DOT/FAA/TC-19/24

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

Assurance of Multicore Processors: Limits on Interference Analysis

March 2020

Final report

U.S. Department of Transportation
Federal Aviation Administration

Technical Report Documentation Page

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

1. Report No. DOT/FAA/TC-19/24	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle ASSURANCE FOR MULTICORE PROCESSORS: LIMITS OF INTERFERENCE ANALYSIS		5. Report Date March 2020 6. Performing Organization Code ANG-E271
7. Author(s) Xavier Jean, Laurence H. Mutuel, Romain Soulat		8. Performing Organization Report No.
9. Performing Organization Name and Address Thales ATM 10950 El Monte, Suite 110 Overland Park, KS 66211		10. Work Unit No. (TRAIS) 11. Contract or Grant No. DTFACT-13-D-00008
12. Sponsoring Agency Name and Address Federal Aviation Administration William J. Hughes Technical Center Aviation Research Division Atlantic City International Airport, NJ 08405		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code Barbara Lingberg, AIR-6B4

15. Supplementary Notes

The FAA William J. Hughes Technical Center Aviation Research Division Technical Monitor was Srini Mandalapu.

16. Abstract

This report builds on previous research that addresses interference issues applied to multicore processors (MCPs). The previously proposed methods to perform interference analysis are organized around a set of assertions that the applicant should be able to defend when interacting with the certification authorities from the standpoint of safety and operations. The three assertions involve the applicant's commitment to a quantified interference penalty, the justification that the committable interference penalty can be trusted, and the justification that the proposed methodology for verification is adequate and feasible. To facilitate the interaction between the applicant and the certification authorities regarding these assertions, six key aspects are developed to support the substantiation of the three proposed assertions; therefore, the proposed methods to perform interference analysis are developed around these six key aspects. The notion of limits to the proposed interference analysis methods is developed by listing conditions under which some of these key aspects might be difficult to fulfill or to be fulfilled, but in an unsatisfying manner. These limits are classified as intrinsic to 1) the methods, 2) the MCP chip they are applied, or 3) the industrial environment. Developing this notion of limits highlighted good properties that the applicant can expect from applying an interference analysis method. The first benefit relates to the determination of interference penalties and the safety arguments substantiating these penalties. The second benefit is the scalability to the equipment's level of criticality and developmental stage. The third benefit is the adjustability with respect to the MCP monitoring and tracing capabilities. The last benefit relates to the adaptability to the industrial context in which multiple actors are involved, each having only partial visibility on the hardware and/or software embedded in the equipment. This report proposes an example of limits to an end-to-end interference analysis method driving test campaigns. This example shows that the question of interference analysis must be considered at an early stage of the computation platform design (i.e., the stage of selection of the MCP). The authors' position is that the question of interference analyses should be considered in guidance material in a way that allows applicants to propose their own methods and defend their argumentation.

17. Key Words Multicore processors, Safety, Interference, Worst-case execution time, Integrated modular avionics, COTS		18. Distribution Statement This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 47	22. Price

Contents

Figures

Tables

Acronyms

Acronym	**Definition**
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DMA	Direct memory access
DoS	Denial of service
DRAM	Dynamic random access memory
IMA	Integrated Modular Avionics
I/O	Input/output
MCP	Multi Core Processor
UD	Usage Domain
WCET	Worst-case execution time

Executive summary

This report builds on previous research that addresses interference issues applied to multicore processors (MCPs). The previously proposed methods to perform interference analysis are organized around a set of assertions that the applicant should be able to defend when interacting with the certification authorities from the standpoint of safety and operations. The three assertions involve the applicant's commitment to a quantified interference penalty, the justification that the committable interference penalty can be trusted, and the justification that the proposed methodology for verification is adequate and feasible.

To facilitate the interaction between the applicant and the certification authorities regarding these assertions, six key aspects were developed to support the substantiation of the three proposed assertions; therefore, the proposed methods to perform interference analysis were developed around these six key aspects:

- The validity of the hypotheses associated with an interference analysis method
- The range of conclusions associated with an interference analysis method
- The scope of supported applicative profiles and combinations of profiles (optional)
- The justification of single-core test cases
- The justification of combinations of test cases for multicore execution
- The collection of evidence and synthesis for the certification authorities

This research focused on the limits to the proposed methods to perform interference analysis. The notion of limits is developed by listing conditions under which some of these key aspects might be difficult to fulfill, or to be fulfilled but in an unsatisfying manner. These limits are classified as intrinsic to 1) the methods, 2) the MCP chip they are applied to, or 3) the industrial environment.

- Limits relative to the interference analysis method point to weaknesses such that additional work on the method is required.
- Limits relative to the MCP component point to weaknesses such that the method or the MCP selection has to be reconsidered.
- Limits relative to the industrial environment point to weaknesses such that the collaboration and information sharing between various actors has to be reconsidered.

Developing this notion of limits highlighted good properties that the applicant can expect from applying an interference analysis method, namely:

- Providing interference penalties to be applied on the software's worst-case execution time and therefore on equipment's performances; and providing safety arguments substantiating these penalties, with good balance between these two objectives.
- Being adjustable according to the equipment's criticality level and developmental stage, especially when these limits are considered at an early developmental stage in which hardware and software building blocks have not achieved a good level of maturity.
- Being adjustable with respect to reachable precision according to the MCP's hardware features, especially the monitoring and tracing capabilities.
- Being adaptable to an industrial context involving multiple industrial actors, each having a partial visibility on hardware and software involved in the equipment. One example of such a context is Integrated Modular Avionics.

This report proposes an example of limits to an end-to-end interference analysis method driving test campaigns. The examples of limits described in each category of key aspects constitute a non-exhaustive list and the selected example does not claim to be universal. Test cases for each processing units might be chosen in various ways, depending or not on targeted applicative profiles. Combinations and execution number of test cases for MCP execution may be proposed differently. Finally, the absence of singularity or other feared event can be argued in other ways. Nonetheless, they show that the question of interference analysis has to be considered at an early stage of the computation platform design (i.e., at the stage of selection of the MCP).

For these reasons, the position of the authors of this report is that the question of interferences analyses should be considered in guidance material in a way that allows applicants to propose their own method and defend their argumentation with very few restrictions, as long as they provide clear answers to fundamental questions, some being developed in this report within the six key aspects.

1 Introduction

1.1 Background

This report builds on previous research findings that addressed interference issues on multicore processors (MCPs), particularly commercial off-the-shelf (COTS) MCPs for which the avionics suppliers face a high level of complexity on hardware and partial information from their manufacturers.

The notion of interference analysis was defined as a means to assess and justify the penalty to apply on the worst-case execution time (WCET) of the software implemented on one MCP core while other pieces of software are running on the other cores. The method driving the interference analysis must be defined and agreed to by the certification authorities as soon as possible in the certification process. This may be a combination of several methods, involving test campaigns and static analyses of the MCP or subparts of the MCP.

The objective of the interference analysis is twofold. First, it has to provide a realistic interference penalty, which will be taken into account when sizing margins. The challenge is to obtain correct penalties that affect performances. Second, it has to provide safety-related arguments justifying that this interference penalty is trustworthy, so that the impact of the interference analysis can be considered in traditional safety analyses methods.

In the previous report on assurance process for MCPs, examples of techniques for interference analysis, interference reduction, interference bounding, and elimination were presented. This report focuses on interference analysis, whereas other techniques are considered as being design choices made by the platform developer to find the best tradeoff between performances and determinism.

In the previous report [3], the authors argued that principles of Integrated Modular Avionics (IMA) provide a relevant framework for use of MCPs in avionics equipment, even when non-IMA based. Although this research targets all classes of avionics systems, not only IMA systems, IMA terminology is used to distinguish the roles of system integrator that have an overall visibility on the hardware and software and the platform provider who will perform the interference analysis.

1.2 Definitions

The following definitions are used in this report:

Initiator:	A component of the processor that has the capability to proactively start operations within the shared resources. Examples of initiators are central processing units (CPUs), direct memory access (DMA) engines, master input/output (I/O).
Target:	A component of the processor that can be requested by initiators and either absorbs this activity (e.g., for write operations) or emits its own activity as an answer (e.g., for read operations) to initiator's requests. Examples of targets are memories and slave I/O.
Usage Domain:	Usage Domain (UD) is the set of constraints defined and mandatorily followed by users to ensure proper behavior of the device.
Interference:	Interferences are alterations of the processor's behavior seen by software running on one core due to activities ordered by software running on other cores. A statement developed in the previous report [3] considers that the impact of interferences is, at first, a problem of performance assessment and timing determinism enforcement. Interferences are undesired phenomena that are considered by the manufacturer as belonging to the functional domain of the processor, but are seen as dysfunctional by the avionics equipment provider, as explained in reference [1].
Interference source:	An interference source is a component on the processor that has simultaneous use by several cores or other initiators that may entail interferences. Examples of interference sources are shared caches and interconnect.
Interference path:	An interference path is a configuration in which a given set of initiators (e.g., cores, DMA engines, master I/O) is allowed to communicate with a given set of targets (e.g., main memory, shared caches, slave I/O) with no restriction. Having an interference path does not necessarily mean interferences will actuallyoccur. It only represents a configuration for which there is a risk of interferences; therefore, interference paths can be seen as test cases in which given initiators are allowed to request given targets (e.g., core 0 targets the main memory while core 1 targets the peripheral component interconnect express (PCIe) controller under which the processor's behavior can be stated.
Interference analysis:	An interference analysis is a process that considers interference paths and their usage according to the processor's UD, and determines those that are acceptable (from a performance and safety point of view) and those that are not, which are referred to as interference channels (see definition below).
Interference channel:	An interference channel is an interference path for which interferences have been actually observed and do not cope with the equipment's functional domain.

1.3 Literature review (update)

As a part of the earlier research task on the Assurance of Multicore Processors in Airborne Systems [3], a preliminary literature review [2] was conducted. This section provides a review of the research topics directly and indirectly focusing on interference issues on MCP. This section also presents the key papers discussed earlier in these domains and provides an updated review to the 2015 publication. For a more complete literature review, see reference [2].

1.3.1 Status of current guidance

No change has been denoted since 2015 in guidance documents published by certification authorities. The most recent guidance specifically focused on MCPs is the joint CRI-MCP (EASA) and CAST-32 (FAA).

As explained in the final report [3], system-level standards such as SAE ARP4754A [4] and ARP4761 [5] are relevant during the safe design phase for equipment embedding an MCP. The rationale defended in [3] is the intrinsic complexity of MCP, which leads to considering a computing platform as a full system and to referring to system-level guidance. IMA guidance, mostly DO-297 [6], is also a relevant reference for interference issues in MCP, as the robust partitioning aspects may not be satisfied.

Even if they do not constitute an official position of certification authorities, EASA studies Mulcors [7] and COTS-AEH [1] have provided inputs to the certification authorities to develop position papers on the issue.

1.3.2 Techniques for interference assessment on MCP

Various interference assessment techniques have been proposed in the literature, most targeting a configuration for which each CPU hosts a single task. Examples of such techniques are as follows:

- Nowotsch [8] proposed a technique called “interference sensitive WCET assessment” using a commercial tool (aiT) to compute an upper bound for competing accesses of concurrent tasks.
- Bin [9-10] introduced the notion of application signature to compute an offline interference penalty for a set of co-running tasks.
- Pellizonni [11] proposed a framework similar to Nowotsch’s WCET to assess the interferences on a memory controller.

Several frameworks [12-13] target interference issues on shared caches, especially instruction caches.

- Paolieri [14] modeled a dynamic random access memory (DRAM) controller and computed worst-case delays for incoming requests flows emitted by CPUs.
- Probabilistic works target interferences on MCP. Diaz recently published an example in reference [15].

1.3.3 Pathological situations sampling on MCP

Publications in this domain usually document denial-of-service (DoS) attacks on MCPs, benefitting from interferences. Historical publications in this domain were delivered by Nowotsch [16] for DoS on interconnect networks and Moscibroda [17] for DoS on DRAM controllers.

Several works have been recently published on this topic; however, the contributions of Blin [18] for DoS attacks on memory access of benchmarks from the open-source MiBench suite are noteworthy.

1.3.4 Programming models for interference reduction, bounding, or elimination

Interference reductions on COTS MCP are obtained by a proper configuration of Quality of Service features when they are available on the MCP (e.g., ARM interconnects) [19].

Software-enforced techniques have been proposed to address interference issues in shared caches and memory controllers. These techniques are called cache coloring [20] and bank partitioning [21], respectively. They are provided within a framework called Single-Core Equivalent Virtual Machines [22].

Girbal described other techniques controlling software execution for enforcement of determinism under the name of Deterministic Platform Software" [23]. An example of a promising technology is the Memguard memory bandwidth regulator [24], which relies on hardware performance counters to limit the concurrent accesses of CPUs to shared resources.

1.3.5 Hardware developments for built-in determinism in MCP

Dedicated hardware designs for intrinsic determinism enforcement on MCP have been explored for the past 15 years. This research area led to several prototypes (Merasa, PRET, CompSoC) but none have been industrialized and entered in a long-term production. The question of developing custom MCP remains a current topic in the avionics community.

In the recent studies, the contributions of Hassan can be noted, as detailed in his PhD thesis [30], and summarized in several publications; the interference was focused respectively on arbitration in DRAM controllers [36], cache coherence mechanisms [35] and arbitration policies in interconnect networks [34]. These aspects are known as a major source of non-determinism on MCPs, mostly because of the interferences they can generate.

1.3.6 System and module-level design for use of MCP in avionics

Operating system designers who address system designers' needs while dealing with constraints imposed by hardware have customarily addressed this area of research. WindRiver released a white paper [28] detailing some properties of their real-time operating system, particularly regarding interference issues on shared caches and busses. SysGo also contributed to this topic in 2016 [29].

The reference paper from Rushby [30] on robust partitioning issues, although published in 2000, remains relevant with regard to interference issues on MCPs when robust partitioning property is required (e.g., for IMA). A summary of literature regarding robust partitioning is provided in reference [31].

1.4 Scope of the report

This report covers the question of limits to the interference analyses. Before the topic of limits can be discussed, more details on interference analyses should be provided. Section 3.1.4 includes examples of techniques that can be useful, but do not constitute solely an interference analysis method.

An interference analysis is driven by applying a dedicated method, which has to be defined by the applicant and agreed to by the certification authorities as soon as possible in the equipment development stage. The authors' position of this report is that test campaigns and offline analyses should be combined to build an interference analysis method. Test campaigns remain central in the proposed approach when dealing with COTS MCP, for which the applicant faces a lack of information disclosed by hardware manufacturers and highly complex chips.

Previous research defined the interference analyses' objectives and placed restrictions on their scope. For instance, the question of interference triggering failure modes on the MCP was excluded to focus on performance and timing determinism issues. Furthermore, stakes were highlighted, mostly dealing with the risk of combinatorial explosions of test campaigns. This report proposes to organize interference analysis methods according to a set of key aspects, detailed in section 2.1. These key aspects are formulated as a set of questions that the applicant should address independently from the way they should be addressed to ease the interactions between the applicant and the certification authorities for questions relative to interference analysis from a safety and an operational point of view. Limits to interference analyses are developed according to these key aspects.

Informally, these key aspects could help to complete and defend the following assertions:

1. Under the following restrictions, I commit on an interference penalty of x%, applied to applicative SW's WCET computed individually.
2. I trust this interference penalty because […].
3. I consider my experimental methodology as relevant and feasible because […].

Assertion 1 is detailed as a key aspect (see section 2.1.3) dealing with supported applicative profiles on the MCP by the platform provider. This key aspect summarizes the kind of constraints applied on applicative software so that the interference analysis can be refined according to these constraints to get tighter, and sometimes realistic, results.

Assertion 2 is refined with two key aspects dealing with safety aspects of interference analyses (see sections 2.1.1 and 2.1.2) and one key aspect dealing with results synthesis and interaction with certification authorities (see section 2.1.6). These key aspects cover the question of interference analyses' trustworthiness hypotheses, conclusion ranges, and evidence collection for synthesis and approval by certification authorities, respectively.

Assertion 3 is refined as two key aspects (see sections 2.1.4 and 2.1.5) dedicated to test campaigns performed on MCPs during the interference analysis. These key aspects raise the question of the relevance of tests executed on each CPU and the combinations of such tests to generate interferences.

This report proposes a classification of limits to interference analysis according to three criteria:

- Limits relative to the method—The method will not achieve its expected results, regarding safety aspects, complexity (i.e., combinatorial explosion of the number of test cases) or results' precision (i.e., oversizing margins because interference issues are not understood).
- Limits relative to the MCP—The method is correct, but the selected processor is not appropriate to implement this method.
- Limits relative to the industrial environment—The method is correct and the processor fits with method's requirements, but the industrial environment makes its implementation unfeasible; for instance, because of the lack of information on MCP components (e.g., interconnect, memory controllers) and/or applicative software.

Finally, this report proposes to illustrate the limits of a method contributing to interference analysis by driving a test campaign. This method aims at taking into account a restricted set of applicative profiles supported by a platform provider to provide tight interference penalties, this set being possibly extended during the whole life of equipment.

2 Limits to interference analysis

The notion of interference analysis was developed in the white paper [2] as a means to assess and justify the performance degradation of an avionics platform with regard to embedded software. This performance degradation is expressed as an “interference penalty” applied to the WCET of each application evaluated separately; therefore, the objective of interference analyses is twofold. It produces a performance evaluation used by an integrator to size margins, but it also introduces some arguments that will be used in safety analyses so as to achieve certification.

As detailed in reference [3], an interference analysis cannot aim at being fully exhaustive on COTS MCP. The number of test cases on an MCP is so significant that no one, even the chip manufacturer, is capable of testing all configurations.

The interference analysis can be seen as a way to drive offline analyses/test campaigns within acceptable time and costs. This allows performance overhead to be correctly assessed and provides evidence that stop criteria are reached. These criteria have to be agreed to at an early stage of the certification process as an activity of safe design, whereas the resulting test campaign deals with safety assessment.

Deciding on an interference analysis method, including the test methodology and its associated stop criteria must be performed carefully because the resulting test campaign may not be capable of achieving its objectives. The purpose of this section is to provide a generic classification of limits to interference analysis. It is divided into two subsections: section 2.1 introduces key aspects constituting an interference analysis, and section 2.2 illustrates limits on interference analysis on these key aspects.

2.1 Key aspects of interference analysis

To show the limits of interference analysis, a set of key aspects is introduced that may be detailed by an applicant at an early stage of the certification process. These key aspects are provided here as examples and should not necessarily correspond to separate entries in a certification plan; however, they raise questions that remain relevant to be addressed and agreed to with certification authorities at an early stage of an equipment development.

2.1.1 Hypotheses conditioning trustworthiness in an interference analysis

As previously stated, an interference analysis method drives analyses and test campaigns performed by the applicant on an MCP, but it also provides arguments to defend safety analyses during certification.

In safety analyses, interference analysis plays a role similar to WCET computation. It turns a source of risk into another source of risk that is hopefully easier to assess. More precisely, an applicant not performing an interference analysis faces the risk of missing deadlines in safety critical software because of unexpected/misunderstood interferences. Conversely, an applicant performing an interference analysis will be able to explain and cover performance issues on embedded software due to interferences, provided that the validity hypotheses of the method used are met.

One important aspect of interference analysis is to clarify the hypotheses under which the method's results can be trusted. The applicant should assess the validity of such hypotheses so as to conduct safety analyses at equipment level.

As an example, one approach developed and discussed in reference [3] relied on the assumption that the behavior of an MCP is not chaotic. This means the processor's behaviors are considered to have a non-null surface, so that they can be reached during close tests—the notion of "closeness" being defined by the applicant. Consequently, for a given set of configurations that are close enough, the applicant can expect to observe close behaviors and, therefore, similar levels of interferences, unless there is a mode change. This mode change would be observed by a discontinuity in interference level.

2.1.2 Range of conclusions from interference analysis

An interference analysis will provide interference penalties, which are engagements supported by a platform provider and are taken by an integrator to add safety margins on software WCET.

Whereas having a set of interference penalties is sufficient for integration, it is insufficient for safety assessment. The interference analysis method needs to be accompanied by argumentation that the applicant trusts these penalties; therefore, the question is about the way to define the range of conclusions derived from an interference analysis.

In the example developed in reference [3] and recalled in section, an example of conclusion is proposed. It is based on the notion of singularity, which consists in close discontinuities in the MCP behavior and, therefore, in observed interference levels. Informally, singularities correspond to isolated behaviors of the processor that may not be covered by tests and constitute a feared event. In this example, the interference analysis would not conclude on the absence of singularities, but would provide a bound on their surface, which can be reused in a safety analysis and adjusted according to the equipment criticality level.

2.1.3 Definition of supported applicative profiles on the MCP

As explained in previous section, interference penalties are performance-related engagements taken by an applicant on an avionics platform embedding an MCP. However, such engagements may be conditioned by restrictions or hypotheses applied to embedded software, especially an applicative one and the way it uses hardware resources. The penalties and engagements constitute the supported applicative profiles for the MCP.

This key aspect is not mandatory. For instance, it may be decided that the set of relevant configuration parameters on the processor are sufficient to drive the interference analysis, the results of which would be valid for any applications. Such a configuration would minimize the impact of sources of non-determinism and the number of interference paths to cover, and finally cover them with worst-case scenarios. A rationale should be provided that these scenarios meet the “worst-case” and should be defended.

The applicative profiles may also be seen as a concept that will be continuously refined during the equipment’s development, as long as the software development is getting closer to its final version or when targeted software is not known with a good level of detail by the platform provider (e.g., on IMA platforms). This concept may also be extended during the equipment’s lifespan (e.g., when new software has to be deployed on the MCP platform). From this point of view, defining an MCP applicative scope can be seen as a prediction of the future needs of hardware resources by applicative software. The interference analysis will figure out the interference penalty while using hardware resources in a similar way.

2.1.4 Justification of individual test scenarios

This key aspect aims at justifying the relevance of the tests that will be deployed on each core of the MCP to generate interferences. The main reason is the extreme complexity of a test space that makes any exhaustive test campaign unpractical; therefore, only a subset of all possible configurations will be covered. Any way to justify a given set of test scenarios rather than another can be proposed, even randomly generated tests.

As an example, report [3] proposed to define and apply a metric on test scenarios, which would detail how much a configuration is close to another one. This metric enables the definition of a notion of coverage with a given granularity over the test space, so that a threshold can be agreed to with certification authorities.

2.1.5 Selection of concurrent tests

This key aspect addresses the test campaign that consists in running test scenarios in parallel to measure the performance overhead introduced by interferences. The number of potential combinations can quickly become significant, so a selection is mandatory. This selection has to be justified.

Reference [3] detailed as an example the notion of stop criteria to justify the following:

- The decision to cover a more or less exhaustive subset of interference paths on the MCP.
- The decision to cover a more or less exhaustive subset of hardware configurations within a given interference path.

2.1.6 Evidence collection

The previous key aspects dealing with MCP's scope and the test campaign, both on the CPU-per-CPU and the concurrent aspects, were tackling design issues and safety. This key aspect deals with the safety assessment: Evidence has to be provided to the certification authorities that the test campaign has been handled according to the previously agreed parameters.

One goal is to define relevant indicators that have the following good features:

- Communication with certification authorities is limited to only significant information.
- The indicators remain independent from the MCP's architecture so that they can be reused in several certification projects.
- The indicators are adaptable to different criticality levels.

These key aspects do not constitute guidance or recommendation but have to be considered as a set of questions that may be legitimately raised during the development of a certified platform embedding an MCP. The purpose of these key aspects is to show the limits of interference analyses in the sense that under some conditions, some of these questions may become very hard to address.

2.2 Classification of interference analysis limits

The classification of interference analysis limits is proposed in this report according to the following aspects:

- Limits relative to the methods—These mostly theoretical limits are to be considered whatever the targeted equipment and the MCP. A deep adaptation of the method is required, or limitation may appear in the criticality level that can be reached.
- Limits relative to an MCP chip—These limits correspond to the situation for which an interference analysis method is suitable but cannot be deployed properly on a given MCP. Such limits may be tackled at an early stage by taking them into account during the MCP selection, or by adapting the method to fit the MCP characteristics.
- Limits relative to the industrial context in which the MCP is used in equipment—These limits correspond to configurations in which an interference analysis method fits the selected MCP but cannot be deployed properly because of the constraints introduced by the industrial environment (e.g., the restriction of information communication).

2.2.1 Limits intrinsic to interference analysis methods

Interference analysis methods are still a research topic; therefore, no consensus method, or family of methods, has emerged in the state-of-the-art yet. As a result, an interference analysis method might be proposed even if some of its aspects have not been experienced up to a high technology readiness level.

As previously explained, interference analyses play a role not only in performance assessment, but also in argumentation for safety analyses. Limits of an interference analysis method appear when one of these two objectives has not been clearly defined, or when both objectives are tackled in an unbalanced manner. Table 1 discusses some of these limits with regard to the key aspects introduced in section 2.1.

Table 1. Summary of key aspects and limits of associated methods

Key aspect	Method specific limits
Trustworthiness hypotheses	When an interference analysis method targets only performance assessment, with a limited focus on argumentation for safety analyses, trustworthiness hypotheses may not be expressed clearly, or may be defined in a way that makes them difficult to verify. When this is the case, it adds a significant complexity to the safety analyses performed on the MCP. An interference analysis method is likely to be refined all along the equipment's development. However, such a refinement might have a negative impact on trustworthiness hypotheses. The main risk here is to discover new hypotheses late in the interference analysis implementation, so that the method becomes irrelevant.

Key aspect	Method specific limits
Range of conclusions	Having a clear understanding of the conclusions driven by an interference analysis is necessary to be able to tackle it correctly from a safety point of view. Similarly to the previous key aspect (trustworthiness hypotheses), an interference analysis method has to be defined so that the focus is balanced between performance assessment and justification of this assessment for safety. Therefore, the conclusions of an interference analysis need to be understood in a safety context and be practical for use in a safety analysis. This entails a risk of not clearly defining these conclusions, or degrading them for as long as the interference analysis gains in maturity. It is preferable that an interference analysis method relies on detailed investigation on which levelis adjustable according to the criticality of the targeted equipment. The same expectation applies to the range of conclusions of an interference analysis.
Supported applicative profiles	An applicative profile can be considered as an abstraction of future applicative software, which may not be fully known during platform development. This key aspect is optional, in that a platform provider may decide to target any kind of application, with the risk that the resulting interference analysis becomes significantly complex to perform on a given MCP. When the platform provider decides to restrict the scope of its platform with applicative profiles, the issue of defining the profiles has to be addressed carefully. These profiles describe the way applications use hardware resources, so that the interference analysis can be performed while taking these profiles into account. This introduces the risk that this abstraction: • Is defined in a way too restrictive with regard to the software's actual behavior (i.e., it carries too much information so that an interference analysis would be valid only in specific cases, for instance in specific execution paths inside an application). Therefore, the interference analysis would have to consider too many applicative profiles to be sustainable on a real equipment. • Requires modifications in the applications, so that only an instrumented code would be evaluated with respect to interference-related issues. Requires information that cannot be observed on a real application without being heavily intrusive in an application's execution flow (e.g., by suspending it with instrumentation frameworks). In this case, the interference levels would be irrelevant.

Key aspect	Method specific limits
Justification of test scenarios on each core	Test campaigns seem unavoidable to perform an interference analysis on a complex COTS MCP, even if they do not constitute the sole investigation means, and would benefit from alternative methods (e.g., involving static analysis techniques). Individual tests to be deployed on one CPU and then run in parallel with other tests will be developed with an objective of stressing some hardware resources (e.g., main memory, shared caches, on chip networks, I/O). They constitute a test suite that will be used during interference analysis. One risk is to accumulate many test cases without a clear understanding of their added value within the test suite. Therefore, this suite could be extended infinitely, whereas the applicant would not detect that additional tests are useless. The consequence would be an interference analysis that is periodically extended for a short duration to run additional test cases (i.e., there is a risk that the interference analysis never produces satisfactory results regarding the coverage of hardware behaviors by tests because the notion of coverage is not properly defined).
Selection of concurrent tests	Individual test cases, deployed CPU by CPU, constitute test suites. There may be one test suite per CPU or test suites used on several CPUs. To generate and measure interferences, test cases will be picked inside test suites, deployed and run in parallel, so that their execution time's overhead due to interference will be measured. Similar to test suites' constitution (see below), the selection of test cases to be run in parallel is a difficult problem because all combinations cannot be covered; therefore, only a subset of all combinations will be tested. The question of the added value of a new combination with regard to the already tested ones needs to be understood by the applicant. The risk is the same as for individual tests constitution: performing an endless test campaign.
Evidence collection	The interference analysis will involve one or more test campaigns, possibly generating a significant amount of data on the measured interference levels, and the correct execution of the test campaigns. These data cannot be injected "as is" in a certification report. Instead, they have to be summarized so that only significant information is exchanged with certification authorities. One risk associated with interference analysis is to launch test campaigns without clarifying the way test results will be processed and what kind of information will be significant. Moreover, when several investigation methods are used (e.g., test campaigns and static analyses), the correlation between each result benefits from being explained at an early stage of the interference analysis, so that the contribution of each method to the final results is clarified.

2.2.2 Limits relative to MCP chips

This section considers an interference analysis method that has been correctly defined and is being implemented on an MCP chip. The limits relative to the MCP component, as presented in table 2, are defined as implementation limits because of its intrinsic characteristics. Therefore, such limits may have significant consequences when they are discovered at an advanced stage of the equipment's development, and may entail iterations in component selection, with significant extra-costs.

Note: This is a focus on interference analysis only so that the limits do not deal with complex COTS issues for which literature exists.

Table 2. Summary of key aspect and MCP component relative limits

Key Aspect	Limits Relative to MCP Component
Supported applicative profiles	An applicative profile is an abstraction of the way embedded software will use shared resources on the MCP, so that test campaigns focus on a set of "supported" applicative profiles. This means that a toolset has to be developed to determine, from a given application, to which applicative profile should be referred. A possible limitation of this toolset is a dependency to specific hardware features that are not implemented on a given MCP. Some monitoring and instrumentation features may be required on the MCP. Examples of such features are: • Low-latency breakpoint insertion capability to plug external debuggers. • Hardware performance counters. For instance, most CPUs used in MCP provide a couple of counters that can sample various hardware events. Additionally, most MCPs implement performance counters within shared resources, so that the multiplexed activities of each initiator (CPU or other master device such as DMA engine) can be sampled. • Tracing capabilities, with more or less intrusiveness in hardware behavior of trace probes. An applicative profile could typically be defined with a limited dependency on hardware features, so that it could be implementable on several kinds of processors with an adjustable level of precision.
Evidence collection	Similar to supported applicative profiles, evidence collection may require monitoring and/tracing capabilities from the MCP. An MCP may not provide sufficient capabilities (e.g., by not implementing enough performance counters), or may not provide such capabilities at all. This may have an impact on the quality of the evidence collected during the test campaign.

2.2.3 Limits relative to the industrial context

This section details the limits of interference analysis relative to the industrial context. These key aspects are summarized in table 3. In this industrial context, the methods are correct and the MCP implements hardware features that meet the methods' requirements. The focus is to explore the potential limits introduced by the industrial context, and more precisely:

- The multi-actor aspects, which are central in IMA, and limits due to existing processes and information not being shared between actors.
- The concurrent development of hardware and software, in which software needs are not known at the early stages of hardware development, so that interference analysis has to be launched with partial information on applicative software.

Table 3. Summary of key aspects and limits relative to the industrial context

Key Aspect	Limits Relative to the Industrial Context
Trustworthiness hypotheses and range of conclusions	Specific information might be required to verify some hypotheses. This information is not available to the MCP platform provider because it is confidential.
Supported applicative profiles	Applicative profiles are a means to exchange information between: • The platform provider, who has to drive the interference analysis according to the supported applicative profiles. • The integrator, who has to consider which applicative profiles to refer to so as to obtain interference penalties and size margins. • Possibly the application provider, who has a targeted applicative profile to cover. These actors may or not belong to the same organization and apply an IMA process or not. Underlying risks are: • Lack of information about applications, either because of the anticipation of their development (the integrator himself may not have a good overview of the applications being integrated until later in the development), or because of confidentiality. • Lack of agreement between actors on the way applicative profiles are defined. • Misunderstanding on the validity of applicative profiles, so that integration is performed on erroneous interference penalties.

2.3 Synthesis

In section 2, the notion of interference analysis methods is structured from the basic assertions exposed in section 1.4 . These methods are developed around six key aspects, one being optional. These aspects represent distinct stakes an applicant should explore. The proposed approach does not apply any constraint on the manner by which the applicant is performing such exploration nor on the manner in which the results of the exploration are reported to clarify the impact of such methods on safety analyses, test campaigns, and offline analyses.

Furthermore, the notion of limits to interference analyses is developed by listing conditions under which some of these key aspects might be difficult to fulfill, or be fulfilled but in an unsatisfying manner. These limits are classified as intrinsic to (1) the methods, (2) the MCP chip they are applied to, or (3) the industrial environment. Each class has various implications with respect to the method design or component selection.

Developing the notion of limits to interference analyses also highlights good properties the applicant can expect from an interference analysis method, namely:

- Providing interference penalties to be applied on the software's WCET and therefore on equipment's performances, and safety arguments substantiating these penalties, with a good balance between these two objectives.
- Being adjustable according to the equipment's criticality level and development stage, especially when considered at an early stage in which hardware and software building blocks have not achieved a good level of maturity.
- Being adjustable in reachable precision according to the MCP's hardware features, especially monitoring and tracing capabilities.
- Being adaptable to an industrial context involving multiple industrial actors, each having a partial visibility on hardware and software involved in the equipment. One example of such a context is IMA.

3 Application of proposed method and associated limits

A common, quick but imprecise approach to tackle interference issues consists in stressing each interference path up to the highest possible level of workload to maximize interferences, measure them, and determine interference penalties. The highest penalty possible is the interference penalty on the platform (i.e., interference penalties are obtained by denial-of-service attacks).

Such a direct method has good properties. For instance, it decouples interference penalties from applicative software (i.e., the key aspect dealing with applicative software profiles is irrelevant).

Moreover, it remains in line with key aspects previously presented, even if additional work would be required to address them explicitly.

However, literature has shown that on COTS MCPs, without a proper configuration and severe restrictions of hardware use, interference levels are high when they as stressed by denial-of-service attacks techniques. This entails a significant risk that the equipment will not meet its expected performance (i.e., the denial-of-service attack strategy "succeeds" in many cases). Preventing these situations by a good configuration of the hardware is a natural approach, and may be sufficient to bring interference penalties down to an acceptable level, but this problem is known to be difficult.

This section proposes an example of a test campaign strategy that extends the previous approach to obtain tighter interference penalties. The objective is to discuss its advantageous and disadvantageous properties. This example is structured around the following patterns:

- A formalism to describe an applicative profile that might be further supported on the platform, and a distance between a test case and such a profile. This enables building a test population on each CPU with test cases "close enough" to the targeted profiles. Hopefully, denial-of-service applications causing severe interferences are filtered out by this strategy as being not representative from the targeted profiles.
- A formalism to describe combinations of test cases taken from each CPU's population. It aims at highlighting test combinations that are most relevant to improve the quality of the test campaign and, conversely, test combinations that have a lower added value for the test campaign.
- A formalism to abstract hardware situations encountered in shared resources of the MCP. This formalism allows for defining metrics about hardware testing quality, which could be proposed in a certification plan.

The purpose of this section is not to propose a "best offer" to address interference issues. The position of the authors of this report is that no method is universal (i.e., most relevant for any kind of MCP and any kind of equipment). Instead, by specifying an end-to-end method, it aims at highlighting hard points and how to address them, and at discussing its good properties and its limits, both theoretically and practically.

3.1 Representation of applicative profiles

3.1.1 Overview

When a piece of software is executed on a CPU, it initiates some electronic activity within the shared resources of an MCP. Building an applicative profile for such a piece of software demands a good overview of both the parameters that influence interferences and a mathematical framework to represent them adequately.

The characteristics retained from the activity (generated by software using hardware resources of an MCP), for each shared resource both for hardware components (e.g., memory, shared caches, interconnect networks, I/O) and software components (e.g., APEX services, lock-protected system calls) are:

- *The workload of traffic generated by the CPU running the software*. This workload has to be compared to the maximum workload that can be handled by the considered resource. Such a maximum workload would be typically reached by a denial-of-service application, possibly using several CPUs.
- *The impulsivity of accesses*. Memories and I/O components are capable of adapting their behavior to process bursts coming from traffic initiators, with various policies. For instance, a memory controller might favor requests from the burst to take benefit from open pages and improve the overall latency, or instead delay such a burst to avoid starvation from other initiators.
- *The locality of accesses.* As for burst management, sequential accesses are detected by hardware components that are capable of adapting their arbitration policies. This is commonly the case for memory controllers.

These parameters are represented by:

- One signature per shared resource, representing the cumulative repartition of workload exercised by the considered initiator. Informally, each point (x, y) of the dataset is defined as follows:
  - *x* is the workload that can be sent on the considered hardware resource (compared to the maximum workload reachable).
  - *y* is the amount of time during software execution in which the workload on the resource is below *x*.
- One signature per shared memory resource, representing the cumulative repartition of non-local workload. This object is similar as the previous one, but local accesses are not considered.

An example of such a signature is shown in figure 1 for a simulated workload. In this example, the average workload is represented around the inflexion zone (in red), whereas the right part of the plot (in green) contains information on the workload's impulsivity. For this case, the example could be summarized as follows: "During this execution, the application has consumed between 10% and 30% of resource X's maximum workload, with no significant peak."

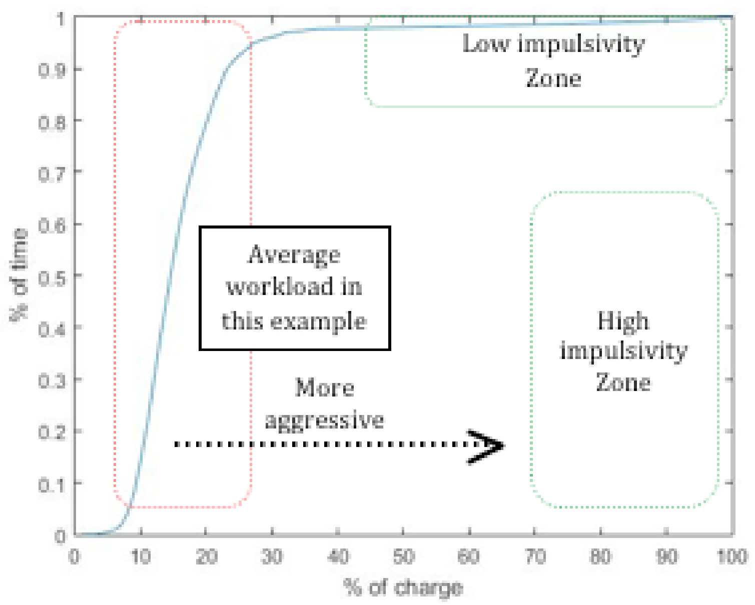


Figure 1. Simulated example of cumulative repartition curve describing workload on a given MCP resource

An applicative profile will aggregate such curves for each shared resource of the MCP, and for a pre-defined quantity of executions of the same application, possibly with code coverage objectives. Figure 2 shows an example of a beam of curves obtained from several simulations of a fictive application running on an MCP and exciting hardware resources.

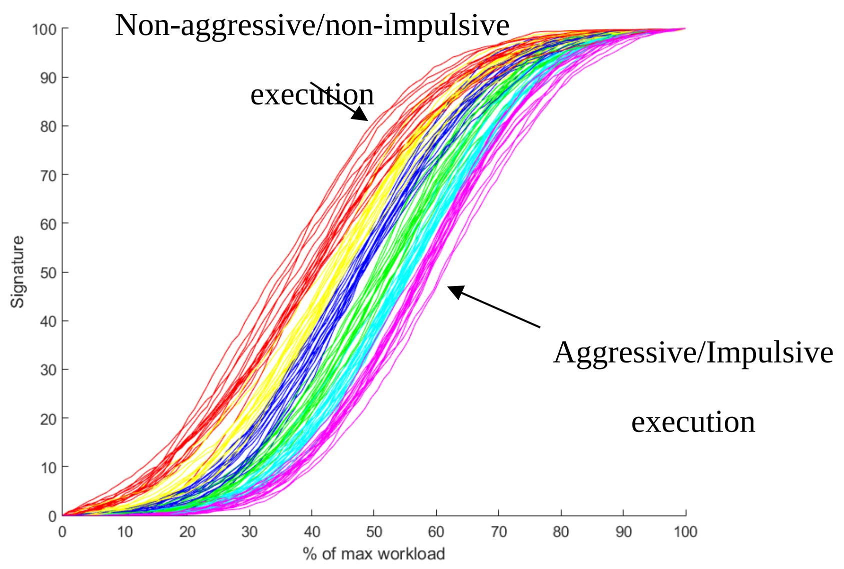


Figure 2. Example of curves obtained from several simulations of the same application

3.1.2 Building a test population on a CPU

A representation of a profile describing the way a given application will excite shared resources, both hardware and software (e.g., operating system services) has been introduced. Such a profile is built from the aggregate of signatures computed during (many) application executions, possibly with a code coverage objective.

Using a similar reasoning, it is possible to define the notion of a representative test with regard to a given applicative profile when the signature of a test fits in the considered profile. Therefore, restraining test populations to representative tests only makes sense for the remainder of the test campaign.

One of the key aspects of interference analysis is the justification of test populations deployed on each CPU. Two alternatives can be proposed to cover this key aspect:

- A test population might be designed to cover the targeted applicative profile, with no limit on the number of test cases required to reach this coverage. In this case, coverage means that, for any signature into the applicative profile, a test case having a close signature is present in the population. Function-relative distances might be used (e.g., L1 distance to define a mesh and a threshold can be proposed).
- A test population might be designed with a fixed number of test cases, but a “best-effort” strategy is to explore a sufficiently rich set of situations (e.g., access patterns) both on hardware and software shared resources. The best effort aspect might be formalized by an entropy measure that will be maximized by the test population.

3.1.3 Formalization

The profile can be obtained either from a trace or from sampling on hardware counters. The following formalization assumes that there is a trace of discrete events occurring in hardware, which seems realistic when considering real-time trace intellectual properties deployed on today's COTS processors (e.g., CoreSight™ for ARM series, Nexus for PowerPC series). Alternative definitions may be proposed from sampling of hardware performance counters.

3.1.3.1 Trace representation and workload computation

Consider a trace of discrete events with their timestamps, denoted as

$$tr = \{t_i\}_{i \in I} \tag{1}$$

By using Dirac notation δ_{ti} the trace function *T* is defined as follows:

$$T : t \mapsto \sum_{i \in I} \delta_{t_i}(t) \tag{2}$$

From this trace function, it is possible to define a workload function by computing a convolution product with a Gaussian centered function:

$$N_a : t \longmapsto \frac{1}{a * \sqrt{2\pi}} . e^{-\frac{t^2}{a^2}} . \tag{3}$$

The parameter, a, can be tuned to lengthen or shorten the period of time during which impulses are grouped together. The convolution product is denoted:

$$T * N_a : t \longmapsto \int_{-\infty}^{+\infty} T(\tau - t) . N_a(\tau) d\tau \tag{4}$$

An example of trace function is shown in figure 3 from a simulated trace, whereas the computed workload is shown in figure 4.

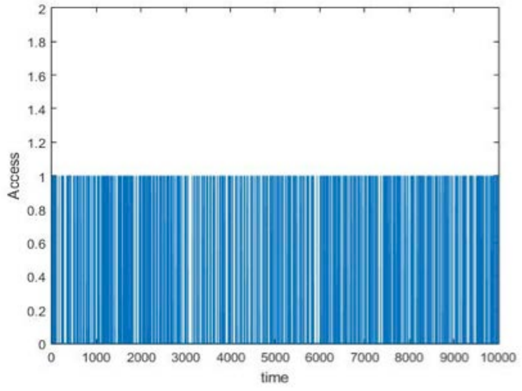


Figure 3. Example of trace function

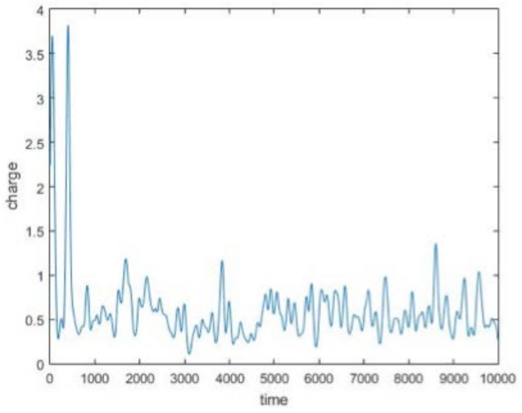


Figure 4. Workload function computed from trace function

Practically, alternative definitions for the workload function are possible—for instance, from sampling on hardware counters. Moreover, all information might not be collectable in a single run, so that several executions in similar conditions, or as close as possible, would be necessary, with some degradation of the sampling quality.

3.1.3.2 Application's signature

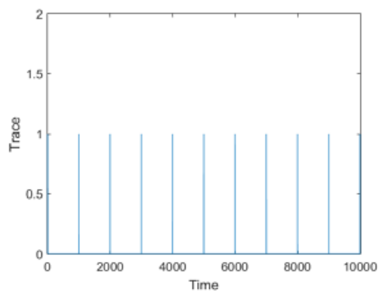
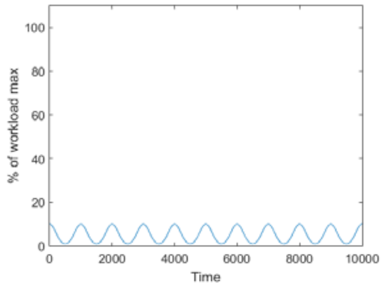
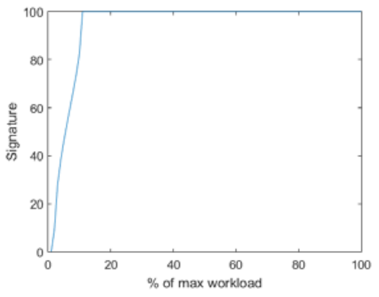
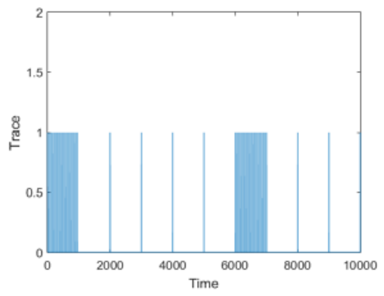
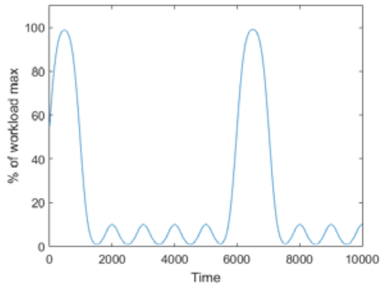
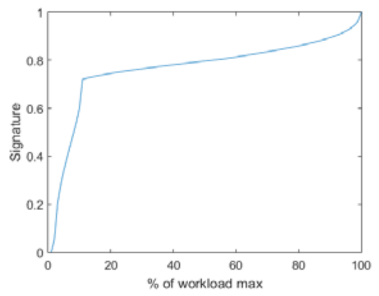
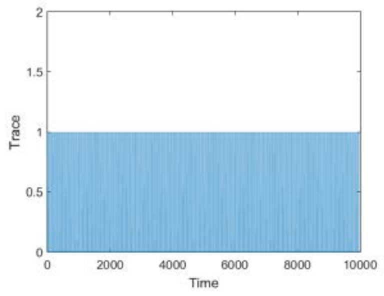
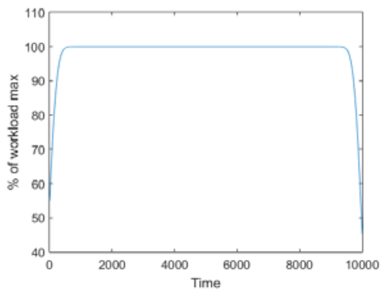
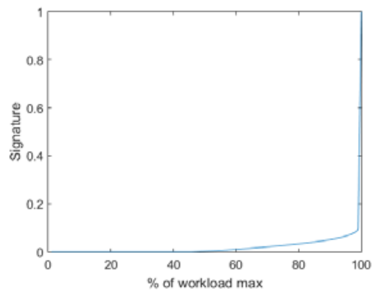
The application's signature has to be computed for each run, and might differ according to the initial conditions. This signature is computed from the workload function as follows:

$$P_T : \omega \mapsto \frac{\int_0^{T_{max}} \mathbf{1}_{Tr_S(t) \le \omega}(t)dt}{T_{max}} \tag{5}$$

3.1.4 Examples

Table 4 shows simple examples computed from traces generated by Matlab. Each trace would correspond to a separate run of an application seen from one shared resource of the MCP. In practice, the resulting applicative profile would aggregate application signatures from several dozens of execution, with views from a dozen shared hardware resources.

Table 4. Examples of computed traces

Simulated trace	Workload function	Application signature
 Low-aggressive trace, no access peak	 Low workload	 Non-aggressive signature
 Trace representing two bursts and a non-aggressive activity	 Computed workload represents the two bursts	 Slight impulsivity represented in the signature, even if the "typical" workload remains low
 Trace representing an intensive workload	 Maximum workload	 Aggressive and impulsive signature

3.2 Combination of test cases for parallel execution on an MCP

3.2.1 Overview

In this context, the following assumptions are made:

- One (single-core) application per CPU, each having a profile. The interference penalty for this combination of profiles is to be assessed. An application deployed on several CPUs could be reduced to several single-core applications.
- One test population for each application, possibly meeting the previous criteria; either coverage of the applicative profile or entropy maximization.

The objective is to select test cases from each population, and build combinations that will be deployed on the MCP during the test campaign to reach quality objectives and/or stop criteria.

An applicative profile described in section 3.1 summarizes the way shared resources are excited during a run of the considered application. A representative test belonging to the population will excite these resources in a similar manner. However, test cases will slightly differ from their targeted application. For instance, they might be a little more aggressive, more impulsive but similarly aggressive, more aggressive and impulsive.

In the remainder of this section, the following (simplified) terminology is used:

- An aggressive (resp. impulsive) test case refers to a test case more aggressive (resp. more impulsive) than its targeted application. “More aggressive” means that the test’s signature crosses the applicative profile’s upper margin, as shown in figure 5.

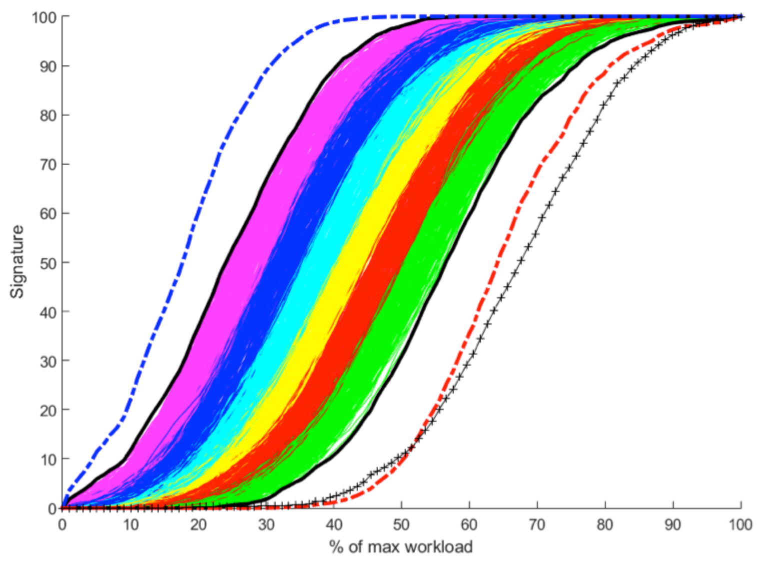


Figure 5. Signature profile for an aggressive test case

- A non-aggressive (resp. non-impulsive) test case will refer to a test case less aggressive (resp. less impulsive) than its targeted application. “Less aggressive” means that the test’s signature crosses the applicative profile’s lower margin, as shown in figure 6.

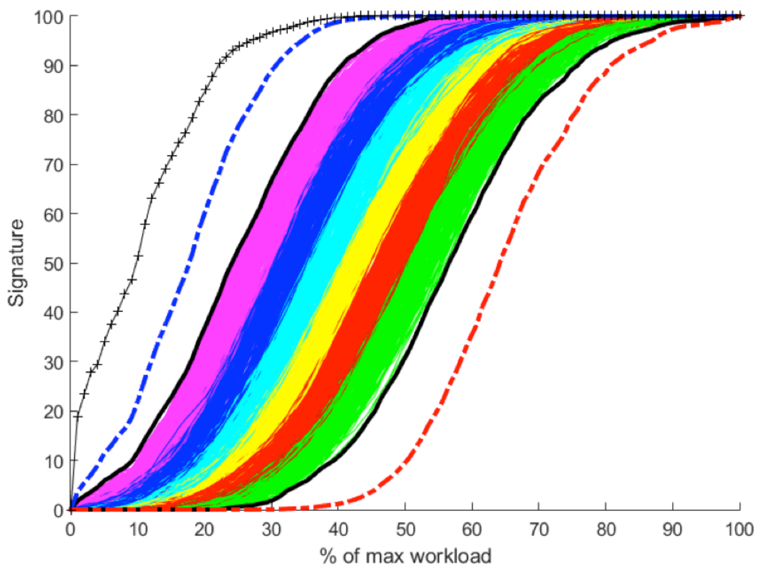


Figure 6. Signature profile for an aggressive test case

- A regular test case will have a signature not intersecting the applicative domain’s upper and lower margins (i.e., it will strictly remain in the same level of aggressiveness and impulsivity as its targeted application, as shown in figure 7).

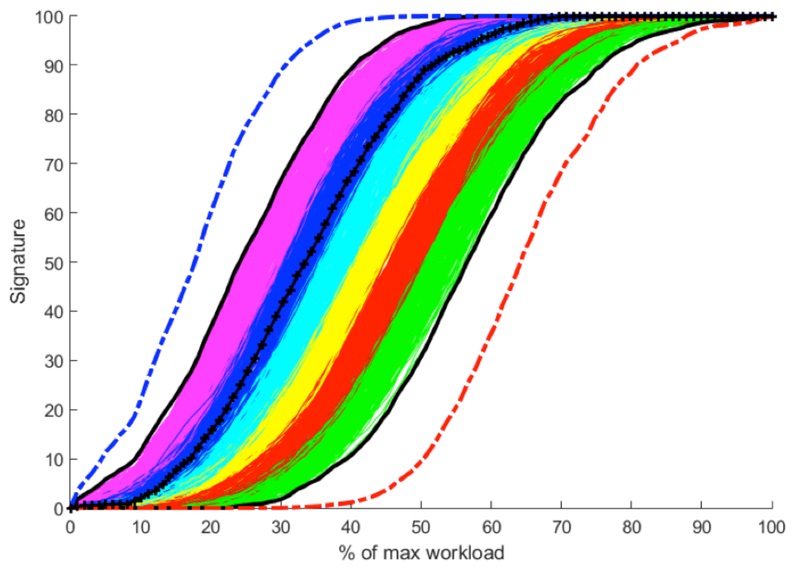


Figure 7. Signature profile of a regular test case

It is important to understand how a test case can differ from its targeted application, and to distinguish classes of “differences” because such information will be correlated to combine test cases among CPUs. For instance, the following combinations will be interesting to evaluate:

- Deploying aggressive and impulsive test cases on each CPU.
- Deploying aggressive (resp. impulsive) test cases in parallel with regular test cases.
- Deploying non-aggressive test cases in parallel with aggressive/impulsive test cases.

The challenge is to explore a sufficiently rich subset of test-case combinations, but still limit combinatorial explosion. However, the notion of “richness” has to be defined more formally, so that a stop criterion can be defined.

3.2.2 Formalization

From section 3.2, the objective is to select test cases on each core to maximize the exploration of hardware behaviors within the bounds of the applicative domains. The proposed formalism expresses the contribution of individual test cases to the global aggressiveness and impulsivity of a scenario combining them.

This formalism uses the following definition: Let *f*: [0,1]→[0,1] be a function and C a compact subset of [0,1]×[0,1]; the following statement will be used for the rest of this document:

$f(x) \prec D$ if and only if $\forall y \in [0,1], (x,y) \in D \Rightarrow f(x) \prec D$ for $\prec \in \{<, \leq, >, \geq\}$

This means that on a regular x-y plot, the representation of f will always be under/over the compact subset C.

In section 3.2, the notion of aggressive (resp. impulsive), non-aggressive (resp. non impulsive), and regular test cases was introduced. This notion is represented with three norms, respectively:

- The first norm is a measure of the "quantity" of the trace that is inside the application domain D: $d_{in,D}: f \; \rightarrow \lambda(1_{f(x) \in D}(x))$ λ being the standard Lebesgue measure.
- The second norm is a measure of the "quantity" of the trace that is more "aggressive," but in the domain D$_\epsilon$, that was admissible in D: $d_{under,D_\epsilon}: f \; \rightarrow \lambda(1_{f(x) \in D_\epsilon \,\&\, f(x) < D}\,(x))$.
- The third norm is a measure of the "quantity" of the trace that is less "aggressive," but in the domain D$_\epsilon$, that was admissible in D: $d_{over,D_\epsilon}: f \; \rightarrow \lambda(1_{f(x) \in D_\epsilon \,\&\, f(x) > D}\,(x))$.

Consequently, for every f such that $\{f(x), x \in [0,1]\} \subset D_\epsilon$ the following property is verified:

$$d_{in,D}(f) + d_{under,D_\epsilon}(f) + \; d_{over,D_\epsilon}(f) = 1 \qquad (6)$$

The function space to be explored is therefore modeled as a compact part of a plane shown in figure 8. As shown in this figure, intersection between this plan and axes correspond to combinations for which all test cases are more (resp. less) aggressive and impulsive than their targeted applications.

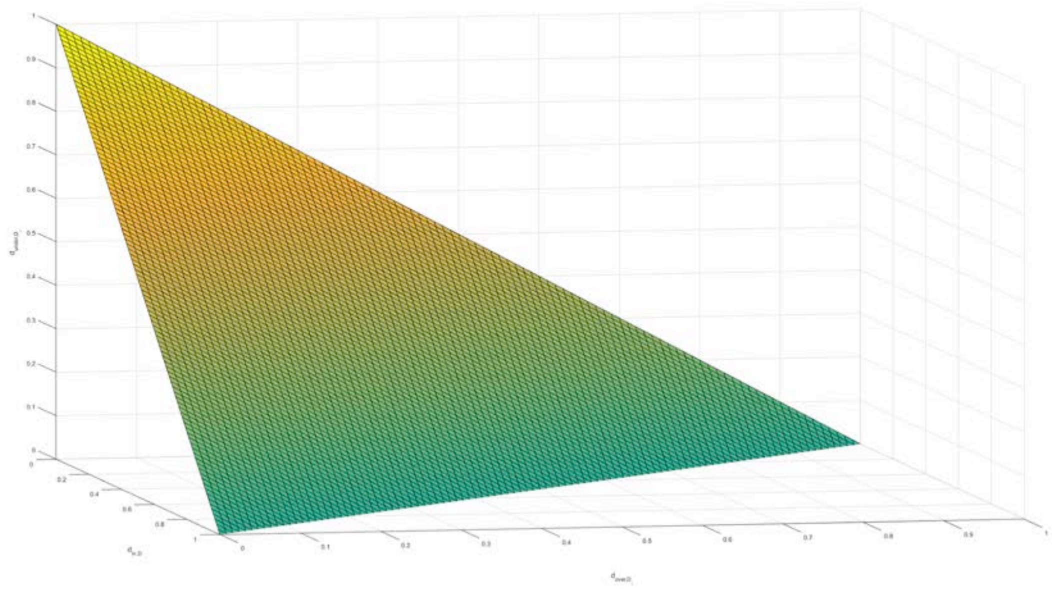


Figure 8. Representation of the function space to be explored

Any combination of test cases that meet applicative domains limitations on each CPU will be associated with a point on this plane. Therefore, it is possible to define a coverage objective stating that for each possible combination of test cases, a "close" combination has been deployed for which interference level has been assessed. This objective is formalized as follows, with a coverage threshold denoted α:

$$\forall(x, y, z) \in [0,1]^3 \; s.t.: x + y + z = 1, \exists T \; a \; trace \; s.t. \; d\big((x, y, z), (x_T, y_T, z_T)\big) < \alpha \qquad (7)$$

(x_T, y_T, z_T) corresponding to the three norms previously defined on test case combinations, and d being the classic distance between two points of R^3.

This framework allows limiting the number of test case combinations to be deployed on the MCP while ensuring relevant ones will be covered (e.g., aggressive tests on each CPU). Moreover, the coverage criterion may be decided at an early stage of the certification process, and can be adapted according to the criticality level of the targeted equipment. Reaching the expected coverage can be considered as a stop criterion.

This section detailed how to select test cases on each CPU and combine them to maximize the diversity of combinations with regard to their aggressiveness and impulsiveness. Informally, it addresses the question of the test campaign's quality from a software's point of view. Section 3.3 addresses a similar problem at the hardware level.

3.3 Quality metrics for hardware testing

The formalism defined in section 3.2 tackled the following question: "How can I build test case combinations to cover relevant scenarios and assess their interference levels?" This is a sizing parameter for the tests database, but it does not tackle the question of the quality of hardware testing. This aspect is tackled in this section.

3.3.1 Overview

Assessing interference level in a given scenario (i.e., for pre-selected test cases deployed on each CPU), may require several executions of this scenario. However, the number of executions has to be justified to claim that the hardware has been correctly tested. This problem is complex, as "correctly testing" an MCP is not clearly defined.

During different executions of the same scenario, the hardware is likely to behave differently. As for the applicative profiles, section 3.1 defines a metric representing the "distance" between hardware behaviors during two executions of a same scenario. By multiplying the number of executions of a given scenario, a given surface of the sets of hardware behaviors covered can be obtained, and it can be seen whether some behaviors are isolated (i.e., obtained in very few cases, and/or under specific conditions).

This metric is defined on the notion of hardware signature, which takes into account the following parameters:

- The density of accesses processed by a given hardware resource (e.g., the memory controllers, shared caches, interconnect networks). These accesses come from all CPUs and initiators like DMA.
- The level of interleaving of concurrent transaction flows emitted by different initiators.

Intuitively, the hardware signature is very close to the applicative signature defined in section 3.1.3.2. It represents similar phenomena (properties on transaction flows), but from the hardware point of view. Here, the way hardware resources process concurrent flows of transactions emitted by CPUs and initiators such as DMA, master PCIe, is represented.

The final objective is to maximize the surface of hardware behaviors covered by a given scenario, and simultaneously leverage the risk of isolated behavior, which could possibly correspond to a singularity. It is not relevant to introduce a coverage objective, as some hardware behaviors may not be reachable. Instead, the surface of hardware behaviors is defined on a k-density metric: a hardware signature collected from one run must be close enough to at least k signatures collected in the other runs.

A stop criterion on the surface obtained and the k-density of the set of hardware signatures can be proposed and argued.

3.3.2 Formalization

Hardware signatures are obtained with a formalism very close to the one used for the applications signatures. It starts from a set of event-based traces containing requests from each initiator to each shared resource of the MCP. These traces might be filtered to retain non-local and/or interleaved accesses.

From these traces, workload functions and then hardware signatures are computed. On each resource of the MCP, a set of hardware signatures will be collected with several runs of same–or-different scenarios.

The notion of k-density is formalized as follows: Considering a set of functions F, such as a set of profile signatures, F is said k-ϵ dense if

$$\forall f \in F, \#\{f' \in F, \|f - f'\|_1 \leq \epsilon\} \geq k, \qquad (8)$$

where #{E} is the cardinality of set S. Intuitively, this means that for every function in the set F, there are at least k other functions in F at a distance of at most ϵ. An example of a k-ϵ dense set of signatures is shown in figure 9.

This defines a stopping criterion for the generation of traces and signatures for the construction of the applicative domain. When this criterion is met, with values k and ϵ agreed to by the certification authorities, a representative set of traces for the monitored targets has been established, as shown in figure 9.

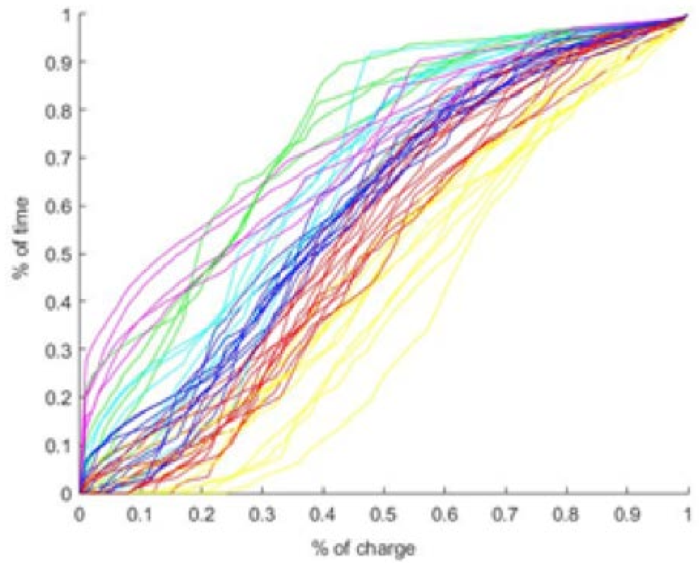


Figure 9. Example of k-ϵ dense set of hardware signatures (k=4)
“close” colors indicate “close” signatures

3.4 Discussion

3.4.1 Good properties and limits of this example

This section details an example of an end-to-end method to drive a test campaign targeting interferences on an MCP. Such a method is strongly linked with the notion of applicative profile, which may not be relevant for any kind of equipment and/or level of criticality. However, it raises relevant questions and challenges for the test campaign to optimize:

- The exploration of test cases around the given applicative profiles.
- The selection of test cases on each CPU to test all relevant combinations by limiting the number of test cases.

- The diversity of MCP components' behaviors being "excited" by the electronic activity of CPUs and initiators during software execution.

The concern is to propose a method as complete as possible with regard to the key aspects defined in this report. Recall the assertions of section 1.4:

1. Under the following restrictions, I commit on an interference penalty of x%, applied to applicative SW's WCET computed individually.
2. I trust this interference penalty because […].
3. I consider my experimental methodology as relevant and feasible because […].

Such assertions would find the following answers by the method detailed in this section:

1. The restrictions are applied to embedded software and hardware properties. An interference penalty will be provided for a given combination of applicative profiles. An avionics platform embedding an MCP would be enhanced with a catalog of supported applicative profiles and combinations; this catalog could possibly be extended during the equipment lifecycle.

   Moreover, hypotheses are made on the MCP. For instance, it is expected that its behavior is not chaotic (i.e., for almost any test case, considering the MCP behavior and associated interference level, it is possible to obtain a similar level of interference by deploying close test cases). The opposite is not true: close test cases might generate different interference levels; otherwise, the processor's behavior would be supposed to be continuous.
2. The interference penalty is considered trusted because
   a. Test cases around each applicative profile have been built, so that the MCP has been tested with a set of test cases, each remaining within its targeted applicative profile.
   b. Test cases on each CPU have been selected to cover all relevant configurations, this point being justified by a projection of the "aggressiveness" and "non-aggressiveness" of each combination on a plan of a 3D space. This plan is covered up to a predefined threshold.
   c. Test campaigns have shown that any hardware behavior was not isolated, and could be reproduced in closed conditions, concluding on the absence of singularity whose surface exceeds the threshold defined previously.
3. The experimental methodology leads to a bounded number of test cases and scenarios. Thresholds on coverage or density objectives will be decided and argued at the beginning of the certification process, during safe design phases. For instance, it is possible to agree

on an interference analysis over 10,000 scenarios for a given criticality level, or 100,000 scenarios for a higher one.

The method described in this example meets this assertion as completely as possible. However, it does not mean it is sound, and it has several limits. A non-exhaustive list is provided in section 3.4.1.1.

3.4.1.1 Specific limits related to the method

- There is little experience to instantiate such a method in a relevant way regarding the criticality level of the targeted equipment. Several parameters are adjustable (see section 3.4.2), but their impact on the method's complexity is not clear. More experience is needed to reach a conclusion on this point.
- There is a risk that some stop criteria cannot be reached. This is the case, for instance, for an objective of covered surface for hardware signatures, especially when the hardware has few possible behaviors.
- Whereas this method specifically aimed at avoiding combinatorial explosions, there remains a risk that the number of test cases grows quickly when some stop criteria become stringent.

3.4.1.2 Specific limits related to the MCP

This method was introduced with an assumption that a real-time trace is available for software execution and hardware resources usage so that workload and signatures can be computed. Although such features are implemented on several MCP available today, they are not largely used on the consumer market, nor documented by manufacturers. Alternative definitions of signatures could be proposed with a looser approximation, but this question remains open today.

3.4.1.3 Specific limits related to the industrial context

This method relies strongly on the way applicative profiles and hardware behaviors are abstracted (i.e., signature definitions). To be usable in a multi-actor context (e.g., for IMA), these representations need to be shared and understood by all actors (e.g., platform provider, module/system integrator, application provider). This introduces the risk of conflicting with internal processes.

3.4.2 Summary of adjustable parameters and stop criteria

This example shows that several parameters and stop criteria are adjustable, and would have to be packaged for a given equipment at a given criticality level.

3.4.2.1 Applicative profiles

Applicative profiles will have to be obtained from applications analysis and testing. There can be (or not) an objective of code coverage when building such applicative profiles. Moreover, static analysis tools can be used in this phase.

Applicative profiles are defined as the way applications “excite” shared hardware resources, the set of interesting hardware resources being adjustable. For instance, it may not be relevant to consider interconnects.

Finally, applicative profiles can be defined with an upper margin (corresponding to more aggressive and/or impulsive test cases), and a lower margin (corresponding to less aggressive/impulsive test cases). The margin’s size can be adjusted according to several parameters, including the equipment criticality level and the use of similar MCPs in other certified products.

3.4.2.2 Test cases combinations

Test cases are combined according to their aggressiveness and/or impulsiveness regarding their targeted applications. The formalism introduced in section 3 projects these parameters on a plan of the 3D space. A stop criterion is the coverage of this plan, so that extreme situations (all test cases are aggressive or non-aggressive) are covered, and a reasonable number of intermediate situations are also covered. The threshold on this plan’s coverage constitutes both an adjustable parameter and a stop criterion.

3.4.2.3 Parameters dealing with hardware signatures

Hardware signatures abstract how shared resources of the MCP were excited by various initiators (e.g., CPUs, DMA) triggered by software execution. Reaching a predefined surface and/or density on the set of hardware signatures can constitute a stop criterion (i.e., the applicant will pursue the test campaign on the MCP until it observes a rich enough set of hardware behaviors, still with no isolated behaviors).

4 Conclusions

This report introduced the notion of interference analysis limits by structuring them around key aspects, addressing respectively:

- Validity hypotheses of an interference analysis method
- Range of conclusions for an interference analysis method
- (Optional) Scope of supported applicative profiles and combinations of profiles

- Justification of single-core test cases
- Justification of test case combinations for multi-core execution
- Evidence collection and synthesis for certification authorities

Limits of interference analyses have been defined as weaknesses regarding one or more key aspects. A classification of interference analyses limits is proposed to use three classes:

- Limits relative to the method, for which additional work on the method is required
- Limits relative to the MCP component, for which the method choice or the MCP selection has to be reconsidered
- Limits relative to the industrial environment, for which the collaboration and information sharing between various actors has to be reconsidered

The examples of the limits described in each category constitute a non-exhaustive list. They show that the question of interference analysis has to be considered at an early stage of the computation platform design, when the choice of an MCP is considered.

Possible answers to key aspects were shown by developing an end-to-end method driving a test campaign, and discussing its good properties and limits. From this example, one conclusion can be driven. There are many degrees of freedom to be considered when defining an interference analysis method, and this example does not claim to be universal. Test cases for each CPU might be chosen in various ways, depending on targeted applicative profiles. Combinations and execution number of test cases for MCP execution may be proposed differently. Finally, the absence of a singularity or other feared event can be argued in other ways.

For these reasons, the position of the authors of this report is to address the question of interferences analyses in regulations in a way that allows applicants to propose their own method and defend their argumentation with very few restrictions, as long as they provide clear answers to fundamental questions, some being developed in this report within key aspects.

5 References

1. EASA Report. (2013). COTS-AEH – Use of complex COTS in Airborne Electronic Hardware – Failure Mode and Mitigation. (CCC/13/001303– Rev. 05).
2. FAA Internal Report. (2016). White Paper on Issues Associated with Interference Applied to Multicore Processors. (SDS-DO005 White Paper #3).
3. FAA Report. (2017). Assurance of Multicore Processors in Airborne Systems. (DOT/FAA/TC-16/51).

4. SAE Standard ARP4754A, 2010, “Guidelines for development of civil aircraft and systems,” SAE International, Warrendale, PA.

5. SAE Standard ARP4761, 1996, “Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment,” SAE International, Warrendale, PA.

6. FAA. (2005, November) Advisory Circular RTCA/DO-297. *Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations,*” Washington DC: RTCA Inc.

7. EASA Report. (2012). MULCORS-Use of Multicore Processors in airborne systems”. (CCC/12/006898 – Rev. 07).

8. Nowotsch, J., Paulitsch, M., Buhler, D., Theiling, H., Wegener, S., Schmidt, M. (2014). *Multi-Core Interference-Sensitive WCET Analysis Leveraging Runtime Resource Capacity Enforcement*. Presented at the 2014 26th Euromicro Conference on Real-Time Systems (ECRTS), Madrid, Spain.

9. Bin, J. (2014). “*Controlling execution time variability using COTS for Safety-critical systems.*” (PhD Thesis).

10. Bin, J., Girbal, S., Daniel, G. P., Grasset, A. and Merigot, A. (2013). “*Studying co-running avionic real-time applications on multi-core COTS architectures,*” Presented at the 7th European Congress On Embedded Real Time Software And Systems (ERTS), Montpellier, France.

11. Pellizzoni, R., Schranzhofer, A., Chen, J.-J., Caccamo, M. and Thiele, L. (2010). “*Worst case delay analysis for memory interference in multicore systems.”* Presented at the Design, Automation Test in Europe Conference Exhibition, Dresden, Germany.

12. Grund, D. and Reineke, J. (2010). “*Toward Precise PLRU Cache Analysis.*” Presented at the 10th International Workshop on Worst-Case Execution Time Analysis (WCET), Brussels, Belgium.

13. Hardy, D., Piquet, T. and Puaut, I. (2009). “*Using Bypass to Tighten WCET Estimates for Multicore Processors with Shared Instruction Caches.*” Presented at the 30th IEEE Real-Time Systems Symposium (RTSS), Washington DC.

14. Paolieri, M., Quiones, E. and Cazorla, F. J. (2013). Timing Effects of DDR Memory Systems in Hard Real-time Multicore Architectures: Issues and Solutions. *ACM Transactions on Embedded Computing Systems, 12*(1s), 64:1–64:26.

15. Díaz, E., et al. (2017). “*MC2: Multicore and Cache Analysis via Deterministic and Probabilistic Jitter Bounding.*” Presented at the Reliable Software Technologies – Ada-Europe 2017. Lecture Notes in Computer Science, vol. 10300. Springer.

16. Nowotsch, J. & Paulitsch, M. (2012). “*Leveraging Multi-Core Computing Architectures in Avionics.*” Proceedings of European Dependable Computing Conference (EDCC), IEEE Computer Society, 132–143.

17. Moscibroda, T. & Mutlu, O. (2007). “*Memory performance attacks: denial of memory service in multi-core systems.*” Proceedings of 16th USENIX Security Symposium, 18:1–18:18.

18. Blin A., Courtaud C., Sopena J., Lawall J., Muller G. (2016). “*Understanding the Memory Consumption of the MiBench Embedded Benchmark*”. In Networked Systems. NETYS 2016. Lecture Notes in Computer Science, vol 9944. Springer.

19. ARM. (2010). *CoreLink™ CCI-400 Cache Coherent Interconnect Technical Reference Manual*. ARM infocenter.

20. Mancuso, R., Dudko, R., Betti, E., Cesati, M., Caccamo, M. and Pellizzoni, R. (2013). "*Real-time cache management framework for multi-core architectures.*" Presented at the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Philadelphia, PA.

21. Yun, H., Mancuso, R., Wu, Z. P. and Pellizzoni, R. (2014). "*PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms,*" 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Berlin, pp. 155-166.

22. Sha, L., Caccamo, M., Mancuso, R., Kim, J.-E., Yoon, M.-K., Pellizzoni, R., Yun, H., Kegley, R., Perlman, D., Arundale, G., et al. (2014). "*Single Core Equivalent Virtual Machines for Hard Real-Time Computing on Multicore Processors*", University of Illinois at Urbana-Champaign Tech. Rep., 2014

23. Girbal, S.; Jean, X.; le Rhun, J.; Pérez, D. G. & Gatti, M. (2015). “*Deterministic Platform Software for Hard Real-Time systems using multi-core COTS.*” Proceedings of the 34th IEEE/AIAA Digital Avionics Systems Conference (DASC), Prague, Czech Republic.

24. Yun, H., Yao, G., Pellizzoni, R., Caccamo, M., Sha, (2013). "*Memguard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multi-Core Platforms.*" Presented at the Real-Time and Embedded Technology and Applications Symposium (RTAS) 2013 IEEE 19th IEEE, pp. 55-64.

25. Hassan, M., Patel, H., Pellizzoni, R. (2015). “*A Framework for Scheduling DRAM Memory Accesses for Multi-Core Mixed-time Critical Systems.*” Presented at the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2015), Seattle, WA.

26. Hassan, M., Kaushik, A. M., Patel, H. (2017). “*A Predictable Cache Coherence for Multi-core Real-time Systems.*” Presented at the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2017), Pittsburgh, PA.

27. Hassan, M., Patel, H. (2016). "*Requirement- and Criticality-aware Bus Arbitration for Mixed Criticality Systems*" Presented at the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2016), Vienna, Austria.

28. Parkinson, P. J. (2017). "*Update on using multicore processors with a commercial ARINC 653 implementation.*" White Paper presented at Aviation Electronics Europe, Munich, Germany.

29. Nordhoff, S. (2016). "*How Hypervisor Operating Systems can cope with Multi-core Certification Challenges*". Paper presented at Aviation Electronics Europe 2016, Munich, Germany.

30. FAA Report. (2000). Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance. (FAA-AR-99/58).

31. Jean, X., Faura, D., Gatti, M., Pautet, L. and Robert, T. (2012). "*Ensuring robust partitioning in multicore platforms for IMA systems*." Presented at IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), Williamsburg, VA, pp. 7A4-1–7A4-9.