De-RISC: the First RISC-V Space-Grade Platform for Safety-Critical Systems

Nils-Johan Wessman[§], Fabio Malatesta[§], Jan Andersson[§], Paco Gomez[¶], Miguel Masmano[¶], Vicente Nicolau[¶], Jimmy Le Rhun[⋆], Guillem Cabo[†,‡], Francisco Bas[†,‡], Ruben Lorenzo[†], Oriol Sala[†], David Trilla[†], Jaume Abella[†]

[§]Cobham Gaisler, Sweden [¶]fentISS, Spain [⋆]Thales Research and Technology, France
[†]Barcelona Supercomputing Center (BSC), Spain [‡]Universitat Politecnica de Catalunya (UPC), Spain

***Abstract*—The increasing needs for performance in the space domain for highly autonomous systems calls for more powerful space MPSoCs and appropriate hypervisors to master them. These platforms must adhere to strict reliability, verifiability and validation requirements since spacecraft for deep space missions are exposed to a harsh environment. Systems must undergo screening and tests against standards for electronic components and software. Unfortunately, currently available space-grade processor components do not meet requirements related to safety that are becoming increasingly important in space applications.**

This paper presents the De-RISC platform, consisting of Cobham Gaisler's RISC-V based SoC, and fentISS' XtratuM Next Generation hypervisor. The platform implements the open RISC-V Instruction Set Architecture, and leverages space SoC IP by Cobham Gaisler, space hypervisor technology by fentISS, multicore interference management solutions by the Barcelona Supercomputing Center, and end user experience and requirement guidance by Thales Research and Technology. At its current state, the platform is already complete and integrated, and starting its validation phase prior to reaching commercial maturity by early 2022. In this paper, we provide details of the platform and some preliminary evidence of its operation.

I. Introduction

Performance requirements for safety and mission-critical systems in the space market grow noticeably to satisfy the needs of increasingly autonomous spacecraft. However, such performance increase must occur within specific boundary conditions such as attaining high integrity levels while allowing mixed criticality operation and providing appropriate performance validation and diagnosis means for system validation and safety measure implementation purposes. Last but not least, export restrictions can impose further constraints on the technologies that can be used for a successful commercialization of space hardware and software.

Multicore processors have arisen as a suitable baseline platform on which to reach high performance levels while respecting boundary conditions for the space market. In particular, the main features that multicores offer are as follows:

- Multicores offer good performance scalability by adding cores (subject to a proper provision and management of shared resources).
- Reliability solutions for one core can be naturally replicated in all cores. Moreover, multicores offer intrinsic redundancy across cores.

However, they are normally implemented using proprietary Instruction Set Architectures (ISAs), such as x86, Arm and SPARC to name a few, and specially for the space market, where SPARC and PowerPC ISAs are particularly popular. Alternative ISAs, such as RISC-V [15], which have been devised to remove proprietary constraints, are not yet adopted for space products.

Moreover, support for performance validation and performance-related safety measures are scarce – if any – since they were not needed for single-core processors and hence, they have not been inherited for multicores. Performance validation of multicore interference can only occur either indirectly building on end-to-end execution times, or partially inferring it from full execution time traces which, nevertheless, may only be obtained with intrusive means, thus destroying the timing behavior intended to be observed. Regarding safety measures to control such interference, they need to build necessarily on pure software means, thus with important overheads at runtime.

On the software side, a number of hypervisors and real-time operating systems (RTOSs) have become industry-ready in recent years on top of multicores for space safety and mission-critical systems, thus completing the hardware/software platform upon which to build space applications. However, those hypervisors and RTOSs necessarily inherit the same ISA as the underlying Multiprocessor System-on-Chip (MPSoC), thus with the same issues related to export restriction.

Overall, while multicores and the corresponding hypervisors/RTOSs for the space domain are already commercial, they build on proprietary ISAs, thus with non-negligible export restrictions, and lack appropriate support for effective multicore interference performance validation and control.

This paper presents the De-RISC platform, which stands for **De**pendable **R**eal-time **I**nfrastructure for **S**afety-critical **C**omputer Systems. De-RISC is the first RISC-V based platform including the Cobham Gaisler's NOEL-V SoC and the fentISS' XtratuM hypervisor. In particular, De-RISC combines multiple technologies and solutions that effectively overcome the challenges for their effective adoption in safety and mission-critical space systems. De-RISC's main features are as follows:

1) It builds upon a NOEL-V based MPSoC by Cobham Gaisler, delivering competitive performance, incorporating appropriate reliability measures for its use in space, and implementing the RISC-V ISA to reduce to a minimum export restrictions.

 DOI 10.1109/SCC49971.2021.00010

2) It incorporates fentISS' XtratuM hypervisor and LithOS RTOS, both of them already qualified for some ISAs, but ported to RISC-V in the context of De-RISC to deliver a complete hardware/software platform usable for space systems.
3) Hardware support to manage multicore interference is incorporated in the form of the Safe Statistics Unit (SafeSU) by the BSC, which provide specific means to monitor and control multicore interference as needed for performance validation and to implement appropriate safety measures.
4) De-RISC platform is undergoing a thorough validation guided by Thales' requirements from the end user perspective, and being assessed with a space use case.

The platform, currently implemented and integrated, is undergoing its first validation steps with the aim of reaching commercial maturity (on FPGA) by March 2022. In the scope of this paper, we introduce the main elements of the De-RISC platform along with some evidence of the effective operation of each of its components.

The rest of this paper is organized as follows. Section II reviews relevant space and RISC-V technologies. Section III presents the NOEL-V based MPSoC including the SafeSU. Section IV presents XtratuM hypervisor and LithOS RTOS. Section V introduces benchmarks and space use cases used for performance validation of the De-RISC platform. Section VI summarizes this work.

II. State of the Art

De-RISC platform consists of a RISC-V MPSoC, hypervisor and RTOS. Therefore, this section presents the state of the art on space MPSoCs, RISC-V microprocessors, and hypervisors/RTOSs suitable for the space domain.

A. Space-grade microprocessors

Mega-constellations and Low Earth Orbit (LEO) missions are either exposed to lower levels of radiation than deep space missions or build upon redundancy at component or system level. Therefore, they can afford using Commercial Off-The-Shelf (COTS) devices since experiencing dependability problems in a single spacecraft is, to some extent, affordable. Instead, deep space missions normally build upon a single spacecraft, and thus, they need space-grade FPGAs or radiation-hardened ASSP devices.

Existing European devices meeting deep space mission needs (i.e. space-qualified) are implementations of processors using the SPARC ISA. Two main component vendors exist for those devices, namely Microchip (formerly Atmel) and Cobham Gaisler. The latter produced introduced the widely used SPARC V7 ERC32 components. In a later generation of products, Atmel shipped the AT697F device, which includes a LEON2FT core with some basic peripheral interfaces such as PCI, UART and GPIO, but lacking interfaces like MIL-STD-1553B and SpaceWire, which are widely used in spacecraft avionics. Those interfaces, however, are available in the Cobham Gaisler's GR712RC device, which includes a dual-core LEON3FT processor [4], thus with a more advanced architecture and peripheral interfaces than the AT697F. The Cobham Gaisler GR740 device outperforms GR712RC since it includes the quad-core LEON4FT processor [5], and includes additional interfaces to further improve the integration level of avionics and through that reduce cost, area and power requirements.

The US market offers also several space-grade microprocessors, mostly based on the PowerPC ISA. Among those devices we can find the DDC/Maxwell SCS750 board, as well as the RAD750 and RAD5545 devices by BAE Systems. Some of those devices have been indeed used even in European missions, such as GAIA (Global Astrometric Interferometer for Astrophysics), which is a space observation mission from the European Space Agency (ESA). Later results show that Cobham Gaisler's GR740 device delivers enough performance to meet the demands of GAIA and hence, replace non-European components in that mission [7]. The MPSoC presented in this paper as part of the De-RISC platform, builds upon GR740 technology and improves it, further delivering higher performance.

There are other efforts to produce space-grade microprocessors such as the H2020 DAHLIA European project. Eventually, those devices will reach commercial maturity, but for now their performance is not yet known.

B. RISC-V microprocessors and cores

Due to its non-proprietary nature, RISC-V ISA has attracted the interest of many institutions, which have developed their own SoCs and cores implementing the RISC-V ISA. Through the RISC-V International web portal one can already find 94 cores and 39 SoCs or SoC platforms [15]. Andes, UC Berkeley, SiFive, Codasip, Syntacore, CloudBEAR, ETH Zurich/U. Bologna, and Microchip are some of the institutions with a larger offering of cores and SoCs. For instance, UC Berkeley and SiFive offer cores and SoCs based on the Rocket one, LowRISC offers the Ibex core, and ETH Zurich and the University of Bologna offer the Ariane and PULPino SoCs. These cores and SoCs, due to the long activity record in the RISC-V arena of the institutions behind them, have become highly popular.

In general, those designs do not offer those features needed for use in safety-related real-time systems, and even less for the space domain, where, on top of safety requirements, the reliability requirements are large due to the harsh environment for operation (e.g. deep space). Among the components available, we identify two of them as relevant in the context of the De-RISC SoC: SiFive's E76-MC embedded processor [16] and Cobham Gaisler's NOEL-V core [6]. The E76-MC offers some features relevant for safety-related systems, but it misses still many others such as watchdog, domain-specific interfaces, etc. The NOEL-V processor core, instead, implements all features needed to be used in space operation. This is the core integrated in the De-RISC SoC by Cobham Gaisler, which is already offered publicly.

C. Space-grade hypervisors and RTOSs

Hypervisors have been deployed in conventional space missions for many years. However, the advent of the so called 'NewSpace' brings further opportunities to hypervisors for the

space domain. Differently to conventional satellites, NewSpace satellites are much lighter (e.g. less than 200 kg instead of few tonnes), smaller, cheaper and less power hungry, and tens, hundreds and even thousands of them are deployed in each mission [18]. Therefore, the number of satellites where space hypervisors can be deployed grows drastically.

In the context of European space missions, fentISS' XtratuM hypervisor is the most popular one. XtratuM guarantees spatial and temporal isolation, as needed for space safety-related real-time systems, thus allowing the efficient deployment of mixed-criticality applications on top of it [10]. XtratuM has already been deployed in 76 satellites of 3 different missions already in space, and it is the hypervisor to be used as part of a number of NewSpace and conventional missions to be launched over the next 5 years. GMV-Portugal offers its Air hypervisor, which has been evaluated in two projects for single-core and multicore SoCs [11] proving its feasibility and sufficient performance. Air is planned to be deployed as part of the INFANTE project. Sysgo's PikeOS [2] is already in the race to be used in space missions, but to the best of our knowledge, PikeOS has not been launched on a space mission so far.

US hypervisor technology has also been used in space missions. Lynx Software Technologies offers the LynxSecure separation kernel hypervisor meeting the needs of applications with strong safety and security requirements. Wind River's hypervisor, analogously to XtratuM, offers support for mixed criticality applications so that they can be integrated onto the same platform without requiring recertification of critical applications. Green Hill's Integrity RTOS, despite not being a hypervisor, meets the most stringent requirements in terms of reliability and security, as needed for space missions.

Among all those hypervisors, XtratuM is the first one reaching commercial maturity for space applications on RISC-V as part of its integration and validation in the De-RISC platform presented in this paper.

III. De-RISC MPSoC

The De-RISC MPSoC architecture has been designed analogous to those of processors in the commercial domain. In particular, the De-RISC MPSoC includes one General-Purpose Processing (GPP) cluster, although the MPSoC is ready to include additional GPP elements if space permits. A GPP is a multicore in itself and, in the case of the De-RISC MPSoC, its GPP includes 4 NOEL-V cores. Varying the number of GPPs and cores per GPP is possible, and ultimately the choice of the most efficient tradeoff depends on the target technology and expected application needs. The schematic of the De-RISC MPSoC is shown in Figure 1.

As shown, the GPP cluster includes, apart from the 4 NOEL-V cores with their respective per-core caches (see Section III-A), a bus infrastructure interfacing cores with components external to the cluster, such as the shared L2 cache and the IO subsystem (see Section III-B). The GPP also includes a debug support module (see Section III-C), the SafeSU to manage multicore interference (see Section III-D), and Performance Monitoring Counters (PMCs). PMCs offer

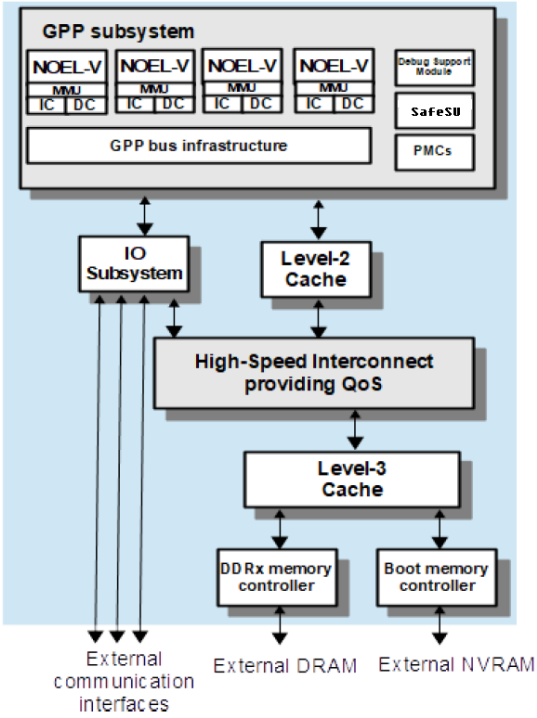

Fig. 1. De-RISC MPSoC.

the usual events such as instruction counts, and cache access counts per cache and type (read/write, hit/miss).

Beyond the L2 cache, the De-RISC MPSoC includes a high-speed interconnect offering quality-of-service (QoS) support. This interconnect is intended to ease the extension of the MPSoC by attaching further GPP clusters and accelerators to it. However, the De-RISC MPSoC, in its first release (the one described in this paper and currently under validation), includes a single GPP, as said before, and no accelerator subsystem since it will be released in a FPGA rather than as an ASIC. While FPGA space is limited, future ASIC implementations will offer extra space for additional computing engines (either GPPs or accelerators), as shown in Figure 2.

Finally, the aforementioned interconnect is attached to a shared L3 cache which, in turn, is connected to the DDR memory controller and to the boot memory controller.

Note that the MPSoC (mainly its peripherals) as well as the NOEL-V cores inherit the fault-tolerance support from the LEON processor family, thus allowing for seamless correct operation despite faults by correcting errors by hardware means. Upon the detection of an uncorrectable error, execution stops to avoid propagating them beyond the actual component affected.

A. NOEL-V RV64 processor core

The Cobham Gaisler's NOEL-V processor core is a 64-bit processor implementing the RISC-V ISA. Its pipeline, shown in Figure 3, is dual-issue in-order, and includes floating point units (supporting floats and doubles), four fully pipelined integer units (two of them in late stages to minimize stalls), support for integer multiplications and divisions, support for

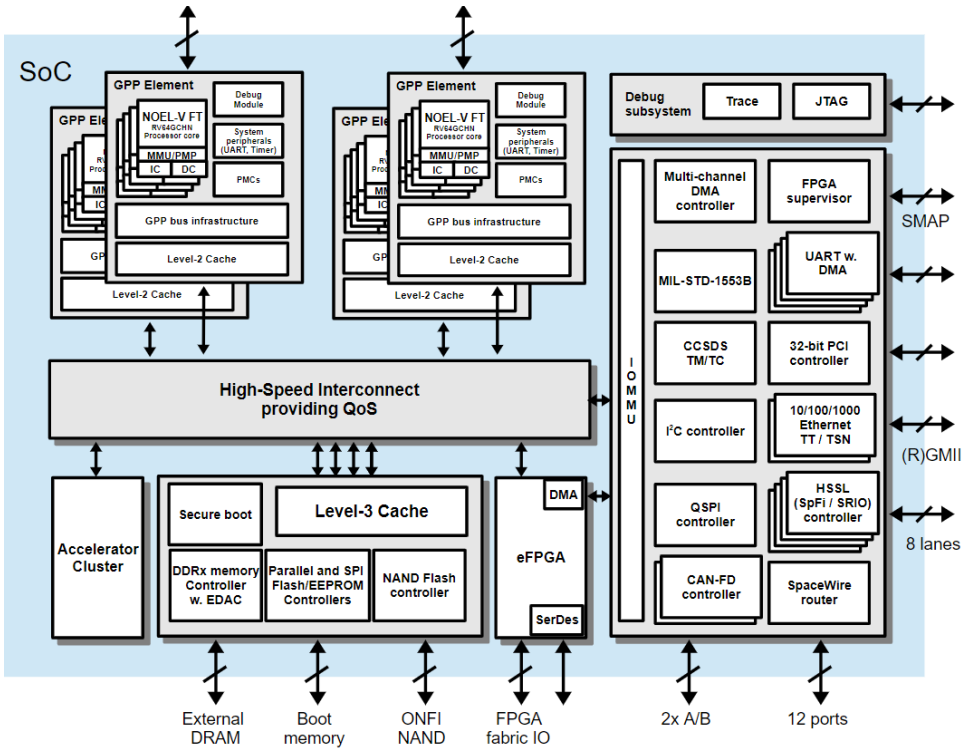

Fig. 2. Envisioned ASIC implementation of the De-RISC MPSoC.

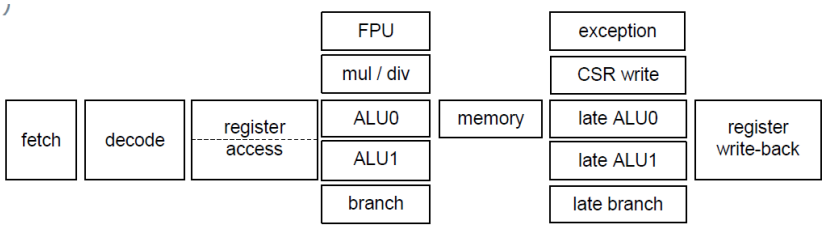

Fig. 3. NOEL-V processor core pipeline.

atomics, a Memory Management Unit (MMU), Physical Memory Protection (PMP), advanced branch prediction units, return address stack, and separate data (DL1) and instructions (IL1) first level cache memories whose size is configurable. The cache controller includes a store buffer allowing back-to-back execution of store instructions, thus reaching a sustained throughput of one store instruction per cycle. The Advanced High-performance Bus (AHB) interface to connect to the GPP bus supports wide data transactions to allow for fast store data transmissions, and fast cache line refill.

Note that, apart from being integrated as part of the De-RISC MPSoC, the NOEL-V processor model is also provided in the Cobham Gaisler IP library (GRLIB). The GRLIB is an integrated set of IP cores that can be connected to the on-chip bus with an appropriate plug&play method and is available in a free open-source version.

B. Communication interfaces and peripherals

Usual peripherals such as timer units, system UARTs and interrupt controllers are included in the De-RISC MPSoC. Those follow standard specifications to guarantee portability across SoCs and software compatibility with drivers matching specifications.

The IO subsystem is architected in a modular way so that the particular IO interfaces implemented can be tailored to match application needs, thus varying the type and number of interfaces integrated. Those include standard interfaces, but also those specific for the space domain, and include the following ones:

- High-Speed Serial Link support through SpaceFibre controllers.
- SpaceWire communication links, connected to an on-chip router.
- 10/100/1000 Mbit Ethernet interfaces.
- MIL-STD-1553B support.
- Controller Area Network Flexible Data-Rate (CAN-FD) interface.
- UART interfaces with DMA support.
- SPI master/slave.
- I^2C master/slave.

- GPIO interface.

Note that, except the parallel PCI interface, the De-RISC MPSoC includes all IO interfaces available in the GR740 microprocessor, including those that would be typically implemented in an external FPGA. In the case of the De-RISC MPSoC, all of them are included on-chip.

Finally, regarding the memory interface, the De-RISC MPSoC supports DDR3 SDRAM with a strong Error Detection and Correction (EDAC) code that tolerates even failures of complete external memory components. NOR and MRAM flash memory devices are supported for boot, which is also possible through the SPI interface. Additionally, NAND Flash memory is also supported as a means of having non-volatile memory storage, which is of particular importance in the space domain.

C. Debug Module

The De-RISC MPSoC includes debug module compatible with the RISC-V debug specification. The debug capabilities include the following:

- Debug connections over several interfaces (including JTAG, Ethernet, and CAN).
- Hart Run Control.
- GPR and CSR register access.
- Program buffer for tracing purposes.
- Triggers (match and instruction count).

D. SafeSU

Last but not least, a novel component of the De-RISC MPSoC when compared with existing space MPSoCs, is its Safe Statistics Unit (SafeSU for short), which provides a number of features to monitor and control multicore interference, thus easing the adherence to specific safety requirements related to real-time, as well to their verification and validation. In particular, the SafeSU includes 3 main features, as described next: (1) the Request Duration Counter, RDC [3], for multicore interference bounding during verification stages, (2) the Cycle Contention Stack, CCS [12], for multicore interference measuring during testing (validation) stages, and (3) the Maximum-Contention Control Unit, MCCU [3], for multicore interference monitoring and controlling during operation.

1) RDC: Request Duration Counter: The RDC [3] allows measuring the highest latency observed for different types of accesses occurring in the AHB component (e.g. a bus) where it is attached to. For instance, in the case of the De-RISC platform, the RDC can monitor the highest latency experienced by read and write operations segregating across data and instruction cache misses, and across read and write operations for the former. Measuring those latencies with a test campaign consisting of different stress tests allows obtaining estimates of the highest latency for each type of event. Those highest latencies can then be used for Worst-Case Execution Time (WCET) estimation to obtain reliable upper bounds to the execution time of real-time tasks.

Additionally, the RDC module of the SafeSU has been extended so that it can be programmed with specific values (e.g. typically the highest latencies per each event type) and used as a safety measure during operation by triggering an interrupt if the latency observed for any event exceeds the pre-programmed upper bound latency for such event. This is useful to identify whether there is some relevant risk of violating any deadline during operation.

2) CCS: Cycle Contention Stack: The CCS [12] measures the actual interference experienced by each core broken down across the cores causing them, thus being a valuable "blaming" mechanism. Figure 4 provides a schematic of the CCS. It is implemented as a table of NxN counters, where N is the core count (4 cores in the case of the De-RISC MPSoC). Counters in a column indicate how much interference a specific core has experienced broken down across contenders, whereas counters in a row indicate how much interference a specific core has caused on each other core. Every cycle, if a specific core is using the shared bus, its grant signal of the AMBA AHB protocol is used to choose the row of counters to update. For each core with the request signal activated (also from the AHB protocol), its counter in that row is incremented. Note that, whenever a core is granted access to the bus, it has both its grant and request signals set, so the corresponding counter in the diagonal will be incremented reflecting how many cycles a specific core has been using the bus effectively.

The CCS can be used both, during validation stages to assess whether pairwise interference is within expected bounds. Note that without the breakdown low interference across some cores could hide excessive interference among others, thus potentially letting undesired behavior escape undetected. The CCS, however, is particularly useful during operation to support safety measures since, upon a deadline overrun, it allows identifying the core(s) causing excessive interference and hence, apply appropriate correction actions (e.g. switching to a different pre-computed task scheduling).

3) MCCU: Maximum-Contention Control Unit: The MCCU [3] organization is analogous to that of the CCS, with NxN counters. However, differently to the CCS, the MCCU is not intended to measure interference but to check whether it exceeds predefined bounds. In particular, users can program the MCCU setting the interference quota that each core can cause in each other core. Whenever such quota is exhausted, the MCCU raises an interrupt so that software layers can take appropriate corrective actions. Such a feature is particularly relevant in mixed criticality scenarios where tasks with specific (high) criticality levels may need protection against shared resource clogging by lower criticality tasks.

The MCCU can operate in two different modes, being each one devised for a different phase of the product lifecycle. If upon an event in which a core c_i causes interference in a core c_j, the MCCU counters are decreased by the maximum latency such event could have (e.g. as given by the RDC), then the MCCU fits timing validation needs. In this case, interference events are assumed to cause the maximum interference possible, thus exhausting quotas as fast as realistically possible, hence indicating whether deadline violations could be possible. If, instead, the MCCU uses actual interference to decrease quotas (e.g. interference recorded in the CCS), then the MCCU can be used as a safety measure during operation. In that case, interrupts are only raised when interference has effectively exceeded quotas.

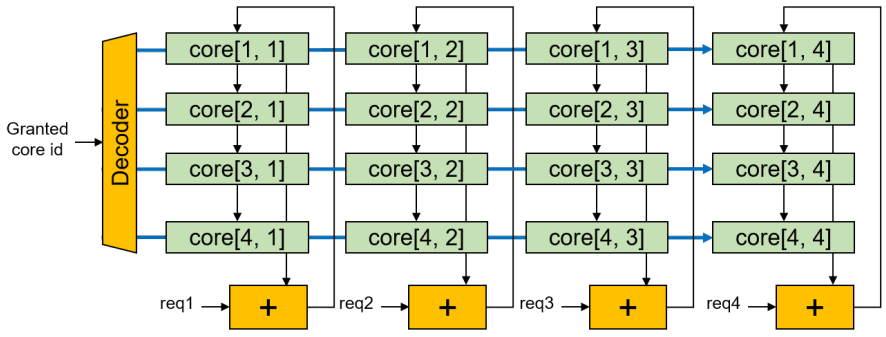

Fig. 4. High-level schematic of the CCS.

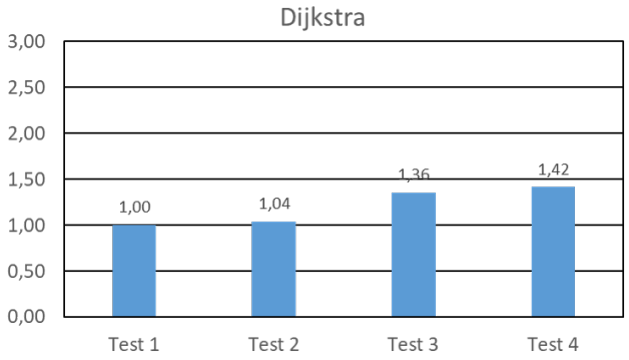

Fig. 5. Normalized execution time for Dijkstra on De-RISC MPSoC.

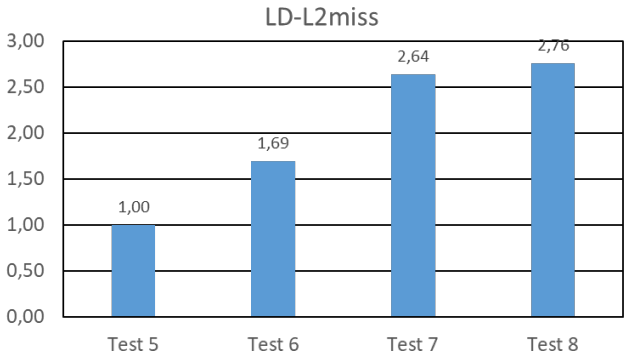

Fig. 6. Normalized execution time for LD-L2miss on De-RISC MPSoC.

E. Stress tests

We perform an initial stress test of the SoC using the three benchmarks described in Section V-A, namely Dijkstra, LD-DL1hit and LD-L2miss. In particular, we use Dijkstra as task under analysis in the first set of experiments (see Table I), and the other two benchmarks as contenders, varying the number of contenders of each type as shown in the following table, with increasing interference.

TABLE I
DIJKSTRA WORKLOADS

TEST	Core 0	Core 1	Core 2	Core 3
Test1	Dijkstra	LD-DL1hit	LD-DL1hit	LD-DL1hit
Test2	Dijkstra	LD-DL1hit	LD-DL1hit	LD-L2miss
Test3	Dijkstra	LD-DL1hit	LD-L2miss	LD-L2miss
Test4	Dijkstra	LD-L2miss	LD-L2miss	LD-L2miss

In particular, *test1* corresponds to the case where Dijkstra experiences no interference in the access to memory since all contenders keep data local in their respective core-local DL1. *Test2* replaces LD-DL1hit by LD-L2miss in Core 3, thus with high interference but only from one core. *Test3* moves from one to two memory-aggressive contenders. Finally, *Test4* includes three memory-aggressive contenders.

Figure 5 shows the normalized execution time for Dijkstra in those 4 scenarios. As shown, as the number of memory contenders (LD-L2miss micro-benchmarks) increases, so does the execution time. However, Dijkstra is not memory intensive, and hence, the impact in execution time due to multicore interference is low. Hence, our preliminary conclusion is that the SoC allows scaling performance for applications taking advantage of local caches despite having aggressive contenders.

TABLE II
LD-L2MISS WORKLOADS

TEST	Core 0	Core 1	Core 2	Core 3
Test5	LD-L2miss	LD-DL1hit	LD-DL1hit	LD-DL1hit
Test6	LD-L2miss	LD-DL1hit	LD-DL1hit	LD-L2miss
Test7	LD-L2miss	LD-DL1hit	LD-L2miss	LD-L2miss
Test8	LD-L2miss	LD-L2miss	LD-L2miss	LD-L2miss

Analogous tests have been performed replacing Dijkstra by LD-L2miss as unit of analysis, as shown in Table II. These tests are expected to expose much higher multicore interference since the task under analysis is memory intensive by performing sustained read operations from memory (missing in all cache levels). As shown in Figure 6, the slowdown experienced by LD-L2miss is much higher than that experienced by Dijkstra. For instance, with 3 contenders, Dijkstra execution time increases by 1.42X whereas LD-L2miss execution time increases by 2.76X. However, despite the much higher slowdown due to the saturation of the target shared resource (DDR memory), LD-L2miss still achieves some performance gains with respect to the single-core case where all 4 copies had been run serially in a single core, thus with an accumulated execution time of 4X w.r.t. a single execution in isolation [13].

Hence, we note that the SoC throughput increases w.r.t. the single-core setup even when using programs highly aggressive in the use of shared resources.

Overall, our preliminary results already evidence that the De-RISC MPSoC integration has been successful, and performance trends are appropriate. Part of our future work as part of the validation phase consists of performing a much larger test campaign with the aim of validating each individual feature in the MPSoC, as well as their integration. The results shown in this paper are preliminary results and additional results will be made available after implementation of planned upgrades to the De-RISC platform's bus infrastructure and Level-2 cache.

IV. De-RISC Hypervisor

Within the software stack of De-RISC, the XtratuM Next Generation (XNG) hypervisor [10] and the ARINC-653 compatible LithOS run-time, both by fentISS, have been ported successfully to the hardware RISC-V architecture.

XNG is a complete rewrite of the XtratuM hypervisor that builds on the fentISS accumulated expertise after working for more than 13 years in the development of the XtratuM variants.

Making use of the mechanisms provided by the hardware, XNG provides time and space isolated execution environments, also known as partitions and minimizes the interference between cores caused by the access to common resources.

A. XNG Overview

The XtratuM building blocks provide the services needed by safety-critical systems such as:

- hypervisor and partition management: through the invocation of hypercalls;
- support for normal and system partitions: a normal partition can manage and monitor its own state whereas a system partition can manage and monitor the overall system and other partitions;
- resource virtualization: through the Partition Virtual Execution Environment (PVEE), making virtual resources available to a partition as if it were the only one using a given resource in the system (CPU, FPU, interrupt controller,...);
- temporal partitioning: based on a cyclic scheduling policy that can be selected among a set of configurable schedules during the hypervisor initialization;
- spatial partitioning: based on the support provided by the hardware, typically the Memory Management Unit (MMU);
- Inter-Partition Communication (IPC): through sampling and queuing ports, inspired in the ARINC-653 standard and implemented via messages as contiguous blocks of data of finite length;
- the Health Monitor (HM) service: which detects faults in the hardware and in XNG itself and responds according to the specified configuration;
- observability of the system: mandatory in space domain applications, through xci, xcon and xtraceviewer observability development tools;

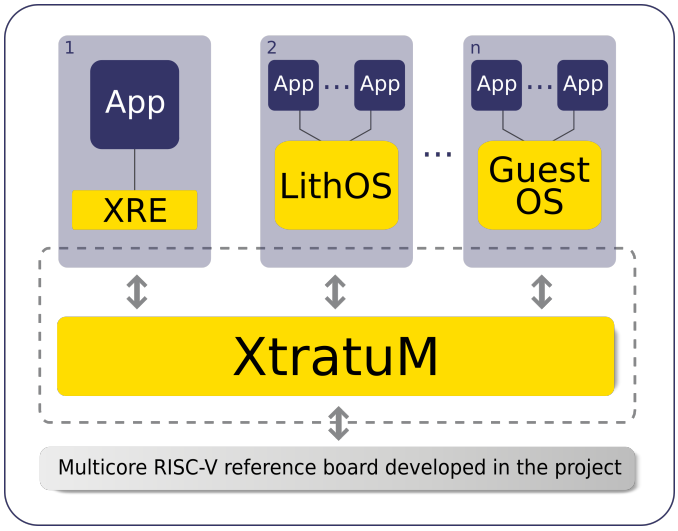

Fig. 7. XtratuM running on top of the De-RISC hardware platform.

- and the XtratuM Configuration File (XCF): a set of XML files to allow the system integrator to configure the system, that is, the hypervisor, the partitions, the scheduling, the health monitor events/actions, the tracing facilities, the resource allocation to each partition and to the hypervisor, or the IRQs and/or I/O devices delegation to a partition, among others.

Figure 7 shows a typical configuration with XtratuM running on top of the hardware platform. The block diagram illustrates a XtratuM-based system architecture with three partitions running on it. Partition number 1 contains the XtratuM Run-time Environment (XRE) together with an application. The other two partitions (2 & 3) are managed by their corresponding guest operating systems, executing their respective application(s).

XRE is a C library component provided together with XtratuM for simplifying the interaction of the application code with the low-level management required by the PVEE, which is a virtual representation of the underlying hardware. XRE allows the execution of bare metal applications over XNG without the need of any guest operating system.

In the partition number 2, LithOS is running as a para-virtualized guest operating system which uses the services provided by XtratuM to offer an ARINC-653 APEX to the concurrent applications running on top. ARINC-653 Application/Executive (APEX) is a standard interface for avionic software applications allowing to build partitioned systems. While XtratuM provides the partitioning and inter-partition communication mechanisms, LithOS builds on top of them to provide a complete run-time solution by including processes, inter-process communication, synchronization mechanisms, error handling and other functions expected from an ARINC-653 compliant run-time. Thus, the combination of XtratuM and LithOS allow a user to build partitioned systems based on ARINC-653.

The partition number 3 is managed by a third-party guest operating system, such as RTEMS or Linux. To run any other

guest OS, a so-called Board Support Package (BSP) specific to the De-RISC platform is required. The BSP is a software layer that interfaces the guest operating system with the hypervisor and hardware.

B. XNG HM: the key for software safety

XNG incorporates the latest hypervisor services developed by fentISS to match the needs of safety-critical systems. Specifically, the Health Monitor (HM) service allows the system integrator to define a Fault Detection, Isolation, and Recovery (FDIR) policy specific for the system and for each partition event. This policy is defined in the XCF.

The HM is in charge of detecting (the event), reacting (with an action) and reporting (in the HM log) fault states from either the hardware, the partitions or internally generated by the hypervisor. It aims at discovering faults at an early stage, trying to solve or confine the faulty subsystem to avoid a failure or to reduce its potentially harmful effects.

The operation of the HM is described as follows:

1) Faults are reported to the HM service as HM events: Regardless of which is the component that reports the fault, each event has an associated faulting element, either the hypervisor or any of the configured partitions.
2) When the faulty element is a partition, the HM applies a mapping function to convert from architecture-specific events to generic events, thus easing the development of portable partitions.
3) The HM logs the generic event and the associated information in the logging space.
4) The HM executes an action configured in the XCF, which can be a specific action for each HM event. Specific actions can also be configured for each event and every partition.
5) Partitions can access the HM logs to process the collected information according to the system needs.

The HM events can be grouped in two categories, hypervisor events and partition events. The hypervisor events include:

- hypervisor-generic events, raised when certain execution conditions are detected, e.g. due to internal sanity or robustness checks;
- architecture specific events, caused by the notification of a processor exception during the execution of the hypervisor.

The partition events comprise:

- partition-generic events, which are not directly raised by the hypervisor but raised by the partition by means of the invocation of the specific hypercall;
- architecture specific events, which indicate a fault in the partition detected by the notification of a processor exception.

The HM actions are grouped in the following categories:

- actions changing the execution state of the partitions (e.g. halting or warm/cold resetting it), thus affecting just the partition for which the HM event is reported.
- Actions changing the execution state of the hypervisor (e.g. halting or warm/cold resetting it), thus affecting the complete system.

When the HM detects an unrecoverable situation, executes a transition to a safe halt state automatically. During this transition, the hypervisor does not rely on any component external to the CPU itself. In this state, the hypervisor executes an endless loop with IRQs disabled, waiting for an external entity to solve the situation (e.g. a watchdog time-out resetting the whole system).

An illustrative example available at the official De-RISC YouTube channel (accessible through De-RISC website [8]) shows the behavior of the XtratuM HM service. In this example, shown in Figure 8, four partitions are running in the on-board computer (OBC) of a micro satellite:

- a critical partition (P0), executing the telemetry and telecommand (TM/TC) subsystem;
- another critical partition (P1), running the Attitude and Orbit Control System (AOCS);
- a non-critical partition, (P2) corresponding to the satellite payload (e.g. image acquisition and processing);
- one more critical partition (P3) that monitors the OBC.

Imagine that, due to a bug in P2, at some point during the image processing execution an access is attempted to a memory area belonging to partition P1 (critical AOCS partition). The access could write some data in the P1 memory area, possibly corrupting the value of one or more variables used to control the satellite orbit. This could cause the propulsion subsystem to act accordingly, based on a incorrect positioning data, causing the loss of the satellite and the mission.

To avoid this situation, the system integrator can configure the HM to catch this event and to act accordingly. In the example, the HM is configured to cold reset P2 (payload partition) when this event occurs, while P3 is monitoring the HM log to detect recurrences of this HM event in P2. When P3 detects that P2 has caused more than three HM events of this type, it switches the system scheduling to a safe schedule (previously defined by the system integrator). In this scheduling, P2 is not included for the sake of the system safety. Having the system in a safe state, P0 (TM/TC partition) notifies the ground segment about the faulty behavior of the payload partition.

C. XNG software development tools

XtratuM development tools have also been ported to support the De-RISC architecture. These are divided in basic software development tools and advanced integration tools. The basic tools include configuration tools (xcparser, elfbdr) and observability tools (xci, xcon, xtraceviewer) whereas the advanced integration tools comprises XPM (XtratuM Project Manager) and the Xoncrete scheduling analysis tool.

D. ECSS development process

The XNG and LithOS porting have been carried out following partially the ECSS development process to ease the future space qualification at ECSS level B. This will make the hypervisor ready for the aerospace market almost right after the project completion.

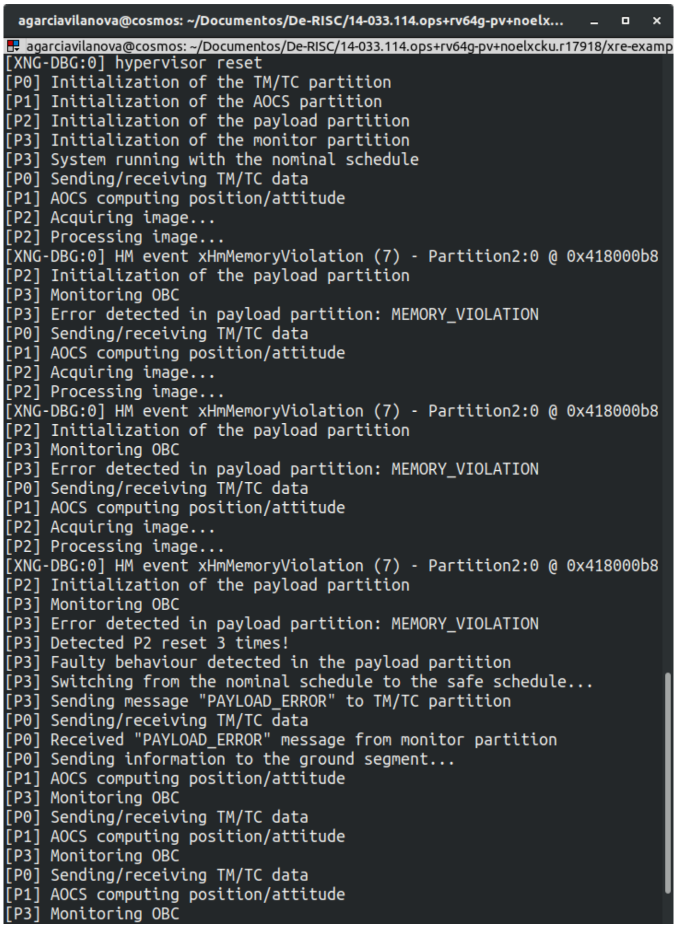

Fig. 8. HM example available at the official De-RISC YouTube channel [8].

V. Performance Validation

Several test applications are in the process of being ported onto the De-RISC platform with the aim of providing evidence of its proper operation with relevant space application as well as relevant benchmarks, as explained in this section.

A. Basic Validation

The first steps towards the validation of the De-RISC platform consist of a number of tests to validate the functionality of the processor core and its compliance with the RISC-V ISA standard, and to measure its performance with reference benchmarks. For the former, we build on those tests provided by RISC-V International, which have been successfully passed. This gives evidence that the basic RISC-V ISA as well as the extensions implemented work properly (e.g. M, A, and FD extensions for integer multiply/divide, atomics and float/double operation respectively).

Regarding performance, we have integrated some benchmarks and obtained the following figures for the high-performance configuration of the NOEL-V core at 80 MHz:

- EEMBC CoreMark benchmark: 4.41 for CoreMark/MHz.
- Dhrystone benchmark: 144,092.2 Dhrystones/second.
- Whetstone benchmark (C Converted Double Precision Whetstones): 14.3 Millions of Whetstones/second.

TABLE III
RAMspeed (Generic) v2.6.0 results (integer and writing).

Block size	Bandwidth	Block size	Bandwidth
1 Kb	458.82 MB/s	256 Kb	68.64 MB/s
2 Kb	469.82 MB/s	512 Kb	53.75 MB/s
4 Kb	475.54 MB/s	1024 Kb	53.80 MB/s
8 Kb	478.30 MB/s	2048 Kb	53.82 MB/s
16 Kb	478.79 MB/s	4096 Kb	53.83 MB/s
32 Kb	460.67 MB/s	8192 Kb	53.81 MB/s
64 Kb	445.98 MB/s	16384 Kb	53.73 MB/s
128 Kb	437.01 MB/s	32768 Kb	53.57 MB/s

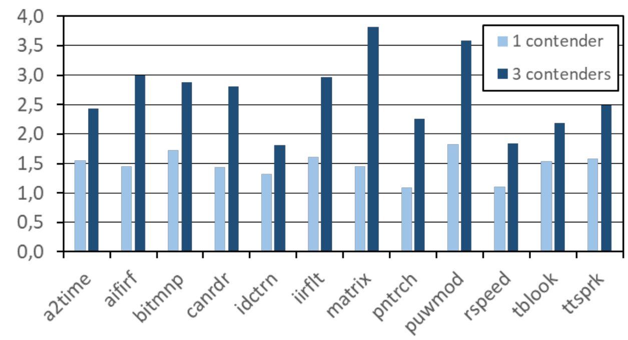

Fig. 9. Normalized execution time for EEMBC Autobench benchmarks on De-RISC MPSoC.

We have also integrated RAMspeed (generic) v2.6.0 benchmark, whose results are shown in Table III.

Other benchmarks representative of a variety of applications, such as the TACLeBench benchmark suite [9] and the EEMBC Autobench [14] have already been integrated. Illustrative results in Section III-E use one of the TACLeBench (Dijkstra). Since EEMBC Autobench are larger and more representative than TACLeBench, we have collected some more detailed results with those to assess the expected performance when executing a variety of embedded applications. We have set data footprints so that an intense use of shared resources (cache memories and bus) is made. In particular, we consider each benchmark running in isolation, with 1 contender and 3 contenders. Contenders are always the very same benchmark evaluated to generate high contention scenarios (i.e. all cores attempt to access shared resources at the same time). Execution time results normalized w.r.t. the isolation case are shown in Figure 9. As depicted, despite contention may be high, the MPSoC effectively allows sharing resources with slowdowns well below 2X and 4X with 1 and 3 contenders respectively, thus achieving a significantly higher throughput than the one that would be reached if running benchmarks serially in one core.

Last but not least, a number of purpose-specific micro-benchmarks have been developed to stress the use of different resources in the MPSoC. For instance, two micro-benchmarks performing sustained read accesses have been used in Section III-E together with the aforementioned TACLeBench. One performs sustained hits to the DL1 (LD-DL1hit), whereas the other misses in DL1 and L2 and accesses memory for each load access (LD-L2miss).

B. Compute-intensive space application

Apart from those tests intended primarily to assess the performance of a single core, described in previous subsection, the multicore platform needs also being evaluated. For that purpose, we have used two types of tests: one building on the micro-benchmarks mentioned before, and another building on a lossless multispectral image compression algorithm, standardized by the Consultative Committee for Space Data Systems as CCSDS-123 [17].

Micro-benchmarks are deployed in a multicore setup in different ways, combining those hitting and missing in the different cache levels, with write and read operations, and activating or deactivating L2 cache partitioning. By smartly selecting the workloads, corner performance cases can be assessed. Regarding the lossless multispectral image compression algorithm, it has been used not only to assess multicore performance, but to evaluate the hypervisor and/or RTOS-based resource management services. In particular, an instance of the application is deployed in each core. Since the application is data intensive, the four copies running together contend for the access to the shared bus, shared cache and main memory.

Overall, those workloads will serve the purpose of evaluating isolation techniques, as well as the features of the SafeSU such as the CCS and the MCCU.

C. Representative Space system application set

The most general validation test consists of the deployment of a single application using all the cores. In particular, a use case application representative of a satellite system is being deployed. The tasks of that application building on standard inter-partition communication mechanisms.

The particular application used is the Command & Data Handling Platform [1]. Such application is intended to be deployed on a multicore, hence it is appropriate to assess the impact of multicore interference, the degree of time isolation provided by the De-RISC platform, and those De-RISC platform components intended to provide safety services related to real-time performance, such as the CCS and MCCU part of the SafeSU. Given that the application has already been used in the evaluation of the LEON4FT GR740 space-grade microprocessor and XtratuM hypervisor in the context of the EMC^2 ECSEL project, it will allow comparing such platform against De-RISC one on a fair basis.

VI. Conclusions

Critical applications in the space domain have increasing requirements due to their high autonomy and system complexity. In particular, requirements relate to performance, safety, validation, reliability and commercial needs. Existing microprocessors effectively meet some of those requirements, but to the best of our knowledge, none of them meets them all.

This paper presents De-RISC, a new hardware and software platform for space applications, meeting all the aforementioned requirements. In particular, the De-RISC platform includes a Cobham Gaisler's MPSoC based on the NOEL-V core, as well as fentISS' XtratuM hypervisor, both of them compliant with RISC-V ISA. Moreover, the MPSoC also includes BSC's SafeSU to manage multicore interference. Last but not least, an extensive validation plan has been set to test the different performance considerations of the platform. The De-RISC platform is nowadays fully integrated and starting its validation phase, and is expected to reach the market in early 2022.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement EIC-FTI 869945. BSC work has also been partially supported by the Spanish Ministry of Science and Innovation under grant PID2019-107255GB.

References

[1] D. Andreetti, F. Federici, V. Muttillo, D. Pascucci, and L. Pomante. Analysis and design of a command & data handling platform based on the LEON4 multicore processor and PikeOS hypervisor, 2017. http://hdl.handle.net/11697/137061.

[2] Jan Bredereke. A survey of time and space partitioning for space avionics. *Technical Report, City University of Applied Sciences Bremen*, 2017.

[3] J. Cardona, C. Hernandez, J. Abella, and F. J. Cazorla. Maximum-contention control unit (mccu): Resource access count and contention time enforcement. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 710–715, March 2019.

[4] Cobham Gaisler. LEON3FT Fault-tolerant processor. https://www.gaisler.com/index.php/products/processors/leon3ft (accessed Feb-2021).

[5] Cobham Gaisler. LEON4 processor. https://www.gaisler.com/index.php/products/processors/leon4ft (accessed Feb-2021).

[6] Cobham Gaisler. NOEL-V processor. https://www.gaisler.com/index.php/products/processors/noel-v (accessed Feb-2021).

[7] Cobham Gaisler. RTEMS SMP executive summary, development environment for future leon multi-core. *RTEMSSMP-ES-001*, 2, 2015. http://microelectronics.esa.int/gr740/RTEMS-SMP-ExecSummary-CGAislerASD-OAR.pdf.

[8] De-RISC Consortium. De-RISC website, 2021. https://www.derisc-project.eu/ (accessed Feb-2021).

[9] Heiko Falk, Sebastian Altmeyer, Peter Hellinckx, Björn Lisper, Wolfgang Puffitsch, Christine Rochange, Martin Schoeberl, Rasmus Bo Sorensen, Peter Wägemann, and Simon Wegener. Taclebench: A benchmark collection to support worst-case execution time research. In Martin Schoeberl, editor, *Proc. 16th International Workshop on Worst-Case Execution Time Analysis (WCET'2016)*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, July 2016.

[10] fentISS. XtratuM Hypervisor. https://fentiss.com/products/hypervisor/ (accessed Feb-2021).

[11] B. Gomes, D. Silveira, L. Gouveia, and L. Mendes. Air hypervisor using RTEMS SMP. In *European Workshop on On-Board Data Processing (OBDP2019), ESTEC (ESA)*, 2019.

[12] J. Jalle et al. Contention-aware performance monitoring counter support for real-time MPSoCs. In *IEEE Symposium on Industrial Embedded Systems (SIES)*, 2016.

[13] J. Nowotsch and M. Paulitsch. Leveraging multi-core computing architectures in avionics. In *2012 Ninth European Dependable Computing Conference*, pages 132–143, 2012.

[14] J. A. Poovey, T. M. Conte, M. Levy, and S. Gal-On. A benchmark characterization of the eembc benchmark suite. *IEEE Micro*, 29(5):18–29, 2009.

[15] RISC-V International. RISC-V International website. https://riscv.org/.

[16] SiFive Inc. SiFive E76-MC Manual v19.08p0, 2019. https://sifive.cdn.prismic.io/sifive%2F08e49813-ffcc-4a6c-9d70-ba2add7ebbe6_sifive+e76-mc+manual+v19.08.pdf (accessed Feb-2021).

[17] The Consultative Committee for Space Data Systems (CCSDS). Lossless multispectral and hyperspectral image compression recommended standard. CCSDS 123.0-B-1, 2012.

[18] Wikipedia. Small satellite, 2021. https://en.wikipedia.org/wiki/Small_satellite (accessed Feb-2021).