Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	**D9.2 – Space Application Concept Report**	
Work package	WP9	LL_Space Application
Task / Use Case	T9.2	UC_MPSoC SW and Tools for Space
Subtasks involved	T9.1, T9.2, T9.5	
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto:werner.weber@infineon.com	
Deliverable responsible	TASI Lucia Amorosi, Lucia.Amorosi@thalesaleniaspace.com Dario Pascucci, Dario.Pascucci@thalesaleniaspace.com	
Version number	v4.0	
Date	04/05/2015	
Status	Final	
Dissemination level	Public (PU)	

Authors

Partici-pant no.	Part. short name	Author name	Chapter(s)
11J	TASI-UNIVAQ	Dario Pascucci	Chapter 1, 6;
15F	TASE-TECNALIA	Sanchez Renedo Manuel	Chapter 2, 3, 4 ;
15Q	ITI	Sergio Sáez, Javier Cano	Chapter 5 ;
01B	AICAS	James Hunt	Chapter 7.

Table of contents

List of figures

List of tables

1. Introduction

1.1 Objective and scope of the document

The ability to properly reuse scarce resources on a spacecraft can determine the success or failure of a mission. The important advances in hardware of the last decade are slowly, but inevitably, being adopted in the space domain: multiprocessor (LEON4), reconfigurable hardware, etc. The challenge is now how to exploit all the capabilities of this new hardware without jeopardising the certification process. The more static and fixed is the behaviour of the system the easier is to certificate it. Unfortunately, the ideas of dynamic reconfiguration or workload balance always plays against the certification due to the difficulty to cover and analyse all the cases. This living lab will analyse and address the potential blocking issues that may prevent the use of the new HW both at the HW level and in the interaction with the software development cycle of highly critical systems.

This Use Case will be focused on the software procedures and tools that will support the proposed MPSoC architectures for space application. The requirements for space software are very stringent and impose strong limitations that condition the selection of the different SW elements (hypervisor/virtual machines, Time/Space Partitioning OS, …).

Most hardware advances has a positive impact on performance, energy consumption, weight, etc. but not on the way the software is designed and developed. The same code can be compiled for the new hardware with minor, or not at all, modifications.

Unfortunately, this is not the case on multi-processor and re-configurable HW where the changes are so large that the software gets affected by the new hardware architecture and the compatibility with the space requirements is strongly compromised. In order to get all the benefits from the new hardware architecture, the software must be aware of the underlying facilities. The interactions between the SoC devices (memory affinity, bus access, contention on shared resources, etc.) has to be taken into account when the software elements (processes, threads and routines) are developed, allocated or dynamically re-adjusted to the computing resources.

Different OSs and tools will be evaluated and checked against the space standards in order to define an MPSoC toolset that guarantees the space worthiness of the different Software developed. The tool-set developments will also comprise TTEthernet tool plug-ins and tool extensions in order to connect the specific space related tool plug-ins to the standard TTEthernet tool environment. Furthermore specific drivers will be provided in the area of TTEthernet based endsystems supporting Sysgo Pike OS. These toolsets will be evaluated and recommended for the rest of the Use Cases.

2. Proposed Massive Data Processing MPSoC Architectures for Space

The proposed massive data processing MPSoC architecture allows to implement and to validate new signal processing algorithms for space applications. It has been designed to be used as a generic platform for the following space signal processing techniques:

- Regenerative transponders
- Adaptive air interface
- Interference cancellation/mitigation technologies
- High efficiency modulation/coding techniques
- Spot beam technology

An application based on sub-band signal processing will be implemented to demonstrate the ability to carry out a multiband analysis.

2.1 Sub-band signal processing application

Radiometers applications require identifying Radio Frequency interferences (RFI). A conventional technique to identify RFI is to use a filter bank with spectral Kurtosis estimator. The Kurtosis estimator allows detecting non-Gaussian signals (interferences). The Kurtosis estimator can be expressed as following:

$$K = n \frac{\sum_{i=1}^{n}(X_i - X_{avg})^4}{(\sum_{i=1}^{n}(X_i - X_{avg})^2)^2}$$

The Kurtosis estimator is applied to each sub-band to obtain the spectral Kurtosis estimator in order to detect the RFI frequency location. The sub-band separation is carried out by a filter bank with computational efficient structures like DFT filter banks.

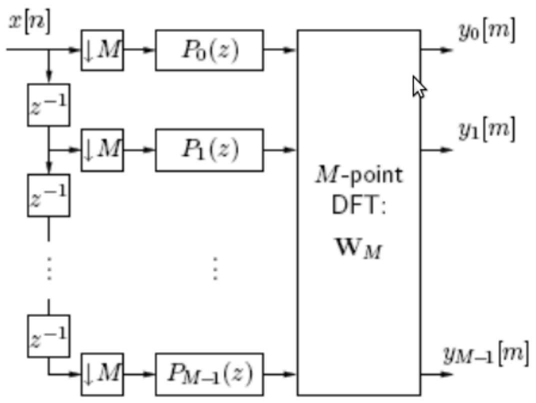


Figure 1: DFT filter bank

The proposed architecture for massive data processing will implement a DFT filter bank with Kurtosis estimator per band in order to integrate a sub-band RFI detection technique for space radiometers.

2.2 Hardware platform for MPSoC Data Processing

The hardware platform is based on two space equivalent FPGA. The first one is a Rad-tolerant ProAsic 3E from ACTEL. This FPGA will integrate a LEON2 processor to control the signal processing algorithms based on real-time operating systems like RTEMS (space-qualified OS). Signal processing algorithms will be integrated in a Virtex5 FPGA specifically useful for high-performance DSP applications.

The proposed board is presented hereafter.

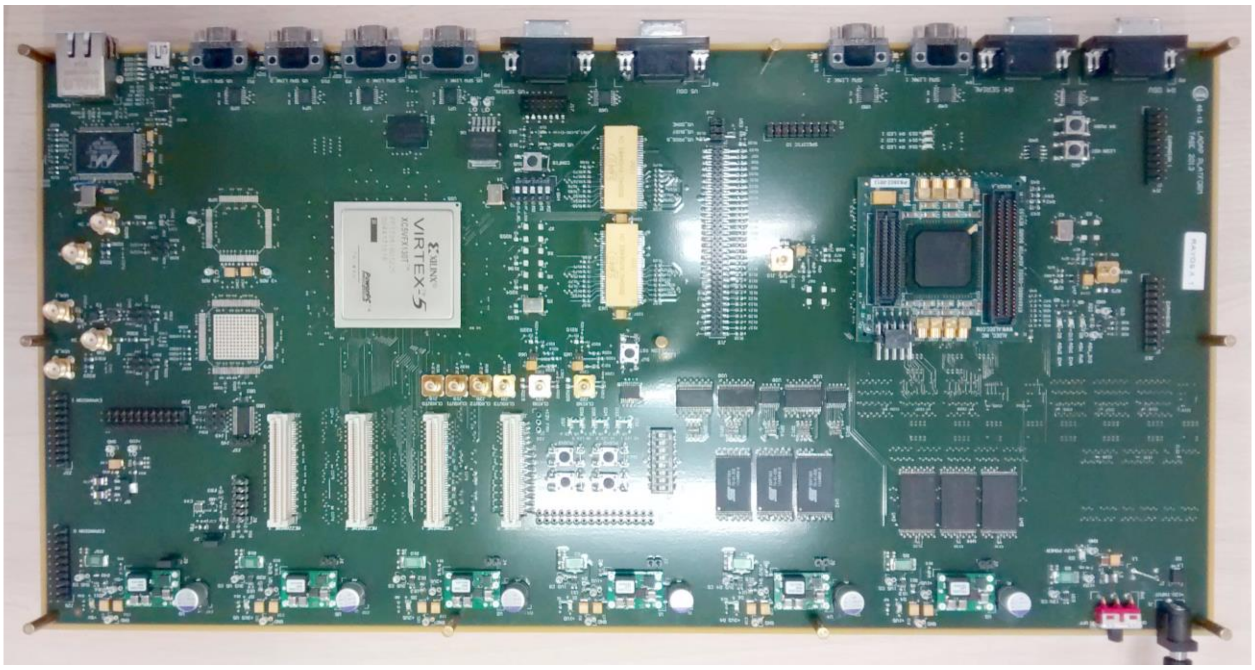


Figure 2: LADAP Board

Next table summarize the memory and interfaces resources of LADAP board for signal processing applications (Virtex5).

FLASH	**SST39VF3201 x3 configuration: 2Mx40 (EDAC)**
SRAM	**CY62177EV30 x3 configuration: 2Mx40 (EDAC)**
SDRAM	**3DSD4G16VS8483 x2**
Ethernet controller	**88E1111**

A DSP controller based on an embedded microprocessor will be used to interface with LEON2 processor and to command and control DSP applications. LEON2 and DSP controller will interact with a dual-port RAM as a mail-box. This technique is widely used as a message passing mechanics. The dual-port RAM will be integrated in the LEON2 memory map by using the specific IO signal in order to use the dual-port RAM as a specific page of the memory. An example of the proposed architecture of the interconnection of both FPGAs can be seen in next figure.

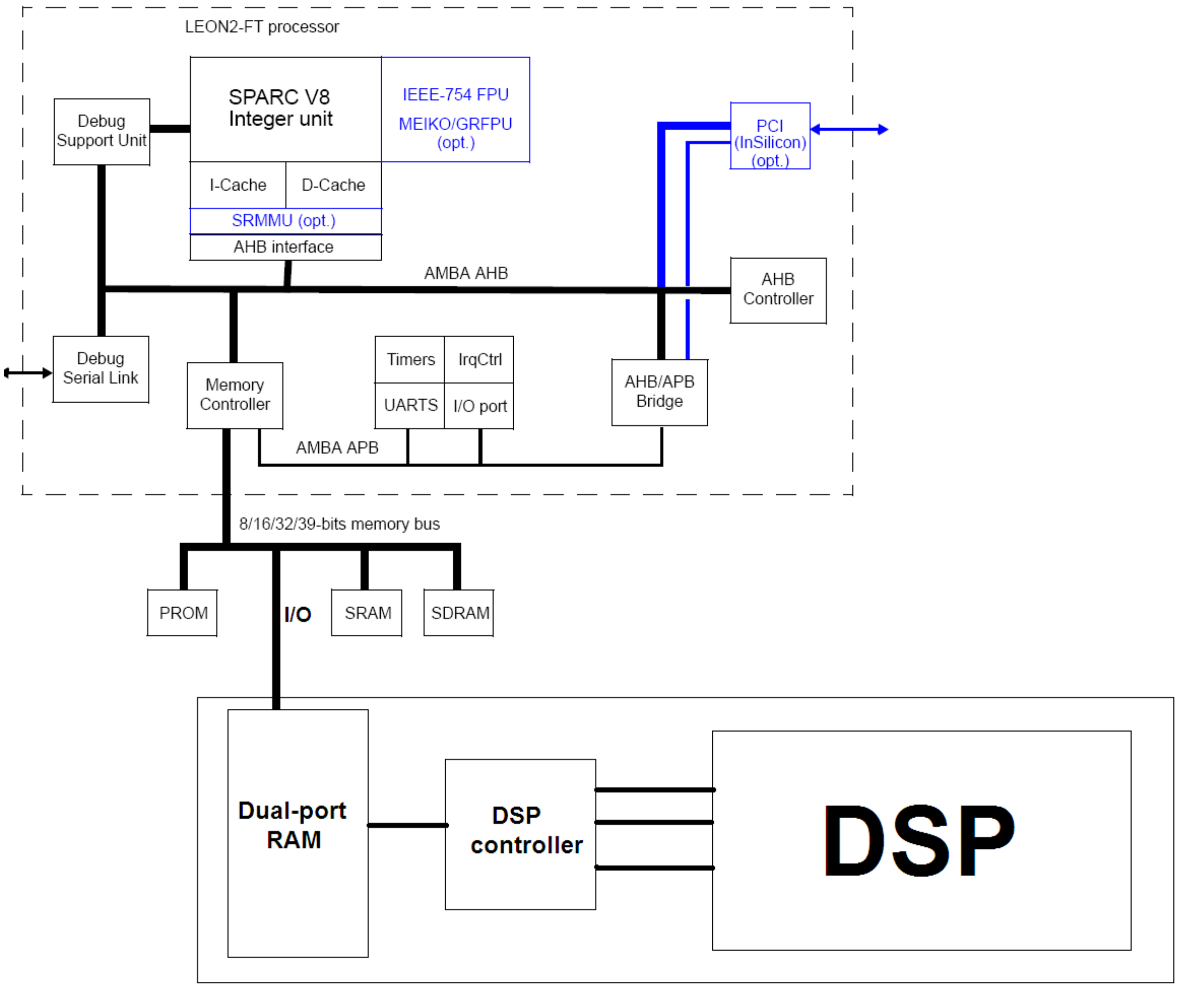


Figure 3: Hardware architecture for On-Board data processing with Dual-port RAM interface

LADAP board with dual-port RAM interface will be also used for Self-Healing Architecture for Space in next section as collaboration between Tecnalia (reconfiguration techniques) and TASE (space-equivalent hardware architecture and LEON2 controller).

3. Proposed Self-Healing Architectures for Space

Dynamic Partial Reconfiguration (DPR) for FPGA devices has many advantages not only for the Space domain but also for many other industrial domains. Nevertheless, the space domain is a very demanding environment exposed to high radiation doses and where reliability of systems is critical. Taking into account these issues, in the context of subtasks T4.3 and T4.4 and the living lab 3 (WP9: Space applications) a platform will be built which allows dynamic reconfiguration of a Xilinx FPGA but including reliability and self-healing features.

The proposed platform will have the following characteristics:

- Designed for Virtex 5Q Devices (certified for Space environment) and implemented on the LADAP hardware board
- The core of the platform will be a Microblaze processor which controls both the reconfiguration process and some of the fault-tolerant features.
- It allows Dynamic Reconfiguration of custom peripherals in the Virtex 5 FPGA device
- It includes fault-tolerant features against radiation induced errors both for error mitigation (Xilinx Isolation Design Flow methodology, watchdog timers interruption …) and error detection and correction (periodic configuration memory scrubbing, continuous check of configuration memory, readback CRC of configuration memory …)
- The platform can be controlled by the software implemented in the LEON processor of the LADAP platform through a communications interface implemented as a shared dual port RAM memory.

3.1 Module Architecture Description

The architecture of the DRM is shown in the following figure:

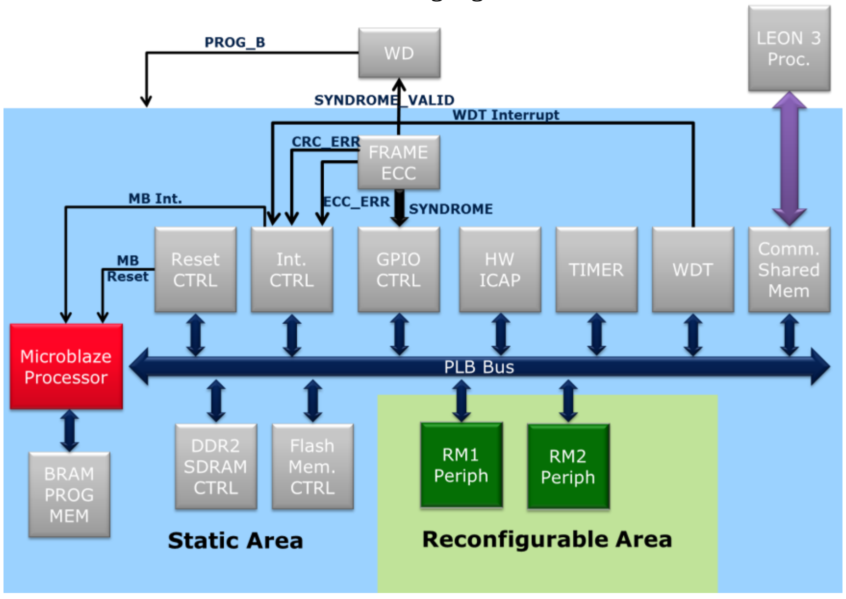


Figure 4: DRM architecture

As it has been already mentioned the FPGA device is divided in 2 different partitions a reconfigurable area when the different modules which are potentially reconfigurable are implemented and a static one where the DRM is implemented. The design in the static area has a higher criticality level as it is in charge of controlling the rest of the modules implemented in the FPGA and also of the communications with the control software in the LEON2 processor; therefore more error protection mechanisms should be implemented on it. The DRM has the following modules:

- **Microblaze processor**: Embedded processor implemented using logic resources. It's the core of the system in charge of controlling the dynamic reconfiguration of peripherals, some of the error protection functions and configuration of the system. The fault-tolerant feature of the processor will be activated. The processor accesses its peripherals with a PLB bus (Xilinx EDK peripheral bus for Virtex 5 devices).
- **Program memory**: It is the 64 Kbits code and data memory used by the microblaze processor. It is implemented using Block RAM memory resources of the device. These resources are protected with the ECC feature to detect and correct radiation induced errors.
- **Reset and Interrupt controllers:** Peripherals in charge of controlling the reset and interrupt request (IRQ) inputs of the microblaze processor. They receive the different reset and interrupt sources and generate the appropriate reset and IRQ signals for the processor.
- **FRAME_ECC primitive:** It's a dedicated resource of Virtex 5 devices in charge of autonomously monitoring the configuration memory. It generates three different output signal:
    - A SYNDROME_VALID signal each time it reads a new configuration frame; this signal is used as a "heart beat" by an external watchdog for SEFI detection.
    - An ECC_ERROR signal together with a SYNDROME signal each time it detects an erroneous frame. The ECC_ERROR is the error flag and the SYNDROME shows the address of the faulty frame so the processor can repair it via the HW ICAP.
    - A CRC_ERROR flag that shows that there is an error in the configuration memory that cannot be located and therefore full memory scrubbing is required.
- **HW ICAP controller:** It is the peripheral in charge of controlling the internal configuration port of the device used during the dynamic reconfiguration process. Data from the partial configuration images is loaded into the FPGA configuration memory through this peripheral.
- **Watchdog Timer:** It is a safety mechanism to avoid processor's crashes. It is a continuously running timer that generates an interrupt signal for the processor each time it overflows, the processor then resets the timer. If the counter overflows twice without a processor reset, then it generates a reset signal that restarts the whole system.
- **Communication Interface Shared Memory:** It is a dual port shared memory that is employed as a communications interface with the control software running on the LEON3 processor of the LADAP board. It is implemented using block RAM memory resources of the FPGA and it is divided in two different areas: In the first one the LEON3 processor writes messages and the microblaze reads them and the other one viceversa.
- **External Memories Controllers:** The system has access to two different external memories: a SDRAM memory and a flash one. The flash memory is employed to store the full and partial configuration images files and the SDRAM memory could be used for storing program data.
- **Timer**: A timer peripheral that could be employed for timing functions of the software running in the microblaze.

Apart from the DRM implemented in the static area of the device, additional modules for different applications can be implemented in the reconfigurable area. These modules are accessible to the microblaze also through the PLB bus, so it can monitor and configure their working. As mentioned before, their functionality can be dynamically changed by the DRM. The images for the different reconfigurable modules are prestored in the flash memory and can be accessed by the processor any time. Additionally, new images for the reconfigurable areas can be provided by the LEON3 processor. These modules have a lower criticality level and, therefore, a lower number of error protection mechanisms are implemented on them. As proof of concept, two different types of modules are implemented: one for a simple mathematical operation (addition, subtraction or multiplication) and another one implementing different FIR filters.

4. Proposed MPSoC Architectures for Space

This section describes an MPSoC demonstrator for space image processing. It is based on image compression and encryption according to CCSDS standards. Next figure shows the block diagram of data processing flow.

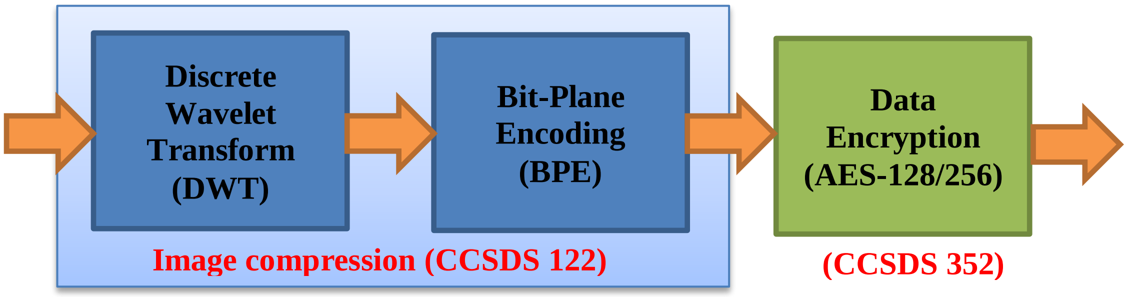

Discrete Wavelet Transform (DWT)
Bit-Plane Encoding (BPE)
Data Encryption (AES-128/256)
Image compression (CCSDS 122)
(CCSDS 352)

Figure 5: Data processing flow for image compression & encryption

4.1 Data compression (CCSDS 122)

The dilemma of modern on-board payload data-processing unit has existed for decades: the data rate of modern cameras in spacecraft is ever-increasing while the down-link is not always available and its bandwidth is limited. To meet the requirements of high data volume and high-speed processing capability, the Consultative Committee for Space Data Systems (CCSDS) has developed three standards related to image compression: CCSDS 121.0-B-1, CCSDS 122.0-B-1, and CCSDS 123.0-B-1. CCSDS 121.0-B-1 is a recommendation for lossless data compression, approved in 1997. The second issue, CCSDS 121.0-B-2, approved in May 2012, makes a few modifications on the first issue. The CCSDS 122.0-B-1,1 approved in 2005 defines the standard for image data compression (CCSDS-IDC). The CCSDS 123.0-B-1, approved in May 2012, specifies a method for lossless compression of multispectral and hyperspectral image data and a format for storing the compressed data. Among the three standards, the CCSDS-IDC is specifically tailored for grayscale monoband images. It was designed for image data from payload instruments of rockets, satellites, or spacecrafts; thus, it satisfies all the memory and computation restrictions of this kind of equipment. The algorithm is based on a three-level two-dimensional (3-l 2-D) discrete wavelet transform (DWT) that performs decorrelation, followed by a bit plane encoder (BPE) that encodes the decorrelated data. The algorithm can provide both lossless and lossy compression.

4.2 Data encryption (CCSDS 352)

Traditionally, security mechanisms have not been employed on civilian space missions. In recognition of the increased threat, there has been a steady migration towards the integration of security services and mechanisms. For example, ground network infrastructures typically make use of controlled or protected networks. However, telecommands, telemetry, and science payload data, are still, for the most part, transmitted over unencrypted and unauthenticated radio frequency (RF) channels. As the threat environment becomes more hostile, this concept of operation becomes much more dangerous. This CCSDS Cryptographic Algorithm Recommended Standard is the foundation for all other CCSDS security Recommended Standards and Recommended Practices.

This CCSDS Cryptographic Algorithm Recommended Standard is necessary because of the increasing interconnection of ground networks; the movement towards joy-sticking of instruments by principal investigators; the decreasing costs for hardware, potentially allowing cheap rogue ground stations to be established; and national trends towards enhancing mission security. These recommended algorithms establish a set of common denominators among all missions for implementing information security services.

AES is a symmetric, block-cipher algorithm operating over 128-bit blocks of data. The algorithm operates over a 128-bit plaintext input block which results in the output of 128-bits of ciphertext (encrypted) data. AES has been adopted by the United States as its official data encryption standard. ISO has also adopted AES as an International data encryption standard. AES has withstood the test of time and has been extremely resilient against attack.

AES may be used in several modes of operation such as cipher-block chaining (CBC), electronic codebook (ECB), cipher feedback (CFB), output feedback (OFB), and counter (CTR). Each of these modes accomplishes the same result – turning plaintext data into ciphertext data.

Each of these modes operates differently with different security strengths. The chaining and feedback modes result in linkages from one block to another which means that if a block is lost or damaged, decryption will be affected since the decryption process also relies on the block linkages. On the other hand, counter mode does not employ any linkage between blocks and therefore can be implemented in parallel.

Following CCSDS specifications will be used:

- **ALGORITHM AND MODE**
  In order to achieve a minimum baseline all CCSDS missions shall use the Advanced Encryption Standard algorithm for encryption.

- **CRYPTOGRAPHIC KEY SIZE**
  CCSDS implementations shall normally use a 128-bit key, but larger key sizes may be chosen for stronger security. AES is key agile and supports key sizes of 128-bits, 192-bits, or 256-bits.

- **ALGORITHM MODE OF OPERATION**
  CCSDS implementations shall normally use Counter Mode (see next figure).

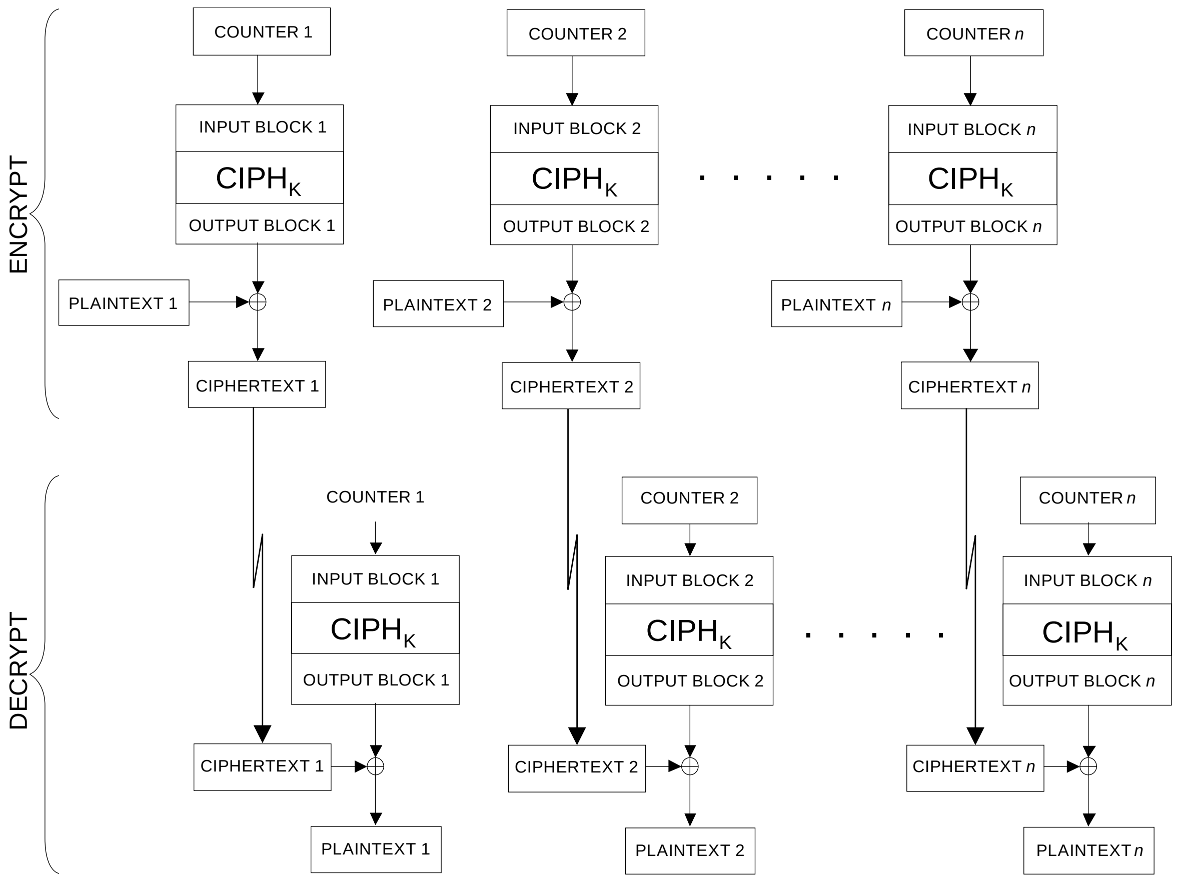


Figure 6: AES-CTR encrypt & decrypt operation

4.3 Hardware architecture

The validation of proposed application will be based on Quad-Core LEON3 processor (SMP). A RASTA hardware platform (Gaisler) will be used for embedded processors.

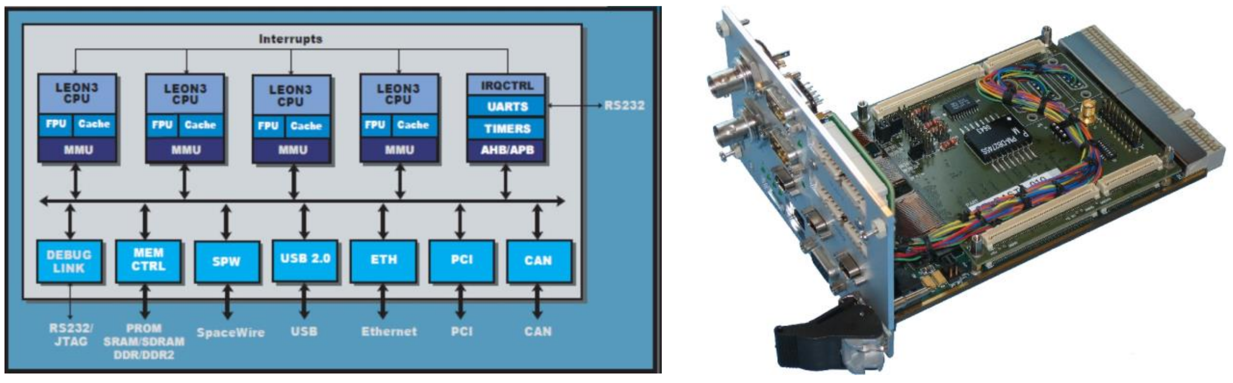


Figure 7: LEON3 Quad-Core architecture and hardware platform

The image compression and encryption algorithms will be parallelized using OpenMP paradigm under linux OS as a proof of concept.

5. Proposed MCSoC SW Modeling Toolset and Procedures – preliminary

5.1 Introduction

New execution platforms provide a great flexibility to improve application performance and flexibility while reducing energy consumption, weight and other important parameters in the space domain systems.

However, space domain applications have just started to consider taking advantage of new execution platform capabilities, such parallel execution over multiprocessor HW, reconfigurable hardware, etc. This progressive adaptation is being limited due to the stringent requirements imposed by the required certification process.

This Use Case will evaluate how to exploit the flexibility provided by the new hardware platforms without jeopardizing the certification of the space domain systems. The development process of space systems will be strongly affected by this new flexibility. The modeling, analysis and implementation of next generation space domain applications has to be adapted to the new execution platform capabilities, while maintaining the hard(strong) constraints imposed by this application domain.

5.1.1 MPSoC modeling and analysis needs for Space Domain

Static resource scheduling clearly helps the certification process. However, this approach can rarely take full advantage of the execution platform capabilities. Two main issues have to be addressed in space application development to exploit the potential benefits of these new hardware platforms:

a) Application models normally does not have the adequate granularity in order to allow the development toolset to extract the medium- and fine-grain parallelism that will help in reducing the response times of key application functionalities.
b) The tradeoff between dynamic reconfiguration of HW functionalities and static resource planning has not been properly addressed in the space application domain.

These challenges have to be addressed by the next generation of tool suites to be used in space domain development to maximize the potential benefits of the new HW platforms.

5.1.2 Existing modeling and analysis environments

There are some modeling tool suites with analysis capabilities that could be good candidates to be used as tool chains in the development process of space domain application, e.g., MAST, Polarsys or Ocarina+Cheddar. However, these toolsets are normally based on well-established modeling languages, such as UML/Marte or AADL, that offer a powerful set of modeling artifacts, but at the same time it is an excessively broad set to be used in a practical and easy way. These artifacts allow a wide range of applications to be modeled and, with some limitations, analyzed. However, it is also this modeling flexibility which limits their use in the restrictive space domain application. The available flexibility does not provide the system engineer any guide about the design process. It is a proper design process what enables a correct system model, and this system will allow for a precise analysis while taking advantage of the new hardware capabilities.

This UC proposes the development of a new toolset based on well-established concepts and analysis, but tailored for the development process of space domain applications over MPSoC. This toolset will guide the system engineer through the application development process (modeling, analysis and implementation) and will also take advantage of new hardware capabilities in a comprehensive and adequate manner.

5.2 Proposed toolset

A software tool suite named "art2kitekt" (a2k) is under development for the design of HW and SW of mixed criticality real-time systems. This tool will provide to the target users an assisted and guided process for modeling the system (both HW and SW) and performing the analysis over those models in order to check if critical requirements are satisfied with this design.

Moreover, this tool also provides some capabilities that foster the development process of the product. On one hand, a2k provides code generation capabilities taking as input the designed system model, analysis algorithm results and the target platform. On the other hand, the tool supports the interconnection with other development tools through the appropriate standards facilitating the sharing of data in the development process.

It is worth noting that, in order to provide this guided system modeling and analysis, the a2k tool organizes features through a set of stages and application domain profiles.

5.2.1 Goal

The main purpose is to provide the system engineer with a new integrated modeling and analysis environment that will be capable to perform the offline analysis, configuration and code generation, while assisting the engineer in the modeling process.

In highly-integrity systems, it is mandatory to pass a certification process. This process is a very important part of the product cost. Thus, following a model development approach that automates analysis and code generation, as well as the generation of evidences to pass the certification process, is needed.

5.2.2 Guided modeling and analysis

To allow the engineer to select among different application domain, a set of **profiles** will be available. This will be useful to bundle different predefined execution platforms which the engineer could make use without the need to input every device element and its parameters to the system, as well as the analysis techniques required for the selected platform and the specific code generation support.

To guide the engineer through the whole modeling and analysis process, a set of **stages** are provided in the tool suite in order to incrementally input system data, perform required analysis, and generate the low-level supporting code for the designed model.

5.2.3 Internal toolset architecture

The internal architecture of the a2k toolset is organized as follows:

System Model Server (shown in Figure 8): This server (a2k server) will provide features for storing, querying and retrieving models to the corresponding client tools.

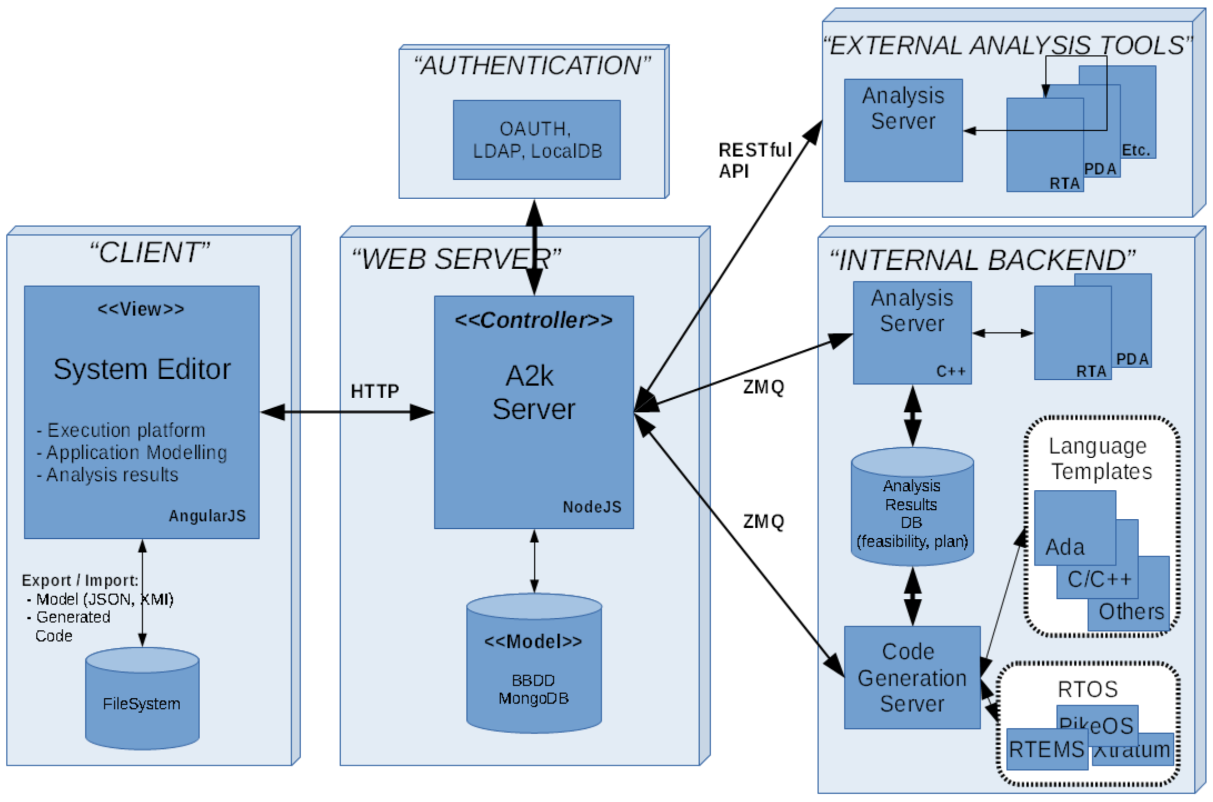


Figure 8: Internal Software Architecture

System Editor: This tool will guide the user (engineer) in the system design by restricting the modeling capabilities, so that it matches the model that can be analyzed by the analysis stage. Usually the editor will provide more design flexibility in the software side than in the hardware part. In this way, the modeler will provide specific interfaces, used as a wizard, in order to guarantee a proper modeling of the system. This tool will store and retrieve the models from the SMS.

Analysis Server: This component will provide a set of analyzing algorithms for each supported execution platform. This server will provide an analysis report that will be sent back to the user and an annotated model to be used by the following stages.

Code Generation Server: This component of the toolset will generate the low-level supporting code and configuration files that will ensure the assumptions and results obtained by the analysis stage.

6. Status of the Art of Multicore Processor for avionics applications Survey

Satellites are considered expensive and unreliable computers may be a risk for the mission. Past space computers were expensive and hard to get, hence the drive to make space computers was based on COTS.

Present-day satellites carry very weak payload and platform computing performance capabilities. In the future, powerful onboard computing is desired. However, electric power for this task is quite limited. This section examines the state-of-the-art available space computer with particular emphasis on multicore processors with high-performance low-power space computers. As part of introduction of survey, a synthesis of the work performed by Ramon-Chips Ltd in their paper of "*Survey Of Processors For Space*" (Ran Ginosar) is here reported. As part of the survey a trade-off and a baseline solution of multicore processors for EMC2 avionics applications will be identified.

6.1 Requirements for using processors in space

Processors for space are required to be tolerant to the following radiation and environmental effects: TID (100-300 kRad for GEO/MEO and beyond, 10-50 kRad for LEO missions), Latch-up, SEU, SET, SEFI, temperature cycles, vibrations and others challenging requirements that make out of game the possibility to use COTS products. Another reliability issue is the complete and permanent failure of a processor or a critical sub-component. None of the methods surveyed here mitigate such failures, and the common mitigation method is based on deploying spare computers in combination with either a central reconfiguration circuit or a distributed recovery mechanism.

Processors for space applications are typically required to achieve the following targets:

- high performance,
- low cost,
- low power dissipation,
- and reliability.

This is sometimes achieved by high integration (for high performance, low cost and low power dissipation) and high tolerance to radiation and environmental effects (for reliability). The problem is that most available processors and integrated systems-on-chip achieve only some of the targets and fail on others. This is indicated below when relative advantages and disadvantages are listed, and exemplified in later sections.

6.1.1 **RH processors**

Certain processors are fabricated on dedicated RH processes. Advantages include: High tolerance to radiation effects, thanks to the RH process; In some cases, such processors achieve high performance. This can be especially true when using custom design methods similar to those employed for the design of COTS high- performance processors; Compatibility with similar COTS devices (example: PowerPC). This results in easy migration of codes and application to the space environment; This approach can offer high level of integration, including the inclusion of special I/O controllers dedicated to space applications.

The disadvantages of using RH processes include: High cost—RH processes have limited use and the high price of modern fab (in excess of one billion dollars) is amortized over a very small market; Not widely available—there are only about couple of RH fabs in the USA and no similar advanced processes elsewhere. Use of the RH processes in the USA is ITAR controlled and is not widely available to non-USA customers;

6.1.2 RHBD Processors, a bridge toward future European multiprocessor

This family contains processors that are designed as ASIC and fabricated on commercial CMOS processes. Radiation hardness is achieved by design techniques in the layout, circuit, logic and architecture areas, hence the name Radiation Hardening by Design (RHBD). Advantages of this family include high tolerance to radiation effects (higher than RH processors), medium cost—more expensive than COTS processors, mostly due to low production quantities and high cost of qualification, but at the same time, they are less expensive than RH processors thanks to using a regular commercial fabrication process. Finally, RHBD processors can offer high integration (inclusion of I/O controllers dedicated to space applications) since they are designed as ASIC and since typically the CPU itself takes only a small portion of the silicon die. Disadvantages of RHBD processors—they are usually slower than COTS processors since they are designed as ASIC chips and not as custom processors.

Most RHBD processors are based on the successful European LEON architecture.

- The SPARC V7 ERC32 and TSC695FL are made by Atmel in France [01]. Originally a 3-chip set, it is now a single chip CPU. It has been used in a large number of satellites and systems in Europe and elsewhere. It achieves 12 MIPS / 6 MFLOPS at very low power (0.3W for the core excluding I/O). The OBC695A V7 SBC is offered by SSTL using Atmel's TSC695. Tiger V7 is another SBC offered by Saab Aerospace using Atmel's TSC695.
- Atmel LEON2 AT697 provides a major step forward from the TSC695FL SPARC V7 processor. It is a LEON2 SPARC V8 processor [02] [03], based on IP core provided by Aeroflex Gaisler. The chip includes the processor, memory interface and a PCI interface and executes at 70-80 MHz. Astrium has developed a SoC derivative of LEON2 named MDPA (Multi-DSP/ µProcessor Architecture) featuring LEON2 at 70 MHz, I/O controllers for 1553, SpaceWire, and CAN [04] [05]. The SoC contains a DSP module with modem, encoder and decoder (suitable for high speed TM/TC, 600 Kbps each direction).
- Aeroflex Gaisler has developed a SPARC V8 processor (LEON3) and SoC architecture and implemented it on ACTEL RTAX rad-hard FPGA [06]. This development revolutionized this field: With rad-hard FPGA, such processors are more widely available and can be customized to specific needs. The main limitation is that ACTEL RTAX FPGA parts are ITAR controlled. The second limitation is performance (clock rate below 25 MHz).
- Saab Aerospace has developed another SoC based on the LEON2FT code-named. COLE [07], intended for their Panther SBC [08]. The SoC design demonstrates a high level of integration, incorporating multiple 1553, SpaceWire and CAN interfaces among others. Astrium has developed SCOC3, a SoC based on LEON3, incorporating a very large set of I/O cores [09] and fabricated on Atmel RH 0.18u process (ATC18RHA).
- Thales Alenia Space developed LEONDARE LEON3 SoC for space. It is based on IMEC DARE RHBD library, which achieves density of 25Kgates/mm2. The architecture combines LEON3 CPU with 3 SpW+RMAP and other cores, planned for 208-pin CQFP and fabricated on UMC commercial 0.18u process through Europractice shuttle service.
- Aeroflex has released UT699, another LEON3FT SoC, that combines PCI bus with several on-chip serial IO cores: 4 SpW, CAN and Ethernet. The ASIC is implemented on 0.25u CMOS, packaged in 352-pin CQFP, operates at 66MHz, and dissipates 5W.
- Aeroflex Gaisler and Ramon Chips developed GR712RC, a dual-core LEON3FT SoC fabricated on TowerJazz 0.18u CMOS. It contains multiple I/O cores and executes at clock rates higher than 100MHz, dissipating less than 2W. In order to support high reliability and high speed and avoid the risks associated with high pin count packages in space applications, a 240-pin CQFP package is used. The processor periphery architecture is based solely on serial I/O cores employing an I/O switch matrix to reduce the number of actually required I/O pins.

RHBD with LEONs processor constitute the bridge toward future European multiprocessors.

6.1.3 **8.1.3 Single COTS Processor with Time Redundancy (SIFT)**

In this approach, a single COTS processor is used together with Software Implemented Fault Tolerance (SIFT), which executes the entire software or certain software sections twice or more. There are two levels of granularity: Instruction level redundancy, where each instruction is executed twice and additional instructions compare the results, requiring compiler transformation of the code, and procedure level redundancy, where the programmer writes the code to invoke certain procedures twice, compare the results and use software for recovery in case of mismatch. The latter approach may also require some additional hardware to protect the critical data and the critical software. The main advantage of this approach is that it is relatively inexpensive. The principal disadvantage (in addition to the added complexity of programming) is the performance penalty: Executing some code twice, comparing the results and recovery in cases of mismatch typically incur substantial performance penalty, possibly nullifying the performance gained by using high speed COTS processors. Three SIFT examples are known:

- CNES created DMT, time replication for COTS microprocessors. Critical software sections are executed twice, compared and a recovery is initiated in case of fault. DMT uses SEE-protected storage to assure safe context. The replication overhead requires processor 4 times faster than the required processing capability. DMT is only a design idea, yet to be implemented in real hardware.
- Another example is a time replication technique based on an instruction level approach that has been developed by Politecnico di Torino (Polito).
- The third one is the ARGOS COTS vs. RH experiment: Stanford University researchers developed the EDDI technique (Error Detection by Duplicated Instructions) implemented in the USAF ARGOS satellite (launched in 1999), based on the IDT-3081 commercial processor (R3000 instruction set). Each instruction is executed twice.

6.1.4 **8.1.4 Triple COTS: TMR at the system level**

TMR (Triple Modular Redundancy) architectures combine three COTS processors and voting logic. The processors do not need to be stopped on SEU. TMR offers high performance (high end COTS such as the latest Pentium or PowerPC processors may be used) and high SEU tolerance –SEU errors, resulting in erroneous bits at the outputs of the processors, are fixed. The downside of TMR is high cost, requiring large area/volume and power, as well as special hardware for voting and usually additional hardware and software for recovery from internal SEU errors (inside the processors) that cannot be fixed by voting and require scrubbing or reset. Two TMR examples are known:

- One example of TMR is Maxwell SCS750 SBC, based on three IBM PowerPC750FX 1.6 GIPS (800 MHz) micro-synchronized microprocessors (i.e. working in lock-step) operating in TMR mode, with a centralized voter integrated in a rad-tolerant FPGA that is functionally immune to upsets.
- Another example is EADS DMS-R, a computer board based on ERC32 (Atmel TSC695) processor, and a fault-tolerant TMR version using 3 boards for the ATV of the ISS (Autonomous Transport Vehicle of the International Space Station).

6.1.5 **8.1.5 TTMR on COTS VLIW processors**

COTS VLIW processors execute multiple instructions in parallel, and the parallel instruction streams are pre-programmed. SpaceMicro (USA) implemented this capability for "time TMR (TTMR)," where each instruction can be executed three times and the results can be compared and voted, all within the same VLIW processors. TTMR offers high performance (in fact, TTMR processors are the fastest available space processors today) and high SEU tolerance, thanks to embedded TMR mechanism, but it is expensive, is limited to VLIW processors, and is hard to generate code for.

The only known examples are SpaceMicro Proton 100k and 200k. The code executes two copies of an instruction, compares the result, on mis-match executes the same instruction the third time and compares for majority voting. Proton 100k SBC uses Equator BSP-16 (up to 1200 MIPS) and Proton 200k SBC uses TI 320C64xx (up to 4,000 MIPS fixed point or 900 MFLOPS).

As part of the space processor literature survey, the following picture show the results of the analysis did by Ramon Chips that reported or estimated performance of the surveyed processors. As can be seen in this figure, TTMR and TMR provide the highest performance, followed by modern RH. Majority of the other processors are based on RHBD solution which is privileged by European Industries for ESA space programs.

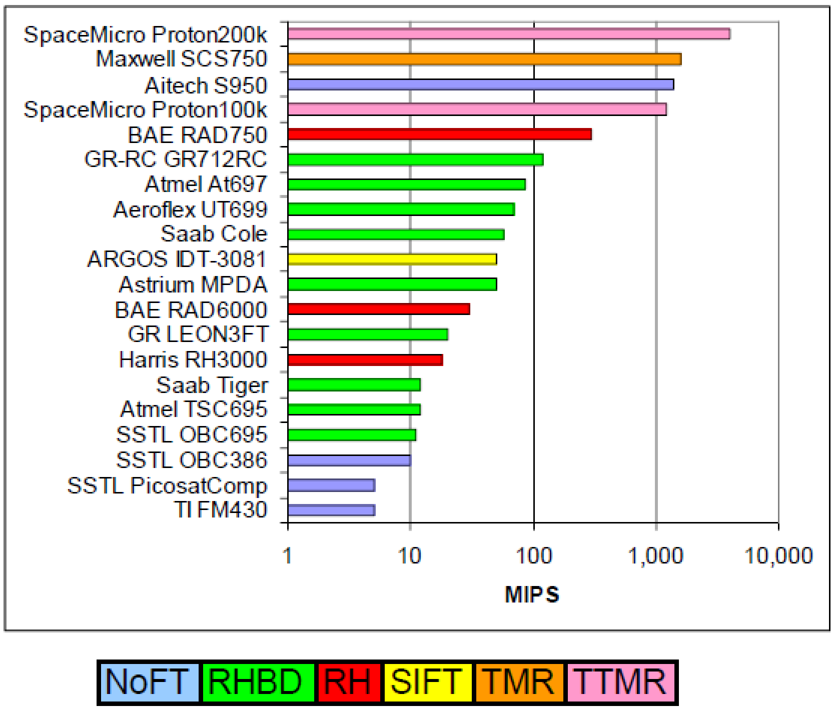


Figure 9: Performance of space processors done by Ramon Chips in the 2012

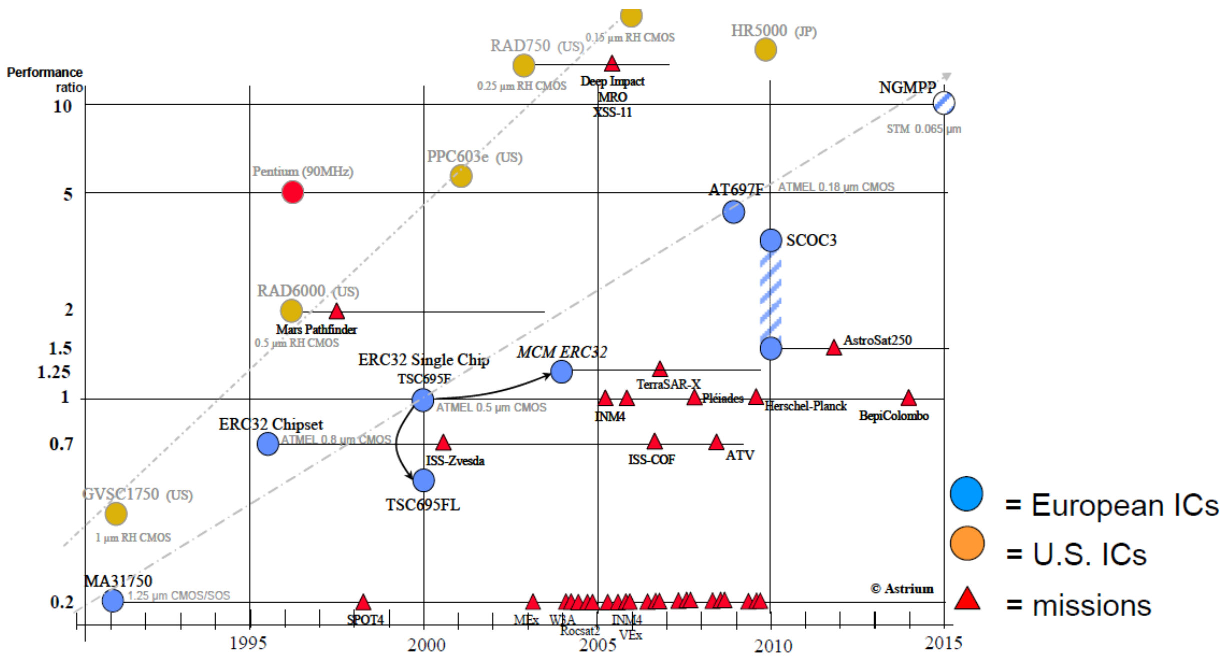
Indeed ESA has invested in the last years studies for space processors for Paylaod and platform applications. In this study it has been collected information, requirements and performance for both PL and platform mission. In this study ESA is pushing for having ITAR free processor requirement.

[01] http://www.atmel.com/dyn/resources/prod_documents/doc4204.pdf
[02] http://microelectronics.esa.int/mpd2007/AT697F.pdf
[03] http://www.atmel.com/dyn/resources/prod_documents/doc 4226.pdf
[04] http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=196&Itemid=151
[05]http://microelectronics.esa.int/mpd2007/MDPA_Pres_Mar ch_2007.pdf
[06] http://www.astrium.eads.net/equipment-product-catalogue/productcatalogue/products/communication-payload-control-and-processing-unit
[07] http://www.gaisler.com/cms/index.php?option=com_conte nt&task=view&id=196&Itemid=151
[08] http://www.itrs.net/
[08] http://microelectronics.esa.int/mpd2007/SCOC3- MPD2007.pdf

6.1.6 European High-Performance Space Processors

Present-day satellites carry very weak payload and platform computing performance capabilities. In the future, powerful onboard computing is desired. However, electric power for this task is quite limited. This section examines the state-of-the-art available space computer with particular emphasis on multicore processors with high-performance low-power space computers.

ESA has defined within their SAVOIR (Space Avionics Open Interface aRchitecture, [SAVOIR]) Initiative working group, a reference architecture for onboard data handling defining the basic building blocks for a typical onboard processing system. Both onboard computers and payload computers elements are driven in future applications by the need to achieve increasing processing performance, but under the constraint of not increasing in parallel the complexity for the applications programmer and minimal increase in power consumption. These constraints are mandatory for acceptance of novel approaches to improve onboard computing performance. EMC2 addresses this coupling of a novel computing HW architecture with a higher layer programming model, both of which need to be developed and optimised jointly.



For future demands in processing performance ESA has provided an assessment of the state of the art and has laid out its plans and expectations recently in technology dossiers for onboard computers, microelectronics and payload data handling [09, 10].

The general outcome is that space applications will require processors reaching upwards of 1 GFLOPS at low power. However, no existing platform achieves this performance. Likewise no platform currently under development claims to deliver this performance in the near future. Note in contrast that each rad-hard Processor core processor unit taken into account for EMC, may have different performance in terms of GOPS with dissipating power > 3W.

The above figure (source: [09]) shows the current state of the art in Europe and the competitiveness to ITAR-controlled products. The figure illustrates clearly that:

1. The current available HW falls far behind the requirements in terms of processing performance (performance values in the above are normalised to ERC32 which offers 10MOPS and 5MFLOPS; relative value of 200, way above the top of the figure, would mark the sought 1 GFLOPS target

2. Up to date (2015) this gap cannot has not been filled by currently planned products
3. European space products are not competitive at all in that aspect, compared to the US market

This clearly indicates the need for novel concepts like EMC^2 to close this gap and regain competitiveness of the European industry. ESA forecasts that a "next generation space digital processor" will perform about 1 GFLOP while dissipating about 1 Watts, or 10 Watts per 1 GFLOPS for a massively parallel processor FPGA. This is in line with the roadmap of space industries and in line with the need to concentrate in a multicore-processor heterogeneous application like avionics application where some inter satellite link functions are allocated.

Despite that it important to distinguish between platform and PL space processor. The latter indeed is requested to have more performing wrt the first one that instead is requested to have multi-functionalities and to implement heterogeneous routine.

[09] Technical dossier on data systems and onboard computers, ESA, TEC-EDD/2011.109/GM
[10] Technical dossier on onboard payload data processing, ESA, TEC-EDP/2011.110/MS

6.1.7 **Considerations on the ARM processors family**

ARM is a leader in microprocessor Intellectual Property. ARM [10] has also targeting his research to the space industry providing a wide range of products.

ARM is the industry's leading supplier of microprocessor technology, offering the widest range of microprocessor cores to address the performance, power and cost requirements for almost all application markets. The ARM Cortex is a family of microprocessors introduced in 2005 by ARM Holdings and based on the ARMv7 instruction set. The Cortex family is divided into series A (Application), the R series (Realtime) and the M series (Microcontroller). The A series is the series addressed to applications processors for smartphone, mobile computing, infrastructure, consumer electronics, netbooks and servers. The R-series is developed for real time applications, hard drives and mission-critical systems, while the M-series is a microcontroller family for engine and industrial control, flash drives and smart cards.

The Cortex-R4 [11] processor is designed for implementation on advanced silicon processes with an emphasis on improved energy efficiency and real-time responsiveness. Cortex-R4 is a 32-bit processor based on the v7R architecture. Cortex-R4 offers support for multicore with Redundant dual-core capability. The Cortex-R4 has 8-stage-dual-issue pipeline in-order with branch prediction. A typical configuration is capable of delivering up to 650 Dhrystone MIPS of performance and a good value for Energy-efficiency, 10 DMIPS/mW.

The ARM Cortex-M0 [12] processor is the smallest ARM processor available. The exceptionally small silicon area, low power and minimal code footprint of the processor enables developers to achieve 32-bit performance at an 8-bit price point, bypassing the step to 16-bit devices. The ARM® Cortex®-M0+ processor is the most energy efficient ARM processor available. It builds on the very successful Cortex-M0 processor, retaining full instruction set and tool compatibility, while further reducing energy consumption and increasing performance. An optimized architecture with a core pipeline of just two stages, enables the Cortex-M0+ processor to achieve a power consumption of just 9.8µW/MHz (90LP process, minimal configuration), while raising the performance to 2.42 CoreMark/MHz.

The ARM architecture could be a suitable solution for space applications. Indeed at the current state the only known Radiation Hardened implementation is from Silicon Space Technology, a privately held American fabless semiconductor company,[13] which has developed an ARM based processor offering superior radiation performance > 300 Krad and latch-up immunity (SEL) in extreme environments. The PA32KASA contains an embedded ARM Cortex-M0 processor with related peripherals supported. It provides 32 Dedicated General Purpose I/O (GPIO) pins, 32 General purpose counter/timers, 2 UARTs end 2 Serial Peripheral Interface (SPI) controllers. It's clock rate is 50MHz @ 25^{0}C, with a low power 3.3V I/O and 1.5V Core.

[10] http://www.arm.com/products/processors
[11] http://www.arm.com/products/processors/cortex-r/cortex-r4.php
[12] http://www.arm.com/products/processors/cortex-m/cortex-m0.php
[13] http://www.siliconspacetech.com/product-type/rad-hard

6.1.8 Multi-core processor for EMC^2 avionic applications

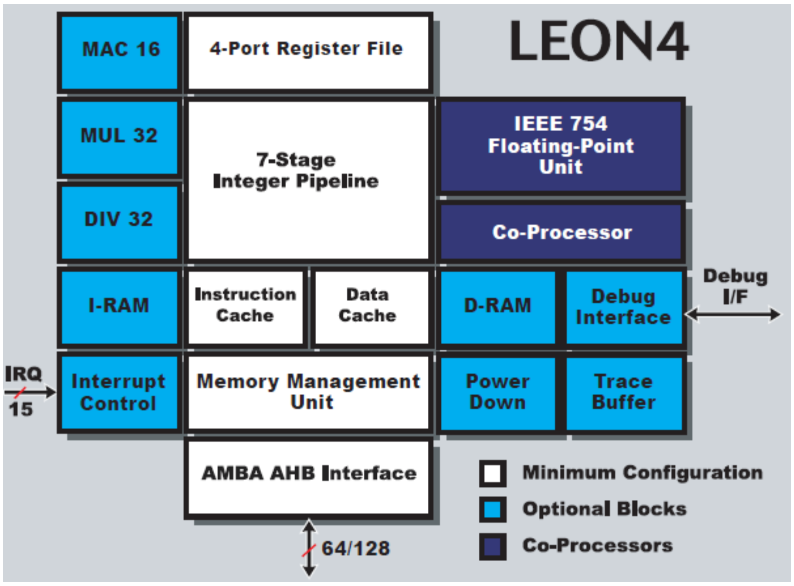
The survey performed in the previous chapter show the available space processors distinguished by European and not European products, performances and mission requirement. As introduced, an important parameter for selecting the right processor is final application for space mission (i.e. payload and platform).

For EMC^2, some functionalities for avionic and inter satellite link need to be implemented in a single processor at platform level. Some requirements have been identified in the frame of EMC^2 project and the space processor survey can help in the selection of the right products. A European multiprocessor seems to be the better solution for this kind of application in the frame of EMC^2. The recently multiprocessor product for space application has been realized by Gaisler Research and sponsored by ESA.

The LEON4 indeed is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. The model is highly configurable, and particularly suitable for system-on-chip (SOC) designs.

The core [11] is interfaced using the AMBA 2.0 AHB bus and supports the IP core plug&play method provided in the Aeroflex Gaisler IP library (GRLIB). The processor can be efficiently implemented on FPGA and ASIC technologies and uses standard synchronous memory cells for caches and register file. The processor supports the MUL, MAC and DIV instructions and an optional IEEE- 754 floating-point unit (FPU) and Memory Management Unit (MMU). The LEON4 cache system consists of separate I/D multi-set Level-1 (L1) caches with up to 4 ways per cache, and an optional Level-2 (L2) cache for increased performance in data intensive applications. The LEON4 pipeline uses 64-bit internal load/store data paths, with an AMBA AHB interface of either 64- or 128-bit.

Branch prediction, 1-cycle load latency and a 32x32 multiplier results in a performance of 1.7 DMIPS/MHz, or 2.1 Coremark/MHz.



The wider interfaces provides higher bus and memory bandwidth which is necessary when designing ASICs with high clock frequencies (800 MHz and above). The LEON4 is fully software compatible with previous LEON processors. The configurability of LEON4 allows designers to optimize the processor for performance, power consumption, I/O throughput, silicon area and cost.

LEON4 can be utilized in both asymmetric multiprocessing (AMP) and symmetric multiprocessing (SMP) configurations. A typical four-processor system is capable of delivering up to 4300 Dhrystone MIPS at 800 MHz.

The basic LEON4 processor core (pipeline, cache controllers and AMBA AHB interface) consumes approximately 30,000 gates and can be implemented on ASIC and FPGA technologies. The processor can reach 125 MHz using Xilinx Virtex-5 FPGAs, or 1500 MHz on fast 32 nm standard-cell technologies.

To save power the LEON4 supports power down mode, individual processor shutdown and clock gating. For the use case of EMC^2 platform application, LEON 4 multi-processor is considered the better solution for avionic application.

[11] http://www.gaisler.com/doc/LEON4_32-bit_processor_core.pdf

6.2 Status of the Partitioned Software Architecture (TSP) for avionics applications Survey

The ARINC 653 specification, defined for aerospace applications, has the goal of providing a standard interface between a given real-time operating system (RTOS) and the corresponding applications. It also provides robust partitioning, with the final goal of guaranteeing safety and timeliness in mission-critical systems.

Partitioned sofware architectures were conceived to fulfill the above requirements. Both, the availability of new processors and an increased necessity of security, have opened new possibilities to use efficiently this approach. The aerospace sector is adapting these concepts on its developments. One of the solutions used to achieve partitioned systems is based on virtualisation techniques. In the following sections we present some operating system suitable to spacial purpose.

8.2.1 RTEMS

RTEMS (Real-Time Executive for Multiprocessor Systems) [1] is a real time operating system (RTOS) designed for embedded processing platforms. Initially developed for military applications, the acronym RTEMS originally stood for Real-Time Executive for Missile Systems or Real-Time Executive for Military Systems. RTEMS is an open source project, actively maintained by On-Line Applications Research Corporation (OAR).

RTEMS can be modeled as a series of layered components (Figure 10) able to provide a set of services to a user application characterized by real time requirements. The executive interface exposed to user applications are grouped into sets called *logical resource manager*.

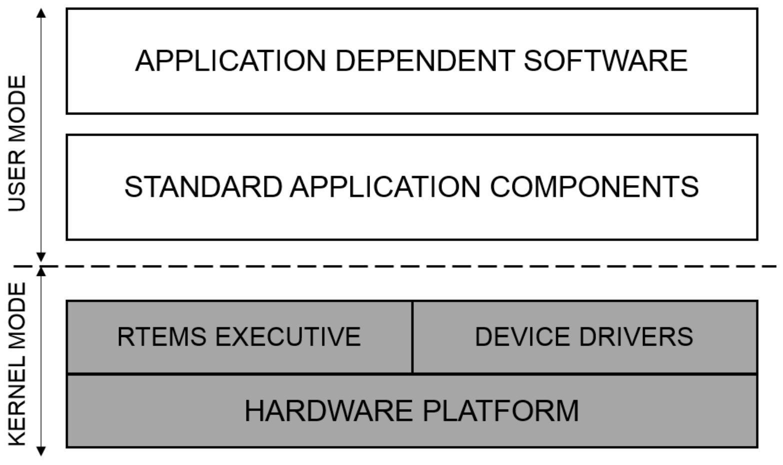


Figure 10: Schematization of RTEMS operating system architecture

The common functions shared by various managers (for example, scheduling, dispatching, object management) are implemented into an *executive core*. This core is based on a set of low-level software procedures, whose implementation depends on the specific CPU.

RTEMS supports various standard Application Programming Interfaces (API), such as POSIX API and µITRON API. It also includes a proprietary API (called Classic RTEMS API) originally based on Real-Time Executive Interface Definition (RTEID) specific.

By means of a flexible plugin system, RTEMS allows the user to select different scheduling algorithms or to include a customized algorithm able to satisfy the requirements of a specific application. RTEMS default scheduling algorithm (Deterministic Priority Scheduling, DPS) implements a priority scheduling with preemption control. By selecting this algorithm, the processor is allocated using a priority-based scheduling algorithm, whereas a round robin policy is used to switch between tasks belonging to groups with the same priority level.

The algorithm guarantees that the tasks executed in a generic instant have the highest priority level among all tasks in the ready state. Tasks with the same priority level are then scheduled according to a FIFO policy.

RTEMS does not provide any form of memory management and there is no concept of process: basically, it implements a POSIX single-process, multithreaded environment. RTEMS defines partitions as contiguous areas of memory divided into fixed size buffers that can be allocated and de-allocated. The system maintains and manages the partitions as a list of buffers.

RTEMS defines messages as user defined, variable length buffers supporting the storage of the communication information. Message exchange between tasks and Interrupt Service Routines (ISR) is managed by means of a message queue and a FIFO mechanism. RTEMS provides synchronization mechanisms (waiting for the arrival of a message in the message queue, polling of the message queue for the verification of the message arrival). A *task manager* handles communication and synchronization.

Currently, RTEMS can be used on a single processor system (as a single software instance) or in a multiprocessor system, supporting both asymmetric and symmetric configuration. As for symmetric multi-processing mode, RTEMS provides appropriate versions of its scheduling algorithms extended to the SMP case.

RTEMS supports the main processing architectures used in aerospace systems, such as SPARC and LEON, PowerPC, MIPS, and ERC32. RTEMS includes a porting of the FreeBSD TCP/IP stack and supports various file systems (for example NFS and FAT).

8.2.2 VXWORKS RTOS

Wind River VxWorks [2] is a real-time networked operating system, specifically designed for the use on distributed processing platforms. It complies with the requirements of various safety standard (for example, DO-178C/EUROCAE ED-12C Level A, ARINC 653 and IEC 61508).
Figure 2 schematizes VxWorks system architecture.

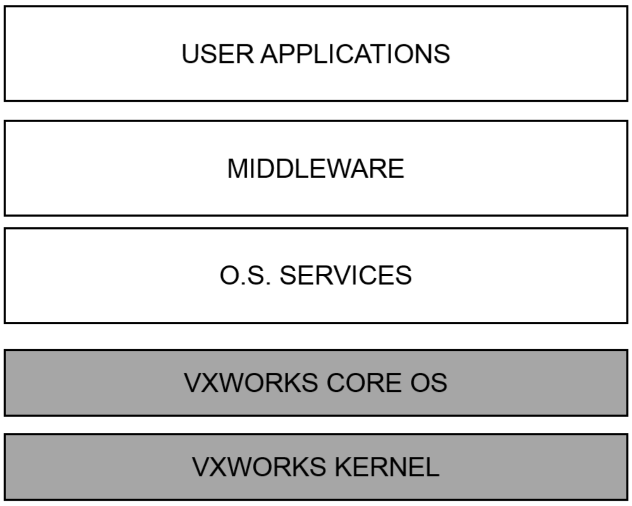


Figure 11: Layered architecture of WindRiver VxWorks operating system

VxWorks supports space partitioning by means of appropriate isolated memory containers based on different virtual memory context. VxWorks manages these contexts via the Memory Management Unit (MMU). The temporal partitioning is supported by means of a scheduling mechanism with preemption. Also in this case, the scheduler is organized in two levels, a priority-base scheduling and a round robin scheduler. The scheduling is static, but it is possible to modify the priority of a task at runtime.

The basic execution unit of VxWorks is the task, roughly corresponding to a UNIX process. A task can be created, deleted, suspended or interrupted (preempted) by other tasks or task itself. Each task has its own set of registers and stack area. Tasks can communicate using UNIX-like inter-processor communication mechanisms.

The VxWorks kernel (Wind kernel) provides semaphores as basic mechanism for task synchronization and for the implementation of mutual exclusion mechanisms. Messages, queues, pipes, sockets, and signals supports instead the communication between tasks. VxWorks also provides shared memory objects for the implementation of messages between tasks running on different processors.

VxWorks provides native support to the multicore environment, using either symmetric or asymmetric mode [3]. It also provides the possibility of leveraging an AMP/SMP mixed mode. VxWorks supports different hardware architectures (e.g. Intel i960, Intel i386, MC680x0, MC683xx, R3000, SPARC, and SPARClite) and provides different APIs (POSIX PSE52, ARINC API, Ada and C).

8.2.2.1 Wind River MILS Platform

WindRiver MILS platform [4] is a solution for the development of devices and systems with high degree of safety. The MILS approach is exploited to ensure a secure allocation of resources and support application isolation. This allows creating a safe and partitioned run-time environment able to provide an efficient base platform for the implementation of secure systems. As required by MILS Specification, a separation kernel is used to create and manage partitions and guarantee the spatial and temporal isolation. The application code runs in user-mode within a single partition, usually on the top of an operating system.

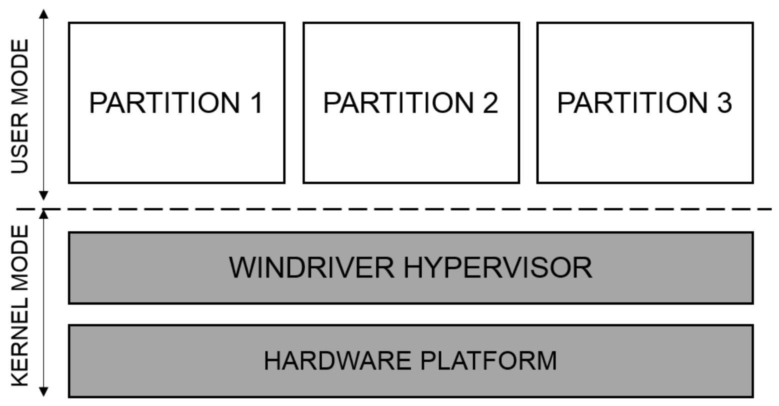


Figure 12: WindRiver MILS platform schematization

The VxWorks separation kernel is based on a two-level, high performance scheduler capable of ensuring a reduced overhead in context switching between virtual boards. The separation kernel manages the first level scheduling, executing the virtual boards in a predefined order. A predetermined time interval within the overall period of scheduling is assigned to each virtual board. The guest operating system running inside the virtual board manages a second-level scheduling, executing threads, tasks and processes according to a local scheduling policy.

An interesting feature offered by the MILS platform is the possibility of transferring a fraction of allocated execution time from a virtual board to another. This allows to increase the performance (i.e. reducing latency in communications, if the time is assigned to a virtual board that runs device drivers), while retaining robustness characteristics.

In a WindRiver MILS based system, the virtual board acts as isolated containers for the allocation of user applications. As mentioned, the virtual board are schedulable entities and have their own memory space and a set of allocated physical hardware resources. This ensures spatial isolation and avoids interferences between the various resources.

Communication is managed by the MILS secure inter-partition communication (SIPC) component. This component handles the communication between virtual boards, ensuring the secure and reliable delivery of data. The SIPC programming interface is derived from the sampling and queuing ports mechanism specified by ARINC 653 APEX. The communication model is based on unidirectional channels, with specific policies controlling the security of the flow.

Alternatively, it is possible to use shared memory as data exchange mechanism (but its use is recommended only for applications with the same level of criticality). The regions of shared memory have configurable access privileges, and they may be useful for the support of legacy applications and device drivers based on this communication mechanism.

VxWorks MILS Platform is distributed in various version, each one targeting a specific application domain. In particular, one version is dedicated to the support of multi-core platforms and asymmetric multi-processing. Each processor runs its own instance of the MILS framework and mechanisms for the exchange of messages between partitions running on different cores are supported.

WindRiver MILS Platform supports various possible execution environments and operating systems. For example, it is possible to cite:

- High Assurance Environment (HAE), a reduced footprint runtime environment supporting critical security features.
- VxWorks Guest OS, a real-time operating system with high level of reliability.

WindRiver Linux Guest OS, a complete version of the Linux kernel, specifically designed to enable legacy Linux applications reuse. The VxWorks fully supports the Intel, ARM, PowerPC/Freescale targets.

On the other hand with reference to the LEON4 target only the 6.7 version has been ported by AEROFLEX Gaisler; the most recent version which is 7 has not been ported on the LEON4.

8.2.3 PIKEOS

PikeOS [6] is a platform providing an RTOS, a type I hypervisor and paravirtualization functionalities. PikeOS allows to satisfy the critical requirements of Integrated Modular Avionics (IMA) systems, such as MILS security requirements, and supports many general features usually provided by traditional virtualization software.
PikeOS architecture is based on a compact micro-kernel providing a set of services. Figure 13 schematizes the software architecture.

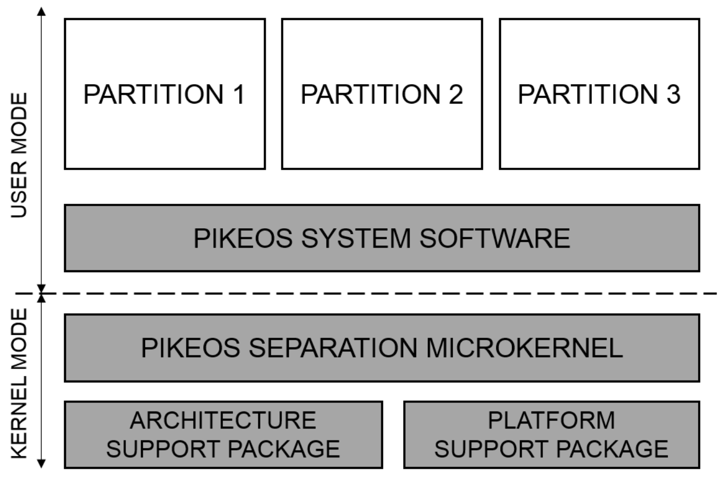


Figure 13: Schematization of PikeOS architecture

Two different software layers compose the intermediate layer between the partitions and the hardware:

- The PikeOS System Software implements partitions, which are applications that run in a secure environment.
- The microkernel provides preemptive multitasking, implementing a scheduling policy based on priorities and supporting the management of time and space.

PikeOS software system, built on top of the microkernel, provides many services that are traditionally in a monolithic kernel as boot applications or file reading. PikeOS supports both native application programming interfaces (APIs) and ARINC 653 APEX specification.

PikeOS scheduler combines time-driven and priority-driven approaches. Specifically, the scheduling period is subdivided in a number of time slices and each time slices is dedicated to each virtual machine. Virtual machines are thus scheduled periodically, receiving a fixed amount of processing resources. This guarantees the time determinism (as virtual machine are executed at fixed time instants) and real time performance. Additionally, a two level priority scheme for virtual machine is adopted, associating a mid-level priority to virtual machine executing time critical tasks and low-level priority to virtual machine executing non-critical tasks. The latter group of virtual machines are executed only when the virtual machines belonging to the first group are sleeping (i.e. not executing tasks), in a round robin fashion. In this way, hard real time requirements are guaranteed for critical application, whereas a best effort is adopted for non-critical ones.

PikeOS supports two ways of communication between virtual machines: ARINC-653 compliant communication ports and shared memory objects. Ports are statically configured communication channels supporting message buffering for reliable and trusted communication. Shared memory objects are instead particularly useful for transferring large amount of data.

PikeOS supports a wide range of single-core or multi-core architectures (e.g. PowerPC, x86, ARM, MIPS, SPARC-v8/LEON, etc.). Multi-core configuration are supported both in symmetric and in asymmetric configurations. In the case of AMP, multiple instances of PikeOS can run in parallel on the different cores. In the case of SMP, the PikeOS scheduler allocates different cores to the partitions maintaining the strict temporal subdivision scheme mentioned above.

Various host operating systems are supported for execution inside different partitions. It is possible to cite:

- Embedded Linux (ELinOS)
- µITRON
- OSEK (AUTOSAR)
- RTEMS
- Windows

Moreover, legacy RTOS can be supported with the addition of specific APIs.

8.2.4 XTRATUM

XtratuM [8] is an open-source, bare metal hypervisor targeting high-criticality real-time systems, which implements the paravirtualization principle.

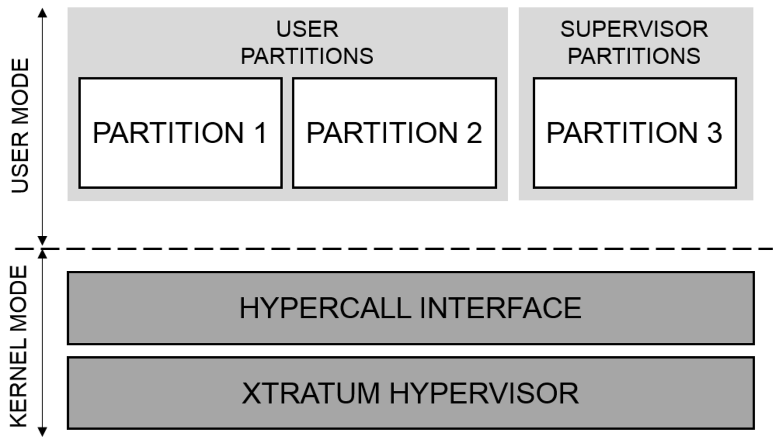


Figure 14: Schematization of XtratuM hypervisor software architecture

The main components of the XtratuM architecture are the following:

- Hypervisor: provides virtualization services to the partitions. The hypervisor is executed in supervisor mode and virtualizes the CPU, the memory architecture, interrupts, and some specific peripheral devices.
- Hypercall interface: defining the paravirtualization services provided by XtratuM. Access to these services is provided through hypercalls.
- Partitions: a partition is an execution environment managed by the hypervisor that uses the virtualized services.

Each partition consists of multiple concurrent processes, sharing access to the processor resources based on application requirements. The term partition code may denote a compiled application running on a bare-metal machine, a real-time operating system and its applications or a general-purpose operating system and its applications.

The scheduling mechanism adopted by XtratuM is compliant with ARINC 653 model: the hypervisor schedules the execution of time-critical partitions following a fixed (static) scheduling scheme, whereas non-critical partitions are executed into dedicated time slices (spare temporal windows).

XtratuM provides a virtual machine similar to the native hardware and allows the execution of a set of partitions (each of which may contain an operating system and its applications). The spatial and temporal isolation properties provided by the hypervisor guarantee the required security level. For example, XtratuM can be used to implement a Multiple Independent Levels of Security (MILS) architecture.

XtratuM hypervisor has been originally designed targeting Intel x86 single-processor systems, and later extended to support SPARC architecture (i.e. Aeroflex Gaisler Leon3 processor). With the addition of Aeroflex Gaisler Leon4 processor support, the possibility of exploiting multicore hardware architecture has been also added [8]. Symmetric multi-processing has been implemented with the addition of proper synchronization mechanisms designed to XtratuM original design.

XtratuM hypervisor supports the x86, LEON2, LEON3 and LEON4 (SPARC v8) architectures. XtratuM support the following execution environments:

- XAL (XtratuM Abstraction Layer) for bare-C applications;
- POSIX PSE51 Partikle RTOS;
- ARINC-653 P1 compliant LITHOS RTOS;
- ARINC-653 P4 compliant uLITHOS runtime;
- Ada Ravenscar profile ORK+;
- RTEMS;
- Linux (supported only for x86 architectures).

7. RT-Java for space Usability Assessment

High level languages, such as Java, use a more robust memory model than the usual languages used for embedded systems development, such as C and C++. The difficulty will using garbage collected languages has been the difficulty of building a garbage collection runtime that does not interfere with the realtime response of the system. JamaicaVM solves this problem by using a patented deterministic, reenterant garbage collector that runs only when a thread allocates and then for a bounded time. No separate garbage collection thread is necessary, so the garbage collector cannot preempt another thread. With this garbage collector and the sematic refinements and APIs defined by the Realtime Specification for Java (RTSJ), the safety advantages of Java can be used for space application that demand realtime response. The virtual machine concept provides an ideal platform for the dynamic system upgrade and reconfiguration that long running space missions need.

7.1 Runtime for Realtime Java Appliciations

JamaicaVM provides tools for compiling a Java program to machine code and linking it with the necessary runtime facilities, such as the garbage collector and platform specific libraries. In order to provide realtime response, JamaicaVM was design with a determinitic garbage collector and implements the refined Java semantics described in the RTSJ and inspired by Ada. In addition, JamaicaVM uses static compilation instead of Just-in-Time (JIT) compilation to ensure that code is has consistent execution behavior.

JamaicaVM implements both priority inversion avoidance mechanisms specified in the RTSJ: priority inheritance and priority ceiling emulation. The former is the default, but the second can be defined for any Java object, upon which the program synchronizes. These mechanisms ensure that a thread in a synchronized block cannot be preempted by a thread with a lower priority than any thread waiting to enter the synchronized block.

In addition to the normal Java threads, which are typically scheduled with a fair scheduler, JamaicaVM offers RealtimeThread and AsyncEventHandler. The RealtimeThread class provides the same functionality of the java.lang.Thread class, except it can be scheduled by the priority preemptive FIFO scheduler. AsyncEventHandler can also be scheduled by the priority preemptive FIFO scheduler, but a handler encapsulates a bit of code that is run each time it is released. A handler can be released by a Timer, an external event, or other software. These two facilties provide the basic framework for writing realtime code.

Conventional Java implementations use a JIT to provide performance in code running in a virtual machine. The code is loaded as bytecode and then compiled after to has been executed some small number of times. This presents an additional problem for realtime systems, because the worst case execution time may be quite a bit longer than the average case. JamaicaVM solves this problem by using static compilation of bytecode to machine code.

JamaicaVM also provides the RTSJ facilities for accessing memory outside the Heap. This is primarily meant for accessing device registers. Memory mapped I/O can be done with conventional Java NIO byte buffers.

7.2 Space Application Framework

A Realtime Java runtime is not sufficient for applications that may run continuously for weeks, months, or even years at a time and then need to be safely updated in flight. Therefore aicas is developing a framework on top of JamaicaVM, that provides secure over-the-air code update and replacement, lifecycle control, and resource management. This framework will enable mission control to dynamically update and replace services during a space mission, without interfering with other code running on the framework.

The framework will include an OSGi runtime environment for basic lifecycle support. The code bundles will be modified to provide stronger separation between bundles and basic resource management. Each bundle will have its own memory management and provide both time and space partitioning between bundles. Multicore systems will be supported with the ability to map bundles to specific cores. A tool will be developed to compile bytecode in a bundle that the framework can dynamically link and load into a running system, so that dynamically loaded code can benefit from static compilation as well.

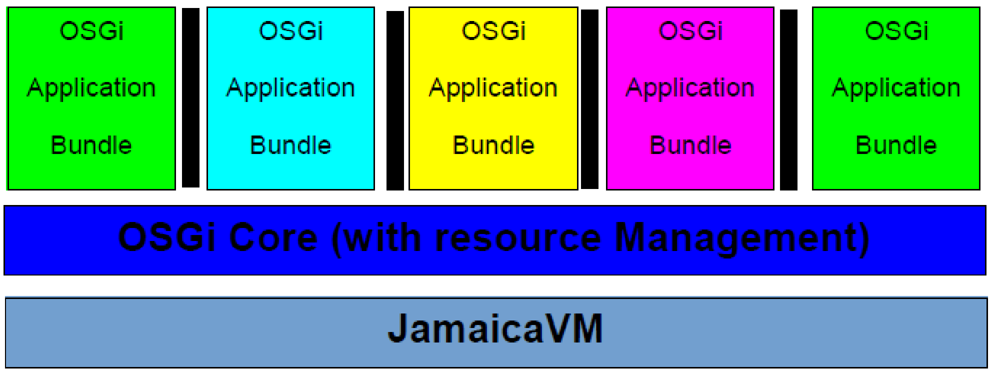


Figure 15: Bundle Management

Each bundle will need to specify both the permission reqired for its task and the percentage of CPU time it requires. The framework will ensure that the bundle does not exceed its time requirements and does not access any API for which it is not authorized. Authorization will use X.509 certificates to validate a bundles source before it is installed.

Some special bundles will be provided to monitor the system, to provide over-the-air bundle installation, and to provide lifecycle control. These bundles will communicate with Mission Control over a secure messaging protocol. Additional bundles will also be provided to support space applications.

8. Conclusions

In the scope of the EMC2 activities it has been evaluated to focus the activities on Time/Space Partitioned environments on top of LEON4 processor. This supports the decision to include in the activity both the PikeOS and the XTRATUM solutions, based on the following considerations:

- They are available for the selected SPARC LEON4 target processor;
- They support a Time/Space Partitioned environment on the selected target processor;
- They implement paravirtualization on the selected target processor;
- They have reached a good maturity level and are fully supported on the selected target processor.

The RTEMS is considered as a possible Guest Operating System in the scope of the virtualized / partitioned system, in particular for the XTRATUM which does not include its own runtime environment (unlike PikeOS which sports its own "PikeOS personality" runtime environment).

9. References

[1] On-Line Applications Research Corporation. RTEMS C User's Guide, Edition 4.10.99.0, for RTEMS 4.10.99.0. Technical Manual, 2013.

[2] Wind River Systems, Inc. VxWorks Reference Manual. Technical manual.

[3] Wind River Systems, Inc. Best Practices: Adoption of Symmetric Multiprocessing Using VxWorks and Intel® Multicore Processors. White paper.

[4] Wind River Systems, Inc. Wind River VxWorks MILS Platform, white paper, 2013.

[5] SYSGO AG, PikeOS Fundamentals, datasheet, 2009.

[6] S. Fisher. Certifying Applications in a Multi-Core Environment: The World's First Multi-Core Certification to SIL 4. SYSGO AG white paper, 2013.

[7] M. Masmano, I. Ripoll, and A. Crespo. XtratuM: a Hypervisor for Safety Critical Embedded Systems. 11th Real-Time Linux Workshop. Dresden. Germany, 2009.

[8] E. Carrascosa, M. Masmano, P. Balbastre and A. Crespo. XtratuM hypervisor redesign for LEON4 multicore processor. ACM SIGBED Review, 11(2), 2014.

10. Abbreviations

Table 1: Abbreviations

Abbreviation	Meaning
EMC^2	Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments
µC	Micro-Controller
PL	PayLoad
RH	Radiation Hardened