Tech Note: Multicore IMA Requires Multicore Interference Mitigation

Overview

Historically, a handful of safety critical operating systems have claimed to support Integrated Modular Avionics (IMA). Recently, there has been an increase in claims of support for IMA on multicore processors. Some of those claims include support for:

- robust partitioning, as required by RTCA DO-297 Integrated Modular Avionics
- symmetric multi-processing and bound multi-processing
- conformance to the FACE™ 3.0 technical standard, including all operational requirements for multicore processors
- commercial off-the-shelf (COTS) RTCA DO-178C DAL A certification evidence

To understand the challenges and potential solutions for any of the above requirements, a more in-depth exam-ina-tion is needed, much more than just tossing around acro-nyms and RTCA document numbers like buzz words. This examination starts with answering some ba-sic questions and then moves onto more complex ones.

There have been IMA solutions for single-core processors for many years. So, why is multicore a big challenge?

The primary duty of an IMA system is to enforce robust partitioning that allows multiple applications to share a platform and its resources. For single-core processors, that was accomplished with partitioning in both space and time. Memory space partitioning enforcement is based on the memory management unit (MMU). This is similar to how cloud servers keep different user spaces separate. Time partitioning enforcement is based on the use of a partition time scheduler, such as defined in the ARINC 653 specification. Transitioning to multicore processors adds another dimension to the enforcement mechanisms required for IMA in the form of managing resources shared by the processor cores. Contention for shared resources, such as interconnect, memory, DMA, and I/O, results in multicore interference that must be mitigated. That is a significant enough problem to warrant an entire position paper from the Certification Authorities Software Team (CAST-32A) dedicated to identifying prob-lem areas relating to safety, perfor-mance, and integrity of software executing in an airborne multicore system. If the IMA system has a certification requirement, the enforce-ment mechanism must comply with the highest level of criticality in the system.

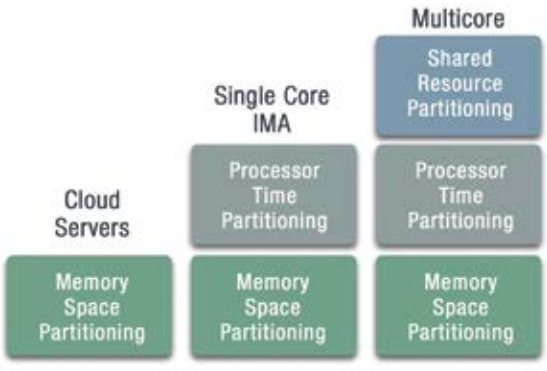


Figure 1. Multicore IMA requires partitioning of shared resources through interference mitigation in addition to partitioning of time and memory space on each processor.

Why is addressing multicore interference so crucial?

Multicore interference can cause dramatic variations in worst-case execution time (WCET) of safety-critical appli-cations. Tests on a quad-core processor show that a single interfering core can increase WCET on another core by a factor of 8x. With multiple sources of interfer-ence from multiple cores, increases in WCET of over 12x have been observed just from the cores accessing DDR memory over the on-chip interconnect. The result is that a low-criticality application could prevent a high-criti-cality appli-cation from completing its intended function before its deadline.

How does this relate to IMA?

One of the key characteristics of an IMA application, as specified in DO-297, is that applications are inde-pendently modifiable. This requires each application to be modifia-ble with little or no impact on other appli-cations and the platform resources and modules. The capability to inde-pendently modify applications is nec-essary to meet the IMA goals of providing cost-effective upgrade paths and introducing new operational capabil-ities without retesting and reverifying the entire system. This capability also al-lows a system integrator to add

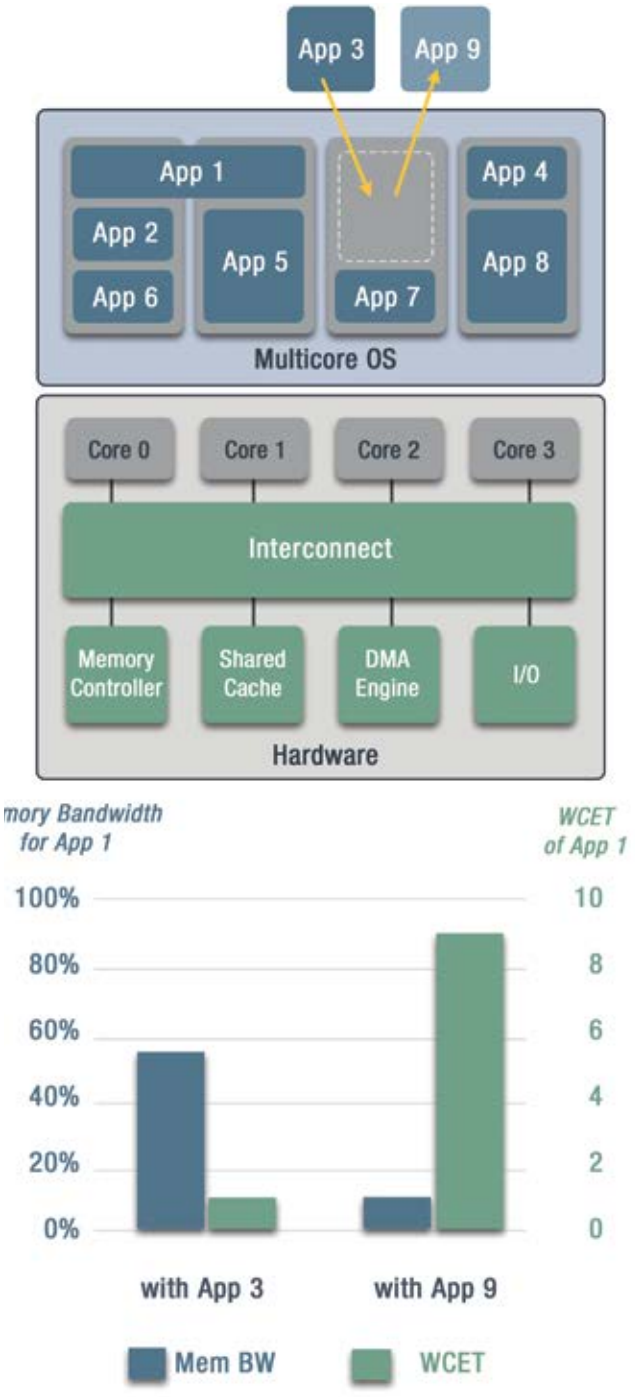


Figure 2. Impact on Critical Application 1: *Contention for shared resources in an IMA system must be carefully monitored and managed. For instance, when applica tions are added or modified, the interference patterns can change. In the above example, App 1's memory bandwidth and WCET are acceptable with App 3 executing simultaneously on Core 2. When App 9 replaces App 3, without management of the shared multicore resources, the memory bandwidth of critical App 1 is reduced to 10% and its WCET increases 9 times, all as a result of the unmanaged use of shared resources.*

or modify applications on top of an IMA platform designed and manufactured by its supplier. Multicore interference is a major obstacle to this goal because the interference from the modified applica-tion could cause an existing applica-tion not to execute correctly or vice versa.

How do operating system providers and system integrators typically address multicore interference?

Some operating system (OS) providers have left the challenge of multicore interference to the system integrator. That makes the challenge infinitely harder because the solution cannot be bolted on at the application level or even the IMA platform level. The same way that MMU sup-port and partition schedules need to be implemented in the OS, the enforcement of the multicore interference mit-igation needs to be in the OS. In all cases, that enforcement mechanism must be at the highest criticality level assigned to the system.

Other OS providers have suggested that a simple solution to multicore interference is to merely increase the size of the partition time windows hosting the applications. Such an increase may result in requiring additional computer resources. More importantly, it does not solve the problem that when new applications are added or existing applica-tions are modified, all the time partition windows need to be examined and possibly adjusted. Nor does it satisfy the IMA objective that applications are independently modifia-ble.

Some subsystem providers attempt to reduce multicore interference by modifying applications to assign all I/O to a particular core or by limiting which cores are active simul-taneously. Those and other types of special case solutions do not support a true IMA system because the mitigation breaks when applications are modified or added.

What is the solution for multicore interference in an IMA system?

Multicore IMA systems require a general solution for miti-gating multicore interference. To be a general solution that allows modifications and additions to the set of appli-cations, the mitigation solution needs to guarantee a portion of shared resources to each application based on its execution requirements. In an IMA system, managing shared resources is the responsibility of the IMA platform, which includes the OS. The OS is the software component

closest to the hardware, so it is in the best position to manage the hardware re-sources.

In the INTEGRITY-178® tuMP™ RTOS from Green Hills Software, the DAL A-compliant Bandwidth Allocation and Monitoring (BAM) functionality provides such a general solution to multicore interference. Using BAM, the system architect can set a guaranteed allo-cation of bandwidth to shared resources for each core based on the execution requirements of the applica-tion(s) running on that core or group of cores. BAM enforces the bandwidth allocation with very fine gran-ularity so that applications have consistent availability to the shared resources.

The approach used by BAM contrasts with an alterna-tive coarse-grain approach where the time granularity is a sig-nificant portion of the duration of a partition time window. With such an approach, an application that exceeds its threshold is suspended and unable to resume execution until its next scheduled time win-dow. Such an approach can be sufficient to bound applications not behaving as intended, but it does not work well for an application that is correctly operating but periodically creates a great deal of interference. This approach does not manage the interference, but only limits the overall execution time of the interfer-ing application. Other applications running concur-rently with the interfering application could still miss their deadlines.

BAM can mitigate applications with periods of high re-source consumption as well as misbehaving interfer-ers. In that way, applications need not be developed to target a specific shared resource access pattern, such as a particular DDR-access profile. Instead, appli-cations are permitted to have naturally occurring peri-ods of execution that require higher frequency DDR accesses. BAM provides a mechanism to smooth out those periods to mitigate potential interference impacts.

What other types of support are needed for multicore IMA systems?

To achieve the throughput and SWaP benefits of multi-core IMA solutions, the software architecture needs to sup-port high utilization of the available processor cores. The more flexible the software multi-processing archi-tecture, the more tools the system architect has to achieve high utilization.

The simplest software architecture for a multicore-based system is Asymmetric Multi-Processing (AMP), where each core runs independently, each with its own OS or a guest OS on top of a hypervisor. Each core runs a different application or set of applications with little or no meaningful coordination between the cores in terms of scheduling. This decoupling can result in substantial un-derutilization due to lack of load balancing, difficulty mit-igating shared resource contention, and the inability to perform coordinat-ed activities across cores such as re-quired for comprehen-sive built-in test. Contrary to some claims, AMP does not reduce the multicore interference problem. Although each processor core is scheduled in-dependently, there remains coupling over the shared interconnect, shared memory controllers, etc. In fact, an AMP-only solution with multiple OSes may even increase the risk for interference on a mul-ticore SOC due to inde-pendent system functions executing concurrently at dif-ferent safety levels.

The modern alternative is Symmetric Multi-Processing (SMP), where a single OS controls all the resources, in-cluding which application threads run on which cores. Bound Multi-Processing (BMP) is an enhanced and re-stricted form of SMP that statically binds an application's tasks to specific cores, allowing the system architect to tightly control the concurrent operation of multiple cores. BMP directly follows the multicore requirement in ARINC 653 Part 1 Supplements 4 & 5, which includes:

"Multiple processes within a partition scheduled to exe-cute concurrently on different processor cores." This means that the software architecture needs to enable a partition (ap-plication) with multiple threads to run across multiple cores in parallel. The Future Airborne Capability Environment (FACE™) technical standard, edi-tion 3.0, requires support for ARINC 653 Part 1 Supple-ment 4.

How can OS companies claim to support IMA, DAL A, and multicore systems without a general solution to multi-core interference and full support of SMP or BMP?

The word "support" can be a bit squishy both in the de-gree of support and in which features are supported sim-ultaneously. For example, support for IMA could be with only one core of a multicore processor running at a time. Likewise, some companies started supporting multicore

processors with just AMP, not SMP or BMP. Even when SMP or BMP is supported, it might only be for virtualized Linux applications, not safety-critical or security-critical applications. Such guest OS Linux-based SMP offerings often dedicate a core or group of cores to run Linux during the entire major frame, which is a significant safety and security vulnerability. Claims of mitigation solutions for multicore interference could be merely allotting 10x processing time to a partition's time window so that the application finishes despite the interference, which is not mitigation at all. System integrators need to ask if they can get it all simultaneously: DAL A applications running in SMP or BMP mode on multicore processors with an IMA platform that provides robust partitioning and mul-ticore inference mitigation that enables the system inte-grator to modify or add applications without retesting and reverifying the entire system.

The bottom line is that an operating system can't claim to be an IMA OS if it can't guarantee the WCET for every IMA application. For multicore systems, that means the OS must provide robust partitioning with DAL A me-cha-nisms that enforce the utilization of shared multicore resources in order to mitigate multicore interference. Any claims of robust partitioning without such mechanisms are just unfounded claims. The mitigation solution needs to be independent of the applications so that the system integrator can modify or add applica-tions with little to no impact on other applications. Only then can they achieve the goals of a true IMA system.

Green Hills® SOFTWARE

Corporate Headquarters

30 West Sola Street • Santa Barbara, CA 93101
ph: 805.965.6044 • fax: 805.965.6343 • email: info@ghs.com • www.ghs.com

European Headquarters

Fleming Business Centre • Leigh Road • Eastleigh • Hampshire S050 9PD • United Kingdom
ph: +44 (0)2380 649660 • fax: +44 (0)2380 649661 • email: info-emea@ghs.com

Safety & Security Critical Products

34125 US Hwy 19 North • Suite 100 • Palm Harbor, FL 34684
ph: 727.781.4909 • fax: 727.781.3915 • email: info-sscp@ghs.com

Green Hills, the Green Hills logo, INTEGRITY, and tuMP are trademarks or registered trademarks of Green Hills Software in the US and/or internationally.
All other trademarks (registered or otherwise) are the property of their respective owners.
© 2019 Green Hills Software. 0719