MPSOC HYPERVISOR: THE SAFE & SECURE FUTURE OF AVIONICS

Steven H. VanderLeest, DornerWorks and Calvin College, Grand Rapids, MI, USA

Dagan White, Xilinx, San Jose, CA, USA

Abstract

Future avionics must provide increased performance and security while maintaining safety. The additional security capabilities now being required in commercial avionics equipment arise from integration and centralization of processing capabilities combined with passenger expectations for enhanced communications connectivity. Certification of airborne electronic hardware has long provided rigorous assurance of the safety of flight, but security of information is a more recent requirement for avionics processors and communications systems. In this paper, we explore promising options for future avionics equipment leveraging the latest embedded processing hardware and software technologies and techniques.

The Xilinx Zynq® UltraScale+™ MultiProcessor System on Chip (MPSoC) provides one promising avionics solution from a hardware standpoint. The MPSoC provides a high performance heterogeneous multicore processing system and programmable logic in a single device with enhanced safety and security features. Combining this processor solution with a safe and secure software hypervisor solution unlocks many opportunities to address the next generation of airborne computing requirements while satisfying embedded multicore hardware and software certification objectives.

In this paper we review the Zynq MPSoC and use of a software hypervisor to provide robust partitioning via virtualization. Partitioning is well established to support safety of flight in Integrated Modular Avionics (IMA) while maintaining reasonable performance. Security is a more recent concern, gaining attention as a vulnerability that can also affect safety in unanticipated ways. Hypervisor-based partitioning provides strong isolation that can reduce covert side channels of information exchange and support Multiple Independent Levels of Security (MILS).

Introduction

Future avionics equipment must provide more performance, flexibility, safety, and security – all while containing costs and restraining the growth of Size, Weight, and Power (SWaP).

Future avionics must improve *performance*. Performance improvements to existing capability by upgrading Line Replaceable Units (LRUs) may involve reducing end-to-end input/output (I/O) latency, improving bus communication bandwidth, or speeding up boot time. Performance improvements can also support new functionality. NextGen features will continue to notch up expectations for avionics to support Performance Based Navigation (PBN) such as Required Navigation Performance (RNP). Data fusion needs are pushing the demand for greater computational bandwidth, particularly for Unmanned Aerial Vehicles (UAVs) that have an increasingly large and diverse array of sophisticated, multispectral sensors[1]. “The prevalent and ever-increasing use of airborne sensors is driving the need for more capable and compact military electronics with which to process, share, and exploit the data acquired.” [2] The flying public on commercial aircraft expects improved connectivity in coming years and this expectation will also push performance: “proliferation of personal electronic devices (PED) in recent years has created a community of consumers accustomed to 24/7 connectivity, including while in flight. By one estimate, 80 percent of all airline passengers carry a smartphone, tablet or laptop.” [3] This new connectivity is not a replacement for other information channels, but in addition to existing systems, such as seat-back displays. Furthermore, many passengers already carry more than one connected device, such as a smartphone and laptop. Ian Dawkins, CEO of OnAir also points to crew expectations for connectivity: “They want to be connected. What is important is having their tablet integrated into the airline IT infrastructure. This connectivity provides real-time passenger

information. It can help with baggage tracking, e-health services, etc." [4]

Future avionics must improve *flexibility*. Flexibility includes expansion capability to add new functions with a faster time to market while maintaining safety and containing costs. [5] Flexibility can foster a competitive supply environment, minimizing costs while maximizing innovation. At the same time, choice in vendors provides a more robust supply chain (fewer sole source risks). Dynamic flexibility, such as load balancing or fail over capability, can be more challenging to certify, but it can also provide higher overall reliability and safety.

Future avionics must improve *safety and security*. Safety is focused on reducing the probability of injury or death due to equipment fault or failure. Historical improvements in safety have been significant, resulting in an excellent safety record for modern aircraft. These improvements have resulted because of a focus on reliability, reducing the number of faults that occur and minimizing their impact if they do occur. Recently, the industry has more clearly recognized the impact that security issues can have on safety, noting that exploited security vulnerabilities could lead to a fault that impacts safety. Security is focused on reducing the probability of information leakage (inappropriate reading of data) or corruption (inappropriate modification of data). Protection of classified military data has long implied information assurance measures [6], but in a highly connected passenger environment, security concerns now also apply to protecting confidential information for commercial customers. In both military and commercial aircraft, security and safety concerns overlap, affecting the design of the avionics system architecture in order to present unauthorized intrusion to any subsystem that impacts safety of flight.

Future avionics must *contain cost*. Cost can be controlled via continued focus on reducing SWAP, such as further consolidation through an Integrated Modular Avionics (IMA) strategy. IMA can reduce the number of Line Replaceable Units (LRUs) and number of wires via coalescence of formerly federated units. For example, Honeywell found that for their work on the A380 avionics, IMA provided "a 50% volume and 40% weight reduction from previous federated surveillance equipment" [7] Cost containment strategies must also address flight certification. While IMA promises to reduce certification costs for reused software in partitions, the partitioning environment itself (both software and hardware) must be certified to the highest level of any of the hosted software. Cost containment must also address obsolescence. For example, Commercial Off The Shelf (COTS) microprocessors are often produced for only a few years, while avionics must last decades: an analysis of the aging of 23 US military aircraft projected the age at retirement to be over 35 years for 18 of them (up to 84 years in one case) [8]. While some parts go obsolete, other systems long continue to function, so new electronics must connect with on-board legacy systems. Mark Grovak, avionics business development manager at Curtiss-Wright Controls Defense Solutions says, "A big part is having the number-crunching power to blend different protocols into something older avionics can understand, utilize, and display." [2]

All of these future improvements in performance, flexibility, safety, and security (while containing costs) will require innovation in avionics hardware and software. This paper is not a comprehensive survey of potential innovations, but rather an introductory look into a few related ideas. For hardware, where multicore processors are an obvious contender to improve avionics, we look at the particular case of programmable versions of multicore chips. For software, virtualization is the associated software technology necessary to fully leverage multicore hardware.

Literature Review

The literature contains only a few published papers on performance of *embedded hypervisors*. A few studies look specifically at hypervisors in safety-critical domains. The Lastera [9] report compares Xen and Qubes performance, but uses simple synthetic benchmarks that may not be representative of complex, real applications. Crespo [10] evaluates overhead for the XtratuM hypervisor, but also uses a synthetic benchmark: a simple counting loop. Campagna [11] briefly reviews available embedded hypervisors (not necessarily in safety critical domains) and then focuses on performance measurements for a prototype hypervisor on an ARM architecture. Their measurements focus on the overhead of the hypervisor as a fraction of the total

resource available (CPU time, memory, I/O bandwidth) using a mix of real and synthetic benchmarks. The lack of published hypervisor performance measurement results that use sufficiently complex and representative workloads is a gap that we hope will be filled in the future.

While there is general agreement that future performance improvements will come from multiple CPU cores on one chip, the aerospace community has been cautious to adopt *multicore technology*. Multicore processors began appearing around 2005, and shortly after, avionics research publications began to address this new technology. The first mention of multicore in an AIAA paper is in 2006 on the topic of flight controls for UAVs [12]. The first mention in a DASC paper was buried in a citation of a 2006 paper, and not until 2008 does the term get direct treatment in two papers: one on improved performance for jet fighter avionics [13] and the other on Multiple Levels of Independent Security (MILS) [14].

The literature on multicore processors in avionics has largely focused on *Worst Cast Execution Time* (WCET) analysis, particularly in the face of cross-core interference for resources. This interference can degrade the robustness of partitioning and thus jeopardize the promise of IMA to provide a consolidated platform with safety as good as that for the equivalent federated systems. Kinnan [15] lays out some of the key challenges to certification of multicore processors, focusing much of his attention on the problem of analyzing interference for shared resources. The MULCORS [16] research study conducted for the European Aviation Safety Agency also identifies flight certification issues related to multicore COTS platforms. They include an overview of processor roadmaps and then present case studies of four example processors. Researchers have explored hardware and software means to prove determinism and isolation for multicore systems, such as the University of Illinois team working on an approach to define single core equivalence [17]. Wilhelm also points to this significant challenge of certification of multicore hardware, namely, the lack of predictability: “interference on shared resource is the main culprit for the variability of execution times and the resulting un-predictability.” [18] He and his colleagues conclude in a later paper [19] that the general case is intractable and suggest methods of configuring the hardware to improve predictability of worst-case timing. Kliem [20] also looks at configuration of hardware to reduce interference, in this case using the flexibility of an Field Programmable Gate Arrays (FPGA). Others have suggested improving predictability via software [21].

While multicore processors present a challenge to *safety certification*, they also present an opportunity to improve safety via redundancy. For example, Abella [22] examines use of multicore in fault recovery via a layer of detection, diagnosis, recovery, and reconfiguration. Huyck [23] considers use of multicore processors in an ARINC 653 system, focusing on various approaches to process scheduling. A Lockheed Martin paper [24] explores the technical and performance aspects of virtualization to support both safety and security on a multicore system, including the challenges of consolidating Input/Output (I/O). Parkinson [25] considers an approach to combining safety and security in one system, describing a system that uses a MILS hypervisor that hosts an ARINC 653 RTOS as a virtualized guest. The paper explains that the ARINC 653 partitioning is delegated to a domain of the hypervisor due to the cost of proving the security attributes: “an ARINC 653 safety critical RTOS is unlikely to meet the security requirements due to the size of the kernel and because it may allow device drivers to be implemented in kernel space.” Apparently, they concluded that adding all the required ARINC 653 functionality to the separation kernel used at the heart of a MILS hypervisor [14] would push its total size beyond that which is feasible for security analysis at the highest levels of assurance, which requires formal methods. While the cost of formal methods analysis is burdensome, limiting the size of the separation kernel may not be sufficient. McDermott [26] claims that separation kernels with full mathematical proof of correctness are no longer practical, particularly in the face of hardware feature creep. Instead, his research group at the Naval Research Lab is exploring how closely one can approach the rigorous isolation of a separation kernel using a separation virtual machine monitor. They use Xenon, a prototype VMM based on a heavily modified version of Xen.

A Future for Airborne Processors

Future avionics will depend on improvements to hardware, software, and the integration of the two. In this section, we examine one promising direction for hardware: multicore processors. In particular, we examine a recent development, the introduction of hardened multicore processors and programmable logic in a single chip.

Programmable Multicore SoC

The use of single core processors is declining. Almost all future performance increases in avionics computer processing hardware will be achieved via adding additional cores: "The last decade has seen the emergence of multicore and manycore architectures…. These architectures are replacing the old monocore processors in all commercial domains, including avionics. The integration of monocore processing modules in IMA architectures is becoming more and more expensive because of the ongoing scarcity of these components. As a consequence, multi and manycore processors will be unavoidable in the next IMA-2G architecture." [27] Christopher Mims points to three limits to faster uniprocessors: memory bottlenecks, limits to Instruction Level Parallelism, and heat dissipation. "Of the three, the power wall is now arguably the defining limit of the power of the modern CPU. As CPUs have become more capable, their energy consumption and heat production has grown rapidly. It's a problem so tenacious that chip manufacturers have been forced to create 'systems on a chip' – conurbations of smaller, specialized processors." [28] The impact of the power wall is visible in the following two charts. Figure 1 illustrates the transistor count for a selection of processors in the last 45 years[1]. The vertical scale is logarithmic, thus what appears to be a relatively straight line represents exponential growth – a visual demonstration of Moore's Law, the doubling of the number of transistors every 18 to 24 months. Compared to the confident upward march in transistors, the accompanying increase in clock frequency stalled in the early 2000's, as shown in Figure 2. At this point, manufacturers could no longer increase performance with faster switching due to the power wall.

[1] Data for Figure 1and Figure 2 is available from the first author.

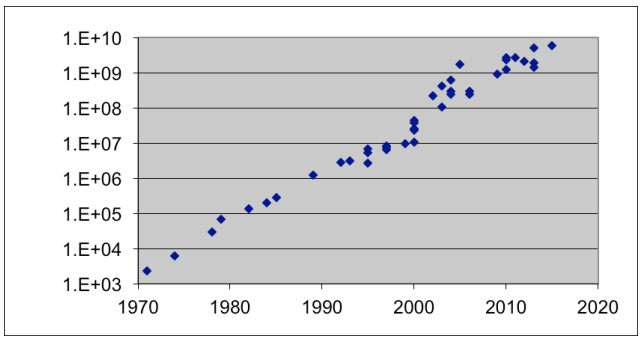

Figure 1: Processor Transistor Count

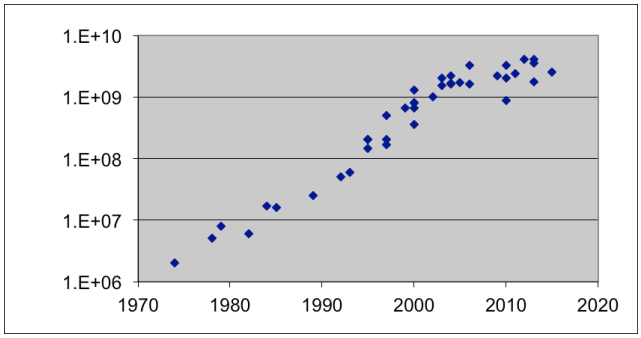

Figure 2: Processor Clock Frequency (Hz)

Once clock frequencies could not be further increased due to power dissipation issues, chip manufacturers turned to multicore approaches to improve performance, i.e., more CPUs at the same frequency rather than one CPU at a higher frequency. However, even multicore may face a power limitation, leading to more and more dark silicon – areas of a computer chip powered off in order to maintain an overall power budget. Reconfigurable computing via an FPGA can provide performance/watt improvements beyond what multicore by itself can achieve.

While multicore benefits for performance improvement are clear, this technology also presents new challenges to safety. "Many-core platforms involve several non-predictable mechanisms for managing the resource sharing which make it hard to ensure time predictability. Most of the works in the literature on many-core architectures are concerned with high performance: the idea is to extend and adapt current operating systems to the new architectural organization. The main concern of an embedded system designer is somehow different: he/she wants to determine the worst-case behaviors in order to verify that the hard real-time requirements

are always met, rather than to study the average performance." [27] Much of the research referenced in the previous section deals with this problem of WCET in the face of cross-core interference. The FAA recently provided guidance for certifying dual cores in position paper CAST-32 [29].

Although multicore architectures are relatively new, the task of analyzing predictability of parallel processing has been with us much longer. Even with a monocore processor, smart memory and I/O devices presented similar challenges. Consider an IMA system implementing partitioning on a single processor computer that contains a Direct Memory Access (DMA) engine. The DMA engine is tasked with moving data in memory for a particular partition. This task is initiated by the processor, but then carried out autonomously by DMA. The system design must account for this use of memory resources outside of the processor and prevent unintended interference, e.g., by pausing the DMA activity at the minor time frame boundary before the next partition starts, or by providing exclusive, dedicated memory access channels in hardware. Similar interference issues existed for peripheral I/O devices well before multicore appeared on the scene. Consider two partitions that use the same Ethernet channel for communication. The system design must account for this shared use of I/O and arbitrate access in order to prevent cross-partition interference.

Programmable Logic and Soft-Core Processors

Today, many avionics platforms incorporate an FPGA coupled with an independent processor, or in some cases an FPGA has been used to replace a discrete processor solution.[30] The flexibility FPGAs offer can provide benefits over a standalone single or multicore processor device. The FPGA is fully customizable, allowing designers to incorporate avionics-specific interfacing standards and high performance signal processing, amongst other things. When an FPGA is used in an avionics platform alongside a processor it is commonly interfaced to a processor through a PCIe, SRIO, or other standard interface, ultimately requiring signaling through a circuit board. In some instances an entire processor may be emulated in an FPGA, and in this case the processor IP core implemented in the FPGA is referred to as a soft-core processor. For instance, Xilinx's DO-254 MicroBlaze processor has been implemented in avionics systems as a replacement for obsolete x86-based processors and as a companion to other graphics processing IP and standard aircraft interfaces. In some cases an MPSoC may be built with two or more soft cores in an FPGA along with graphics processing cores and other IP cores. With full control over hardware implementation in a soft core MPSoC design, one can address many challenges associated with multicore architectures albeit with a system that may lag performance and consume more power than a modern COTS processor. This being said, soft-core processor solutions can offer several advantages over COTS processor solutions. Obsolesce of COTS devices is a key concern for avionics developers, and this can be mitigated when using programmable logic devices that are typically offered on the market for 10 to 15+ years [31], combined with the fact that the soft-core processor can be ported to future FPGA architectures while preserving the processor implementation and associated embedded software. The marriage of multicore processor with programmable custom logic in a single device offers yet another option with significant compelling advantages over discrete processor and FPGA approaches.

Case Study: Xilinx Zynq UltraScale+ MPSoC

The Zynq MPSoC offers unprecedented capabilities, incorporating a multiprocessor system and programmable logic, within a single device while providing fine grain control of the hardware, allowing avionics developers to satisfy both safety and security objectives while meeting system level performance and functional needs. Xilinx developed this device from the ground up with safety and security features as device level requirements. Safety and security have been addressed at multiple levels with robust isolation and partitioning capabilities, built-in safety test routines, full Error Correcting Code (ECC) features, etc. Figure 3 shows a block diagram of the Zynq MPSoC.

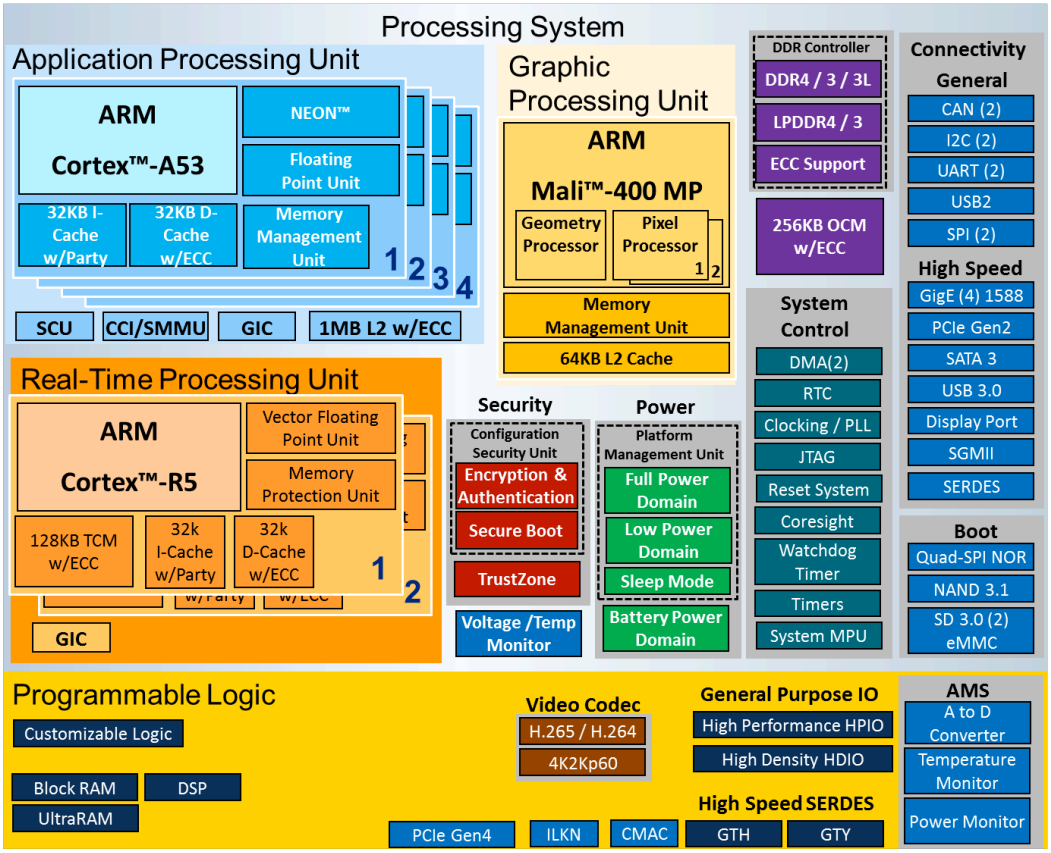

Figure 3: Zynq MPSoC Block Diagram (used with permission of Xilinx)

The quad-core Cortex-A53 Application Processing Unit (APU) is a power-efficient, high-performance processor cluster, supporting 32-bit or 64-bit processing at 1.3 GHz with the ARMv8 instruction set, and with support for TrustZone and hardware virtualization which can be leveraged with a hypervisor to meet both safety and security needs. With 2.3 DMIPS/MHz, a single processor core can run at 2,990 DMIPS, scaling to 11,960 DMIPS with four cores. In addition to meeting high performance needs, the APU supports safety critical missions with Error Correcting Code (ECC) on L2 cache and L1 data cache, and parity on L1 instruction cache. Each core can be enabled or disabled as required for the target design application, and the cores can be managed with a hypervisor to support time and space partitioning of embedded software applications. The APU subsystem can meet current certification guidelines with a single or dual-core use while allowing future system enhancements and upgrades beyond a single or dual-core as certification policy and experience expands. In this manner, the APU allows for controlling costs associated with hardware re-spins and recertification, giving avionics developers room to grow their product lines and/or build multiple products on common hardware platforms.

A dual-core Cortex-R5 Real-time Processor Unit (RPU) complements the APU, supporting complete isolation of the most critical tasks or precise real-time tasks. The dual RPU cores are capable of being run in lockstep, with full ECC on data cache and tightly coupled memory, and parity on instruction cache. These cores can be run up to 600MHz. With up to 1.67 DMIPS/MHz per core, 1,000 DMIPS is achievable in a single core or in lockstep use, or 2,000 DMIPS using both cores. The RPU can run critical code completely autonomous from the APU, sharing no resources, or the RPU may be used to run the critical code while offloading processes to, or checking processes that are run on, the APU. Design of the device with power gating and power islands offers flexibility for the avionics developer while

maintaining the ability to achieve safety and security in the end system.

Independent "configuration security" and "power management" units enable designers to securely boot and control the device to maintain safety and security. Fine-grained power gating control and dynamic power management allows for various approaches to satisfy certification of airborne systems. The configuration security unit additionally offers increased error detection and signaling capabilities to control safe states, resets, and error management. Encryption and anti-tamper features are also designed into the MPSoC.

The processing system also includes an on-chip memory and integrated Double Data Rate (DDR) memory controller, a Graphics Processor Unit (GPU), and a rich peripheral set. The memory system consists of a DDR controller capable of supporting 32-bit and 64-bit DDR interfaces with ECC, as well as a 256 Kbyte on-chip memory with ECC support. A Mali-400 MP GPU is added for applications with embedded graphic user interfaces or applications with 2D and 3D graphics display needs. The GPU would require additional assessment for certifiable applications, with special treatment for OpenGL or OpenVL software drivers. A number of common peripherals are hardened in the processing system, including PCIe, gigabit Ethernet, other serial connectivity interfaces, and nonvolatile memory interfaces.

The processing system detailed above is connected to a programmable logic fabric that looks a lot like an independent FPGA device. However, the programmable logic on the MPSoC benefits from direct connection to the hardened processing subsystem through multiple on-chip high performance parallel AXI interconnects. The direct on-chip connection provides a significant performance boost compared to devices connected at the circuit board level by PCIe, SRIO, etc. The Zynq MPSoC should not be considered as merely an FPGA. It is really a COTS heterogeneous multicore processor hardened in logic, with memory-mapped connections to fully customizable peripherals, co-processors, or some combination of the two.

The programmable logic varies in logic cell density, block memory, UltraRAM, Digital Signal Processing (DSP) blocks, PCIe cores, video codec unit, high-speed transceivers, and I/O count based on the chosen device within the Zynq MPSoC line-up. The processing system remains constant across all current devices in the family. Common package and pin-out options allow scaling device capability for a single target board, containing costs through development of scalable, platform-based end-system solutions. Performance, power, and cost can be easily scaled for mixed feature sets that target different airborne platforms. Specific avionics connectivity peripherals such as ARINC664p7/AFDX™, ARINC 818, ARINC 429, MIL-STD-1553, etc. are all optional peripherals that can be incorporated in the MPSoC through programmable logic and connected directly to any of the processing cores.

A hypervisor with TrustZone support may not only be used to partition cores but it can also control channels of communication between the APU and customized peripherals or soft-core co-processors that are resident in the programmable logic. Optionally, the user may isolate the RPU and APU and allow some customized programmable logic functions to connect only to the RPU. A number of system configurations are possible and the fine grain control of the device hardware allows flexibility to meet system safety, security, and performance goals while supporting certification objectives.

In December 2014, Xilinx began an engagement with the MCFA (Multi-Core For Avionics) consortium to provide a channel between Xilinx and the avionics industry to address common certification challenges, specifically on the Zynq MPSoC. During a March 2015 kickoff meeting, Xilinx hosted several leading avionics companies as well as Xilinx Alliance Members that are focused on addressing the avionics certification challenges on MPSoC. It is now the view amongst many involved in the community that certifying multicore solutions will require addressing hardware and software certification objectives from a system-level view rather than independently.

Hypervisor as a Future Approach

Future avionics will depend on improvements to hardware, software, and the integration of the two. In this section, we examine one promising direction for software: virtualization via an embedded hypervisor.

Safety, Security, and Performance

Safety and performance have long been a focus of flight electronics. In the last two decades, avionics for larger aircraft have contained costs via the consolidation provided by IMA. Safety of flight for IMA systems has been guided by DO-297, DO-248C (especially section 4.14 "Partitioning Aspects in DO-178C/DO-278A") and by ARINC 653, which specifies time and space partitioning for safety-critical avionics.

Early approaches to IMA have sometimes given up performance in order to maintain rigorous partitioning, such as flushing the cache before each partition begins execution. The industry has continued to refine and optimize trade-offs of performance in order to meet safety objectives, but we must now add additional objectives related to security. While safety and security needs often result in aligned design decisions, this is not always the case. Security is sometimes sought by introducing obscurity, which can be at odds with a safety process that requires clear visibility into the inner workings of the system. Security is sometimes sought by introducing randomization, which can be at odds with a safety process that requires deterministic, predictable behavior. Beyond the design decisions, the lifecycle processes dictated by safety certification are mostly aligned with those dictated by security. That is, both strive to provide design assurance – evidence that the design correctly meets the requirements. "Although the detailed requirements of safety and security acceptance are often different, sufficient commonality is visible in the acceptance processes to encourage us to seek cost savings by eliminating duplicate effort. Certifiers need to be presented with convincing, objective arguments that the system has the safety and security properties that are required. The methodology therefore must be based on the presentation of direct arguments, and supporting evidence, that systems have the necessary properties and behavior, and don't have any undesirable properties, to be safe and secure." [32]

Achieving the trifecta of safety + security + performance will be a challenge for our design processes and practices. One software technology that is aligned with multicore hardware is partitioning achieved through the virtualization technology of the hypervisor.

Virtualization and the Hypervisor

Virtulization is the idea of abstracting hardware so that it appears as a virtual (rather than physical) service to software. The system software (or possibly hardware) mechanism that manages and enforces this abstraction is called a Virtual Machine Monitor (VMM), or more commonly, a hypervisor, illustrated in Figure 4. While an operating system manages applications, a hypervisor manages operating systems. It provides virtual containers (called domains or partitions), with each one serviced by a virtual machine, so that multiple guest operating systems can run on a single computing platform containing one or more processor cores. A hypervisor manages the hardware on behalf of the software, preferably in as simple a way as possible. Consequently, hypervisors do not usually contain the more sophisticated features of a typical operating system, such as file systems, network protocol stacks, and graphical user interfaces. These services, which are often distinctly different between various OS vendors, are offered by the guest operating systems in partitions, not by the underlying hypervisor. This allows the hypervisor to host a variety of OS's on a single hardware platform. For example, the Green Hills Microvisor can run guests such as Microsoft Windows, Linux, and Android [33].

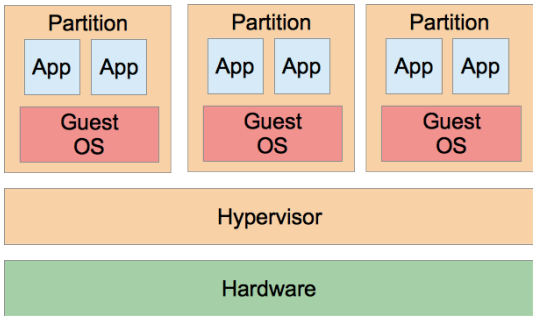

Figure 4: Partitioning Via Hypervisor

Hypervisors provide separation and isolation of virtual machines. This separation characteristic is enforced by hardware mechanisms such as timer hardware that can interrupt the current thread of execution on the CPU, as well as a memory management unit (MMU) that can segregate pages of memory. These hardware mechanisms must be guarded, typically via hierarchical protection rings, so that partitioning cannot be subverted by the hosted guest operating systems or applications.

A hypervisor is a natural fit for providing the partitioning needs for safety under ARINC 653 in an IMA system. Virtualization provides time and space partitioning implicitly. On the other hand, the specific Application Programming Interface (API), communication mechanisms, health monitoring, and other services dictated by ARINC 653 are not common in most hypervisors and must be developed or added [34]. These extensions must be added at multiple levels of the architecture, as illustrated in Figure 5.

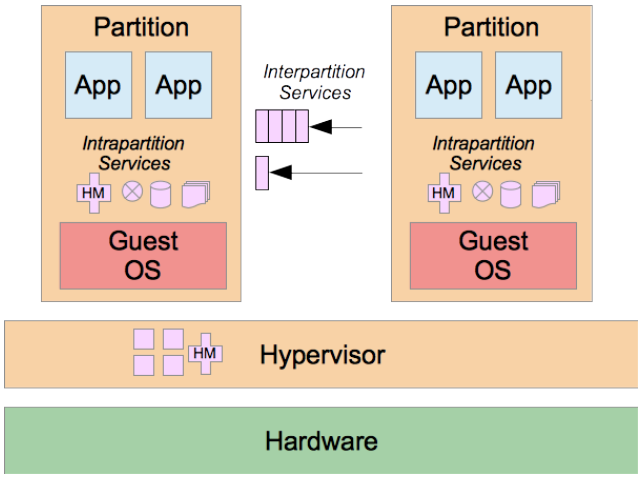

Figure 5: ARINC 653 Extensions to Hypervisor

Bieber suggests reconfiguration as a means to address faults, something a hypervisor might provide via live migration of Virtual Machines: "Reconfigurable IMA should be able to change the configuration of the platform by moving applications hosted on a faulty computing module to spare computing modules." [27] The idea is to leave spare VM capacity in reserve, in case a fault is detected in an active partition (Figure 6). When the system detects the fault, perhaps via a Health Monitor (HM) facility, it then moves (or restarts) the partition in the spare VM, as shown in Figure 7. The challenge with this approach is that certification for such a system must provide evidence of correctness for each possible configuration, both for normal operation and all possible fail over re-configurations.

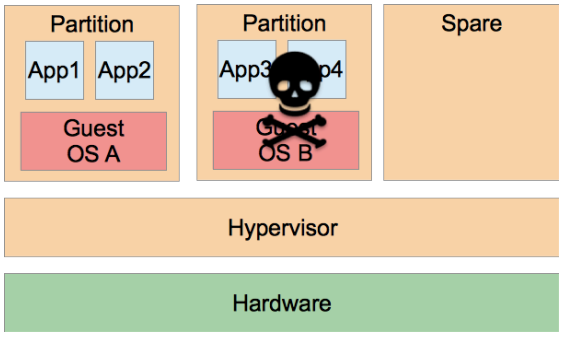

Figure 6: Partition Experiences Fault

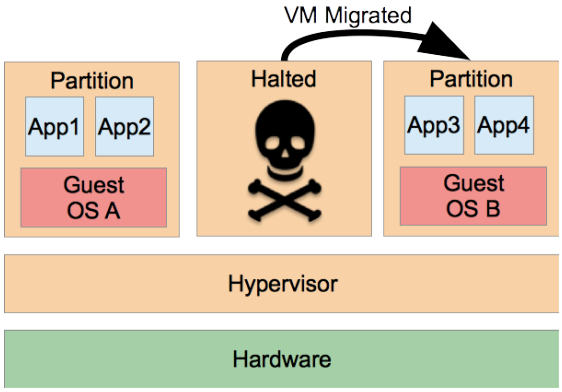

Figure 7: Fail Over via VM Migration

The isolation of hypervisor partitions is a natural fit for security needs as well, reducing the likelihood of covert side channels illicitly passing information from one security domain to another. Multiple Levels of Independent Security (MILS) typically depends on a separation kernel, which "is similar to the 'partitioning kernels' used in integrated modular avionics (IMA), but is more aggressively minimized in order to achieve higher assurance (an avionics kernel will typically be upwards of ten thousand lines of code, while DO-178B Level A approximately corresponds to EAL 5)." [14] EAL is the Evaluation Assurance Level of the Common Criteria, a standard for security with levels that range from a low of 1 to a high of 7. The highest two levels are only attainable if formal methods analysis are utilized.

Case Study: Hypervisors for Zynq MPSoC

For a case study, in this section we explore the architecture of several hypervisors that vendors are porting to the new Zynq UltraScale+ MPSoC. By reviewing these techniques, it offers insight into methods that could be used for avionics.

Xen Hypervisor

Xen is an open source hypervisor used in much of the world's cloud computing (such as Amazon Web Services). It not only supports x86 architectures, it has also been making its way into the embedded world with early support for PowerPC, but more recently for ARM architectures. In Xen terminology, the virtual machine partitions are called domains. Xen keeps the hypervisor layer fairly thin by delegating much of the configuration and I/O to the domains, particularly the privileged system domain called dom0. For example, dom0 configures the partition schedule on the cores and then uses hypercalls to inform the hypervisor, which implements and enforces it.

For I/O virtualization, a device that is used exclusively by a single partition might be memory-mapped directly to the partition, so that the device driver is hosted there, rather than in the hypervisor. This is termed "passthrough". The ARM System Memory Management Unit (SMMU) on the new Zynq MPSoC will allow such mappings.

For a shared device that must be arbitrated between multiple partitions, Xen uses a split driver model. The API of the driver is presented to the guest OS in the partition, providing a top-half wrapper to the underlying interpartition communication to dom0, which hosts the bottom-half of the driver that communicates with the actual device and provides the arbitration logic. Figure 8 shows the split-driver approach. The unprivileged guest domain (domU) contains only the wrapper logic to allow communication with dom0. The privileged system domain, dom0, contains the bulk of the driver logic, including the block communication with other partitions, arbitration logic, and the code to communicate with the I/O hardware.

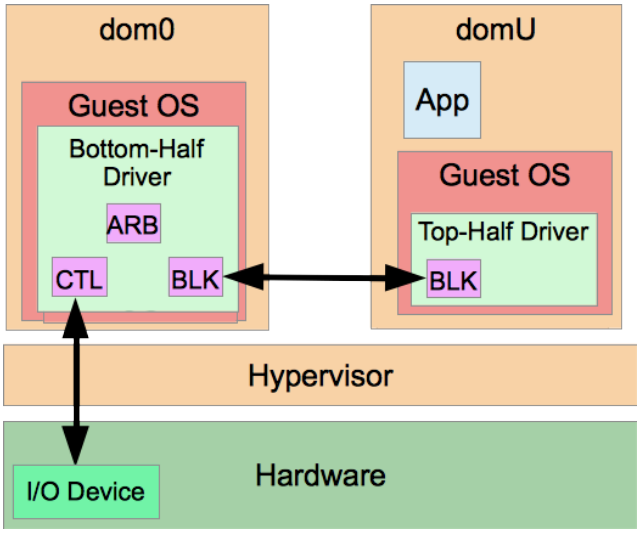

Figure 8: Xen Split-Level I/O Driver

The split driver approach illustrates an underlying philosophy that aligns well with the information security principle of least privilege. Specifically, by pushing functionality to the layer of least privilege still sufficient to its task, there is a benefit for safety and security, since the layers with most privilege (such as the hypervisor kernel) will require the most rigorous design assurance. Keeping those high privilege layers as small as possible is thus a means of cost containment.

Xen manages the quad-core Cortex-A53 on the Zynq MPSoC, as shown in Figure 9. The system partition, dom0, establishes the mapping of guest partitions to a schedule on one or more of the four available CPU cores. Xen uses the concept of CPU pools, allowing different subsets of cores to be defined. Partitions are assigned to virtual CPUs, and these vCPUs are assigned to a pool, each of which has a configured schedule. Xen also allows CPU "pinning", where a physical core is mapped exclusively to a single partition. Flexibility in scheduling may be useful to allow an avionics system integrator to ensure that a single safety-critical partition run within a VM is assigned all cores. By preventing other partitions from running simultaneously on other cores, we simplify analysis of resource interference, such as multiple VMs vying for L2 cache or memory bus bandwidth. An exception could be made for dom0, provided it only services I/O and other system needs for the running partition and no others.

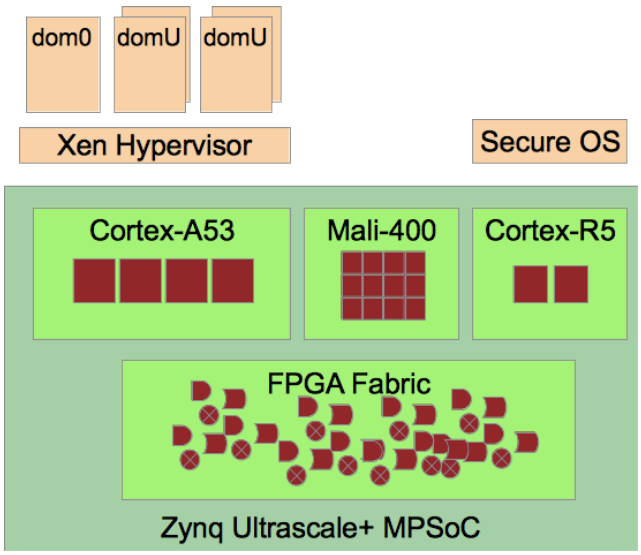

Figure 9: Xen on the Zynq MPSoC

The hypervisor controls access to other chip resources, such as the Mali-400 GPU or the FPGA fabric, though not necessarily as an intervening layer, such as when the SMMU provides direct access to a resource. As with scheduling of partitions, dom0 configures I/O access and then the hypervisor implements and enforces it. For example, dom0 could grant a single guest partition exclusive (passthrough) access to the GPU and map it directly into the memory space of that guest. Similarly, an I/O device, such as Ethernet or USB could be mapped directly to a guest for exclusive access. What if more than one partition needs access to the same I/O resource? One way to arbitrate this access is in software, via the split-driver model discussed earlier. However, if the software approach is too slow, limiting bandwidth or latency, hardware-based arbitration may be possible, implemented in the FPGA fabric, presenting a partitioned I/O resource that dom0 can individually map into the guests. Certification of the arbitration logic must be done to a design assurance level equivalent to the highest criticality of any of the serviced guests, whether implemented in software certified in accordance to DO-178C or in hardware certified in accordance to DO-254.

LynxSecure Separation Kernel Hypervisor

Lynx Software Technologies plans to port their LynxSecure Separation Kernel Hypervisor from the current x86 processor family to the ARM architecture, including the new Zynq MPSoC. This hypervisor is designed with the intent to certify safety up to DO-178 Level A and to certify security up to Common Criteria EAL-7 [35]. Therefore the kernel is kept as small as possible, with much of the functionality pushed up to the partitions, in order to reduce the level of effort for certification. For example, communication between virtual machines (Inter-VM) is performed via a shared memory area configured at initialization time, according to the system-defined security policies. They keep IMA in mind by providing an ARINC 653 scheduler. Figure 10 illustrates the architecture of the LynxSecure virtualized environment. Their Secure Virtual Device Server allows for users to place their own drivers in the system. If no RTOS is needed in the partition, an application can run "bare metal" using the LynxSecure Application (LSA) minimal C run-time library.

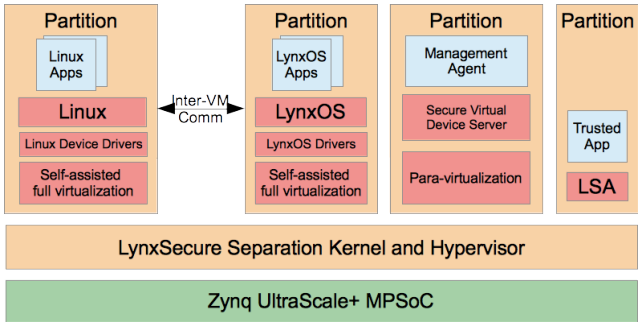

Figure 10: LynxSecure Hypervisor

Mentor Graphics Multicore Framework

Mentor Graphics has been working with a number of customers that are transitioning to heterogeneous multicore devices like the Zynq MPSoC from Xilinx. These clients struggle to manage the complexity offered by the silicon and different ways one can separate and isolate software blocks from each other in these consolidated systems. As a result, they have developed the Mentor® Embedded Multicore Framework [36] – a comprehensive set of runtime environments and tools that allow developers to configure and deploy multiple operating systems and applications across homogeneous or heterogeneous processors, whether device requirements call for native, trusted, or supervised system configuration. For example, with the Zynq MPSoC, one could run multiple instances of Linux and/or an RTOS natively on top of a hypervisor on the Cortex-A53 cores, while also

managing RTOS or bare metal applications on the Cortex R5 cores, as shown in Figure 11.

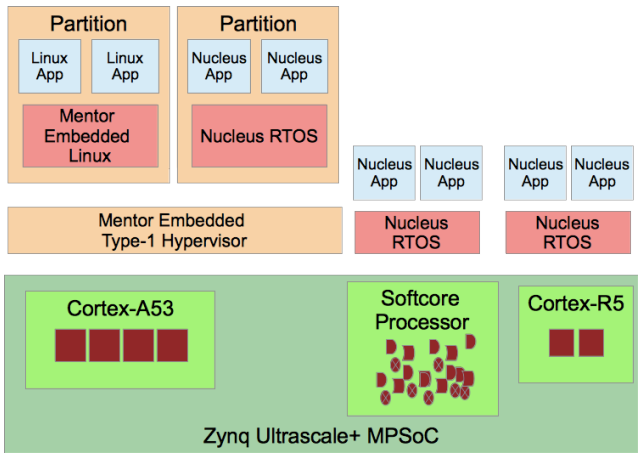

Figure 11: Multicore Framework

When running operating systems natively, one has to perform duties that in supervised configuration are handled by the hypervisor such as booting the cores, partitioning the resources, loading and unloading operating systems and applications, communication between tasks, operating systems, and cores just to name a few. To do so, Mentor has expanded the features and capabilities that were already present in many Linux distributions, such as *virtIO* for I/O mapping, *rpmsg* for signaling, and *remoteproc* for task management. Mentor extended these constructs to make them work not only within one Linux runtime environment, but across multiple operating systems.

Although the Multicore Framework is a runtime environment that executes on MPSoC, the story is not complete without corresponding tools. To help developers analyze resource contention and timing issues, Mentor Sourcery Code Bench is integrated with runtimes, allowing developers to visualize applications and operating systems running natively or on top of a hypervisor in a single common time-frame, thus providing a system-wide view of component behavior, shared resources usage, and possible contention.

Conclusions

In this paper, we have reviewed programmable multiprocessor system-on-a-chip hardware, hypervisor software to manage that multicore hardware, and used the forthcoming Zynq UltraScale+ MPSoC as a case study.

Multicore is the only feasible path forward to improve processor performance and thus must be accommodated by the avionics safety and security communities as has been done in many other domains. Separation provides safety and security by isolating independent functions and providing well-defined and characterized channels for interaction between those functions. Separation reduces the likelihood of covert side channels (security breach) as well as reducing chances of unanticipated consequences or failures (safety violation). In order to realize the benefits of IMA, separation must be clearly demonstrated. The validation of separation requires developing an argument that spans hardware and software and thus vendors of subsystems must cooperate with the system integrator (and perhaps with each other) in order to provide coordinated evidence of safety and security properties. IMA security and safety properties of hardware and software cannot be considered in isolation – the whole system must be certified.

Acknowledgements

The authors wish to thank David Beal of Xilinx, Will Keegan of Lynx Software Technologies, and Felix Baum of Mentor Graphics for their advice on this paper.

References

[1] B. Liu, E. Blasch, Y. Chen, A. J. Aved, A. Hadiks, D. Shen, G. Chen, "Information Fusion in a Cloud Computing Era: A Systems-Level Perspective," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 29, No. 10, pp. 16 – 24, Oct. 2014.

[2] Howard, Courtney E., 2013, "Intelligent Avionics," *Military & Aerospace Electronics*, v24, n2.

[3] Selinger, Marc, 1 Sep 2013, "Passenger Connections," *Avionics Today*, http://www.aviationtoday.com/av/issue/feature/Passenger-Connections_79939.html .

[4] Holmes, Mark, 13 Apr 2015, "Passenger Experience Conference Talks Increasing Connectivity ROI Through Operational Efficiencies," *Avionics Today*, http://www.aviationtoday.com/av/topstories/Passenger-Experience-Conference-Talks-Increasing-Connectivity-ROI-Through-Operational-Efficiencies_84774.html.

[5] Black, Randy, and Mitch Fletcher. 2004, "Next generation space avionics: a highly reliable layered system implementation," in *Digital Avionics Systems Conference, IEEE/AIAA 23rd,* IEEE.

[6] Alves-Foss, Jim , Paul W. Oman, Carol Taylor, W. Scott Harrison, 2006, "The MILS architecture for high-assurance embedded systems" *Int. J. of Embedded Systems* Vol. 2, No.3/4, pp. 239 – 247.

[7] Ramsey, James W., 1 Feb 2007, "Integrated Modular Avionics: Less is More," *Avionics Today*, http://www.aviationtoday.com/av/commercial/Integrated-Modular-Avionics-Less-is-More_8420.html.

[8] Erwin, Sandra I., January 2015, "Military Challenged to Maintain Decades-Old Aircraft," *National Defense*, p. 20.

[9] Lastera, Maxime, Eric Alata, Jean Arlat, Yves Deswarte, David Powell, Bertrand Leconte, and Cristina Simache, 2011, *Characterization of Hypervisors for Security-Enhanced Avionics Applications*, No. 2011-01-2805, SAE Technical Paper.

[10] Crespo, A., *et al.*, "Multicore Partitioned Systems based on Hypervisor," August 24-29, 2014, "Multicore partitioned systems based on hypervisor," *19th World Congress The International Federation of Automatic Control*, Cape Town, South Africa.

[11] Campagna, Salvatore, and Massimo Violante, 2012, "On the Evaluation of the Performance Overhead of a Commercial Embedded Hypervisor," *First Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale* (MEDIAN'12), pp. 59-63.

[12] Gidado-yisa, I. and E. Johnson, 26-29 September 2006, "State-Based Scheduling of Real-Time UAV Flight Control Avionics Tasks," Infotech@Aerospace.

[13] Sutterfield, B., 2008, "Future Integrated Modular Avionics for Jet Fighter Mission Computers," *Digital Avionics Systems Conference. IEEE/AIAA 27th*, IEEE.

[14] Boettcher, Carolyn, Rance DeLong, John Rushby, and Wilmar Sifre, 2008, "The MILS Component Integration Approach to Secure Information Sharing." *Digital Avionics Systems Conference. IEEE/AIAA 27th*, IEEE.

[15] Kinnan, Larry M., 2009, "Use of Multicore Processors in Avionics Systems and its Potential Impact on Implementation and Certification," in *Digital Avionics Systems Conference*, *IEEE/AIAA 28th*, IEEE.

[16] Thales Avionics, 12 Dec 2012, "MULCORS - Use of Multicore Processors in Airborne Systems," Research Project EASA. 2011/6.

[17] Sha, Lui, Marco Caccamo, *et al*., 2014, "Single Core Equivalent Virtual Machines for Hard Real-Time Computing on Multicore Processors." Technical Report, 2014. https://www.ideals.illinois.edu/bitstream/handle/2142/55672/SCE_Magazine_v15_LONG.pdf

[18] Wilhelm, Reinhard, *et al.*, October 2009, "Designing Predictable Multicore Architectures for Avionics and Automotive Systems," *Workshop on Reconciling Performance with Predictability*, Grenoble, France.

[19] Cullmann, Christoph, Christian Ferdinand, Gernot Gebhard, Daniel Grund, Claire Maiza, Jan Reineke, Benoıt Triquet, and Reinhard Wilhelm, 2010, "Predictability Considerations in the Design of Multi-Core Embedded Systems," *Proceedings of Embedded Real Time Software and Systems*, pp. 36-42.

[20] Kliem, D. and Voigt, S., 2012, "A Multi-core FPGA-based SoC Architecture with Domain Segregation," *ReConFig*.

[21] Awan, M., 2015, "Towards Certifiable Multicore-based Platforms for Avionics," *21st IEEE Real-Time and Embedded Technology and Applications Symposium*.

[22] Abella, J. *et al*., 2011, "Towards Improved Survivability in Safety-critical Systems," *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, IEEE.

[23] Huyck, Patrick, 2012, "ARINC 653 and Multi-core Microprocessors – Considerations and Potential Impacts," *Digital Avionics Systems Conference*, *IEEE/AIAA 31st,* IEEE.

[24] Gaska, Thomas, Brian Werner, and David Flagg, 2010, "Applying Virtualization to Avionics Systems – The Integration Challenges," *Digital Avionics Systems Conference, IEEE/AIAA 29th*, IEEE.

[25] Parkinson, Paul, 2011, "Safety, Security and Multicore," in *Advances in Systems Safety,* Springer London, pp. 215-232.

[26] McDermott, John, *et al.*, 2012, "Separation Virtual Machine Monitors," *Proceedings of the 28th Annual Computer Security Applications Conference.* ACM.

[27] Bieber, P., *et al.,* 2012, "New Challenges for Future Avionic Architectures," *Aerospace Lab,* v1, n10.

[28] Mims, C., 12 Oct 2010, "Why CPUs Aren't Getting Any Faster," *MIT Technology Review*, http://www.technologyreview.com/view/421186/why-cpus-arent-getting-any-faster/

[29] FAA, May 2014, "Multi-core Processors," Certification Authorities Software Team (CAST) Position Paper CAST-32.

[30] Duren, Russ, 2006 "Options for upgrading legacy avionics systems." *Journal of Aerospace Computing, Information, and Communication* v3, n6, pp. 251-259.

[31] Guzmán-Miranda, H., *et al*., 2011, "Coping with the Obsolescence of Safety- or Mission-Critical Embedded Systems using FPGAs," *Industrial Electronics, IEEE Transactions on* 58.3, pp. 814-821.

[32] Lautieri, S. *et al.,* 2005*,* "SafSec: Commonalities between Safety and Security Assurance," *Constituents of Modern System-safety Thinking*, Springer London, pp. 65-75.

[33] Green Hills Software 2015, "Integrity Multivisor," http://www.ghs.com/products/rtos/integrity_virtualization.html.

[34] VanderLeest, Steven H., 2010, "ARINC 653 Hypervisor," *Digital Avionics Systems Conference, IEEE/AIAA 29th,* IEEE.

[35] Lynx Software Technologies, 2015, "LynxSecure: Software Security Driven by an Embedded Hypervisor," http://www.lynx.com/pdf/LynxSecureDatasheetFinal.pdf.

[36] Mentor Graphics, 2015, "Multicore Framework," http://s3.mentor.com/public_documents/datasheet/embedded-software/multicore-framework-ds.pdf.