Applying MILS to multicore avionics systems

Paul J. Parkinson
Principal Systems Architect
Wind River
Swindon, United Kingdom
Paul.Parkinson@windriver.com

***Abstract*— The implementation of the Multiple Independent Levels of Security (MILS) software architecture on modern microprocessor architectures has become technically feasible in recent years. This allows MILS-based systems to host applications and data of multiple security classifications concurrently on a uniprocessor platform at affordable cost. In this paper, the potential requirements for the implementation of a separation kernel to support MILS systems on multicore processor architectures will be considered, and the design challenges associated with its potential implementation on the NXP (formerly Freescale) QorIQ™ P4080 multicore processor will be discussed. Finally, the potential use of a MILS Multicore separation kernel in two use cases will be presented - a Cross-Domain System (CDS) network gateway, and a Multi-Level Secure (MLS) Integrated Modular Avionics (IMA) platform.**

Keywords—MILS; multicore; security; MLS; CDS; ADN

I. ADOPTION OF MULTIPLE INDEPENDENT LEVELS OF SECURITY

Historically, commercial organizations and governments have categorized information at different security classifications, based on varying criteria including information value, sensitivity, and the impact of disclosure.

Information at different security classifications was traditionally physically isolated in separate domains. The methods used to enable authorized information flows between security domains have varied, but have often involved manual transformation of information which has fundamentally limited the speed of analysis of information and decision-making.

More recently, there has been a drive towards automation of the information flow process between different security domains. This enables decision-making to be accelerated, in order to provide benefits to applications as diverse as commercial business and banking operations, through to sharing information with coalition forces in theatre operations.

Initially, these multilevel secure computer systems were built using multiple, physically separated computers, networks, and displays. This technique, known as "air gap" security, required expensive equipment and occupied a large footprint in terms of Size, Weight and Power (SWaP). Whilst there have been efforts to address the multi-level security requirement through the development of monolithic, secure operating systems running on a single computing platform, their development and security certification would have taken ten or more years and at unaffordable cost due to the large size and complexity of the trusted computing base (TCB) [1].

In 1984, John Rushby proposed an alternative approach for secure embedded systems, utilizing a small trusted computing base as part of a layered software architecture, providing separation between different domains on a single processor [2]. This provided the foundation for the Multiple Independent Levels of Security (MILS) software architecture which was presented by Mark Vanfleet at the Open Group Security Forum in 2002. The MILS architecture provides a means of overcoming the development time and certification cost issues associated with large monolithic operating systems, through the use of a separation kernel (SK) built on four fundamental security policies:

- *Information Flow*. This defines the permitted information flows between partitions.
- *Data Isolation*. This ensures that a partition cannot access resources in other partitions.
- *Periods Processing*. This ensures that applications within partitions execute for the specified duration in the system schedule.
- *Fault Isolation*. This defines that a failure in one partition does not impact any other partition within the system.

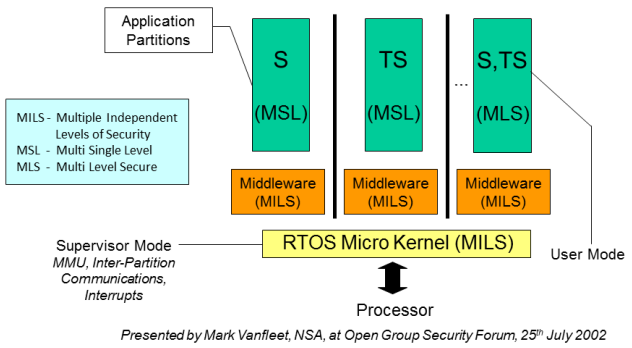

Fig. 1. Multiple Independent Levels of Security (MILS) architecture

These four policies create an architecture where the separation kernel is *Non-Bypassable*, *Evaluatable*, *Always Invoked* and *Tamper Proof*, which is known as *NEAT*. This means that the size of the trusted computing base implemented by the separation kernel, shown as "RTOS Micro Kernel (MILS)" in Fig. 1, has the potential to be very small, as all

other functionality can be migrated either to middleware or applications running in partitions. This layered assurance approach means that the amount of code which needs to undergo rigorous scrutiny as part of a high assurance security evaluation can be very small when compared to a traditional monolithic operating system.

In 1984, the microprocessors available on the commercial market did not provide sufficient performance to host multiple applications concurrently or the ability to separate them in a robust manner. However, recent advances in some commercial microprocessors have included hardware virtualization support which can be utilized by the implementation of a separation kernel using a *Type 1* (or native, bare metal) *Hypervisor* [3]. This has enabled the development and deployment of MILS systems on uniprocessor architectures [4].

II. Advent of Multicore Processors

A. Disruptive Technology

There has been a long-term trend in processor development, described by Moore's Law [5], whereby the transistor count that can be placed on an integrated circuit increases at an exponential rate, doubling every two years. This trend is expected to continue for a number of years yet, although there is disagreement about when the limit for miniaturization of transistors will be reached. However, this approach has approached the limit of viability because as processor clock frequencies have climbed, power consumption has soared.

The result has been a *disruptive change* to the processor evolutionary development trend through the introduction of multicore processor devices. Multicore processors combine two or more independent processor cores into a single integrated circuit (IC) which can each be run at lower clock frequencies, resulting in lower overall power consumption whilst still achieving increased overall performance.

B. Multicore Software Paradigms

The parallelism of different types of applications can vary enormously. For example, some numerically-intensive applications are inherently sequential, whereas other applications such as radar, sonar or image processing are often inherently parallel. This variation has led to the development of a range of multicore software configurations (Fig. 2).

Symmetric Multiprocessing (SMP) involves the use of a single instance of an operating system running across multiple processor cores. This has traditionally involved the operating system (OS) running across multiple single-core processors, but the advent of multicore processors has resulted in a trend to consolidate a systems architecture comprising multiple, single-core processors into a single, multicore processor-based architecture, thereby reducing SWaP footprint.

Asymmetric Multiprocessing (AMP, or sometimes referred to as ASMP) uses a different approach to SMP, by treating each core as a separate processing element. This enables multiple instances of the same application to be replicated across multiple cores and to operate on separate sets of data. In addition, it is also possible to run different operating systems on different cores if required. A typical, multicore, embedded system may run a real-time operating system (RTOS) on one core for real-time sensors or control, and a general-purpose operating system such as Linux on another core to provide a graphical user interface. Multicore AMP systems can be extremely complex, especially in consolidated systems where multiple operating systems are used. This means that additional effort needs to be expended on ensuring that the system configuration allocation of hardware resources to individual guest operating systems is correct. However, the effectiveness of this approach without use of a supervisor is debatable, as it is reliant on the co-operation of the individual guest operating systems.

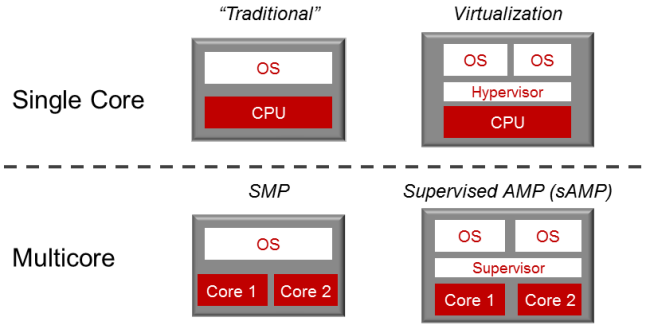

Fig. 2. Primary multicore software configurations

Virtualization extends the concept of supervision further, through the use of a hypervisor running across the cores. The hypervisor is responsible for creating a virtualized environment for each core, configuring the memory protection required for each OS, and loading and booting the relevant operating systems. This provides strong separation between the operating environments on each core, and typically could be used to enable the consolidation of applications, which previously ran on separate processors, onto a multicore architecture; for example, this could involve a general purpose OS such as Windows or Linux, and an RTOS such as VxWorks.

These primary multicore software configurations can each be applied to different types of computing problems, and are becoming widely deployed due to the widespread commercial availability of multicore processors. For a technical discussion of the respective architectures and challenges for their deployment, refer to [6].

III. MILS & Multicore Convergence

The increasing deployment of the MILS-based systems on single-core processors, and the increasing commercial availability of multicore processors, has led to the convergence of these two technology trends, resulting in the requirement to extend the MILS architecture to exploit multicore performance security-critical systems.

This requirement is now becoming more compelling as the number of single-core processor shipments continues to decline as multicore processors become more widespread; in fact,

some semiconductor companies may manufacture single-core processors as dual-core processors but with the second core disabled, in order to achieve manufacturing efficiencies.

A. Architectural Challenges

There are some significant challenges in relation to using multicore processors in MILS systems, in particular, that of *application isolation* and *core separation*. Research into these areas has been undertaken in recent years in the related discipline of safety-critical systems, and although the end-goals of safety certification and high-assurance security are different, there is some overlap in their requirements.

For example, research undertaken by the FAA [7] has highlighted issues around suitability of particular multicore processor architectures for safety-critical avionics applications. The FAA research highlighted the consequences of incorporating shared resources in a specific multicore processor architecture, the NXP (formerly Freescale) PowerPC MPC8572 dual-core processor. In particular, application performance and determinism on one core was found to be significantly adversely affected by the contention for shared resources by the actions of the other core, resulting in a denial of service (DOS).

Whilst the impact of this contention can potentially be mitigated for some safety-related avionics applications through careful application design and rigorous worst-case execution time (WCET) analysis [8], the use of shared resources such as memory controllers and Level 2 (L2) and Level 3 (L3) caches on multicore architectures presents a greater challenge which is unique to security, that of covert channels of information transfer. In single-core processor architectures, it may possible to limit the bandwidth of a covert channel due to a shared resource through the design of the MILS separation kernel partition scheduler. However, on a multicore architecture, as the applications may be executing simultaneously on individual cores, the bandwidth of covert channels could potentially increase dramatically, making them much more dangerous.

Although, the FAA and EUROCAE have not yet published formal policies on the use of multicore processors in safety-critical avionics applications, position papers such as CAST-32 [9] and MULCORS [10], published in both US and Europe respectively, have described new objectives for multicore safety. In addition, the ARINC653 [11] specification for integrated modular avionics (IMA) systems was updated in 2015 to support the development of multicore IMA systems.

Consideration must be given to the fact that most multicore processor architectures are designed to support a wide range of multicore software configurations (as discussed earlier), rather than specifically for high-assurance security applications. Some individual processor variants, however, within a multicore processor architecture family may provide specific features which make them more suitable for use in a MILS system. An example of this is the NXP QorIQ P4080 communications processor [12], which has eight e500mc cores, each with a private L2 cache, which eliminates one of the potential covert storage channels.

The P4080 architecture also employs the CoreNet Fabric to provide separation between the cores whilst enabling multiple memory accesses to be performed in parallel, avoiding the contention issues associated with a traditional bus architecture. This provides a powerful capability which can be utilized in a MILS multicore implementation to reduce the bandwidth of a potential covert channel between specific applications.

However, the P4080 processor only has two memory controllers and a single L3 cache which is shared by all eight cores, which means that if more than two cores are executing (non-cached) code, there is potential for contention for access to physical memory, thus providing a covert channel. Disabling the L3 cache can help to reduce the bandwidth of the covert channel.

Finally, in a MILS multicore environment, there is also a requirement to be able to boot each of the cores in a secure manner; and the ability to achieve this may be determined by the capabilities provided by the underlying processor architecture.

B. MILS Multicore Design & Implementation Considerations

The potential requirements for a separation kernel to support MILS on a multicore processor architecture, and a specific implementation on the NXP QorIQ P4080 processor, will be considered in the following subsections.

1) MILS Multicore Software Architecture

The primary high-level requirement is to enable high-assurance security applications to run securely on a multicore processor architecture. There are a number of potential software architectures which can be employed, based on the primary multicore software configurations discussed earlier; but an approach based on Asymmetric Multiprocessing (AMP) may provide better security characteristics and enables the reuse of a MILS separation kernel implementation for a single-core processor (also referred to as *unicore*) to the greatest extent possible.

A processor using 32-bit addressing provides a 4 Gbyte address space. Whilst this may be sufficient for running a separation kernel on a single processor, and also on a single core on the P4080, it would not be sufficient if the 4 Gbytes had to be shared between eight cores (i.e., only 512 Mbytes per core). This can be solved by migrating a 32-bit separation kernel implementation to a 64-bit implementation, but this would add complexity and cost just to provide increased address space.

Instead, this issue can be overcome by using the P4080, as its CoreNet interconnect fabric supports 36-bit addressing, providing the processor with a 64 Gbyte address space, whilst employing 32-bit addressing per individual core, with the processor memory management unit (MMU) enforcing separation between the address spaces of individual cores. This 32-bit compatibility helps to minimize the scope of changes required to port an existing unicore separation kernel implementation to the P4080 multicore architecture.

For these reasons, the decision was taken to implement VxWorks MILS Platform, Multi-core Edition [13] on the P4080 processor using AMP mode, where each core (and its

associated hardware resources) hosts an independent instantiation of VxWorks MILS. Thus, a separate unicore VxWorks MILS system, complete with separation kernel and applications running in partitions, runs on each core.

2) Core Virtualization

Microprocessors typically provide two privilege levels, *User Mode*, for user application context, and *Supervisor Mode*, which allows the execution of privileged instructions, which is usually reserved for use by the operating system kernel. The introduction of a secure hypervisor running beneath the partitions containing applications and operating systems results in a requirement for a third privilege level.

The P4080 implements three privilege levels, including a *Hypervisor Mode*, which enables a guest operating system (GOS) running in a partition to have memory protection from the applications running on top of it, without relying on the separation kernel for enforcement, as this is performed by hardware (Fig. 3). This means that Guest OSes running in partitions do not require significant paravirtualization, which is required on processors which do not provide hardware virtualization support.

Therefore, the full hardware virtualization support of the P4080 reduced the effort required to implement guest OS support on VxWorks MILS Platform, Multi-core Edition.

3) Secure Boot

The use of a multicore processor architecture greatly increases the complexity of the initialization and boot sequence for the platform. This leads to a requirement to ensure that the processor is placed into a known state before each core is booted, in order to ensure that security issues are not introduced through initialization and boot of multiple cores being performed in the wrong order.

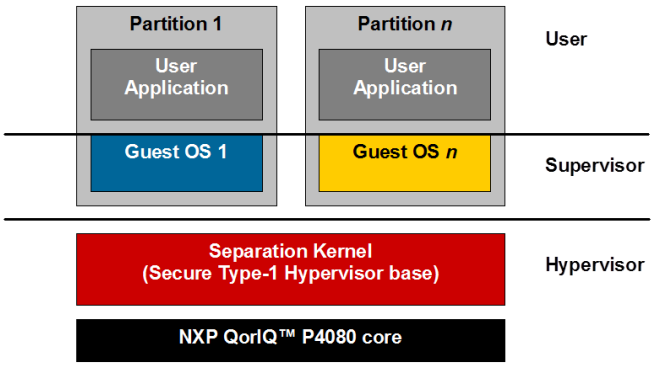

Fig. 3. MILS Hypervisor Virtualization Model

The exponential growth in complexity of a multicore system increases the risk that the system architect may configure the system incorrectly, so automated tool support is required to provide continuous validation and feedback to prevent the introduction of system configuration errors which could lead to security issues.

This capability was implemented in VxWorks MILS Platform, Multicore Edition using XML configuration files which are compiled into binary objects and linkable images using the Wind River XML configuration generation tools, including XML Configuration Verification (VerCon-X) and Configuration Vector Generation (CVGEN). These configuration images are then loaded on the target to be referenced during system boot. These XML configuration tools allow developers to make changes to application and system configuration information without rebuilding and retesting the entire system.

4) Core Schedule Synchronisation

MILS unicore systems provide the ability to handle information at multiple security classifications or from multiple security domains, as the MILS time-slice scheduler enforces the *sequential* processing of individual applications or domains; i.e., an individual time-slice within the MILS system schedule allows only one partition to execute within that time-slice duration.

In a MILS multicore system, each core executes its own MILS separation kernel independently, which means that it also can execute its own independent schedule. This provides the potential for individual cores to be processing information at different security classifications, or from different domains, concurrently: i.e., at the same instant in time. This *independent time-sliced scheduling* may be permissible for a MILS system where there are multiple independent applications processing information at the same security classification or from the same security domain.

However, for some MILS multicore systems, if the underlying processor multicore architecture provides any resources which are shared between cores, these could potentially be used by one core to monitor or disturb performance on another core, providing a potentially high-bandwidth covert storage channel which could be exploited to perform the unauthorized flow of information between cores.

Although the P4080 provides full separation of cores through separate memory address ranges, separate L2 caches, and the CoreNet interconnect fabric (which together provide better overall separation than some other processor architectures), the P4080's cores still share a number of hardware resources, such as the two memory controllers for use with the eight P4080 cores.

Therefore, there is a requirement to be able to minimize these potential covert storage channels to limit their bandwidth and impact in MILS multicore systems where this would be a security issue.

This requirement could be implemented through the use of a *synchronized time-sliced scheduling model*, which enforces synchronization of core schedules. This approach would enable the system architect to control the cores as if they were a single entity, executing at a single security level. By synchronizing schedules across cores, the system architect can ensure that at any instant of time, all data being processed on all cores are all at the same security level, or from the same domain, to minimize the impact of any covert channels between partitions on different cores. The use of Synchronized Time-Sliced Scheduling does not remove the potential for covert channels, but does eliminate illicit cross-level information flow. This will be illustrated through analysis of a use case in a later section.

5) Inter-Core Communications

There is also an important requirement to be able to communicate between the cores securely, to enable an application hosted in a partition on one core to communicate with an application running on another core. This should be implemented in a manner which is transparent to the application hosted in the partition. This location transparency would provide benefits in terms of software portability as it enables the system architect to modify the design and move applications to different cores without impacting the application implementation.

The implementation approach taken in VxWorks MILS Platform, Multi-core Edition was to extend the secure inter-partition communication (SIPC) component, previously implemented in the VxWorks MILS unicore implementation, in a location transparent manner suitable for use in the multicore implementation. This also reuses an API designed around the ARINC 653 APEX queuing and sampling port interfaces, which provides a programming environment familiar to ARINC 653 developers, as well as enables ARINC 653 applications to be ported to VxWorks MILS Platform.

6) Independent Core Configuration

A MILS multicore system is extremely complex, and to enable the successful configuration of the entire platform, an approach which allows individual cores to be configured independently is required. This would also enable a MILS multicore system to be composed incrementally, and would enable the configuration and composition of the system to be reordered in a way which fits in with a MILS development organization and internal processes.

This could be achieved through the use of XML-based configuration of individual cores as well as the entire MILS multicore system, using automated tool support to continuously validate the configuration and provide feedback to the system architect. This was the approach that was taken for the implementation of VxWorks MILS Platform, Multicore Edition, again using the XML configuration and tool capabilities described in requirement 3) above.

IV. Cross-Domain System using MILS Multicore

In this section, the requirements for a Cross-Domain System (CDS) network gateway will be considered, and potential implementation approaches using MILS unicore and MILS multicore will be discussed.

A. Cross-Domain System Requirements

A typical use case for a Cross-Domain System (CDS) requiring multiple levels of security is a network gateway between networks of different security classifications or security domains. This could be deployed as a secure gateway in a wide range of difference scenarios, for example between a corporate enterprise network and the public internet, or between government or military networks of different security classifications. In each of these scenarios, the generalized use case and topology is the same (Fig. 4).

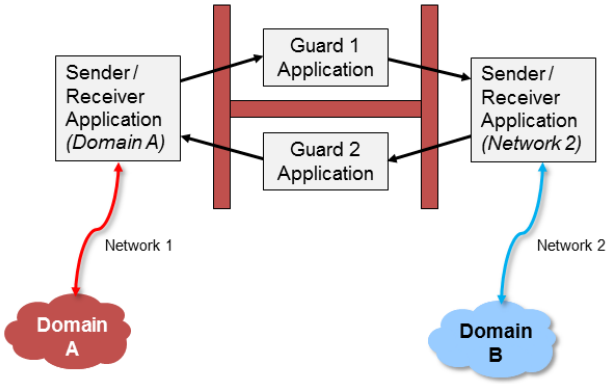

Fig. 4. Cross-Domain System network gateway

The CDS will be connected to two different networks, and should have applications which receive messages from one network and pass them to a guard, which determines whether to forward the messages based on a pre-defined security policy and examination of the messages' data contents, or to discard the messages. A similar configuration with a separate guard (and potentially a different security policy) will operate in the reverse direction.

B. Cross-Domain System, MILS unicore implementation

In a MILS unicore implementation, this could potentially be implemented using a minimum of four partitions: sender/receiver for Domain A (App_1), guard for Domain A, sender/receiver for Domain B (App_2), and guard for Domain B. The applications would be isolated from each other by the MILS separation kernel, and only information flows according to the system security policy would be allowed (Fig. 5).

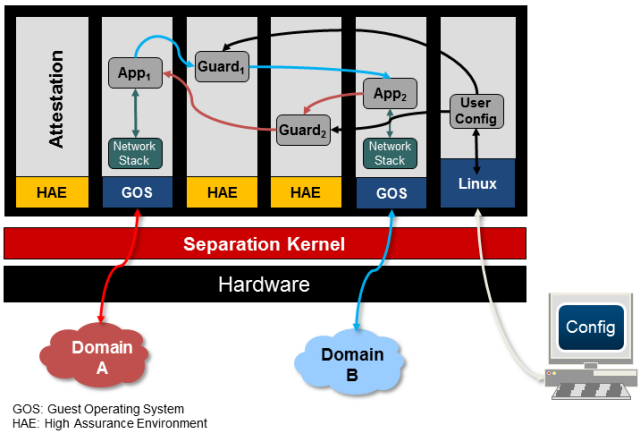

Fig. 5. Notional MILS-based Cross-Domain System network gateway

The sender/receiver applications, App1 and App2 respectively, which would each have their own instance of a network stack which would use a separate Ethernet interface mapped exclusively into each partition's address space. The sender/receiver applications would also be responsible for transformation of data between network interface and the pre-

defined unidirectional SIPC channels to/from the guard partitions, thus enabling protocol break between networks to protect against protocol-based attacks.

In a typical system, this would be augmented with additional partitions which would be used for attestation and system configuration, respectively.

The "Attestation" partition would be used to run some of the abstract machine tests (AMT, colloquially, "built-in test" or BIT) to validate parts of the SK in relation to the secure state requirement.

The "User Config" partition would enable the system to be dynamically configured by a Trusted User for different security policies depending on the mode of operation, which would be enforced by the guard applications. The integration of the User Config partition eliminates the need for an additional, external computer for this purpose, and would communicate with the guard partitions via SIPC to enable the guards to select between pre-defined security policies statically configured within the guard partitions.

The system could execute on a single processor using the MILS unicore *time-sliced scheduling*. This would allow each partition (and corresponding application) to execute sequentially for a pre-defined duration, before the next partition is scheduled. In this configuration, the maximum network throughput in both directions is determined by the relationship between the time taken to process an individual message and the duration of the individual time-slices (minor frames). In addition, the latency of transferring a message from one network to another is determined by the length of the major frame (overall sequence of partition time-slices). The potential for covert storage channels may be relatively low (depending on the processor architecture), as individual applications will run consecutively and never concurrently. The potential for covert timing channels may be greater, but these may be mitigated by proactive steps to limit their bandwidth.

C. *Cross-Domain System, MILS multicore implementation*

A MILS multicore implementation may provide many more potential options for system implementation approaches and also the potential for higher system performance throughput (when compared to a MILS unicore implementation), due to the ability to run applications concurrently on separate processor cores.

However, the system will need to be architected carefully to ensure that the potential for covert channels of communication is minimized. This could be achieved by allocating the applications to different processor cores and using the Synchronized Time-Sliced Scheduling approach to limit the number of security levels or domains being processed at the same time (Fig. 6).

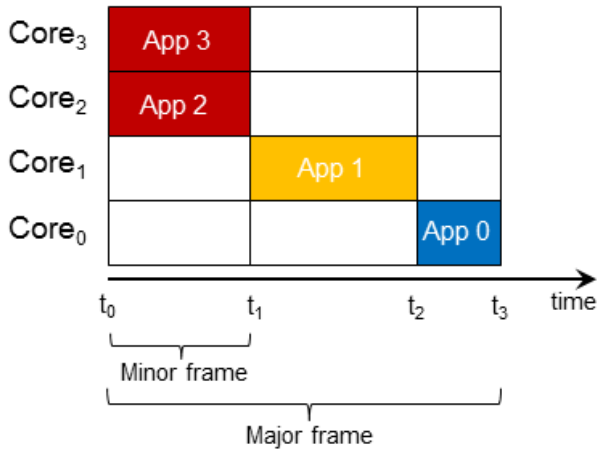

Fig. 6. Non-overlapping synchronised time-sliced schedule

This means that if a covert channel is exploited, then only information at the same security classification or from the same domain will be exchanged. The system architect would also have the ability to define separate schedules with varying overlaps of application times-slices, and provide a trusted application with the ability to switch from one pre-defined schedule to another pre-defined schedule, depending on the system's mode of operation or current level of threat.

V. MULTI-LEVEL SECURE IMA SYSTEMS USING MILS UNICORE

In this section, the requirements for Multi-Level Secure (MLS) Integrated Modular Avionics (IMA) platforms will be considered, and potential implementation approaches will be discussed in relation to two use cases: using a MILS unicore implementation and using a MILS multicore implementation.

A. *Multi-Level Secure IMA System Requirements*

Many IMA systems have been successfully deployed based on the ARINC 653 [11] software architecture. This enables multiple applications to run at different levels of safety-criticality on the same processor. However, ARINC 653 does not explicitly address the requirements for supporting multiple applications at different security classifications on the same processor, known as Multi-Level Secure (MLS).

An example in civil avionics of consolidating multiple safety-critical applications onto the same IMA processor is a satellite internet communications system, which would provide cockpit voice communications via broadband satellite internet connection as a backup to cockpit very high frequency (VHF) radio systems. As part of the airworthiness security process defined by DO-326 [14], this system would need to ensure that a potential external security threat via the satellite communication link could not subvert the safety-critical functions hosted on the IMA platform.

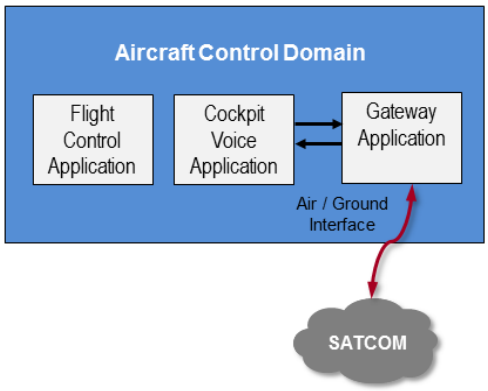

Fig. 7. Multi-Level Secure Civil IMA platform

Similarly, a semi-autonomous unmanned combat air vehicle (UCAV) that takes off in civil airspace before requesting a vector into uncontrolled airspace would need to communicate with civilian air traffic control (ATC) without disclosing classified mission data, and would therefore have an MLS requirement when using an IMA platform.

A simplified representation of the civil avionics use case is shown in Fig. 7; the topology for the UCAV use case is conceptually similar – in both cases one or more IMA applications need to be securely isolated from the avionics application connected to an external network.

For further background discussion on the convergence of safety, security and multicore, refer to [6].

B. *Multi-Level Secure IMA, MILS unicore implementation*

In a MILS unicore implementation, the satellite internet communications system could potentially be implemented using a minimum of four partitions: flight control application, cockpit voice application, guard for data received from the satellite communications network, and network gateway application.

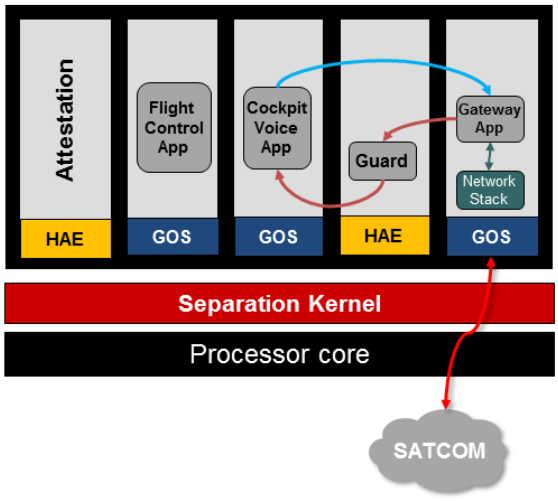

Fig. 8. Notional Multi-Level Secure IMA platform, MILS unicore

These applications would be isolated from each other by the MILS separation kernel, and only information flows according to the system security policy would be allowed. In a typical system, this would be augmented with additional partitions, such as for attestation (Fig. 8).

C. *Multi-Level Secure IMA, MILS multicore implementation*

Alternatively, this use case could be implemented using a MILS multicore architecture (Fig. 9). This could exploit the additional cores for increased performance throughput (as in the case of *Cross-Domain System, MILS multicore implementation* discussed earlier), by running the flight control application on an instance of the MILS separation kernel on a dedicated core, and the cockpit voice satcom applications on a separation kernel on a different processor core. Again, the potential security impact of covert channels could be minimized through the use of synchronized schedules.

In addition, if the system were implemented on the P4080 processor, the two cores could be mapped to use different memory controllers, to avoid the potential risks associated with shared bus contention and resource starvation associated with some multicore processor architectures.

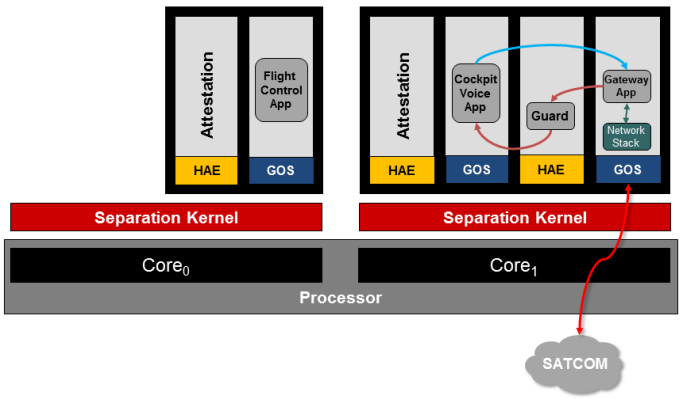

Fig. 9. Notional Multi-Level Secure IMA platform, MILS multicore

VI. CONSOLIDATION OF IMA DOMAINS USING MILS MULTICORE

In this section, the requirements for consolidation of avionics domains will be considered, and a potential Multi-Level Secure (MLS) implementation approach using MILS multicore will be discussed.

A. *IMA Domains Consolidation System Requirements*

The systems architecture of modern civil aircraft consists of multiple networks (Fig. 10) which are used for a diverse set of functions, including: (a) flight-safety related control and navigation systems (Aircraft Control Domain); (b) airline business and administrative support (Airline Information Services Domain); Passenger entertainment (Passenger Information & Entertainment Services Domain); and (d) Passenger Owned Devices Domain (PODD) [15].

The systems connected to these networks have different requirements in terms of safety and security. Firewalls or

secure gateways are used between the different aircraft data networks (ADN) to implement isolation and enforce authorized information flows. Alternatively, a data diode may be used to restrict data flows to one-way communication, but this can increase the complexity of system design, as TCP uses bidirectional communication, and unidirectional UDP/IP communication may not be suitable for some types of applications.

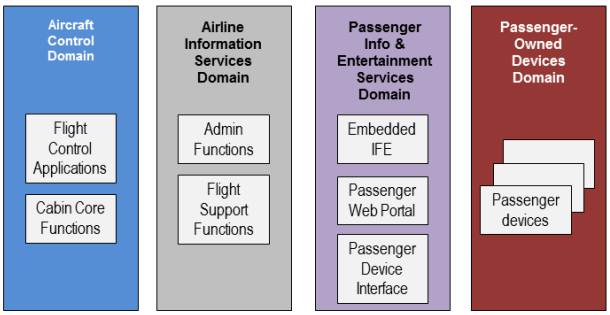

Fig. 10. Aircraft Data Network domains

Increased passenger demand for in-flight internet connectivity, and competition between aircraft manufacturers and airlines to provide best-in-class service, are driving the requirements for increased performance throughput and reduced latency of communication between to passenger-owned devices and the Passenger Information & Entertainment Services Domain systems, whilst maintaining domain security.

As evidenced by the increasing use of composite airframe materials in lieu of heavier steel and aluminum, there is increased emphasis on the part of aircraft manufacturers to reduce both fuel consumption and greenhouse gases emissions by reducing the weight of aircraft. The physical constraints of lack of spare cabin space for additional line replaceable units (LRUs) to support passenger devices will only increase SWaP pressures even further. This problem could be resolved through the consolidation of multiple IMA domains using a MILS multicore system architecture.

B. IMA Domains Consolidation , MILS multicore implementation

This use case could potentially be implemented using a MILS multicore architecture, hosting the Airline Information Services Domain and the Passenger Information & Entertainment Services Domain on separate P4080 cores (which is notionally similar to the dual domains in Fig. 10.) The Passenger Information and Entertainment Services Domain MILS system could be replicated across multiple cores in order to segregate passenger cabins if required. Any spare cores could be reserved to provide additional processing capacity in case of increase in SATCOM data link bandwidth in the future.

VII. CONCLUSION

The commercial realization of the Multiple Independent Levels of Security software architecture in recent years has enabled the development of high-assurance multi-level secure systems at affordable cost. The advent of multi-processor architectures has caused a technology disruption to programming models. The capability to merge the MILS software model on a state-of-the-art multicore processor architecture provides the potential to implement high-assurance multi-level secure systems with increased performance throughput (when compared to unicore), though with additional, potential security threats that need to be addressed.

ACKNOWLEDGMENT

The author wishes to thank the following Wind River colleagues for their input into this paper: P. Chen, C. Constantinides and T. Preyssler.

DISCLAIMER

This paper is intended to keep interested parties informed of the subject matter herein for educational purposes only. This paper is protected and subject to copyright law. The author and Wind River Systems, Inc. (“Wind River”) are not responsible for any errors or omissions in the content of this paper or for damages of any kind arising from the reliance, use or performance of any of the contents of this paper under any circumstances. This paper is provided "AS IS" and without any warranty of any kind, expressed or implied.

REFERENCES

[1] J. Alves-Foss (University of Idaho), B. Calloni, (Lockheed Martin), M. Dransfield (NSA/IAD), J. Luke (AFRL), L. MacLaren (Boeing), G. Uchenick (Objective Interface Systems), WM. Vanfleet (NSA/IAD), “Enabling the GIG”, PowerPoint presentation, MILS community, 2004-2005.

[2] J. Rushby, “A Trusted Computing Base for Embedded Systems”. Proceedings 7th DoD/NBS Computer Security Conference, Gaithersburg, Maryland,p294-311, 24-26th September 1984. http://www.csl.sri.com/users/rushby/abstracts/ncsc84-tcb.

[3] G. Popek, R. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures". Communications of the ACM 17 (7): 412 – 421, 1974. http://dl.acm.org/citation.cfm?doid=361011.361073

[4] P. Parkinson, A. Baker, “A High Assurance Systems Development using the MILS Architecture”, Wind River technical white paper, November 2010. http://www.windriver.com/whitepapers/high-assurance-systems-development-mils-arch/.

[5] G. Moore, “Cramming more components onto integrated circuits”, Electronics, Volume 38, Number 8, IEEE. 19th April 1965.

[6] P. Parkinson, “Safety, Security and Multicore”, Advances in System Safety, Proceedings of Nineteenth Safety-Critical Systems Symposium, Springer, 8-10th February 2011. http://link.springer.com/chapter/10.1007%2F978-0-85729-133-2_13.

[7] R. Mahapatra, J. Lee, N. Gupta, and R. Manners, “Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 5 Report”, DOT/FAA/AR-11/5, US Federal Aviation Administration, May 2011. http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/11-5.pdf.

[8] C. Cullmann, C. Ferdinand, G. Gebhard, D. Grund, C. Maiza (Burguière), J. Reineke, B. Triquet, R. Wilhelm, “Predictability Considerations in the Design of Multi-Core Embedded Systems”, Embedded Real-Time Systems & Software conference, 19-21 May 2010. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.172.4533.

[9] CAST-32, “Multicore Processors”, position paper, Certification Authorities Software Team, US Federal Aviation Administration, May

2014. https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-32.pdf

[10] "MULCORS – Use of MULticore proCessORs in airborne systems", Research Project EASA.2011/6, European Aviation Safety Agency. http://easa.europa.eu/system/files/dfu/CCC_12_006898-REV07%20-%20MULCORS%20Final%20Report.pdf

[11] ARINC Specification 653P0, "Avionics Application Software Standard Interface, Part 0, Overview of ARINC 653", ARINC, 3rd August 2015. http://store.aviation-ia.com/cf/store/catalog_detail.cfm?item_id=2496.

[12] "P4080 QorIQ™ Multicore Communication Processor Reference Manual", Rev. 2, NXP (formerly Freescale), May 2014. http://www.nxp.com/products/microcontrollers-and-processors/power-architecture-processors/qoriq-power-architecture-processors/qoriq-p4080-p4040-p4081-multicore-communications-processors:P4080?fpsp=1&tab=Documentation_Tab.

[13] "VxWorks MILS Platform 3.0, Multi-Core Edition", product note, Wind River. http://www.windriver.com/products/product-notes/0739-MILS-3.0-Multi-core-Edition-Product-Note.

[14] DO-326, "Airworthiness Security Process Specification", RTCA Inc., December 2010. http://www.rtca.org/store_product.asp?prodid=1057.

[15] W. Cecil, "Safely Connecting AIS & PIES Domains – Approaches and benefits of shareing Aircraft Networked resources", Aviation Electronics Europe, 26th March 2015.

