mathematical foundation, the latter approach based on the 20% adjustment does not, which affects its confidence for future architectures.

VI. Conclusions and Future Work

We have described the foundations of the PROXIMA MBPTA approach and the benefits it brings over the current practice with measurement-based timing analysis. We have also summarized the state of PROXIMA in the development of the technology apparatus required for the sound application of MBPTA. Finally, we have shown early results obtained with a real avionics case study on the PROXIMA tool chain on a FPGA platform in which bits of the hardware randomisation technology have been implemented. Until the end of the project we plan to consolidate the transition to multicores, both with customized designs in the FPGA and COTS designs.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROXIMA Project (grant agreement 611085). Carles Hernández is jointly funded by the Spanish Ministry of Economy and Competitiveness (MINECO) and FEDER funds through grant TIN2014-60404-JIN. Jaume Abella has been partially supported by the MINECO under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717.

References

[1] PROARTIS EU-FP7 project. http://www.proartis-project.eu, 2010.

[2] Pikeos website. http://www.sysgo.com/products/pikeos-rtos-and-virtualization-concept/, 2014.

[3] J. Abella et al. Heart of Gold: Making the improbable happen to extend coverage in probabilistic timing analysis. In *ECRTS*, 2014.

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

[5] I. Agirre et al. A safety concept for a railway mixed-criticality embedded system based on multicore partitioning. In *DASC*, 2015.

[6] I. Agirre et al. IEC-61508 SIL 3-compliant Pseudo-Random Number Generators for Probabilistic Timing Analysis. In *DSD*, 2015.

[7] I. Bate and A. Burns. An integrated approach to scheduling in safety-critical embedded control systems. *Real-Time Systems Journal*, 25(1):5–37, Jul 2003.

[8] J.V. Bradley. *Distribution-Free Statistical Tests*. Prentice-Hall, 1968.

[9] F.J. Cazorla et al. Upper-bounding program execution time with extreme value theory. In *WCET Workshop*, 2013.

[10] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

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

[12] Enrico Mezzetti et al. Randomized caches can be pretty useful to hard real-time systems. *LITES*, 2(1), 2015.

[13] Leonidas Kosmidis et al. Applying measurement-based probabilistic timing analysis to buffer resources. In *WCET Workshop*, 2013.

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

[15] Mikel Fernández, Roberto Gioiosa, Eduardo Quiñones, Luca Fossati, Marco Zulianello, and Francisco J. Cazorla. Assessing the suitability of the NGMP multi-core processor in the space domain. In *EMSOFT*, 2012.

[16] S. Hutchesson and N. Hayes. Technology transfer and certification issues in safety critical real-time systems. In *Digest of the IEE Colloquium on Real-Time Systems*, number 98/306, April 1998.

[17] J. Jalle et al. Bus designs for time-probabilistic multicore processors. In *DATE*, 2014.

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

[19] L. Kosmidis et al. Probabilistic timing analysis on conventional cache designs. In *DATE*, 2013.

[20] L. Kosmidis et al. Containing timing-related certification cost in automotive systems deploying complex hardware. In *DAC*, 2014.

[21] L. Kosmidis et al. Probabilistic timing analysis and its impact on processor architecture. In *DSD*, 2014.

[22] S. Kotz et al. *Extreme value distributions: theory and applications*. World Scientific, 2000.

[23] RTCA. DO-178C, software considerations in airborne systems and equipment certification, 2011.

[24] L. Santinelli et al. On the sustainability of the extreme value theory for WCET estimation. In *WCET Workshop*, 2014.

[25] https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-20.pdf. *Position Paper CAST-20: ADDRESSING CACHE IN AIRBORNE SYSTEMS AND EQUIPMENT*. Certification Authorities Software Team (CAST), 2003.

[26] https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-32.pdf. *Position Paper CAST-32: Multi-core Processors*. Certification Authorities Software Team (CAST), 2014.

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

[28] 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.

[29] M. Ziccardi et al. EPC: Extended Path Coverage for Measurement-based Probabilistic Timing Analysis. In *RTSS*, 2015.