Compliance with the Future Airborne Capability Environment™ (FACE) Standard

White Paper

Written in the run up to the FACE consortium's annual Technical Interchange Meeting and Exhibition in September 2020, this white paper provides some background on the FACE architecture with a focus on Wind River and Rapita Systems' "one stop shopping" ecosystem for Units of Portability (UoP) and the tools to test, integrate, and certify systems based on the FACE Technical Standard.

RAPITA SYSTEMS A DANLAW COMPANY

FACE™ Future Airborne Capability Environment

WNDRVR

Contents 2

1. Operating System Segment

In this chapter, we summarize the FACE architecture to illustrate how the Operating System Segment (OSS) provides a foundation to the other segments.

The US Department of Defense continues to push for the use of open architecture solutions as a means to get better avionics hardware into the field more quickly and at lower cost. One source of such solutions is the Future Airborne Capability Environment (FACE™) consortium, which was founded in 2010 to develop an open architecture technical standard and business models for implementing this standard.

U.S. DEPT OF DEFENSE

The consortium is managed by the Open Group and has members from government, education, and industry sectors. Member organizations must be incorporated in the US, and individual members must be US citizens, although the annual FACE Technical Interchange Meeting typically includes an exhibition that is open to all. The publications released by the FACE Consortium are publicly available and free to download. Currently, 91 organizations and over 1,000 individuals are members of the consortium. The consortium sponsors four general working sessions a year, called the FACE F2F. Volunteer members do the detailed work of the consortium through technical and business working groups overseen by a steering committee and advisory board.

One of the most significant products of the FACE consortium has been the development of a technical standard, which is currently at revision 3.0. According to the FACE website, "The FACE Technical Standard is the open avionics standard for making military computing operations more robust, interoperable, portable and secure. The standard enables developers to create and deploy a wide catalog of applications for use across the entire spectrum of military aviation systems through a common operating environment."

FACE Consortium

The FACE standard originated from US Navy open architecture programs to improve interoperability and software portability for avionics software applications.

The goal of FACE is to drastically reduce the development and deployment cycle of new capabilities in military airborne platforms from six years to as little as six months.

1.1 FACE Architectural Segments

The concept of the FACE standard is to provide a reference architecture based on segments, which can be composed to meet final system requirements. Variations in the content of the segments, including application code, allows the system architect flexibility in designing and building the end system. FACE provides the logical interfaces between these segments to allow for portability and re-use. Figure 1, modified from a figure in V3.0 of the FACE Technical Standard (TS), shows how segments are related in the TS.

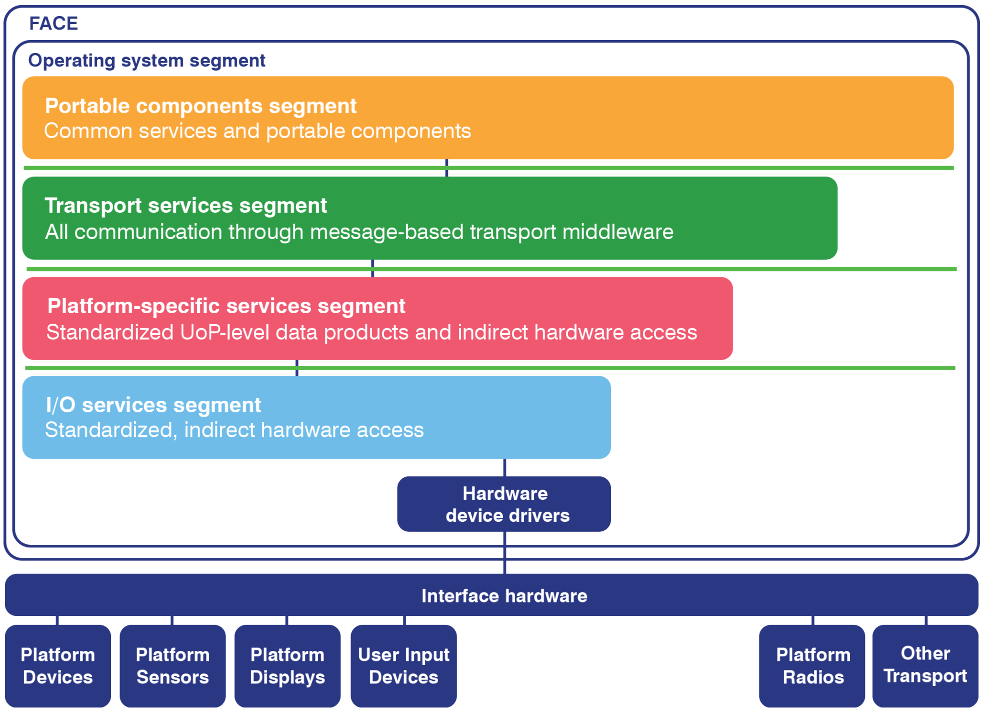

FACE
Operating system segment
Portable components segment
Common services and portable components
Transport services segment
All communication through message-based transport middleware
Platform-specific services segment
Standardized UoP-level data products and indirect hardware access
I/O services segment
Standardized, indirect hardware access
Hardware device drivers
Interface hardware
Platform Devices
Platform Sensors
Platform Displays
User Input Devices
Platform Radios
Other Transport

Figure 1 – FACE architectural segments

The standard contains five segments:

1. **Operating System Segment (OSS)** – foundational services provided by an operating system; we will cover this in more detail below. The remaining four segments depend on the OSS and thus it is shown beneath the others in Figure 1.
2. **I/O Services Segment (IOSS)** – normalizes the interface to I/O devices.
3. **Platform-Specific Services Segment (PSSS)** – provides platform services, such as data services, logging, health management and graphics (interface to GPU).
4. **Transport Services Segment (TSS)** – provides communication services.
5. **Portable Components Segment (PCS)** – offers capability or business logic.

The interfaces between these segments,as defined by the FACE TS, are key to constructing a FACE-compliant system. For the OSS, we are interested in the supported APIs, and which fit into different profiles: Security, Safety, and General Purpose.

- **The Security Profile** is most restricted and has a minimal set of Application Programming Interfaces (APIs) for high assurance applications.
- **The Safety Profile** is a superset of the security profile, with more APIs, and is intended for applications that require safety certification, this also has 2 further profiles, Base and Extended.
- **The General Purpose profile** has the most APIs and supports applications that do not necessarily need RT or deterministic response.

To maintain commonality across different FACE component suppliers, solutions are tested against these API sets and given a certificate of conformance. For example, Wind River has FACE 2.0 conformance for VxWorks 653 (Safety Base Profile), FACE 3.0 conformance for Helix Virtualization Platform (Security & Safety Base Profiles) and FACE 3.0 for Wind River Linux (General Purpose Profile). In fact, Wind River VxWorks 653 was the first product to go through FACE certification, and Wind River Linux is the only Linux-based system to achieve FACE Conformance.

The FACE standard builds on existing standards rather than creating new ones, so for the OSS it builds on the POSIX (Figure 2) and ARINC 653 standards.

Vxworks 653

Wind River's VxWorks 653 Platform was the first COTS RTOS to achieve the Future Airborne Capability Environment (FACE™) conformance certification for the FACE Operating System Segment (OSS) Safety Base Profile.

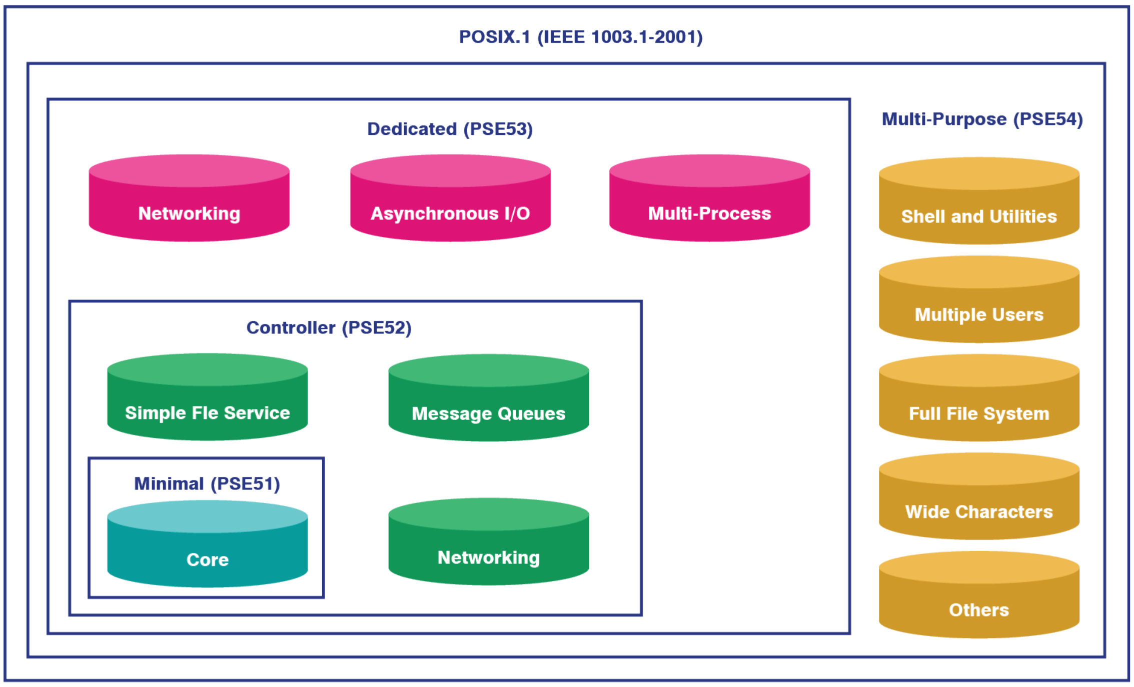


Figure 2 – POSIX standards of 2003 and 2008

1.2 Partitioning

The FACE standard also requires partitioning, depending on the profile. Partitioning is used in many types of computing systems; Wind River has a good white paper on this subject, titled "Enabling the migration to Future Aerospace & Defense Systems". Partitioning supports modularity, including support for the concept of Integrated Modular Avionics (IMA). The isolation properties provided by the partitioning of distinct applications are essential to achieve the promise of the FACE standard. Why is this? First, interoperability and smooth integration require isolation so that there are no surprises (due to unanticipated interactions) when new independent functions are added to a system. Second, certification of mixed-criticality systems is founded on the isolation of partitions.

The General Purpose Profile uses space partitioning, while the Safety and Security Profiles require both Time and Space Partitioning. These requirements stem from the ARINC 653 standard for the Safety Profiles, the FACE standard says "Time partitioning must be used when running Safety Profile-based software components that are dependent upon an ARINC 653 operational environment."

With the advent of powerful multi-core processors, the drive to use partitioning to separate critical applications is accelerating. With single core processors, the ability to host applications at different criticalities was achieved using time & space partitioning in an Integrated Modular Avionics (IMA) architecture developed to overcome issues of Space, Weight, and Power (SWaP) in commercial aircraft. This was needed to meet the increasing requirements for greater functionality within the airframes at the time, such as the Airbus A380 and Boeing 787.

However, what this basically did was "share" the CPU resource across multiple applications, and while this achieved the goals of an IMA Architecture, it also impacted the performance allocated to applications. With the latest multicore applications, however, this performance impact can be mitigated by allocation across multiple cores. These systems are breaking ground on safety certification of complex systems.

About ARINC 653

The Avionics Application Standard Software Interface (ARINC 653) is a software specification designed for space and time partitioning in safety-critical systems, which allows for hosting applications on multiple cores of the same processor using the Integrated Modular Avionics architecture.

1.3 No performance requirements

The FACE standard intentionally avoids dictating performance or quality of applications, focusing instead on developing systems with a standard interface with defined behavior. This open standard approach has a significant benefit in that it levels the playing field. All vendors must meet the same API and must then compete on performance, quality, tool support, depth of airworthiness evidence, etc. For example, multiple vendors might provide a FACE OSS that has been certified conformant to the FACE technical standard, with each providing the same expected API to interface with other elements of the system. However, the timing characteristics – such as response time, partition window jitter, etc. – may vary widely between systems produced by different vendors. Some vendors might provide a package of flight certification artifacts, while others do not, and the strength of the tool ecosystem related to the OSS may vary significantly between vendors.

2. FACE components

In this chapter, we focus on the benefits of modularity in an open architecture and discuss the features that distinguish offerings by different vendors, with a focus on the Operating System Segment (OSS) component of the FACE architecture.

For the OSS, Wind River provides a competitive advantage while maintaining modularity. Part of that advantage is a rich ecosystem of tools to validate the performance of the system and provide assurance evidence for airworthiness. Since the OSS is foundational to the architecture, it is important that the ecosystem for it is closely aligned with the operating system. For example, flight certification of multicore avionics systems require assurance that multicore interference channels have been mitigated. The Rapita Systems CAST-32A Compliance Solution provides that assurance with tools and artifacts that validate and verify the system.

Commercial solution for CAST-32A Compliance

Rapita Systems CAST-32A Compliance Solution provides an end-to-end solution for supplying certification evidence to satisfy DO-178C and CAST-32A objectives.

For more information, visit rapitasystems.com/products/cast-32a

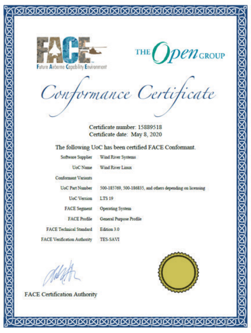
2.1 FACE modularity leads to interchangeability

The modular concepts of the FACE standard provide a reference architecture based on segments that can be integrated to meet final system requirements. Variations in the content of each segment, including application code, allows the system architect flexibility in designing and building the end system out of compatible modules. The FACE standard defines the logical interfaces between these segments to allow for modularity. This promotes the re-use of software components and enables common functionality across military systems. By using the defined API standards described in the previous chapter, this allows components to be easily moved between conformant systems developed by different vendors.

In order to ensure the segments can be used in a modular fashion, it is vital that components are tested against the applicable FACE Standard. Conformance is verified by checking that a component conforms to the FACE Technical standard as a “Unit of Conformance” or “UoC”. (The term “Unit of Portability” is sometimes used instead of “Unit of Conformance” to highlight the portable aspects of these components.)

This is done by running a FACE Conformance Test Suite, achieving certification of conformance through reviews by a FACE Verification Authority and a FACE Certification Authority, and registering the result on the FACE Registry with the FACE Library Administrator.

If a system architect starts designing a system and needs to use an OS with a specific profile, then they can look at the FACE Registry and choose one of the UoCs that have been certified against that profile. There are multiple revisions of the FACE Technical Standard, so a Version Number is also specified for each product in the registry. Figure 3 shows certificates for UoC in the General Purpose Profile and Safety Profile which is compliant with the FACE Technical Standard 3.0.



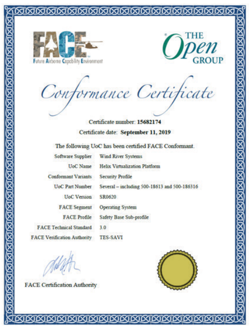
Conformance Certificate

Certificate number: 15889518
Certificate date: May 8, 2020

The following UoC has been certified FACE Conformant.

Software Supplier	Wind River Systems
UoC Name	Wind River Linux
Conformant Variants	
UoC Part Number	500-185769, 500-186835, and others depending on licensing
UoC Version	LTS 19
FACE Segment	Operating System
FACE Profile	General Purpose Profile
FACE Technical Standard	Edition 3.0
FACE Verification Authority	TES-SAVI

[illegible]
FACE Certification Authority



Conformance Certificate

Certificate number: **15682174**
Certificate date: **September 11, 2019**

The following UoC has been certified FACE Conformant.

Software Supplier	Wind River Systems
UoC Name	Helix Virtualization Platform
Conformant Variants	Security Profile
UoC Part Number	Several – including 500-18613 and 500-186316
UoC Version	SR0620
FACE Segment	Operating System
FACE Profile	Safety Base Sub-profile
FACE Technical Standard	3.0
FACE Verification Authority	TES-SAVI

[illegible]
FACE Certification Authority

Figure 3 – Certification of conformance for UOCs tested againts FACE profiles

A System Architect will typically choose several UoCs that will integrate into the final software system. Of note here is that several UoCs could coexist within the same processor address space, and so integration considerations and coordination of these UoCs is vital. This is particularly true of performance and interference challenges at integration time. FACE conformance testing only checks the UoC against an OSS profile and within a single segment, and so will not check all software behavior or performance. It's also worth noting that if you have several UoCs, then you would need UoC to UoC communications, which requires use of the FACE Transport Services Interface. Inter-UoC communication ishown in Figure 4 from the FACE Technical Standard 3.0.

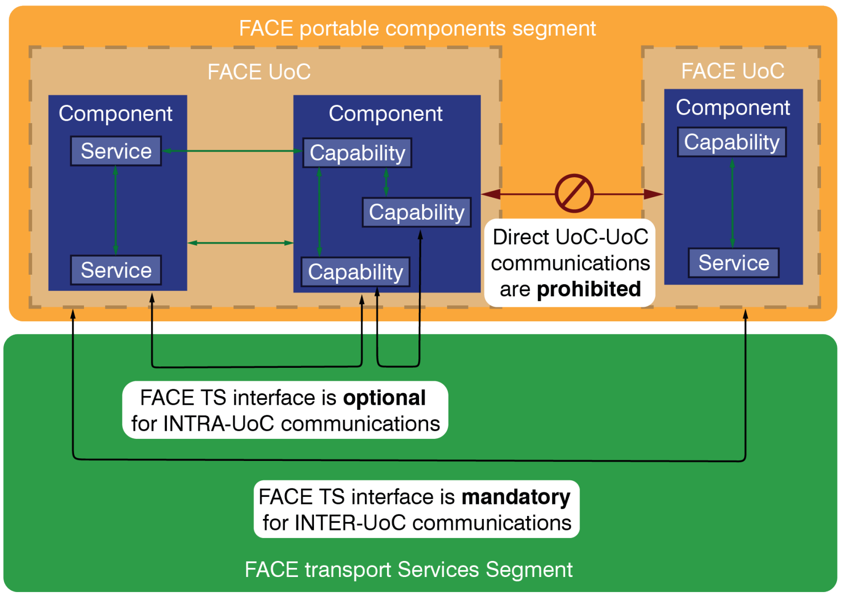

FACE portable components segment
FACE UoC
Component
Service
Service
Component
Capability
Capability
Capability
FACE UoC
Component
Capability
Service
Direct UoC-UoC communications are **prohibited**
FACE TS interface is **optional** for INTRA-UoC communications
FACE TS interface is **mandatory** for INTER-UoC communications
FACE transport Services Segment

Figure 4 – Example PCS inter-UoC and Intra-UoC communications

FACE Conformance Test Suite

The FACE Conformance Test Suite exists to help verify a software components compliance with the standard.

Conformance is verified through automated testing and inspection of specific software artifacts.

2.2 Benefits of interchangeability

The FACE approach aligns well with the concept of Open Systems Architectures (OSAs). The OSA approach enables several business drivers:

- Better buying power
- More affordability
- Faster time to deploy
- Rapid capability injection
- Software reuse

One interesting aspect of the FACE Consortia was the early decision to develop not only a Technical Standard, but also a Business Standard that provides guidance in the value proposition and business case for the FACE approach. This is currently at Version 2.0 and is available on the FACE Documents Web Site.

Being certified conformant to a UoC in the FACE standard provides confidence that system integrators have choices – they can drop a particular UoC from any vendor into their system and it should work "out of the box" with other components. This fulfils several of the business drivers (reuse of components, rapid capability injection, and more affordability) as the cost of development of the UoC is spread across several programs. An added benefit is that the system integration cost should also be lower due to the well-defined layers between UoCs within the standard.

Competition is generated between vendors to provide their value add and capability expertise while still conforming to the FACE standard. This in turn gives government procurement agents better buying power when looking for alternative or complementary solutions.

At the same time, it means that the government is no longer stuck with one vendor, breaks out of vendor lock-in, and is able to choose the best solution rather than being forced into using the same vendor.

For vendors producing products, this also provides benefits of standardization across projects, which reduces development costs and risks of schedule slips that could impact multiple programs. The reuse of components means better return on investment and allows vendors to focus on their core strengths and innovation.

However, having functional components that adhere to the standard does not provide everything you need. Each component will have different performance, quality, certification artifacts, and tools. Each will also have different business models and lifecycle support solutions. These must be taken into account when making final selections.

FACE Business Guide

This guide serves as a reference for executives, military executive officers, and senior leadership from both the Government and industry to understand the value proposition of the FACE approach.

2.3 Interchangeability is the minimum

The benefits of interchangeable parts are clear in everyday society. Metric wrenches all fit a metric socket. Street legal automobiles all fit on public roads. The benefits of such standardization are clear for wrenches and cars. The benefits for software are also clear. However, interchangeable parts set the minimum expectation. All wrenches should fit their sockets and all automobiles should fit the road. All wrenches, however, vary in strength and durability and all cars vary in top speed and fuel efficiency. So too, interchangeability for standardized software is the minimum. An OSS that is certified to be conformant to the FACE Technical standard has merely been shown to "fit the road". Specifically, the Application Programming Interface (API) specified by the standard confirms that the software components that make up the system will fit together and communicate with each other correctly. This compatibility in itself, however, does not indicate anything about the quality of the component, such as its speed or efficiency.

Thus, interchangeability through conformance to standards such as the FACE Technical Standard is the ticket for entry to the FACE market, but this is the minimum. Vendors that meet this starting criterion for a specific FACE UoC compete on a level playing field. For example, customers seeking a FACE OSS will likely make decisions on purchasing based on the following:

- **Performance**: Components produced by different vendors will vary in their performance metrics such as end-to-end latency, response time, determinism, reliability, etc.

- **Certification**: Conformance to the FACE technical standard may provide some testing and artifacts that may be useful for demonstrating airworthiness, but this is only a start. The packages offered by vendors in support of flight certification will vary. On this note, the FACE EA-25 subcommittee outlines advice on which FACE activities and artifacts may contribute to flight certification evidence.

- **Ecosystem**: The tools associated with each vendor will vary widely. Especially for the OSS, a rich ecosystem of tools is important to the successful development and integration of the system. The standardization provided by the FACE technical standard can sometimes mean that tools work across any vendor offering, but this is not always the case, as tools may need to connect "under the hood" as well.

Example: Tools

Example tools include the Integrated Development Environment (IDE), debugging tools, testing tools to support requirements-based and functional testing, structural coverage analysis tools, timing analysis tools, requirements traceability tools, etc.

2.4 Assuring partitioning

2.4.1 The FACE standard requires partitioning

The FACE standard requires that compliant systems in the Safety and Security Profiles include support for time and space partitioning. Partitioning is an isolation technique used in many types of computing systems. This technique supports modularity, for example through Integrated Modular Avionics (IMA). The separation properties provided by the partitioning of distinct applications are essential to achieve the promise of the FACE standard.

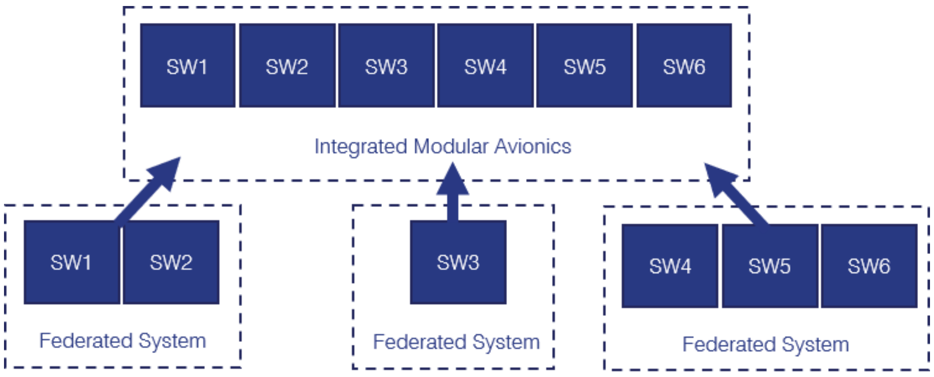


Figure 5 – Integrated modular avionics

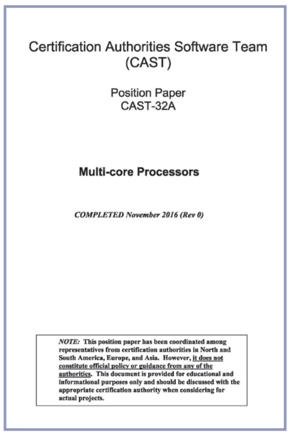
With the advent of powerful multi-core processors, the drive to use partitioning to separate critical applications is accelerating. With single core processors, the ability to host applications at different criticalities was achieved using time and space partitioning in an Integrated Modular Avionics (IMA) architecture developed to overcome issues of Space, Weight, and Power (SWaP) in commercial aircraft. This was needed to meet the increasing requirements for greater capability within common airframes at the time such as the Airbus A380 and Boeing 787. However, on a unicore system, the CPU resource was shared (time-sliced) across multiple applications, and while this achieved the goals of an IMA Architecture, it also impacted the performance allocated to applications. With the latest multicore applications, this performance impact can be mitigated by allocation across multiple cores. At the same time, the use of multicore processors introduces new challenges to assuring airworthiness.

2.4.2 The FACE standard does not assure partitioning

Although the FACE technical standard requires partitioning in several of its defined profiles, the standard only requires an interface to a partitioned operating environment (ARINC 653). It does not require assurance that the environment correctly implements the isolation mechanisms that provide the primary benefit of partitioning. This makes sense as the FACE technical standard is intended to foster interchangeability and modularity: it is not a substitute for flight certification, but only a starting point. The FACE EA-25 Airworthiness subcommittee is currently working on a white paper to “ensure that FACE conformant software, associated artifacts, and the process to produce them … contribute towards evidence of Air Worthiness where possible, explicitly citing common requirements, processing and guidance.”

The system integrator would be wise to consider the significant hurdles to prove airworthiness from the earliest stages of design, especially for features central to the overall architecture such as partitioning. When implemented correctly, partitioning simplifies integration and certification because each partition can be considered one at a time. However, this approach requires that the partitioning itself is proven to a very high level of rigor. Rigorous isolation of partitions is not something that can be tacked on at the end, but must be incorporated as a fundamental design feature from the beginning. Certification of software for Civilian flight in the US and Europe is guided by the RTCA DO-178C/ED-12C standard. The DO-297 and ARINC 653 standards provide guidance on partitioning environments to isolate independent software functions. Military airworthiness authorities use similar approaches to evaluate software.

The challenge to assure the isolation of partitions is even greater for computing platforms that use multicore processors. On a unicore processor, only one partition can run at a time, providing some natural isolation. On a multicore processor, partitions can run simultaneously on different cores, creating multiple channels of potential interference between functions. Multicore interference can degrade the isolation between partitions and thus must be mitigated. While the use of multi-core systems for critical software is a relatively new technology, the US Federal Aviation Administration (FAA) currently provides guidance for assuring multicore systems in a position paper, CAST-32A.



Certification Authorities Software Team (CAST)

Position Paper
CAST-32A

Multi-core Processors

COMPLETED November 2016 (Rev 0)

NOTE: **This position paper has been coordinated among representatives from certification authorities in North and South America, Europe, and Asia. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects.**

Figure 6 – CAST-32A position paper

Planning for Isolation

“Rigorous isolation of partitions is not something that can be tacked on at the end: it must be incorporated as a fundamental design feature from the beginning.”

3. Assured partitioning for FACE systems

Avionics system designers and integrators designing to the FACE standard under the Safety, Safety-Extended or Security Profiles must include an ARINC 653 partitioned operating environment in their architecture. System integrators need guidance on how to be successful with implementing partitioning while maintaining performance and achieving flight certification.

3.1 Benefits of partitioning

In the commercial aerospace world, since the first days of Integrated Modular Avionics, it was recognized that partitioned systems could provide benefits in terms of both mixed-criticality systems and recertification of future systems (for updates and changes to applications).

The ARINC 653 standard was developed to define how to construct partitioned systems, which support hosting multiple applications at different design assurance levels on the same computing platform.

Example: Partitioning applications on multicore processors

The flight management system application and the flight navigation application might have been hosted on separate Line Replaceable Unit (LRU) computing hardware in the past, the modern processors are fast enough to host both applications, provided partitioning ensures they do not interference with one another's functionality. Partitioning enforces modularity and provides portability through a standard API, as well as contains faults, thus easing integration and certification.

Wind River, together with Rapita Systems, can help you build a FACE system with the performance and determinism you need, along with the assurance artifacts you need to achieve flight authorization cost-effectively.

For more information, visit rapitasystems.com/cast-32a

3.2 Partitioning required but not assured

In Supplement 5 of ARINC 653 Part 1, the scope of the standard is well defined: "ARINC 653 is intended for use in a partitioned environment. To assure a high degree of portability, aspects of the partitioned environment are discussed and assumed. However, this specification does not define the complete system, hardware, and software requirements for partitioning, nor does it provide guidance on proper implementation of partitioning, and in particular, robust partitioning. It must not be construed that compliance to ARINC 653 assures robust partitioning."

FACE requires compliance with the ARINC 653 Part 1 standard. Thus, conformance to the FACE technical standard implies that partitioning is provided, but robust partitioning is not assured. Further work beyond simply meeting the FACE and ARINC 653 standards is necessary to provide safety assurance evidence toward flight certification of such a system.

3.2.1 Assurance of partitioning

Because ARINC 653 defines the interfaces and functionality of partitioning but not the assurance, this lack of guidance for safety certification of Integrated Modular Avionics systems led to the development and creation of RTCA DO-297 (EUROCAE ED-124) "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations", which sets out guidance on safety certification of IMA systems. This document introduced the concepts of roles and responsibilities such that you could allocate resources where needed and have clear guidelines on who does what to ensure compliance with the safety standards.

The standard states that "The IMA Platform should be capable of providing robust partitioning and other protection means that allow multiple applications to share a platform and its resources." Further, it introduces the concept of Robust Partitioning which "will ensure that any hosted application or function has no unintended effect on other hosted applications or functions." The standard includes a complete section (3.5) on robust partitioning and how to ensure that it meets the requirements of an IMA system.

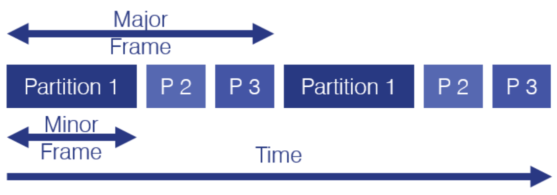


Figure 7 – Software partitioning mechanism

Time partitioning allows a unicore system to support multiple partitions, each hosting an independent application, as shown in Figure 7. A fixed schedule of partitions is repeated each major time frame (e.g., every 50 ms). Within the major time frame, each partition is scheduled within a minor frame that is a fixed offset from the start of the major frame. Time partitioning, also known as time slicing or multiplexing, ensures that only one partition is using the computing platform at a time.

Robust Partitioning

Robust partitioning assures that any hosted application or function has no unintended effect on other hosted applications or functions.

If partitioning is sufficiently robust, then each partition runs independently, without knowledge of the other partitions or interference from them.

Proving the correctness of time partitioning, even on a unicore is challenging. Although only one partition can run at a time with only a single core, partitions could still interfere with each other directly by causing unconstrained partition jitter, i.e. variation in the time for each partition scheduled time slot without a deterministic bound. The end of each minor time frame is enforced by the partitioned Operating System (OS), usually using an interrupt-based system timer that invokes the OS at the end of the minor time frame, allowing it to perform a partition switch. During this partition switch, the OS saves the state of the partition that is finishing its minor frame, then determines which partition should be run in the next minor frame, sets a new timer, and then begins execution of the new partition. If partitioning is sufficiently robust, then each partition runs independently, without knowledge of the other partitions or interference from them.

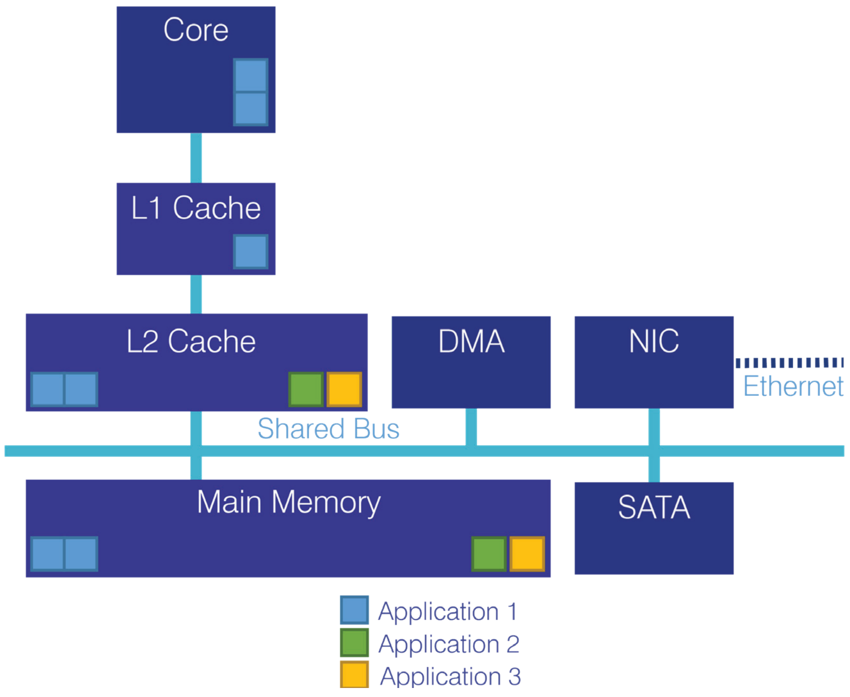
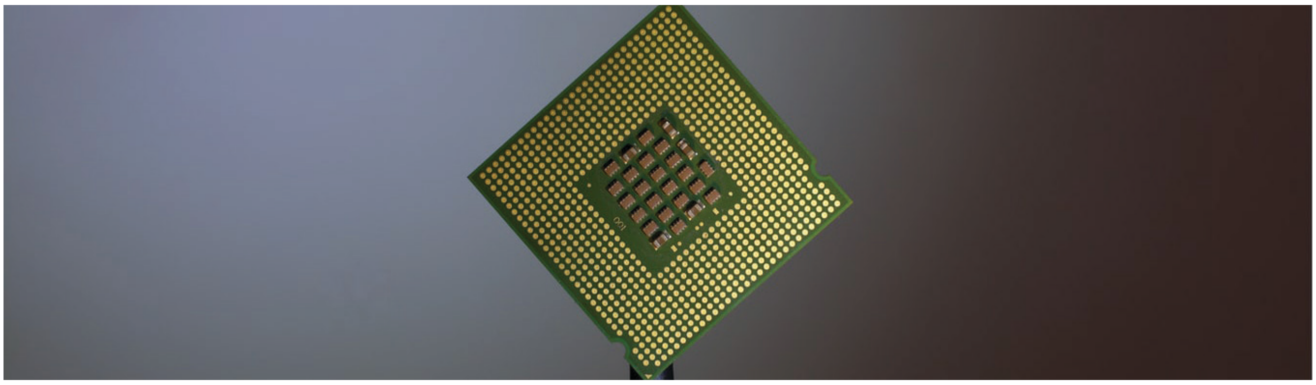


Figure 8 – Unicore computer architecture hosting three applications

Partitions on different cores can also interfere with each other indirectly. For example, one partition could start an operation using a bus master other than the CPU, whose activity extends past the end of the partition's scheduled time slot and thus overlaps and possibly contends with another partition's activity. For example, in Figure 8, each partition running on the unicore during its minor time frame likely accesses main memory. While the other partitions might have data in some parts of that memory, they do not interfere with each other as they are not running simultaneously. However, if Partition 1 starts a DMA access between locations in main memory, this access has the potential to continue running past the end of the minor time frame for Partition 1, potentially interfering with memory access by Partition 2. This interference allows Partition 1 to impact the performance of Partition 2, thus the system would not have robust partitioning. A well-designed partitioned OS will thus need to curtail all access by a partition to shared resources in the system outside of the partition's assinged minor time frame.

4. Assured multicore partitioning for FACE systems



In this chapter, we will discuss the benefits of using multicore processors in the avionics industry and how partitioning can be applied to complex multicore systems.

Avionics system designers and integrators designing to the FACE standard are adopting the Multicore processors (MCPs) to meet future performance demands. MCPs provides the avionics industry with platform that can provide greater performance in a lower footprint, translating to systems with lower Size, Weight, and Power (SWaP).

Regardless of these benefits, OEMs in the avionics industry are pressured to adopt MCP technology when making upgrades to replace legacy single-core designs because nearly every new processor on the market today is based on multicore technology.

The benefits of partitioning still hold even when implemented on an MCP, including portability, modularity, reduced integration effort, and reduced certification effort (due to incremental and compartmentalized assurance). However, MCPs also introduce additional complexity in implementation and certification, prompting a need for guidance that helps system integrators develop and certify systems using MCP technology.

The critical embedded industry is moving towards the use of multicore rather than singlecore processors due to improved performance and diminishing availability of their singlecore counterparts.

The increased performance of multicore systems per unit area is sought after in the embedded aerospace industry due to the physical space constraints and increasing complexity of such systems.

4.1 Guidance for assurance of multicore systems

DO-178C, DO-297, and ARINC 653 were all written in the context of single core processors, so the introduction and use of MCPs adds further complexity. The FAA CAST-32A position paper addresses MCP assurance and a recently proposed FAA advisory circular intends to formalize that guidance, harmonizing with the European Union Safety Agency (EASA). The new guidance will be referred to as AC 20-193 in the US under the FAA and the same document will be referred to as AMC 20-193 in Europe under EASA.

Federal Aviation Administration

EASA European Aviation Safety Agency

CAST-32A extends the partitioning concept for multicore processors: "Robust Time Partitioning (on an MCP) is achieved when, as a result of mitigating the time interference between partitions hosted on different cores, no software partition consumes more than its allocation of execution time on the core(s) on which it executes, irrespective of whether partitions are executing on none of the other active cores or all of the other active cores."

Although each US military service has its own airworthiness authority, evidence of airworthiness according to FAA guidelines is often accepted as part of the certification effort. Specifically, CAST-32A is one of the FAA guidance documents that some military programs are currently using when considering the adoption of MCP systems. In addition, starting in 2019, augmentation has been underway on Section 15 "Computer Systems and Software" of MIL-HDBK-516C Airworthiness Certification Criteria. Of the 42 criteria listed in section 15 of the document, 20 have been identified as needing augmentation to account for the use of MCPs. An update to MIL-HDBK-516C is expected in the coming year.

The certification applicant for a FACE conformant system running on a multicore processor will need to meet either civilian flight certification guidelines (such as DO-178C, DO-297, and AC 20-193) and/or military airworthiness guidance (such as the augmented MCP requirements for MIL-HDBK-516C). Conformance to the FACE technical standard in itself does not ensure that these standards are met, though, at the time of writing, the FACE EA-25 subcommittee is currently working on a white paper to provide more detailed advice on which FACE activities and artifacts may contribute to flight certification evidence for standards such as DO-178C and MIL-H.

A(M)C 20-193

A(M)C 20-193 will be a document that is a joint effort by EASA and the FAA to provide guidance on certification of multicore systems. A(M)C 20-193 will build on industry advancements that are aiding the certification process for multicore processors and will recommend best practices to consider when dealing with MCPs.

For more information, visit rapitasystems.com/amc-20-193

4.2 Assuring partitioned multicore systems

When multicore processors first started appearing in avionics, early adopters avoided some of the certification issues by simply deactivating all but one of the cores. This worked, because in many cases that single core offered better performance than older unicore processors. However, as the years have gone by, the individual cores have not improved much further in performance, meaning that significant gains in functionality through higher processor computational throughput can only be achieved by using more than one of the cores in an MCP.

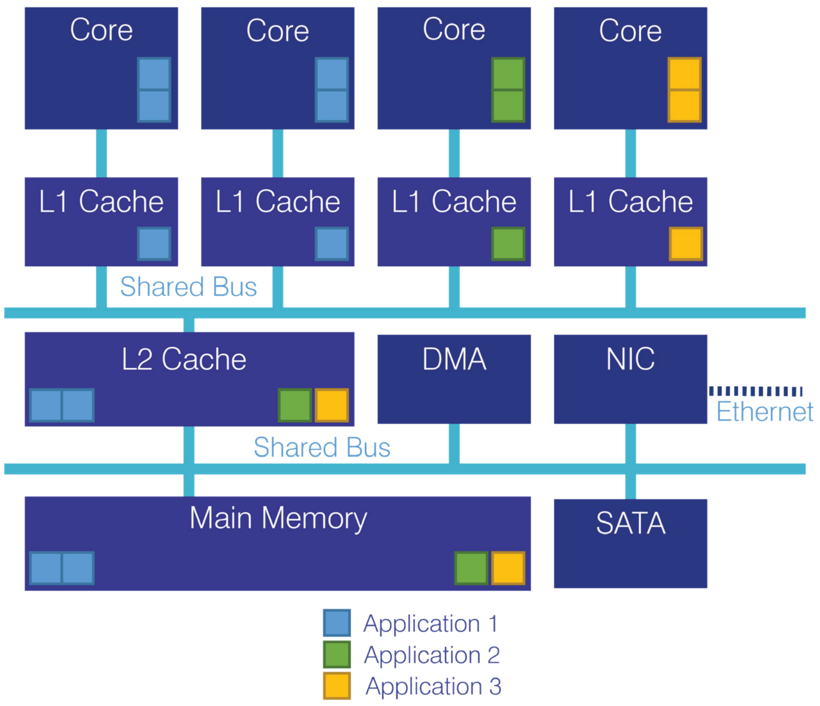


Figure 9 – Multicore computer architecture hosting three applications

Proving the correctness of time partitioning on even a unicore system is challenging, as we discussed on page 15. In an MCP system, multiple applications can run simultaneously, each on its own core. Some resources are private and exclusive to a single core, such as the L1 cache, as shown in the example four-core architecture in Figure 9. However, other resources are shared, such as the L2 cache, main memory, and I/O devices. The applications on different cores can thus contend for access to these shared resources, potentially impacting each other’s performance, which breaks down the isolation between partitions.

One simplified approach to achieving time partitioning is by only scheduling a single partition at any given time but allowing that partition to use more than one core. This approach is limited, as only multi-threaded application software can take advantage of multiple cores, while single-threaded cores leave the other cores idle during their time window of the schedule, as shown in Figure 10.

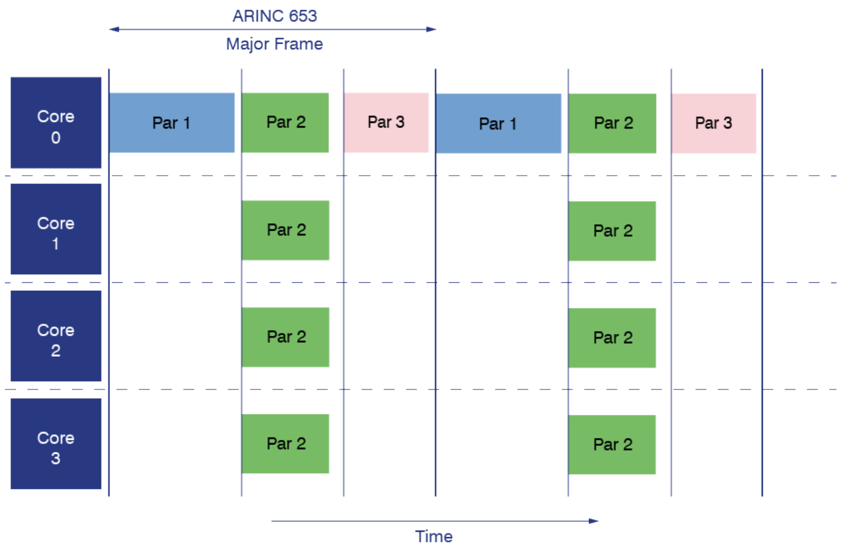


Figure 10 – Simplified multicore ARINC653 schedule restricted to one partition at a time

Only recently have early adopters of MCPs attempted to use all cores with mixed-criticality software by following the guidance in DO-297 and A(M)C 20-193. Figure 11 illustrates the complex nature of this approach, where multiple independent applications can run within the same partition window (each on its own core), even if the applications are certified at different design assurance levels. In this case, independent partitions with different criticality, i.e. different software levels, are able to run simultaneously on different cores within the same minor time frame of the schedule. An example of this complex approach is given by the work Collins Aerospace has done with Wind River as described in their joint white paper.

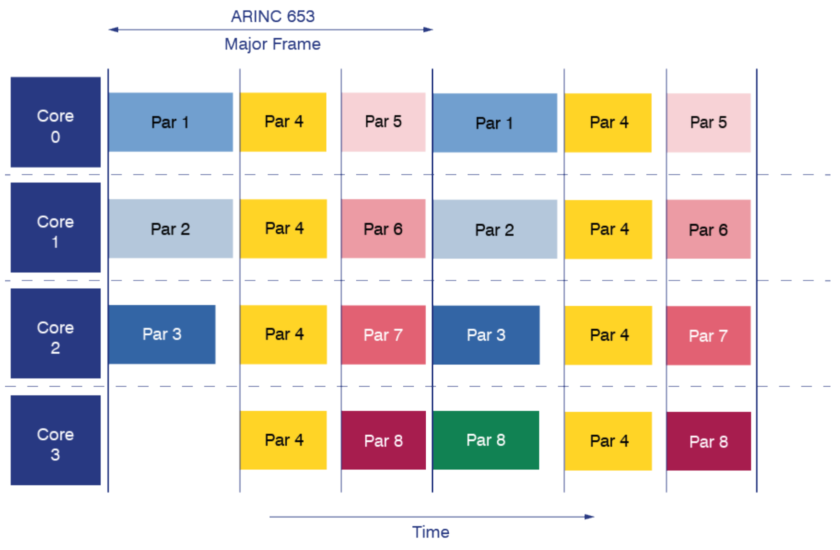
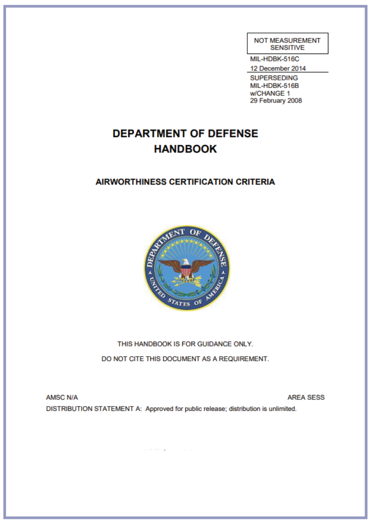


Figure 11 - Complex multicore ARINC653 schedule

Proving the correctness of time partitioning on a mixed-criticality multicore system is quite challenging since partitions can run simultaneously on different cores. In multicore systems, independent partitions running on distinct cores can compete for shared resources. That contention can cause delays, thus increasing the software's Worst-Case Execution Time (WCET). While an application running in a partition may meet its requirements when no other cores are active, interference from functions running on other cores may increase the application's WCET to the point that it no longer meets its requirements. A variety of resources may cause such interference because the processor cores share access, including lower levels of cache memory, the main memory, I/O devices, and bus interconnects.

A(M)C 20-193 provides guidance on assuring a multicore system, listing ten objectives that must be met to demonstrate that multicore interference channels have been mitigated. The planned additions to MIL-HDBK-516C also include criteria to analyze interference channels. CAST-32A defines an interference channel as a property of the computing platform "that may cause interference between independent applications." All interference channels within the avionics system must either be eliminated or else their impact must be sufficiently reduced such that all applications meet their timing requirements even in the presence of the worst-case level of interference from other cores. Similarly, the MIL-HDBK-516C additions require that "execution rates provided by the executive/control structure … are consistently obtainable and sufficient to safely provide the required performance." That is, even in the face of multicore interference, timing requirements must be met. Successfully meeting the objectives of these standards requires both attention to mitigation of interference channels during the early life cycle stage of design and careful attention to measurement methods during the later life cycle stage of verification.


NOT MEASUREMENT SENSITIVE

MIL-HDBK-516C
12 December 2014
SUPERSEDING
MIL-HDBK-516B
w/CHANGE 1
29 February 2008

DEPARTMENT OF DEFENSE HANDBOOK

AIRWORTHINESS CERTIFICATION CRITERIA

THIS HANDBOOK IS FOR GUIDANCE ONLY.
DO NOT CITE THIS DOCUMENT AS A REQUIREMENT.

AMSC N/A AREA SESS
DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

Figure 12 – MIL-HDBK-516C document

Find out more about WCET

Worst-case execution time is the maximum length of time a task takes to execute on a specific hardware platform. WCET is a metric commonly used in reliable real-time systems which have a non-negotiable deadline for execution.

For more information, visit rapitasystems.com/wcet

4.3 Mitigating multicore interference

Mitigation of multicore interference requires that partitions be isolated from each other. Multiple separation mechanisms to achieve partitioning are available to the system designer and integrator, which may be implemented in hardware and/or software.

Some isolation mechanisms are provided by the hardware. Selecting a processor that gives each core exclusive resources can eliminate contention for those resources. For example, at least one level of cache memory is typically provided that is distinct and exclusive to each core. For CPU-intensive applications with strong locality of access, a private L1 cache is often sufficient to render the software largely insensitive to interference from other cores. When resources must be shared, the hardware may provide isolation by ensuring equitable access. Even then, one core's access may impact access by another core, breaking down the separation. Unfortunately, silicon providers may not provide better determinism in cases like this because it would impact raw performance.

Although hardware can provide some isolation mechanisms, most processors are designed to optimize the average execution time on all cores, often at the expense of the WCET on any one core. While this is a good trade-off for many commercial applications, it is a problem for safety-critical avionics as it represents a form of multicore interference. Thus, additional isolation approaches beyond those implemented through hardware are necessary.

Some isolation mechanisms are provided by the RTOS. A multicore RTOS can manage processor cores, ensuring that the usage of any one hosted application is deterministically bounded within its partition so that all applications meet their requirements – even when multicore interference is present. An example of such an RTOS is the Wind River VxWorks 653 Multi-Core edition. (Figure 13)

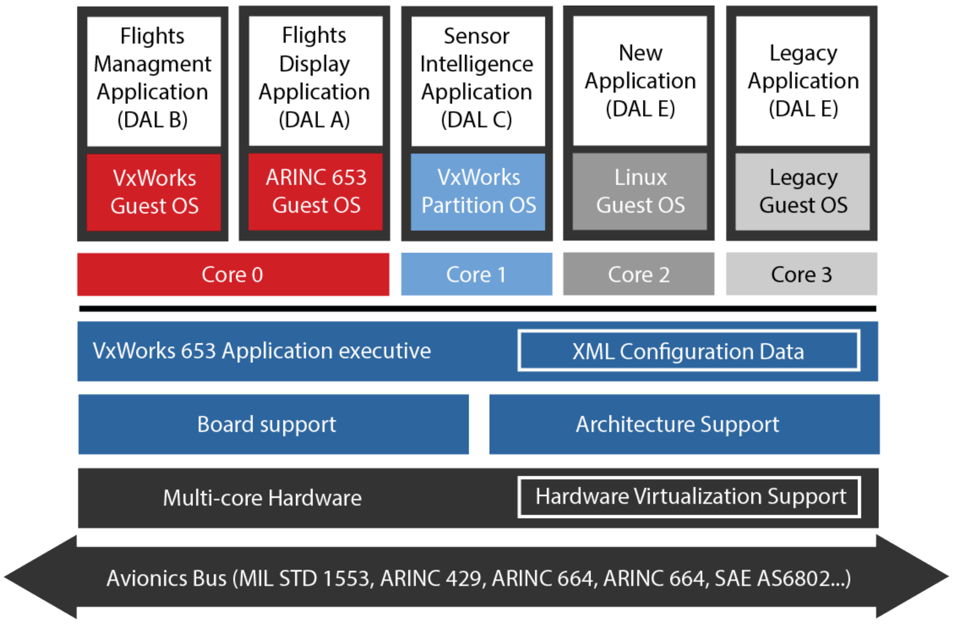

Flights Managment Application (DAL B)
VxWorks Guest OS
Flights Display Application (DAL A)
ARINC 653 Guest OS
Sensor Intelligence Application (DAL C)
VxWorks Partition OS
New Application (DAL E)
Linux Guest OS
Legacy Application (DAL E)
Legacy Guest OS
Core 0
Core 1
Core 2
Core 3
VxWorks 653 Application executive
XML Configuration Data
Board support
Architecture Support
Multi-core Hardware
Hardware Virtualization Support
Avionics Bus (MIL STD 1553, ARINC 429, ARINC 664, ARINC 664, SAE AS6802...)

Figure 13 – Wind River VxWorks 653 Multi-Core edition time scheduler

Find out more about VxWorks 653

VxWorks 653 is an integrated modular avionics platform that enables workload consideration of safety-critical applications.

For more information about Wind River and VxWorks, visit windriver.com

4.4 Verifying mitigation of multicore interference

Verification that multicore interference channels have been mitigated is an essential step in meeting A(M)C 20-193 objectives. As no one has yet flight-certified a civilian aircraft with a mixed-criticality multicore system, approaches to verification are only just appearing. Nevertheless, best practices for verification are emerging based on interference generators.

Interference generators are carefully crafted software applications with a small code footprint that create high bandwidth use of a targeted resource. For example, Rapita Systems has a library of interference generators called Rapi**Daemons** that target resources such as a shared L3 cache, DDR main memory, or DMA. These interference generators are used to create inter-core stress on the targeted resource. For example, Figure 14 shows Rapi**Daemons** running on cores 1, 2, and 3 with the application running on core 0, where the Rapi**Daemons** create interference for access to the main memory as well as any shared resource along the logical path to that resource, including interconnects and cache.

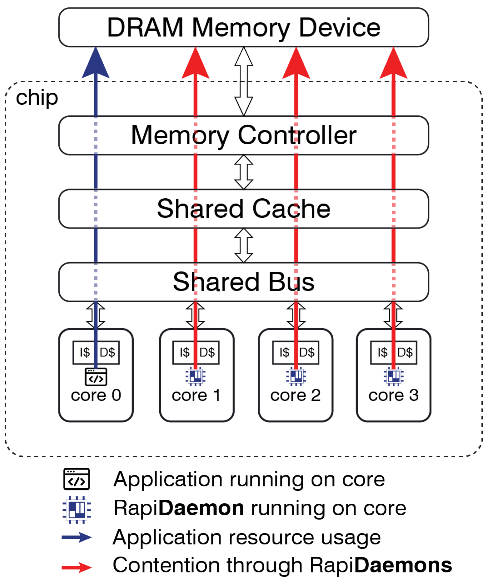

DRAM Memory Device
chip
Memory Controller
Shared Cache
Shared Bus
I$
D$
core 0
core 1
core 2
core 3
Application running on core
RapiDaemon running on core
Application resource usage
Contention through RapiDaemons

Figure 14 – Measuring multicore interference with Rapi**Daemons**

Comprehensive analysis of multicore interference requires the identification of all potential interference channels. The system architecture must be reviewed for shared resources, including potentially obscure or hidden channels where cores can contend, such as interconnects that are not equitably arbitrated or last level caches with insufficient write ports. For each interference channel identified, an interference generator must be designed and calibrated to stress that channel. For systems at the highest design assurance levels, such as DO-178C Levels A and B, this testing must be done with independence.

4.5 The Rapita approach to multicore timing analysis

The Rapita Systems approach to multicore timing analysis uses two phases within a test methodology for measuring multicore interference channels. The first phase is platform characterization, wherein the platform is defined as both the computational hardware as well as the RTOS. In this phase, the outer bounds of possible multicore interference are checked by competing Rapi**Daemons** against each other on all cores. Because Rapi**Daemons** are tuned to stress a single targeted shared resource, this provides a signature of the performance possible when cores compete continuously for that resource. This phase can be done even before application software is available.

The second test phase is software characterization. Individual software applications intended for use in flight are first measured running alone on a core while the other cores are dormant, providing a baseline measurement of the software timing behavior. Next, the increase in WCET is measured in the presence of Rapi**Daemon** adversaries running on the other cores. By comparing the WCET with and without the Rapi**Daemons** adversaries running, worst-case multicore interference can be quantified. Furthermore, the WCET with multicore interference can be compared with system requirements to determine if they are still met — thus demonstrating whether or not interference channels have been sufficiently mitigated.

Integrated Modular Avionics (IMA) designed with robust partitioning according to DO-297 allow for the incremental acceptance of assurance evidence, accumulating certification artifacts over the course of several independent verification efforts. That is, each application can be measured independently of the other applications to verify the mitigation of multicore interference channels. No partition application will produce contention higher than that produced by the Rapi**Daemons**.

Assurance evidence can be collected for each partition application by itself, although all planned applications will be integrated together in the final system. Furthermore, the timing tests for existing partitions need not be repeated when new software is added in the future when allocated to spare partition slots. This incremental verification approach enabled by robust partitioning is a key factor in reducing the number of tests that must be performed.

Even with incremental verification benefits, the complexity and number of interference channels in a typical multicore avionics system can still lead to a large number of tests that must be completed. Thus, tool automation is important to keep the schedule and cost of verification reasonably constrained. The Rapita Systems approach to multicore timing analysis automates key stages of the workflow using both Rapi**Daemons** interference generators and the Rapita Verification Suite software, as illustrated in Figure 15.

However, not everything can be automated. Human wisdom from the system and test engineers is still needed to identify interference channels in the system architecture at the start of the process and to properly interpret results at the end of the process.

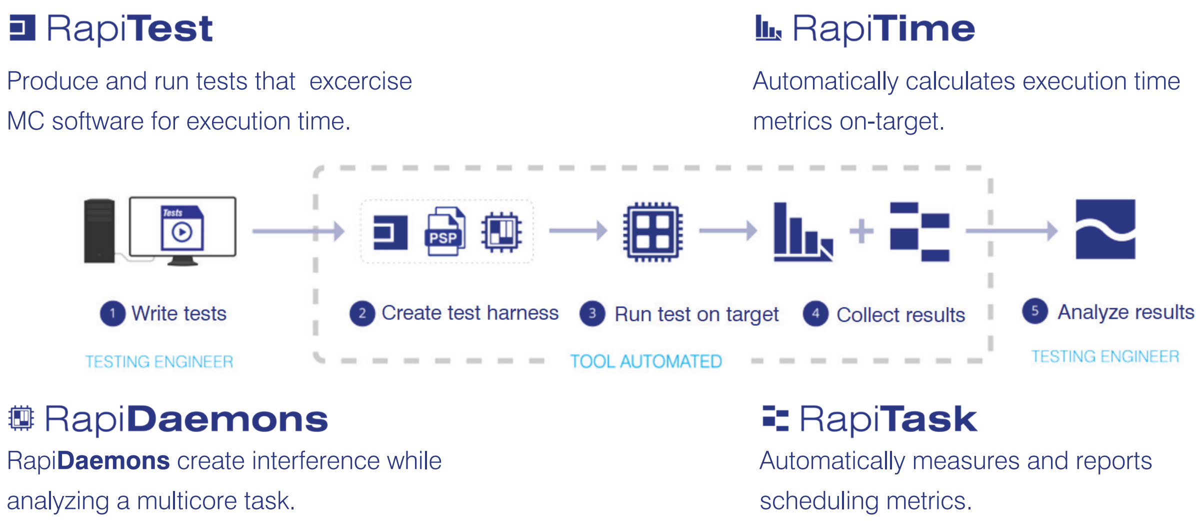

RapiTest
Produce and run tests that excercise MC software for execution time.
RapiTime
Automatically calculates execution time metrics on-target.
Tests
PSP
1 Write tests
TESTING ENGINEER
2 Create test harness
3 Run test on target
4 Collect results
TOOL AUTOMATED
5 Analyze results
TESTING ENGINEER
RapiDaemons
RapiDaemons create interference while analyzing a multicore task.
RapiTask
Automatically measures and reports scheduling metrics.

Figure 15 - Tool automation through Rapi**Daemons** and the Rapita Verification Suite

5. Leveraging conformance artifacts for airworthiness

The previous chapter focused on partitioning in multicore processors and guidance for safety certification of avionics systems based on multicore processors, including multicore aspects of partitioning.

Airworthiness is discussed in a range of documents, including:

- RTCA DO-178C "Software Considerations in Airborne Systems and Equipment Certification" – describes the process of developing software that will be flight certified
- RTCA DO-254 "Design Assurance Guidance for Airborne Electronic Hardware" – describes the process of developing complex electronic hardware that will be flight certified. Referenced by AC 20-152.
- RTCA DO-248C "Supporting Information for DO-178C and DO-278A" – provides additional details on concepts in DO-178C, including using previously developed software, MC/DC coverage, independence and partitioning.
- RTCA DO-297 "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations" – describes roles and processes for developing IMA systems and the importance of partitioning.
- ARINC 653 Part 1, Supplement 5 – adds support for multicore, but in terms of partitioning and assurance it is clear that "It must not be construed that compliance to ARINC 653 assures robust partitioning".
- FAA AC 20-193 "Use of Multi-Core Processors" – formalizes the objectives from CAST-32A related to certifying avionics systems based on multicore processors. A draft of this Advisory Circular was released in October 2020. After a comment period, the final version is expected to be published in 2021.
- US DoD MIL-HDBK-516C "Airworthiness Certification Criteria" – provides guidance for certification of avionics systems (including hardware and software). An update is in progress to address multicore considerations, but this has not yet been published.
- US Army "Multi-Core Processor (MCP) Airworthiness Requirements" – this guidance is in draft form and has not yet been formally released.

5.1 FACE conformance is not airworthiness

As explained in previous chapters, certification of FACE conformance only guarantees that a Unit of Conformance (UoC) honors the FACE standard. It is not intended as a solution for meeting airworthiness requirements. Indeed, the FACE standard recognizes that some systems do not require safety certification, such as systems built on the FACE General Purpose Profile using Linux as the operating system. The Safety and Safety-Extended Profiles intentionally limit the breadth of functionality included in a FACE UoC, which can help limit the scope of effort needed to generate certification evidence, but using this profile is simply a starting point.

This brings us to some important conceptual differences when comparing FACE conformance to flight certification (approval of airworthiness). Conformance to the FACE standard is granted to a component, not to an entire system. The FACE Verification Authority (VA) verifies conformance for a UoC, which may be a FACE architectural segment or part of a segment. A UoC cannot span more than one architectural segment, though a "UoC Package" can be comprised of UoCs that span certain FACE segments to form a single logical entity. Thus, FACE conformance is not granted to the entire system, even if it is made up partially or entirely of FACE-conformant UoCs; only the individual UoCs can be claimed to be conformant.

On the other hand, airworthiness is granted to an entire system, not to components. Certification artifacts may be associated with one component, such as the OS, and these artifacts may be accumulated into the overall certification package for the aircraft. An individual component can be described as certifiable, or perhaps as certified within a named system, but the stand-alone component cannot be labeled as "certified".

For example, certification artifacts for Wind River VxWorks 653 v2.5 on PowerPC (which is FACE-conformant to the Safety Base Profile against Technical Standard v2.0) include all documentation required to satisfy the requirements of RTCA DO-178C, consisting of over 65,000 hyperlinked files. This includes items such as the Plan for Software Aspects of Certification (PSAC) and Software Accomplishment Summary (SAS) along with all design reviews, code review, test reviews, functional tests and coverage results. In addition to supporting evidence for the OS, there is also evidence for qualified development tools, and for newer releases, evidence supporting CAST-32A for multicore.

How does FACE conformance work?

"FACE conformance is not granted to the entire system, even if it is made up partially or entirely of FACE conformant UoCs. Only the individual UoCs can be claimed to be conformant."

5.2 FACE conformance supports airworthiness

The job of the FACE UoC supplier is to provide the software as well as associated certification artifacts. The job of the system integrator includes pulling together FACE UoCs to form the overall avionics system design. If acting as the certification applicant, they must also collate the certification artifacts for each UoC into the overall certification package for the system.

The FACE Consortium has two committees whose work helps smooth this process. First, the Technical Working Group (TWG) on Software Safety has been active since nearly the beginning of the consortium. Currently led by Glenn Carter and Joe Wlad, this committee has a mandate to ensure that each edition of the FACE Technical Standard does not interfere with subsequent efforts to obtain flight certification. That is, this group uses a “do no harm” approach, employing preventative measures and eliminating potential airworthiness issues from the technical standard. Second, the EA-25 Airworthiness committee was formed in 2020 with a mandate to augment the preventative maintenance of the TWG Safety group by further identifying how the effort toward FACE Conformance could be leveraged to aid in demonstrating airworthiness. The deliverable for this team is a white paper identifying activities and associated design artifacts generated while pursuing FACE conformance that map to evidence appropriate to support flight certification. Thus, “do no harm” is expanded to “do some good”. The white paper from this committee is expected sometime in 2021.

One high-level example of FACE conformance supporting airworthiness can be found in the FACE reference architecture, which is shown in Fugure 1 on page 4. Both civilian and military airworthiness guidance typically require a software architecture to be defined. A system designed using FACE UoCs inherently builds on a reference architecture that has been publicly reviewed and standardized. The documentation of the architecture, its interfaces, and specification are all artifacts that can contribute to meeting the architecture definition expectations for airworthiness. The system integrator still has some work to do to demonstrate that the architecture is implemented correctly and to ensure that architecture-related requirements are reviewed and pass normal verification and validation.

A more specific example of FACE conformance supporting airworthiness can be found in the FACE technical standard requirements for time partitioning and space partitioning under the Safety and Security Profiles. The isolation provided by partitioning not only contributes to portability and reusability but also eases certification for Integrated Modular Avionics (IMA) systems. A number of airworthiness standards expect partitioning as a means of breaking down a complex system into separate logical components that can be independently and incrementally assured. Partitioning cannot be an afterthought; it is a fundamental mechanism of the computing hardware and RTOS and thus must be part of the design philosophy from the beginning. Because the FACE technical standard requires it, this helps to ensure that a development program starts out on the right track. The OSS supplier provides much of the evidence to verify

System integrator

The system integrator has to demonstrate that the architecture is implemented correctly and to ensure that architecture-related requirements are reviewed and pass normal verification and validation process.

and validate the partitioning environment. The system integrator must then demonstrate that the partitioning environment is implemented and configured appropriately so that the supplier evidence can be accepted. In addition, the system integrator must allocate system resources to partitions and demonstrate that the total resource usage is within the system capacity. Resource allocations include a portion of time on one or more processor cores, a portion of main memory etc.

5.3 Tools

Flight certifying a system comprised of FACE-conformant elements can be a daunting task. Wind River can support your efforts by supplying the Operating System Segment (OSS) along with certification artifacts (as listed in the FACE library/registry) including DO-178 plans such as the PSAC and Software Development Plan (SDP) as well as verification evidence, such as Software Verification Test Cases and Procedures, and Software Test Results. A qualified development tools suite is also included that allows you to develop, configure, build, debug, test, re-test, and certify each independent application independently, incrementally, and asynchronously.

Rapita Systems can support your certification efforts with its suite of verification tools that automate much of the Verification & Validation (V&V) process, including tools for generating unit and system tests, automating test runs and reporting, analyzing structural coverage, and measuring and reporting timing behavior. For multicore systems, Rapita also provides a CAST-32A Compliance Solution that addresses airworthiness for avionics systems by verifying that multicore interference channels are properly mitigated.

RapiTest

The R**VS** Rapi**Test** plugin provides an easy-to-use platform for managing and executing multicore timing analysis tests.

RapiTime

The R**VS** Rapi**Time** plugin automates the collection of on-target timing metrics including WCET.

5.3 About the authors

Steven H. VanderLeest

Principal Engineer for Multicore Solutions at Rapita Systems.

Published on topics related to avionics safety and security in the IEEE Digital Avionics Systems Conference as well as SAE Aerotech. Dr. VanderLeest is also the vice-chair of the FACE EA-25 committee on airworthiness.

Alex Wilson

Director of Aerospace and Defense Market Segment for Wind River,

Responsible for A&D Market Segment in EMEA, APAC, and Japan. As part of the Wind River Aerospace & Defense Industry Solutions Team, Alex is responsible for business strategy including product requirements, sales growth strategy and ecosystem development.

Note: The views expressed in this white paper are the sole opinion of the authors and do not represent official positions of the FACE consortium.

6. Want to learn more?

If you are interested in finding out more about aerospace and defense solutions, visit the Wind River Aerospace & Defense webpage where you gain access to a wide range of white papers and videos about this topic.

www.windriver.com/solutions/aerospace-and-defense

Rapita Systems regularly releases new material and runs training courses on multicore timing analysis worldwide. To make sure you're notified, sign up to our mailing list.

www.rapitasystems.com/newsletter