Worst Case Execution Time and Power Estimation of Multicore and GPU Software: A Pedestrian Detection Use Case

Ivan Rodriguez Ferrandez, Alvaro Jover Alvarez, Matina Maria Trompouki, Leonidas Kosmidis, Francisco J. Cazorla

Barcelona Supercomputing Center (BSC) and Universitat Politecnica de Catalunya (UPC); email: {ivan.rodriguez, alvaro.jover, matina.trompouki, leonidas.kosmidis, francisco.cazorla}@bsc.es

Abstract

Worst Case Execution Time estimation of software running on parallel platforms is a challenging task, due to resource interference of other tasks and the complexity of the underlying CPU and GPU hardware architectures. Similarly, the increased complexity of the hardware, challenges the estimation of worst case power consumption. In this paper, we employ Measurement Based Probabilistic Timing Analysis (MBPTA), which is capable of managing complex architectures such as multicores. We enable its use by software randomisation, which we show for the first time that is also possible on GPUs. We demonstrate our method on a pedestrian detection use case on an embedded multicore and GPU platform for the automotive domain, the NVIDIA Xavier. Moreover, we extend our measurement based probabilistic method in order to predict the worst case power consumption of the software on the same platform.

1 Introduction

Applications with real-time requirements such as the ones used in automotive and other safety critical systems, require the computation of worst case execution time (WCET). However, the increased complexity of modern platforms targeting these domains, which includes multiple cores and GPUs, makes the computation of the WCET a very challenging task.

Conventional timing analysis methods such as static timing analysis require detailed information about the hardware architecture which are not available for these platforms, especially for GPUs. Similarly, traditional measurement-based methods (MBTA) are unable to deal with the uncertainty introduced by shared caches and GPUs. Measurement-Based Probabilistic Timing Analysis (MBPTA) [1] can address this complexity, allowing to compute WCET on multicores [2].

However, MBPTA requires certain properties to be present in the platform in order to be used, namely *independent and identically distributed (i.i.d)* execution times ensuring that execution conditions at analysis match the ones at system operation. These properties require either specially designed hardware [3] [4] or software randomisation [5] on Commercial-off-the-shelf (COTS) platforms. However, software randomisation methods have been only demonstrated so far on CPUs. In this work, we demonstrate for the first time that software randomisation is also applicable on GPUs, and in particular its static variant which works at source code level, known as Toolchain Agnostic Software Randomisation (TASA) [6].

In addition to the Worst Case Execution Time, safety critical systems require the estimation of their power consumption. Such an estimation is useful in order to guarantee the adequate thermal design of the system, so that it stays within its operational limits, otherwise there is a risk of unavailability of the safety critical system. However, the complexity of the hardware complicates this estimation using measurement based techniques [7], since during the experiments performance during analysis time, there are power measurement peaks which might not be observed.

Usually manufacturers provide a conservative upperbound of the worst case power consumption, known as Thermal Design Power (TDP), however this can result in overprovisioning of the system resources. The use of Extreme Value Theory (EVT) [8], which is one of the foundations of MBPTA, has been shown possible in the past for the worst case power consumption of a simulated time-randomised multicore CPU based on the NGMP [7]. Similar to MBPTA, hardware or software randomisation is required to enable the use of this method. In this work, TASA can provide this property.

In particular, we demonstrate our method on a pedestrian detection use case deployed on the automotive platform NVIDIA Xavier, showing that it is possible to obtain competitive WCET estimations with respect to traditional measurement based methods, as well as to estimate the worst case power consumption, too.

2 System Architecture

This work has been developed within the H2020 project UP2DATE [9], which aims to provide safe and secure updates on complex, heterogeneous platforms featuring multiple CPUs and including accelerators such as GPUs. The benchmarking of the available embedded high performance platforms early in the project, led to the selection of the NVIDIA

© 2023 Owner/Author. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in: Ivan Rodriguez Ferrandez, Alvaro Jover Alvarez, Matina Maria Trompouki, Leonidas Kosmidis, and Francisco J. Cazorla. 2023. Worst Case Execution Time and Power Estimation of Multicore and GPU Software: A Pedestrian Detection Use Case. Ada Lett. 43, 1 (June 2023), 111–117, https://dx.doi.org/10.1145/3631483.3631502

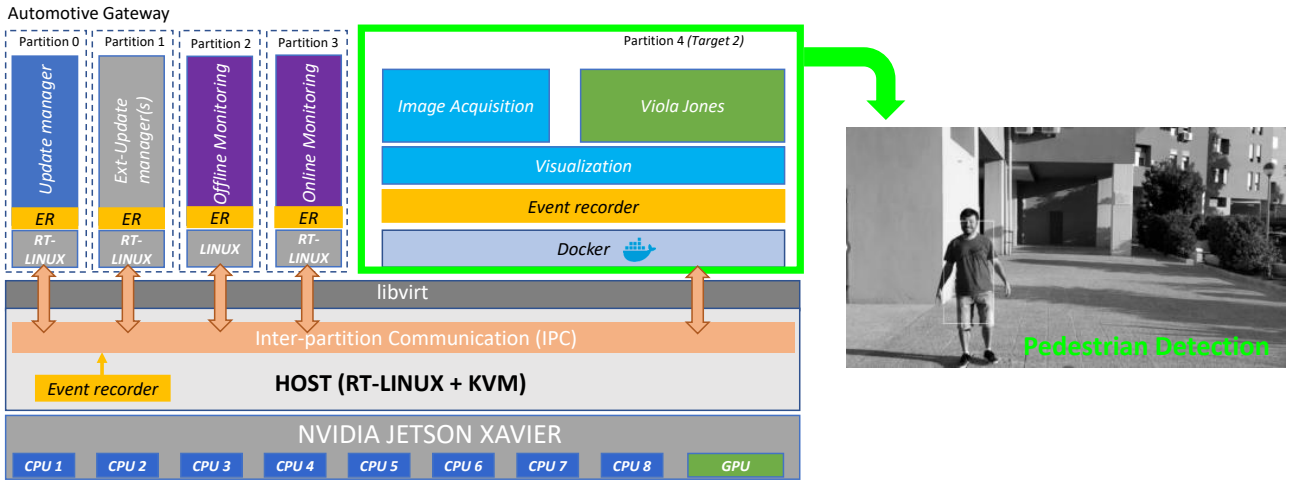


Figure 1: UP2DATE System architecture based on the Xavier Automotive Gateway Platform, featuring multiple CPUs and a GPU.

Xavier automotive platform [10]. This platform contains 8 ARM v8 compatible CPUs designed by NVIDIA, codenamed as Carmel. Each of the CPUs has private instruction and data caches, and shared L2 and L3 caches, connected to a 32GB LPDDR memory. Moreover, a GPU with 8 Streaming Multiprocessors (SMs) is connected to the LPDDR memory.

Since software updates are key for the project, it is important not only that to compute the WCET of safety-related software, but to remain also valid after an update. This led us to the following decisions regarding the system architecture. In order to facilitate software updates as well as time and space isolation required for the deployment of safety critical applications, the use of virtualisation has been decided [10] in order to create application partitions. For the deployment of CPU only partitions, the Linux KVM hypervisor has been selected, while for partitions requiring access to the GPU, the NVIDIA supported docker solution is used, since currently it is the only way that the GPU can be accessed from a virtualised environment on this platform. Figure 1 shows an overview of the system architecture considered in the UP2DATE project.

In order to simplify per-partition timing analysis, each partition is statically bound to a CPU and uses a single CPU. Similarly, in order to avoid the timing implications of GPU sharing, only a single partition is accessing the GPU. Overall, up to 8 partitions can be deployed on the platform, one for each CPU, and up to 1 of them is able to access the GPU. Given the large number of CPUs and the available memory of the platform (32GB), this approach is feasible. If the number of functionalities which need to be deployed are larger than the number of cores, they can be consolidated in one of the partitions.

3 Use Case Description

The UP2DATE project incorporates several industrial use cases from the railway and automotive domains [9]. However, none of them – which are based on legacy industrial applications deployed in existing single core safety critical platforms – had a high computational requirement needing a GPU. For this reason, we have developed a research use case which resembles a computationally intensive automotive task, such as pedestrian detection. Pedestrian detection is an important task needed for the implementation of Automatic Emergency Breaking functionality which since 2023 is required in all new cars sold in the European Union [11].

As a basis of the UP2DATE research use case we used a previously developed open source multi-CPU, multi-GPU pedestrian detection benchmark [12], based on Viola-Jones [13] [14]. While this classic machine vision algorithm has lower accuracy compared to modern implementations based on deep learning, it has the advantage of *explainability*, which is important for functional safety certification.

The main code changes are the code conversion to use only one CPU and GPU, and the instrumentation with the UP2DATE monitoring middleware, which allows to measure the execution time of the CPU and GPU tasks of the application, as well as the collection of performance counter measurements for both CPU and GPU. Moreover, the application has been enhanced with a new pedestrian data set, which has been used for project demos at dissemination events. The modified application is released also as open source at [15].

The application consists of 3 tasks. The first task performs the image acquisition. The second task processes the image in order to detect the pedestrians using the Viola-Jones method and returns the coordinates and the bounding boxes of the detections, which can be used as input by subsequent tasks e.g. for the implementation of the automatic emergency breaking functionality. Finally, the third task is responsible of displaying the detection, as well as saving the resulting image for logging purposes, e.g. for post accident examination for insurance or legal reasons.

4 GPU Software Randomisation

In order to enable the application of MBPTA on a conventional CPU and GPU platform, we apply software randomisation. Existing software randomisation methods [5] [6] are only available for CPUs. Moreover, dynamic software randomisation – the method with the highest Technology Readiness Level (TRL), demonstrated with several industrial use cases from the avionics [16] and the space [17] domain – is not available for ARM processors and more importantly, it is not possible to be implemented on GPUs. The reason is

that dynamic software randomisation requires compiler modifications. However, GPU vendors and especially NVIDIA are notorious for using proprietary languages, such as CUDA, with closed source compiler and runtime systems, which prevent the implementation of such functionality.

Toolchain Agnostic Software Randomisation (TASA) [6], was developed to overcome this limitation, by performing software randomisation at source code level. TASA relies on the principle that compilers generate object code in the same order they appear in the source code. However, TASA was prototyped as a proof of concept, only for CPUs and demonstrated with small benchmarks. Moreover, it only supported ANSI C and had no support for the C preprocessor, which limited its capability to work with industrial grade code.

In this work we have reimplemented TASA from scratch within the CIL [18] source-to-source compiler framework. CIL is capable of parsing any C code, including gcc extensions, which makes it capable of working even with the Linux kernel, which is likely the most complex and largest C code base. This makes our TASA reimplementation, which we have open sourced at [19] to achieve the highest TRL on CPU. CIL provides features which are useful for the application of TASA such as inclusion of function prototypes as well as a linker at source code level, which allows to merge all application's source code files in a single source code file which can be randomised with TASA.

In order to enable GPU software randomisation, we extended CIL to support the CUDA programming language. CUDA is a C/C++ extension for GPU programming, therefore our implementation only supports its C subset, due to the capabilities of our compiler toolchain. Using the NVIDIA binary utilities such as `cuobjdump`, we verify that similar to the application of TASA in CPUs, the location of functions and variables in the program executable follows their definition order in the source file. Moreover, we verify with microbenchmarks that the randomisation of GPU code randomises GPU cache conflicts, which in turn provide random execution times.

5 Methodology

In order to compute the WCET of the pedestrian detection use case which is considered our *Unit of Analysis (UoA)*, first we measure the timing of the pedestrian detection partition in isolation. In conventional measurement based timing analysis (MBTA) for single core systems, the maximum execution time ($max(ET_{UoA_{isol}})$) is inflated by an engineering margin (EM) in order to compute the WCET:

$$WCET_{MBTA} = max(ET_{UoA_{isol}}) \times EM \quad (1)$$

In order to adapt the WCET to account for the contention created by other partitions, we use two different approaches, similar to [2].

In order to make the estimation completely independent of the software running on the other partitions, and therefore support arbitrary updates of any partition, we use a *Fully Time Composable (FTC)* approach. This approach considers the worst possible contention from the other partitions running in the cores 2 to 8 ($WC_i,\ i \in [2,8]$), therefore at system operation the actual contention of other partitions is lower than the one considered at analysis:

$$WCET_{FTC} = max(ET_{UoA}) \text{ under } WC_i \text{ , where } i \in [2,8] \text{ and } max(ET_{UoA}) \geq max(ET_{UoA_{isol}}) \quad (2)$$

In order to obtain the $WCET_{FTC}$, we perform the measurement of the UoA, which is executed in the CPU core 1, while the rest of the cores in the system (7 cores, from CPU 2 until CPU 8) are executing a program which generates the maximum contention. In order to achieve this effect, we are using a microbenchmark which always misses in the L2 and L3 shared caches, similar to [2]. Therefore, whenever the software of the pedestrian detection partition (UoA) attempts to access memory (L2, L3 or the LPDDR) it suffers multicore contention from 7 more cores.

Although the $WCET_{FTC}$ estimation obtained with the Full Time Composable scenario is completely independent of the actual software executed in the rest of the CPU cores or partitions (given the one-to-one mapping of partitions to CPU cores), this comes at the expense of excessive pessimism in the WCET.

However, the resulting measured pessimism of FTC is still lower than the theoretical upperbound of the multicore contention. Note that the theoretical upperbound of this method is to consider that in an 8 core processor, in the worst case the expected maximum slowdown that can be experienced due to memory contention is 8×. The reason for this is the following: assuming that the memory hierarchy serving the 8 cores of the system is completely fair, each core will get 1/8th of its bandwidth.

In terms of Power consumption, the same Fully Composable method can be applied. The power consumption is observed under the same worst case contenders in order to generate a higher power consumption than the one that is going to be experienced during the system deployment.

The manufacturer-provided Thermal Design Power (TDP) is also a safe upper bound that can be used, regardless of the workload executed in the other cores.

In order to alleviate the pessimism of FTC, we also consider a *Partial Time Composable scenario (PTC)*. In this case, the maximum contention ($max(C_i)$) generated by each partition is measured using the UP2DATE middleware in terms of cache misses per second. Then a higher contention threshold (TH_i) is decided for each partition (CT_i), which can account for future updates of that partition.

$$CT_i = max(C_i) \times TH_i \quad (3)$$

Based on this threshold, a configurable contender is created, which generates at least CT_i memory contention. By replacing each partition with its corresponding configurable contender, the partition under analysis (the pedestrian detection partition in our case) is software randomised, and its

Table 1: Pedestrian detection execution time measurements (ms) in isolation (left) and FTC (right)

	Time per processed image (ms)	
	isolation	**FTC**
Minimum	17,519	18,605
Average	19,00468	22,24689
Maximum	26,521	111,923

probabilistic Worst Case Execution time is computed under that configuration ($WCET_{conf}$) using MBPTA.

The purpose of using software randomisation in the software under analysis is two fold: first, it allows to compute the WCET using MBPTA. The second reason is that the measured execution time under contention depends on the alignment in time of the memory requests generated by the other partitions. Unlike the FTC case in which every time that the pedestrian detection application tries to access memory suffers contention from the other 7 cores, in the PTC scenario this contention can be much lower. Therefore, there is a chance that the software under analysis accesses memory when the memory subsystem is not occupied serving requests by other CPUs/partitions. However, software randomisation changes both the number of miss requests generated from the software under analysis (i.e. due to the different conflicts among them in the caches) as well as the execution time of the task under analysis. Therefore, it allows to cover more cases during the analysis of the system.

In that case, the partial time composable WCET ($WCET_{PTC}$) is defined as:

$$WCET_{PTC_{conf}} = WCET_{conf} \text{ under } CT_i \text{ , where } i \in [2, 8] \quad (4)$$

It is worth noting that this PTC WCET is only valid for the particular configuration ($conf$ i.e. number of partitions and their specific thresholds) from which it was derived. For example, a given configuration may use less than 8 cores, e.g. if the number of partitions deployed in the system is lower. However, if a software update enables the use of an additional core, then the PTC WCET is no longer valid and the analysis needs to be repeated.

Moreover, this $WCET_{PTC_{conf}}$ continues to remain valid for as long as any new version of a partition after an update does not exceed the contention threshold CT_i used in the determination of the WCET. At system operation, the UP2DATE monitoring middleware ensures that each partition stays within the limit defined in the particular configuration used during the analysis and takes corrective action. Examples of corrective actions are restarting the offending partition, stopping the partition if it is not a high criticality one, or rolling back to the previous software version. In our case, no violations have been observed, providing additional evidence that our estimations are valid.

For the energy consumption under the Partially Composable scenario we apply the same methodology. Again the software randomisation of the unit under analysis is used in order to generate power peaks that might not be observed during the measurements performed at analysis, e.g. due to the increased number of misses generated by a given memory layout, and predict its probabilistic worst case using Extreme Value Theory, similar to [7]. In addition, the configurable contenders allow to upperbound the power consumption of the activity in the other cores.

For the implementation of the worst case contention of the FTC we are using a microbenchmark that always misses in the various cache levels similar to [2], as well as the linux *stress* benchmark for memory traffic and we select the one that generates the highest slowdown to the pedestrian detection partition. For the configurable contenders we use the same approach, but the contention of each contender is tuned using the *cpulimit* linux utility, in order to match the selected CT_i value.

6 Evaluation

In the left column of Table 1 the maximum observed execution time ($max(ET_{UoA_{isol}})$) from equation 1) per frame is 26,521 ms. In traditional measurement-based timing analysis methods (MBTA) applied on single core setups, this time is inflated by an engineering margin (EM) based on prior experience with the platform, in order to account for unknown latencies introduced by the architecture. In the PowerPC MPC 755 processor used in avionics, this EM is 20% (therefore the EM in equation 1 is 1,20 in this case) when the L2 cache of this single core processor is disabled [20]. Enabling the L2 cache would make the engineering margin much higher, and therefore cancel out much of the performance benefit offered by the use of the cache. The reason for this is that the L2 cache is unified and write-back, which means that the worst case for each cache entry is encountered when the replaced entry is dirty, and therefore its value needs to be written to memory before storing the new contents of the cache line. Consequently, level two or higher level caches are not used in industrial deployments of real-time systems or they are used under very specific configurations. To our knowledge, the only solution capable of analysing multi-level caches with a measurement based method, including the particularly challenging case of unified caches, is using time-randomisation [21].

For the NVIDIA Xavier which contains 3 cache levels and additional high-performance features compared to the MPC 755 such as dynamic branch predictor, prefetchers and multicore contention etc, there is no prior experience which can suggest an appropriate EM, although the previous example indicates that a higher EM is expected. However, for the sake of simplicity for the rest of this section we will assume optimistically that 20% is a sufficient EM also for NVIDIA Xavier as it has been done for comparison reasons in the MBPTA literature [20] [21]. Therefore, we assume that the WCET estimation per frame is $WCET_{MBTA} = 26,521 \times 1,20 = 31,8252$ms.

The average power consumption per frame with this setup is 9,99W while the maximum observed (high watermark) power consumption per frame is 10,973W. Note that the NVIDIA-supplied TDP (Thermal Design Power) of the platform we are using is 30W. In the absence of a reliable way to measure

Table 2: Analysis of each UP2DATE partitions in terms of cache misses per second

	use case	online monitoring	update_manager	external_update_manager	offline_monitoring
L1 Accesses	126953948,353333	1059333	511360,7	218952,5	1701044
L1 Misses	783343,2	48374,79	35733,05	19553,89	73019,85
L2 Accesses	81260172	6660086	2490600	2133653	3290321
L2 Misses	1926062	888311,5	399974,2	344016,6	560166,6

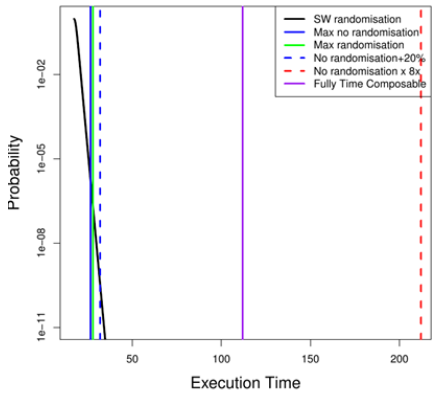


Figure 2: PTC pWCET curve obtained using MBPTA and comparison with conventional timing analysis.

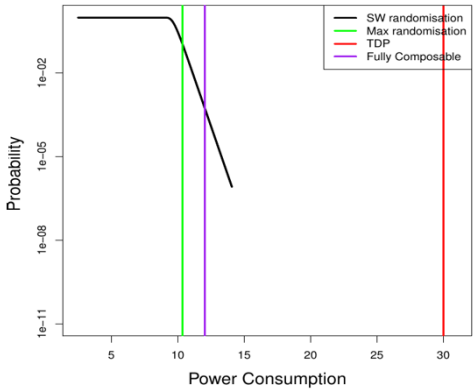


Figure 3: Partially Composable Probabilistic Worst Case Power Consumption curve obtained using EVT and comparison with TDP.

worst-case power consumption, this is the safe upper-bound that has to be used as a worst-case estimation.

Then, we profile the pedestrian detection partition under heavy stress to observe the worst-case execution time in terms of multicore contention for the FTC setup, in which the software executed in the other cores is replaced by worst-case contenders. Therefore the observed execution time is a safe upper-bound of the actual execution time when any other software is executed in the other cores. In the right column of Table 1 the maximum observed execution time ($WCET_{FTC}$) per frame is 111,923ms, which is 4,2× higher than the baseline execution time ($max(ET_{isol})$). Since the NVIDIA Xavier contains 8 CPUs, the theoretical maximum bus contention can be up to 8×. Note that this is a conservative estimate, since prior work has shown that the combined effect of bus, a shared L2 cache and memory controller of a 4 core processor can be as high as 9× and under certain cases, more than 9× [22]. Therefore, under the assumption that 8× can be the contention upper-bound for NVIDIA Xavier, this means that the use of worst-case contenders in UP2DATE, can provide a WCET improvement in the full time composable scenario of 52%.

In terms of power consumption per frame, the average power consumption in the FTC scenario (i.e. under worst case contenders) is 11,35W, while the worst case observed (high watermark) power consumption is 12,05W, which is 10% higher than the baseline. Considering again that the safe upper bound of the power consumption is the 30W TDP provided by the manufacturer, the use of worst-case contenders in UP2DATE provides a 60% improvement.

Next, we analyse each UP2DATE partition in terms of cache misses in order to build a custom configurable contender which can replace each one. The obtained values are presented in Table 2. We select a tight threshold (TH_i) of 1%, which is one of the tightest PTC estimations we could provide.

Finally, we replace each UP2DATE partition with configurable contenders and apply software randomisation to the pedestrian detection, so that MBPTA can be used to estimate its probabilistic WCET (pWCET) and power consumption. We perform 500 randomisations and collect measurements over 2000 experiments, which we verify that satisfy the i.i.d property using the Box-Ljung test for independence and the Kolmogorov- Smirnov test for identical distribution.

Figure 2 shows the pWCET curve obtained using Measurement-Based Probabilistic Timing Analysis (MBPTA). The black line as well as the maximum observed execution time with software randomisation, are slightly longer than the corresponding one of the non-randomised version. This is expected since software randomisation allows to explore much more memory layouts and contention alignments in time, so that their timing effect is taken into account. We notice that for the cut-off probability 10^{-12} the estimated WCET ($WCET_{PTC}$) is slightly higher than the optimistic 20% engineering margin which was assumed to be enough for Xavier. This partial time composable pWCET is 4× smaller than the fully time composable WCET as well as 7,5× smaller compared to the theoretical maximum contention that can be experienced among the 8 cores of the NVIDIA Xavier.

The application of MBPTA for the power measurements shown in Figure 3 predicts that the power consumption can be up to 15W per frame. This estimate is larger than the highest observed power consumption of 12,05W in the fully composable case. This is valid because despite the fact that

the worst-case contenders can increase power consumption, they are not able to increase the power of the platform to its maximum, since this depends also on the task under analysis. This estimate is 2× better than the conservative use of TDP.

7 Conclusion

In this paper we presented the Worst Case Execution Time and Power Estimation of a GPU application executing on the NVIDIA Xavier embedded multicore and GPU platform. We have achieved our goal using software randomisation which we enabled for the first time on GPU. Timing and power estimates per frame are 50-60% improved in the fully composable scenario compared to the theoretical contention (number of cores) and power consumption bounds (TDP). The partial composable scenarios provide a 4× improvement compared to the fully composable scenario in timing, but more conservative power results which are 50% better than the TDP.

8 Acknowledgments

This work was funded by the Ministerio de Ciencia e Innovacion - Agencia Estatal de Investigacion (PID2019-107255GB-C21/AEI/10.13039/501100011033 and IJC-2020-045931-I), the European Commission's Horizon 2020 programme under the UP2DATE project (grant agreement 871465), an ERC grant (No. 772773) and the HiPEAC Network of Excellence.

References

[1] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla, "Measurement-Based Probabilistic Timing Analysis for Multi-path Programs," in *24th Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

[2] E. Díaz, M. Fernández, L. Kosmidis, E. Mezzetti, C. Hernández, J. Abella, and F. J. Cazorla, "MC2: Multicore and Cache Analysis via Deterministic and Probabilistic Jitter Bounding," in *Reliable Software Technologies - Ada-Europe*, 2017.

[3] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla, "A Cache Design for Probabilistically Analysable Real-time Systems," in *Design, Automation and Test in Europe (DATE)*, 2013.

[4] M. Fernández, D. Morales, L. Kosmidis, A. Bardizbanyan, I. Broster, C. Hernández, E. Quiñones, J. Abella, F. J. Cazorla, P. Machado, and L. Fossati, "Probabilistic Timing Analysis on Time-randomized Platforms for the Space Domain," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.

[5] L. Kosmidis, C. Curtsinger, E. Quiñones, J. Abella, E. D. Berger, and F. J. Cazorla, "Probabilistic Timing Analysis on Conventional Cache Designs," in *Design, Automation and Test in Europe (DATE)*, 2013.

[6] L. Kosmidis, R. Vargas, D. Morales, E. Quiñones, J. Abella, and F. J. Cazorla, "TASA: Toolchain-agnostic Static Software Randomisation for Critical Real-time Systems," in *Proceedings of the 35th International Conference on Computer-Aided Design (ICCAD)*, 2016.

[7] D. Trilla, C. Hernández, J. Abella, and F. J. Cazorla, "An Approach for Detecting Power Peaks During Testing and Breaking Systematic Pathological Behavior," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 2019.

[8] S. Kotz and S. Nadarajah, *Extreme Value Distributions*. Imperial College Press, 2000.

[9] I. Agirre, P. Onaindia, T. Poggi, I. Yarza, F. J. Cazorla, L. Kosmidis, K. Grüttner, M. Abuteir, J. Loewe, J. M. Orbegozo, and S. Botta, "UP2DATE: Safe and Secure Over-the-air Software Updates on High-performance Mixed-Criticality Systems," in *23rd Euromicro Conference on Digital System Design (DSD)*, 2020.

[10] A. Jover-Alvarez, A. J. Calderón, I. Rodriguez, L. Kosmidis, K. Asifuzzaman, P. Uven, K. Grüttner, T. Poggi, and I. Agirre, "The UP2DATE Baseline Research Platforms," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[11] P. E. Ross, "Brakes that Slam Themselves: Automatic emergency braking will become standard in Europe," *IEEE Spectrum*, vol. 59, no. 1, 2022.

[12] M. M. Trompouki, L. Kosmidis, and N. Navarro, "An Open Benchmark Implementation for Multi-CPU Multi-GPU Pedestrian Detection in Automotive Systems," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017.

[13] P. A. Viola and M. J. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[14] P. A. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, 2004.

[15] A. Jover-Alvarez and L. Kosmidis, "Open-Source UP2DATE Pedestrian Detection Use Case Repository." https://gitlab.bsc.es/up2date/violajones-tasa.

[16] L. Kosmidis, C. Maxim, V. Jégu, F. Vatrinet, and F. J. Cazorla, "Industrial Experiences with Resource Management Under Software Randomization in ARINC653 Avionics Environments," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2018.

[17] F. Cros, L. Kosmidis, F. Wartel, D. Morales, J. Abella, I. Broster, and F. J. Cazorla, "Dynamic software randomisation: Lessons learned from an aerospace case study," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.

[18] G. C. Necula, S. McPeak, S. P. Rahul, and W. Weimer, "CIL: Intermediate Language and Tools for Analysis and Transformation of C Programs," in *Conference on Compiler Construction (CC)*, 2002.

[19] L. Kosmidis, "Open-Source TASA repository." https://gitlab.bsc.es/lkosmidi/tasa_cil.

[20] F. Wartel, L. Kosmidis, C. Lo, B. Triquet, E. Quiñones, J. Abella, A. Gogonel, A. Baldovin, E. Mezzetti, L. Cucu, T. Vardanega, and F. J. Cazorla, "Measurement-based Probabilistic Timing Analysis: Lessons from an Integrated-Modular Avionics Case Study," in *8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2013.

[21] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla, "Multi-level Unified Caches for Probabilistically Time Analysable Real-Time Systems," in *Proceedings of the IEEE 34th Real-Time Systems Symposium (RTSS)*, 2013.

[22] M. Fernández, R. Gioiosa, E. Quiñones, L. Fossati, M. Zulianello, and F. J. Cazorla, "Assessing the Suitability of the NGMP Multi-core Processor in the Space Domain," in *Proceedings of the 12th International Conference on Embedded Software (EMSOFT)*, 2012.