The Multi-Core Challenge: A Practical Approach to CAST-32A & AMC 20-193 Compliance

Multi-core processors (MCPs) offer increased computing power without significant increases in Size, Weight, and Power (SWaP) which makes them appealing to the developers of embedded applications across the sectors. But safety-critical, hard real-time applications are closed loop, meaning that they are allowed only a tight time window to gather data, process that data, and update the system. Consequently, multi-core processors have two key characteristics that make their adoption a challenge.

Unlike single processor applications there is no efficient algorithm that will find an optimal schedule of multiple tasks on two or more processor cores such that all tasks meet their deadline. Only by imposing some level of restriction to their use can a Worst-Case Execution Time be guaranteed (Figure 1).

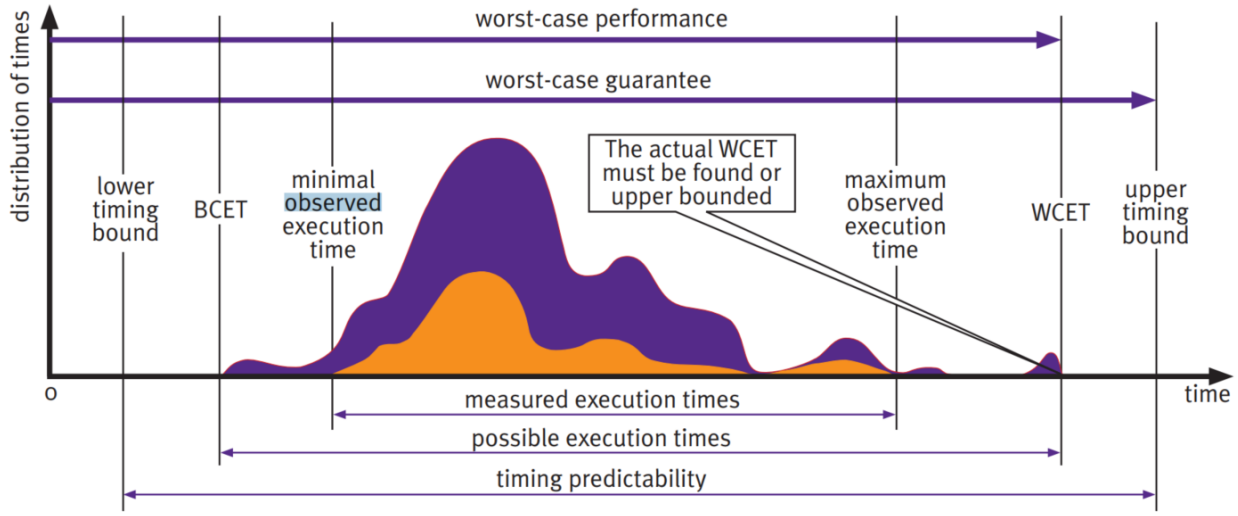

Figure 1: The worst-case execution time in context

Exacerbating that problem, hardware interference can occur in multi-core processors anywhere hardware resources are shared. For example, Figure 2 shows a shared hierarchical memory. Interference is possible in many places, and these interference channels cause the execution-time distribution to spread. Instead of a tight peak, the distribution of execution times becomes wide with a long tail. Sharing of critical resources, such as L2 cache and a memory subsystem, can impact the execution time of software and result in unsafe failure conditions.

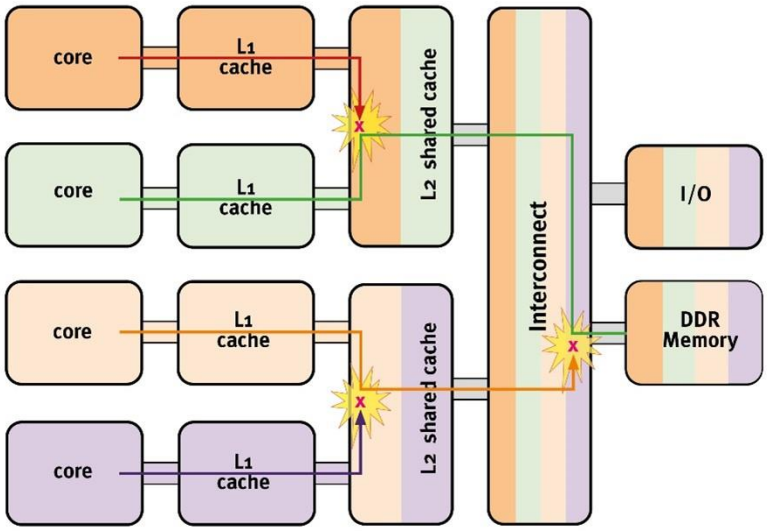

Figure 2: A representation of hardware interference in shared hierarchical memory

CAST- 32A & A(M)C 20-193

Compliance with the objectives of DO-178C, “Software considerations in airborne systems and equipment”, is the primary means of obtaining approval of software used in civil aviation products.

CAST-32A, “Multi-core processors”, is a position paper that was written by the Certification Authorities Software Team (CAST). CAST were a group of civil aviation regulatory officials from across the world, and their position papers look to clarify issues that relate to the formally established standards in that sector. CAST-32A, the last CAST paper, addresses the application of MCPs in DO-178C compliant projects and is supplemental to that standard (Figure 3).

CAST papers do not represent official guidance, but they are authoritative, and their recommendations are generally integrated into subsequent formal standards. The recommendations of CAST-32A (with amendments in recognition of feedback) have been integrated into EASA AMC 20-193. FAA AC 20-193 is expected to be very similar to AMC 20-193, and the pair are known collectively as A(M)C 20-193 (Figure 3).

The recommendations in these documents require developers of certifiable, safety-critical applications to solve the scheduling and hardware interference challenges. They must demonstrate that the systems they deliver will safely perform their intended functions by detailing how MCPs are applied in their systems, and how their safe operation is assured.

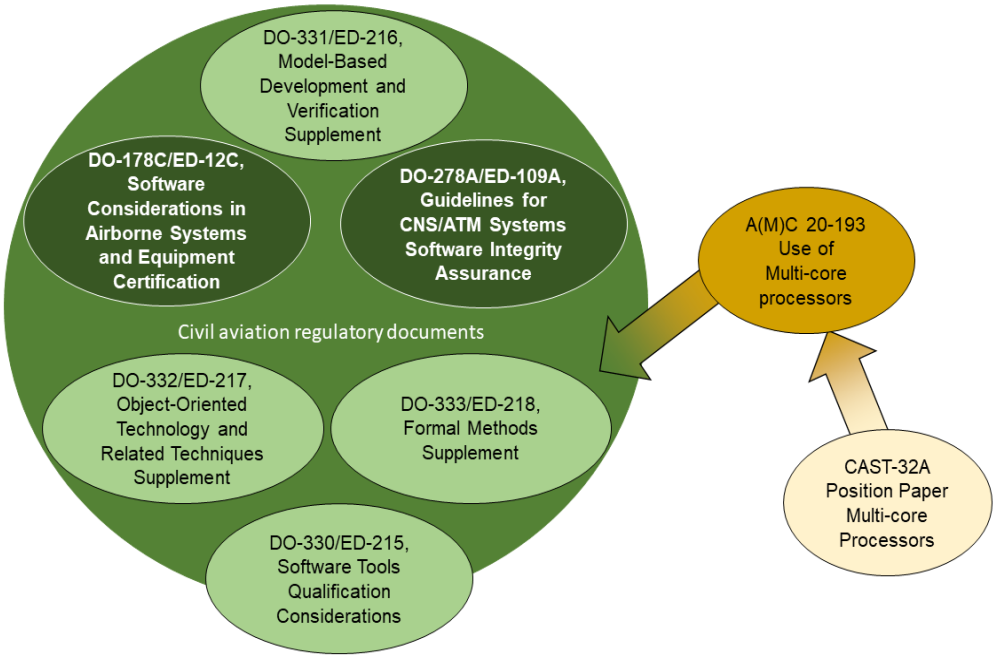

Figure 3: A representation of relevant civil aviation regulatory and advisory documents

Overcoming the Challenges

The functionality of a CAST-32A compliant system is reliant on WCET calculations and measurements that are only possible for all practical purposes if robust partitioning is in place. That robust partitioning is dependent on correct hardware and RTOS configuration.

PikeOS by SYSGO is an RTOS including a hypervisor-based partitioning for both resources and CPU time, and supports multi-core processors.

The LDRA tool suite provides the facilities to demonstrate that the resulting application leverages the configured operating system to deliver the required compliant outcome.

Building a Compliant, Safe, and Secure System with PikeOS

PikeOS combines a microkernel, which ensures the real-time behaviour and basic resource management with a hypervisor. The hypervisor ensures safety by way of separation and controlled information flow.

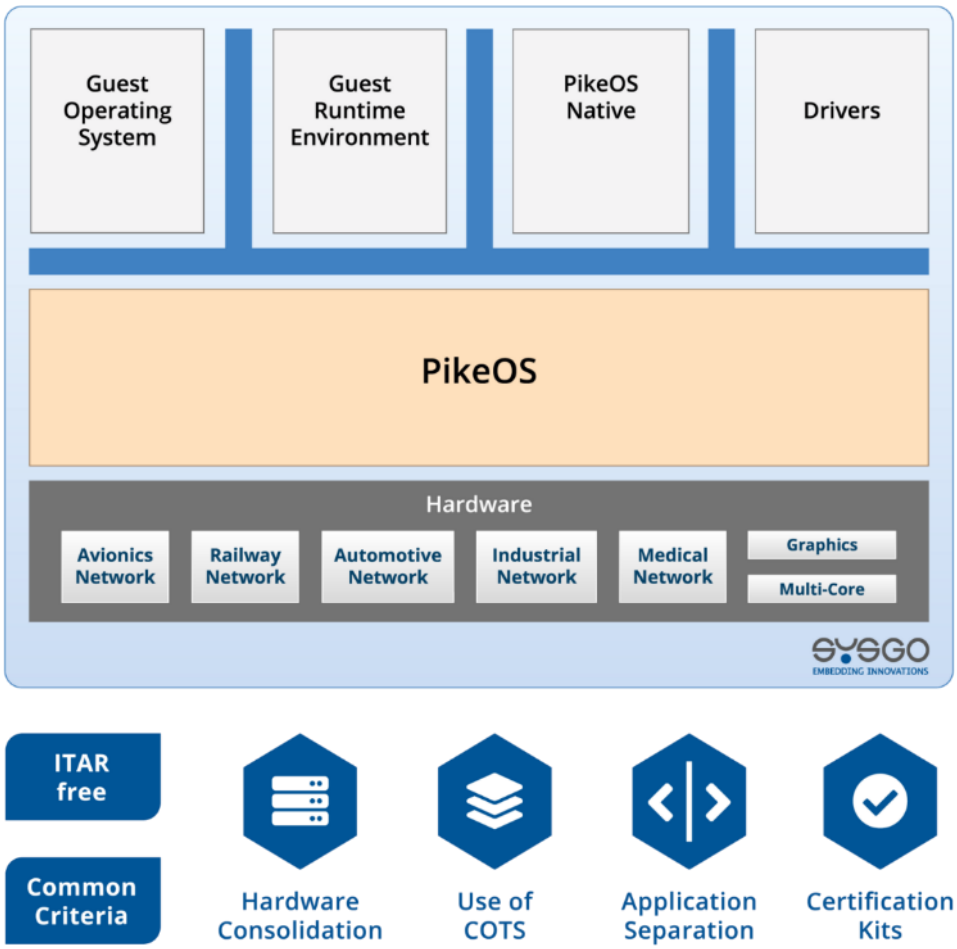

Figure 4: PikeOS general view

The basic components in the PikeOS hypervisor are virtual machines, which are called ‘resource partitions’. Partitions may host either a full-blown guest operating system like Linux or a run-time environment like POSIX or ARINC 653. The separation kernel architecture of PikeOS allows applications with mixed levels of criticality to coexist on the same system.

Addressing Cache Contention

Caches can be an essential cause of interference in an MCP. As L2 or L3 caches may be shared between cores (depending on the architecture), significant performance impact on running partitions can occur. For instance, when one core’s data set completely fills a shared L2 cache, with other cores also attempting to use it, effective cache performance can be degraded to that of the data bus. One malfunctioning application can similarly affect the performance of others as cache access might become limited.

There are a number of possible mitigations offered by PikeOS:

1. The division of the cache between cores, so that the use of the cache is separated into blocks per application. In the event of an application malfunction only the block of cache assigned would be impacted.
2. Cache bandwidth monitoring could be implemented to allow for a malfunctioning application to be sanctioned, e.g. restarting or stopping the offending application. This could also be used in unison with the division of the cache, to further negate the impact of an application malfunction on cache.
3. Finally, cache-colouring could be utilized at Platform Support Package (PSP) level, *provided there is adequate hardware support for it within the platform*.

Limiting Memory Contention

RAM may become saturated by an application, through normal use or malfunction. Hardware malfunctions, or a malfunctioning device driver, may also generate continuous data bursts via DMA/PCIe agent. The results are the same: memory overload can increase partition execution times, leading to the failure of time-critical applications.

The possible mitigations within PikeOS are:

1. Monitor memory bandwidth using performance counters and apply a sanction policy to the offending application - e.g. shutdown, restart.
2. Limit memory bandwidth of offending DMA/PCIe agents.
3. Implement kernel-level driver to monitor and control DMA requests.
4. Map virtual memory spaces to physical memory in order to reduce interference effects and risk.

Avoiding Interrupt Overload

Another risk inherent with MCPs is interrupt overload, causing the slow-down of partitions across both the core that is managing the interrupt and the other cores, where interrupts may be shared. The cause of such an overload can reside within hardware or software, including abnormal interrupts generated by other hardware but also throughout normal operation, considering interrupt operations are asynchronous when compared to the rest of the system.

Mitigations include:

1. Implement an interrupt limiter in the PSP to control the effects of spurious interrupt bursts.
2. Use the PikeOS time partitioning facilities to schedule execution of drivers during non-critical sections.

Better Scheduling with Time Partitioning

Another possible risk that requires careful management is that of the scheduler resource. Traditionally, there may be an inherent execution performance risk due to the overhead of the scheduler across multiple processor cores, as called by an application or driver.

PikeOS, on the other hand, distributes time by means of time partitioning.

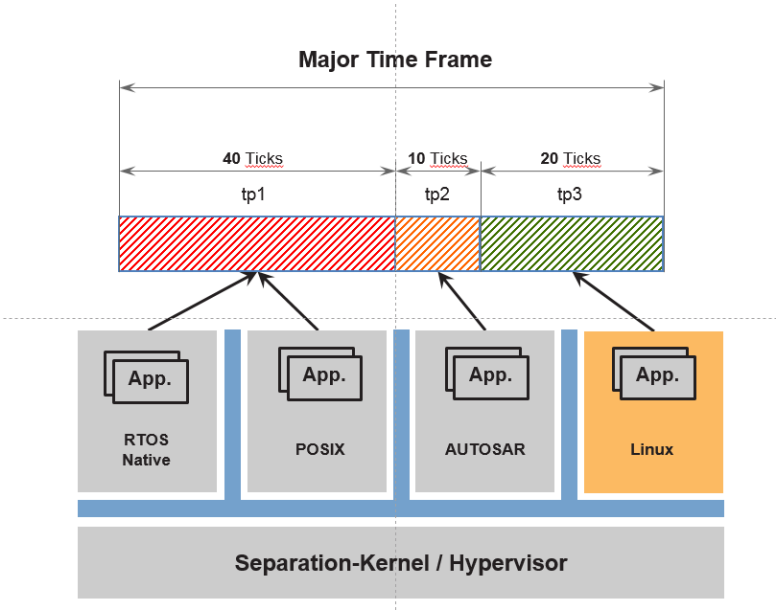

Figure 5: PikeOS Time Partitioning

PikeOS uses a two-stage scheduler. The first-level scheduler divides the available time into time partitions and assigns one or more resource partitions to a time partition. The second level scheduler schedules the threads running in each partition based on priority and First in First out (FIFO). A defined sequence of time partitions are grouped into a major time frame and executed cyclically whenever a major time frame has finished.

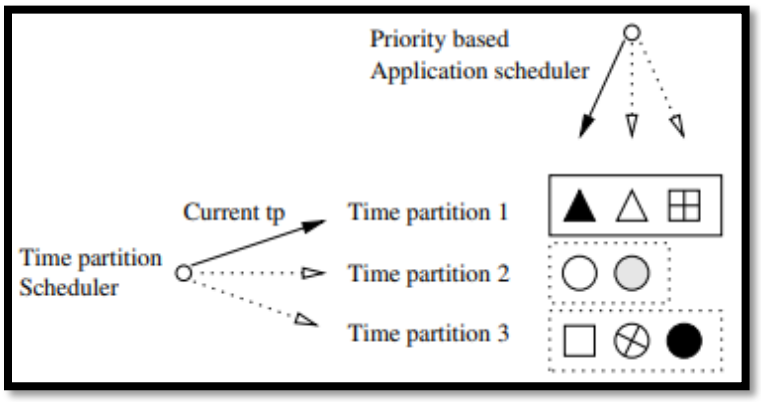

Figure 6: PikeOS scheduler

PikeOS' graphical time partitioning can be used alongside careful design to mitigate scheduling risks. This can also be extended through the assignment of partitions to one or many processing cores. PikeOS partitioning analysis, provided within the product's certification kits, provides guidance on how best to design a solution with least impact on system partition switch time.

Figure 7 shows a possible configuration of the PikeOS partition scheduler. The assumed platform is based on a quad-core (A, B, C, D) MCP. The major time frame is divided into three partition windows. One critical single core application (3) shall have exclusive access to one of

the cores and have exclusive access to the entire platform during its time window. One performance-demanding partition shall have exclusive access to the remaining three cores during its time window. One time slot is shared between two resource partitions. One partition running on two cores and another running on one core. The configuration provides a maximum level of isolation for the critical application by accepting a waste of CPU time. Partition 3 is the only partition executing on core 'C' and during the time slice of time partition 3 there is no other partition execution. This eliminates any interference on hardware and software level.

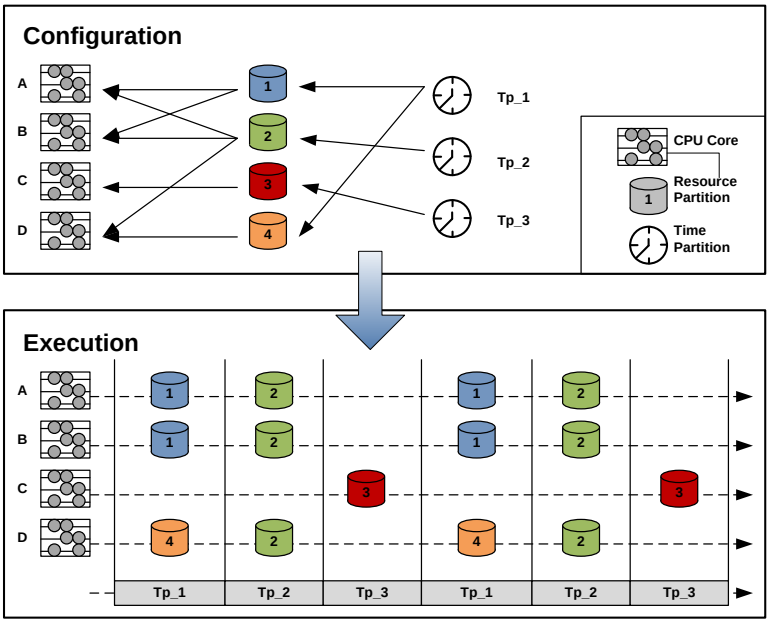

Figure 7: Example Configuration and Run-Time Model using the PikeOS Partition Scheduler

Accessing Operating System Objects

Operating System objects are also shared resources within the context of a multi-core processor, thus there is a risk of contention between applications. This could lead to an impact on execution time of any partition, due to competition for OS object access.

To mitigate this:

1. PikeOS includes fine-grained locking to limit critical sections and therefore insure coherency of individual OS objects. For example, ARINC 653 avionics partitions running in parallel never conflict on OS kernel objects.
2. WCET analysis of PikeOS services is also done with contention of locks for relevant services.
3. PikeOS partitioning analysis is provided.

Tackling IPC Contention

Depending on the method of Inter-Process Communication (IPC) which is utilized, there is a risk that partitions **not** involved in IPC are impacted.

PikeOS can provide IPC via sampling and queuing ports which not only isolate IPC overhead to the partitions involved, but also present no performance overhead whilst the IPC channel is quiet.

Queuing ports operate on a FIFO principle, whereas sampling ports provide a single message buffer that is automatically updated by a write operation.

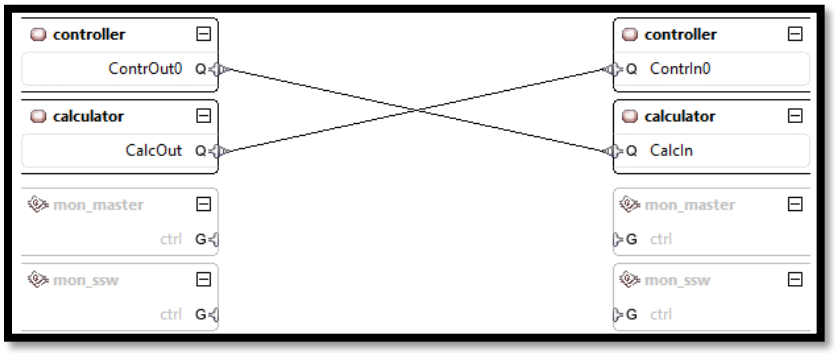

Figure 8: Queuing Port Example

These mechanisms do **not** impact partitions not participating in the communication. They do **not** impact the partner partition outside of the send/receive routines either.

Demonstrating CAST- 32A & AMC 20-193 Compliance Using the LDRA tool suite

The correct configuration and implementation of a PikeOS-based system will help to fulfil the standards' objectives, but evidence of that correctness is required for the system to be compliant.

Adherence to Requirements and Objectives

The TBmanager component of the LDRA tool suite is a requirement traceability tool. It provides continuous, live traceability to show that all project requirements have been fulfilled, and all standards objectives met – including DO-178C, CAST-32A, and/or AMC 20-193. Evidential artefacts are linked with the project requirements and the standards' objectives to provide dynamically updated traceability, addressing a significant project management pain point.

These artefacts include design documents generated externally to the LDRA tool suite, and validation and verification reports reflecting such as code coverage and data and control coupling coverage reports. TBmanager provides an effective summary showing which activities have been complete and what still need to be done and includes impact analysis capabilities to show the likely consequences of changed requirements or failed tests.

The removal of the MCP_Resource_Usage_2 objective relating to configuration settings from AMC 20-193 reinforces the benefits of automated traceability to multiple standards. The issues it addresses remain but are now covered by the document AMC 20-152A (COTS-8) instead.

Objectives	Artifacts	Assets	Last Verifi...	Objective ...	Objective St...	Objective S...	Placeholders
MCP_Accomplishment_Summary_1	0	1	Thu Sep 30 ...	Accomplish...	CAST-32A	Partial	100.00%
MCP_Error_Handling_1	0	1	Thu Sep 30 ...	Failures	CAST-32A	Partial	100.00%
MCP_Planning_1	0	0	Thu Sep 30 ...	Software Pl...	CAST-32A	Partial	0.00%
MCP_Planning_1:1	0	1	Thu Sep 30 ...	MCP Proces...	CAST-32A	Partial	100.00%
MCP_Planning_1:2	0	1	Thu Sep 30 ...	MCP Active...	CAST-32A	Partial	100.00%
MCP_Planning_1:3	0	1	Thu Sep 30 ...	MCP softwa...	CAST-32A	Partial	100.00%
MCP_Planning_1:4	0	1	Thu Sep 30 ...	MCP Dyna...	CAST-32A	Partial	100.00%
MCP_Planning_1:5	0	1	Thu Sep 30 ...	IMA platform	CAST-32A	Partial	100.00%
MCP_Planning_1:6	0	1	Thu Sep 30 ...	Robust Res...	CAST-32A	Partial	100.00%
MCP_Planning_1:7	0	1	Thu Sep 30 ...	Methods an...	CAST-32A	Partial	100.00%
MCP_Planning_2	0	0	Thu Sep 30 ...	Software / ...	CAST-32A	Partial	0.00%
MCP_Planning_2:1	0	1	Thu Sep 30 ...	MCP shared...	CAST-32A	Partial	100.00%
MCP_Planning_2:2	0	1	Thu Sep 30 ...	Hardware ...	CAST-32A	Partial	100.00%
MCP_Resource_Usage_1	0	1	Thu Sep 30 ...	MCP config...	CAST-32A	Partial	100.00%
MCP_Resource_Usage_2	0	1	Thu Sep 30 ...	Means of M...	CAST-32A	Partial	100.00%
MCP_Resource_Usage_3	0	1	Thu Sep 30 ...	Interferenc...	CAST-32A	Partial	100.00%
MCP_Resource_Usage_4	0	0	None	Available R...	CAST-32A	Unfulfilled	0.00%
MCP_Software_1	0	1	Thu Sep 30 ...	Applicable ...	CAST-32A	Partial	100.00%
MCP_Software_2	0	1	None	Data and C...	CAST-32A	Partial	16.67%

Figure 9: The TBmanager component of the LDRA tool suite

Data and Control Flow Coupling (DDFC)

Data coupling is defined in DO-178C as "*The dependence of a software component on data not exclusively under the control of that software component*", and control coupling as "*The manner or degree by which one software component influences the execution of another software component*."

The MCP_Software_2 objective in both CAST-32A & AMC 20-193 extends those principles to consider data and control coupling across cores. This includes specific examination of shared data – such as shared memory – and investigating when it comes into scope, is accessed, is modified, and goes out of scope. CAST-32A extends DO-178C's view to look at this dynamically – measure when it is executed – to determine whether all data and control couples between system elements are tested.

Data and Control Flow Coupling (DDFC) was introduced in DO-178B but DO-178C expanded on the requirement for its measurement. It is now best examined in the context of code coverage. LDRA's DDFC report provides a comprehensive view of what the data and control couples in the system are and whether they are executed or not. For CAST-32A, users can filter to focus on elements that are shared across cores.

The notes associated with MCP_Software_2 have changed a little to emphasise that interference may occur between tasks of a single component when tasks execute on different cores. The WCET analysis capabilities of the LDRA tool suite are invaluable in these circumstances to demonstrate that any interference is adequately mitigated.

Evaluating Worst-Case Execution Time (WCET)

The objectives MCP_Resource_Usage_4 and MCP_Software_1 raise the issue of Worst-Case Execution Time, or WCET. Even in the case of single core processors, the calculation of WCET based on first principles is not a trivial exercise. Several methods exist, including end-to-end measurements of execution times, and manual static analysis techniques such as counting and summing assembler instructions for each function, loop etc.

Although static analysis tools do exist for the purpose, the calculation of a definitive value of WCET by mathematical analysis is not soluble in the general case. According to Reinhard Wilhelm et al., any such approach will therefore require approximations to be applied which have to be correct, but not necessarily complete. The result is that such tools will necessarily err "on the safe side" which is better than nothing, but in an environment where precision is everything, it cannot be ideal – even without the additional vagaries introduced by MCPs.

Robust Partitioning

MCP_Software_1 highlights the benefits of robust partitioning such as that provided by PikeOS, but also describes the measures that are to be taken where there is no such facility.

PikeOS provides mechanisms to remove interference risk in terms of resources and time, allowing WCET to be determined separately for each core. In this case, the dynamic WCET facilities of the LDRA tool suite provide evidence that the features provided by the RTOS are correctly configured to the needs of the application and are therefore mitigating the challenges associated with multicore processors.

If there is no such robust partitioning, WCET measurement is even more vital to show that interference is properly mitigated.

Static Analysis

Static analysis does have a part to play in a pragmatic approach to WCET analysis. Metrics calculated using the TBvision component of the LDRA tool suite allow for the identification of the sections of the code most demanding of processing time. The time taken for the execution of those sections is best assessed dynamically using the application target environment.

Halsteads (*Cashregister.c*)

File	Total Operators	Total Operands	Unique Operators	Unique Operands	Vocabulary	Length	Volume
Total for Cashregister.c	149	230	16	56	72	379	2338

Figure 10: Static analysis can be used to identify the most demanding algorithms in the code base.

Dynamic WCET Measurement

The TBrun component of the LDRA tool suite allows users to measure timing of specific functions dynamically. Consider a typical application with various tasks that run at different

rates. The key objective is to ensure that these tasks all have time to finish in their allotted times and can't interfere with each other.

The LDRA tool suite can measure WCET dynamically, measuring the performance of each complete-cycle algorithm in turn on a running system. It includes the capability of loading one or more cores, forcing hardware interference to play a part. Figure 11 shows examples of how the statistics are represented in the LDRA tool suite. The graphs show the variation in performance of the same system under differing loads, with the extended tail of the stress loaded system shown on the right.

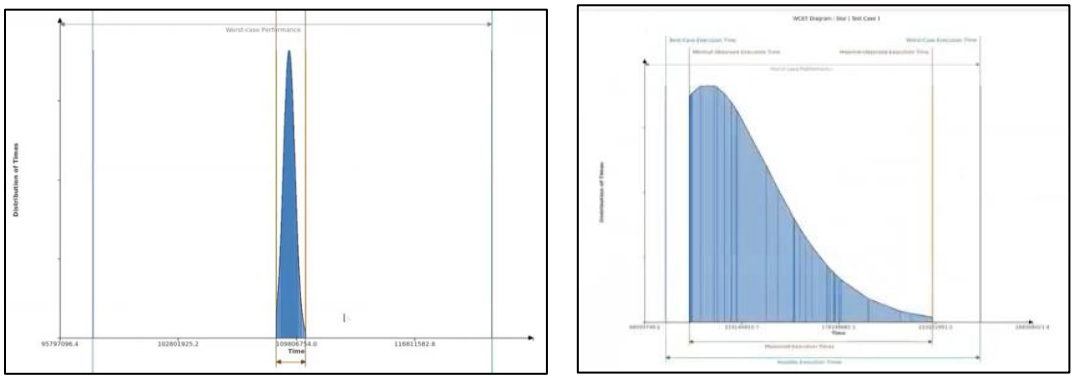

Figure 11: Measuring WCET using the LDRA tool suite

Summary

Multicore processors (MCPs) have become ubiquitous. However, they present significant challenges for the developers of certifiable, safety-critical applications. Unless interference patterns on MCP shared resources are bound and controlled, the certification of safety-critical software is impossible. But without the effective management of CPU utilization, much of an MCP's increased computing power is wasted.

The operating system selection and use of its capabilities to address the CAST-32A / AMC 20-193 objectives can have a large impact not only on the system performance but can also greatly influence the degree of engineering effort it takes to certify a safety-critical system.

For a system to be compliant, evidence of its compliance is also required. The use of multicore processors and the demands of CAST-32A introduce more objectives and requirements to the already demanding DO-178C environment. Automating traceability to evidence of compliance with the standards' objectives and the project's requirements addresses a considerable project management pain point.

Providing evidence of compliance with both DO-178C and the additional demands of CAST-32A / AMC 20-193 is another headache. An integrated set of tools capable of combining the most appropriate static and dynamic techniques for verification and validation activities - particularly including the verification of Worst-Case Execution Times (WCET) – is essential.

A combination of SYSGO's PikeOS DO-178C certified hypervisor technology and the comprehensive LDRA tool suite provides an ideal platform for the effective exploitation of the capabilities of MCPs in certifiable, safety-critical applications.

Company Profiles

For more than 40 years, LDRA has developed and driven the market for software that automates code analysis and software testing for safety-, mission-, security-, and business-critical markets. Working with clients to achieve early error identification and elimination, and full compliance with industry standards, LDRA tools trace requirements through static and dynamic analysis to unit testing and verification for a wide variety of hardware and software platforms. Boasting a worldwide presence, LDRA has headquarters in the United Kingdom, United States, Germany, and India coupled with an extensive distributor network.

For more information on LDRA and the LDRA tool suite, please visit www.ldra.com.

SYSGO is the leading European manufacturer of embedded software solutions such as the real-time operating system and hypervisor PikeOS and the embedded industrial-grade Linux ELinOS. Since 1991, SYSGO has been supporting customers in the Aerospace, Automotive, Railway and IIoT industries in the development of Safety-critical applications. SYSGO was the first company worldwide to achieve the Safety requirement level SIL 4 for its multi-core capable real-time operating system and hypervisor PikeOS. PikeOS version 5.1.3 meets the Common Criteria at the EAL 5+ level for ARMv8, x86_64 and PPC and is also certified according to the strictest Safety standards such as IEC 61508, EN 50128, EN 50657 and ISO 26262, thus enabling application development according to the "Safe & Secure by Design" principle. For industrial embedded systems, SYSGO also offers ELinOS, a Linux distribution with real-time extensions for embedded systems. Furthermore, solutions such as the Railway development platform (SAFe-VX) and the Secure Automotive Connectivity Platform (SACoP) for secure data transfer in, with and between automobiles are available.

SYSGO works closely with its customers such as Samsung, Airbus, Thales, Continental, etc., throughout the entire product life cycle and supports them in the formal certification of software according to international standards for functional and IT Security. SYSGO is headquartered in Klein-Winternheim near Frankfurt, has subsidiaries in France and the Czech Republic and maintains a worldwide sales network. The company is ISO 9001:2015 and IEC/ISO 27001:2013 certified and part of the European Thales Group.

More information at www.sysgo.com