Measurement-Based Probabilistic Timing Analysis for Multi-path Programs

Liliana Cucu-Grosjean$^{\Upsilon}$, Luca Santinelli$^{\Upsilon}$, Michael Houston$^{\Phi}$, Code Lo$^{\Upsilon}$, Tullio Vardanega$^{\Theta}$
Leonidas Kosmidis$^{\psi}$, Jaume Abella$^{\psi}$, Enrico Mezzeti$^{\Theta}$, Eduardo Quinones$^{\psi}$, Francisco J. Cazorla$^{\psi,\varsigma}$

$^{\Upsilon}$ *INRIA, France.* $^{\Phi}$ *Rapita Systems Ltd., UK.* $^{\Theta}$*University of Padua, Italy.*
$^{\psi}$ *Barcelona Supercomputing Center, Spain.* $^{\varsigma}$ *Spanish National Research Council (IIIA-CSIC), Spain.*

Abstract—**Probabilistic Timing Analysis reduces the cost of acquiring the required knowledge to obtain trustworthy WCET estimates, which is one of the main limiters to the application of static timing analysis to increasingly complex hardware platforms. In this paper, we present a sound measurement-based probabilistic timing analysis technique based on Extreme Value Theory. Our technique provides probabilistic WCET estimations with a pessimism of at most 15% with respect to the tightest possible WCET estimation obtainable with any probabilistic timing analysis technique. Moreover, our technique requires a low number of runs of the application on the target platform, 650 at most, making it more attractive to industry.**

I. Introduction

The high-assurance segment of the embedded systems industry is compelled to continually increase the functionality and performance of its products. This can only realistically be achieved by adding complex performance-enhancing features to the system architecture, as there is a competing drive to minimise cost and power consumption. However this is at odds with the common approaches to validating the timing behaviour of safety-critical systems.

Advanced hardware features, e.g. multiple cores and several cache levels, challenge a deterministic view of the system. While the behaviour of individual components is deterministic, the combination of possible hardware states makes precise modelling of worst-case behaviour difficult. Static timing analysis techniques are intrinsically limited by the complexity of constructing a sufficiently accurate model of the hardware, with consequences for the precision of a WCET estimate if the knowledge of state is incomplete. As the model becomes more detailed, the cost of computing tight WCET upper bounds becomes prohibitive, and in some cases infeasible.

End-to-end measurement-based analysis relies on extensive testing performed on the real system. High-coverage input data, designed to stress the system, is used during testing to record the longest observed execution time (a 'high watermark'). An engineering safety margin is then added to make allowances for the unknown. However, reasoning about the validity of the margin is extremely difficult – if possible at all – especially when the system may exhibit discontinuous changes in timing due to pathological cache access patterns, untested paths, or other unanticipated timing behaviour.

The cost of acquiring the required knowledge to perform trustworthy analysis can be significantly reduced by adopting a hardware/software architecture where execution timing behaviour minimises – or even eliminates – dependence on execution history, which is the main factor affecting the internal hardware state. Several approaches are possible, and in this paper we achieve this independence by introducing time-randomisation to architectural features which contribute to timing variability [6]. We couple this with new Probabilistic Timing Analysis (PTA) techniques. To some extent it may seem counter-intuitive, using 'probabilistic' bounds in systems requiring high integrity. However, the reality is that probabilistic modelling is a close match to the nature of the system. Let's assume an aircraft on with a computing system on which PTA is to be applied. The mechanical parts in the aircraft are designed with a probability of failure in mind. Effects such as radiation, repeated physical stress and extreme temperatures introduce a low, but finite probability of failure for the computing hardware itself. Hence, the system as a whole has a distinct probability of failure. In that sense, timing failures can be considered just another type of failure that the system may experience. The objective of probabilistic timing analysis is to provide WCET estimations which are 'safe enough' for the application, and keep the overall failure rate of the system below the domain-specific threshold of acceptability (e.g. 10^{-9} [2]).

Analysis approach: So far, two main approaches to PTA have been proposed. Static PTA (SPTA) derives a-priori probabilities from a model of the system [7]. While reducing the amount of information needed from the platform and program to perform the analysis, this variant still requires a significant amount of information about the internal behaviour of architectural components. The second variant of PTA, Measurement-Based PTA (MBPTA), requires much less information, which makes it more attractive for industry. MBPTA derives probabilities for execution times by collecting observations of end-to-end runs of an application running on the target hardware. While a description of an SPTA technique was given in [7], a MBPTA technique is still missing. Some efforts have been made in using Extreme Value Theory (EVT) for the problem of WCET estimation, but these methods are not based on appropriate assumptions about the statistical properties of the hardware as shown in [16], and are not obviously applicable by industry due to the simplified scenarios used.

In this paper, we present for the first time a MBPTA technique based on a sound application of EVT that allows the analysis of multi-path programs. We identify the requirements that this technique imposes on the experimental setup and target platform so that the probabilistic WCET (pWCET) estimations we obtain can be applied with high confidence as an upper bound on execution time, and hence can be

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROARTIS Project (www.proartis-project.eu), grant agreement no 249100.

used in hard real-time systems. We pay special attention to the number of observations required by the user in order to obtain statistical significance, and for EVT to provide high-quality pWCET estimations. Our results show that our EVT-based MBPTA provides pWCET estimates with less than 15% pessimism with respect to Static PTA, which provides the tightest pWCET estimations that can be achieved with probabilistic timing analysis. The number of execution time observations required is feasibly low, 650 observations (runs) at most for all the benchmarks used in this paper.

Paper organisation: Section II introduces EVT. Section III details the requirements and how to apply EVT to single-path programs. Section IV extends the approach to multi-path programs. Section V presents our experimental setup and the results obtained. In Section VI we present the related work. In Section VII we state the main conclusions of this study.

II. Extreme Value Theory for Timing Analysis

Let us assume a given processor and program for which we would like to provide pWCET estimates. Further assume that the execution of the program is observed 1,000 times and the execution times are collected. If those execution times can be modelled with independent and identically distributed (i.i.d.) random variables [19], then the pWCET of a program can be derived by collecting enough observations to construct an Empirical Cumulative Distribution Function, ECDF. From the inverse ECDF it is possible to compute an estimated pWCET as $\overline{pWCET}$ for a desired probability. Such an approach has an important requirement that we must have a good model of the tail-end behaviour of the ECDF; however, we would need a vast number of observations to model these low-probability events. In this case, as the number of observations is 1,000, the maximum confidence is $p(execution\ time > WCET_{estimation}) = 10^{-3}$.

An alternative approach is to predict from the ECDF the pWCET estimate with a probability smaller than 10^{-n}, where n is a required level of confidence. Different mathematical solutions may be proposed and in this paper we are interested in Extreme Value Theory (EVT). This theory estimates the probability of occurrence of extremely large values which are known to be rare events [9]. More precisely EVT predicts the distribution function for the maximal (or minimal) values of a set of n observations, which are modelled with random variables. Thus, this theory is analogous to Central Limit Theory but instead of estimating the average, EVT estimates the extremes [15].

Theorem 1. *[15] Let $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ be a sequence of i.i.d. random variables and let $\mathcal{M}_n = max\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$. If F is a non degenerate distribution function and there exists a sequence of pairs of real numbers (a_n, b_n) such that $a_n \geq 0$ and $lim_{n\to\infty} P(\frac{\mathcal{M}_n - b_n}{a_n} \leq x) = F(x)$, then F belongs to either the Gumbel, the Frechet or the Weibull family.*

The main result of EVT is provided in Theorem 1, where F denotes the common distribution function of n random variables. In our approach, random variables are used to model the execution time of programs. In order to apply Theorem 1 two main hypotheses must be verified: that the n random variables are independent and identically distributed (see Section III-B) and the existence of the sequence of real numbers (a_n, b_n).

The Gumbel, Frechet and Weibull are particular cases of the Generalised Extreme Value (GEV) distribution which has the following Cumulative Distribution Function (CDF):

$$F_\xi(x) = \begin{cases} e^{-(1+\xi\frac{x-\mu}{\sigma})^{\frac{1}{\xi}}} & \xi \neq 0 \\ e^{-e^{-\frac{x-\mu}{\sigma}}} & \xi = 0 \end{cases}$$

GEV is defined by three parameters: shape (ξ), scale (σ) and location (μ). By estimating these three parameters, we prove the existence of the sequence of real numbers (a_n, b_n). If the shape parameter $\xi = 0$ then F_ξ is a Gumbel distribution, which is the distribution we use in this analysis.

When EVT is applied, we must estimate its three parameters [14] and hence the appropriate CDF (Gumbel, Frechet or Weibull) is selected. To that end, a goodness-of-fit test is needed. However, not all tests are appropriate for fitting extreme values. For instance, in [18] the authors use the χ^2 test which is known to perform incorrectly for extreme values in any classical test[14]. One test that is proven correct when fitting the Gumbel CDF is the exponential tail (ET) test [14].

EVT has previously been applied to the WCET estimation problem in [9], [18]. The main limitations of these two papers, which are shown in [16], are the following: the assumption of independent and identically distributed random variables, and fitting a continuous distribution to discrete values. In Section III we correct the application of EVT to WCET estimation by proposing solutions to these limitations. Moreover, we also provide an alternative to the χ^2 test in [18] for fitting the Gumbel CDF. Finally, we show how to apply EVT to multi-path programs.

A. Hypotheses

In this section we detail the hypotheses needed for a correct application of EVT.

Independence and identical distribution: The main hypothesis for Theorem 1 is that the sequence of random variables is independent and identically distributed. We provide here the definition of this notion. In following sections we show how to achieve this property when applying EVT to obtain pWCET estimations.

Definition 1 (Independent RV)**.** *Two random variables $\mathcal{X}$ and $\mathcal{Y}$ are independent if they describe two events such that the occurrence of one event does not have any impact on the occurrence of the other event.*

Definition 2 (Identically distributed RV)**.** *A sequence of random variables is identically distributed if all random variables have the same probability distribution.*

We say that a sequence of random variables is independent and identically distributed (i.i.d.) if they have the same distribution function and they are independent from one another. A sequence of (fair) die rolls where each roll is a random variable is i.i.d., since the random variables describe the same event and the outcomes are obtained from independent events (as the outcome of one roll is independent of the outcome of another roll).

Selection of an EVT distribution: In order to apply Theorem 1, parameters describing a valid EVT distribution must be chosen (Gumbel, Frechet or Weibull). This requires an appropriate statistical test. Previous applications of EVT to WCET estimation [9], [18] indicate that the Gumbel distribution fits the problem of WCET estimation. We validate this hypothesis by checking whether a given ECDF fits the Gumbel CDF, for which we use the exponential tail (ET) test [14]. The ET test assumes that the data set is modelled by n i.i.d. random variables $\mathcal{X}_1, \cdots, \mathcal{X}_n$ described by the CDF F. The ET test compares this to the Gumbel CDF: $\mathcal{H}_0 : F = F_\xi$. If the test rejects the hypothesis $\mathcal{H}_0$, then the hypothesis $\mathcal{H}_1 : F \neq F_\xi$ must be true. The details for applying ET to our data are provided in Section V.

B. Steps in the application of EVT

There are two main steps in the application of EVT.

Grouping: The objective of this step is to collect, from the original distribution, the values which fit into the tail, and hence can be modelled with the Gumbel distribution. The values used as input for EVT are grouped into *blocks* of equal length m by applying the *block maxima* method. The maximum value observed in each block constructs a new sample, the block maximum series.

The size of the blocks determines the portion of the original distribution that is considered the tail. The larger the block size, the better the fit to the tail; however, increasing the block size results in fewer blocks and thus fewer values in the final sample. In the extreme case, if a single block is used, we will have only one block maximum value corresponding to the maximum of the original distribution. Conversely, if we use as many blocks as elements in the original distribution, the distribution of the block maximum values will resemble the entirety of the original distribution, rather than capturing the essence of the tail.

The theoretical basis of the block maxima method is built on the following theorem which says that the Gumbel CDF fitting to some data fits also to maxima of blocks of those data.

Theorem 2. *[10] [Grouping] Let $\mathcal{X}$ be a random variable. The GEV fitting to the n variables $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$ obtained from $\mathcal{X}$ fits to the maxima of any blocks obtained from $\mathcal{X}$.*

Distribution fitting: If the ET test shows that our observations follow a Gumbel distribution, we next estimate the two remaining parameters, μ and σ using the block maxima series. Their practical estimation is obtained by applying linear regression to the QQ-plot of our data [11]. A QQ-plot (quantile plot) is a plot of the empirical quantile values of observed data against the quantiles of the standard form of a target distribution. Given that the block maxima values follow a Gumbel distribution, then the points on the QQ-plot form a straight line [11]. The slope and the intercept of the best-fit line through these points can be used as estimators for the Gumbel μ and σ parameters, respectively.

III. EVT FOR SINGLE-PATH WCET

When applying EVT to the problem of providing pWCET estimates it is necessary to ensure that the assumptions made by EVT hold, so that the estimates obtained are sound. As we show in this section, these hypotheses have implications for the way experiments are run and the behaviour of the system. In particular, EVT assumes that the execution times of the system are independent between runs, and identically distributed.

We assume that the program for which a pWCET is to be estimated contains no system calls and that it is run in isolation. This is not a fully realistic scenario for production systems, but the assumptions match those used for current WCET analysis techniques; the interaction with other tasks and the operating system is taken into account during system integration. For this section, we further assume that the program contains a single-path of execution (in the next section we focus on multi-path programs).

In a fully deterministic system, a single execution path, when run from identical starting conditions, should have a single execution time. In practice this is not the case due to interference effects caused by hardware features such as DRAM refresh cycles, and in any case it is rare, or even impossible, to ensure that execution begins from identical starting conditions every time. Past approaches have attempted to use EVT to model this execution time variability, but there are no statistical guarantees about the nature of the variability. Rather than attempt to model such complex behaviour, our approach is to bring the behaviour closer to the model, thus ensuring that the variability in execution time conforms to distributions which can be modelled with statistical methods such as EVT.

By ensuring that timing behaviour of each source of variability is independent, the combination of latencies introduced will converge on average to a closed form distribution. In most cases this is the normal distribution due to the finite variance of the input distributions, according to Central Limit Theorem. Sources which cannot be modelled by independent random variables must be considered to always take their upper bound.

A. Input Data Dependence

For the single path case, with known, fixed input data on a simple architecture, we can exactly compute the distribution of execution times which will be observed when the path is executed by using a static probabilistic model of the hardware behaviour [7]. This provides an excellent way to evaluate the behaviour of EVT when applied to a time-randomised architecture, and gives greater confidence that the method is still sound when applied to more complex architectures.

In order to perform measurement-based analysis, the user must provide a range of input data to the program, designed to stress the program and produce worst-case behaviour. The choice of the data may affect the timing behaviour of the software, even if it does not affect the path taken. For example, dividing by some numbers takes more cycles than others, and data structures which contain pointers may refer to the same or different memory locations depending on the input, which affects the memory access pattern, and hence cache latencies.

Users cannot be asked to provide the 'worst' or even statistically representative data during the testing phase; they do not know (in general) what this data will be at deployment time, so we must design the platform in a way which prevents any temporal benefit being taken from input data.

There are hardware proposals which force operations, such as divisions, to always take their upper-bound execution time, which is a safe (pessimistic) assumption [21]. In [21] the authors introduce a 'Worst-Case Mode', that simply delays each request to a resource until its maximum execution time is observed. This is configurable by software and once active this technique is transparent to the executing programs. For the purposes of this paper such a technique has to be present in the target architecture. It will be activated during the collection of execution times and can be switched of during deployment.

A similar solution can be applied to data-dependent memory accesses which must be flagged so that effects due to the input data can be accounted for. Most current Instruction Set Architectures (ISAs) include some hint bits in the operation code of memory operations, which are used, between others, to advise the processor whether to cache some data or to activate the data prefetcher. In the former case, if the designer detects that the data for a given load is going to have a single use, it activates this bit for the opcode of the load. Analogously, in our case, input data dependent memory operations have to be identified in the object code and flag them by means of these hint bits to prevent cache or other performance-improving features being used for these instructions.

B. Independent and Identically Distributed Observations

In order for the i.i.d. properties to hold in our case, the architecture we propose must ensure that each execution run is independent. For the purposes of this experiment, we will ensure that runs are independent by forcing the state of the hardware to be reset before each run begins.

Identical distribution at the path level requires only that the latencies which make up the total execution time are independently chosen each time a path is executed. Each possible execution time that a path can take must have an associated probability. Note that this does not require that the probability distribution for an instruction is independent of the sequence of preceding instructions, only that the observed timing behaviour in one run does not affect the timing behaviour of other execution runs.

To achieve this goal certain processor resources, such as caches, must be time randomised [7]. A more detailed description of the hardware requirements is out of the scope of this paper and is presented in [7]. The architecture we use for this study (see Section V-A) presents the required properties to provide i.i.d. timing behaviour.

C. Continuous vs. Discrete Functions: minimum number of observations

In [16] it is shown that using the Gumbel distribution function, or any continuous function, to approximate a discrete function of the execution time values produced by a program, could yield unsafe results which do not bound the execution time from above at all points. In [16] there are proposed different solutions to this problem by either fitting the continuous function to the upper bound of the excedeence distribution function and thus overestimating the discrete function, or by adding an offset to the distribution which accounts for this effect. For instance for the single path we consider here, the maximum offset required is likely to be a small number of cycles.

In our paper we introduce an overestimation of the discrete function when applying EVT (Section II-B) through the block maxima method before the distribution fitting is applied. This produces a shift of the distribution, which makes the worst-case estimate pessimistic when compared to the 'real' distribution of execution times in our architecture. In order to obtain high confidence in such a distribution, we require that the number of observations is sufficient to produce an accurate description of the program's execution time. We call this the **minimum number of observations**, and we define it as follows:

Definition 3. *For a given sequential program P and an architecture A, n is the minimum number of observations if there does not exist $m \in \mathbb{N}$ where $m > n$ such that the Gumbel CDF obtained for n observations is an upper bound for the WCET of P and also exceeds the CDF obtained for m observations.*

We obtain the *minimum number of observations* by comparing the variation in the EVT tail projection as we consider an increasing number of runs. We follow an iterative process where 1) we start by running $N_{current} + N_{delta}$ times the program under study on the target platform. 2) We then, make two tail projections with EVT one using $N_{current}$ runs and the other using $N_{current}+N_{delta}$. 3) If the difference between the two EVT distributions is below a given *difference threshold* we stop the process. If it is over the threshold, we consider all previous runs as $N_{current}$, or in other words we make $N_{current} = N_{current} + N_{delta}$, and we make N_{delta} more runs, consider the part of the sample and repeat the process from step 2.

As we keep adding more runs to $N_{current}$ the proportional effect that N_{delta} new runs has on the EVT reduces, so a convergence of the algorithm is ensured. On the other hand, there is no guarantee that there is a strict convergence so we can have some local minima. In order to take into account this non strict convergence, we change the algorithm so that instead of stopping the process as soon as the difference between the two EVT distributions is below a given difference threshold, we stop the process when this is the case in several consecutive iterations. For the benchmarks used in this paper, considering 5 consecutive iterations below the threshold is sufficient to deal with the hysteresis introduced by local minima.

The metric we use to compare the two EVT distributions (functions) is called the continuous rank probability score defined as $CRPS = \sum_{i=0}^{+\infty}[f_{\mathcal{X}}(i) - f_{\mathcal{Y}}(i)]^2$ [5]. This test is defined for any two distribution functions $f_{\mathcal{X}}$ and $f_{\mathcal{Y}}$ as long as they operate on the same value domain. In this paper we set as difference the threshold 10^{-1}. This threshold has to be interpreted as the significance level α in hypothesis tests. The lower its value the larger is the confidence on the result. It is up to the user of our technology to set a proper value to this difference threshold. However, note that the lower the threshold the higher the required number of runs (observations).

D. Step-by-step application of EVT

EVT is applied to single-path programs by following these steps for a number of rounds until a stable result is obtained:

1) *Collect observations* – The first task is to gather a number observations of the execution time of the program. We suggest starting with around 100 values. In each round, an additional N_{delta} observations are made and included in the data used for the next steps. This process is described in Section III-C.

2) *Grouping* – Block maxima sampling, described in Section II is used to convert the measured frequency distribution into a worst-case distribution suitable for application of EVT.

3) *Fitting* – The Gumbel distribution which best fits the observation is estimated. For this we use parameter estimation based on [14] as shown in Section II.A.

4) *Comparison* – If this is not the first round, the 1-CDF for the current round is compared to the result of the previous round using the CPRS metric.

5) *Convergence* – When the CPRS metric reports values below a given threshold (0.1 in the examples in this paper) for a number of consecutive rounds (5), we consider the distribution to have converged, and no more observations need to be collected.

If the distribution has not yet converged, the process starts another round at Step 1.

6) *Tail extension* – The final parameters for the Gumbel distribution are used to compute the execution time values for probabilities down to a lower bound. In our examples we use 10^{-16} as this is the limit of precision for floating point numbers in our prototype tool. Greater precision can easily be obtained using an arbitrary-precision floating point implementation such as MPFR [12].

This process is used to produce the results in Section V-C.

IV. EVT FOR MULTI-PATH WCET

The application of EVT to multiple paths is straightforward in principle, if 1) it can be ensured that the resulting compound distribution obtained from multiple paths still exhibits the necessary i.i.d. properties. Moreover, two more restrictions have to be imposed on the strength of the claims which may be made about the results, specifically 2) a sufficient number of observations must have been made and 3) the result is only applicable to a set of observed paths.

Independent and identically distributed observations In order that the i.i.d. property holds, it is important that the input data (and hence the path) for each run is chosen randomly during the block maxima method stage. This may be achieved either by selecting inputs randomly when running the tests and grouping them sequentially, or by testing all inputs and selecting results randomly (without replacement) when performing grouping.

Minimum Number of Observations. EVT is insensitive to the relative frequencies of different paths in the measured data: the process of extracting the block maxima distribution allows the data corresponding to worst-case paths to dominate the result. There is however a requirement that each path has been observed a minimum number of times to allow the behaviour to be sufficiently well characterised (see Section III-C). One possible way to ensure this property is to execute each input from the test set at least this minimum number of times.

Path Coverage. Extending this problem in another direction, the result is only valid for the paths which are observed. Claims on the pWCET for untested paths cannot be made, as the execution times for those paths would no longer be drawn from an identical distribution. This inherently couples the concept of i.i.d. with the path coverage achieved by the input data.

The issue of path coverage introduces a problem for the usefulness of the result, as full path coverage in complex applications is infeasible: the combination of loops and branches creates an explosion in the number of possible paths through the program. Therefore we must impose a caveat on the result obtained by EVT: that it is valid only for a set of 'observed paths', that is, those paths covered by the tests.

In order to apply our method in a realistic scenario therefore, it will be necessary to strike a balance between precision and practicality. This may be achieved by limiting the scope of the program being considered for pWCET estimation. For instance, the units over which EVT may be applied could be defined as the largest partitions of the control-flow graph over which full path coverage has been observed. Once estimates have been calculated for these units, the results may be combined using a tree-based composition, as in [4]. This allows the testing burden to be reduced, at the expense of increasing pessimism in the overall estimate. Despite the limitations, this approach allows greater confidence than other measurement-based techniques that the underlying model supports such a probabilistic, composable approach.

A. Step-by-step application of EVT

The step-by-step application of EVT was described for single-path programs in Section III-D. The only change for multi-path programs is to Step 1, collection of observations.

With multiple paths, each path must have sufficient observations to characterise its behaviour. Therefore, the number of observations required is multiplied by the number of paths. This means that we should start with, for example $100 \times P$ where P is the number of inputs, assuming that each input exercises a distinct path.

We should also increase the number of additional observations in each round to $P \times N_{delta}$ so that the paths each have a sufficient number of observations to determine a change in the estimated distribution.

V. Experimental Evaluation

In this section we begin by presenting the experimental environment we use to evaluate our analysis technique. Next, we introduce the SPTA as probabilistic based timing analysis technique for single-path programs. That is applied as reference for some experiments. Finally, we present the results, in terms of pWCET estimations, that we obtain for both single-path and multi-path programs.

A. Experimental Set-up

One of the requirements that EVT imposes on the data to be analised is the i.i.d. hypothesis. In our scenario this means that the random variables describing the execution times of the program under analysis have to be i.i.d.. One of the ways to achieve this is by randomising the timing behaviour of the underlying platform [7]. Our MBPTA technique (named EVT-MBPTA) applies, by construction, to any architecture ensuring this property. The processor architecture we use in this paper is an example of such architecture and its properties are tested through appropriate tests.

We run all experiments on a modified version of SoCLib [1], an open platform for virtual prototyping of multi-processors system on chip (MP-SoC). We model a pipeleined processor featuring data and instruction caches. In particular, we use 4KiloByte, 1024-way, 4-byte line, fully-associative data and instruction caches deploying a random-replacement policy. The latency of each instruction is fixed, except for the first stage (fetch) in which the instruction cache is accessed. The latency of the fetch stage depends on whether the access hits or misses in the instruction cache: a hit has 1-cycle latency and a miss has 100-cycle latency. After the decode stage, memory operations access the data cache so they can last 1 or 100 cycles depending on whether they miss or not. Overall, the possible latencies of a non-memory operation are (N+1, N+100) depending whether it hits or not in the instruction cache, where N if the number of cycles it takes executing the instruction (e.g. integer additions take 1 cycle). Memory operations have 3 possible latencies (1+1, 1+100, 200). 2 cycles in case it hits both instruction and data caches, 200 in the opposite case and 101 when it hits only one of both caches.

Benchmarks. We use eight benchmarks of the EEMBC Autobench suite [22] as reference benchmarks for the analysis of single-path programs. EEMBC Autobench is a well-known benchmark suite that reflects the current real world demands of some embedded systems. There are 8 EEMBC Autobench benchmarks: a2time (AT), aifirf (AI), cacheb (CA), canrdr (CN), puwmod (PU), rspeed (RS), tblook (TB) and ttsprk (TT).

For multipath-program analysis we use several Mälardarlen benchmarks [17], which are commonly used in the community to evaluate and compare different types of WCET analysis tools and methods. In particular we used: bs (BS), cnt (CNT), compress (COM), crc (CRC), insertsort (INS), qsort (QSO) and select (SEL). For each of these programs we developed several input sets that exercise different paths.

We also designed a multipath syntentic benchmark to better understand how the multipath-program time analysis works. In the synthetic benchmark we know (1) the different paths of the program, (2) the different input set that exercise each path and (3) the longest path. Thus we may compare the pWCET estimation we obtain with our single-path time analysis when applied to the worst case path against the pWCET estimation we obtain with our multiple-path time analyis when applied to a set of paths which includes the worst case path.

B. SPTA

Static probabilistic time analysis was introduced in [7] and it is used in this paper as a reference point with which to compare our MBPTA approach. Note, however, that our EVT-MBPTA *does not use* SPTA to derive any pWCET estimation. Instead we use it to evaluate the pWCET estimations provided by EVT-MBPTA since SPTA provides the precise execution time distribution of the program, that is, the tightest pWCET estimation that can possibly be obtained with probabilistic timing analysis techniques will correspond to the SPTA distribution. However, this comes at the cost of requiring more information about the target platform and the program under study than our EVT-MBTA approach.

In order to use SPTA it is necessary to derive the possible latencies every instruction may take and the probability associated with each latency[1]. This information is usually represented in the form of an execution time profile. Listing 1 shows an example for a sequence of instructions, where i_h, i_m are the probability of hit and miss in the instruction cache respectively; and d_h, d_m are the probability of hit and miss in the data cache.

```
1:  {(2,             101,                            200),
      (i_h0 + d_h0,   i_h0 * d_m0 + i_m0 * d_h0,      i_m0 + d_m0)}
2:  {(2,             101),
```

[1]Note that SPTA also requires that execution times for individual instructions are independent, but not identical distribution between different instructions .

```
   ($i_h1$,          $i_m1$)}
3: {(2,              101,                 200),
   ($i_h2+d_h2$,     $i_h2*d_m2+i_m2*d_h2$,    $i_m2+d_m2$)}
```

Listing 1. Execution time profile for a sequence of instructions. Two access memory (instruction 1 and instruction 3) and one does not (instruction 2).

Each line contains the execution time profile for one instruction: the first tuple are the latencies in the event each possible hit/miss combination occurs. As explained above the possible latencies of a non-memory operation are (X+1, X+100) depending on whether the instruction cache access is a hit or not. In the case of the second instruction in Listing 1, X – the inherent instruction latency – is equal to 1. Memory operations, i.e. instructions 1 and 3 in Listing 1, have three possible latencies: (2, 101, 200).

The numbers following this group are the probabilities of observing the corresponding latencies. For a fully-associative random-replacement cache such as the ones used in this paper, these probabilities can be computed easily for each instruction with the formula [25][7] $P_{hit} = \left(\frac{N-1}{N}\right)^k$ where k is the reuse distance and N the number of sets in the cache.

Based on this trace of execution time profiles, we start an iterative process in which we *convolve* them [7]. The convolution of two execution time profiles leads to a new one in which all possible pairs of execution times from the two execution time profiles are added and the probabilities multiplied. For example, let $E_1 = \{(2, 101, 200),\ (0.1, 0.4, 0.5)\}$ and $E_2 = \{(2, 101),\ (0.6, 0.4)\}$ be two execution time profiles.

Then their convolution is:

$$E_r = \{(2{+}2,\ 2{+}101,\ 101{+}2,\ 101{+}101,\ 200{+}2,\ 200{+}101),\ (0.1{\times}0.6,\ 0.1{\times}0.4,\ 0.4{\times}0.6,\ 0.4{\times}0.4,\ 0.5{\times}0.6,\ 0.5{\times}0.4)\}.$$

And given that $101 + 101 = 2 + 200$ we collapse all duplicate pairs resulting in:

$$E_r = \{(4,\ 103,\ 202,\ 301),\ (0.06,\ 0.28,\ 0.46,\ 0.2)\}.$$

This process is repeated for all the instructions of the program. The final execution time profile is the complete execution time distribution of the program, including the worst-case behaviour in the tail. This can be used to make pWCET estimates by examining the exceedence function.

C. Results for Single-Path Programs

In order to obtain sound pWCET estimations through the application of EVT, the platform/data under consideration has to meet certain statistical properties. One of the ways to check whether this is the case is via *hypothesis testing*.

Data fits the Gumbel distribution. We start with the hypothesis that our execution time observations fit the Gumbel distribution: $\mathcal{H}_0 : F = F_\xi$. To that end we use the ET test [14] as introduced in Section II-A.

When applying the ET test, we check a relation that is proved equivalent to the hypothesis $\mathcal{H}_0$ in [14]. This relation verifies that a certain parameter $\hat{q}_{ET,n}$ calculable for any set of data belongs to an imposed interval. This interval is also proved calculable for any set of data. For all our data we observed that $\hat{q}_{ET,n}$ belongs to the expected interval.

Table I
P-VALUE FOR THE IDENTICAL DISTRIBUTION TEST FOR DIFFERENT VALUES OF m FOR A SAMPLE OF N=10,000 OBSERVATIONS. Z-VALUE FOR THE INDEPENDENCE TEST

	AT	AI	CA	CN	PU	RS	TB	TT
m	Identical distribution test (p-value)							
100	0.34	0.11	0.13	0.41	0.99	0.67	0.77	0.89
500	0.36	0.81	0.24	0.62	0.55	0.84	0.21	0.57
1000	0.16	0.75	0.91	0.88	0.71	0.16	0.92	0.36
N	Independence test (Z-value)							
10,000	0. 66	0.81	0.92	0.84	0.69	0.73	0.76	0.81

Independent and identically distributed observations. The application of EVT also requires that the data are i.i.d., which is ensured by a combination of hardware with suitable randomisation properties, as discussed in [7], and a suitable experimental methodology when observing the executions of the program under consideration (see Section III).

We start by checking that the data are identically distributed. We apply the two-sample Kolmogorov-Smirnov [24] test in which any two sample sets are compared. The Kolmogorov-Smirnov (KS) statistic quantifies a distance between the empirical distribution functions of two samples. The *null hypothesis* $\mathcal{H}_0$ is that the both samples are identically distributed and the *alternative hypothesis* $\mathcal{H}_1$ is that the samples are not identically distributed. For our experiments we use a *significance level* $\alpha = 0.05$, which is a common choice in this type of test. The outcome of the test is called the *p-value*. If the p-value is higher than α the null hypothesis cannot be rejected, which means that both samples are identically distributed.

Table I shows the p-value obtained by applying the KS test to the execution times obtained for a sample of 10,000 observations. From this *original sample* we create two *smaller samples* of m elements. For this experiment we use three values for m: 100, 500 and 1000. We populate the smaller samples by randomly taking elements from the original sample, which ensures that the smaller samples maintain the same statistical properties as the original [24]. In our case, the property of interest is the distribution. Then we apply the two-sample KS test to the two smaller samples. In Table I we observe that in all cases the p-value is higher than 0.05 which indicates that data are identically distributed.

In order to prove that samples are independent we use the *runs test* for randomness [5]. This test is used to check whether a series of binary events can be considered to be randomly distributed, and hence independent. In this test, the null hypothesis is that the data are randomly distributed (independent) and the alternative hypothesis is that the data are not randomly distributed (not independent).

A run is defined as a sequence of increasing values or a sequence of decreasing values. The number of values is the length of the sequence. In a random data set, the probability that the $(i+1)$th value is larger or smaller than the ith value follows a binomial distribution, which forms the basis of

Table II
MINIMUM NUMBER OF OBSERVATIONS PER EEMBC AUTOBENCH BENCHMARK

AT	AI	CA	CN	PU	RS	TB	TT
300	650	400	450	500	300	500	300

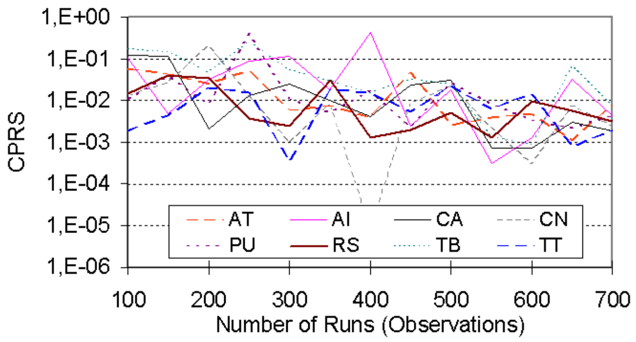


Figure 1. Benchmark CRPS variation as we increase number of runs. $N_{delta} = 50$ and threshold= 0.1

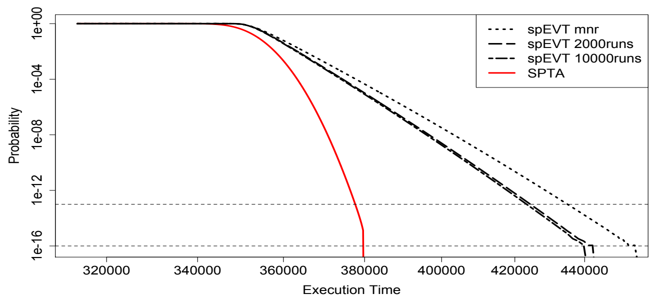
the runs test [5]. The first step is to compute the sequential differences $(d_i - d_{i-1})$: positive values indicate an increasing value and negative values indicate a decreasing value.

From the differences we compute the expectation for number of runs R, given by $E(R) = 2mn/N$, where m is the number of sequences of consecutive positive values, n the number of sequences of consecutive negative values, and N is the total sample size. The variance of the number of runs R is also computed: $V(R) = \frac{2mn\times(2mn-N)}{N^2\times(N-1)}$.

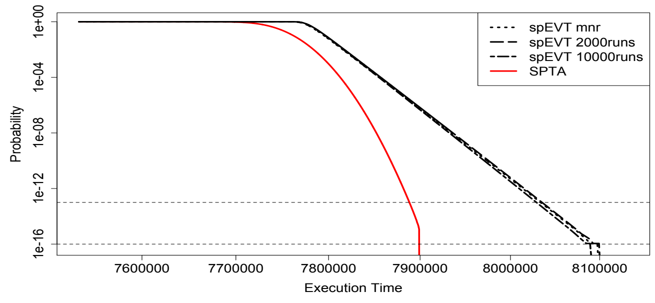
Let r be the number of values in the sample. It is known that when m or n tend to infinity then $Z = \frac{r-E(R)}{\sqrt{V(R)}} \to N(0,1)$, where $N(0,1)$ is the standard normal distribution [5]. This means that the test statistic, Z, is asymptotically normally distributed. Hence, we compare Z with a standard normal table for a given significance level α. At the 5% significance level, a Z-value with an absolute value greater than 1.96 indicates non-randomness so the null hypothesis is rejected.

The last row in Table I shows the Z-value computed for each EEMBC benchmark with 10,000 runs. We observe that for all the benchmarks under consideration the runs we collect from them are independent.

Minimum Number of Observations Figure 1 shows the variation of CRPS as we add more runs using $N_{delta} = 50$. In order to obtain the minimum number of observations we use the algorithm presented in Section III-C, which focuses on the convergence of the EVT tail projection as more runs are considered, taking CPRS as the convergence metric. We set the threshold to 0.1 and N_{delta} to 50. We observe that although CPRS decreases, it is not monotonically decreasing. For this reason our convergence algorithm takes into account small variations. Table II shows the minimum number of runs required for each EEMBC benchmark. On average we need 425 runs per benchmark. The average time required to run an EEMBC benchmark on our simulation infrastructure is about 1 minute, so the total time required was $425 \times 1 \times 8/60 = 57$ hours. This is equivalent to less than one hour of simulations in a 64-node cluster. It is worth noting that for building Figure 1 we used 700 runs, which provides satisfactory convergence for a threshold of 10^{-1}. However, to achieve convergence at lower CPRS values, e.g. 10^{-3} we would need to increase the number of observations.



(a) rspeed



(b) aifirf

Figure 2. Single-path pWCET estimation for two benchmarks

pWCET estimations. Figure 2 shows the pWCET estimations provided by MBPTA for two EEMBC benchmarks, *rspeed* and *aifirf* (due to space constraints we cannot show pWCET estimations graphically for all benchmarks). In Figure 2 the x-axis shows the pWCET estimation and the y-axis shows the associated probabilities.

Note that SPTA and EVT can be projected to arbitrarily low probabilities. For the purpose of illustration, we define our range of interest for the probabilities as $[10^{-13}, ..., 10^{-16}]$. This is based on the observation that for the aerospace commercial industry at the highest integrity level DAL-A the maximum allowed failure rate in a piece of software is 10^{-9} per hour of operation, [2]. In current implementations, the highest frequency at which a task can be released is 20 milliseconds (2×10^{-3}) [2]. Hence, the highest allowed failure rate per task activation is 2×10^{-11}, which is covered by our range of probabilities.

We show pWCET estimations for our single-path (sp) EVT technique when using the calculated minimum number of runs (mnr), 2000 and 10000 runs. For comparison purposes we show the exact execution time estimation we obtain with SPTA. We observe that, for every WCET value, we can compute its exceedance probability. The probabilistic WCET function is smooth, that is, it does not present abrupt changes. Moreover the EVT curve is an upper-bound to the SPTA

Table III
MBPTA pWCET estimations for EEMBC benchmarks w.r.t. SPTA

probability	AI	AT	CA	CN	PU	RS	TB	TT
10^{-13}	9%	5%	6%	5%	5%	2%	2%	6%
10^{-16}	15%	7%	8%	7%	7%	3%	3%	9%

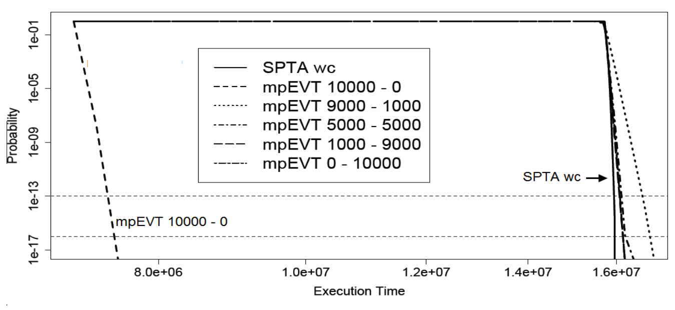


Figure 3. pWCET estimations for the synthetic benchmark

curve. The pessimism introduced increases as the probability decreases: for probability 10^{-13} it is 2% and 9% respectively for *aifir* and *rspeed*, increasing to 3% and 15% respectively for a 10^{-16} probability. We also observe that the pWCET estimations provided by EVT have little variation for numbers of runs higher than the minimum number of runs: observed variations are smaller than 3% for all benchmarks for 10^{-13}.

Table III shows the pWCET estimations provided by our EVT technique with respect to the estimations provided by SPTA for the selected EEMBC Autobench benchmarks. We observe that for the 10^{-13} confidence level, the overestimation is less than 9% and for 10^{-16} it is less than 15% for all benchmarks, which is quite acceptable.

D. Results for Multi-Path Programs

pWCET estimations for the synthetic benchmark. In order to check the results EVT-MBPTA obtains for multi-path programs we use the synthetic benchmark presented in Section V-A. This is a multi-path program for which we know the worst-case path, which is significantly longer than any other path. As a reference comparison point we take the pWCET estimation we obtain with our single-path technique when applied to the worst path. We compare it with the pWCET estimations we obtain when apply the multi-path technique to different sets of paths in which we include the worst path. For all the experiments in this section, we also applied the test for independence and identical distribution described in the previous section.

Figure 3 shows the pWCET estimation when varying the number of runs for the worst path and for the non-worst paths. We label each estimation as xxx-yyy where xxx is the number of runs we consider of non worst paths and yyy the number of runs from the worst case path. For instance, 0-10000 is the pWCET estimation when we consider 10000 runs of the worst path and 5000-5000 is the pWCET estimation when we consider 5000 runs of non worst-case paths and

Table IV
MBPTA pWCET estimations for the Malardalen benchmarks. Only worst-case path w.r.t. all paths

probability	BS	CNT	COM	CRC	INS	QSO	SEL
10^{-13}	7%	2%	4%	1%	9%	7%	5%
10^{-16}	8%	2%	5%	1%	11%	8%	6%

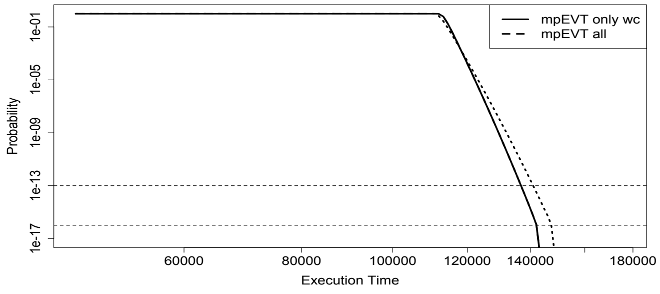


Figure 4. pWCET estimations for the *select* benchmark

5000 runs of the worst-case path. Note that, for every path we consider we make at least the minimum number of observations.

We compare all estimations with the result we obtain with SPTA for the worst-case path (*SPTA wc*). First, and most importantly, we observe that all multi-path estimations that consider the worst path are an upper bound for the SPTA. In the particular case of the 10000-0 estimation, we consider no runs from the worst path in the EVT. This case illustrates our claim in Section IV: pWCET estimates for untested paths cannot be made, as the execution times for those paths would no longer be drawn from an identical distribution. The other EVT estimations that consider the worst path are all an upper bound for the SPTA and have a small variation between them, which indicates that the frequency at which the worst-case path is exercised with the given input data set affects the provided WCET estimation only slightly.

pWCET estimations for real benchmarks. We compare here the pWCET estimations we obtain for different Malardalen WCET benchmarks. As in the previous experiments, the data passed all Gumbel fitting and i.i.d tests. We compare the pWCET estimation when we consider 10,000 runs from the worst-case path (wc) against the case in which we consider 10,000 runs from all paths including the worst-case path (all). Figure 4 shows the results for the *select* benchmark. Again we observe that as soon as the worst-case path is considered as part of the data set, EVT provides results comparable to considering only the worst-case path. Table IV shows the pessimism of the latter over the former, which is always less than 11% for the benchmarks we have tested.

VI. Related Work

The first papers in the real-time community with bearing on our work used the terms *stochastic analysis* [13], *probabilistic analysis* [23], *statistical analysis* [3] and *real-time queuing theory* [20] interchangeably. Since the publication [8], the notion of *stochastic analysis* of real-time systems has been used regularly by the community, regardless of the approach.

In this paper we use the terms *probabilistic analysis* or *statistical analysis* depending on the approach on which our solution is based. While the former provides the probability or chance of occurrence of future event, the latter searches for a model or some properties when studying some (often large) mass of data of observed past events.

As explained in previous sections, some previous proposals have used EVT to provide WCET estimations [18][9]. The first paper considering the statistical estimation of worst-case execution times is [9], in which extreme value statistics are used to model the WCET. More recently [18] improved the analysis provided in [9] by addressing some flaws in the previous work and produced a refined method also based on extreme value statistics (we develop more on this flaw below). In this paper we cover the problems shown in [16] on the way EVT is applied in the former two papers: the hypothesis of independent and identical distributed random variables and the continuity of the Gumbel distribution function. We also provide a solution to the wrong utilisation of the χ^2 test in [18] to best fit the Gumbel distribution function, since this test is known not to perform correctly for extreme values.

This paper is also the right place to underline that one of the 'mistakes' found by [18] in [9] are not really mistaskes: [18] notes that [9] does not apply EVT to estimate the WCET, but the execution time of a program. That is, the execution times considered when apply EVT are not the high (worst-case) execution times but all observed times. To solve this problem [18] introduces the block maxima technique when EVT is applied, so that in each block only the high (worst) observed execution time is taken. Both approaches are fine, since in reality in [9] the authors use an Exceedance Distribution Threshold Method (EDTM), that selects the largest execution times. In practice, EDTM is known to be less robust to small number of observations than EVT with the block maxima method.

VII. Conclusions

Probabilistic techniques reduce the cost of acquiring the required knowledge to perform trustworthy WCET analysis, which makes them attractive for industry. In this paper we have presented for the first time a measurement-based PTA technique based on a statistically sound application of EVT which enables the analysis of multi-path programs. We have also identified the statistical properties which hardware and software must demonstrate in order to ensure that probabilistic WCET estimations are sound. We have shown the appropriate tests with which to check whether those properties are present. We have also given a step-by-step description of how to apply our EVT-MBPTA technique for both single-path and multi-path programs.

Our results show that, for failure probability levels smaller than required in the highest software criticality levels in commercial avionics (i.e. 10^{-9} in DAL-A) our EVT-MBPTA single-path technique provides pWCET estimations less than 15% higher than Static PTA. This is an acceptable overestimation considering that SPTA provides the exact distribution of execution times of the program. For multi-path programs EVT-MBPTA provides tight pWCET estimations and presents the key feature that it is independent of the frequency in which the paths under consideration are exercised. This significantly reduces the requirements on the user of our technology, while providing a tight upper bound for pWCET estimations.

References

[1] SoCLib. http://www.soclib.fr/trac/dev.

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

[3] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *the 19th IEEE Real-Time Systems Symposium (RTSS98)*, pages 4–13, 1998.

[4] Guillem Bernat, Antoine Colin, and Stefan Petters. pwcet: A tool for probabilistic worst-case execution time analysis of real-time systems. Technical Report YCS-353-2003, Department of Computer Science, The University of York, 2003.

[5] Bradley. *Distribution-Free Statistical Tests*. Prentice-Hall, 1968.

[6] Alan Burns and David Griffin. Predictability as an emergent behaviour. 2011.

[7] Francisco J. Cazorla, Eduardo Qui nones, Tullio Vardanega, Liliana Cucu, Benoit Triquet, Guillem Bernat, Emery Berger, Jaume Abella, Franck Wartel, Michael Houston, Luca Santinelli, Leonidas Kosmidis, Code Lo, and Dorin Maxim. Proartis: Probabilistically analysable real-time systems. Technical Report 7869 (http://hal.inria.fr/hal-00663329), INRIA, 2012.

[8] J.L Díaz, D.F. Garcia, K. Kim, C.G. Lee, L.L. Bello, López J.M., and O. Mirabella. Stochastic analysis of periodic real-time systems. In *the 23rd IEEE Real-Time Systems Symposium (RTSS02)*, pages 289–300, 2002.

[9] S. Edgar and A. Burns. Statistical analysis of WCET for scheduling. In *the 22nd IEEE Real-Time Systems Symposium (RTSS01)*, pages 215–225, 2001.

[10] Gumbel EJ. *Statistics of Extremes*. Columbia University Press, 1958.

[11] D. Faranda, V. Lucarini, G. Turchetti, and S. Vaienti. Numerical convergence of the block-maxima approach to the generalized extreme value distribution. Technical Report arXiv:1103.0889, March 2011.

[12] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2):13:1–13:15, June 2007.

[13] M.K. Gardner and J.W. Lui. Analyzing stochastic fixed-priority real-time systems. In *the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS99)*, pages 44–58, 1999.

[14] M. Garrido and J. Diebolt. The ET test, a goodness-of-fit test for the distribution tail. pages 427–430, 2000.

[15] B.V. Gnedenko. Sur la distribution limite du terme maximum d'une seris aleatoire. *Annals of Mathematics*, 44:423–453, 1943.

[16] David Griffin and Alan Burns. Realism in Statistical Analysis of Worst Case Execution Times. In *the 10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010)*, pages 44–53, 2010.

[17] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The Mälardalen WCET benchmarks – past, present and future. pages 137–147, Brussels, Belgium, July 2010. OCG.

[18] J. Hansen, S. Hissam, and G. A. Moreno. Statistical-based wcet estimation and validation. In *the 9th International Workshop on Worst-Case Execution Time (WCET) Analysis*, 2009.

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

[20] J.P. Lehoczky. Real-time queueing theory. In *the 10th IEEE Real-Time Systems Symposium (RTSS96)*, pages 186–195, 1996.

[21] Marco Paolieri, Eduardo Quinones, Francisco J. Cazorla, Guillem Bernat, and Mateo Valero. Hardware support for WCET analysis of hard real-time multicore systems. In *ISCA*, Austin, TX, USA, 2009.

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

[23] T.S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.C. Wu, and J.S Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In *the 2nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS95)*, pages 164–174, 1995.

[24] Feller W. *An introduction to Probability Theory and Its Applications*. Wiley, 1996.

[25] S. Zhou. An efficient simulation algorithm for cache of random replacement policy. In *the 7th IFIP international conference on Network and parallel computing (NPC2010)*, pages 144–154, 2010.