Journal of Aerospace Information Systems
Vol. 16, No. 11, November 2019

Leveraging the Openness and Modularity of RISC-V in Space

Stefano Di Mascio,[*] Alessandra Menicucci,[†] and Eberhard Gill[‡]
Delft University of Technology, 2629 HS Delft, The Netherlands
and
Gianluca Furano[§] and Claudio Monteleone[¶]
European Space Agency, 2200 AG Noordwijk, The Netherlands

DOI: 10.2514/1.I010735

This paper proposes a roadmap to address present and future needs in space systems with RISC-V processors. RISC-V is an open and modular instruction set architecture, which is rapidly growing in popularity in terrestrial applications. To satisfy different applications with contrasting requirements in satellite data systems, four different types of processors are identified: 1) low-area/low-power microcontrollers, 2) on-board computers, 3) general-purpose processors for payloads, and 4) enhanced payload processors for artificial intelligence. Several solutions based on RISC-V are proposed for each of these types of processors and compared with proprietary commercial-off-the-shelf and space-grade solutions. An extensive analysis of the results available from literature is conducted to show that RISC-V has the potential to solve such a wide range of needs. This paper will also show the unprecedented number of open-source implementations and models that were developed in a relative short time on a single instruction set architecture. Future space systems could benefit from many of those developments, and this work identifies and highlights what is still missing to satisfy the specific needs of processors for space, especially in terms of fault tolerance and technology readiness level.

I. Introduction

WHILE in many cases space systems employ novel technologies and spin-off processes are in place to introduce them in terrestrial applications, processors for terrestrial applications are more advanced (e.g., in terms of performance) compared with the ones employed in space thanks to larger markets available and shorter time to market. Furthermore, processors in space face unique challenges due to the effects of the space environment, like ionizing radiation causing single event effects (SEEs) and total ionizing dose (TID) effects, and to the unfeasibility of direct physical maintenance (e.g., replacement of a component). For this reason space systems are typically based on radiation-hardened by design or by process (RHBD and RHBP) processors, explicitly designed and validated for usage in space. Rad-hard processors typically lag more than a decade behind their commercial counterparts in terms of performance, and the gap is widening every year (e.g., the commercial PowerPC-750 roadmap showed a performance improvement of ~2400× from the early 1990s to 2005 [1], whereas the rad-hard version was improved by only ~300× [2]). This was mainly due to lower frequencies and smaller caches, both due to larger technology nodes and use of RHBD cells. Furthermore, processors used in space are typically slower due to the long qualification process for space-grade components and to a risk-averse behavior of space industry that prefers components and technologies that already have a history of proven spaceflight. For this reason, developments are classified according to technology readiness level (TRL) [3].

The idea of a "spin-in" of commercial-off-the-shelf (COTS) processors for terrestrial applications in space systems is becoming increasingly popular [4,5], given the niche size of the space market compared with other embedded markets like mobile and automotive. By "spin-in" we mean the reuse in space systems of developments meant for terrestrial applications, and this can be put in place at different levels of a space system:

1) Subsystem: The spin-in of a full subsystem (e.g., a single-board computer) provides the maximum reuse of developments from terrestrial applications. The drawback in this approach is the total lack of specific solutions for space in the manufacturing process. Also, it leaves room for fault-tolerant (FT) solutions only at software and at system level (e.g., spare boards). Up to now this approach is limited to low-cost demonstration/university missions.

2) Component: An application-specific integrated circuit (ASIC) developed for other applications is included in a space subsystem. The work in [5] shows that the "cost of ownership" of COTS components (composed of procurement cost, cost of lot acceptance test, cost of FT mechanisms, and their validation) is potentially higher than a space-grade equivalent. Therefore, Ref. [5] concludes that only the contribution of COTS components to system performance can justify their use in space avionics.

3) Intellectual property (IP) core: To cope with an extremely high level of complexity, licensable and reusable blocks (IP cores) are employed during the design of a system-on-chip (SoC). IP cores for terrestrial applications can be included as is or after ad-hoc modifications to increase their fault tolerance in an ASIC or field-programmable gate array (FPGA). Reference [6] describes how an e600 PowerPC Core has been employed in a space-grade payload processor.

Even when the development of the processor starts from scratch, the designers can either choose to define their own instruction set architecture (ISA) or use an ISA already available. ISAs are (sometimes expensive) IPs that enable the reuse of a certain software ecosystem (compilers, debuggers, integrated development environments). Furthermore, maintaining a software ecosystem is by far the biggest industrial challenge in the development of a processor [8]. The first processors employed in space were based on ISAs specifically designed for airborne computers meant to be used in military avionic systems (e.g., MIL-STD-1750 [9]). With the introduction of proprietary commercial ISAs like x86, MIPS,[**] and

Received 10 March 2019; revision received 31 May 2019; accepted for publication 10 July 2019; published online Open Access 5 September 2019. Copyright © 2019 by Delft University of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

[*]Ph.D. Candidate, Faculty of Aerospace Engineering, Space Systems Engineering; s.dimascio@tudelft.nl.
[†]Assistant Professor, Faculty of Aerospace Engineering, Space Systems Engineering; a.menicucci@tudelft.nl.
[‡]Professor, Faculty of Aerospace Engineering, Space Systems Engineering; e.k.a.gill@tudelft.nl.
[§]On-Board Computer Engineer, Microelectronics and Data Systems Division, European Space Technology Centre, Keplerlaan 1; gianluca.furano@esa.int.
[¶]On-Board Computer Engineer, Microelectronics and Data Systems Division, European Space Technology Centre, Keplerlaan 1; claudio.monteleone@esa.int.
[**]"MIPS" is an acronym universally accepted to indicate both the "Microprocessor Without Interlocked Pipelined Stages" ISA and the millions of instructions per second metric to measure processing speed.

PowerPC the space sector could rely on software ecosystems and developments from the commercial field. In the nineties, the European Space Agency (ESA) chose SPARC for its ERC32 and LEON processors, as it was the only solution available at that time providing both openness and diffuse software support. Currently the European space industry (as well as a large part of the worldwide space community) is using LEON-based SoCs in all ongoing and planned missions [10]. The market share evolution of ISAs for space application is shown in Fig. 1.

The following subsections explain how openness and modularity make RISC-V a good candidate at each level of spin-in identified.

A. Openness

Regardless of the level where the spin-in takes place, the main advantage is the reuse of the software ecosystem and of the results and experience of a much larger community working on terrestrial applications. For instance, a bug in a compiler or in model of a processor can be found more easily compared with a compiler or a model of a processor used only for space applications. The advantage is even stronger when this community is built around an open and free ISA, as everyone can contribute to advancing the state-of-the-art. For this reason, wide adoption by industry and academia is crucial. These considerations were critical for the adoption of SPARC by ESA in the nineties. However, nowadays SPARC lost momentum in terrestrial applications, mainly because of the predominance of the proprietary x86. New markets with different requirements in terms of energy efficiency (e.g., mobile devices) have led to the success of proprietary ISAs from ARM. Nevertheless, RISC-V (an open and free ISA like SPARC) has risen in popularity in recent years. RISC-V was originally developed by UC Berkeley to support computer architecture research and education oriented at hardware implementations, because a flexible, open, and free ISA fit for such purpose was not available [11]. The spread of an open and free ISA like RISC-V already enabled a vast field of research activities for terrestrial application (security, artificial intelligence [AI], etc.), as accessing proprietary architectures is costly and limits what can be done with a certain product or within a certain research activity. The availability of a software ecosystem supported by a large open community ignited an unprecedented amount of developments, with several announcements and/or releases of open-source implementations. RISC-V is backed also by big players of the commercial field, as companies like Google, Qualcomm, IBM, NVIDIA, Samsung, and Western Digital are members of the RISC-V Foundation. Following this trend the MIPS ISA is planned to be licensed free of royalties starting in 2019. As a matter of fact, concerns are growing about monopolistic positions in the embedded market, as ISA owners protect their IP not allowing freely available implementations and do not allow free-market competition from many core designers, thus ultimately preventing reuse. The adoption of a popular free and open ISA can thus lead to shorter time to market and lower costs from reuse. The space industry can spin-in developments from other industries, focusing efforts mainly on improvements concerning specific needs in space applications and without wasting efforts on other activities. However, implementations of open ISAs can also be proprietary. Below we describe how openness can help enhancing two crucial aspects of space systems: security and fault tolerance.

1. *Security*

DARPA funded RISC-V in its very beginning and is continuing to fund other activities related to the spin-in of open-source IPs in trustable electronic systems [12]. The reason behind this is that open-source IPs and open ISAs can reduce the costs, time, and complexity required for secure and trusted custom SoC design, as detailed information available with open-source IP can be found by inspection. Ad-hoc improvements or modifications for secure and trusted application are much easier, avoiding the need to design everything from scratch and focusing only on the critical part and thus ultimately increasing reuse [12].

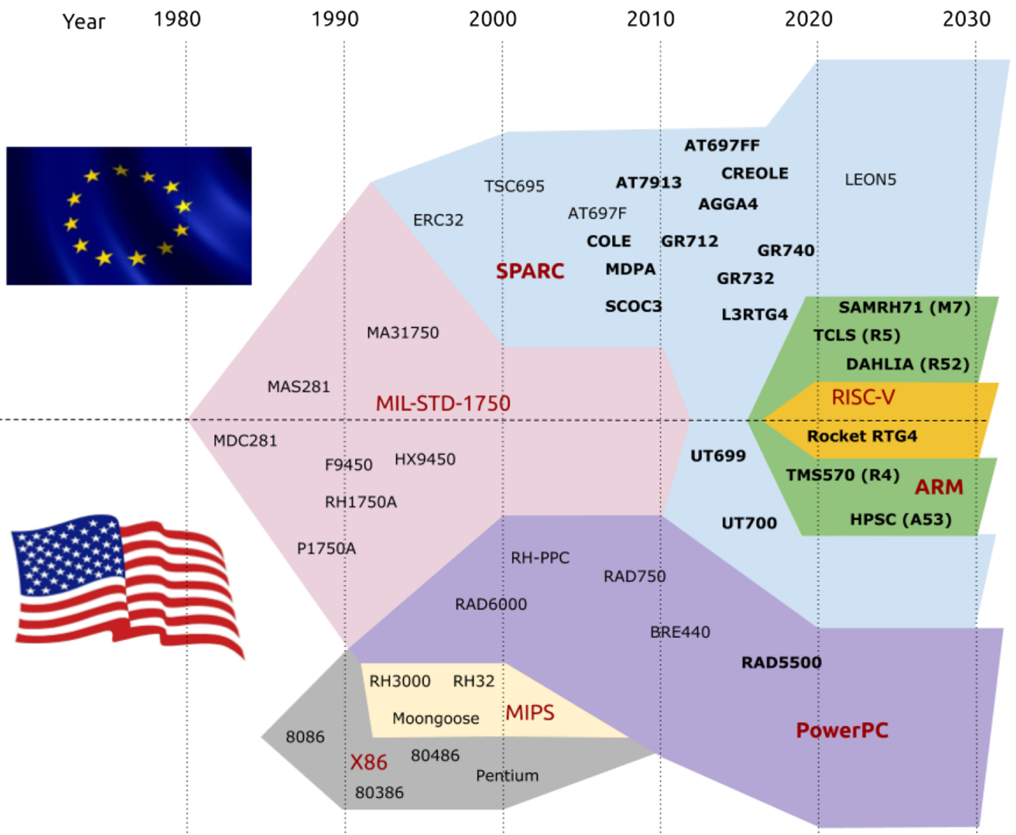

Fig. 1 Main CPU ISAs and their market share evolution in Europe and the United States (in bold SoCs). The figure is an updated version of the one contained in [7].

The work in [13] is an example of how the openness of RISC-V enables academia to work on systematic approaches to eliminating attack surfaces instead of fixing specific security holes and provide strong security guarantees against cache timing and memory access pattern attacks. Another example is [14], which proposes an extension of RISC-V against temporal and spatial memory attacks.

2. *Fault Tolerance*

Much of the considerations on open-source IP cores and security can be applied by the space industry to enhancements in fault tolerance of existing open-source COTS IP cores. Fault tolerance is the ability of a system to prevent an error in a unit to become a failure to provide a certain service at system level. This is achieved using redundancy in different ways (hardware, information, time, etc.). Redundancy typically causes penalties in terms of area/power/frequency to the design. Despite this, it should be noted that LEON processors do not have penalties in terms of microarchitectural efficiency compared with the respective "non-FT" version when errors are not present (e.g., same CoreMark/MHz is given for both versions). More details on the penalties of FT features in LEON processors are given in Sec. III.B.

Assessing how much redundancy is required to achieve a certain level of fault tolerance of the system during operation is crucial to avoid designs with severely area/power/frequency penalties and little or no enhancements in dependability compared with COTS processors with similar features, as not all faults will cause wrong outputs. Fault tolerance can be assessed by fault injection (FI) [15] and observation of the effects on the behavior of the system. FI can then be employed to evaluate the vulnerability to faults in each unit of an implementation using the architectural vulnerability factor (AVF), the probability that a fault in particular unit will cause an error on the outputs of the processor [16]. The overall error rate of a subsystem is then given by the raw soft error rate (SER), due to radiation in a certain environment (orbit, shielding, solar activity, etc.) on a certain technology (e.g., cross sections) and the sensitive area (e.g., the number of bits), multiplied by the AVF for that particular subsystem.

The openness of RISC-V comes in handy also to speed up the evaluation of fault tolerance of a given system, as the unique combined openness and wide adoption from academia enabled unprecedented research possibilities on dependability of processors. In [17], the outcomes of fault injection in cores based on RISC-V (Rocket, BOOM, PULPino), SPARCV8 (LEON3), and ALPHA (IVM) are compared to find the correlation due to the same ISA and by exclusion quantifying the fraction of errors due to differences in the microarchitectures. The possible failures of the fault injection are classified as "Silent Data Corruption," "Unexpected Termination," and "Hang." The results show a strong correlation between the rates of occurrence of each of these outcome for several software programs among the RISC-V implementations (the minimum Pearson coefficient reported is 0.936, and it has been obtained between Rocket and BOOM for the "Hang" outcome in the extreme case of two bit-flip injected simultaneously in random positions) and a weak or even negative correlation between RISC-V implementations and implementations based on other ISAs. This finding is then employed to predict the rates for a particular processor knowing the rates of another for a particular program. The heuristic model proposed in [17] predicts with an error smaller than 7%, requiring only 3 software programs in the training group.

B. Modularity

The main feature of the RISC-V ISA specifications is modularity, which allows to cover a large spectrum of applications and to increase software reuse (adding new instructions as optional extensions instead of releasing new versions of the whole ISA). The RISC-V manual is structured in two volumes, one for the user-level ISA and the other describing the privileged architecture.

RISC-V allows both standard (see Table 1) and nonstandard extensions (defined outside the specifications). The RISC-V ISA is defined as a base integer (I) ISA, which must be present in any implementation, plus optional extensions to the base ISA. A subset of the integer base (E) can optionally be implemented when an implementation targets small 32-bit microcontrollers, with 16 general-purpose registers instead of 32. The standard defines a "general" subset (G) (comprising the IMAFD subsets) as the set of extension required for general-purpose computing systems.

The privileged architecture contains three privilege levels: user (U), supervisor (S), and machine (M) mode. The hypervisor (H) mode (designed to support type 1 hypervisors) has been removed and the encoding space reserved, as the RISC-V community is focusing on hypervisor support via an extended S mode suitable for both type 1 and type 2 hypervisors [18]. An implementation can employ just the user mode, the user and machine modes (when security is a concern), or all of the three modes for implementations targeting Linux-like operating systems (OSs).

C. Purpose, Goal, and Outline

An overview of the state-of-the-art of RISC-V was already given in [7]. Since then the state-of-the-art is evolving so quickly that a 1-year-old report on the state of the RISC-V toolset and on the IP cores available is already outdated (for instance, a big player like Western Digital recently announced that the code of their 32-bit 2-way superscalar RISC-V implementation will be soon available to the public [19] and many others could follow). For this reason, a long-term roadmap for RISC-V processors in space should focus on the evaluation and integration of "profiles" of RISC-V processors (compatible at software level) for specific space applications and not on a specific RISC-V implementation. The modularity of RISC-V, combined with its open and free nature, allows designers to work on a certain profile of processor, and then evaluate which

Table 1 User-level standard extensions [11]

Subset	Name
Integer	I
Integer multiplication and division	M
Atomics	A
Single-precision floating-point	F
Double-precision floating-point	D
Quad-precision floating-point	Q
Decimal floating-point	L
16-bit compressed instructions	C
Bit manipulation	B
Dynamic languages	J
Transactional memory	T
Packed-SIMD extensions	P
Vector extensions	V
User-level interrupts	N

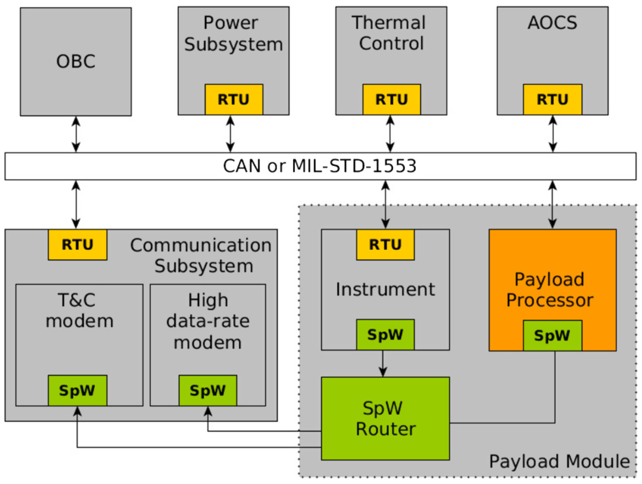

Fig. 2 Distributed satellite data system architecture based on processors with different target applications.

implementation is more fit for purpose (in terms of fault tolerance, performance, security, power, area, etc.).[††] This approach mitigates the risk of tying the space community to a specific implementation that is considered the best available today, but could be outperformed by a newer one or could be discontinued by the original designers/developers in the future. The results reported in [17] confirm this approach also from a point of view of fault tolerance, as the behavior related to soft error can be predicted with an accurate estimation using results from a different implementation with the same ISA.

Also, while the focus of [7] was on why using RISC-V in space applications, the goal of this work is to show how to employ RISC-V in space to enable new applications and improve satellite data system architectures. To do this, in Sec. II we introduce a reference satellite data system architecture, identifying the different processors needed and how RISC-V processors described in literature compare with state-of-the-art solutions (our focus is mainly on processors employed in European missions, although this is a quite sizeable share of the world's space avionic market). To choose the best ISA subset and microarchitecture for each profile, a critical discussion of the benchmark employed to evaluate performance is carried out. Such a critical discussion is crucial, as measuring and comparing performances of processors with different ISA, technologies, frequency, and memory size is a process with large uncertainties. Knowing the limitations of the benchmarks and of the metric employed for comparison, together with an understanding of the microarchitectural differences behind large part of those differences, is key for a correct evaluation of different solutions. For instance, historically, performance of embedded processors was measured with Dhrystone. Dhrystone is a single program and it is so short that can fit in the level 1 (L1) caches of many state-of-the-art processors, thus exercising only a small portion of the core (i.e., integer unit, instruction cache, and data cache). It should be noted that, as microcontrollers are typically simple (no cache, single core), their performance is mainly due to the pipeline and its interaction with the main memory. For this reason, Dhrystone can still be employed as a benchmark for simple implementations like the ones described in Sec. II.A, but it is less effective for the ones described in Sec. II.C due to the presence of large caches, memory management unit (MMU), and multiple cores. However, in the timed part of Dhrystone calls to libraries are executed and thus performance changes depending on the libraries used by the compiler. For this reason, CoreMark is increasingly popular also to benchmark simple implementations.

From the results of the discussion in Sec. II, in Sec. III we propose several generic "profiles" of RISC-V processors for space, with expected performance in technology- and frequency-independent metrics. We then introduce the models typically employed in the development of space processors and discuss what is still missing for RISC-V IP cores compared with the process of adoption of the LEON2FT in space systems, proposing a roadmap that can be applied for each of the defined RISC-V profile. Finally Sec. IV draws conclusions.

II. Processors for Satellite Data System Architectures

Throughout the 1980s and early 1990s space computer systems were bottlenecked by the very slow CPU performance (the first European single-chip processor was the ERC32, providing 14 MIPS @ 20 MHz [10]), thereby limiting what applications could do. Driven by Moore's law, transistor counts increased significantly over the years, improving system performance and enabling new computing possibilities thanks to SoC integration.

The period between 1990 and 2010 was characterized by centralized computing that revolved around an on-board computer (OBC) and nonintelligent (i.e., without software) remote terminal units (RTUs), as described in [22]. Improved connectivity and process improvements in the period from 2010 enabled a shift toward decentralized computing, with a proliferation of connected intelligent devices and sensors, along with a shift to localized computing. The latter moves processing closer to the data, effectively improving latency, bandwidth, and energy use.

The distribution of intelligence throughout the whole satellite, as shown in Fig. 2, requires processors tailored to several power, processing power, size, and real-time requirements. The figure does not include any redundancy schemes and only one instrument connected to the SpaceWire router is represented (when more instruments are present they can be connected in parallel to the one represented). Time-deterministic, low-performance, low-power, low-area "microcontroller" processors (Sec. II.A) can be employed in RTU blocks. The payload processor can either be a "general-purpose" processor (Sec. II.C) or an enhanced version capable of enabling on-board decision making (Sec. II.D). A payload module with an enhanced payload processor capable of making decisions requires less data bandwidth or, in the best case, only the low-speed telemetry and command (T&C) link to send to Earth only selected and meaningful data and/or the results of on-board processing.

The following subsections show how RISC-V solutions compare with state-of-the-art space-grade and COTS processors that fit best for each application in the architecture described in Fig. 2.

[††]For instance, Shakti processors have been classified from their designers in several "classes" [20]. The main difference with this work is that we define generic profiles from the user perspective and that we explicitly include considerations on fault tolerance for space and specific problems of space applications. In the Shakti project "Fault-Tolerant" is just one of the many classes defined ("Servers," "Mobile," etc.), even if the authors of [21] also plan to add fault-tolerant features to the superscalar implementation.

A. Microcontrollers

RTUs are usually present on medium- to large-size spacecraft. RTUs can offload the OBC from many real-time tasks involving data acquisition and actuators. In the past, the RTU was typically a unit without software [22]. However, there is a trend to have at least a low-performance microcontroller in the RTU in order to implement locally autonomous acquisition and control sequences; to gather the analog and digital telemetry from sensors and units (temperature and pressure); to calibrate raw measurements; to provide the conditioning for analog sensors; to control the reaction wheels, gyros, star trackers, sun sensors, propulsion, and other units in the attitude and orbital control system (AOCS); to control solar array drive mechanisms; to distribute power to heaters; to distribute (in some cases also downconverting) power to active loads; and to control and monitor payload units. For instance, Ref. [23] describes a satellite architecture employing RTUs containing an RTAX-based microcontroller. This trend led to development of the DPC from Thales Alenia Space [24] and of the GR716 from Cobham Gaisler [25], both funded by ESA and based on 180 nm RHBD ASIC technology, as well as several other commercial radiation-tolerant microcontrollers.

The use of RTUs also increases modularity, as the RTU can be replicated and the pin-out and software modules of the OBC are less dependent on the specific satellite. Furthermore, discrete observation signals with direct connections to the OBC imply penalties in mass. The use of RTUs connected to the OBC with a bus (e.g., CAN or MIL-STD-1553) reduces cable mass of the satellite and the pin count of the OBC. Reference [26] proposes a plug-n-play platform for small satellites development based on several sizes of the RTU, for example:

1) nanoRTU ($32 \times 32 \times 6.2\ \text{mm}^3$, 0.2 W) for temperature sensors, electrical power switches, magnetic torque control, active heat control, solar arrays, and batteries.

2) microRTU ($70 \times 30 \times 10\ \text{mm}^3$, 1.5 W) for propulsion control, mass memories, and T&C communication.

For this type of applications, where limited performances are requested to the processor, "general-purpose" processors (described later in Sec. II.C) are usually considered not suitable due to high power consumption, large area, presence of caches, and speculation. The ARM Cortex-M0+, a 32-bit 2-stage single-issue processor, is an example of state-of-the-art proprietary implementation employed for terrestrial applications with similar constraints. In the past ESA has evaluated the ARM Cortex M0+ for a ProASIC3-based microcontroller both without modifications and applying flip-flop level triple modular redundancy (TMR) together with safe finite state machine encoding (and reset for unexpected states), employing automated tools [27].

In [28], several RISC-V processors from the PULP platform are described and compared with state-of-the-art ARM cores, Cortex-M0+ included. Figure 3 shows comparisons in terms of performance (CoreMark/MHz), area (kGE), and area efficiency (CoreMark/MHz/kGE) of the Zero-riscy (2 stages, based on the RV32EC subset), Micro-riscy (2 stages, based on the RV32IMC subset), RI5CY (4 stages, based on the RVC32IM subset plus nonstandard extensions), ARM Cortex-M0+, ARM Cortex-M0 (3 stages, an older implementation targeting the same applications of the Cortex-M0+), ARM Cortex-M3 (3 stages), and ARM Cortex-M4 (3 stages). Each parameter is normalized to the maximum value found from the comparison. Comparison between ARM cores and cores from [28] in terms of power is not reported here, as they require measurements with the same technology, voltage, and frequency and running the same software. Instead, area can be easily compared using gate equivalents (GE), which allow a technology-independent measure of the area. Also, a larger area in terms of GE typically implies a larger probability of soft errors.

The implementations can be binned in three groups according to performance: the low-end implementations (Micro-riscy), the midend implementations (Zero-riscy, ARM Cortex-M0, ARM Cortex-M0+), and the high-end implementations (RI5CY, ARM Cortex-M3, ARM Cortex-M4).

The midend generally achieves high area efficiency in terms of CoreMark/MHz. While the ARM Cortex M0+ has the highest area efficiency, it is remarkable that new developments can be based on an open and free implementation (like Zero-riscy) developed by academia with similar performance as state-of-the-art proprietary processors.

The Micro-riscy is optimized for less intensive calculations, defined in [28] as "pure control code," as opposed to the CoreMark based on arithmetic calculations. As a matter of fact, the SoC described in [30] uses the Micro-riscy for power management. In [11], the use of RV32E instead of RV32I is expected to save 25% of area, whereas Micro-riscy saves more (38.62%) compared with the Zero-riscy mainly because it removes also the M extension (no hardware multiplier and divider) [28]. It should be noted that, although this leads to a large improvement in terms of area, it causes large penalties in terms of performance in terms of CoreMark/MHz (−62.7%) with smaller improvements in power consumption (−10.9%). The main reason is that power consumption is dominated by the prefetch buffer accessing the memory that remains the same in both designs [28]. For this reason, the power efficiency of the Micro-riscy is significantly lower than the one of the Zero-riscy in terms of CoreMark/W (−57.49% at 0.8 V and 55 MHz). It should be noted that RISC-V specifications allow the M extension to coexist with E [11] (this would increase substantially area efficiency and performance or Micro-riscy), whereas they do not allow F extension to coexist with E because the area of an floating point unit (FPU) is typically so large that the saving in area due to 16 general-purpose registers instead of 32 would be marginal.

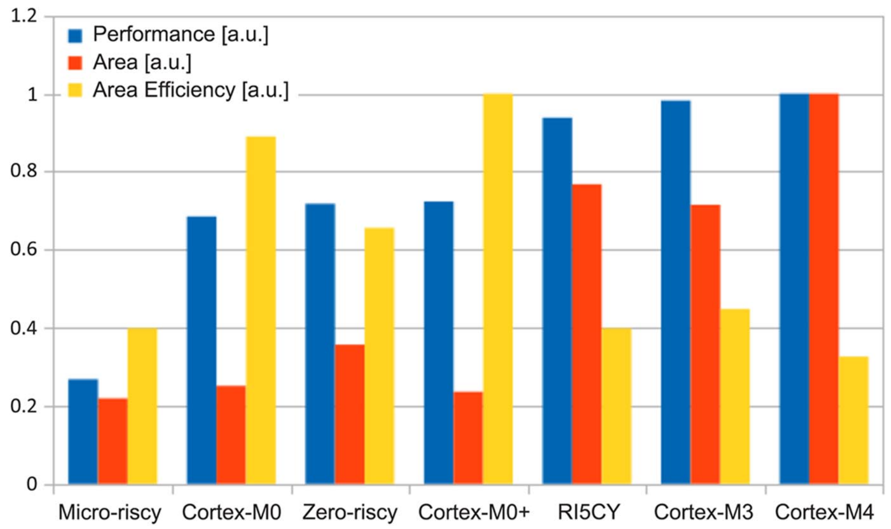

Fig. 3 Performance, area, and area efficiency for several implementations targeting microcontrollers. The data are derived from [28], except for the performance of the Cortex-M3 and Cortex-M4 (from the vendor and [29]).

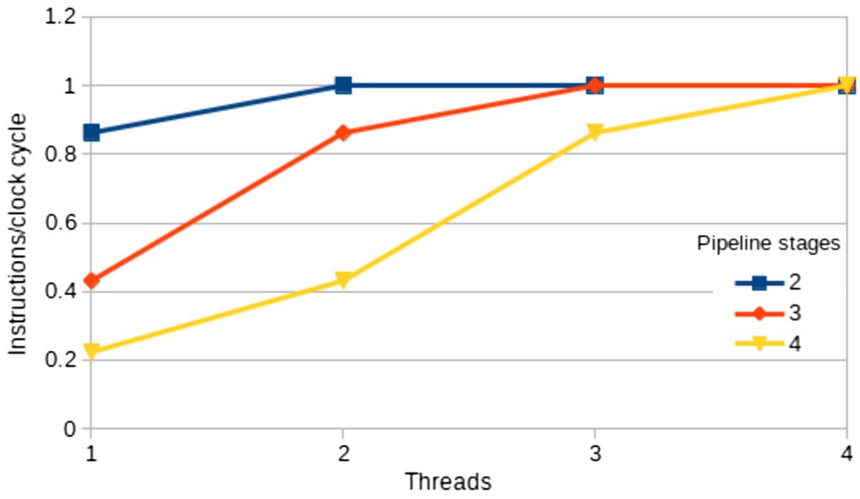

Fig. 4 IPC for Klessydra when running basic integer kernels. The IPC has been calculated multiplying the MIPS (from [34]) by the clock period, for several pipeline lengths and number of threads.

Both Micro-riscy and Zero-riscy employ the C extension to reduce code size,[‡‡] and a RISC-V extension for bit manipulation (B) is planned in the current specifications [11]. The latter would reduce considerably code size and increase performance of typical operations on single bits in embedded software, by providing, for instance, instructions to read, write, clear, or set a certain subset of bits from a register.

The RI5CY processor provides instead nonstandard instructions for bit manipulation [33]. The RI5CY core includes also nonstandard instructions for auto-incremental load and store, multiply-accumulate (MAC), hardware looping, and single instruction multiple data (SIMD) operations. It is an implementation targeting control applications requiring digital signal processing (DSP) capabilities, similarly to the ARM Cortex M4 (which also add DSP extensions to the ARMv7 ISA). RY5CY is 6.1 times faster than the Micro-riscy for the 2D convolution integer kernel employed as a benchmark in [28]. Although the area efficiency for the RY5CY is lower than the one of the Micro-riscy when running CoreMark, RY5CY is 2.83 more area-efficient when running the integer kernel. DSP kernels sometimes require floating-point operations. For this reason, the ARM Cortex M4 can optionally include an FPU (Cortex-M4F), and in [30] the RI5CY has been extended with the *F* extension.

Defining the optimal number of pipeline stages from the number of instructions per cycle (IPC) given in [28] is not possible, as the three cores are based on different ISA subsets. A more homogeneous set of cores is Klessydra [34], a VHDL (VHSIC hardware description language) model of a RISC-V processor (RV32IM) designed for the PULPino SoC with a configurable pipeline of 2, 3, and 4 stages and supporting up to 4 hardware threads with a simple fine-grained implementation that changes thread every clock cycle (independently from whether the current one is stalled or not) [35].

Fine-grained multithreading is interesting for two reasons: it allows an efficient handling of software (SW) techniques to increase fault tolerance employing redundant threads [36], and it increases the average number of IPC without the penalties on time determinism due to speculation, as can be seen in Fig. 4. As a matter of fact, some processors provide branch prediction to increase the number of IPC, which must be disabled if time determinism is required. The work in [37] shows instead how to deal with real-time applications and fine-grained multithread on microcontrollers.

IPC is a figure of merit of how much a certain pipeline is used over time and should not be confused with absolute performances: pipelines with more stages may have worse IPC but still shorter executions times because they can achieve higher frequencies and employ more instruction-level parallelism (even if a larger part of it is wasted compared with shorter pipelines). For instance, the 2-stage version achieves an absolute performance of 63 MIPS with 2 threads (being fully used), whereas the 3-stage version with 2 threads achieves 92 MIPS (being used at the 86.2%).

However, as can be seen in Fig. 4, the longer the pipeline, the worst is the penalty in IPC if no branch prediction or multithreading is present. The absolute performance of the processors shown in Fig. 4 for a single thread is 54 MIPS for the 2-stage version, 46 MIPS for the 3-stage version, and 30 for the 4-stage version [34]. This shows that, for time-deterministic implementations without speculation, a short pipeline (2–3 stages) is the optimal choice also in terms of absolute performance.

B. On-Board Computers

Table 2 shows that OBCs have dramatically improved in terms of power (more than 6× reduction), mass (4× reduction), dimensions (5× reduction)[§§], and computational speed[¶¶] (4× improvement) since the introduction of SoCs in space. However, the improvement in computational speed was mainly due to the increase of the pipeline stages and frequency and not to the microarchitectural efficiency of the processors, as the average number of IPC has increased only by 1.2×.

The main reason is that OBCs mainly deal with real-time tasks leveraging real-time operating systems (RTOS) like RTEMS. Even the newest ones are based on single-core single-issue in-order processors like the LEON2FT and the LEON3FT, because higher performance processors employ microarchitectural features (e.g., superscalar execution, out-of-order execution, speculation, multicore, several level of caches), which poses challenges to the development of real-time software. For instance, Ref. [42] focuses on the evaluation of the effects of multicore and caches on the worst-case execution time (WCET) analysis and [43] on out-of-order (OoO) superscalar processors.

Even if there is a tendency to centralize software on the OBC to simplify software development and qualification, the performance of the whole data system can be increased distributing software for data acquisition/processing and simple control applications in other spacecraft subsystems

[‡‡]The RISC-V specifications [11] claim that 50–60% of the RISC-V instructions in a program can be replaced with C instructions, resulting in a 25–30% code-size reduction. Reference [31] reports similar code size for RV32EC and ARM Thumb-2 when considering the SPEC CPU2006 benchmark suite. As the frequency of instructions and the number of registers employed vary with the application, Ref. [32] defines a nonstandard extension to reduce code size for a specific application.

[§§]The data provided in Table 2 for power, size, and weight refer to a single board with a single computer (processor, memories, etc.). Typically two of them are employed (to overcome common-mode failures on a single board), using an additional board to handle the redundancy at board level.

[¶¶]Performances in Table 2 are given in MIPS, which is a metric that allows fair comparison of different implementations of the same ISA, as it counts only the number of instructions per second without evaluating their impact on the execution times. As SPARCV8 added multiply and divide instructions to SPARCV7, the improvements of SPARCV8-based OBCs over the ones based on SPARCV7 are actually underestimated.

Table 2 Improvements of OBCs achieved with the advance of the state-of-the-art of SoCs [38–41]

OBC	GOCE CDMU	GAIA CDMU	OSCAR (SCOC3)
Year	2009	2013	2014
Manufacturer	TAS-I	RUAG-S	AIRBUS
CPU (ISA)	ERC32 (SPARCV7)	LEON2FT (SPARCV8)	LEON3FT (SPARCV8)
Frequency [MHz]	24	80	80
Computational speed [MIPS]	17	65	68
IPC	0.71	0.81	0.85
Pipeline stages	4	5	7
Power [W]	$\leq$ 90 (average)	$\leq$ 40 (average)	15 (peak)
Mass [kg]	21	16	5.2
Dimensions [mm^3]	$470 \times 272 \times 332$	$420 \times 270 \times 276$	$250 \times 150 \times 216$

rather than using higher-end processors for OBCs. This approach is also suggested by the fact that sensors, actuators, and payloads are inevitably distributed throughout the whole satellite.

Furthermore, “command and control” algorithms run by OBCs have a very low number (around 10 or less) of operations per byte read from the memory (operational intensity), whereas higher-end applications like image processing have a much larger operational intensity (e.g., 10,000). For this reason, while the latter are effectively sped up by increasing the computing capabilities of the processors (e.g., increasing parallelism), the former are essentially limited by the memory bandwidth. Section II.D introduces the “roofline” model as a tool to distinguish between these two kinds of algorithms and to evaluate when an increase in computing capabilities is actually needed for a specific algorithm on a specific platform.

For all the aforementioned reasons, a RISC-V substitute of the LEON3FT (when used as main OBC processor) would require a similar microarchitecture, achieving similar performance with the drawback of giving up long fight heritage and large amount of OBC software legacy for the LEON processors. On the other hand, a RISC-V substitute could leverage a larger software ecosystem in the future for new developments, building on a much larger user base for (for example) the maintenance of SW toolchain.

The Rocket is the closest RISC-V implementation available to the LEON3, and a more detailed comparison is given in Sec. II.C, where multicore versions of these processors are evaluated as low-end general-purpose processors for payloads.

C. General-Purpose Processors

Figure 2 shows a payload processor taking care of preprocessing and sending data through the high-data rate modem. For a payload processor, sometimes the constraint of hard real-time can be loosened and general-purpose computing embracing a soft real-time approach can be adopted instead, expecting a certain quality of service (QoS). In this case it is possible to use largely supported OSs like Linux for general-purpose computing to enable a shift in the paradigm of software development, greatly increasing reuse and enabling complex applications. An example of general-purpose processor for space is the GR740 [25], a quad-core LEON4FT processor originating from ESA’s European Next Generation Microprocessor (NGMP) initiative [42]. In [44] the GR740 is compared with other rad-hard platforms (e.g., RAD5545) and it is found to be the most energy efficient in terms of GOP/J (giga operations per joule).

General-purpose processors employ an MMU to translate page-based virtual addresses to physical addresses. The address table necessary for such translation is contained in main memory and accessing it for each address is not an optimal solution. For this reason, translation look-aside buffers (TLBs) are typically employed to reduce the average time per address translation in a similar manner as caches are employed to decrease the average access time to data and instructions in main memory. Caches store recently accessed data (temporal locality) and other contiguous memory locations (spatial locality) to reduce the time spent accessing the main memory. To increase the average performance, memories are organized in hierarchies, as shown in Fig. 5. In the core there are registers, which are the fastest and smallest memories. At the interface between the core and the interconnect there are L1 caches for instructions (i-cache) and data (d-cache), which are fast but small, whereas between the cores and the main memory other larger but slower levels of cache (typically shared between the cores and other masters) are placed for an optimal access time/missing penalty trade-off. It should be noted that caches and layered memory architectures increase performance for the average case, whereas access times for the worst case get longer and time determinism is penalized.

Figure 6 shows how different (single-core) general-purpose implementations of RISC-V compare to state-of-the-art solutions (year of release included within brackets) in terms of performance (CoreMark/MHz). The collected data are not intended to show the superiority of one processor

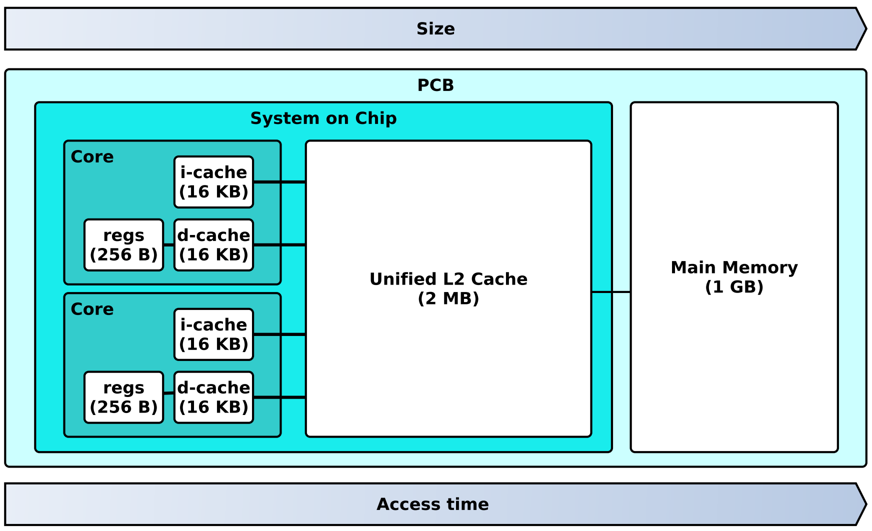

Fig. 5 Typical memory hierarchy with typical memory sizes for a dual-core general-purpose processor.

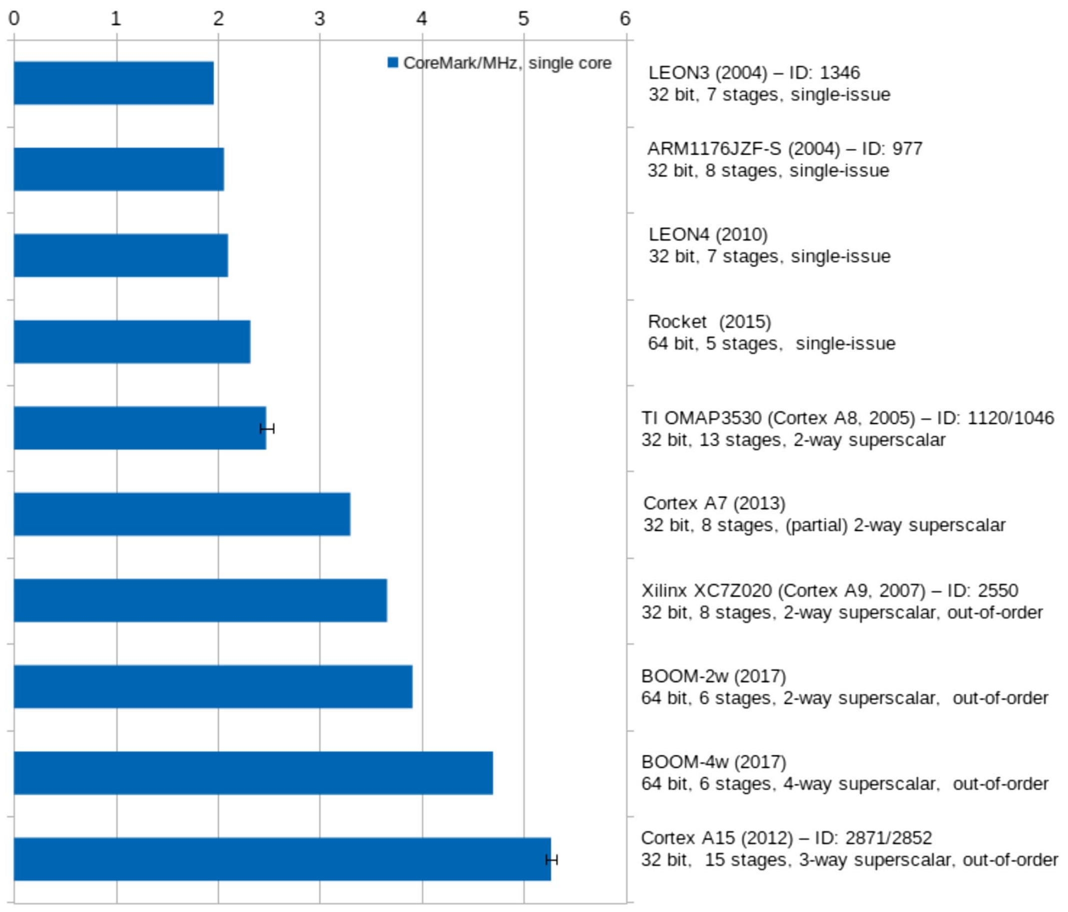

Fig. 6 Performance of single-core processors and their main features. ID of the benchmarks is provided for data taken from [29]. Data for LEON4 are from the vendor; data for Rocket and BOOM are from [45]. Data for Cortex A7 are from [46]. Error bars are given when more results are available.

over another, but to show in some ways the opposite: the final performance of a single core for this kind of applications is mainly determined by the level of ILP employed, especially considering the large uncertainties involved in the comparison of performance for processors of this complexity (e.g., compilers, different size of caches, different technology). For instance, different compilers and compiler flags can cause large differences in the final result. In [29], four different measurements (IDs: 1119/1120/1046/1050) on the Texas Instrument OMAP3530 are given, involving two different compilers (Sourcery G++ 4.4-179 and GCC 4.3.3) and two different optimization flags (−O2 and −O3). The minimum score is 12.75% less than the maximum measured score, whereas using different compilers with the −O3 flag leads to a reduction of the 5.4% of the score. The −O3 flag is able to activate only a subset of the optimization flags, and other optimization flags like "-funroll-all-loops" can affect the performance. However, all data from [29] employed in Fig. 6 have the −O3 flag.

Figure 7 bins the general-purpose processors from Fig. 6 only according to their ILP and shows the strong correlation between ILP and performance. The values found can be used to provide an estimation of target performance for an implementation with a certain ILP. However, this approach has some limitations.

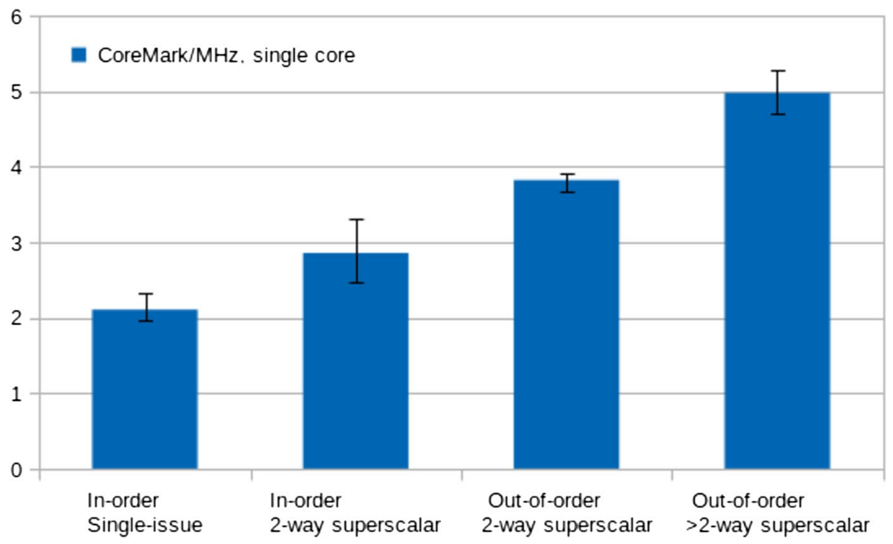

Fig. 7 Expected performance of implementations employing different levels of ILP.

For instance, specific instructions can improve performances, for example, single-cycle MAC instructions can effectively speed up the execution of CoreMark [28]. Also microarchitectural features can increase performance, like dual-arithmetical logical units (ALUs), as [47] shows that substantial improvements can be obtained and [46] describes a 10-stage 2-way superscalar pipeline achieving 5 CoreMark/MHz with dual-ALUs. Furthermore, other kinds of parallelism can increase performance. For instance, the thread-level parallelism (TLP) employed in the MIPS i6500, a 9-stage processor with simultaneous multithreading (SMT), reaches 5.6 CoreMark/MHz for 2 threads [46]. In addition, Fig. 6 does not include description of the degree of speculation employed in the implementations (e.g., branch prediction), but, as a general rule, the wider and deeper the pipeline, the more speculation is required to not waste a large amount operation cycles (e.g., increasing the pipeline depth reduces IPC due to functional latency [48]). For instance, LEON4 employs a static branch prediction scheme (always taken) that exploits the fact that loops, if taken, are typically iterated more than once. To increase IPC further, the Rocket processor [49] employs a branch history table for dynamic branch prediction (prediction learning from past outcomes of the branch), a branch table buffer (BTB) for dynamic branch target prediction, and a return stack to compensate the bad performance of a BTB for returns from functions (the same function can be called from several positions within a program).

Some considerations about the metric employed in the comparison are also required. CoreMark/MHz is a convenient metric to compare fairly implementations in different technologies, but neglects the increase in the maximum allowed frequency for correct operation due to shorter data paths in deeper pipelines and measuring only the improvement due to simultaneous execution of more suboperations from more instructions. Finally, comparing processors only using a metric evaluating performance without considering area efficiency of ILP solutions may lead to implementations too big for RHBD technologies and space-grade FPGAs, especially because area efficiency typically decreases when ILP is increased. For instance, the data from [45] shows that the single-issue Rocket (core+L1 cache) has an area efficiency of $4.64\ \mathrm{CM/MHz/mm^2}$, the 2-way superscalar OoO BOOM has $3.55\ \mathrm{CM/MHz/mm^2}$, and the 4-way version $3.36\ \mathrm{CM/MHz/mm^2}$ (all of them in 45 nm technology).

Keeping in mind all these limitations, we suggest in Fig. 7 that an in-order single-issue processor is not enough to reach 3 CoreMarks/MHz and that more than 2-way superscalar and out-of-order (OoO) execution are needed to achieve 5 CoreMark/MHz. When we consider other kinds of parallelism and microarchitectural solutions mentioned, either simultaneous multithreading or a deep pipeline ($\geq$ 10) plus dual-ALUs and MACs can help reaching 5 CoreMark/MHz.

It should be noted also that Figs. 6 and 7 consider only performances of single cores, whereas many general-purpose processors provide more than one processing core. Processor-level parallelism (PLP) is typically exploited employing a symmetric multiprocessing (SMP) approach, based on an OS scheduler assigning a thread for each core. In this approach, all the cores are connected to the bus and have access to a shared memory, each of them having their private L1 caches. The problem of memory coherence in such architectures becomes crucial, and the choice of the coherence protocols greatly affects performance and fault tolerance. For instance, as noticed in [50], write-through protocols are inherently more robust than write-back protocols: in case of single error the L1 caches can correct the error employing just the parity and reading again the corrupted data from lower levels (e.g., L2). In this case, only the main memory is strictly required to have correcting codes, as in write-through protocols data are available in more copies. For this reason, LEON processors use a simple invalidation write-through policy. However, write-through protocols generate large amount of traffic and this is especially a problem in single-layer buses like AHB, where only a transaction is allowed per time. Use of switch fabric (which reduces contention allowing more than one transaction per time) allows in general a more efficient scaling with PLP, with a penalty in terms of area occupied by the interconnection. Also, write-through protocols make intertask interference worse compared with write-back protocols like MESI [42]. It is clear that SMP processors with L2 caches heavily rely on increase of memory area (as shown in Fig. 8), making memories the biggest contributors to the SER of the processor and requiring error detection and correction (EDAC) codes and cell interleaving.

Figure 9 shows the performance enhancement due to the increase of the number of cores for several SMP implementations of the Cortex A9. The figure shows that a state-of-the-art SMP implementation can provide a 3× improvement in terms of CoreMark/MHz. SMP processors typically provide up to 4–8 cores, as they are limited by the sequential model of the software and the use of shared resources (memories and interconnects). The impact of sequential tasks is described by Amdahl's law [53], which divides a general problem in a parallelizable part (p) and a sequential part (s), and assumes that the size of the problem remains constant with the increase of PLP. From [53], it can be seen how this leads (for an infinite amount of parallel processors) to the theoretical maximum speedup of $1/s$, meaning that (under Amdahl's assumptions) the speedup of a program using multiple cores in parallel is limited by the time needed for the sequential fraction of the program. In an SMP platform this means that the capability of the OS to parallelize the workload is the final bottleneck. Therefore, increasing the number of processors provides only marginal improvements from a certain number of processors onward, independently from how efficiently resource sharing problems (e.g., access to shared memories) are solved.

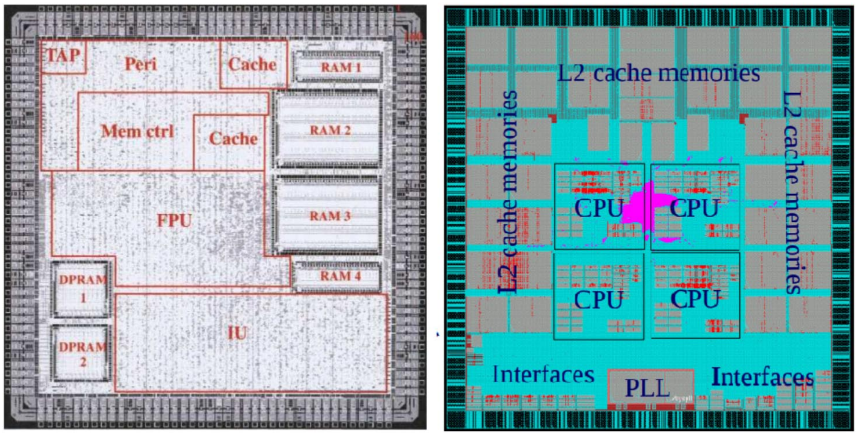

Fig. 8 On the left, the first silicon implementation of LEON codenamed "LEON-Express" (2001) [51]. On the right, the layout of the GR740, a state-of-the-art quad-core SoC based on LEON4FT [52].

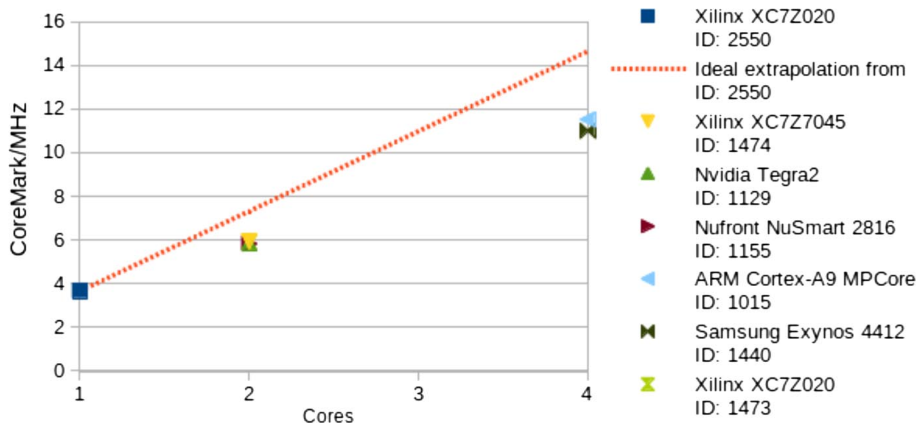

Fig. 9 CoreMark/MHz for different SMP implementations of the Cortex A9. In red is the ideal enhancement obtained by multiplying the value for a single core by the number of cores. All data from [29].

D. Processors to Enable On-Board Decision Making

On-board decision making helps overcome typical bottlenecks of space systems like the downlink bandwidth. Link bandwidth is a problem especially for severely resource-limited spacecraft like smallsats (e.g., it can take hours to days to downlink an image from a CubeSat to the ground) [54]. The capability of filtering out images that do not meet certain criteria (data reduction) can greatly increase the potentiality of CubeSats, as improvements in sensors generate increasingly larger data volumes and storage space on CubeSats is typically very limited [54]. Reference [55] describes the validation on a CubeSat of intelligent algorithms to discard images with no information and other autonomous capabilities (e.g., computer vision). Many proposed future missions are supposed to produce enormous volumes of data. To enable them, Ref. [55] suggests that either significant communication advancements or data reduction techniques are needed. However, advances in both capabilities would provide systems designers with a larger design space where better trade-offs are possible for missions producing large volumes of data.

On-board decision making can enable also swarms [56], active debris removal (ACD) by vision-based navigation (VBN) [2] and in the future even cost-effective asteroid mining [57]. Capabilities of scientific missions can also be improved by on-board decision making. For instance, the introduction of increasing on-board decision-making capabilities to enable autonomous navigation and autonomous science in the Mars rovers decreased the need of sending data to be processed and interpreted on Earth; thus the rovers were able to explore more on the surface of the planet instead of waiting for commands, overcoming also latency of operations commanded from ground [58].

On-board decision making requires some degrees of AI to be present on-board. AI is a broad definition, and the type of AI employed changed over time. Regardless of the level of AI employed, AI typically implies a large amount of matrix calculations on a large set of data. Our focus in this subsection is on how to speedup matrix calculations, while handling large data sets may require different solutions compared with general-purpose processors [59] and different memory consistency models [60].

In a first phase, systems were “smart” because they were provided with fixed rules defined by experts that worked for a specific application within certain expected boundary conditions. This approach is typically referred to as DSP. As an example, Ref. [2] describes several “tailored” algorithms for VBN. Today terrestrial applications are in a new phase where systems employ statistical learning to adapt their behavior to unexpected changes. This approach is typically referred to as machine learning (ML). Some terrestrial applications rely on convolutional neural networks (CNNs). CNNs are composed of discrete time convolutions, which are basically matrix-vector products. CNNs are booming in terrestrial applications, thanks to the availability of large data sets from “big data” and the availability of graphics processing units (GPUs) that enabled elaborations of large data sets in reasonable timescales [61]. This approach is typically referred to as deep learning.

GPUs have paved the way for implementations heavily based on PLP (often the term “manycore” is employed to highlight an heavier use of PLP compared with the “multicore” processors described in Sec. II.C), employing new software models. For instance, NVIDIA released CUDA in 2007 [62], adopting a single instruction multiple thread (SIMT) model. SIMT implementations act on data of different threads, allowing software to scale transparently to hundreds of processing cores [62]. As a matter of fact, the programming models and architectures employed in manycore processors are such that PLP is exploited in a way closer to Gustafson’s law rather than Amdahl’s. Gustafson in [53] objected the assumption of Amdahl that the size of the problem remains constant when PLP is increased. Instead, he assumed that it is the run time that remains constant, thus increasing the amount of calculations included in the problem, assuming that the workload can scale with the number of processors. Under this assumption, the sequential part of the software becomes more and more negligible when the number of processors is increased, giving a linear increase with the number of processors.

In [2], several types of platforms (“general-purpose” processors described in Sec. II.C, manycore DSPs, FPGA-accelerated processors, GPUs) are compared with regard to several benchmark algorithms representative of VBN for space applications. The results are reported in Fig. 10. Error bars have different meanings depending on the platform: for the LEON4FT the uncertainty stems from the fact that the performance is extrapolated from a single core running at lower frequency. For the 66AK2H14 it stems from slightly different results found in literature. For Zynq it originates from the fact that acceleration resulted in various acceleration factors compared with “general-purpose” processors depending on the specific algorithm. Finally, for the desktop GPUs it stems from different chip models [2].

Figure 10 shows that the highest energy efficiency is achieved by processors employing custom FPGA-based on-chip accelerators tailored to the specific application, whereas high-end GPUs provide the highest performance but with the worst energy efficiency of all the 28 nm platforms analyzed. Data from [2] also show that GPUs for mobile devices have performances 13 times lower than the ones for desktops without substantially increasing energy efficiency (the best-case estimation is even lower).

Hardware accelerators and their integration within an SoC are a solution to make AI pervasive by decrease power consumption and increase performance [63]. However, given the niche-sized market for space applications, developing a custom hardware accelerator for each algorithm is too expensive as the cost cannot be spread on large volumes as it is possible for instance for mobiles. The hardware acceleration described in [2] is achieved connecting the accelerator to the bus or crossbar so that no change to the ISA is required (as is typically done on Zynq platforms). The extendability of RISC-V enables ISA application-specific extensions to increase performance in a specific application for tightly coupled accelerators, as described in [64]. Also this approach has a high cost, especially because it makes reuse of the software ecosystem more difficult

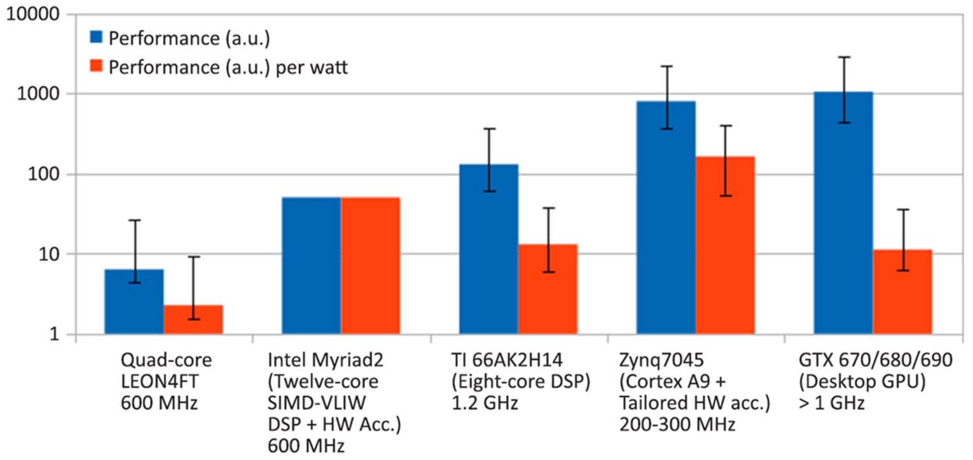

Fig. 10 Comparison of five different platforms in terms of performance and energy efficiency. All the devices are based on 28 nm technologies for fair comparison, except LEON4FT, which is based on a larger technology node. All data are derived from [2] and “a.u.” stands for “arbitrary unit.”

and requires forking from the standard toolchain. An approach to solve this issue is to provide software support for an ISA extension defining a generic interface for accelerators, as the Rocket Custom Co-processor (RoCC) included in the Rocket core [65].

The low performance of a “general-purpose” processor, like the quad-core LEON4FT, is easily explained by the fact that this kind of processors relies mainly on ILP and includes little or no data-level parallelism (DLP). Data-parallel processors execute an instruction on more elements of a vector. For instance, the sum of two vectors is sped up m times when the ALU is replicated m times. Qualitatively, to equal the gain due to parallelism, the nonparallel processor should execute the addition m times faster, which is already difficult for $m = 4$. As a matter of fact, the second best thing found in [2] in terms of energy efficiency is the Myriad 2 [66], a manycore SIMD processor. The 66AK2H14 has been included in the comparison to show that a similar choice can also lead to a different performance/energy efficiency trade-off. An example of manycore SIMD processors based on RISC-V targeting ultra-low power applications is the PULP platform [67], which provides an 8-core array of RI5CY cores with 2 shared FPUs plus a Zero-riscy for control (both of them already described in Sec. II.A). The work in [30] describes an ASIC implementation (Mr. Wolf) of the platform on a low-power 40 nm technology, reporting a peak performance of 1 single precision (SP-)GFLOPS and 7 GOPS, a maximum energy efficiency of 18 SP-GFLOP/J and 30 GOP/J and a maximum frequency of 450 MHz. As noted in [68], even if SIMD solutions have been successful on the consumer market, they are a suboptimal choice for an ISA. SIMD processors are usually implemented replicating the ALU and providing new opcodes to the ISA in order to exploit this parallelism. This makes ISA instructions dependent on the implementation-specific degree of parallelism and new instructions must be added when new specific operations are needed. As a consequence, in order to increase performance and assure backward compatibility, also the number of instructions has to increase. Under this assumption, the maximum performances for processors employing this approach have a theoretical upper limit because at some point no opcodes will be available anymore. Moreover, when the software executes instructions with less data parallelism than provided by the hardware, some of the ALUs will remain idle, and thus the hardware will be underused.

For these reasons, Ref. [68] proposes vector processors (based on the concept of vector lane) as an improved version of data-parallel processors, exploiting both spatial (ALU replication) and temporal reuse. In these platforms, the vector length and the types of elements are configured before the operations, reducing hardware overhead and not requiring a new instruction for each new operation supported by the hardware. RISC-V dropped the proposal of a SIMD standard extension (P) and proposes a “Vector” (V) standard extension (in the process of being standardized), derived by the development of the Hwacha coprocessor [69]. The “V” extension aims to be flexible and reconfigurable for different data types and sizes on the run, with the goal to support both implicit autovectorization in (OpenMP) and explicit SPMD (OpenCL). As the capability of autonomous decision making becomes crucial in many applications, vector lanes in processors could be as common in the future as FPUs are today.

The authors of [68] also compare vector processors to manycore multiple instructions multiple data (MIMD) implementations. An example of MIMD implementations is the Epiphany series. The Epiphany-III has 16 cores and achieves 1 GHz on 16 nm technology, with a declared peak performance of 32 SP-GFLOPS and energy efficiency greater of 16 SP-GFLOP/J [70]. The Epiphany-IV with 64 cores achieves 800 MHz with a declared peak performance of 102 SP-GFLOPS and an energy efficiency greater than 51 SP-GFLOP/J [71]. Finally, the Epiphany-V core with 1024 cores reaches a declared peak performance per clock cycle of 2048 DP-FLOP/cc [72]. Another manycore implementation is the 16 nm Celerity SoC [65], containing 5 Rocket RV64G cores, an array of 496 RV32IM cores, and a binarized neural network (BNN) accelerator coupled using RoCC. The authors of [65] acknowledged the key role of the RISC-V ecosystem in enabling a small team of junior graduate students to design and tape out a complex SoC in just 9 months. However, they report also to have faced some challenges, including limited documentation, lack of reference implementations in an industry-standard hardware description language, and the lack of a stable release schedule across the entire ecosystem [65].

In [68], a manycore MIMD implementation composed of an array of simple RISC-V implementations (not containing any SIMD features) with 4, 8, and 16 cores is compare against a Rocket with a Hwacha coprocessor with 1, 2, and 4 vector lanes. The results show that, for similar area, the vector lane solution generally performs better and with lower power consumptions. The main reason behind this is that the multiple cores in an MIMD array execute copies of the same instructions across multiple data elements. This approach leads to lower area and power efficiency, as the instruction fetch mechanism is one of the most expensive components of a processor (as already noted in Sec. II.A).

A dual-core SMP implementation of Rocket with a single-lane Hwacha (version 3) each is described in [69] (Raven-3), taped out on a 45 nm process and achieving 1.3 GHz. Performances are measured with a double-precision floating-point matrix-multiplication kernel (reported to achieve 78% of peak performance). The values found are 16.7 double precision (DP-)GFLOP/J at 0.65 V and 250 MHz and 1.72 DP-GFLOPS at 0.55 GHz and 0.8 V (3.13 DP-FLOP/cc) and a 4.03 DP-GFLOPS peak performance for the kernel at 1.3 GHz and 1.2 V (3.10 DP-FLOP/cc). While Hwacha is a coprocessor to be placed next to a Rocket core, in [73] a stand-alone data-parallel processor (LACore) based on a nonstandard extension of RISC-V is described. Even if it was not validated with an ASIC implementation, Ref. [73] provides a full benchmarking of the implementation using hand-written optimized versions of the HPC Challenge (HPCC) Benchmark kernels to compare the LACore implementation against a baseline single-issue RISC-V implementation without DLP (both in gem5). The absolute performance of LACore and

Table 3 Absolute GFLOPS and speedup of LACore compared with a scalar RISC-V pipeline [73]

Architecture	DGEMM [DP-GFLOPS]	FFT [DP-GFLOPS]	PTRANS [DP-GFLOPS]	HPL [DP-GFLOPS]	STREAM triad [GB/s]
LACore	~16 (p,w)	1.88 (p)	~0.6 (p)	2.55 (p)	103 (p)
		0.55 (w)	0.15 (w)	1.52 (w)	7.25 (w)
Speedup vs RISC-V	80.4× (p, w)	4.23× (p)	~3.5× (p)	4.67× (p)	13× (p)
		2.1× (w)	1.52× (w)	5.88× (w)	5.2× (w)

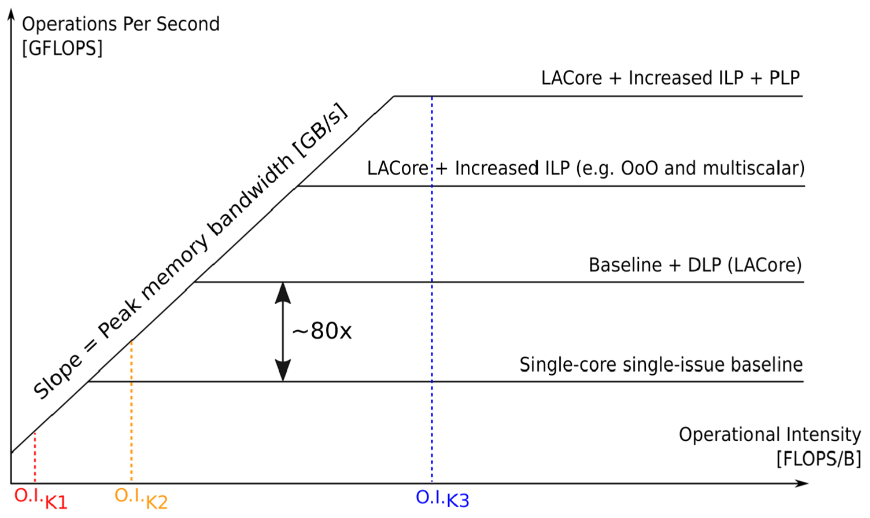

Fig. 11 Example of roofline models for four platforms employing incremental levels of parallelism for three different kernels (K1, K2, and K3) with different OI (incremental improvements not to scale).

the speedups obtained against the scalar RISC-V baseline are reported in Table 3. The frequency is 3 GHz for the scalar part and 1 GHz for the data-parallel part of the pipeline for LACore and 3 GHz for the RISC-V baseline. For each benchmark values for the peak performance (p) and for the maximum tested workload (w) are given, depending on the size of the matrix for DGEMM (p and w:1024) and for PTRANS (p:2^6, w:2^{10}), and the length of the vector for FFT (p:2^{12}, w:2^{20}) and STREAM Triad (p:2^{12}, w:2^{20}). LACore substantially increases performance of all the kernels in the HPCC compared with the scalar RISC-V implementation, except for the RandomAccess (as reported in [74]) because it does not imply vectorial operations but measures the peak capacity of the memory subsystem while performing random updates to the system memory (to evaluate the performance of a processor to deal with large and segmented data sets) [75]. The other kernels in HPCC are HPL (solving a linear system of equations), DGEMM (real matrix-matrix multiplication), STREAM (which measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel like Triad [***]), PTRANS (which exercises the communications where pairs of processors communicate with each other simultaneously), and FFT (a complex one-dimensional discrete Fourier transform). The efficiency of DLP for matrix operations is confirmed from the data in Table 3. As a matter of fact, Ref. [73] provides an estimation of the area penalty of 2.53× for LACore over the RISC-V baseline. This means that LACore improves area efficiency up to 31.77 times for the DGEMM, but only up to 1.67 times for the FFT.

Even if there is no established benchmark suite for this kind of applications, as opposed to what CoreMark is for embedded applications and SPEC for desktops, the performance of these processors are generally benchmarked like done in [73], measuring the GFLOPS for several kernels. Performance of a certain kernel in terms of GFLOPS on a certain platform can be related to its operational intensity (OI) using the “roofline” model [76], with OI being the number of FLOP per byte read from off-chip memory for a certain kernel. In this model, a platform is represented by a slope given by the memory bandwidth and a “ceiling” representing the peak computational power, as shown in Fig. 11. Figure 11 adds advanced ILP (e.g., OoO execution) and multiple cores (PLP) to LACore as incremental enhancements. The performance of a kernel in Fig. 11 is given by the intersection of the vertical colored line given by its OI and either the slope or the flat “ceiling” of a certain platform. However, the measured GFLOPS are typically lower than the ones at the intersections because of nonidealism not taken into account in this simple model. Kernels intersecting the ceiling are limited by computing capabilities, and their performance can be enhanced with additional levels of parallelism (like the kernel K3 is for each of the four platforms presented in Fig. 11 and the kernel K2 is only marginally and only for the addition of DLP). Kernels limited by the slope are constrained by the memory bandwidth and therefore are not improved by higher levels of parallelism (e.g., the incremental use of DLP does not speedup the kernel K_1). When considering LACore and the roofline model in Fig. 11, RandomAcces is an example of algorithm like K1, FFT an example of kernel like K2, and DGEMM of a kernel that can be assumed like K3. DGEMM is typically compute-bound for all sizes and reaches about 95% of the peak performance (flat line) [77].

Another approach to benchmark processors for AI is to measure the time needed to carry out a specific task. For instance, MLPerf is a proposed benchmarking suite for ML for both training and inference on a wide spectrum of applications ranging from mobile devices to cloud services [78]. Another example is DAWNBench, an end-to-end benchmark for deep learning [79] to measure training time and inference latency for image classification and question answering [79].

III. RISC-V Roadmap for Space

A. RISC-V Profiles

In Sec. II different implementations have been analyzed for each type of processor identified in satellite data systems. From this analysis, several “profiles” of RISC-V processors are defined in Table 4, focusing on the features (ISA subsets and nonstandard extensions,[†††] ILP, PLP), suggested

[***]The operation executed is $z \leftarrow \alpha x + y$, where z, x, and y are vectors and α is a scalar.

[†††]The ISA subsets and the nonstandard extensions employed define also if DLP is present.

Table 4 Features and expected performances of RISC-V profiles

Profile	Reference Impl.	ISA Subset (optional)	ILP	PLP	Benchmark	Target performance
uC (LE)	Zero-riscy	RV32E(M)C (B)	2–3 st., SI, IO	1	CoreMark Control code	1 CM/MHz
uC (ME)	Micro-riscy/Cortex-M0/M0+	RV32ICM (B)	2–3 st., SI, IO	1	CoreMark Dhrystone	2.4 CM/MHz
uC (HE)/A (Single-core LE)	RI5CY/Cortex-M3/M4(F)	RV32IMC(F) (DSP NSE or V)	3–5 st., SI, IO	1	CoreMark Int. Kernels	3 CM/MHz 2 OP/cc
OBC	LEON3FT/Rocket	RV32GC	5–7 st., SI, IO	1	CoreMark	2 CM/MHz
GP (LE)	LEON4FT/Rocket	RV64GC (N)	5–7 st., SI, IO	1–4	CoreMark	2–6 CM/MHz
GP (ME)	ARM Cortex-A7/A8	RV64GC (N)	8–13 st., 2-w, IO	1–4	CoreMark	3–9 CM/MHz
GP (HE)	BOOM (2-w)/Cortex-A9	RV64GC (N)	6–8 st., 2-w, OoO	1–4	CoreMark	4–12 CM/MHz
GP (VHE)	BOOM (4-w)/Cortex-A15	RV64GC (VN)	6–8 st., > 2-w, OoO	1–4	CoreMark	5–15 CM/MHz
A (Manycore LE)	Mr. Wolf/Epiphany-III	RV32IMFC (DSP NSE)	3–5 st., SI, IO	8–32	HPCC (DGEMM)	2–16 SP – FLOP/cc
A (Vector Coprocessor)	Raven-3 (single lane)	RV64GCV	5–7 st., SI, IO	1–4	HPCC (DGEMM)	2–6 DP – FLOP/cc
A (Manycore HE)	Epiphany-IV/V	RV64IMFD	3–5 st., SI, IO	64–1024	HPCC (DGEMM)	32–512 DP – FLOP/cc

HE, high end; IO, in order; LE, low end; ME, middle end; NSE, nonstandard extension; *n*-w, *n*-way superscalar; OoO, out of order; SI, single issue; VHE, very high end.

benchmark, expected performance (for each type of processor several levels of performance and/or DLP/PLP approaches are proposed) and reference silicon-proven implementations available.

The three profiles identified for microcontroller applications of RISC-V (uC) are essentially derived from the three IP cores presented in [28] for the Internet of Things (IoT). The uC LE is for "pure control" applications (e.g., power management and latchup protection of COTS components), the uC ME is for control applications requiring more calculations (e.g., AOCS units), and the uC HE for sensors producing larger amounts of data (e.g., cameras). The LE profile is still benchmarked with the CoreMark, but "pure control code" (e.g., a non-preemptive task-scheduler and a driver to interact with peripherals) like the "Runtime" employed in [28] is much more representative of real applications for this profile. As the uC HE can be seen as a step toward Autonomous (A) processors (a single-core version of the A Manycore LE), a pessimistic estimation of 2 OP/CC is reported by dividing the GOPS performance when running integer kernel on the Mr. Wolf [30] by the number of cores.

For uC processors, short pipelines are preferred to achieve high IPC without speculation to avoid penalties on time determinism. These processors also target deterministic interrupt handling for time-critical applications, with fixed and minimized latency for interrupts. The RISC-V architecture model for managing exceptions, interrupts, and registers is simple and suitable for implementations of processors with deterministic timing behaviors (e.g., no windowed overlapping registers, overflow/underflow to be managed).

The expected performances in terms of CoreMark/MHz are similar or greater to the ones expected from the OBCs because shorter pipelines lead to better IPC. However, lower maximum frequencies in general lead to lower absolute performance compared with microarchitectures for OBCs. Additionally, OBCs require support for RTOSs.

The features and the expected performances of the four profiles of general-purpose (GP) processors stem directly from Figs. 7 and 9, with the LE version being a multicore extension of the OBC profile with relaxed real-time constraints. The RV64GC profile is proposed as it is the subset targeted at the moment by the Linux ports for RISC-V,[‡‡‡] and all the three levels of privileges described in Sec. I.B are required. These processors are required to support general-purpose OSs but sometimes also RTOSs for hard-real time processing. Development of real-time software becomes more critical by increasing ILP and speculations; thus the different profiles give several possible time-determinism/performance trade-offs, along with area/performance and power/performance trade-offs. The use of the *N* extension can be evaluated, as delegating some interrupts to user-level processes reduces the overhead both in terms of number of cycles and in terms of cache misses [80].

To enable on-board decision making, several PLP/DLP approaches for the "Autonomous" profiles (A) are proposed. All of them are related to silicon-proven implementations, and for each of the profiles a range of expected FLOPS/cc is given, with some extrapolations. For the A Manycore LE the lowest expected performance is taken from Mr. Wolf, which has 8 cores and 2 shared FPUs. Assuming an ideal improvement of 4× for 32 cores and assuming an ideal 4× increase of performance with an FPU per core, an optimistic estimation of 32 FLOP/cc for a 32 cores version can be done. An expected performance of 16 FLOP/cc is taken for a more conservative estimation. This estimation is half the peak FLOP/cc given for the 16-core Epiphany in [70], from which we can deduce a conservative derating factor of 0.25×. For this reason, the target performance of the Manycore HE is multiplied by 0.25 compared with the peak values given in [72] for a conservative estimation. For A (Vector coprocessor) the maximum value found for Raven-3 is divided per 1.5 to estimate performance for single core and multiplied by 2 to have estimation for four cores, in accordance with Fig. 9. The manycore profiles can be seen as MIMD extension of the microcontroller profiles, whereas the Vector coprocessor profile is an extension of the SMP GP processors with the addition of tightly coupled coprocessors. The HPCC has been selected as reference benchmark suite as it contains popular kernels (e.g., DGEMM).

All the metrics employed (like CoreMark/MHz) to estimate the expected performance in Table 4 have been chosen to be technology independent and frequency independent so that penalties for fault-tolerant features required in space (like the ones described in [51]) will not cause lower expected performances compared with the non-FT reference versions, even if they will cause lower frequency (therefore lowering the absolute performance, e.g., the absolute CoreMark score) and larger area (for instance, reducing the number of cores that can fit in a certain chip, thus limiting PLP).

[‡‡‡]While atomic instructions are generally needed especially to boot Linux on multicore, a smaller subset would be strictly required, as, for instance, the need of the C extensions is a requirement introduced by the Linux ports available at the moment.

B. From the RISC-V ISA to Flight-Proven Processors

Section II analyzed several silicon-proven IP cores based on RISC-V. However, a RISC-V-based IP core validated for the space environment with a long history of flight heritage like LEON2FT is not available yet. Nevertheless, the European space industry has already most of the tools (software support, models, etc.) available to evaluate the potential of future FT implementations. Different models of a processor are typically used depending on the purpose during the development of a system (some of those are more fit to develop software, some others to explore design space of an hardware implementation, some others to estimate performance and some others to synthesize an FPGA or ASIC implementation) and on the TRL of the implementation (for instance, before tapeout the most mature hardware model is the FPGA prototype). Models of processors can be classified in simulation and hardware models and provide several levels of abstraction generating a large spectrum of possible accuracy/speed points. For instance, during software development and hardware prototyping, bugs may be triggered on the order of billions of cycles. Whereas on a hardware model a billion of cycles take a dozen of seconds, on an RTL C++ model generated from Chisel 1 billion cycles take 8 h [45]. On the other hand, RTL models provide a level of observability usually not achievable with hardware prototypes.

1. Simulation Models

Simulation models are reported below with a critical discussion on their accuracy, performance, and space relevance. The results of this discussion are reported in Table 5.

1) Instruction Set Simulator (ISS): Once the required standard extensions and privilege levels are defined, the profile and the software can be evaluated in an ISS like Spike (the "golden reference" of the RISC-V ISA), QEMU, and RV8 [81]. Microarchitecture features and memory behavior are not represented, and so the accuracy in terms of performance and fault tolerance is very limited. From the data provided in [81] a penalty factor of 5 over native x86 code can be taken for these simulators. Assuming an x86 processor running native code at 100 GIPS, a performance of 20 GIPS is then expected.

2) Event-based simulators: An example is gem5, which provides several possible trade-offs of accuracy/speed in between ISA simulators and cycle-accurate simulations [82] by providing some microarchitectural features (a minimal single issue model, an improved version representing timings of memory, a more detailed pipelined in-order model, and a pipelined out-of-order model), memory models, and emulation or execution of system-level services [83]. Several RISC-V implementations from literature are described in gem5 for the good accuracy/speed trade off of this solution during design exploration. For instance, Ref. [73] uses gem5 models to compare different architectures instead of using real hardware to remove variables as process technology, clock speeds, and cache configuration in order to compare the processor architectures only. Reference [84] shows that gem5 is about 40 times faster than Chisel C++ RTL models in terms of IPS. Another example of use of these simulators is that of virtual prototypes, like the one described in [85] for LEON processors. This platform is intended to enable fast full-system simulation, low-level SW development, and design-space exploration for multiprocessor SoCs. It is based on state-of-the-art design specification languages (such as SystemC/TLM2) and with three abstraction levels: loosely timed, approximately timed, and RTL (this level is typically not included in virtual prototypes and we will not consider it in Table 5). Different levels of abstraction can be employed for each part of the system, according to the required accuracy/speed trade-off. In [86] a RISC-V virtual prototype is described, and in [87] a RISC-V virtual prototype is reported to be in the range of 30–220 MIPS (depending on the accuracy).

3) Speed-optimized Cycle-Accurate (CA) models: These include implementation-specific and microarchitectural features to estimate timings correctly. Space industry heavily relies on this tool to develop software in a faster way and to provide WCET analysis. For instance, the use of a cycle-accurate model of LEON (TSIM) is key to provide a lightweight, scalable, and cost-efficient solution for integration and validation of satellite subsystems and payload equipment containing embedded software [88]. The vendor reports TSIM to reach 60 MIPS on a Intel i7-2600K @3.4GHz PC, which is close to the MIPS reported for the OBC in third column of Table 2 (−13%). However, the situation gets worse when the hardware prototype works at higher frequencies (e.g., the GR740 is reported to have a maximum frequency of 250 MHz).

4) Hardware Description Level (HDL) model (also referred to as IP core or RTL model): This is a crucial model for an implementation, as not only is the most accurate simulation model, but it is also employed to synthesize the processor for FPGAs and ASICs. RISC-V is often associated with the Chisel HDL because Rocket, the reference RISC-V implementation, is written in Chisel. Chisel has a remarkable potential as a tool to create parameterized architecture, enabling and facilitating academic research [89]. From the Chisel model both a Verilog RTL and an C++ RTL model can be generated. Reference [89] reports that the C++ models is around eight times faster than a simulation with commercial tools on the Verilog RTL considering both compile time and run time, when booting an OS on a Rocket. However, the performance of C++ RTL models generated from Chisel is in the order of some KIPS. The space industry lacks of competence and experience in designing with high-level description languages, and modifying the autogenerated Verilog is not a viable option, as modifying autogenerate nonhuman readable code is risky and damages maintainability. Furthermore, autogenerated code may be not acceptable for hi-rel applications, as there is no established flow in the industry and an additional layer in between hardware description and implementation is added. IP cores for space are typically in VHDL, as it is historically widely adopted by the European space industry [90]. However, as opposed to Chisel, which has not been adopted by industry yet, new possibilities like SystemVerilog could also be taken into account as industry and academia heavily rely on this language for terrestrial applications. For instance, the PULP platform is written in SystemVerilog, which greatly simplifies verification, and this is becoming a sensible problem for space industry as VHDL was fine to verify simple IP cores but shows its limits as the complexity of the IP cores needed in space increases.

From the point of view of space applications, IP cores can be classified into two categories:

1) COTS IP cores (or non-FT): All the RISC-V IP cores analyzed in Sec. II are at this level, where specific features for fault tolerance calibrated for space applications are not given. LEON processors have non-FT versions that can be considered at a similar level of fault tolerance compared with COTS IP cores.

Table 5 Simulation models and their features

Sim. model (example)	Speed	Accuracy	Space Rel.	Typical use
ISS (Spike)	VH	L	M	ISA reference model
Modular platform (gem5)	$M+$ to $M-$	$M-$ to $M+$	L	Design exploration (processor arch.)
Virtual Prototype (SoCRocket)	H to M	$M-$ to $M+$	M	Design exploration (SoC level), SW dev.
Speed-optimized CA (TSIM)	H	H	H	SW development
High-level description (Chisel)	L	VH−	L	Parametric and fast design
RTL (VHDL/SystemVerilog)	L	VH	H	Design, functional simulation, synthesis
Gate-level (Verilog netlist)	VL	VH+	H	Gate-level simulation

Speed, accuracy, and space relevance are compared using a coarse scale from very high (VH) to very low (VL). A finer scale using "+" and "−" is used to indicate smaller differences.

Table 6 Comparison of area between LEON2 and LEON2FT on a 250 nm ASIC technology (the table is taken from [51])

Module	LEON2, mm^2	LEON2FT, mm^2	Area penalty, %
Integer unit	0.86	1.61	87
Cache controllers	0.17	0.35	105
Peripherals	0.45	0.90	100
Register file	0.19	0.24	26
Cache memory (16 KB)	2.42	2.59	7
Total	4.09	5.69	39

2) FT IP cores: Ref. [51] reports the penalties in terms of area (Table 6) and frequency for a LEON2FT without FPU against a LEON2 on the same 250 nm ASIC technology. The total overhead without considering RAM cells is instead 100% [51]. The main improvements for the FT version are TMR at flip-flop level with separate clock trees, two parity bits on caches, and (32,7) BCH code on the register file. The penalty in frequency compared with the non-FT version is relatively small (−7.4%) and it is due to the use of TMR flip-flops on the critical path. Furthermore, hardware overhead due to the handling of errors in the register file is optimized by reusing hardware already available for traps. For instance, when an uncorrectable error is found in the register file of the LEON2FT an error trap is generated. Instead, when a correctable error is found the register file is written back with corrected data and then the processor jumps back to the failing instruction [51].

The difference between the two types of IP cores is becoming less pronounced over time, as larger cache memories with smaller technologies tend to have considerable SER even in terrestrial applications (for instance, Rocket employs EDAC codes for the caches).

2. *Hardware Models*

1) FPGA: RISC-V processors for space should be portable to a large spectrum of FPGAs, like it is possible today with the LEON processors. Characteristics of FPGAs employed in the development of space systems vary depending on the purpose of the model (e.g., prototype vs flight system).

a) COTS SRAM-based (e.g., Xilinx Spartan6, Xilinx Kintex UltraScale, and Xilinx Artix7): They are the best solution for rapid prototyping and evaluation because of their mature toolchain, large capacity, and availability of several FPGA-specific IP cores from the vendors. However, SRAM-based FPGAs require an additional level of protection when operating in space compared with an ASIC because large part of the sensitive area is composed of the configuration memory. For the Ultrascale series, Xilinx provides the SEM IP core for single event upset (SEU) detection and correction in the configuration memory with additional fault injection capabilities to allow the user to evaluate the dependability of the final solution [91].

b) COTS flash-based (e.g., Microsemi IGLOO2, Microsemi ProASIC, Microsemi PolarFire, Microsemi SmartFusion2): They can be suitable platforms for low-cost missions, as flash-cells are inherently immune to SEUs. However, as these FPGAs are not explicitly validated for the space environment, other problems may arise, like single event transient (SET) in critical parts of the SoC (e.g., phase-locked loop), single event latchups (SELs), low-dose TID failures, and relatively high SEU in the user memory. For instance, Ref. [92] reports the results of heavy ions and proton tests for SEE of an IGLOO2. "High current" status triggered by particles (suspected SEL) is reported (while, for instance, space-grade RTG4 is SEL immune [93]).

c) Space-grade SRAM-based (e.g., Xilinx Virtex 5QV): These devices are typically well characterized in terms of effects of radiation and provide meaningful improvements over COTS SRAM based (Ref. [94] claims a 1000× improvement in SEU hardness of the cells of the configuration memory). However, SRAM cells of the configuration memory and user memory are still sensible to SEUs, and scrubbing of the configuration memory is then required. For instance, in [94] 3.80E-10 upsets/bit/day in geostationary orbit (GEO) at an altitude of 36,000 km are given for the configuration memory.

d) Space-grade flash-based (e.g., Microsemi RTG4): Flash cells are inherently immune to SEUs, but user memory is still vulnerable to SEU. This means that the same level of fault tolerance employed for ASICs can be considered enough. In [93] the tolerance to TID effect of the RTG4 is compared with the one of the SmartFusion2 and a significant improvement is found, with a TID tolerance tested up to 160 krad. No SELs were observed for an LET of 103 $(\mathrm{MeV} \cdot \mathrm{cm}^2)/\mathrm{mg}$ at 100°C, and the SER of the sequential elements (flip-flops with TMR) is at least three orders of magnitude better than the simple flip-flops in the SmartFusion2. No multiple-bit upsets (MBUs) were observed in the block RAM of the RTG4 because of interleaving.

e) Space-grade antifuse (e.g., Microsemi RTAX-S): The configuration is fixed, and so radiation cannot change configuration memory at all. This comes with large penalties in terms of frequency and capacity compared with space-grade flash-based FPGA. The vendor reports the RTAX-S to achieve a frequency 3 times lower than the one of the RTG4 and to have a capacity in terms of logic elements (LE) 7.5 times smaller.

2) ASIC prototype: A silicon-proven processor is typically seen as a mature processor that can be evaluated for inclusion within systems. In space the situation is more complicated, and for this reason an ASIC demonstrator is even more important when the additional difficulties in the use of FPGAs are considered.

a) Commercial technologies: While in the past ASIC based on commercial technologies were susceptible to destructive SEE events (SEL and single effect hard errors [95]), newer silicon-on-insulator (SOI) technologies are immune to SEL [96] and typically susceptible to hard failures only for TID. The SER caused by SEUs and SETs must nevertheless be evaluated. An example of an ASIC prototype based on commercial technology was the AT697E, an ASIC prototype of the LEON2FT from Atmel based on a 180 nm CMOS process.

b) RHBD technologies: In Europe many ASICs are based on the RHBD DARE library for the UMC 180 nm CMOS technology [97] and the newer C65SPACE (65 nm) by STMicroelectronics [98]. The AT697F was based on the ATC18RHA (180nm) rad-hard library, providing an increase in resilience to TID effects from 60 to 300 krad compared with the AT697E.

3) Space-grade component: The space-qualified version of a processor typically employs a ceramic and hermetic package, extended temperature range (from −55 to 125°C), and extended qualification flow (according to QML-V and QML-Q space grade flows [99]). Radiation performance may vary:

a) The "radiation-tolerant" ATmegaS128 has an LET threshold for SEL of 62.5 $(\mathrm{MeV} \cdot \mathrm{cm}^2)/\mathrm{mg}$ @ 125°C and TID tolerance up to 30 krad.

b) The "rad-hard" AT697F (from the same vendor) is reported to have TID tolerance up to 300 krad and LET threshold for SEL of 95 $(\mathrm{MeV} \cdot \mathrm{cm}^2)/\mathrm{mg}$ @ 125°C.

4) Flight-proven component: According to [3] a component has achieved flight heritage after 2 years of nominal performance under nominal mission conditions. The AT697 flew for the first time in 2008 and remained fully functional for more than 5 years, achieving flight heritage, which was key for broad market acceptance of the LEON2FT [100]. Together with the LEON processors, also IP libraries like GRLIB have been proved

Table 7 Simulation and hardware models from a COTS IP core to a flight-proven solution

Model	Strength	Weakness
COTS IP core (LEON2, Rocket)	Many available, large user base	Fault tolerance to be assessed
FT IP core (LEON2FT)	Designed keeping into account SEEs	Larger area and power consumption, lower maximum frequency
SRAM FPGA (LEON3FT on Virtex-5/ 5QV)	Large capacity and rapid prototyping (COTS versions)	Config. memory vulnerable to SEU (even for space-grade versions)
FLASH/Antifuse FPGA (LEON4FT on RTG4)	Configuration memory not vulnerable to SEU	Less capacity (compared with COTS SRAM FPGAs)
ASIC proto. (comm.) (AT697E)	High performance, large capacity; process is immune to SEL (SOI)	Risk of destructive SEEs (bulk CMOS) and/or low TID failures
ASIC proto. (RHBD) (AT697F before qualification)	Representative of final performance and radiation-hardness	Larger area, lower frequency compared with commercial technologies
Space-grade comp. (AT697F after qualification)	Mature solution (TRL = 8)	More expensive than ASIC prototypes, late on market (~year)
Flight-proven comp. (AT697F after flight)	Proven solution (TRL = 9)	At least 2 years needed for flight-proven (> 2-year-old technology)

successfully in space. Therefore integration in GRLIB would enable RISC-V processors to reuse all the interfaces already available for LEON-based SoCs with flight heritage like SpaceWire, MIL-STD-1553B, and CAN.

Given the discussion on the models adopted by the space industry, a roadmap to bring a non-FT or COTS IP core (e.g., LEON2 or Rocket) into a flight-proven ASIC component (e.g., AT697F) is given in Table 7, summarizing with examples which steps can be taken. A more detailed roadmap may also evaluate intermediate hardware models (e.g., space-grade FPGAs and ASIC prototypes) for flight.

IV. Conclusions

Several profiles of processors based on RISC-V, which are expected to improve satellite data system architectures compared with (mainly European) state-of-the-art processors for space, are identified. This paper shows how the open nature and modularity of the RISC-V ISA allow designers to implement the optimal microarchitecture for different applications, ranging from low-performance/low-power microcontrollers to OBCs to processors for payload applications to high-performance processors for ML and DSP. The large number of RISC-V implementations described, some of them with very different requirements and target applications, also shows how RISC-V allowed an unprecedented amount of research in a relative short time on a single ISA. Studies on the same open ISA can be easily employed to compare and evaluate microarchitectures to solve currently unaddressed needs in space systems. The introduction of RISC-V in space will contribute to providing a range of alternatives to proprietary solutions to enable new capabilities in satellites data systems, as concerns are growing about monopolistic positions in the embedded market. This work provides knowledge to choose proprietary solutions when strictly needed to achieve a certain level of performance or efficiency and choose several degrees of openness when possible, enabling the next generation of space data systems to be based on an effective mix of the two approaches. A flexible roadmap to bring RISC-V processors from COTS IP cores to flight-proven components is proposed, based on the models typically employed in the development of space processors. Future work will show how to enhance performance and fault tolerance for different profiles of RISC-V processors, evaluating the resulting FT IP cores with design explorations on ASIC technologies and FPGAs.

Acknowledgments

This work was supported by the European Space Agency under the NPI Program, Cobham Gaisler AB, and Delft University of Technology.

References

[1] Hillman, R., Swift, G., Layton, P., Conrad, M., Thibodeau, C., and Irom, F., "Space Processor Radiation Mitigation and Validation Techniques for an 1,800 MIPS Processor Board," *Proceedings of the 7th European Conference on Radiation and Its Effects on Components and Systems, RADECS*, IEEE Publ., Piscataway, NJ, 2003, pp. 347–352.

[2] Lentaris, G., Maragos, K., Stratakos, I., Papadopoulos, L., Papanikolaou, O., Soudris, D., Lourakis, M., Zabulis, X., GonzalezArjona, D., Furano, G., et al., "High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation," *Journal of Aerospace Information Systems*, Vol. 15, No. 4, 2018, pp. 178–192.
doi:10.2514/1.I010555

[3] ECSS, "ECSS-E-HB-11A Technology Readiness Level (TRL) Guidelines," 2016, https://ecss.nl/home/ecss-e-hb-11a-technology-readiness-level-trl-guidelines-1-march-2017/.

[4] Esposito, S., Albanese, C., Alderighi, M., Casini, F., Giganti, L., Esposti, M. L., Monteleone, C., and Violante, M., "COTS-Based High-Performance Computing for Space Applications," *IEEE Transactions on Nuclear Science*, Vol. 62, No. 6, 2015, pp. 2687–2694.
doi:10.1109/TNS.2015.2492824

[5] Pignol, M., "COTS-Based Applications in Space Avionics," *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, IEEE Publ., Piscataway, NJ, 2010, pp. 1213–1219.
doi:10.1109/DATE.2010.5456992

[6] Frasse-Sombet, S., Brunel, Y., and Mariottini, S., "Bringing High-Performance Microprocessors up to Space Level Reliability," *2015 IEEE on Aerospace Conference*, IEEE Publ., Piscataway, NJ, 2015, pp. 1–8.
doi:10.1109/AERO.2015.7118947

[7] Di Mascio, S., Menicucci, A., Furano, G., Monteleone, C., and Ottavi, M., "The Case for RISC-V in Space," *Applications in Electronics Pervading Industry, Environment and Society, ApplePies, 2018*, edited by S. Saponara, and A. De Gloria, Vol. 550, Lecture Notes in Electrical Engineering, Springer, Cham, 2019, pp. 319–325.
doi:10.1007/978-3-030-11973-7_37

[8] Asanović, K., and Patterson, D. A., "Instruction Sets Should be Free: The Case for Risc-v," EECS Department, Univ. of California, UCB/EECS-2014-146, Berkeley, CA, 2014.

[9] Shrimpton, S. J., "An Implementation of MIL-STD-1750 Airborne Computer Instruction Set Architecture," Royal Aircraft Establishment, RAE-FS-403, Farnborough, England, 1981.

[10] Furano, G., and Menicucci, A., *Roadmap for on-Board Processing and Data Handling Systems in Space*, Springer International Publishing, Cham, 2017, pp. 253–281.
doi:10.1007/978-3-319-54422-9_10

[11] Waterman, A., and Asanović, K., "The RISC-V Instruction Set Manual-Volume I: User-Level ISA-Document Version 2.2," *RISC-V Foundation*, May 2017, https://content.riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf.

[12] Salmon, L. G., "A Perspective on the Role of Open-Source IP in Government Electronic Systems," *7th RISC-V Workshop Proceedings*, Milpitas, CA, Nov. 2017.

[13] Costan, V., Lebedev, I. A., and Devadas, S., "Sanctum: Minimal Hardware Extensions for Strong Software Isolation," *USENIX Security Symposium*, Berkeley, CA, 2016, pp. 857–874.

[14] Menon, A., Murugan, S., Rebeiro, C., Gala, N., and Veezhinathan, K., "Shakti-T: A RISC-V Processor with Light Weight Security Extensions," *Proceedings of the Hardware and Architectural Support for Security and Privacy*, ACM, New York, 2017, pp. 2:1–2:8.
doi:10.1145/3092627.3092629

[15] Gil, D., Martínez, R., Busquets, J. V., Baraza, J. C., and Gil, P. J., "Fault Injection into VHDL Models: Experimental Validation of a Fault Tolerant Microcomputer System," *Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, Springer–Verlag, Berlin, 1999, pp. 191–208.
doi:10.1109/EURMIC.1998.711835

[16] Mukherjee, S. S., Weaver, C., Emer, J., Reinhardt, S. K., and Austin, T., "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE Computer Soc., Washington, D.C., 2003, pp. 29–40.
doi:10.1109/MICRO.2003.1253181

[17] Cho, H., "Impact of Microarchitectural Differences of RISC-V Processor Cores on Soft Error Effects," *IEEE Access*, Vol. 6, 2018, pp. 41302–41313.
doi:10.1109/Access.6287639

[18] Waterman, A., and Asanovic, K., "The RISC-V Instruction Set Manual, Volume II: Privileged Architecture," *RISC-V Foundation*, Document Version 1.1, 2017, https://content.riscv.org/wp-content/uploads/2017/05/riscv-privileged-v1.10.pdf.

[19] Western Digital, "RISC-V SweRV Core™ Available to Open Source Community," Available Online at https://blog.westerndigital.com/risc-v-swerv-core-open-source/ [accessed 31 May 2019].

[20] Gala, N., Menon, A., Bodduna, R., Madhusudan, G. S., and Kamakoti, V., "SHAKTI Processors: An Open-Source Hardware Initiative," *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, IEEE Publ., Piscataway, NJ, 2016, pp. 7–8.
doi:10.1109/VLSID.2016.130

[21] Gupta, S., Gala, N., Madhusudan, G. S., and Kamakoti, V., "SHAKTI-F: A Fault Tolerant Microprocessor Architecture," *2015 IEEE 24th Asian Test Symposium (ATS)*, IEEE Publ., Piscataway, NJ, 2015, pp. 163–168,
doi:10.1109/ATS.2015.35

[22] Boi, R., Brigati, S., Francesconi, F., Ghidini, C., Malcovati, P., Maloberti, F., and Poletti, M., "Switched-Capacitor Litton-Code Matched Filter for Satellite ODBH Bus," *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI (Cat. No.99CH36349)*, Vol. 2, IEEE Publ., Piscataway, NJ, 1999, pp. 69–72.
doi:10.1109/ISCAS.1999.780621

[23] Angulo, M., Mi, J. M., de Vicente, P., Prieto, M., Rodriguez, O., de la Fuente, E., and Palau, J., "Development of the MicroSat Programme at INTA," *Small Satellites for Earth Observation*, Springer, Dordrecht, The Netherlands, 2008, pp. 41–54.
doi:10.1007/978-1-4020-6943-7_4

[24] Fossion, M., Van Esbeen, A., Van Humbeeck, T., Geerts, Y., Geukens, E., Redant, S., Jansen, R., and Monteleone, C., "A Mixed-Signal Radhard Microcontroller: The Digital Programmable Controller (DPC)," *AMICSA 2014—Fifth International Workshop on Analogue and Mixed-Signal Integrated Circuits for Space Applications*, Geneva, Switzerland, 2014.

[25] Andersson, J., Hjorth, M., Johansson, F., and Habinc, S., "LEON Processor Devices for Space Missions: First 20 Years of LEON in Space," *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, IEEE Publ., Piscataway, NJ, 2017, pp. 136–141.
doi:10.1109/SMC-IT.2017.31

[26] Bruhn, F., Selin, P., Kalnins, I., Lyke, J., Rosengren-Calixte, J., and Nordenberg, R., "Quadsat/pnp: A Space-Plug-and-Play Architecture (Spa) Compliant Nanosatellite," *Infotech@Aerospace 2011*, AIAA Paper 2011-1575, 2011.
doi:10.2514/6.2011–1575

[27] Evans, A., Ortega, C. U., Marinis, K., Costenaro, E., Laroussi, H., Obbe, K., Magistrati, G., and Ferlet-Cavrois, V., "Heavy-Ion Micro Beam and Simulation Study of a Flash-Based FPGA Microcontroller Implementation," *IEEE Transactions on Nuclear Science*, Vol. 64, No. 1, 2017, pp. 504–511.
doi:10.1109/TNS.2016.2633401

[28] Schiavone, P. D., Conti, F., Rossi, D., Gautschi, M., Pullini, A., Flamand, E., and Benini, L., "Slow and Steady Wins the Race? A Comparison of Ultra-Low-Power RISC-V Cores for Internet-of-Things Applications," *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, IEEE Publ., Piscataway, NJ, 2017, pp. 1–8.
doi:10.1109/PATMOS.2017.8106976

[29] "Coremark Scores," Available Online at https://www.eembc.org/coremark/scores.php [accessed 21 Dec. 2018].

[30] Pullini, A., Rossi, D., Loi, I., Mauro, A. D., and Benini, L., "Mr. Wolf: A 1 GFLOP/s Energy-Proportional Parallel Ultra Low Power SoC for IOT Edge Processing," *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, IEEE Publ., Piscataway, NJ, 2018, pp. 274–277.
doi:10.1109/ESSCIRC.2018.8494247

[31] Waterman, A. S., "Design of the RISC-V Instruction Set Architecture," Ph.D. Thesis, UC Berkeley, Berkeley, CA, 2016.

[32] Cao, Z., Lv, Q., Wang, Y., Wen, M., Wu, N., and Zhang, C., "A Compression Instruction Set Design Based on RISC-V for Network Packet Forwarding," *Journal of Physics: Conference Series*, Vol. 1026, IOP Publishing, 2018, Paper 012001.
doi:10.1088/1742-6596/1026/1/012001

[33] Gautschi, M., Schiavone, P. D., Traber, A., Loi, I., Pullini, A., Rossi, D., Flamand, E., Gürkaynak, F. K., and Benini, L., "Near-Threshold RISC-V Core with DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 25, No. 10, 2017, pp. 2700–2713.
doi:10.1109/TVLSI.2017.2654506

[34] Cheikh, A., Cerutti, G., Mastrandrea, A., Menichelli, F., and Olivieri, M., "The Microarchitecture of a Multi-Threaded RISC-V Compliant Processing Core Family for IoT End-Nodes," *ApplePies 2017: Applications in Electronics Pervading Industry, Environment and Society*, Springer International Publ., Cham, 2019, pp. 89–97.
doi:10.1007/978-3-319-93082-4_12

[35] Caprita, H. V., and Popa, M., "Design Methods of Multithreaded Architectures for Multicore Microcontrollers," *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, IEEE Publ., Piscataway, NJ, 2011, pp. 427–432.
doi:10.1109/SACI.2011.5873041

[36] Mukherjee, S. S., Kontz, M., and Reinhardt, S. K., "Detailed Design and Evaluation of Redundant Multi-Threading Alternatives," *Proceedings 29th Annual International Symposium on Computer Architecture*, IEEE Publ., Piscataway, NJ, 2002, pp. 99–110.
doi:10.1109/ISCA.2002.1003566

[37] Dean, A. G., "Efficient Real-Time Fine-Grained Concurrency on Low-Cost Microcontrollers," *IEEE Micro*, Vol. 24, No. 4, 2004, pp. 10–22.
doi:10.1109/MM.2004.27

[38] Furano, G., "Issues in (Very) Rad Hard Systems: An ESA Perspective on Use of COTS and Space Grade in JUICE Mission," *JUICE Instrument Workshop*, Darmstadt, 2011.

[39] ESA, "GOCE," Available Online at https://earth.esa.int/web/eoportal/satellite-missions/g/goce[accessed 21 Dec. 2018].

[40] RUAG SPACE, "Command & Data Handling Factsheet," 2013, https://www.ruag.com/sites/default/files/2016-12/Command_Data_Handling.pdf.

[41] ESA, "GaiaNIR - CDF Study Report: CDF-175(C)," Oct. 2017.

[42] Fernández, M., Gioiosa, R., Quiñones, E., Fossati, L., Zulianello, M., and Cazorla, F. J., "Assessing the Suitability of the NGMP Multi-Core Processor in the Space Domain," *Proceedings of the Tenth ACM International Conference on Embedded Software*, ACM, New York, 2012, pp. 175–184.
doi:10.1145/2380356.2380389

[43] Li, X., Roychoudhury, A., and Mitra, T., "Modeling Out-of-Order Processors for WCET Analysis," *Real-Time Systems*, Vol. 34, No. 3, 2006, pp. 195–227.
doi:10.1007/s11241-006-9205-5

[44] Lovelly, T. M., and George, A. D., "Comparative Analysis of Present and Future Space-Grade Processors with Device Metrics," *Journal of Aerospace Information Systems*, Vol. 14, No. 3, 2017, pp. 184–197.
doi:10.2514/1.I010472

[45] Asanovic, K., Patterson, D. A., and Celio, C., "The Berkeley Out-of-Order Machine (Boom): An Industry-Competitive, Synthesizable, Parameterized Risc-v Processor," Univ. of California at Berkeley Tech. Rept. UCB/EECS-2015-167, Berkeley, CA, 2015.

[46] Synopsys, "ARC HS4x and HS4xD CPUs: New Dual-Issue Architecture Boosts Embedded Processor Performance," The Linley Group, May 2017, http://linleygroup.com/synopsys/whitepaper/pdf/.

[47] Bardizbanyan, A., and Larsson-Edefors, P., "Exploring Early and Late ALUs for Single-Issue in-Order Pipelines," *2015 33rd IEEE International Conference on Computer Design (ICCD)*, IEEE Publ., Piscataway, NJ, 2015, pp. 543–548.
doi:10.1109/ICCD.2015.7357163

[48] Hrishikesh, M. S., Jouppi, N. P., Farkas, K. I., Burger, D., Keckler, S. W., and Shivakumar, P., "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," *Proceedings 29th Annual International Symposium on Computer Architecture*, IEEE Publ., Piscataway, NJ, 2002, pp. 14–24.
doi:10.1109/ISCA.2002.1003558

[49] Asanovic, K., Avizienis, R., Bachrach, J., Beamer, S., Biancolin, D., Celio, C., Cook, H., Dabbelt, D., Hauser, J., Izraelevitz, A., et al., "The Rocket Chip Generator," EECS Department, University of California, Tech. Rept. UCB/EECS-2016-17, Berkeley, CA, 2016.

[50] Jouppi, N. P., "Cache Write Policies and Performance," *Proceedings of the 20th Annual International Symposium on Computer Architecture (ISCA '93)*, ACM, NY, pp. 191–201.
doi:10.1145/165123.165154

[51] Gaisler, J., "A Portable and Fault-Tolerant Microprocessor Based on the SPARC v8 Architecture," *Proceedings on International Conference on Dependable Systems and Networks*, IEEE Publ., Piscataway, NJ, 2002, pp. 409–415.

[52] Weigand, R., "NGMP → GR740. Status and Roadmap. Vision for Future," Available Online at http://microelectronics.esa.int/gr740/NGMP-Status-2014-11-06.pdf [accessed 21 Dec. 2018].

[53] Gustafson, J. L., "Reevaluating Amdahl's Law," *Communications of the ACM*, Vol. 31, No. 5, 1988, pp. 532–533.
doi:10.1145/42411.42415

[54] Gillette, A., Wilson, C., and George, A. D., "Efficient and Autonomous Processing and Classification of Images on Small Spacecraft," *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, IEEE Publ., Piscataway, NJ, 2017, pp. 135–141.
doi:10.1109/NAECON.2017.8268758

[55] Doubleday, J., Chien, S., Norton, C., Wagstaff, K., Thompson, D. R., Bellardo, J., Francis, C., and Baumgarten, E., "Autonomy for Remote Sensing - Experiences from the IPEX CubeSat," *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, IEEE Publ., Piscataway, NJ, 2015, pp. 5308–5311.
doi:10.1109/IGARSS.2015.7327033

[56] Izzo, D., and Pettazzi, L., "Autonomous and Distributed Motion Planning for Satellite Swarm," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 449–459.
doi:10.2514/1.22736

[57] Erickson, K., "Optimal Architecture for an Asteroid Mining Mission: Equipment Details and Integration," *Space 2006*, AIAA Paper 2006-7504, 2006,
doi:10.2514/6.2006-7504

[58] Gaines, D., Anderson, R., Doran, G., Huffman, W., Justice, H., Mackey, R., Rabideau, G., Vasavada, A., Verma, V., Estlin, T., et al., "Productivity Challenges for Mars Rover Operations," *Proceedings of 4th Workshop on Planning and Robotics (PlanRob)*, London, 2016, pp. 115–125.

[59] Mantovani, P., Cota, E. G., Pilato, C., Di Guglielmo, G., and Carloni, L. P., "Handling Large Data Sets for High-Performance Embedded Applications in Heterogeneous Systems-on-Chip," *2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*, IEEE Publ., Piscataway, NJ, 2016, pp. 1–10.
doi:10.1145/2968455.2968509

[60] Cui, H., Cipar, J., Ho, Q., Kim, J. K., Lee, S., Kumar, A., Wei, J., Dai, W., Ganger, G. R., Gibbons, P. B., et al., "Exploiting Bounded Staleness to Speed Up Big Data Analytics," *2014 USENIX Annual Technical Conference (USENIXATC 14)*, USENIX Association, Berkeley, CA, pp. 37–48.

[61] Lemley, J., Bazrafkan, S., and Corcoran, P., "Deep Learning for Consumer Devices and Services: Pushing the Limits for Machine Learning, Artificial Intelligence, and Computer Vision," *IEEE Consumer Electronics Magazine*, Vol. 6, No. 2, 2017, pp. 48–56.
doi:10.1109/MCE.2016.2640698

[62] Nickolls, J., Buck, I., Garland, M., and Skadron, K., "Scalable Parallel Programming with CUDA," *ACM SIGGRAPH 2008 Classes*, ACM, New York, 2008, p. 16.
doi:10.1145/1365490.1365500

[63] Sheffield, D., Anderson, M. J., Lee, Y., and Keutzer, K., "Hardware/Software Codesign for Mobile Speech Recognition," *INTERSPEECH, 14thAnnual Conference of the International Speech Communication Association,* , ISCA Archive, Lyon, France, 2013, pp. 627–631.

[64] Ng, H.-C., Liu, C., and So, H. K.-H., "A Soft Processor Overlay with Tightly-Coupled FPGA Accelerator," arXiv preprint, arXiv:1606.06483, 2016.

[65] Aporva, T. A. K. A.-H., Davidson, A. S. D. S., Rao, P. G. G. L. A., Torng, A. R. N. S. C., Xie, L. V. B. V. S., and Zhao, C. Z. R., "Experiences Using the RISC-V Ecosystem to Design an Accelerator-Centric SoC in TSMC 16nm," *1st Workshop on Computer Architecture Research with RISC-V (CARRV 2017)*, 2017.

[66] Barry, B., Brick, C., Connor, F., Donohoe, D., Moloney, D., Richmond, R., O'Riordan, M., and Toma, V., "Always-on Vision Processing Unit for Mobile Applications," *IEEE Micro*, Vol. 35, No. 2, 2015, pp. 56–66.
doi:10.1109/MM.2015.10

[67] Rossi, D., Conti, F., Marongiu, A., Pullini, A., Loi, I., Gautschi, M., Tagliavini, G., Capotondi, A., Flatresse, P., Benini, L., et al., "PULP: A Parallel Ultra Low Power Platform for Next Generation IoT Applications," *2015 IEEE Hot Chips 27 Symposium (HCS)*, IEEE Publ., Piscataway, NJ, 2015, pp. 1–39.
doi:10.1109/HOTCHIPS.2015.7477325

[68] Dabbelt, D., Schmidt, C., Love, E., Mao, H., Karandikar, S., and Asanović, K., "Vector Processors for Energy-Efficient Embedded Systems," *Proceedings of the Third ACM International Workshop on Many-core Embedded Systems*, ACM, New York, 2016, pp. 10–16.
doi:10.1145/2934495.2934497

[69] Lee, Y., Waterman, A., Avizienis, R., Cook, H., Sun, C., Stojanović, V., and Asanović, K., "A 45nm 1.3GHz 16.7 Double Precision GFLOPS/W RISC-V Processor with Vector Accelerators," *ESSCIRC 2014—40th European Solid State Circuits Conference (ESSCIRC)*, IEEE Publ., Piscataway, NJ, 2014, pp. 199–202.
doi:10.1109/ESSCIRC.2014.6942056

[70] Adapteva, "E16G301 EPIPHANY 16-CORE MICROPROCESSOR Datasheet," http://www.adapteva.com/docs/e16g301_datasheet.pdf.

[71] Adapteva, "E64G401 EPIPHANY 64 CORE MICROPROCESSOR Datasheet," http://adapteva.com/docs/e64g401_datasheet.pdf.

[72] Olofsson, A., “Epiphany-V: A 1024 Processor 64-Bit RISC System-on-Chip,” arXiv preprint arXiv:1610.01832, 2016.

[73] Steffl, S., and Reda, S., “LACore: A Supercomputing-Like Linear Algebra Accelerator for SoC-Based Designs,” *2017 IEEE International Conference on Computer Design (ICCD)*, IEEE Publ., Piscataway, NJ, 2017, pp. 137–144.
doi:10.1109/ICCD.2017.29

[74] Steffl, S., and Reda, S., “A RISC-V Based Linear Algebra Accelerator for SoC Designs Samuel Steffl, Brown University,” *7th RISC-V Workshop Proceedings*, Milpitas, CA, 2017.

[75] Aggarwal, V., Sabharwal, Y., Garg, R., and Heidelberger, P., “HPCC RandomAccess Benchmark for Next Generation Supercomputers,” *2009 IEEE International Symposium on Parallel Distributed Processing*, IEEE Publ., Piscataway, NJ, 2009, pp. 1–11.
doi:10.1109/IPDPS.2009.5161019

[76] Williams, S., Waterman, A., and Patterson, D., “Roofline: An Insightful Visual Performance Model for Multicore Architectures,” *Communications of the ACM*, Vol. 52, No. 4, 2009, pp. 65–76.
doi:10.1145/1498765

[77] Ofenbeck, G., Steinmann, R., Caparros, V., Spampinato, D. G., and Püschel, M., “Applying the Roofline Model,” *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE Publ., Piscataway, NJ, 2014, pp. 76–85.
doi:10.1109/ISPASS.2014.6844463

[78] “MLperf Website,” Available Online at https://mlperf.org/ [accessed 21 Dec. 2018].

[79] Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C., and Zaharia, M., “Dawnbench: An End-to-End Deep Learning Benchmark and Competition,” *NIPS ML Systems Workshop*, Long Beach, CA, 2017.

[80] Parker, M., “A Case for User-Level Interrupts,” *ACM SIGARCH Computer Architecture News*, Vol. 30, No. 3, 2002, p. 17.
doi:10.1145/571666

[81] Clark, M., and Hoult, B., “rv8: a High Performance RISC-V to x86 Binary Translator,” *First Workshop on Computer Architecture Research with RISC-V (CARRV)*, Boston, MA, 2017.

[82] Roelke, A., and Stan, M. R., “Risc5: Implementing the RISC-V ISA in Gem5,” *1st Workshop on Computer Architecture Research with RISC-V (CARRV)*, Boston, MA, 2017.

[83] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., et al., “The Gem5 Simulator,” *ACM SIGARCH Computer Architecture News*, Vol. 39, No. 2, 2011, pp. 1–7.
doi:10.1145/2024716

[84] Ta, T., Cheng, L., and Batten, C., “Simulating Multi-Core RISC-V Systems in Gem5,” *Workshop on Computer Architecture Research with RISC-V*, Los Angeles, CA, 2018.

[85] Schuster, T., Meyer, R., Buchty, R., Fossati, L., and Berekovic, M., “SoCRocket—A Virtual Platform for the European Space Agency’s SoC Development,” *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, IEEE Publ., Piscataway, NJ, 2014, pp. 1–7.
doi:10.1109/ReCoSoC.2014.6860690

[86] Herdt, V., Große, D., Le, H. M., and Drechsler, R., “Extensible and Configurable RISC-V Based Virtual Prototype,” *2018 Forum on Specification & Design Languages (FDL)*, IEEE Publ., Piscataway, NJ, 2018, pp. 5–16.
doi:10.1109/FDL.2018.8524047

[87] Charif, A., Busnot, G., Mameesh, R., Sassolas, T., and Ventroux, N., “Fast Virtual Prototyping for Embedded Computing Systems Design and Exploration,” *11th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, ACM, New York, 2019.

[88] Cordero, F., Mendes, J., Kuppusamy, B., Dathe, T., Irvine, M., and Williams, A., “A Cost-Effective Software Development and Validation Environment and Approach for LEON Based Satellite & Payload Subsystems,” *Proceedings of 5th International Conference on Recent Advances in Space Technologies—RAST2011*, IEEE Publ., Piscataway, NJ, 2011, pp. 511–516.
doi:10.1109/RAST.2011.5966889

[89] Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avižienis, R., Wawrzynek, J., and Asanović, K., “Chisel: Constructing Hardware in a Scala Embedded Language,” *DAC Design Automation Conference 2012*, IEEE Publ., Piscataway, NJ, 2012, pp. 1212–1221.
doi:10.1145/2228360.2228584

[90] Habinc, S., and Sinander, P., “Using VHDL for Board Level Simulation,” *IEEE Design Test of Computers*, Vol. 13, No. 3, 1996, pp. 66–78.
doi:10.1109/54.536097

[91] Maillard, P., Hart, M., Barton, J., Chang, P., Welter, M., Le, R., Ismail, R., and Crabill, E., “Single-Event Upsets Characterization & Evaluation of Xilinx UltraScale™ Soft Error Mitigation (SEM IP) Tool,” *2016 IEEE Radiation Effects Data Workshop (REDW)*, IEEE Publ., Piscataway, NJ, 2016, pp. 1–4.
doi:10.1109/NSREC.2016.7891745

[92] Davis, S. C., Koga, R., and George, J. S., “Proton and Heavy Ion Testing of the Microsemi Igloo2 FPGA,” *2017 IEEE Radiation Effects Data Workshop (REDW)*, IEEE Publ., Piscataway, NJ, 2017, pp. 1–6.
doi:10.1109/NSREC.2017.8115454

[93] Rezzak, N., Wang, J., Dsilva, D., and Jat, N., “TID and SEE Characterization of Microsemi’s 4th Generation Radiation Tolerant RTG4 Flash-Based FPGA,” *2015 IEEE Radiation Effects Data Workshop (REDW)*, IEEE Publ., Piscataway, NJ, 2015, pp. 1–6.
doi:10.1109/REDW.2015.7336739

[94] Xilinx, “DS192 - Radiation-Hardened, Space-Grade Virtex-5QV Family Data Sheet: Overview,”

[95] Haran, A., Barak, J., David, D., Keren, E., Refaeli, N., and Rapaport, S., “Single Event Hard Errors in SRAM Under Heavy Ion Irradiation,” *IEEE Transactions on Nuclear Science*, Vol. 61, No. 5, 2014, pp. 2702–2710.
doi:10.1109/TNS.2014.2345697

[96] Roche, P., Autran, J., Gasiot, G., and Munteanu, D., “Technology Downscaling Worsening Radiation Effects in Bulk: SOI to the Rescue,” *2013 IEEE International Electron Devices Meeting*, IEEE Publ., Piscataway, NJ, 2013, pp. 31.1.1–31.1.4.
doi:10.1109/IEDM.2013.6724728

[97] Redant, S., Marec, R., Baguena, L., Liegeon, E., Soucarre, J., Thielen, B. V., Beeckman, G., Ribeiro, P., Fernandez-Leon, A., Glass, B., et al., “Radiation Test Results on First Silicon in the Design Against Radiation Effects (DARE) Library,” *IEEE Transactions on Nuclear Science*, Vol. 52, No. 5, 2005, pp. 1550–1554.
doi:10.1109/TNS.2005.855818

[98] Clerc, S., Abouzeid, F., Gasiot, G., Daveau, J., Bottoni, C., Glorieux, M., Autran, J., Cacho, F., Huard, V., Dugoujon, L., et al., “Space Radiation and Reliability Qualifications on 65nm CMOS 600MHz Microprocessors,” *2013 IEEE International Reliability Physics Symposium (IRPS)*, IEEE Publ., Piscataway, NJ, 2013, pp. 6C.1.1–6C.1.7.
doi:10.1109/IRPS.2013.6532051

[99] Department of Defense, “MIL-PRF-38535J, Performance Specification, Integrated Circuits (Microcircuits) Manufacturing, General Specification for,” 2010.

[100] ESA, “LEON’s First Flights,” Available Online at http://www.esa.int/Our_Activities/Space_Engineering_Technology/LEON_s_first_flights [accessed 21 Dec. 2018].

U. Durak
Associate Editor