Optimal Multicore Processing for Safety-Critical Applications

Contents

Introduction

For many years, Moore's Law advances in microprocessors were delivered primarily as increases in processor frequency. Ever since single-core frequencies started to peak in the 2000s, multicore solutions have delivered the latest performance increases for general computing.

Safety-critical and Multi-Level-Security (MLS) applications have been slow to utilize multicore architectures due to the complexity of validating and certifying software and hardware architectures. Of principal concern is how an application running on one core can interfere with an application running on another core, thereby affecting the determinism, quality of service, and ultimately safety. Yet, if the concerns over multicore operation can be addressed, the benefits of smaller size, lower power, and increased performance can be realized.

To help with the safety-critical implementation of multicore processors, several standards have been updated to address multicore issues. ARINC 653 addresses space and time partitioning of real-time operating systems (RTOS) for safety-critical avionics applications. A 2015 update, ARINC 653 Part 1 Supplement 4, addresses multicore operation, including a requirement to support Bound Multi-Processing (BMP). Supplement 5 continues that requirement The Future Airborne Capability Environment (FACE™) technical standard version 3.0 from the Open Group addresses multicore support by requiring compliance with Supplement 4. The Certification Authority Software Team (CAST), including participants from the FAA, EASA, and TCCA, has published a position paper with guidance on certification for multicore systems called CAST-32A. Together, these documents provide the requirements for successfully using multicore solutions for even the highest design assurance level (DAL A) of RTCA/DO-178C and EUROCAE/ED-12C "Software Considerations in Airborne Systems and Equipment Certification."

Benefits of Multicore

If utilizing multicore processors entails more effort and more risk, why bother? The right multicore software architecture can yield a host of benefits:

- **Higher Throughput—**Multi-threaded applications running on multiple cores scale in throughput. Multiple single-threaded applications can run faster by each running in their own core concurrently. Optimal core utilization enables throughput to scale linearly with the number of cores.
- **Better SWaP—**Effective use of multiple cores enables consolidating applications previously running on separate single-core processors to run on separate cores in a single multicore processor. This consolidation can mean reduced size, weight, and power (SWaP). For airborne systems, lower SWaP translates to lower costs and longer flight time. To achieve the desired SWaP results, optimal core utilization is required (i.e. minimal idle time on each core), in a deterministic and high-assurance manner.
- **Room for Future Growth—**The performance potential of multicore processors allows for new requirements and applications to be added in the future. How easy it is to either relocate existing applications to a different core or add new applications to available spare capacity on one or more cores will depend on the flexibility of the operating system architecture and the breadth of tools available for recertification.
- **Longer Supply Availability—**Most single-core chips are obsolete or close to obsolete, and part obsolescence dictates that only multicore processors are available. Moving to a multicore chip allows for choosing a processor at the start of its supply life.

Challenges for Multicore in Safety-Critical Applications

In a single-core processor, multiple safety-critical applications may execute on the same processor by robustly partitioning the memory space and processor time between the hosted applications. Memory space partitioning dedicates a non-overlapping portion of the memory to each application running at a given time. The memory management unit (MMU) in modern processors enforces that memory allocation. Time partitioning divides a fixed time interval, called a Major Frame, into a sequence of fixed sub-intervals referred to as partition time windows. Each application is allocated one or more partition time windows, with the length and number of windows being factors of the application's worst-case execution time (WCET) and required repetition rate. The operating system (OS) ensures that each application is provided access to the processor's core during its allocated time. Applying these safety-critical techniques to multicore processors, however, requires overcoming several complicated challenges, the most difficult being interference between cores via the shared resources.

Interference Between Cores

In a multicore processor, each processing core has limited dedicated resources. All multicore hardware architectures include some shared resources such as memory controllers, DDR memory, I/O, cache, and the internal fabric that connects them.

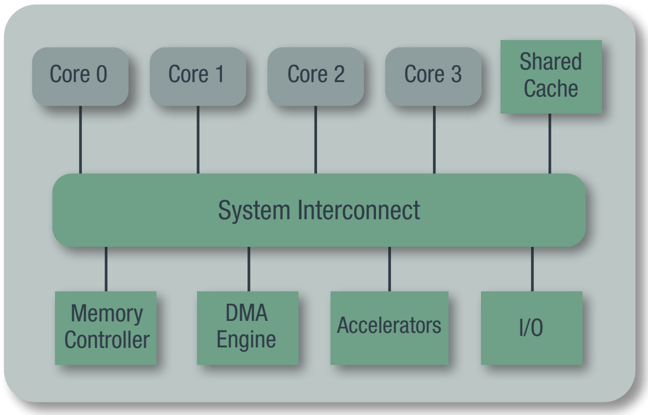

Figure 1: *Separate processor cores (**gray**) share many resources (**green**) ranging from the interconnect to memory and I/O.*

When more than one core tries to concurrently access a given resource there is contention. As a result, a lower criticality application/partition could keep a higher criticality application/partition from performing its intended function, such as causing a screen to freeze. For example, with multiple sources of interference from multiple cores, increases in WCET of over 12x have been observed in a quad-core system when cores only access DDR memory over the interconnect (i.e. no I/O access). Due to shared resource arbitration and scheduling algorithms in the DDR controller fairness is not guaranteed and interference impacts are often non-linear. In fact, tests have revealed a single interfering core increasing WCET on another core by a factor of 8x.

The main certification guidance for addressing interference in multicore processors comes in the form of a joint position paper developed primarily by the Federal Aviation Administration (FAA) and European Aviation Safety Agency (EASA) called CAST-32A. For interference, that document covers managing the interference channels and verifying the use of shared resources. See the sidebar for a full description of the two primary interference objectives.

One approach to addressing multicore interference is for the system integrator to create a special use-case based on testing and analysis of WCET for every application/partition and their worst case utilization of shared resources. Special use-case solutions can lead to vendor lock and re-verification of the entire system with the change of any one application/partition. Special use-case integration is a significant barrier to the implementation and sustainment of an integrated modular avionics (IMA) system. Without operating system mechanisms and tools to support the mitigation of interference, sustainment costs and risks are very high. Changes to any one application will require complete WCET re-verification activities for all integrated applications.

A better approach is to have the operating system manage the interference based on the availability of DAL A runtime mechanisms, libraries, and tools.This approach gives the system integrator an effective, flexible, and agile solution that addresses CAST-32A objectives. Such a general solution simplifies the addition of new applications without major changes to the system architecture, reduces re-verification activities, and in most cases eliminates dependencies on the original system integrator.

CAST-32A Objectives for Interference Mitigation

MCP_Resource_Usage_3: The applicant has **identified the interference** channels that could permit interference to affect the software applications hosted on the MCP cores, and has **verified** the applicant's chosen **means of mitigation** of the interference.

MCP_Resource_Usage_4: The applicant has identified the available resources of the MCP and of its interconnect in the intended final configuration, has allocated the resources of the MCP to the software applications hosted on the MCP and has **verified that the demands for the resources of the MCP and of the interconnect do not exceed the available resources** when all the hosted software is executing on the target processor.

NOTE: The need to use Worst Case scenarios is implicit in this objective.

Porting Single-Core Software Designs to Multicore

Although porting an existing safety system to a multicore platform provides more computing resources, the worst case execution time of a given application can increase due to longer latency to access shared resources or inference from accesses by other cores. New analysis is needed to determine if other resources such as memory, memory controllers, and inter-core communications can become the new bottleneck. Even if every resource runs faster, changes in relative performance can often cause an application to stop working or behave in a non-deterministic manner.

The level of difficulty of the port will depend on a number of factors such as:

- whether the constituent applications were designed to be multi-threaded
- whether there are underlying assumptions about the order of execution of the application time partitions
- differences in communication delays between cores versus within a core
- the amount of margin between the WCET and the available processing time
- whether the multicore approach requires separating I/O into a separate single core as an interference mitigation approach, thereby requiring re-architecting of most applications

Effective Utilization of Multicore Resources

In order to achieve the throughput and SWaP benefits of multicore solutions, the software architecture needs to support high utilization of the available processor cores. This requires support for full multicore features, ranging from enabling concurrent operation of cores (versus available cores being forced into an idle state or held in reset at startup) to providing a mechanism for deterministic load balancing. In general, the more flexible the software multi-processing architecture is, the more tools the system architect has to achieve high utilization. This topic is specifically discussed in ARINC 653 Part 1 Supplement 4 section 2.2.1 as multiple processes (i.e. threads) within a partition executing concurrently across multiple cores and as concurrent partition execution.

Multicore for Integrated Modular Avionics

Integrated Modular Avionics (IMA) combines many avionics processing functions onto a set of shared processing resources. This integrated architecture depends on the application software being portable across a set of common hardware modules. To efficiently use multicore processors for IMA, the different avionics functions are required to be assignable to processor cores in a flexible manner. But without specific multicore operating system capabilities, the flexibility previously associated with single-core IMA solutions is eliminated, and IMA systems based on multicore processors become a serious integration and sustainment risk. For example, IMA system architects and designers may make their initial schedule and core assignments based on utilization estimates, often derived from data sheets or

DO-254 Certifiable Multicore Hardware Complements DO-178C Multicore Software

By Curtiss–Wright Defense Solutions

Full safety certification of the aircraft requires DO-254 certification for the hardware in addition to DO-178 for the software. Current and emerging aerospace requirements demand hardware processing capability that can support multiple functions and applications with mixed levels of safety criticality. These requirements, along with intense computational needs and architectures that include multicore processors, highlight a very clear and pressing need for RTOS technology capable of preventing performance degradation and shared resource contention.

Hardware architectures that include multicore processing technology must be deliberately designed to set the number of active cores and the execution frequency, to specify which MCP peripherals are activated, and to determine the hardware support for shared memory and cache. In safety-critical applications, a multicore processor must be carefully selected and its host board architected, based on several key factors, including a processor's service history, availability of manufacturing and quality data, I/O capabilities, performance levels, and power consumption.

Take, for example, the NXP® QorIQ® T2080 Power Architecture® processor on the Curtiss-Wright VPX3-152 safety-certifiable single board computer (SBC). The quad-core T2080 is capable of meeting the performance requirements of many DAL A applications at a relatively low level of power consumption, and will be logging DAL A flight hours beginning early in 2019. When compared against its smaller-package variant, the T2081, the T2080 also provides additional capability that makes it a more complete and certifiable solution. Despite offering similar power levels, a key differentiator for the T2080 is its 16 available SerDes lanes (compared to the T2081's eight), which effectively doubles the number of functions that can be directly serviced from the processor. This simplifies the overall board design and certification effort. For systems with even more demanding SWaP constraints, Curtiss-Wright also offers a DO-254-certifiable SBC featuring NXP's Layerscape 1043A Arm® processor, which is designed specifically for providing power-efficient 64-bit processing at a low power consumption level.

The full capability of our carefully selected multicore processors is realized when complemented by an RTOS that enables system designers and integrators to utilize all available compute power from the processor's cores in a high-assurance manner. To that end, all of our safety-critical multicore SBCs support INTEGRITY®-178 tuMP™, which provides deterministic, user-defined core and scheduling assignments that can ensure the performance capabilities of multicore hardware are fully achieved.

limited empirical results. As software enters into the integration phase, which is late in the development cycle and costly to change, the following occurs: (1) more features are added to the software applications, (2) hardware does not perform as expected, and (3) the original 50% spare utilization drops to 10% utilization. As a result, the initial partition schedule and core assignments need massive rework.

In order to avoid these hidden pitfalls, the underlying multicore operating system needs to support the following: (1) ease of application migration across cores, (2) ability to easily define new or multiple Major Frame schedules, and (3) ease of adding and/or removing cores assignments for a new or existing application.

Safe and Secure OS Virtualization in IMA and Open Mission Systems

In an effort to expand the benefits of IMA in a multicore environment, some avionics architectures may require support for Linux, Android, or Windows so that application-specific software can run in a virtualized Guest OS partition. This is often associated with non-critical and non-real-time applications. In addition to core assignment flexibility, such a virtualization environment has a critical dependency upon the deterministic use of the shared processor resources.

Similarly, Open Mission Systems (OMS) seeks to utilize a standard Open Compute Environment (OCE) based on enterprise operating systems and enterprise processor architecture. Because security is a concern with enterprise-level solutions, virtualization is typically employed in an effort to isolate applications from each other.

One of the biggest challenges of a virtualized avionics environment is providing a robust security environment. Although traditional hypervisors attempt to isolate virtualized Guest OSes from each other and system resources, that security is only as good as the underlying hypervisor. Once a security flaw is exploited, an application in one Guest OS can gain access to data in another Guest OS. An often overlooked type of security vulnerability can come through denial of service or even degraded service if the hypervisor does not provide mechanisms to enforce the fair use of the shared multicore resources.

One answer to those security concerns is to utilize a virtualization layer in user space running on top of a separation kernel. That approach minimizes the kernel size while enforcing application isolation and permitting only explicitly authorized communication flow. If the applications will be operating at multiple levels of security (MLS), then a high assurance kernel is required, and it needs to be able to guarantee information flows between the Guest OSes and their applications through an MLS guard/downgrader. This can be challenging, as some separation kernels do not support running an MLS application like a guard/downgrader, thus limiting their ability to meet cross-domain requirements.

In addition to that security advantage, running the virtualization layer on top of the separation kernel instead of inside a bare-metal hypervisor also has the performance advantage for real-time applications. Only the non-real-time applications running in the Guest OS will pay the virtualization performance penalty. The real-time applications can run directly on the seaparation kernel acting as the Host RTOS, thereby realizing the full real-time performance.

Software Multi-Processing Architectures

Like multi-processor systems, the software architecture on multicore processors can be classified by the amount of sharing and coordination among cores. 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 hypervisor/Guest OS pair. Each core runs a different application with little or no meaningful coordination between the cores in terms of scheduling. This decoupling can result in underutilization due to lack of load balancing, difficulty mitigating shared resource contention, and the inability to perform coordinated activity across cores such as required for comprehensive built-in test.

The modern alternative is Symmetric Multi-Processing (SMP), where a single OS controls all the resources, including which application threads are run on which cores. Processes and threads can be coordinated across cores, and utilization can be maximized. This architecture is easy to program because all cores access resources “symmetrically,” freeing the OS to assign any thread to any core.

Not knowing which threads will be running on which cores is a major challenge and a risk for deterministic operation in safety-critical systems. To address this, CAST-32A references the use of Bound Multi-Processing (BMP). BMP is an enhanced and restricted 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 Supplements 4 and 5 section 2.2.1, which states: “Multiple processes within a partition scheduled to execute concurrently on different processor cores.”

INTEGRITY-178 tuMP Multicore Solution

The INTEGRITY-178 tuMP high-assurance RTOS is the leading multicore RTOS for safety-critical applications. INTEGRITY-178 tuMP complies with ARINC 653 Part 1 Supplements 4 and 5, which requires BMP capability in ARINC 653 partitions. The RTOS also complies with ARINC 653 Part 2 Multiple Processor Cores Extensions, which requires SMP support in ARINC 653 partitions. INTEGRITY-178 tuMP was the first RTOS certified to the FACE technical standard version 3.0, and remains the only RTOS certified for all three major multicore processor architectures— Arm, Intel®, and Power Architecture.

INTEGRITY-178 tuMP’s Time-variant Unified Multi-Processing (tuMP) approach provides maximum flexibility for porting, extending, and optimizing safety-critical and security-critical applications to a multicore architecture. It starts with a time-partitioned kernel running across all cores that allows any combination of AMP, SMP, and BMP applications to be bound to a core or groups of cores called affinity groups. It then adds time-variance so that partition time windows do not need to be aligned across cores.

The INTEGRITY-178 tuMP operating system’s added capability to change core assignments, as required for IMA, can be used selectively to give some of the applications for a given partition time window more processing time and resources, or it can be used to run a whole new set

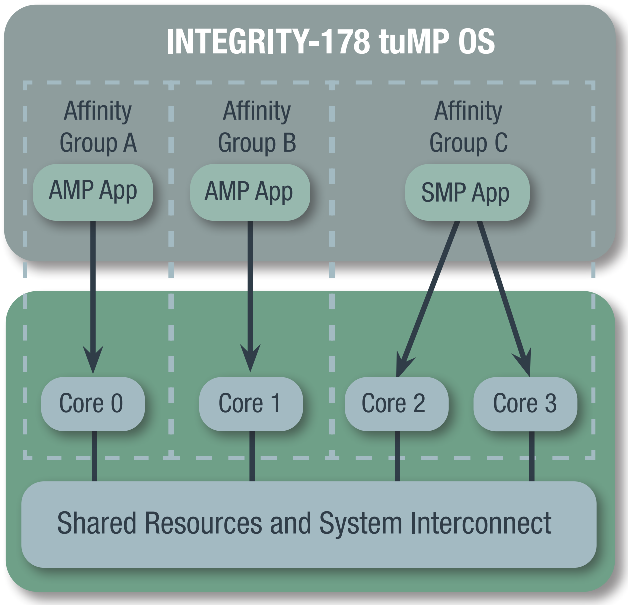

Figure 2: *Example of mixed AMP and SMP/BMP in INTEGRITY-178 tuMP on a multicore processor.*

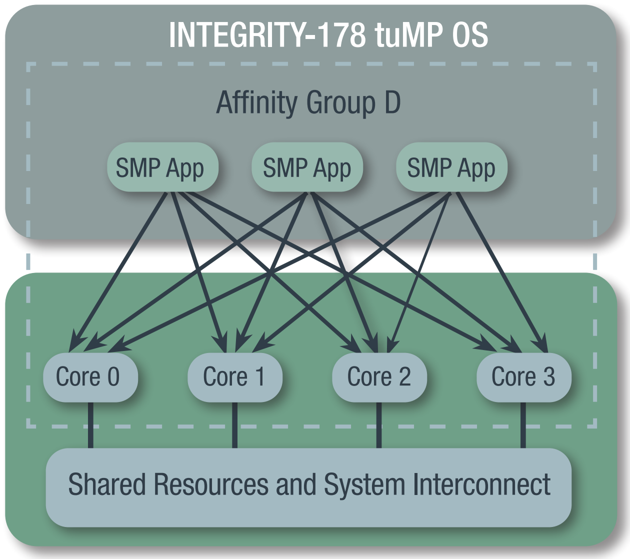

Figure 3: *Example of full SMP mode in INTEGRITY-178 tuMP.*

of applications. For example, a critical image processing function may have certain modes of operation where the complexity of the image data being processed requires additional cores but the deadline remains the same.

The capabilities of INTEGRITY-178 tuMP enable multiple independent safety- and security-critical applications to execute on a multicore operating environment in a predictable, bounded, and application-independent manner. The

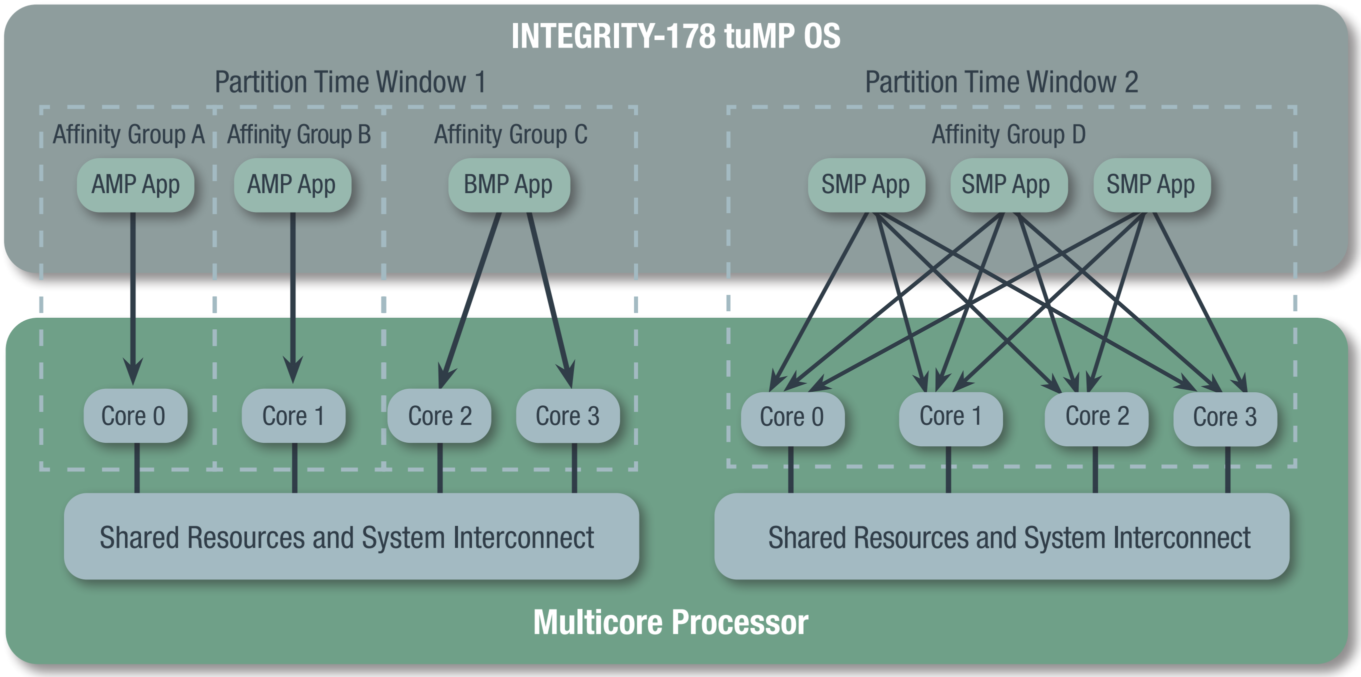

Figure 4: *The time-variant capability of INTEGRITY-178 tuMP allows different bindings of applications to cores in different partition time windows.*

tuMP partition-enforcing scheduling method results in a unified OS that provides practical time-variant scheduling of AMP, BMP, and SMP applications simultaneously.

INTEGRITY-178 tuMP Examples

With a unified multi-processing architecture, INTEGRITY-178 tuMP enables the use of AMP, BMP, SMP or any combination of them in the same processor. In the first example (Figure 2), there is one application bound to core 0 and another application bound to core 1, so they are operating in AMP mode. A third application is bound to cores 2 and 3, therefore its threads can go to either core in SMP mode or they could be bound to a specific core in BMP mode via task affinity. Because there is no overlap in core assignments, they all can execute simultaneously.

In the second example (Figure 3), there are three applications, each of which can execute threads on any of the four cores. This full SMP mode allows the operating system to perform deterministic load balancing according to the overall task priorities, resulting in optimal performance.

Supporting multiple applications executing on one or more cores in the same partition time window enables support of event-driven applications such as client-server, where the clients' requests are immediately serviced by a server executing in the same time partition window (instead of being delayed to another partition time window). Likewise, interrupt-driven applications are immediately serviced with this architecture.

The third example (Figure 4) shows the flexibility of INTEGRITY-178 tuMP. Here the application configuration from the first example runs in the first partition time window, and then the application configuration from Figure 3 runs in the second partition time window.

One use of this flexibility is to enable built-in test (BIT) while using a virtualized/Guest OS. Typically the Guest OS, such as Linux or Windows, runs on a dedicated core consuming all of the core's processing time and resources. The continuous BIT suite, however, needs to run on all the cores at the same time in order to coordinate the testing of all the cores and shared resources. Both requirements can be accommodated using INTEGRITY-178 tuMP by having the BIT application assigned to all cores in a separate partition time window.

These same techniques can be used to plan for future expansion. By leaving one core unused and one partition time window unused across all cores, an existing application can grow in either direction or both. A new application can slide into the unused core for one or more time windows, or can use one or more cores in the spare time window.

This ability to assign applications across cores and time windows maximizes flexibility to port existing applications, add new capabilities to meet future needs, and to adapt designs to meet certification requirements.

Interference Monitoring and Mitigation

As mentioned earlier, multicore processor architectures include several shared resources such as memory, cache, I/O, and the interconnect that connects the cores to the other resources. Contention for these shared resources can create interference between cores, even when there is no explicit data or control flow between applications on different cores. These interference channels can cause non-deterministic behavior in critical software applications.

INTEGRITY-178 tuMP includes a Bandwidth Allocation and Monitoring (BAM) capability to observe interference channels and mitigate them. Based upon more than 60 staff-years of research and development into multicore interference analysis and mitigation strategies, BAM monitors and enforces the allocation of bandwidth to shared resources for each of the cores. Green Hills has implemented an internal mechanism for INTEGRITY-178 tuMP bandwidth allocation and monitoring that uniquely uses an extremely small time quantum in order to enforce the cores' use of shared multicore resources as opposed to the typical approach using high-level fault detection. The BAM mechanism expects the obvious: that applications do NOT have a fixed bandwidth utilization curve by default, but BAM enforces a fixed bandwidth utilization curve for them.

The system architect decides how much bandwidth to allocate to each core based on the functional requirements of the applications or their design assurance levels. When applications on a particular core reach the threshold bandwidth for a given BAM time quantum, that core is cut off from consuming shared resources until the next BAM time quantum. Using this mechanism, a DAL-A application running on core 0 can be allocated a set amount of resources, such as 60% of the total bandwidth, while the other 3 cores could be allocated only 15%, 15%, and 10% respectively. BAM is developed to DO-178 DAL A objectives, and it allows integrators to mitigate interference issues.

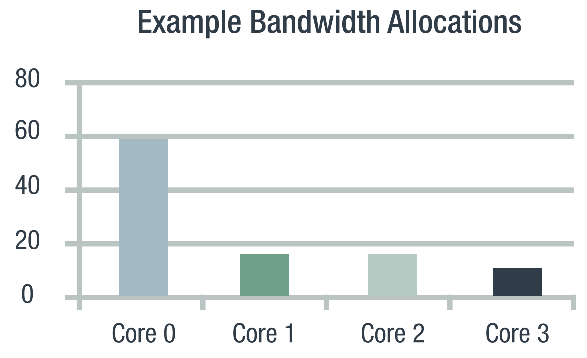

Figure 5: *Example bandwidth allocations set and enforced using BAM.*

Setting the proper bandwidth allocation requires analysis and testing of the application. To aid in that analysis, Green Hills Software provides interference and DMA generating libraries and a bandwidth reporting library. The interference and DMA generating libraries are tailored to each processor architecture and contain hundreds of interference profiles to simulate interference beyond what is found in typical applications. Running the interference and DMA generating libraries on all cores not used by a particular application concurrently with the application execution provides the new multicore worst case execution timing (WCET).

The bandwidth reporting library uses the interference and DMA generating libraries to get a measured picture of the total amount of bandwidth available after accounting for the interference. Knowing the total bandwidth available will aid in setting the bandwidth allocation thresholds in BAM. The bandwidth reporting library runs the interference and DMA generating libraries across a configurable number of cores concurrently. Specific subsets of the hundreds of interference profiles can be selected in order to tailor the evaluation more closely to the expected applications, and custom interference profiles can be created. The available bandwidth will depend not only on the processor model but also the memory type, clock speed, configuration registers, and which interference profiles were selected to approximate the final application configuration.

Case Study: Critical Airborne Systems Based on Multicore Processors

by CMC Electronics

CMC Electronics provides smart displays and processing computers for airborne applications. Our aircraft OEMs and system integrators have demanding requirements for the next generation of smart platforms: critical systems must 1) follow open architecture principles, 2) be capable of hosting software applications developed to the highest assurance levels, 3) provide sufficient growth capability, and 4) operate in harsh environments at high temperatures.

To meet these challenges, we introduced the MFD-3068 Smart Multi-Function Display and the PU-3000 series of Avionics computers, CMC's 3rd generation of Smart Displays and Processing Computers. Both systems leverage our next-generation of MOSArt™ (Modular Open System Architecture) middleware and a multicore processor supported by a high integrity operating system. MOSArt was founded on non-proprietary industry standards for the partitioning of applications (ARINC 653). In selecting an operating system it was clear that it must enable our critical systems to achieve the throughput and hardware consolidation benefits available from a multicore processor. The operating system's software architecture must also be capable of meeting the open systems requirements for multicore systems as defined in ARINC-653 Supplement 4. Equally important from a system certification, growth and sustainment point of view, the operating system's features and capabilities must enable EAV to mitigate, on a continuing basis, the risks associated with multicore processors in critical systems.

The selection of a multicore processor offered longer term availability while providing significant flexibility in meeting the processing throughput and room for growth desired by our customers. To capitalize on the potential of this architecture and after a very thorough and extensive trade study, we selected Green Hills Software's INTEGRITY-178 tuMP Multicore Real-Time Operating System (RTOS) because it met both our short and long term throughput goals without jeopardizing our safety certification requirements. Its Unified Multi-Processing approach, including the usage of Affinity Groups, provides the flexibility and integrity necessary to meet these challenges.

Taken together, the interference and DMA generating libraries, bandwidth reporting library, and BAM runtime mechanisms provide the tools necessary for a system integrator to determine multicore worst case execution times, mitigate interference, and certify multicore systems. These tools provide a complete solution to mitigating multicore interference. These interference mitigating capabilities provided by Green Hills Software reduce certification risk and enable faster time-to-market by simplifying the verification and analysis activities.

BAM reduces risk and simplifies the development, integration, deployment and sustainment of critical systems. BAM enables optimal core utilization in critical systems yielding superior SWaP reduction and Spare Computing Capacity. This bandwidth allocation and monitoring capability is essential for IMA OEMs and developers to meet the IMA requirement that applications are independently modifiable. The capability to independently modify applications is necessary to meet the high-level IMA goals of providing cost-effective upgrade paths and introducing new operational capabilities without retesting and reverifying the entire system.

Security: The Final Frontier

Today's safety-critical systems face a variety of threats from both unintentional and malicious actors. If the software is changed maliciously or even unintentionally from the certified configuration, it is no longer safe. Bottom-line, a system that is not secure puts safety at risk. Nor is it sufficient to have separate OS products with one being safe and another being secure, as the primary OS or hypervisor needs to be both safe and secure. Green Hills recognizes this requirement by building both safety and security into the same INTEGRITY-178 tuMP RTOS.

NEAT Security Policy Attributes

The four main security attributes of a high-assurance separation kernel (i.e. security monitor):

Non-bypassable:	An application cannot bypass the security monitor.
Evaluatable:	The security monitor is modular, small in size, and sufficiently low in complexity to support rigorous evaluation.
Always-invoked:	Each and every access and communication is checked by the security monitor.
Tamperproof:	The system prevents unauthorized changes to the security monitor code, configuration, and data.

One proven approach for a Multiple Independent Levels of Security (MILs) operating system is to architect it as a separationb kernel. A separation kernel is intended to fully isolate multiple partitions and control the information flows between applications/partitions and external resources. In part, that includes protection of all resources from unauthorized access, isolation of partitions except for explicitly allowed information flows, and a set of audit services. The result is that a separation kernel provides high-assurance partitioning and information flow control that satisfy the non-bypassable, evaluatable, always invoked, and tamperproof (NEAT) security policy attributes.

In 2007, the Information Assurance Directorate of the U.S. National Security Agency (NSA) published the Separation Kernel Protection Profile (SKPP), a security requirements specification for separation kernels suitable to be used in the most hostile threat environments. In 2008, INTEGRITY-178 became the first and only operating system to be certified against the SKPP. That certification was to the highest Evaluation Assurance Level (EAL 6+) for general software products. Even though the SKPP has now been sunsetted, the evaluation criteria remain the strictest the industry has seen and is still specified by programs of record. INTEGRITY-178 tuMP continues to meet the SKPP's rigorous set of functional and assurance requirements for those customers needing it.

Beyond the approval as a MILS separation kernel, INTEGRITY-178 tuMP provides a complete set of APIs that were also evaluated by the certification authority for use by Multi-Level Security (MLS) applications within a secure partition, e.g. an MLS guard, which is a fundamental requirement in a cross-domain system. Because both safety and security are designed into the same product, those secure APIs include support for multithreading, concurrent execution on multiple cores, and flexible core assignments at the configuration file level, all within the secure MLS environment. The unique bandwidth allocation and monitoring capability in INTEGRITY-178 tuMP can be used to thwart denial-of-service attacks from compromised partitions/applications resulting from the unintended or malicious use of the multicore processor's shared resources.

Essential Multicore RTOS Requirements for Critical Airborne Systems

Time-Variant Unified Multicore Processing	Optimal core utilization in a high-assurance and deterministic manner resulting in maximum system throughput, lower system SWaP, and greater spare computing capacity
Shared Resources Bandwidth Enforcement	Bandwidth allocation and monitoring of shared multicore resources in order to mitigate the risk of interference from the shared resources and simplify the development, integration, deployment, and sustainment of critical systems
Independent Subsystem Decomposition	The ability to assign one or more cores to applications and partition time windows independently of other subsystems
Multicore Standards Support	Complete support for open standards addressing multicore processing, including ARINC 653 Part 1, Supplements 4/5 and FACE version 3.0
Secure Guest OS Virtualization	Reduces certification burden of the hypervisor to the same level as its actual Guest OS application. Guest OS partitions are subject to the same Bandwidth Enforcement as the non-virtualized applications (eliminates risk of interference caused by Guest OS and its applications).
Tightly Integrated Development Tools	A development environment tightly integrated with the high-assurance multicore RTOS that has a proven track record of success for C, C++, and Ada
Safety-Critical Middleware	DAL A-compliant file system and networking components based on a client/server design – server resides in any core or partition and serves multiple clients at different safety levels

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 SO50 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