ETUDE DES TYPES D'ERREUR ASSOCIES AUX PANNES DANS LES PROCESSEURS MULTICOEUR

INVESTIGATION OF ERROR TYPES ASSOCIATED WITH FAILURES IN MULTICORE PROCESSORS

Laurence H. Mutuel
Sierra Nevada Corporation
444 Salomon circle
Sparks, Nevada 89434, USA

Xavier Jean
Thales Research & Technology
1 avenue A. Fresnel
91767 Palaiseau cedex, France

Vincent Brindejonc
Thales Air Systems
Hameau de Roussigny
91470 Limours, France

Résumé
Cet article est la continuation d'un portefeuille de recherche sur la sécurité des composants sur étagère, dont l'utilisation se répand dans les systèmes avioniques complexes (Condra, 2014). Parmi les composants complexes sur étagère, les processeurs multicoeurs sont l'un des plus étudiés pour leur impact potentiel sur la sécurité associée à l'indéterminisme (Bieth, 2013). Un article précédent (Jean, 2015) discuta des aspects temporels ainsi que des méthodes pour déterminer le Worst-case Execution Time (WCET). Cet article se concentre sur la détermination des types d'erreurs spécifiques des processeurs multicoeurs, la détermination de leurs effets, les moyens de détection et de mitigation de ces erreurs, ainsi que l'existence de critères pour contenir la panne résultante ou pour récupérer de cette erreur.

Summary
This paper is a continuation of a research thrust on safety assurance for Commercial Off-the-Shelf (COTS) which use is becoming widely spread in complex avionics (Condra, 2014). Among complex COTS, multicore processors are one of the components being investigated in terms of their potential safety impacts relative to non-determinism (Bieth, 2013). A previous paper (Jean, 2015) addressed timing issues and the methods to determine Worst-case Execution Time (WCET). The present paper focuses on the determination of error types specific to multicore processors, the determination of the effects of these failure modes, the means of detection and mitigation of these errors, and the existence of criteria for containment of the resulting failure or recovery from the error.

Context

For a multitude of domains, the user demands for significantly increased performance, while size, weight and power remains constrained. This results in an increased share of multicore processors (MCPs) in the current market of highly complex semiconductor devices. In the safety domain that is aerospace domain, legacy embedded avionics systems are based on single core processors. The growing and more widespread use of multicore processors in other domains has a collateral impact on the aerospace domain: single core processors disappear from the manufacturer's roadmaps, so that single core processors may become obsolete.

Aircraft software applications themselves expand in capabilities that require increased computational performance (e.g., advanced signal processing, manipulation of large quantities of stored information) while achieving gains in size and power (e.g., Integrated Modular Avionics). Multicore processors address this need, so that their usage is foreseen to steadily increase. Multicore processors are currently used in airborne electronic hardware, although the technology is conservatively targeting multicore processors with no more than two active cores.

This paper focuses on aircraft software applications that could not run on single core processors for performance reasons. A multicore processor has been selected as the target architecture in the early stages of the design process. Furthermore, an initial assessment may have oriented the software development team towards dual core processors, for which the FAA CAST paper #32 provides guidance, or towards processors with more than two active cores, in which case the FAA Issue Paper on multicore assurance (or equivalent EASA Certification Review Item) will need to be addressed.

Problem Statement

The capabilities of multicore processors stretch the current assurance processes for both software and hardware. The supporting design and verification tools may not be adapted either. Since these processors were not initially designed with aircraft applications in mind, a preemptive investigation of the potential safety concerns is warranted. If the multicore processor technology driven by the overall market shows a faster growth than the aerospace domain is able to address assurance concerns, the potential for a long-standing 'catch-up' situation is likely. The investigation therefore needs to be not only preemptive but also prognosticative.

A key aspect of multicore-specific safety-related concern is associated with the demonstration of a deterministic behavior (Jean, 2012). The need for determinism is requested at aircraft level and depends on the aircraft function being considered. These functions may develop several types of dysfunctional behaviors potentially linked to non-determinism issues. As multicore processors address several of these functions or complex way to realize one of them, it is in this context that they have to be included in a safe design and in this context, their safety assessment has to be considered.

Method

The assessment of safety-related issues in civil aerospace systems is performed using analysis methods such as Functional Hazard Assessment, Preliminary System Safety Assessment, and System Safety Assessment using the guidelines of the Society of Automobile Engineers (SAE) Aerospace Recommended Practice (ARP) ARP4761 “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment” (SAE, 1996), and ARP4754A “Guidelines for Development of Civil Aircraft and Systems” (SAE, 2010).

These analyses are typically conducted at system level while Failure Modes and Effects Analysis (FMEA) is applied at equipment level. It has already been stressed (Roger, 2014) that challenges in multicore processors are closer to challenges encountered at system-level than to the equipment's ones. Therefore, the approach to multicore processor safety is to apply the system-level methods.

1. Description of Multicore Processors for Hazard Identification

Following the safety process of ARP4754A, the analysis of multicore processors follows the steps indicated below:

- Step 1: Identification of functional failures or hazards, or “feared events”
- Step 2: Determination of the hazard’s effects
- Step 3: Definition of potential mitigation methods.

1.1 Breakdown Levels

The methodology described in Reference (Bieth, 2013) is applied to multicore processors to determine their failure modes. Three breakdown levels are considered:

- Black-box breakdown level: the failure modes pertain to the output flows of the multicore processor. These failures are observable outside of the multicore processor, so that architectural means can detect, identify and mitigate these failures,

- Grey-box breakdown level: this level allows to identify key elements such as the interconnect, the shared caches, cores, and peripherals. This level of description must remain clear of intellectual property, but allows for the identification of generic faults, failure modes, and their common causes,

- White-box breakdown level: this level of description would allow the identification of internal failure modes and the description of the failure path down to impact on the multicore processor’s output. This level of description is rarely accessible to the Airborne Electronic Hardware (AEH) manufacturer; the issue is therefore to achieve confidence in the coverage of the failure mode realized by the capture process.

1.2 Abstraction Levels

Independently from the achievable breakdown level, three abstraction levels may be used to analyze the information exchanged internally between the multicore processor and its environment. These levels are:

- The functional level,
- The logical level, and
- The physical level.

Functional failures are typically expressed in either the loss or the malfunction of the function being rendered. At the logical level, the failure modes are described with respect to the states of the exchanged message. At the physical level, failure modes relate to the physical characteristics of the signals, which become ineffective for complex systems such as multicore processors. One exception to this statement is the identification of failure modes related to the technology, such as single event effect, voltage/current scaling, *etc.*

2. Determination of Failure Modes

The objective is to derive a generic fault model for a multicore processor. Considering failure modes through the perceived behavior caused in the multicore processor allows for creating a failure mode meta-classification at the logical level. This classification is then applied to the multicore processor described as a black-box. Since the main concern with multicore processors is the potential for non-determinism, this section describes the failure modes and builds such a meta-classification.

2.1 Types of Failures

The guideline document ARP4754A (SAE, 2010) distinguishes the following types of failures:

- Random hardware failure: failure occurring at a random time, which result from one or more of the possible degradation mechanisms in the hardware,

- Systematic failure: failure related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors. In this category, common failure modes will be investigated.

In terms of their impact, failures are further classified into the following categories:

- Transient failures (or soft errors),
- Permanent failures (or hard errors).

2.2. Failure Modes at Logical Level

The methodology in reference (Bieth, 2013) provides a classification method for logical-level failure modes, depicted on Figure 1.

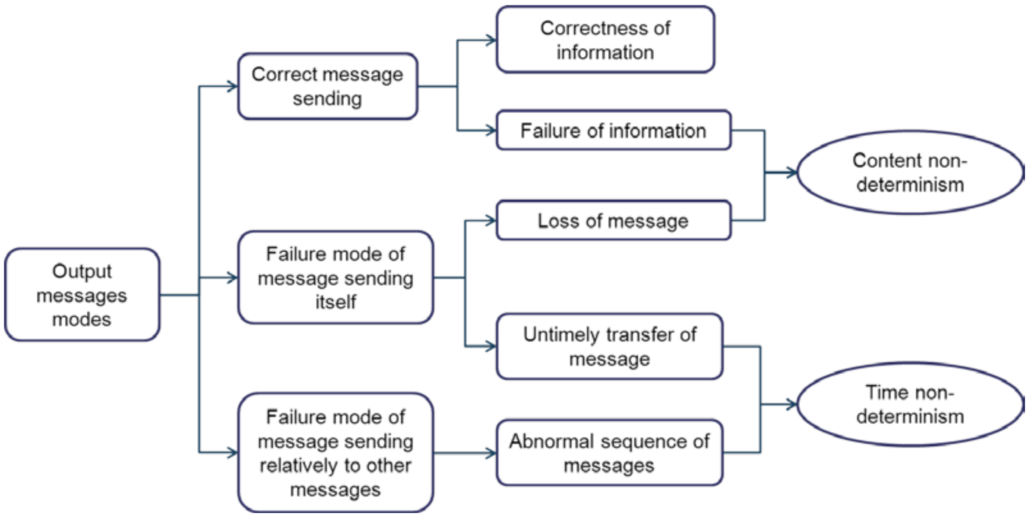
Output messages modes
Correct message sending
Failure mode of message sending itself
Failure mode of message sending relatively to other messages
Correctness of information
Failure of information
Loss of message
Untimely transfer of message
Abnormal sequence of messages
Content non-determinism
Time non-determinism

Figure 1: Overview of failure modes at logical level (from Bieth, 2013)

That classification relies on a description of a multi-core processor as a set of components (see Figure 2) exchanging messages.

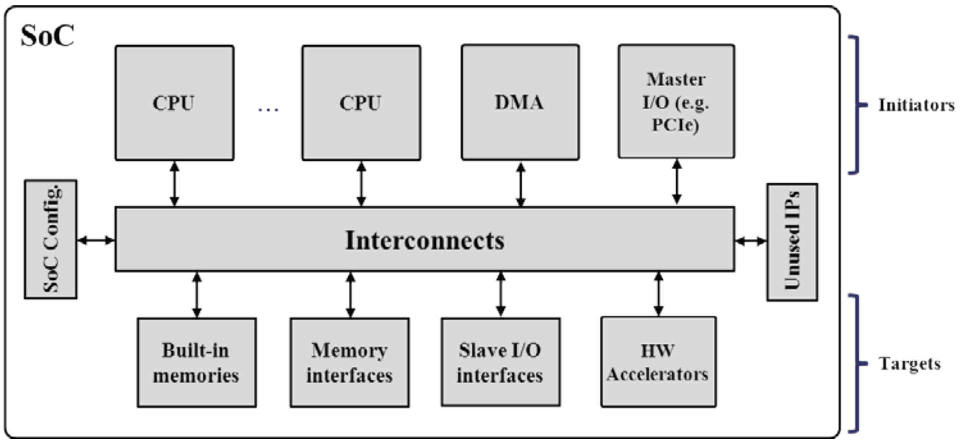

Figure 2. Overview of a multi-core processor at functional level

Focusing on timing-related failure modes, the starting point is the possible states of message transmission:

- The message is normally received,
- The message is lost,
- The message is incorrect; or
- The message is untimely received (with this condition being repetitive).

The causal factors for the failure of the message transmission are of two generic types:

- A structural or temporal disturbance of the message encapsulating information that prevents the entry into admissible states defined for the information,

- A disturbance in the transmission of the message (although it was sent at the correct time) so that the state transition is not realized, is realized in an untimely manner, or is realized between two states that should not be linked (forbidden state transition).

The failure modes are therefore classified into (Brindejonc, 2013):

- Loss of message: this mode represents the absence of message delivery when it should have been emitted. As an independent mode, this mode is meaningful if and only if the energy state at the loss of message is not an admissible logical state of the information. If the energy state is admissible (e.g., after a short circuit on the transmission line) the message will be interpretable by the receiver (e.g., as a zero value). A modeling decision (Bieth, 2013) leads to assimilating the above condition to an impossible transition of information to another state,

- Untimely transfer of message: this mode includes two sub-cases. The untimely can point to a message transfer that is in advance of the expected time. The other aspect is a late message, also referred to as abnormal latency,

- Abnormal sequence of messages,

- Untimely or forbidden transition of information: this failure mode is also denoted information corruption of erroneous information transfer,

- Impossible transition of information: this failure mode merges with several of the categories above depending on the specific conditions. For example, if this failure mode affects only a part of information burst, then it corresponds to erroneous information transfer. If it affects a complete information burst or several bursts, it corresponds to abnormal latency in information transfer.

The possible behaviors resulting from a multicore processor sending a message fall into three exclusive categories:

1. The message was delivered at the right time and in the right sequence. Either the content of the message is correct or it is erroneous,

2. Sending the message fails. This failure mode can be reached either because the message was never delivered or when the message is untimely transferred (too late or too early),

3. The message is sent out of sequence.

Through this simple example, two modes of non-determinism are extracted:

- Non-determinism related to content: this mode is associated with the transmission of erroneous information or the loss of output message(s),

- Non-determinism related to timing: this mode is associated with the untimely transfer of message or the abnormal sequence of messages.

Non-determinism related to timing can be caused not only by the failures associated with the processor, but also by behaviors compliant with the processor specification, such as missing application deadlines. On the other hand, the failures associated with the processor (systematic or random) are the only ones that can lead to content determinism. Therefore, in the framework of processors used in AEH, content determinism is ensured through the processor specification.

2.3 Failure Modes at Hardware-Software Interface

The software-hardware interface in a multicore processor is the source of additional and new failure modes. While these modes can be described as inherited from the comprehensive failure model described in the previous section, they are detailed below based on the responsibilities of the hardware towards the software:

The first responsibility of the multicore processor hardware is to realize the computation requested by the software, which is functionally described as:

- Get software instruction,
- Get data needed for the computation, and
- Push the computation results.

The failure modes associated with these functions are:

- No program instruction or data is retrieved,
- Erroneous instruction or data is retrieved,
- Latency in data delivery (or maximum execution time drift).

In an Integrated Modular Avionics (IMA) platform or multicore context in which more than one application is embedded, an additional responsibility of the hardware relates to partitioning of the software application. The associated failure mode can be expressed as:

- Tasks are not sequenced in the intended order across applications.

Finally, the non-respect of partitioning can lead to the following failure modes:

- Loss or blocking of program instruction output (equivalent to “no program instruction output”),
- Cross-corruption of two independent software applications (equivalent to “erroneous calculation output”),
- Latency in program instruction output.

Once the failure modes of the multicore processors are determined at black-box level down to the Intellectual Property, these failure modes are classified with respect to their impact on non-determinism, as it represents the primary safety concern (Bieth, 2013).

3. Determination of Effects

For the determination of safety impacts, a top-down analysis refines the need for determinism as it applies to the functions embedded on the multicore processor. The complementary bottom-up analysis is more traditional in addressing the sources of non-determinism that can be encountered in the design of a multicore processor.

3.1 Top-down Analysis in Multicore Processor Design

System design widely applies top-down methods, for example, when the selection of a processor is based not only on the evaluation of low level criteria such as performance but also on high-level non-functional aspects (e.g., leveraged buy-out, recurring engineering costs) that are driven by system requirements.

In the case of a multicore processor, the enforcement of determinism is required and is first and foremost a matter of fault identification and coverage. The use of a top-down approach, depicted on Figure 3, can bring additional high-level requirements pertaining to the execution model. This will ease the identification of issues related to non-determinism. These additional requirements could include the mapping of functions onto the commercial-off-the-shelf perimeter, including task allocation onto cores and the function’s use of shared resources. The execution model also relates to the operating system model used to execute the applications. This characterization of the resources and their usage is commonly known as the usage domain on an IMA platform.

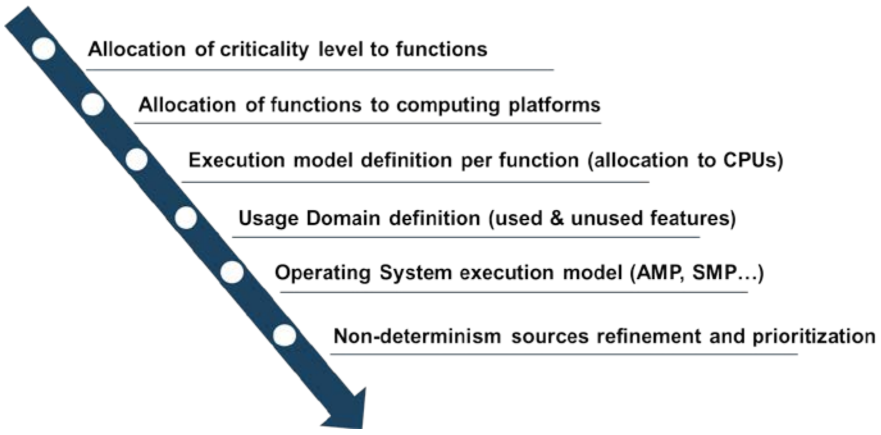
Allocation of criticality level to functions
Allocation of functions to computing platforms
Execution model definition per function (allocation to CPUs)
Usage Domain definition (used & unused features)
Operating System execution model (AMP, SMP…)
Non-determinism sources refinement and prioritization

Figure 3: Overview of top-down analysis in Multicore Design

The application of the top-down approach is a necessary step to set bounds on the investigation of sources for non-determinism as part of the bottom-up approach. While it facilitates industrial and certification processes by characterizing at high-level sources of non-determinism, this approach is not sufficient. The main benefits are to classify these sources of non-determinism so that the main contributor of non-determinism can be later identified, and to recommend the most relevant approach at system/architecture level prior to breaking down to component level.

In short, the top-down approach provides the selection criteria for the multicore processor itself and for selecting the intellectual property to be activated or configured within the multicore. This configuration defines the usage domain of the multicore in which determinism can be guaranteed.

For a generic design of avionic platforms with multicore processors, this usage domain can be viewed as new requirements to guarantee the correctness of the system to meet the global performance objective (e.g., bounded latencies). For IMA platforms, it corresponds to an extended usage domain compared to the existing one. Adding these system requirements will ease further the evaluation of the number of non-determinism sources, their types and their importance. Also, the available (or the lack of) processor mechanisms to mitigate their effects can be evaluated at an early stage of the development.

The top-down approach is developed along three axes: function or tasks allocation, execution model or software architecture, and system-on-chip (SoC) hardware architecture. The approach addresses both isolation and mitigation of sources for non-determinism through three main goals:

- The isolation of sources for non-determinism in the top-down strategy,
- The limitation of accessible usage domain in which non-determinism has to be reduced,
- The preparation of possible mitigation strategy if non-determinism sources are still present in the usage domain.

3.2 Bottom-up Analysis in Multicore Processor Design

The bottom-up approach, depicted on Figure 4, aims at obtaining some guarantees over the whole multicore processor behavior, from a systematic analysis of all the multicore processor components and their interactions. The analysis consists in studying sources of non-determinism, such as failure modes (including common modes), and worst-case behaviors for predictability concerns. The analysis is performed on each component regardless of other component's behavior.

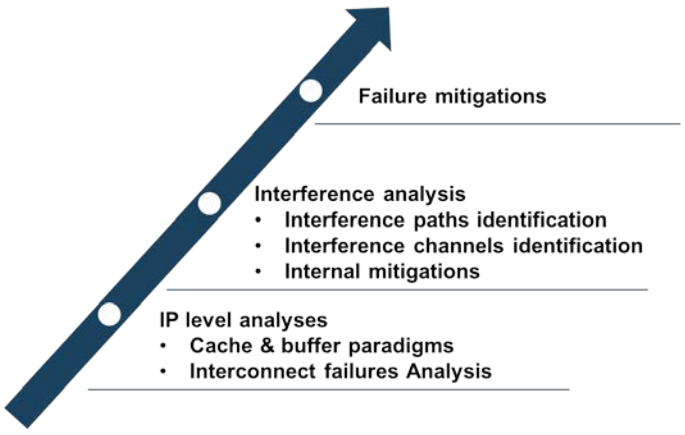
Failure mitigations
Interference analysis
• Interference paths identification
• Interference channels identification
• Internal mitigations
IP level analyses
• Cache & buffer paradigms
• Interconnect failures Analysis

Figure 4. Overview of bottom-up analysis in Multicore Design

This approach is the most commonly developed in the literature, and is included in the CAST Position Paper #32 (CAST, 2014). The main challenge is the need for information on the processor's architecture and behavior. While reaching a sufficient level of detail is often feasible for some components (e.g., the processor core), it is more difficult for key multicore processor components (e.g., interconnects). Such detailed complete set of information cannot typically be obtained from the device manufacturer. Therefore conservative hypotheses are commonly applied to proceed with the approach.

The bottom-up analysis is applied to the meta-model of multicore processor as described in section 2. The multicore processor is modeled as a set of autonomous intellectual property blocks connected by interconnects. Each intellectual property element may function either proactively ("initiators") or reactively ("targets"). A master intellectual property block may initiate traffic at any time, while a slave intellectual property block only initiate traffic to answer requests from a master intellectual property block.

A bottom-up analysis identifies many sources of non-determinism, which may originate from similar phenomena and mechanisms. Therefore, a generic classification of sources of non-determinism is useful prior to performing a bottom-up analysis.

3.2.1 Sources of Content Non-determinism

Similarly to any processor, a multicore processor performs two types of electronic operations: data processing and data exchange. Sources of content non-determinism affect the soundness of operations for processed data with respect to the usage domain. The associated non-determinism may result from persistent faults in the hardware, the use of hardware features outside of the specified domain assessed by the multicore processor manufacturer, the use of embedded microcode outside of the specified domain assessed by the multicore processor manufacturer, an update of embedded microcode, or a misconfiguration of the component with respect to the usage domain.

These faults must be assessed using structural failure mode analyses (e.g., failure mode and effects analysis or FMEA for random failures, except for ageing effects for which other statistical models may be used). Moreover, a previous study led by the European Aviation Safety Agency (EASA) explicitly recommended restricting the multicore processor usage to a usage domain that complies with the manufacturer's specifications (Bieth, 2010).

Regarding the integrity of exchanged and/or stored data, this integrity can be corrupted by external events, such as single event upsets, or internal hardware random failures. Their probability of occurrence may be assessed from statistical data based on the type and size of internal memory and buffers used to exchange the data and other reliability information. On recent multicore processors, error correcting codes (ECC) mechanisms are widespread to mitigate such effects.

In conclusion, as far as sources of content non-determinism are concerned, a multicore does not differ much from a single core processor. The main difference remains the increased complexity of the processor's internal architecture, behavior and configuration space.

3.2.2 Sources of Timing Non-determinism

The main sources of non-determinism associated with multicore processors are related to timing issues. Interferences are the most cited source of timing non-determinism, but it is not the only one. Sources of timing non-determinism are separated into two classes: sources that are local to each component and sources resulting from the combined activity of several master components on a multicore processor.

For sources that are local to each component, activities performed locally by components inside a processor are data processing and data transfers. For example, ECC checking is data processing, while row selection in a dynamic random access memory (DRAM) is considered as data transfer. While sources related to data processing closely depend on each component, sources related to data transfer are generic and can be refined according to the cache or buffer paradigms. Cache paradigm considers data transferred to facilitate or accelerate further accesses. A cache is closely coupled with a replacement logic that manages its content. Uncertainty on cache content is therefore a recurrent source of non-determinism. Buffer paradigm considers buffers used either to pipeline consecutive or concurrent accesses, or to hide the complexity of I/O internal logic. When a buffer is full, incoming requests are delayed and thus entail a global slow-down of the whole activity on the processor. Uncertainty on a buffer's capacity to handle incoming requests is a source of non-determinism.

Sources resulting from the combined activity of several master components on a multicore processor correspond to interferences.

3.3 Classification of Impact for Interferences

An interference analysis takes as input a set of interference paths (i.e. processor's configurations with given initiators and targets activated, see Figure 5), filters it according to the usage domain restrictions (platform and processor), identifies interference channels (i.e. interference paths for which interferences have been observed), and tags each interference channel as:

- Acceptable: interferences occur, but a bound has been found on the interference penalty, and this penalty meets the performance objectives for the equipment,
- Unacceptable: the interference penalty could be determined (i.e., a bound can be found) but it does not meet the performance requirements,
- Unbounded: the interference penalty could not be determined,
- Faulty: tests on the interference path have triggered a failure mode that was not discovered or documented by the manufacturer. Further investigation is required.

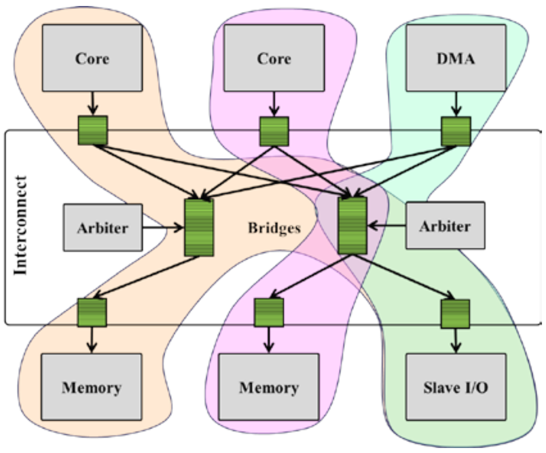

Figure 5. Example of interference paths

No more than one interference penalty can be defined per interference channel, but a same interference penalty may apply to several interference channels. In the first three bullet points, the interference channels are not associated with a failure mode, and are therefore only considered from the performance standpoint. In the fourth and last bullet point, the interference channel has triggered a failure mode and must be analyzed in-depth from a safety standpoint. Even if the failure modes are a concern for the equipment safety, they are addressed by several analyses that apply to complex COTS. Interference analysis can simply be aware of this eventuality and report them, if any are found.

The main objective of the interference analysis is to provide a trustworthy performance assessment on the processor. The main challenges are:

- Dealing with a high number of interference paths: in many cases, it will not be possible to identify all paths. Hence a subset may be covered, as long as a rationale can be provided to justify that the reduced coverage is sufficient. For instance, it can be shown that the gathered interference penalties apply to uncovered interferences paths,

- Obtaining a trustworthy interference penalty on a given interference path. This is challenging because of the high complexity of the hardware mechanisms, the eventuality of black-box and/or partially documented components within the processor, and the limitations of testing capacities (e.g., the impossibility of reaching heavy stress levels with given initiators). Defining tests space and clarifying its limits in terms of which situations are reachable and which ones are not may be helpful.

For both challenges, the recommendation is to agree with the approval authority on termination criteria to limit the complexity of the interference analysis. Meeting these criteria can be a sufficient condition to ensure the trustworthiness of interference penalties.

Various approaches and termination criteria can be proposed; they can be more or less formalized, and more or less computational. For instance, human-in-the-loop processes can be introduced, including experts' reviews, collaboration with the manufacturer, and collected in-service experience. The in-service experience can cover the processor itself or similar ones, such as the processors in the same series or using the same intellectual properties. These approaches may be developed alongside formal analyses performed on the whole processor, or specific components. The worst-case behavior of some components can be found, which is supporting the trustworthiness of interference penalty. However, the equipment provider performing the interference analysis should be aware that this might not always be the case. What is important is to know and document the limitations of the interference analysis, so that the rationale associated with the interference penalty is properly justified.

A specific concern points to processors that contain black-box components. From a performance's point of view, the following events can be feared:

- Unbounded behavior within the limits of the test's exploration space: for instance, it can be necessary to consider an infinite test space when its bounds are not clearly defined. Hence, asymptotic behaviors can be considered for test configurations that are not reached, but do not seem unreachable. These asymptotic behaviors may not be bounded, thus the interference channel becomes unbounded,

- Hidden mode changes and/or subcomponents' activation within the black-box component: these events introduce discontinuities in the processor's behavior. Having discontinuities is not a problem in itself, as long as their impact is covered. What is important is to secure the absence of singularities, so that the component's behavior is unknown. One of the results of the interference analysis dealing with black-box components can be the absence of singularities within a given limit, according to metrics defined over the test space.

The interference analysis mainly brings obligations in terms of results (have a sufficient coverage of interference paths) to identify interference channels and to provide trustworthy interference penalties for each of them. Exhaustiveness is probably not reachable using any one method, because of the "black-box" situation when using COTS. Nevertheless, trustworthiness should be reached by combining a semi-formalized approach, knowledge of the device from the multicore processor supplier, test campaigns, or any other method.

Finally, with this approach, the equipment provider is left the opportunity to propose and argue the methods, together with their level of formalization, automation and human intervention. This plan can be proposed and defended in front of certification authorities at an early stage of the certification process, and then be refined during the safe design phase of the 'V' cycle. As an output, the interference analysis produces the set of tagged interference channels. The unacceptable and unbounded channels can be studied with a mitigation objective in mind. Faulty channels are analyzed from a safety standpoint first, before being considered for mitigation.

3.4 <u>Effects of Interconnect Failures</u>

The functional failures associated with transaction services could lead to the faulty execution of the embedded software without raising any error. The associated analysis to determine the probability that these types of fault do not occur within the embedded system is called the interconnect integrity analysis and is part of the mitigation by determination of interconnect usage domain.

In case of undetected internal failure, the interconnect may propagate an error to the targeted core and potentially to an external monitor. The interconnect integrity is a mitigation against the propagation of such errors. In this case, the platform-level reliability may benefit from the interconnect integrity by sanctioning the target core for the transaction which corruption was stopped from propagating.

The local effects include:

- Masked error: the subsequent block may ignore the error if it is only evaluated when other signals have a specific value,
- Corrupted flow control unit,
- Lost flow control unit,
- Erroneous sending of flow control unit: the flow control unit is sent to the wrong port,
- Delayed flow control unit transmission,

- Blocked version control (VC) buffer.

The global or system-level effects include:

- Violation of Quality-of-Service (QoS), which can be temporary or permanent,
- Loss of packet, including the condition wherein the packet is delivered to the wrong recipient,
- Corruption of packet,
- Corruption of return route, which is relevant when acknowledgement is expected,
- Blockage of VC buffer.

4. Mitigation Techniques

4.1 Mitigation Techniques per Feature

Mitigation techniques can be proposed per multicore processor features that impact the design in safety-critical systems:

- Variability of Execution Time,
- Conflicts among Services and/or Transactions,
- Cores Interconnect Switch,
- Cache Architecture Structure,
- Shared Services,
- Shared Data,
- Inter-core Interrupts,
- Access to Peripherals,
- Programming Languages.

Table 1. Mitigation Means for MCP Features

MCP Feature	Mitigation Means
Variability of Execution Time	Worst case execution time (WCET) strategy for assessment, measurement, and continuous monitoring.
Service/transaction Conflicts	Software-controlled scheduling of tasks or processes.
Cores Interconnect Switch	Interconnect usage-domain definition.
Cache Architecture Structure	MCP-related cache management and cache consistency verified by trusted and privileged software.
Shared Services	Similarly to the implementation of Airborne software Programming Interface (API), services are offered via a trusted and privileged software.
Shared Data	Application of design rules. For example, all data that is not explicitly shared among cores is automatically tagged as private and duplicated in the cores.
Inter-core Interrupts	Interrupts are accepted only when expected (wait-for-interrupt is a rule-based implementation), or restrictions are placed on the use of inter-processors interrupts.
Access to Peripherals	Allocation of shared Input/output configuration, or shared memory space via trusted and privileged software. This allocation can be done either directly or via configuration controlled configuration tables.
Programming Languages	Determination of an adequate strategy for multi-processing programming, such as pre-emptive versus co-operative strategy.

4.2 Online Error Detection, Recovery and Repair Schemes

In multicore processor architectures, memories spanning a large portion of the die are successfully protected using error correcting codes. The focus of online error detection is thus the other elements: the cores, the memory hierarchy control logic (memory consistency) and the interconnect logic. Applicable online error detection techniques can be classified as:

- Redundant execution approaches. These techniques use the inherent replication of cores and threads in MCPs,

- Periodic built-in self-test (BIST) approaches. These techniques leverage the built-in test mechanisms traditionally used for the detection of manufacturing defects. They have a historically poor coverage for in-service failure (which is key in the context of process shrinking) and do not protect against soft errors,

- Dynamic verification approaches (also known as online testing). These techniques check that system-invariants are maintained, independently from the implementation,

- Anomaly detection approaches.

Furthermore, online error recovery techniques can be classified into two broad categories:

- Forward Error Recovery (FER). These techniques detect and correct errors without requiring rollback/recovery mechanism. They must use redundancy (e.g., TMR lockstep),
- Backward Error Recovery (BER). These techniques are based on periodically saving system state (checkpointing) and rolling back to the latest validated checkpoint following the detection of a failure.

4.3 Mitigation of Interferences

Interferences mitigation comes at the end of the process and applies to interference channels. As an extension, we consider that mitigation techniques can also be injected in the initial definition of the usage domain, before performing the interference analysis, with the goal to make the analysis simpler.

The mitigations can be internal, and thus be reported as additional restrictions within the usage domain and/or specific configuration tradeoffs. Alternatively, the mitigations can be external, i.e. hosted by a third-party hardware.

The case of an internal mitigation is the richest because it implies new iterations with the interference analysis. In the first iteration, an internal mitigation takes care of the specificities of the hardware and software architectures to influence the definition of usage domain. Additionally, it may orient the equipment's development toward one of the following paradigms:

- Apply interference reduction techniques, use specific hardware features (if any), but do not try to bound interferences with software,
- Ensure that interferences are bounded within the interference channel by internal means, or
- Ensure that interferences are eliminated by construction.

Each of the three paradigms above has their advantages and drawbacks. Selecting one of them has a strong impact on the rest of equipment's lifespan, as it is significantly complex to recover from such a choice.

The case of external mitigation is more straightforward. Indeed, an external mitigation for an interference channel consists of a detection means and a sanction strategy, in accordance with safety objectives. Detection is the bottleneck of an external mitigation strategy and detecting that an interference channel has 'too much' interferences is not well defined. A proactive detection strategy can be proposed and simply deployed. For instance, hardware events can be monitored at runtime, and unexpected events, or usage domain violations can entail errors that are handled by health monitors, either internal or external. However, it becomes difficult to give a rationale about the coverage of situations with unintended interferences.

Another approach consists in addressing detection means at a higher level, for instance by monitoring deadline misses. However, it becomes arguable to assimilate this detection as a consequence of interferences. Finally, a long-term mitigation could also consist in collecting statistics on the processor's behavior, both at hardware and software levels. These statistics feed a database maintained by the equipment provider, who can reuse such information in other equipment's projects. Hence, unintended events such as deadline misses could be documented with a snapshot as precise as possible of the hardware and software conditions in which the problem occurred, so that the equipment provider could investigate.

As a conclusion, no external mitigation strategy dealing with interferences seems to fit optimally. Interference mitigation should thus combine several detection means and propose adapted sanctions, in accordance with the safety objectives.

Conclusion

The increased level of complexity introduced by MCPs is pushing the limits of the feasibility of conventional bottom-up analyses. To obtain a scope adjustment on the level of effort for the bottom-up analysis, the performance of two additional steps prior to the bottom-up analysis is recommended. These steps include an independent determination of applicable safety and real-time constraints and a top-down analysis targeting non-determinism.

Substantiating information for these analysis is obtained from system-level safety assessment processes: determining the failure modes of hazards, determining the effect and implementing mitigations. The application of these processes to multicore processors is in particular of relevance when considering interferences.

Out of the various sources of non-determinism in MCPs, interferences are undoubtedly an undesirable behavior for usage in avionics equipment, regardless of the number of cores. Interferences make the processor's performance assessment complex to achieve and therefore raise safety issues. The main risk is to trigger timing failures on the whole equipment, even if data failure modes might also be discovered during the interference analysis.

In line with the approach to safety analysis for non-determinism, the authors propose a safety process to specifically address the question of interference. The process is inspired by partitioning analyses performed on integrated modular avionics systems. Because of the wide variety of equipment, processors being used, level of criticality, and needs in terms of real-time requirements, no off-the-shelf generic solution seems to be practicable. Therefore, such a process ought to be instantiated on a case-by-case basis, and presented in the certification plan as soon as possible. Its definition should contain acceptability and termination criteria that will be applied during the safety assessment phases.

Several points need be considered. First, no limitation on the number of cores seems relevant, even if a processor that embeds more cores makes the interference analysis more complex. Second, exhaustiveness of analyses is not likely to be reachable on COTS processors containing either highly complex and/or black-box components. Hence, confidence should be obtained by combining several analysis methods, e.g., semi-formalized analyses coupled with information shared with the manufacturer and expert feedbacks on the processor, or similar ones in avionics. Safety experts must drive these methods. Similarly, the interference analysis should be performed with final applications, if possible. When it is not the case, representative applications fitting with the usage domain can be used, even if trustworthiness may be more complex to achieve.

Finally, interference mitigations, especially interference elimination techniques, do not displace the need for interference analysis, which brings evidence of interference control.

References

Certification Authorities Software Team (CAST), “Multi-core Processors,” Position Paper #32, May 2014.

L. Condra, G. Horan, H. Forsberg, D. Matthews, J. Peterson, A. Martin, S. Barbagelata, K. Lillestolen, D. Redman, B. Petre and R. Manner, “AFE75 COTS (AEH) Issues and Emerging Solutions Report,” FAA Report, 2014.

P. Bieth and V. Brindejonc, “COTS-AEH – Use of complex COTS (Components-Off-The-Shelf) in Airborne Electronic Hardware – Failure Mode and Mitigation,” 2013.

SAE, ARP-4754A, “Guidelines for Development of Civil Aircraft and Systems,” 2010.

SAE, ARP4761, “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,” 1996.

V. Brindejonc, G. Marcuccilli and S. Petit, “System FMECA in the framework of ISO 26262,” Proceedings of Lambda-Mu 17, October 2010.

X. Jean, S. Girbal, A. Roger, T. Megel, V. Brindejonc, “Safety Considerations for WCET Evaluation Methods in Avionic Equipments,” Proceedings of the 2015 DASC, 2015.

X. Jean, M. Gatti, G. Berthon and M. Fumey, “The Use of Multicore Processors in Airborne Systems,” CCC/12/006898 rev. 07, 2012.

Acknowledgments

This research has been performed under FAA Aviation Research Division’s Software and Electronics Section contract SDS-TO05. The technical monitor for this research was Srini Mandalapu of ANG-E271 Software and Electronics Section.