USING SPACEWIRE IN A SECURELY PARTITIONED COMPUTING ARCHITECTURE

Session: SpaceWire Missions and Applications

Long Paper

Peter Mendham

SciSys UK Ltd, Clothier Road, Bristol, BS4 5SS, UK

Knut Eckstein, James Windsor

ESA/ESTEC, 2200 AG Noordwijk ZH, The Netherlands

E-mail: peter.mendham@scisys.co.uk, knut.eckstein@esa.int, james.windsor@esa.int

ABSTRACT

SciSys is leading an ongoing ESA study into the development of an embedded systems software architecture which provides the capability to partition multiple applications in a safe and secure manner. This architecture targets future dual-use spacecraft shared by multiple payload developers and operators. We consider the requirements placed on a SpaceWire onboard communications system by such an architecture in terms of spatial partitioning of data and temporal partitioning of shared resources such as communications links. The resulting discussion elicits concrete requirements on both the hardware and software of the onboard SpaceWire elements.

1 INTRODUCTION

SciSys is leading an ongoing ESA study into the development of an embedded systems software architecture which provides the capability to partition multiple applications in a safe and secure manner. As a baseline reference architecture, the study considers a dual use spacecraft with a single onboard computer handling both platform and payload operations with the latter performed by separate partitioned applications.

Such a securely partitioning architecture places a number of requirements on an onboard communications system in terms confidentiality and integrity of data transmitted on shared resources such as communications links. As an increasingly popular communications technology, the applicability of SpaceWire to such a future system architecture is crucial, and the ability of a SpaceWire communications architecture to meet the requirements is the focus of this paper.

Section 2 introduces the concept of *Time and Space Partitioning* (TSP) and its applications to onboard processing. Whilst TSP has typically been applied for reasons of safety and ease of development and integration, we discuss the implications of applying TSP in an environment where security needs must also be met. The following three sections elicit concrete requirements on both SpaceWire hardware and software. In Section 3, a hardware-focussed analysis lists mechanisms by which current SpaceWire hardware interfaces may be utilised by a securely partitioned computing platform, and presents key concepts for future consideration. Link- and network-level partitioning in routers is considered in section 4, for example time-scheduling, such as in SpaceWire-(R)T. In Section 5, a complementary analysis discusses the role of software services, such as SOIS, in a securely partitioned system. The paper concludes by summarising the central role that SpaceWire can play in a securely partitioned spacecraft architecture.

2 Time and Space Partitioning for Security and Safety

The concept of TSP in software systems has been established for some time, and has seen successful application in a wide variety of domains, including avionics, automotive systems, enterprise servers and handheld mobile devices.

2.1 Time and Space Partitioning

TSP is a technique which permits the sharing of a computing platform between multiple independent applications. *Spatial partitioning* indicates the division of shared resources, such as memory, which may be utilised by multiple applications simultaneously. Spatial regions such as memory address ranges can be limited to exclusive access by one application, or shared access by multiple applications can be granted. *Temporal partitioning* indicates the division of shared resources, such as a simple processor, which cannot be utilised by multiple applications simultaneously. Such resources must be wholly 'owned' by a single application at any point in time, and are shared by multiplexing the access to the resource by applications in time.

Through suitable enforcement of temporal and spatial partition boundaries, TSP provides an integrated environment in which applications cannot interfere with each other. This can be used for many purposes, including the isolation of sensitive applications, and permitting applications to be developed, validated and potentially certified separately. A partitioned system is shown in Figure 1.

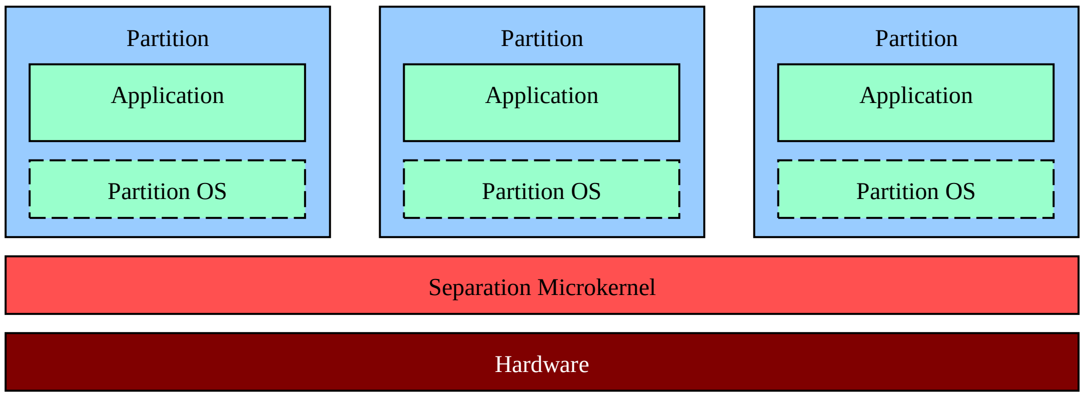

Figure 1: TSP Architecture

TSP is typically implemented using tight coordination between operating system (OS) and hardware features. A memory management unit (MMU) gives the OS an efficient mechanism to enforce spatial partitioning of mapped memory. A MMU may be used to protect memory regions from undesired read or write operations by applications which should not have access. Temporal partitioning is typically enforced through a periodic timer interrupt which cannot be blocked or intercepted by applications. The interrupt is handled by the OS, which is responsible for saving the state of temporally partitioned resources and preparing them for access by another application.

The Integrated Modular Avionics (IMA) architecture [1], deployed on recently developed commercial aircraft such as B787 and A380, heavily utilizes TSP. In more traditional avionics architectures each element of the system is provided on a number of dedicated hardware units. In contrast, IMA permits accommodation of the computing functions of several system elements in common computing hardware and the use of shared communication networks. Typically this reduces the weight of the system and costs in development, system supply and maintenance.

The space industry has a number of similar requirements to both the aviation and automotive industries, which has lead to the investigation of the potential for “spinning-in” IMA technology, using TSP, into the space domain with studies such as the ESA IMA for Space activity [2].

2.2 Partitioning for Security

As part of the ongoing Securely Partitioning Spacecraft Computing Resources activity (ESA Contract No. 22186/09/NL/LvH) led by SciSys, a study was conducted into the future security needs for primes and agencies. The study identified a wide range of scenarios in which security needs may affect onboard computing resources, such as multi-agency spacecraft, dual-use missions and assisting spacecraft manufacturers in meeting export regulations. In the reference architecture, a spacecraft features two imaging payloads: one standard resolution, to which access is not restricted; and one very high resolution payload, to which access must be restricted for commercial, legal or national security reasons. To ensure that the security needs for such a mission are met, the onboard systems must be able to guarantee confidentiality and integrity of information relating at different security levels. To reliably ensure that these security needs on a computing platform, the system designer has a number of options:

- A so-called “system-high” approach, operating the entire spacecraft, including all software applications, at the highest level of security required. This has major implications on the cost of system development and validation.

- Separate the elements of the system which correspond to different security levels onto separate hardware, limiting their interaction and adding a mass, power and volume penalties.

- Permit applications handling information at different security levels to be combined on a single computing platform by associating security information with each system entity (applications, devices etc.) and controlling access through security policies and mechanisms implemented in the operating system.

In the third approach presented above, the resulting multi-level security (MLS) operating system is typically large and complex, preventing assurance of its operation. A solution to this problem is to require the operating system to provide only reliable separation mechanisms; security levels and policy then become issues for applications and the operating system becomes simpler.

This solution is known as Multiple Independent Levels of Security (MILS) [3] and is, in many ways, similar to the TSP used in systems such as IMA, with the addition of security requirements. A system taking a MILS approach will be structured like that shown in Figure 1, with a small operating system, the separation microkernel, enforcing time and space partitioning, and applications in partitions, which may have their own operating systems.

This architecture shares many features with those of hypervisors and virtual machine monitors. The partition applications should be unaware of the other partitions on the system; in theory it should not matter if two partitions are executed on the same, or different, hardware platforms from each other. In a system where the largest concern is safety this separation is enforced to ensure a partition, either through normal operation or by malfunction, can not unintentionally influence the integrity of another partition. In a secure system this concept is extended to enforce both the integrity and confidentiality domains for the partitioned applications, i.e. it must not be possible to transfer information between two partitions, except where explicitly permitted by security policy. At the same time, the spectrum of statistical threats of malfunction and environmental influence is enlarged by the presence of a qualified, malicious human attacker. In the context of confidentiality, a *covert channel* is defined as any communication between partitions that is in contravention of the the system security policy. Two types of covert channels are typically considered: a *storage channel* involves the modification of a shared object, the state of which is used to transfer information; whereas a *timing channel* involves affecting the relative timing of observable events, such as the observation of the storage or timing events of one partition by another, which is used to impart information.

2.3 SpaceWire in a TSP System

Clearly, an onboard computer does not exist in isolation: it must interface to other onboard devices in order to receive inputs and produce outputs. SpaceWire is growing in popularity as an onboard communications medium, and is increasingly being used to interface onboard computers to both payload and platform devices. The effects of introducing SpaceWire into a securely partitioned system (i.e. one using secure TSP) fall into three categories:

- the interface between the onboard computer and the SpaceWire network, this may be part of a System-on-Chip (SoC) or it may be a separate device;
- the SpaceWire network itself, and what can be done to avoid the costs of creating two, or more, independent networks for different security levels;
- the communications software architecture, executing on the onboard computer, which is used by applications to interact with SpaceWire devices.

These topics are addressed in the following sections. As will be shown, SpaceWire is well suited for use in a securely partitioned architecture: a routed network is inherently more flexible and easier to secure than a shared medium bus such as MIL-STD-1553B or CAN.

3 SECURELY PARTITIONING THE SPACEWIRE INTERFACE

Where the SpaceWire network interfaces to the onboard processor, consideration must be given to the principles of TSP. Such considerations impact the way in which the SpaceWire interface(s) is/are to be connected to the processor, affecting address decoding and the use of interrupts and DMA.

As mentioned above, the MMU is the typical mechanism for enforcing spatial partitioning. In order to control access to interfaces, such as SpaceWire, all parts of the interface must be memory mapped. The use of alternate address spaces which are not controlled by the MMU, such as dedicated I/O spaces, should be avoided. Although the use of such spaces is typically restricted to a privileged processor mode, this requires the operating system to be involved in I/O transactions, as no application is permitted to run in a privileged mode, and increases its complexity. Where a system has multiple SpaceWire interfaces, it may be advantageous to permit these interfaces to be used by individual partitions, independently from one another. This requires that the memory-mapped resources associated with each interface are distinct (i.e. no shared registers) and that they fall into separate memory pages, so that access may be controlled with the MMU. The use of an MMU implies that an application sees only virtual, rather than physical, addresses. An interface, such as SpaceWire, however, will see physical addresses if it attempts a Direct Memory Access (DMA) transaction. Additionally, a contiguous region of virtual memory does not necessarily ensure a contiguous region of physical memory: it may therefore be difficult for a partition application to set up and manage DMA transactions without considerable operating system assistance. One way to address these issues is to introduce an I/O MMU, mapping virtual to physical addresses for devices.

Although these techniques ensure spatial separation, the use of DMA causes issues with temporal partitioning. A DMA transaction claims use of the memory bus, and access to the memory, which prevent access by the processor. Although the processor may be executing out of cache this cannot be guaranteed. Should a DMA transaction on behalf of one partition occur during the processor time allocated to a second partition, this may affect the safe operation of the second partition and also permit the second partition to observe one part of the first partition's operation. Without special provisions, therefore, DMA is not a safe technique in a TSP system, whether or not security is a concern. One way to permit the use of DMA is to utilise dedicated, dual-ported, DMA regions for each interface (see Figure 2). Access to these regions by the SpaceWire interface is largely independent to that of the processor and processor timing is not affected by DMA accesses. Buffer management operations, such as updating read and write pointers, need to be designed to permit concurrent access.

Just as an asynchronous (as far as processor execution is concerned) DMA transaction adversely affects temporal partitioning, so does the use of interrupts. In a similar manner to DMA transactions, the occurrence of an interrupt will change the execution flow of the processor and the timing of an executing partition. If hardware buffers are appropriately sized a minimum interrupt inter-arrival time can usually be assumed and with a suitable scheduling scheme, applications may meet their real time deadlines. However, such impacts on predictable partitioning are likely to be observable from unrelated partitions, creating an obvious covert channel. The only reliable way to mitigate this risk is to use polling to service interfaces.

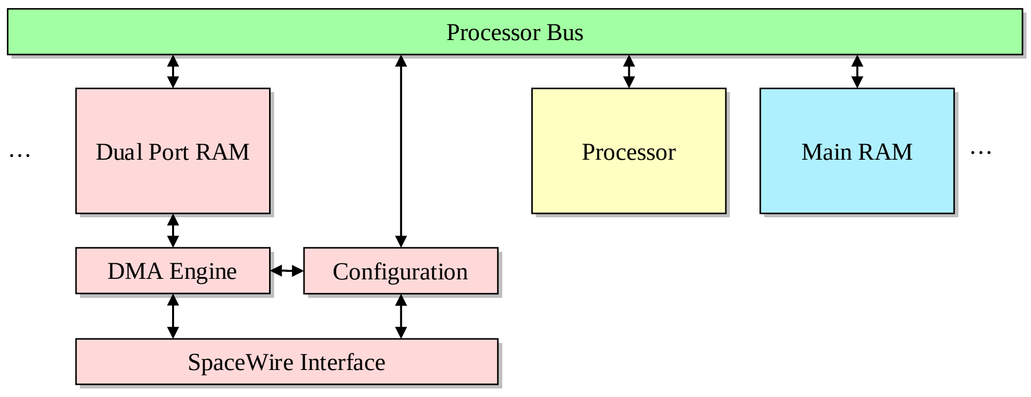

Figure 2: SpaceWire Interface using DMA on Dual-Port RAM

In the case of a SpaceWire interface, providing FIFOs are sufficiently sized for link transmit and receive rates, it should be possible to handle data transfers using polling. The handling of time-codes is potentially more complicated, and depends on the interpretation of time-codes in the system. If the arrival time of a time-code is critical, for example to update a local time counter, the handling of this may be best done without software intervention, as waiting for a polled response may introduce an unacceptable delay.

4 Securely Partitioning the SpaceWire Network

The most obvious way to separate SpaceWire resources corresponding to different security levels is to construct several distinct SpaceWire networks. These networks can be accessed from separate interfaces at the processor, using the guidance above. However, if these separate networks are any more that single links, this technique involves the duplication of routing resources, potentially resulting in mass, power and complexity penalties.

One way to safely partition a SpaceWire network is to restrict the destination address at the head of packets transmitted through an interface. Such a restriction may be implemented in either hardware or trusted software and involves only the first byte of the packet. Firstly, by restricting packets to use only logical addressing, it can be assured that the interface may not address packets to the configuration port of a router, or any other device. This ensures the safety and security of a configured network. Secondly, by restricting the logical addresses to which an interface may address packets, the network may be spatially partitioned along lines of function, security, or whatever suits the mission. This scheme directly associates logical addresses with security domains, and may require multiple logical addresses per node. This argument holds as long as logical addresses are never deleted by a router, so-called *regional logical addressing*. If this is not the case, and logical addresses are deleted, a further check must either be made at the interface (for the second logical address) or, more elegantly, at the entry point to this new region. The only interfaces that may configure the network are those permitted to use path addressing, which, having unrestricted access to all devices, must be fully trusted. The scheme outlined here permits the strict spatial partitioning of a SpaceWire network, requiring changes only to interface hardware and/or software, with no changes necessary to routers.

A spatially partitioned network is sufficient providing that no resources, i.e. devices or links, are shared between partitions. As routers are generally full crossbar devices, they may be shared between partitions without compromise. Where resources are shared, these must be temporally partitioned. As with the partitioning of processor execution time, to ensure both safety and security this network bandwidth partitioning must be *deterministic*.

Recent work has seen a number of proposals for the introduction of deterministic and/or low-latency traffic on SpaceWire networks, with the aim of handling time-critical command and control data. The SpaceWire-RT [4], SpaceWire-T [5] and SpaceWire-D [6] protocol proposals all share a common approach in that network bandwidth is time division multiplexed, with SpaceWire time-codes indicating the boundaries between time slots. This technique is deterministic and requires no change to network routers for its operation. Time slots are pre-allocated to interfaces by the system designer, such that the available bandwidth of any shared resource, including those shared between multiple interfaces, is not exceeded. As with the spatial partitioning case, this primarily relies on trusted network interfaces on all devices. The handling of non-trusted devices would require the enforcement of the network schedule either between device and router, or within a modified router.

Another proposal for the division of network bandwidth, utilises *virtual channels* to multiplex many channels of traffic over a single link, and by extension, an entire network [7]. The bandwidth of a link is given to whichever virtual channel has data to send and is of the highest priority. Low latency is assured by permitting high priority traffic to pre-empt low priority traffic. Whilst this scheme does successfully permit ad-hoc low latency traffic, without careful design low-priority traffic can be starved from a network. Furthermore, for secure partitioning purposes, the delivery of traffic of any but the highest priority is non-deterministic. This non-determinism may be accounted for in terms of system safety, but gives rise to the potential for covert channels, the risk of which is application dependent. The use of virtual channels, then, may not be sufficient on its own, requiring the introduction of time division multiplexing to ensure fairness and determinism. In this case the various virtual channels need not be treated hierarchically, but could be scheduled, for example, cyclically at routers. Whilst elegant and guaranteeing temporal partitioning, such a technique would require wholesale changes to both interfaces and routers.

As can be seen from the discussion in this section, SpaceWire routing resources form a critical part of a network. Existing routers may be utilised through the introduction of trusted software and/or hardware at every interface. This assumes that the target of the security effort is the TSP system, and that the potential complexity of such a trusted interface is sufficiently low to permit security assurance. If the network itself is to be secured against un-trusted devices, logical address and time slot verification functions would have to be incorporated into routers.

Trust in routing resources is key to the operation of a securely partitioned SpaceWire network. The configuration of routers has the potential to enforce spatial and, perhaps in the future, temporal partitioning. However, during the power up of a router, before it has been configured, it is potentially vulnerable to configuration by an un-trusted device. One way to prevent this issue is to associate a small configuration ROM with each router, from which a failsafe configuration may be loaded. This would “lock down" the network into a safe and secure state, pending configuration from a trusted source.

5 SECURELY PARTITIONING ONBOARD COMMUNICATIONS SOFTWARE

The previous sections have left open the issue of where, in a partitioned architecture, the software stack responsible for communications should be placed. In Section 3, the possibility for assigning SpaceWire interfaces to partitions was discussed; however, in Section 4, the introduction of sanity checks on destination logical addresses could require trusted interface software. This indicates a trade-off between a decentralised approach, in which the operations of partitions are self-contained and may be validated separately; and a more centralised approach in which a single, trusted, partition is responsible for managing the SpaceWire interfaces.

The latter approach was taken by SciSys in the Payloads with Resource-efficient Integration for Science Missions (PRISM) project. Here, the RTEMS operating system was modified to include temporal partitions, including the enforcement of both processor execution time and I/O bandwidth budgets. Including the concept of I/O partitioning in the system architecture permitted the I/O handling software to be located in a single system partition, and shared between other applications. Whilst a promising approach for integration and increased safety, the PRISM operating system is relatively complex and would be difficult to assure for security purposes.

The best way to reduce the potential assurance effort is to minimise the size of the trusted component. A good example is the application of the CCSDS SOIS stack []. Here, the SpaceWire subnetwork services could be associated with an individual interface, perhaps in a trusted partition, whereas the application support services would be placed in one or more un-trusted partitions. Especially where there is good hardware support, the SpaceWire subnetwork services may be relatively simple, increasing the potential for assurance. The SOIS architecture provides a good template for the decentralisation/centralisation split in the onboard communications architecture.

6 CONCLUSIONS

6.1 REQUIREMENTS FOR THE USE OF SPACEWIRE IN A SECURELY PARTITIONED ARCHITECTURE

As a routed network of flexible topology, SpaceWire has the potential to integrate well into a securely partitioned system. However, there are a number of challenges when applying secure TSP to SpaceWire: some are specific to SpaceWire and others less so.

At the interface level, consideration should be given to the potential for spatially protecting interfaces using the MMU. This requires memory-mapped hardware, laid out in consideration for page boundaries. Shared resources, such as the processor, must be deterministically shared in time and therefore cannot be disrupted by asynchronous events such as DMA or interrupts. The design of SpaceWire interfaces and driver software should account for this.

At the network level, it is possible to spatially partition a SpaceWire network using logical address verification at interfaces and careful network design. Deterministic temporal partitioning, such as that employed in SpaceWire-(R)T and SpaceWire-D, meets the requirements for TSP; whereas the use of virtual channels may not when applied alone. With the addition of time-division multiplexing it becomes more powerful. Simple network TSP can be introduced using trusted hardware/software at interfaces, and is sufficient if the focus of security requirements is a software TSP system. The introduction of un-trusted devices on a SpaceWire network necessitates modifications to router technology. To ensure router enforcement of partitioning, routers should ideally use a configuration ROM to ensure a trusted boot.

At the software level, where an interface must be trusted, and shared between multiple partitions, the trusted code should be minimised. The subnetwork layer in SOIS is a logical point to split the communications stack.

6.2 Looking to the Future

SpaceWire has the potential to play a central role in TSP architectures including securely partitioned onboard systems, providing that consideration is given to the topics presented in this paper. Some issues, such as those at the interface level, can be addressed in the short term, either by use of existing devices or minor modification to current IP. A SpaceWire network can be securely partitioned using current technology, providing all node interfaces are trusted. Relaxing this caveat requires modification to routers: as the arbiter of network traffic, routers are the appropriate place for the secure control of network bandwidth. For the current generation of SpaceWire devices such changes would be an addition or modification to standard behaviour; as thought is given to the next generation of SpaceWire technology, it is suggested that (secure) TSP should be carefully considered as it is likely to become an important technique in the design of spacecraft onboard systems.

7 References

1. Airlines Electronic Engineering Committee, 'ARINC Specification 651: Design Guidance for Integrated Modular Avionics', 1997.

2. Integrated Modular Avionics for Space. ESA ITT AO/1-6295/09/NL/LvH.

3. Alves-Foss, J. et al, 'The MILS Architectures for High Assurance Embedded Systems"' International Journal of Embedded Systems, 2(3/4), 2006.

4. University of Dundee, "SpaceWire-RT: Initial Protocol Definition', Version 2.1, Available from the minutes of the 12th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

5. Parkes, S., "SpaceWire-RT and SpaceWire-T', Available from the minutes of the 14th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

6. University of Dundee, "SpaceWire-D Protocol', Available from the minutes of the 14th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.

7. Cook and Walker, 'Virtual Networks', Available from the minutes of the 13th SpaceWire Working Group, ://spacewire.esa.int/WG/SpaceWire/, 2010.