IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

On-Board Computer for CubeSats: State-of-the-Art and Future Trends

ANGELA CRATERE[1], LEANDRO GAGLIARDI[2], GABRIEL A. SANCA[3], FEDERICO GOLMAR[4], and FRANCESCO DELL'OLIO[5] (Senior, IEEE)

[1]Micro Nano Sensor Group, Department of Electrical and Information Engineering, Polytechnic University of Bari, 70126 BA, Italy (e-mail: a.cratere@phd.poliba.it)
[2]Instituto de Ciencias Físicas, Universidad de San Martín-CONICET, Yapeyú 2068, 1650, Buenos Aires, Argentina (e-mail: lgagliardi@unsam.edu.ar)
[3]Instituto de Ciencias Físicas, Universidad de San Martín-CONICET, Yapeyú 2068, 1650, Buenos Aires, Argentina (e-mail: gsanca@unsam.edu.ar)
[4]Instituto de Ciencias Físicas, Universidad de San Martín-CONICET, Yapeyú 2068, 1650, Buenos Aires, Argentina (e-mail: fgolmar@unsam.edu.ar)
[5]Micro Nano Sensor Group, Department of Electrical and Information Engineering, Polytechnic University of Bari, 70126 BA, Italy (e-mail: francesco.dellolio@poliba.it)

Corresponding author: Francesco Dell'Olio (e-mail: francesco.dellolio@poliba.it).

ABSTRACT Over the past three decades, the acceptance of higher risk thresholds within the space industry has facilitated the widespread integration of commercial off-the-shelf (COTS) components into avionics and payloads, leading to a remarkable transformation in the design of space missions. This transformation has led to the emergence of the *New Space Economy* and the widespread adoption of lean or small satellites in general, particularly CubeSats. CubeSats are now widely used in commercial, scientific, and research applications due to their versatility, affordability, simplicity of development, and accelerated development timelines. On-board computing plays a crucial role in the design of CubeSat missions, as increasingly high-performance computational requirements are needed to meet the challenges of future missions. This paper systematically reviews the state-of-the-art of CubeSat Command and Data Handling (C&DH) sub-system, covering both hardware components and flight software (FSW) development frameworks. It presents an analysis of the key features and recent developments of on-board computers (OBCs) in commercial and academic institutional projects funded by governments, agencies and public institutions. It further examines the effects of space radiation on avionics components and discusses the main fault-tolerance techniques used in CubeSat platforms. Finally, this paper highlights trends and hazards for future CubeSat avionics and identify potential directions for future developments in high-performance on-board computing. By synthesizing contemporary research and industry insights, this paper aims to shed light on CubeSat OBC design, providing an overview of the existing technology landscape and the challenges to be addressed for next-generation mission needs.

INDEX TERMS Command and Data Handling (C&DH), CubeSats, On-board computer (OBC), Small satellite avionics

I. INTRODUCTION

ALL space missions aim to generate and collect data. Data acquisition is a key consideration in the design of space systems, as it plays a fundamental role in defining their objectives. Generally, space missions are based on a main payload that produces data specifically for different purposes, such as Earth observation (EO) and remote sensing, astrophysics, space environment characterization, heliophysics and space weather, and the demonstration of new technologies. Such payload data constitute the largest data set of a mission. However, space missions additionally produce a wide range of other data, including telemetry (TM), which records the health of the spacecraft and its subsystems (power supply voltages, current draw and temperatures), attitude data, log files and configuration settings [1]. The command and data handling (C&DH) subsystem is responsible for managing the data produced onboard the satellite. This system's functions include the collection, preparation, and storage of both housekeeping and mission data, which can be used onboard or transmitted to ground stations. In addition, the C&DH subsystem often executes additional tasks, including receiving, validating, decoding, and distributing commands to other subsystems, detecting faults arising from the interaction of space radiation with electronic components and subsequent system recovery, security functions, and spacecraft timekeeping [2, 3].

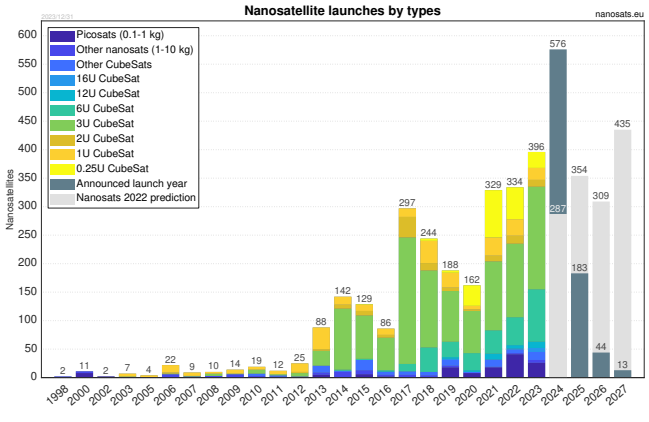

FIGURE 1. Yearly launches by nanosatellite types. From [4].

In recent decades, the space sector has undergone a significant transformation due to advances in miniaturization techniques for payloads and electronic components. This factor, combined with an increased risk tolerance leading to the widespread use of commercial off-the-shelf (COTS) electronics also in space, has resulted in a growing interest in new mission concepts based on limited size, weight, power, and cost (SWaP-C), such as CubeSats. For approximately 15 years, CubeSats have been key instrument technologies dominating the space market and facilitating access to space. Until 1 January 2024, more than 2000 CubeSats have been launched [4]. According to some forecasts, almost two thousand CubeSats will be launched in the next four years (Fig. 1, [4]).

The CubeSat standard, formally known as the CubeSat Design Specification (CDS), was introduced by Stanford and California Polytechnic State Universities in 1999 [5, 6]. It specifies that a standard unit (1U) is a cube of 10 cm on one side (to be precise, 10 cm $\times$ 10 cm $\times$ 11.35 cm) with a mass of up to 2 kg [7]. A 1U can be used as a standalone satellite or can be arranged with others to build larger CubeSats. The main advantage of this standardization is the possibility for launch vehicle manufacturers to implement universal deployment systems independently of the CubeSat manufacturer, such as the Poly Picosat Orbital Deployer (P-POD) or the NanoRacks CubeSat Deployer (NRCSD;[8]). The CubeSat standard also provides specific design requirements and protocols, simplifying the development process. Following the popularity of the 1U and 3U CubeSats, an advanced standard for larger CubeSats (6U, 12U and 27U) was proposed in 2011 to enable the enhancement of CubeSat capabilities and increase utilization [9].

Although originally intended as educational tools or low-cost technology demonstrators [6, 10, 11], CubeSats are currently used in a variety of applications, from commercial and telecommunication purposes [12, 13] to high-value scientific missions [8, 11]. They offer significant cost advantages compared to traditional satellites, with a development cost of approximately USD 200 000, as opposed to the USD 150-350 million required for conventional satellites. In addition, CubeSats typically have short development times of less than one or two years, compared to the 5-15 years that are required for large mission concepts [14]. These advantages have motivated worldwide government agencies to actively support CubeSat missions, leading to the emergence of long-term programs dedicated to CubeSat technologies, such as the National Aeronautics and Space Administration's (NASA's) Cubesat Launch Initiative [15]. The European Space Agency (ESA) has provided funding for several CubeSat missions, including those with interplanetary targets (e.g., [16, 17]), as part of its General Support Technology Programme (GSTP). The Italian Space Agency (ASI) has funded the Alcor program, which includes at least 20 CubeSat missions, and has participated in the development of the first two Italian CubeSats for deep-space applications, namely LiciaCube [18] and ArgoMoon [19]. The introduction of the CubeSat standard has also opened up space exploration to private companies, leading to the emergence of the so-called *New Space Economy*, and has provided scientific opportunities for small educational institutions.

Recently, the popularity of CubeSat platforms has been driven by advances in silicon processes and electronics miniaturization techniques, which enable the integration of complex processor architectures into a single Field Programmable Gate Array (FPGA) or System-On-a-Chip (SoC; [3]). CubeSat C&DH subsystems have reached a state of relative maturity, with a wide range of implementation options available. The number of companies and research institutes worldwide developing their own avionics subsystem for CubeSats is increasing steadily [4]. In this context, this paper aims to systematically review the state-of-the-art in OBCs for CubeSats, primarily targeting research and commercial projects funded by government agencies and private companies. Although there are several works dealing with the C&DH subsystem [2, 3, 20–22], even for small satellites [23, 24], none of them discusses in detail OBCs intended for CubeSat platforms ([1] provides only a general overview of CubeSat subsystems). Therefore, this paper meticulously reviews recent advances in CubeSat OBCs, in order to provide developers and researchers with an understanding of currently available technologies, their limitations, and the breakthroughs required for future mission needs. The purpose is to provide an overview of existing and developing architectures, valuable both to illustrate existing options for those choosing to purchase a commercial OBC for their mission, and to support architecture and component selection for those deciding to design a customized OBC based on mission-specific requirements.

The structure of the paper is as follows: §II provides basic knowledge about the C&DH subsystem, its functions, its general architecture, and the main implementation options used in CubeSats. Furthermore, it analyzes the effects of space radiation on avionics systems and the main mitigation techniques used. §III reviews the OBC systems currently available on the market or under development, analyzing their architecture and performance based on the nature of

their processing core. §IV discusses the tools used for flight software (FSW) development. §V presents future trends in avionics, identifying potential directions for next-generation systems. Finally, §VI summarizes the main lessons learned from this literature review and concludes the paper.

II. OVERVIEW ON C&DH SUBSYSTEM: FUNCTIONS, COMPONENTS AND TECHNIQUES FOR RADIATION EFFECT MITIGATION

In this review, we specifically refer to the term *avionics* as the subsystems, electronic components and functions featured in the C&DH module.

Typically, there are two data management segments within a classical C&DH subsystem, each of which has different purposes and functions [20]. The first segment is responsible for monitoring and surveilling the satellite bus, while the second segment aims to monitor the payload, as well as process the data collected by the payload to support the spacecraft's science mission. These two data management segments can be concentrated in a single processing system or distributed between two separate processing units, namely the main OBC and the payload computer. Therefore, there are two different approaches in the design of the C&DH subsystem. In *centralized architectures*, all functions are concentrated in a single powerful and reliable computing infrastructure, which serves as the main OBC for satellite platform monitoring and TM management (including the processing of data produced by the attitude determination and control subsystem; ACDS), and as a processing unit for payload data. In some cases, this approach has been implemented for smaller satellites due to the constraints on the mass and volume of electronic components imposed by their limited size (e.g.,[25]). However, it is generally poorly adopted because this type of architecture suffers from potential system-level failures due to its dependence on a single processor or computing unit. Moreover, this design requires high energy consumption, limited system reconfiguration possibilities and complex interfaces [23]. A centralized OBC, which also incorporates payload computer functions, is poorly scalable and must be redesigned for each mission. In *distributed architectures*, which are the most commonly used approach, functions are divided and shared across a computing architecture consisting of multiple modules. In this case, platform and payload control functions are assigned to different devices and processors. This approach is favoured because basic satellite tasks (such as platform management, TC decoding and maintenance reports) consume less power. More demanding tasks, such as attitude determination and control, processing of scientific data produced by the payload, intensive signal processing (e.g., data compression, time/frequency domain filtering, and feature extraction), fault monitoring or cryptography, require very high processing power and more advanced processors [21]. A distributed approach ensures the scalability of the OBC, as the modular architecture allows the hardware and software subsystems to be reused in future missions.

The design of the C&DH subsystem is a very demanding process that uses typical system engineering design techniques. It starts with the definition of the system functions, requirements, and specifications, and leads to the definition of the system architecture and the selection of electronic components. This is often an iterative process, which should consider the requirements of other satellite subsystems (this approach is called *concurrent engineering*) and, since these are systems intended for space, also the extremely harsh environmental conditions and the effects of radiation. In this section, we present the conventional avionics architecture typically used as a reference in the C&DH subsystem and OBC design process and analyze in detail the effects of radiation in avionics systems.

A. CONVENTIONAL AVIONICS ARCHITECTURE

The design of the avionics architecture is generally influenced by the nature of the mission, leading to significant variability in existing avionics systems. Space agencies, institutions and companies have long emphasized the need to standardize avionics systems, with the aim of making these systems scalable, enabling the reuse of elements and modules in different missions, and increasing the efficiency of subsystem design, leading to a reduction in costs and development time. To achieve this goal, it is essential to identify recurring architectural elements and precisely define their functions, interfaces, and interconnection protocols. [3, 22]. In this context, the ESA's Space Avionics Open Interface Architecture (SAVOIR) initiative [26] established a functional architecture for classical satellites that serves as a reference for avionics standardization efforts.

Fig. 2 shows a functional view of the SAVOIR reference architecture. In this general concept, the OBC functions include [3, 22]:

- TM handling functions. These functions include the collection of fundamental TM and the generation/encoding of TM data packets according to the Telemetry Transfer Frame (TTF) protocol. This standardizes the data structure for space data transmission over a TM link.
- Telecommand (TC) handling functions. These include receiving, authenticating, and decoding TCs sent from ground stations, distributing commands to control vital spacecraft functions, and implementing security measures to protect against unauthorized TCs. Optionally, they could also provide for the cryptography of data transmitted on the downlink.
- I/O peripheral and interface (I/F) handling functions. These support physical sensor and actuator I/Fs to acquire essential spacecraft data. Sensor monitoring can be managed directly in point-to-point mode or via a data concentrator function, typically implemented in one or more remote terminal units (RTUs)/remote interface units (RIUs).
- Time management functions. These provide a timer and generate events of synchronization, including the potential inclusion of Global Navigation Satellite System (GNSS).

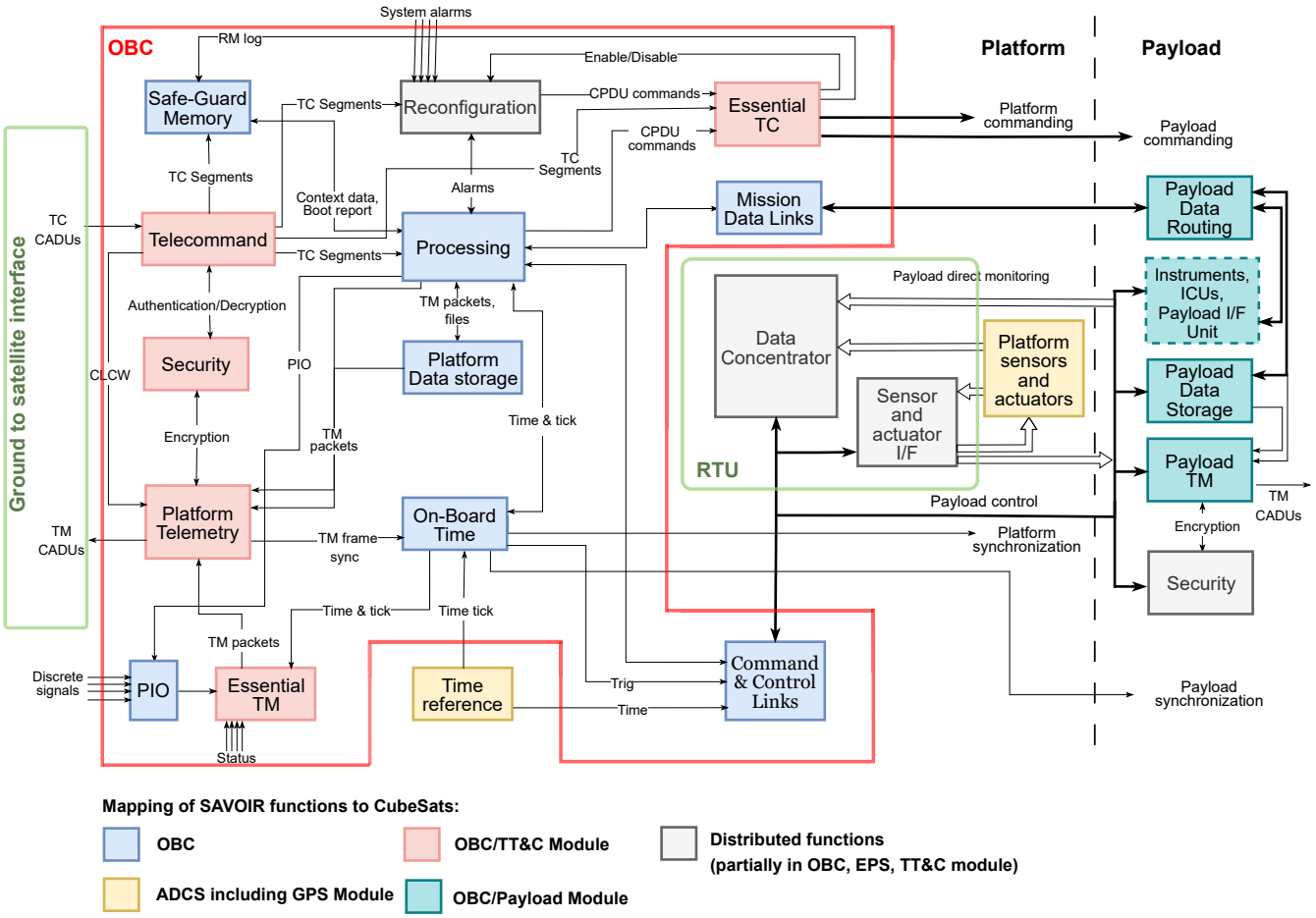

FIGURE 2. SAVOIR avionics functional reference architecture [26]. The red box marks the functional blocks included in the OBC. The colored squares map the general SAVOIR functions in the various subsystems of a CubeSat. Depending on the specific CubeSat mission, TCs and TM can be handled by the main OBC or the telemetry, tracking, and command (TT&C) subsystem. Similarly, the management of the payload data and its TM may be the task of the OBC, the payload computer or both.

- Processing and storage functions. These are responsible not only for storing and processing the critical satellite bus data but also for storing and executing the FSW.
- Fault detection and reconfiguration functions. These identify and correct faults, thus maintaining the correct operation of the processing functions even when errors occur.
- Payload data routing functions. These manage the monitoring and control of the payload units. Optionally, a function for storing telemetric payload data during periods when there is no contact with the ground stations can be included.

In the case of conventional satellites, the term C&DH subsystem is a hyperonym for various platforms and devices for processing data onboard the satellite, while the on-board computer (OBC) usually refers to hardware and software components specifically designed to manage the satellite platform. The FWS component (§IV) is responsible for space platform surveillance and control tasks, autonomy, and TM data processing. The HW functionalities of the OBC are limited to logic-arithmetic operations, data storage, and signal and data transmission. In the case of CubeSats, this distinction becomes more blurred, as the OBC can in many cases perform several functions within the C&DH subsystem and share tasks with other subsystems, as shown in Fig. 2. This is because there is no single, standardized approach to designing OBCs for nanosatellite platforms, and the specific functions often depend on the mission case.

The core of an OBC is the Central Processing Unit (CPU), which is responsible for managing high-level processes within the system. In general, several CPU implementation options are available, using different embedded processors. For CubeSats platforms, most often adopted CPU architectures are quite varied. Early systems use microcontroller (µC)-based architectures for their simplicity of implementation (often integrated with FPGA-based HW accelerators). More recent systems typically opt for integrated SoC architectures. The use of ASICs in CubeSat avionics systems is

uncommon, mainly because their design and implementation are complex and expensive. Further discussion of the OBC implementation choices in CubeSats is provided in §III.

B. RADIATION EFFECTS IN SPACE

The space environment poses harsh challenges to electronic components, mainly due to ubiquitous radiation. Space radiation produces a chaotic and inhomogeneous environment, characterized by a wide range of particles, energies, and fluxes and is strongly influenced by the level of solar activity. Radiation risk changes significantly in different orbital regions, such as low Earth orbits (LEOs), geostationary orbits (GEOs), and deep space trajectories. Three main sources of space radiation have been identified [27]:

- **Radiation belts**. These belts surround planets that have a magnetic field, such as Earth, Jupiter, Saturn and Uranus. These belts are responsible for trapping charged particles, especially electrons (e^-) and protons (p^+). The magnetic field accelerates e^- up to energies of the order of 30 MeV and p^+ up to energies of the order of 500 MeV, creating potentially dangerous zones for electronic devices near these celestial bodies. Around the Earth, the radiation belt is called the *Van Allen belt* and is a particularly critical source of radiation for LEO missions. However, this radiation source must also be considered when planning fly-by or randez-vouz missions in proximity to other planets with a magnetosphere.
- **Solar flares**. During solar events, such as so-called flares, solar particle events (SPEs) or coronal mass ejections (CMEs), p^+ with energies of up to 500 MeV and, to a lesser extent, heavy ions with energies of up to 10 MeV/nucleon can be emitted. Flares are influenced by the solar cycle, with a higher number of events occurring during periods of solar maximum.
- **Cosmic rays**. These high-energy ion streams come from outside the Solar System, probably produced by supernova explosions. Interstellar electromagnetic fields and shock waves scatter the ions and accelerate them to energies of thousands of GeV.

Radiation can damage or cause malfunctioning of electronic devices and systems, causing the so-called radiation-induced effects [28, 29]. These effects can have a substantial impact on the reliability of space systems, causing *faults*, i.e., incorrect hardware or software states, resulting in *errors* in programs or data structures. Such errors can lead to *failures* of space system components [30]. Radiation-induced errors can be classified into two categories: "soft" errors, which can be recovered with power cycles and can cause only temporary component failures, and "hard" errors, which can lead to permanent effects, causing hardware degradation and/or damage [29].

The effects of radiation on semiconductors should not be underestimated during the C&DH design, especially in CubeSats, in which COTS components are commonly used. These effects can be broadly classified into two main groups: cumulative effects and single-event effects (SEEs). The cumulative effects comprise ionizing phenomena, such as the total ionizing dose (TID), and nonionizing phenomena, such as the displacement damage (DD). Cumulative effects, as the term indicates, involve gradual changes in the operational parameters of devices. SEEs, instead, induce abrupt alterations or transient behavior in circuits. The following subsections analyze these effects and their impact on integrated circuit (IC) technologies in detail.

1) Cumulative effects

TID accounts for the amount of radiation that impacts electronic components during mission lifetime and is a critical parameter for the design of electrical circuits incorporating metal-oxide-semiconductor field-effect transistors (MOSFETs;[29, 31]). Charged particles, especially e^- and p^+, can directly or indirectly ionize semiconductors, creating electron-hole (*e-h*) pairs in the silicon dioxide layer. Fig. 3 shows the physical process behind the TID degradation mechanism and its impact on the behavior of a MOSFET. The holes created within the oxides led to progressive changes in the device performance parameters. For instance, positive charges collected in gate oxides lead to a decrease and increase in threshold voltage in N-MOS and P-MOS transistors, respectively, as positive charges gradually activate or inhibit gate activation [28, 31]. In addition to threshold voltage shifts, trapped charges are also responsible for increased current leakage, reduced carrier mobility, and increased noise levels. In worst-case scenario, functionality may be completely disabled because of the high leakage current and inability to shut off current between the transistor source and drain [32, 33].

TID is typically measured in rad, where 1 rad(material) is defined by an amount of energy equal to 100 erg deposited per gram of target material. Since the energy absorbed per unit mass varies among materials, the type of material is always specified [e.g., rad (Si)]. The SI unit is gray [Gy], which is equivalent to 100 rad.

Being a cumulative and long-term effect, TID increases over time, causing gradual semiconductor performance degradation and eventual failure of MOS devices [29, 32, 33]. Mitigation of this effect should always be considered in the design of avionics systems for space systems. TID can be mitigated through shielding, careful selection of inherently radiation-resistant components, or redundancy (§II-C). However, it must be emphasized that most CubeSat missions operate in LEO and with a short lifetime [34]. This considerably reduces the risk of TID failures for such missions. Short-duration missions in LEOs typically encounter relatively low TIDs, ranging from 1 to 10 krad(Si) per year, depending on shielding [35–38]. Commercial ICs typically fail after being exposed to 3-30 krad of radiation [39]. Furthermore, TID is a relevant - but not critical - factor in modern COTS µCs because of manufacturing technology and size scaling, miniaturized oxides in sub-90 nm devices being less susceptible to charge accumulation. Instead, these devices are much more

vulnerable to SEEs [33]. The risk of degradation for TID increases dramatically for long-duration missions or deep-space trajectories.

The second main cumulative radiation effect, called the DD effect, refers to the gradual degradation of the electrical and optical properties of semiconductor devices due to structural damage in the crystal lattice. This damage is caused by collisions of nonionizing radiation particles, which displace Si atoms from their original lattice sites [31, 40]. In space, p^+, neutrons, and heavy ions are the main sources of DD, while e^- contribute less because of their smaller cross section. Photons can also cause DD indirectly by producing high-energy secondary e^- through the Compton effect, especially with prolonged or repeated exposures [33, 41]. DD primarily affects devices whose operating characteristics depend on the bulk properties of the materials, such as photonic and optoelectronic devices. On satellites, the most affected components are photovoltaic cells in solar panels, charge-coupled devices (CCDs) used in many payloads as particle detectors, and photodiodes used in optical communications and camera chips [33]. The performance of MOS devices in OBCs is relatively insensitive to the DD, as they are surface devices that are less affected by bulk defects. Shielding and careful component selection should also protect any critical parts of OBCs against this effect. [41].

2) SEEs

SEEs are caused by the prompt interaction of a radiation particle (with energy up to several hundred MeV/nucleon in the case of heavy ions) with MOS devices. SEEs are not cumulative effects. The physical mechanism that produces SEEs can be attributed to the charge deposition induced by the passage of protons and heavy ions in the oxide, followed by the charge collection at the output node of the circuit [27, 31]. Charge collection occurs immediately after the creation of traces left by ionizing particles in the oxide and is driven by three main processes, namely, drift in the depletion region, diffusion, and channeling, as shown in Fig. 3. These three charge collection processes result in a transient current pulse being driven through the device.

Charge deposition is usually modeled using the linear energy transfer (LET) quantity [27, 29], which is the amount of energy transferred per unit length traveled in the target material per unit specific mass density (MeV $\cdot$ cm^2/mg). The higher the energy of a charged particle crossing a MOS device, the more energy is deposited on it. Based on the quantity of energy deposited, various effects result, such as single-event transients (SETs), single-event upsets (SEUs) and single-event latch-ups (SELs). The former two are typically soft failures, while the latter is a hard error.

a) **SET**. This term refers to the creation of a transient spike of current or voltage in the signal path, which causes various effects [29]. These are mostly soft errors, often repairable with power cycling. One possible effect of a SET is the interaction of the produced current pulse with the electronic system internal clock signal. This interaction can widen the clock signal at the trailing edge and can narrow it at the leading edge. This affects the processing speed of the system as a function of the clock signal [42]. In complex systems with synchronized computing nodes, the precise synchronization of all nodes can be severely compromised by a critical SET. Since charged particles can strike electronic components at any time and any place, there is no possible countermeasure to protect the system against an SET except complete radiation shielding [29]. If the voltage spike is captured by a memory cell, register or flip-flop and causes a state change, the SET can become an SEU.

b) **SEU**. This occurs when the energy transfer induced by a charged particle causes a stored bit to change, resulting in a change in the state of a flip-flop or memory unit cell (from 1 to 0 or vice versa). The state change can also affect several adjacent memory cells, resulting in a so-called multiple-bit upset (MBU). The effect of an SEU depends on the location and function of the affected bit. Nondestructive effects include corruption of the information stored in a memory element and thus a change in its state. This can be resolved by updating the elements with the correct value (memory scrubbing or device restart). Destructive effects can result in damage to the CPU program, such as computation errors, deadlocks, or incorrect command execution [29]. In LEO, p^+ trapped in the radiation belt and produced during SPEs are the most significant source of SEUs [43, 44]. SEUs are considered a common occurrence for static random access memory (SRAM) devices exposed to radiation, since such devices are based on flip-flop technology. In contrast, flash devices, which are based on electrically erasable programmable read-only memory (EEPROM) technology, are immune to SEUs [29]. Component reliability with respect to SEUs is a major concern when developing OBCs for space applications, especially when COTS components are used. Although cumulative effects and SETs can be mitigated (even partially) by using adequate shielding, this is often not enough to prevent SEU-induced errors. Therefore, their management through the use of fault-tolerant techniques is critical in spacecraft. Watchdog timers and information redundancy techniques, such as error detection and correction (EDAC) schemes, are two of the most classic methods used to mitigate SEUs (see §II-C).

c) **SEL**. This is a hard error. It occurs when the energy of the incident particle is so high that it induces an anomalous high-current state in the device, resulting in permanent device failure. If there is no overcurrent protection circuit, the device will burn out. This makes SEL one of the most dangerous SEEs for MOS devices [29]. SELs can be avoided by controlling the power consumption of the chip and separating it from the power supply line if a high current state occurs. SELs can occur in both SRAM and flash devices and this mainly relies on the hardware architecture of the device.

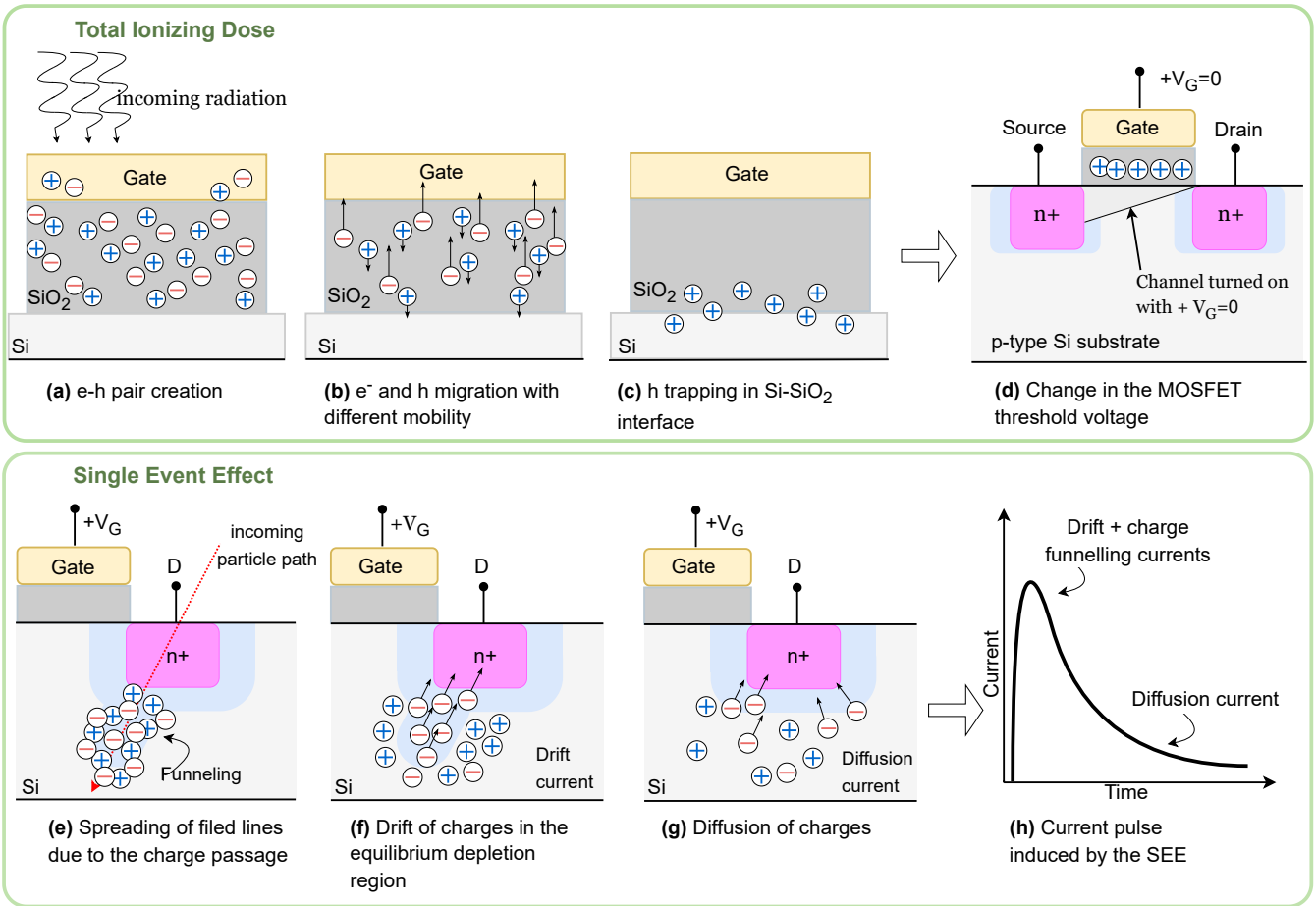

FIGURE 3. Effects of space radiation on electronic components. TID is a cumulative effect due to charge accumulation in the SiO2 layer or at the SiO2-Si interface. This is due to **(a)** the generation of e-h pairs in the oxide as a result of radiation-induced ionization; **(b)** the migration of low mobility holes to the SiO2-Si interface; and **(c)** the trapping of gaps by Si substrate defects or at the interface. **(d)** Charge accumulation leads to a progressive change in the features of MOSFETs, such as a decrease in threshold voltage in the case of an n-MOS. SEEs are due to the impact of single particles that create e-h pairs along their trajectory and generate **(e)** the spread of electric field lines, a phenomenon known as funneling, **(f)** the drift of charges in the equilibrium depletion region and **g** the charge diffusion, inducing **(h)** a transient current pulse that can lead to either a temporary or permanent failure. [31]

C. MITIGATING SPACE RADIATION EFFECTS

Managing the effects of radiation on electronic systems is a fundamental aspect of electronic design for space applications. This task can be extremely challenging and costly. There are several strategies to reduce the radiation effects. The most basic approach involves the use of appropriate packaging and shielding. However, this method increases the overall weight of on-board components and is often insufficient to properly protect against ionization phenomena caused by high-energy photons and particles [32]. Furthermore, shielding sometimes compounds radiation effects due to nuclear reactions induced within the packaging materials, which potentially lead to the creation of secondary particles. This process, in turn, induces errors and faults [32, 43]. Shielding, which is always employed in satellites, must therefore be rigorously combined with other mitigation techniques.

Two other approaches include using radiation-hardened (rad-hard) devices and/or implementing fault-tolerant techniques [45]. Rad-hard systems refer to those designed to be resistant to radiation-induced effects within defined limits limits (see Tab. 1). This resistance is achieved by using appropriate techniques during the manufacturing process [46]. Their development and implementation are often demanding and very expensive. Therefore, although they are widely used in large satellites, their integration into CubeSats remains limited due to their energy consumption, cost constraints, and mission lifetime. CubeSats, especially those intended for low orbits, often incorporate COTS components due to their affordability and quick availability in the market. However, COTS components are more sensitive to radiation than space-grade components [47, 48]. Therefore, their reliable use requires strategic integration with rad-hard components, mitigation techniques and extensive testing [23]. This section summarizes some of the most common key techniques for mitigating radiation-induced system failures within CubeSats. These include hardware redundancy, memory and firmware protection techniques, and protection circuit integration. Multiple of these techniques are often combined when developing an

TABLE 1. Radiation tolerance levels of COTS, rad-tolerant and rad-hard components. Data from [3].

Hardness level	COTS	Rad Tolerant	Rad Hardened
TID [krad]	2-10	20-50	$> 10^2$ to $> 10^3$
SEU Threshold LET [MeV · cm^2/mg]	< 5	20	> 60
SEU Error Rate [errors/bit-day]	10^{-4}	10^{-7}-10^{-8}	10^{-10}-10^{-12}

OBC. The choice of a particular technique depends on the specific OBC design and is influenced by factors such as on-board processing capacity, mission duration and budget, component specifications, and radiation environment.

1) Hardware Redundancy

This technique is very successful in mitigating cumulative effects [47] and SEUs [32]. Redundancy involves the introduction of a duplicate device or component within the system that performs the same functions. Redundancy can be static or dynamic. Dynamic redundancy, also known as *cold redundancy*, activates an unpowered backup component when the original one becomes unreliable or fails completely. Dynamic redundancy is a system-level fault-tolerant technique that can mitigate degradation due to cumulative effects, in that, when one component fails due to radiation accumulation, the unpowered component can take over its tasks. Static redundancy, known as *hot redundancy*, involves the simultaneous operation of redundant components and is used to handle SEU-induced errors [45, 49].

Two commonly used static redundancy techniques are dual modular redundancy (DMR) and triple modular redundancy (TMR; [41]). They are based on the use of two or three identical modules (one of which acts as a *voter* in the case of TMR) operating in synchrony, performing the same functions simultaneously and comparing their respective outputs. These architectures allow for the detection of a single error in a logical path when the outputs of the modules do not match, rebooting the system when a mismatch is detected [32]. DMR can be implemented at the processing unit level of an OBC, by including two independent chips on the board in a lockstep configuration ([50]); e.g., [51]), or by using multicore processors [52]. TMR can be used at the level of hardware logic (e.g., [53]) or with other components, such as memories (e.g., [54]). The TMR approach improves fault tolerance and reliability, but carries a higher risk of multiple failures over time, potentially reducing the system lifetime [45]. For CubeSat applications, this may be acceptable due to the often short mission duration.

2) Memory Protection Techniques

Memory systems, especially SRAMs, are highly sensitive to SEUs. Memories, in particular caches and register files, are also the main components affected by SEUs in processors [55]. The most commonly used memory protection techniques on OBCs involve employing memory hardware redundancies, utilizing information redundancy techniques such as EDAC strategies, and implementing periodic memory scrubbing.

EDAC codes are a particular class of error correction codes (ECCs) that use check bits to detect and recover errors in read or transmitted data. These codes can be implemented in hardware, software or a combination of both [56]. When implemented in hardware, the memory architecture is extended to accommodate additional check bits and is called EDAC-corrected memory or ECC memory. Software EDAC strategies are more cost-effective and are implemented in the fault detection isolation and recovery (FDIR) module, which is typically part of the FSW. Hamming codes are the most popular EDAC solutions [57–59]. They are linear block codes that allow for single-error correction and double-error detection (SECDED). In this approach, the number r of check bits, required to correct for 1-bit errors in a dataword of m bits, can be calculated according to $m + r + 1 \leq 2^r$. An example is the Hamming (12, 8) code, which converts an 8-bit dataword (m) into a 12-bit codeword ($m + r$), that is capable of correcting for a 1-bit error. More advanced codes, such as the Golay, Reed-Solomon (R-S) and Bose-Chaudhuri-Hocquenghem (BCH) codes, are able to detect and correct multiple errors [60].

To prevent two different particles from hitting the same word during ECC checks and to avoid the accumulation of errors that could result in MBUs, EDAC strategies are often combined with periodic memory scrubbing [61]. Memory scrubbing consists of periodically reading and comparing all data in memory, reloading them if an error is detected. By frequently checking the memory, the ECC can recover an incorrect value before further bit errors occur [32].

In recent years, a new trend that has rapidly emerged in space avionics is the adoption of novel memory technologies, which show higher tolerance to TID and SEEs than standard memories. Among these technologies, ferroelectric RAMs (FeRAMs, [62]) and magnetoresistive RAMs (MRAMs, [63]) are now widely adopted in OBCs, chosen for their ability to retain data in hostile environments, due to their operating principles. FeRAMs store data as electrical charges in polarized ferroelectric capacitors, while MRAMs store data as *magnetic* charges in magnetic tunnel junctions (MTJs). Their unique operating technology makes them inherently rad-hard, showing reduced vulnerability to SEEs and resistance to a TID of up to 1 Mrad [32]. Other technologies exhibiting similar radiation-tolerant properties, such as resistive RAM (RRAM, [64]) and potentially Phase-change Memories (PCMs, [32]) have also been tested in relevant environments and are poised for future opportunities. A comparison between the performances of these memory technologies and traditional memories is shown in Tab. 3.

3) Firmware Protection Techniques

Ensuring the reliability of the FSW is a crucial task for a successful space mission. This is particularly true when consider-

ing the growing complexity of the on-board software resulting from the use of CubeSats for increasingly complex mission purposes, the challenging real-time processing capability requirements, and the strong hardware-software dependence in avionics systems. In addition, just like data storage memories, program memory is also subject to SEU-induced failures, thus requiring the integration of fault-tolerance techniques that enable in-orbit firmware maintenance. Two key protection methods are firmware replication [65] and in-orbit firmware updating [66]. Firmware replication involves storing multiple copies of the firmware in different memory allocations or different memories, allowing the firmware to be protected against any corruption of its primary version [65]. A bootloader, a critical software component in the system, checks the integrity of the firmware image and, in case of failure, uploads the uncorrupted version into the program memory [23, 47]. However, it is worth noting that that the bootloader itself represents a single point of failure in a system with only one hardware component or OBC. Indeed, if the bootloader fails, the entire system could become inoperable, compromising mission objectives. To mitigate this risk, hardware-level redundancy, such as multiple physical components in a TMR configuration or multiple OBCs, can be implemented. The in-orbit firmware updating uses a transfer protocol and bootloader to transmit and upload a new version of the firmware image through the uplink channel, providing a facility to upgrade the firmware directly in orbit [66]. This tehnique allows for solving software issues and introducing new functions after satellite launch.

With the widespread adoption of multicore processors and systems even in CubeSat OBCs, other fault-tolerance software methods are emerging, such as firmware task rescheduling, task swapping, and checkpointing [67, 68]. In task rescheduling, a scheduler reassigns the tasks of the core affected by a fault to other processing cores in the system. Task swapping comprises methods that prevent the occurrence of a failure by monitoring the reliability of different cores for all executing applications and evaluating which specific task scheduling can reduce the soft error rate. Last, checkpointing periodically saves the state of a process during error-free execution and, in the event of a fault, restarts task execution to before the error occurred.

4) Protection circuits

These circuits include watchdog timers and overcurrent protection circuits, which provide protection against SEUs and SELs, respectively. The watchdog timer is a clock countdown register that can be internal or external to the processor. When the timer is expired, it then resets the processor (unless the timer is updated by the processor itself), returning it to a predetermined restart position. Such timers can be implemented in both hardware and software and are used to monitor the condition of a processor [69]. If the processor jumps to an incorrect memory location due to an SEU, the watchdog timer resets the processor to a previous state using a checkpoint, thus restoring operations.

Overcurrent and overvoltage protection circuits detect high-current or high-voltage levels and trigger a reset of the power supply, preserving the device from loss of functionality induced by SELs. These circuits are typically small hardware devices that are integrated into the power supply circuit of the OBC board. They include bypass diodes and resistors, which can be used to suppress current and voltage spikes, respectively [23], and latching current limiters (LCLs), which provide timed and controlled monitoring of overcurrent states and interrupt the supply line in case of power overload [70].

III. CUBESAT OBCS STATE-OF-THE-ART

The landscape of CubeSat OBCs mirrors the evolutionary trajectory observed in larger spacecraft C&DH subsystems. The current generation of processors exhibits robust capabilities to manage the computational demands of CubeSat missions, offering a spectrum of options ranging from cost-effective COTS solutions to customized proprietary platforms. The emergence of new applications for LEO satellites, along with the migration of COTS electrical, electronic, and electromechanical (EEE) devices from the automotive industry to the space sector - driven by a higher acceptance of risk by the space sector - impacted the growth of the market surrounding CubeSats, which offer low-cost, rapid development solutions for technology demonstration and new mission concepts. This growth has resulted in a proliferation of companies and research institutions developing their own OBC solutions.

The currently available avionics systems can be categorized into two main types: commercial OBCs and mission-specific OBCs [71]. This differentiation is crucial to gain insight into the varied landscape of OBC systems utilized in the realm of CubeSat technology.

Commercial OBCs are single- or multi-board computers that are tested and ready as a finished product on the global market. They are referred to as *commercial* because they are turnkey devices that can be easily integrated into a satellite. In our analysis, this category includes OBCs marketed by leading private CubeSat companies, such as Alén Space (Spain), EnduroSat (Bulgaria), GomSpace (Sweden), Innoflight (US), ISISpace (Netherlands) and Xiphos (Canada) among others. These OBCs represent versatile solutions for different mission objectives and are designed to serve a wide range of customers and applications. They generally have a wide variety of communication interfaces, making them easy to integrate on-board. This strategic choice not only optimizes the distribution of nonrecurring engineering costs across multiple missions, but also improves the efficiency of software development through the judicious reuse of resources - attributes of considerable allure in a fiercely competitive market landscape. Moreover, they are based on established EEE components, which increases their attractiveness for critical missions requiring proven reliability [23].

Mission-specific OBCs, on the other hand, are often developed in-house by universities, research centres, or companies for missions with specific requirements and objectives. In terms of components and manufacturing costs, they

are generally less expensive than commercial OBCs because they only include the modules needed to optimize resources. However, they require large investments in terms of research and development and have the disadvantage of being designed specifically for a particular mission and thus being less versatile. Among others, our analysis places the CubeSpace processor board family developed by NASA Goddard Space Flight Center (GSFC) in this category. Although these devices feature commercial aspects, such as the incorporation of COTS components and the goal of a flexible architecture adaptable to various missions, they are primarily tailored to meet the specific needs and requirements of NASA missions, thus falling into the mission-specific OBC category.

In this section, we present a review of commercial and mission-specific OBCs currently available on the market or under development. Interesting surveys on avionics intended for small satellites already exist in the literature [23, 24], laying the groundwork for similar work to our exploration. However, they focus on the general category of SmallSats. Here, our focus narrows to CubeSat-type satellites, emphasizing the critical role of tailored OBCs in meeting strict constraints of low mass, small form factor and low power [71].

A. ARCHITECTURES

In this analysis, we categorize OBCs based on their processing units, which may include traditional µCs, rad-hard µPs, FPGAs, and hybrid processors such as FPGA/GPU SoCs. The nature of the processing core of an OBC and its attributes, first of which are processing capabilities, speed and power consumption, are the main factors shaping the overall performance of an OBC. Therefore, they must to be carefully analyzed. We analyzed more than fifty CubeSat OBC projects, both commercial and mission-specific, designed by companies and research centers worldwide. Our bibliographic search was based on satellite martketplaces such as *satsearch* and *SatCatalog*; public databases such as [4]; and scientific databases such as *Scopus*, using "on-board data handling" and "on-board computer" as keywords and filtering specifically for "CubeSat" and "nanosatellite". Our targets were mainly research and commercial projects (scientific or technological demonstration missions) funded by government agencies and private companies. However, we also included secondary (academic/educational) projects in our analysis when we considered the results to be technologically interesting or useful from a historical perspective. Our focus was to provide a qualitative assessment of OBCs, identifying trends in their design choices, and emphasizing an understanding of their key attributes rather than quantitative metrics. A representative list of the OBCs considered in our investigation is shown in Tab. 5.

1) COTS µCs

COTS µCs have traditionally been the preferred option for early CubeSat avionics systems designed for LEO missions, especially those with shorter durations and lower processing requirements. The choice of COTS µCs is largely influenced by their numerous advantages, including low power consumption, minimal weight, proven reliability, easy availability in the commercial market, and user-friendly programming. These features make them perfectly suited to the limited SWaP-C resources typical of nanosatellite missions. These µCs are typically built on reliable processor architectures, providing the simplest and most cost-effective solution to perform basic OBC tasks, such as telemetry management and storage, command execution, and basic attitude determination and control system (ADCS) functions. Several COTS µCs have been shown to tolerate the radiation levels typically found in LEO environments [72], making them suitable for CubeSat applications. Among the options are several µCs from world-leading manufacturers, such as Texas Instruments (TI; MSP430), Microchip Technology (including some PIC, ARM-based or AVR µCs), ST Microelectronics (STM32).

The µC architecture most commonly used in CubeSat OBCs is based on the ARM Cortex family of processors [73]. These processors are widely used in CubeSat avionics for a multitude of compelling reasons. They have low power consumption, offer a balance between processing power and energy efficiency, and are readily available on the commercial market. In addition, the ARM Cortex family offers a wide range of devices with different performance levels and features, allowing space mission designers to choose the most suitable option for their specific mission requirements. Furthermore, radiation-hardened versions of some ARM Cortex processors are available, which could enable avionics systems based on them to be easily adapted to the challenges of the space environment beyond the LEO. Lastly, ARM-based processors benefit from a strong developer community that provides valuable development and debugging support, making them seamlessly integrated into nanosatellite OBCs. These advantages make ARM Cortex-based µC a common choice for a variety of CubeSat avionics applications, ranging from basic sensor interfacing to more complex data management and processing tasks. Several commercial and mission-specific OBCs are based on ARM Cortex-M and ARM Cortex-R µCs (see Tab. 5 and [74–76])

The choice between the ARM Cortex-R and Cortex-M for avionic system design depends on the mission-specific requirements and associated trade-offs. The Cortex-R series is specifically designed for safety-critical applications that require robust real-time support. These processors often feature redundancy schemes for fault tolerance, such as ECCs, lock-step execution, and other safety mechanisms. The Cortex-M series is designed for highly power-efficient applications that require modest deterministic real-time processing. In CubeSat OBCs, ARM Cortex-M µCs, have emerged as the preferred and most popular option [73], mainly due to their affordability, high availability, and robust development ecosystem.

Several factors influence the process of selecting a particular µC for avionics applications, such as computational performance, power consumption, memory requirements, pe-

ripheral needs, compatibility of the operating system (OS) and the availability of an integrated development environment [77]. This process can be challenging and may require extensive testing and qualification procedures [78]. Ultimately, the decision depends on mission requirements and priorities and is often a trade-off between computing and processing capabilities and energy efficiency. For missions with tight energy constraints and low power consumption as a priority, alternative µC choices include MSP430 by TI [79, 80], PIC µCs by Microchip Technology or and 8-bit and 16-bit AVR µCs by Atmel (now Microchip Technology). Such µC families are renowned for their energy efficiency and simplicity of implementation and are suitable for missions engaged in simple data handling and low-power operations, which operate with less demanding computing requirements. In particular, the MSP430 finds application not only in OBCs [51, 81, 82] but also in payloads [83, 84] and other critical subsystems designed for battery-powered and energy-constrained scenarios. However, these µCs have limitations in terms of processing capacity and peripheral functions compared to more robust µC families.

To conclude, it is important to note that while COTS µC are the simplest and most cost-effective option for implementing basic OBC functions, their processing limitations make them unsuitable for high computational cost operations or data-intensive applications. In addition, their deployment in missions characterized by elevated radiation levels decreases their reliability. Consequently, some missions opt for higher performance architectures, such as hybrid architectures, SoCs or GPUs.

2) Rad-hard processors

Due to advancements in satellite technology and payload miniaturization, interest in deep-space CubeSat missions has grown in recent years [8]. The design of the C&DH subsystem for these missions is extremely challenging due to the high risk of radiation exposure, which is particularly significant for the space environment beyond the LEO. To ensure the reliability of the electronics and prevent failures caused by high levels of ionizing radiation, the OBCs of these missions cannot rely exclusively on COTS components. It is essential to incorporate radiation-hardened (rad-hard) processors or devices into these missions, especially considering their scientific relevance and costs. One of the most famous and powerful rad-hard µPs, specifically designed for space applications, is the RAD750 µP [85, 86], developed by BAE Systems. This µP has the same architecture and operation as the commercial IBM PowerPC 750, but is specifically designed to be resistant to a TID of up to 200 krad (Si). It is widely used in large space missions (e.g., Mars Reconnaissance Orbiter, MRO; Fermi Gamma-Ray Space Telescope, FGST; James Webb Space Telescope; JWST), claiming a flight heritage of almost 20 years. It is also available in the standardized CompactPCI 3U or 6U formats [87], which is suitable for use in minisatellites [23, 24] or 3U-6U CubeSats. Although the RAD750 has been proposed as the OBC for some deep-space CubeSat projects [88], no deep-space CubeSat mission has been launched to date using this processor. In fact, although RAD750 offers the highest radiation tolerance, its cost is prohibitive for use in nanosatellites [24, 89]. Conversely, LEON processors (based on the SPARC V8 architecture) are widely used in OBCs for deep-space CubeSats. These processors were developed specifically for space applications by ESA and are available in different models (LEON2, LEON3, LEON3-FT, etc; [90]). The FERMI OBC developed by Argotech (Italy) [91] used onboard LiciaCube and ArgoMoon, is based on this type of processor. Although the widespread adoption of rad-hard processors in the CubeSat architecture is still in a nascent stage of development, with this lack of progress mainly attributed to the significant financial investment required for the development of new rad-hard devices, these processors are expected to be increasingly used in future ESA and NASA nanosatellite missions. This prediction is especially relevant given based on the growing interest in using CubeSats beyond LEO.

3) FPGAs

FPGAs have gained considerable attention in recent years for CubeSat missions, particularly due to their reprogrammable hardware nature, enabling fast and efficient parallel processing, or making them suitable for rad-hard architectures.

The use of FPGAs as central processing units in nanoatellites, both in OBC systems [53, 92] and payloads [93], has been explored for years. FPGAs differ from traditional CPUs or µCs in that they consist of logic gates that can be configured to execute any function and offer parallel processing capabilities. These features offer several advantages over conventional processing units. Specifically, FPGAs allow the design of customized, application-specific architectures that exploit parallel processing capabilities to perform multiple tasks simultaneously. For data-intensive applications that benefit from parallelism and hardware acceleration, such as image processing and analysis and computer vision applications, FPGAs have shown higher energy efficiency compared to conventional hardware accelerators such as multi-core CPUs and GPUs [94].

Additionally, FPGAs allow system inputs and outputs to be reconfigured as needed, enabling system interfacing with multiple memories, sensors, and peripherals [24]. Their parallel processing capability makes them particularly suitable for complex tasks such as image processing, data compression, and sensor data fusion. Numerous space applications can benefit from the use of FPGAs for real-time on-board data pre-processing, resulting in transmission bandwidth savings, such as synthetic aperture radar [95], hyperspectral imaging [96], and optical remote sensing data processing [97]. Additionally, the programmable and configurable nature of FPGAs makes them suitable for designing whole systems on a single chip, enabling conventional processors, and even multicore and multiprocessor systems [98], to be implemented using logic synthesis. These types of processors are called *soft-core* processors. Commercial OBCs by Sky Labs (Slovenia;[99])

and AAC Clyde Space [100] are examples of architectures that utilize soft-core processors implemented on FPGAs.

FPGAs encompass three primary process technologies: antifuse, FLASH, and SRAM based. Antifuse FPGAs are favored in early space applications [101, 102] as they boast resilience against SEEs and lower power consumption. However, they have the disadvantage of lacking reconfigurability because they are strictly one-time programmable. Flash FPGAs offer advantages similar to those of antifuse FPGAs, including non-volatility and reduced power consumption, but it is difficult to make them highly integrated and low-cost because of their manufacturing complexity. SRAM FPGAs benefit from frequent updates driven by µP advancements that make them the most attractive candidates for implementing complex satellite control algorithms [53], but have the disadvantage of being highly susceptible to SEUs [103, 104]. Rad-hard SRAM FPGAs are currently available and are utilized in some concepts for deep-space applications (such as the NG-MEDIUM device by NanoXplore; [105]; [106]). However, they have high costs that make them prohibitively expensive for smaller and cheaper CubeSat designs [93]. Actually, the radiation susceptibility of these devices can be mitigated by exploiting the reconfigurable hardware nature of FPGAs, which allows the implementation of specific fault-tolerant computing strategies, including internal logic redundancies [92, 104] and memory scrubbing techniques (such as the *Readback* and *Blind* scrubbing [107, 108]). These techniques make FPGAs suitable for designing rad-tolerant and rad-hard soft-core processors and systems. The ESA's LEON processors [90] are famous examples of rad-hard soft-core processors, in which radiation tolerance is achieved by incorporating redundancy and fault tolerance at the design level. The rad-tolerant architecture developed for the OBC of the RadSat mission also shows how, using a modern FPGA, an acceptable level of TID immunity can be achieved without using expensive rad-hard µPs [89].

Because CubeSats have stringent SWaP-C constraints, FPGAs can be a favorable choice for implementing OBCs and data processing systems, as they provide better performance (in terms of speed) while reducing components and allowing easy implementation of error mitigation techniques.

4) Hybrid architectures, SoCs and fault-tolerant hybrid systems

The emergence of hybrid architectures has the potential to revolutionize CubeSat on-board computing, greatly improving its performance in terms of processing speed and capability. Hybrid architectures refer to architectures that combine heterogeneous processing units, such as CPUs+FPGAs or CPUs+GPUs, in the same chip (SoC), module (System-on-a-Module, SoM) or board. The main advantage of using hybrid architectures in space applications is the possibility of assigning applications and algorithms to the portion of the device for which they are best suited to achieve the desired optimal performance [109], improving the speed and throughput of the system.

SoCs are the most popular examples of hybrid architectures. These devices integrate processor cores, memories, peripherals, and FPGA and/or GPU fabrics into a single chip. This integration realizes the advantages of combining different processing modules into one IC, resulting in reduced power consumption, size, and weight, while simultaneously enhancing processing capabilities. As a result, SoCs are ideal for CubeSat applications that require high performance, compact size, and efficient power consumption.

Most state-of-the-art CubeSat OBCs are based on hybrid architectures, particularly CPU + FPGA SoCs (see Tab. 5). Popular COTS SoCs include the AMD-Xilinx Zynq-7000 family (featuring a dual-core ARM Cortex-A9 processor and a 28-nm SRAM FPGA; [110]), MicroSemi SmartFusion 2 FPGAs (including a ARM Cortex-A9 core and a SEU-immune flash FPGA; [111]) and AMD-Xilinx Zynq UltraScale+ MPSoC devices [112], available both as CPU+FPGA SoCs (CG devices; featuring a dual-core ARM Cortex-A43, a dual-core ARM Cortex-R5F and a 16-nm finFET FPGA fabric) and CPU+FPGA+GPU SoCs (EG devices, which integrate also a Mali-400 GPU). COTS SoCs are predominantly used in CubeSat OBCs for LEO non-critical applications. Popular examples include the Xiphos Q-Card family (Q7s and Q8s including a Zynq-7000 and a Zynq UltraScale+ MPSoC EG respectively; [113], [114]), AAC Clyde Space Kyrten-M3 (SmartFusion 2 SoC; [115]), Space Inventor Z7000-P4 (Zynq-7000; [116]), and Innoflight CFC-400 (Zynq UltraScale CG; [117]).

Recently, a Zynq UltraScale+ MPSoC EG device was deployed as deep neural network (NN) hardware accelerator onboard the Leopard Data Processing Unit (DPU), developed by KP Labs for the Intution-1 hyperspectral mission [118]. This CubeSat, launched in late 2023, is the first in-orbit demonstration of a deep NN integrated on the edge in a COTS FPGA-based accelerator. This system can be used to accelerate machine learning (ML) and deep learning (DL) models for on-board image and data processing, such as real-time object detection and classification, and data compression, applications that have attracted significant interest in recent years, especially in the EO field [119]. Leopard DPU is capable of completing up to three trillion operations per second, demonstrating the high capabilities of FPGA-based SoCs to accelerate artificial intelligence (AI) inference in space.

Due to the increasing interest in integrating AI algorithms directly onboard satellites, particularly for multispectral and hyperspectral EO missions, many projects have suggested to use CPU+GPU SoCs in the CubeSat field [120–122]. GPUs are devices consisting of lightweight cores and on-chip memories that were originally designed to accelerate graphics applications, but are now used as hardware accelerators in general-purpose computing. For example, they are used to increase parallelism in software programs and to accelerate any kind of high-performance application, such as clustering of very large data sets [123]. Different CPU+GPU SoCs, such as the Nvidia Jetson X2i [124] and Xavier NX [125] series and

the AMD Embedded G-Series SoC family [126], have been proposed in many CubeSat payload processing systems. The educational nanosatellite Multiview Onboard Computational Imager (MOCI), developed by University of Georgia and planned to be launched by 2024, is one of the first CubeSat to include a Nvidia X2i in the design of its Accelerated Flight Computer (AFC; [127, 128]). Similarly, the Space Edge One (SE-1), a small form factor (0.25 U) on-board computing system developed by Spiral Blue (Australia; [129]), utilizes a Jetson Xavier NX and underwent space testing in 2023. The Unibap SpaceCloud iX5 (Sweden; [130]) is based on a AMD G-Series SoC and is deployed in the Hyperspectral Thermal Imager (HyTI) CubeSat [131], launched in March 2024.

In addition to the goal of integrating complex AI algorithms in space, there is a clear need for high-reliability hybrid solutions. COTS SoCs offer attractive trade-offs between SWaP-C, processing performance and flexibility, but are not rad-hard by design, thus requiring fault mitigation techniques. Recent developments integrate high-performance COTS SoCs with rad-hard or rad-tolerant components in *fault-tolerant* hybrid OBC architectures. In these systems, the COTS SoC acts as a high-performance node (HPN) for the most demanding computations, while the rad-tolerant component acts as a reliable computing node (RCN) and external supervisor. NASA GSFC has pioneered this type of hybrid approach through the SpaceCube family of processing systems, with SpaceCube V3.0 being the latest development [132]. SpaceCube V3.0 is a computing system (3U-220mm form-factor) that features two HPNs, a Zynq UltraScale+ CG device and a Xilinx-AMD Kintex UltraScale FPGA [133], while a rad-tolerant FPGA (Microchip Technology's RTAX FPGA; [134]) acts as an external supervisor responsible for monitoring, configuring, and scrubbing the HPNs. SpaceCube V3.0 is also available in a smaller version (1U form-factor), SpaceCube v3.0 mini [135], which features a Kintex UltraScale FPGA as a HPN and a rad-tolerant ProSIC FPGA (Microchip Technologies; [136]) as a high-reliability supervisor and watchdog. Such a fault-tolerant hybrid architecture is also featured in some of the systems mentioned above (such as the Xiphos Q-card family, Innoflight CFC-400 and Unibap SpaceCloud iX5) and in the latest high-performance payload processors such as multiMIND [137], the COTS-based Highly Integrated Computer System (CHICS; [138]) and the Scalable On-board Computing for Space Avionics (ScOSA; [139]). multiMIND was developed by Thales Alenia Space Germany as part of the EIVE E/W band demonstration mission [140], which successfully flew in 2023. It is based on a Zynq UltraScale+ MPSoC EG device, with a rad-hard Vorago Cortex-M0 µC [141] acting as a robust watchdog and anti-latch-up supervisor circuit. CHICS is currently being developed jointly by EVOLEO Technologies GmbH and Airbus Defence and Space Ottobrunn. It uses a Zynq UltraScale+ MPSoC as the central processing node, which is partitioned into isolated safe and non-safe areas according to the criticality of the application, while a rad-tolerant PolarFire FPGA (soft-core RISC-V; [142]) acts as the external supervisor. Finally, ScOSA, currently under development at the German Aerospace Center (DLR), is a modular SAVOIR-based architecture that also combines COTS-based HNPs with rad-tolerant RCNs, this time in variable numbers and configurations. The minimum configuration uses two Zynq 7020 SoCs as redundant HPNs, while an FPGA, with a rad-tolerant LEON3 soft-core processor, serves as RCN.

B. DESIGN CONSIDERATIONS

There are several key elements to consider when designing or selecting an OBC for CubeSat applications. These elements must comply with the strict constraints of low power, low mass and small form factor imposed by the standard. This section briefly discusses the factors that must be considered, such as processing core performance, power efficiency, supported types of memory and communication interfaces, and reliability in the space environment [23, 71, 74]. By incorporating these factors, the selection or design of the OBC can be customized to meet the specific requirements and challenges posed by different mission profiles.

Processing core performance. When designing or selecting an OBC, the performance of the processing core is an important parameter to consider. There are several figures of merit for assessing the performance of a processor (such as the clock rate, million instructions per second [MIPS] or million floating-point operations per second [MFLOPS]). However, for more complex CPU architectures, specific benchmarks should be used [143, 144]. The selection of the processing core should consider performance, power consumption and cost, mission data processing and transmission needs, and the role of the OBC in payload management. In missions with very limited power and where the OBC has simple tasks (limited to on-board health monitoring, TM storage, TC execution, and avionics management), µCs are the most effective choice. In contrast, in satellites where the OBC has to handle the payload as well, the payload determines the computing demands. Some payloads may require significant computing resources and the use of hardware accelerators (FPGAs and/or GPUs). The processing core of the OBC should also be chosen considering the amount of data to be processed and downloaded to Earth, as it has to handle the data, the bandwidth of the communication subsystem, and the storage memory (type and capacity).

Memory types. In addition to the processor, the performance of an OBC depends on the memory. There are four different types of memory in an OBC: boot memory, working memory, safeguard memory and mass storage. The boot memory is a nonvolatile read-only memory (ROM) that contains the FSW firmware and, in particular, the boot loader. This memory is usually implemented as EEPROM [74] or NOR flash memory [71]. These types of memory are inherently immune to SEE, so they are used to store critical FSW processes. The working memory is RAM memory, usually SRAM or dynamic RAM (DRAM), used for the real-

time execution of FSW, including the OS kernel and software applications for satellite control. The boot loader loads the OS kernel from the ROM to the RAM and initializes the OBC [74]. The safeguard memory retains its contents after the processor is reset or reconfigured and is used to store critical FSW parameters and satellite configuration data (e.g., the health status of sensors and actuators). The FSW continuously stores selective data in the safeguard memory, which can be accessed after a fault. The safeguard memory is typically a nonvolatile memory (EEPROM, FeRAM, MRAM; [71]). Finally, the mass memory is a nonvolatile memory used to store housekeeping and payload data during periods of no contact with ground stations. Storage memories should be chosen carefully based on several factors. They should be sized considering that the satellite can transmit data to the ground only when it is in a contact window, which often lasts for only a few minutes in the case of LEO satellites. Furthermore, to allow for efficient data transmission, their capacity must be optimized to fit the communication bandwidth, because a storage capacity that exceeds the transmission speed to ground causes resource misallocation. The type of memory should be chosen on the basis of its reliability in maintaining the integrity of the stored data and its resilience to radiation effects. Several options with varying performances are available, summarized in Tab. 3.

Power efficiency. Power efficiency is a key consideration in the design of the OBC and the choice of CPU and memories, and is directly influenced by the size of the satellite and the available power budget. Although the CubeSat standard itself does not specify power limits, and power availability can vary significantly depending on the size and configuration of the CubeSat, the power available for the C&DH subsystem is generally severely limited. As reported in [145], the total power available on a CubeSat is typically 1-2 W for a 1U CubeSat, 5-6 W for a 3U CubeSat, and for larger CubeSats (6U+) can be around 20 W with body-mounted solar cells, potentially reaching up to 100-120 W with deployable solar cells. As a result, the power consumption of the C&DH subsystem is typically limited to a few W [145] or a few tens of W. As shown in Tab. 5, depending on technology, OBCs for CubeSat used for simple platform control and to handle routine satellite tasks can consume up to 1-5 W, while more complex systems can consume up to tens of W.

Reliability. It is crucial to design the OBC to maximize its reliability in harsh space radiation environment. This involves selecting components with sufficient radiation resistance, considering factors such as the orbit and the TID to which they will be exposed. While short-term missions in LEOs typically encounter low radiation levels [35–38], extended-duration or interplanetary missions face much higher levels of exposure.

In the realm of µCs, notably, MSP430 and dsPIC33 demonstrated resilience in radiation environments of up to 20 krad (Si) and 15 krad (Si) respectively [72]. Extensive research and characterization of the effects of radiation were also conducted on ARM Cortex-M architectures, which demonstrated their ability to maintain functionality when exposed to up to 50 krad (Si) [146, 147].

FPGAs generally show good TID performance, with resilience levels varying depending on the technology. Modern SRAM FPGAs can function properly in the presence of high ionizing radiation exposure (e.g., Xilinx FPGAs functioning properly up to $\geq$ 100 krad (Si), with values depending on the family [103]), but their reliable use in space requires specific techniques to mitigate SEU-induced errors [148]. Flash FPGAs generally have a lower acceptable TID due to the radiation-induced charge build-up in the floating gate (e.g., ProASIC 3 FPGA family, which can withstand 20-30 krad of TID [148]), although there are flash FPGAs with rad-hard TID performances (e.g, RTG4 FPGA [149], with TID tolerance up to 100 krad).

Considering the widespread use of COTS components in CubeSats and the growing interest in long-duration and/or beyond LEO missions, which entail significantly higher radiation exposure, it is crucial to conduct a thorough analysis of the radiation susceptibility of the chosen components and evaluate potential mitigation strategies (§II-C) from the early stages of system development.

Communication Interfaces. Ensuring proper data handling and effective communication between the OBC and other satellite subsystems is critical to mission success. Selecting the appropriate data bus is a critical aspect of satellite and OBC architecture design. This choice can be challenging as communication interfaces can potentially represent a bottleneck in the data processing and delivery chain. Therefore, a careful selection of components based on a trade-off analysis between several high-level requirements, such as the data rate, power consumption, availability, simplicity and reliability in the space radiation environment, is paramount [150]. Tab. 4 provides an overview of the data buses most commonly used in CubeSats, highlighting their performance and applications in some CubeSats. This section briefly covers these data buses, their main distinctive features and constraints, and general recommendations for the data bus selection.

CubeSats typically use several types of buses, the most common of which include linear and point-to-point configurations, with some recent projects proposing wireless data buses (e.g., [151]), which have not yet reached technological maturity. Linear buses are used to connect multiple bus nodes together using the same set of wires or lanes and are widely used in CubeSats because of their simplicity, ease of implementation, and the limited amount of wiring they require [152], although they have limited performance (e.g., data rate) compared to point-to-point topology. The latter can connect only two nodes together, offering higher performance and speed. Linear buses from the military and aeronautical industry, such as MIL-1553, are widely used in large missions, but are very rarely employed in CubeSats due to their high complexity and cost [145]. Instead, the data buses commonly used in CubeSats come from the commercial and

automotive industry, exploiting the widely available components and design possibilities, and include the Inter-Integrated Circuit (I2C), Controller Area Network (CAN) and RS-485, which are predominant in the majority of current systems [145, 152, 153].

The I2C bus is the most adopted linear bus in CubeSats [153], due to its simplicity, low power consumption (50-150 mW depending on the number of the nodes) and the high availability of commercial ICs with I2C interfaces. It operates with two wires to connect multiple nodes in a master-slave configuration, with data rates of 100 kbps up to 400 kbps and effective data throughputs in practical implementations in CubeSats of approximately 60% of the data rate. It is designed for short distances, limited to several tens of centimeters in practical cases [145, 152, 153]. Despite its popularity, the I2C bus caused a lot of in-orbit issues in numerous CubeSats projects, especially bus lockups [154], representing a potential single point of failure in CubeSat missions [155–157]. Being a non-differential bus, I2C is it less robust compared to differential buses like CAN and RS-485 and is more susceptible to bit-errors [154], hence requires a careful error handling, such as supplementary watchdog timers and bus lockup protection circuits, and buffers to protect remote chips to parasitic powering [153, 157]. For all these reasons, and for the high number of CubeSat in-orbit failures caused by the I2C bus, several developers recommend the CubeSat community to avoid I2C buses and to prefer more reliable busses, such as CAN and RS-485 [156, 157] However, when the choice of I2C cannot be avoided (e.g. because some on-board devices only provide I2C interfaces), it is strongly recommended to use the I2C buses only to connect sensors non-critical for the operation of the satellite and always accompanied by fail safe mechanisms [154, 157]. A more reliable linear bus, which is becoming a viable alternative in CubeSats, is the CAN bus [158], designed to operate reliably in harsh environments and offering an embedded error handling. It supports higher data rates, up to 1 Mbps or 5 Mbps for enhanced versions like CAN FD, and is particularly valued for its ability to handle multiple nodes with minimal wiring, with slightly higher power consumption than I2C (150-300 mW, depending on the number of the nodes; [152, 153]. However, the CAN bus has a high protocol overhead, which reduces, in a practical implementation in CubeSats, the effective data throughput to about 15% of the data rate [145, 152]. For higher throughput and higher performance systems, the choice of linear bus can fall on the RS-485 [150]. It is a differential asynchronous serial bus that generally uses a Universal Asynchronous Receiver-Transmitter (UART) at the data layer, and requires an external differential driver. In terms of robustness, due to its differential signal nature, it is more robust than I2C but it has not embedded error handling, making it less robust than CAN. It supports higher data rates, similar to CAN bus (around 1 Mbps), but with a smaller protocol overhead, resulting in enhanced effective data throughput in CubeSats, approximately 60% of the data rate, with a lower power consumption (10-100 mW; [145, 152]). One drawback is that, unlike CAN and I2C, this bus is specified only at the physical level, and the development of its data protocol is left to the developer. For use in CubeSats, the data protocol therefore requires to be entirely specified and standardized [145]. In terms of trade-off between reliability, power consumption and effective data throughput, CAN and RS485 are equally recommended in CubeSat applications. The exact choice will be dependent on the exact mission and will most likely be between two factors: if reliability is of primary concern, then CAN is the most likely choice; if the low power consumption and high data throughput are of high importance, then RS485 is the better choice [150].

For subsystems and components requiring dedicated, high-speed connections to the OBC, point-to-point links are used. Popular choice in CubeSats include Serial Peripheral Interface (SPI), Universal Serial Bus (USB), Ethernet and RS-422 [145, 153] and more advanced options like SpaceWire, the only data bus specifically designed for space applications and whose implementation in CubeSat OBCs is growing in the last years [100, 159]. Among these, SPI is the most popular option until now, since is supported by the vast majority of microcontrollers and, hence, it does not require any external ICs to operate. However, it has several drawbacks, such as the large amount of chip select lines that must be connected to the pins of the OBC (thus, if the amount of lines exceeds the maximum amount supported by the OBC, it requires the use of multiplexer chips to reduce the amount of chip select pins). Furthermore, like I2C, it has low reliability, as it has no built-in safety mechanisms, and requires buffer chips to prevent parasitic powering [154, 157]. For these reasons, it has been reported that, similar to I2C, SPI should be avoided as much as possible in future space applications and replaced by buses with higher reliability [157]. The USB serial bus and the SpaceWire network can be considered a viable alternative to SPI, or be used as high performance buses to implement high-speed transfer payload data [150, 159], as they provides different methods for fault detection and a high data throughput.

IV. FLIGHT SOFTWARE STATE-OF-THE-ART

One of the crucial aspects of the successful operation of a space mission is the FW. The FSW serves as the backbone of an OBC, and is responsible for managing various critical functions and ensuring program execution. It is responsible for communicating instructions to the satellite to enable the execution of operations required for mission success. The FSW is usually considered as the set of software running on the C&DH subsystem, but it also includes all software running on the electronic modules of all subsystems and payloads available on the satellite. [160].

Considering the complexity of the C&DH subsystem and the harsh radiation environment in which a satellite is normally immersed, it is necessary to equip the on-board avionics with an FSW that meets characteristics such as *extensibility*, *scalability*, *modularity*, *reusability*, and *reliability* [161]. Existensibility signifies that changes to software and programs

must not affect the functionality of the base system, which must always ensure its operability. Extensibility is directly related to scalability, that is, the ability to customize the FSW for different goals, making the system suitable for different mission purposes. Modularity signifies that functionality is well defined in specific modules and implies flexibility in integrating or removing functions, such as the ability to handle multiple payloads. Finally, reusability and reliability reflect the portability of software across multiple hardware platforms and the ability to handle any anomalies through fault-tolerance techniques, respectively.

As technology continues to evolve, current nanosatellite FSWs have undergone significant developments, enabling improved performance, reliability, and adaptability. When developing an FSW for a space mission, two major design decisions must be made: the open-source framework, which will serve as the basis for building the software applications, and the operating system (OS), which manages hardware resources. In the following subsections, we discuss the features that define the state-of-the-art of FSWs developed for nanosatellites, presenting the general architecture of an FSW and focusing primarily on the frameworks and OSs used.

A. FSW FRAMEWORKS

As the complexity of nanosatellite missions continues to grow, so does the demand for advanced software to manage the intricacies of an OBC. Therefore, if the right tools are not provided, FSW design can become very challenging, thereby causing very long development times, high costs, and potentially unreliable end results. Therefore, to develop an FSW in a proper way, a *framework* is needed. A framework is a supporting logical architecture on which software can be designed and implemented, and provides a series of code block snippets that can be reused to facilitate FSW development and reduce development time [23].

There is a set of selection criteria to consider when choosing the optimal open-source framework for FSW development, depending on the OBC hardware [160]. First, the availability of the framework source code and respective documentation should be considered. The source code should be available in an open internet repository, with a comprehensive user manual and several examples and applications showing how to use it (if possible, a software development kit should also be available). In addition, the chosen framework should have a small footprint, i.e., occupy the minimum possible space in memory and CPU load, and a certified flight heritage, i.e. have proven its success in previous missions. Finally, the framework should be able to evolve and improve based on missions. Therefore, it should have optimal quality attributes (reliability, well-defined semantics, modularity and portability in different OSs and processors), a well-established development community providing long-term support, and the possibility of standardization based on the Consultative Committee for Space Data Systems (CCSDS) recommendations and protocols.

A further important consideration when it comes to open-source frameworks is their security and cyber resiliency attributes. While these frameworks come with many advantages, such as reduced development costs and community support, their open-source nature can lead to security issues, such as code tampering, dependency risks, and exposure to vulnerabilities [162]. Because source code is publically accessible, anyone can review, modify, and potentially hack it, compromizing the authenticity and integrity of the software. This can introduce faults, bugs, and vulnerabilities into the software that could result in the corruption or modification of data and commands, as well as the execution of incorrect on-board actions, posing a risk to the integrity and functionality of satellite systems and potentially causing catastrophic mission loss [163]. Open-source software often relies on numerous dependencies that, if not analyzed for security, can introduce additional vulnerabilities. Open-source software can be examined by anyone for security vulnerabilities, which can become a prime target for malicious attacks aimed at gaining control of satellite operations [164]. This poses a challenge for cybersecurity of space missions, particularly for high-engagement scientific and commercial missions funded by government agencies or private companies. The Open Source Security Foundation (OpenSSF) identifies some basic criteria for evaluating the security of an open-source framework through the use of the "Security Scorecards" tool [165]. These security best practices include: conducting peer reviews for each request to merge new contributions to detect various unintentional problems, including vulnerabilities; completing continuous integration testing and code analysis before merging new contributions to reduce the number of vulnerabilities; using automatic dependency update tools to identify obsolete or insecure requirements and apply updates; and avoiding dangerous workflow patterns that could allow malicious authors to gain write access to the repository. In addition, the framework has to be actively maintained, with regular patches and bug fixes, and actively tested [162]. Although these criteria alone can not guarantee system security, they constitute a set of quality assurance best practices that framework developers and administrators should adhere to during the development and maintenance of open-source software [164]. In addition, cyber resilient space missions should implement appropriate protection mechanisms to prevent unauthorized access along the TC/TM data channel, which is the main target of attacks. Comprehensive protection of satellite communication links should include authentication, integrity checks and encryption. The uplink should be checked for integrity and authentication to ensure that the satellite remains protected from the control of unauthorized persons [166–168].

FSW architecture. Typically, the FSW has a layered structure, as shown in Fig. 4. The lowest layer is the *hardware*, which represents the physical components of the computing system, including the processor, memory, storage devices, I/O devices, etc. This hardware layer executes instructions provided by the software layers above it. The available hardware

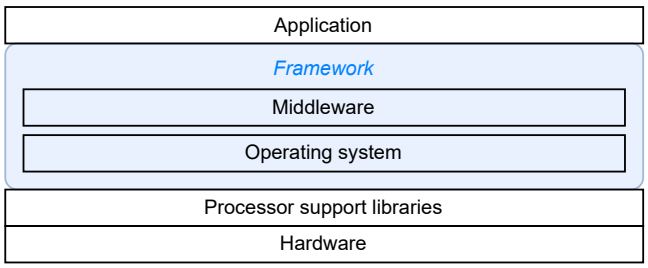

FIGURE 4. FSW general layered model.

is the first aspect to consider when designing an FSW, as it imposes limitations on the software itself. Computing power, memory capacity, peripherals and communication protocols depend on the hardware. Specific software applications also depend on the hardware. For example, the high processing power of COTS processors and their consequent widespread use have resulted in an increase in functional software requirements and the development of programs responsible for FDIR [48]. In addition, the architecture of the FSW is inherently tied to the available hardware and the possibility of choosing a centralized or distributed system (§II).

The *processor support libraries* layer includes device drivers, libraries, and low-level software that interact directly with the processor and other hardware components. It provides support for tasks such as handling interrupts, managing memory at a low level, and interacting with hardware peripherals. The *Operating System* is a crucial layer that manages hardware resources and provides a set of services to applications. It abstracts away the hardware complexities, allowing applications to run on different hardware platforms. The OS handles tasks such as process and memory management, and device drivers. The *middleware* layer acts as a bridge between the application layer and the OS. It facilitates communication and data exchange between applications and the OS. In some computer systems, middleware is part of the OS itself. Middleware and OS are often formalized in the *framework*, which comprises software tools and libraries that provide a base for building applications. This is an abstraction layer that avoids low-level details and provides a set of tools and conventions for developers to work with. A framework may have its own optimized OS distribution, or provide application program interfaces (APIs) for abstract OSs that allow compatibility with multiple OSs. The last layer is the *application* layer, that is the closest layer to the end-user. It includes the software applications with which users interact directly. The application layer interacts with the lower layers to provide services and functionality.

1) Core Flight System

The core Flight System (cFS) [169, 170] is a standardized open-source FSW framework (complying with the CCSDS standard) and a set of reusable software applications developed by the NASA GSFC. It is specifically designed to provide an FSW development environment that is reliable, robust, scalable, readily maintainable, and testable. cFS provides a runtime development environment independent of both the OS and hardware, and a set of basic and reusable software applications, typically part of an FSW. Three are the main features of the cFS framework: a dynamic run-time environment, a layered software architecture, and a component-based design.

The 3-layer architecture of the cFS is shown in Fig. 5. It consists, from bottom to top, of [169]:

- **Abstraction Library Layer**. This is the lowest-level layer and contains a set of software libraries that enable the FSW to interface with a real-time OS (via the *OS Abstraction Layer*, OSAL), and with hardware resources (via the *Platform Support Package* - PSP - layer). It provides APIs for abstract OSs and hardware systems, allowing the framework to adapt to multiple OSs (RTEMS, VxVorks, FreeRTOS) and different processors (such as MCP750, RAD750, SPARC LEON3).
- **cFE Core Layer**. This represents the core of the FW and contains a set of mission-independent core services that can be reused and configured according to mission needs. These services enable the management of the cFE and cFS applications (*Executive Services*; ES), the exchange of messages between applications (*Software Bus*; SB) and their modification (*Table Services*; TBL). They also provide most of the basic functionality for the operation of a spacecraft, such as sending error messages to the ground (*Event Services*; EVS) and onboard time synchronization (*Time Services*; TIME).
- **Mission and cFS Application Layer**. It consists of reusable software modules that provide standard spacecraft functionalities shared by all missions, such as data or command storage, memory and housekeeping data management, mission scheduling, etc. Currently, 13 open-source cFS applications have been tested and made available to the user. This layer also provides mission-specific modules containing the functionality of particular space missions. In fact, this layer also serves as an environment for developing new mission-specific FSW applications (in C and C++).

cFS has an important flight heritage and has been implemented in several large spacecraft, including the Lunar Atmosphere and Dust Environment Explorer (LADEE), and the Magnetospheric MultiScale (MMS) and Global Precipitation Measurement (GPM) missions [169]. It was used in the Dellinger CubeSat that flew in 2017 [171, 172].

2) KubOS

KubOS [173] is an open-source Linux-based FSW framework developed by the software start-up KubOS Corporation (US). The company provides a software developed kit (SDK) that allows users to create their own KubOS project [174]. The KubOS platform contains a number of microservices that provide the basic critical functionalities required by an FSW

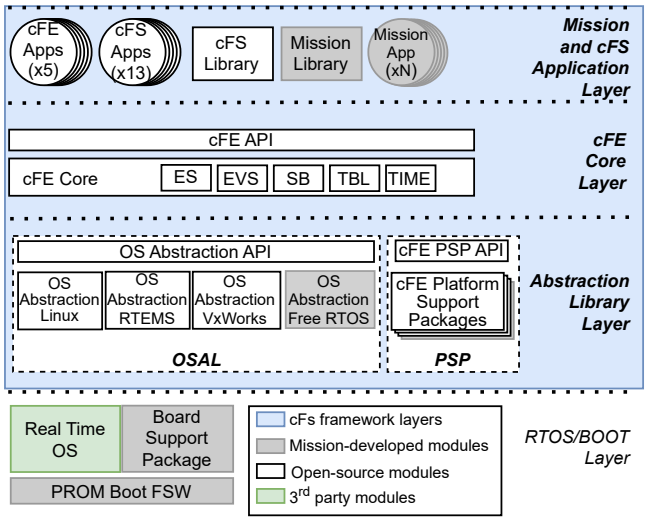

FIGURE 5. Three-layered architecture and software components of the NASA cFS framework [169].

and provide a reliable development environment for mission-specific FSW applications. KubOS features a 3-layer architecture, as shown in Fig. 6, which are (from the bottom to up):

- **KuBOS Linux**. This layer provides the basic OS, software abstractions for communication busses (e.g., SPI, I2C), and drivers needed to communicate with connected hardware devices. It provides its own customized Linux distribution, which offers higher abstraction and development capabilities for software application design. The OS is coupled with a proprietary bootloader (*U-boot*) responsible for loading, updating and recovering it in case of failures, to mitigate the risks inherent in the use of Linux in space applications.
- **KuBOS Services**. This layer provides a collection of persistent micro-services that allow interaction with the satellite hardware and perform basic functionality. These include: Reusable inter-mission *Hardware Services*, which expose the functionality of each hardware device connected to the OBC (ADCS, GPS, radio, etc) to the rest of the bus; *Core Services*,for monitoring the OBC, telemetry management, task scheduling, and application management; and *Payload Services*, which are custom-designed for particular payload hardware and are not intended to be reused between missions. The microservice-based architecture, in which each critical component is a separate process, allows specific changes in one service to be made without affecting the others, making improvements and upgrades easier.
- **Mission applications**. This layer contains a series of user-level programs that can be executed when needed and govern the high-level behavior of the satellite (e.g. management of states, execution of defined tasks, monitoring of on-board behavior or payload operations). These applications are written by users using several supported programming languages (C, Python, Rust and Lua).

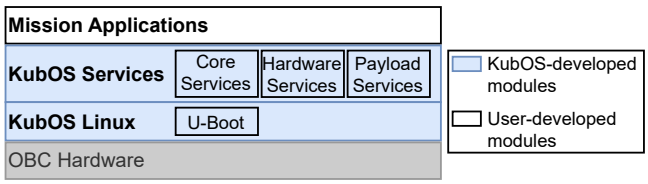

FIGURE 6. Three-layer structure, services and modules of the KubOS framework [173].

KubOS currently supports 3 CubeSat OBC boards (ISIS-OBC, Pumpkin Motherboard Module 2, Beaglebone Black Rev. C), but has no flight heritage yet. In addition, this framework has not yet been standardized as a reference architecture.

3) NanoSat MO

The NanoSat MO Framework ([175]) is a standardized open-source framework designed by the ESA specifically for nanosatellite missions, with the aim of simplifying the development of software applications, their control and management, and their deployment in satellite platforms. It is a spin-off of the standardized CCSDS Mission Operations (MO) Services Architecture (CCSDS 520.0-G-3 standard, [176]).

Fig. 7 shows the architecture of the NanoSat MO framework. The main feature of its software architecture is the independence between the application layer (hosting software applications) and the underlying middleware and hardware platform. To create this independence, the framework structure comprises two main sets of services: the MO *Monitor and Control Services* (M&C Services) and new *Platform Services* [177, 178]. The M&C Services are standardized MO services that are already defined and provided by the CCSDS MO Services Architecture. They enable monitoring and control of applications (through e.g., parameter state provisioning, command invocation, and alert notification; [176]), and allow applications to be managed from the ground or other applications. Platform services are developed specifically for the NanoSat MO Framework and contain sub-services that allow management of the satellite hardware peripherals, enabling applications to interface with the platform (such as: ADCS service, Optical Data Receiver Service, etc. [178]).

The unique architecture of the NanoSat MO framework makes it completely independent of the underlying hardware technology (and even the OS). It also provides the ability to develop software applications on the ground, which can then be installed and deployed in the satellite, and started and stopped from the ground at any time. NASA provides an SDK that facilitates the creation of applications based on the Java language [179].

NanoSat MO has a proven flight heritage in nanosatellites. It was developed as part of ESA OPS-SAT CubeSat, an experimental mission launched in 2019 with the purpose of testing and demonstrating new software and mission operation con-

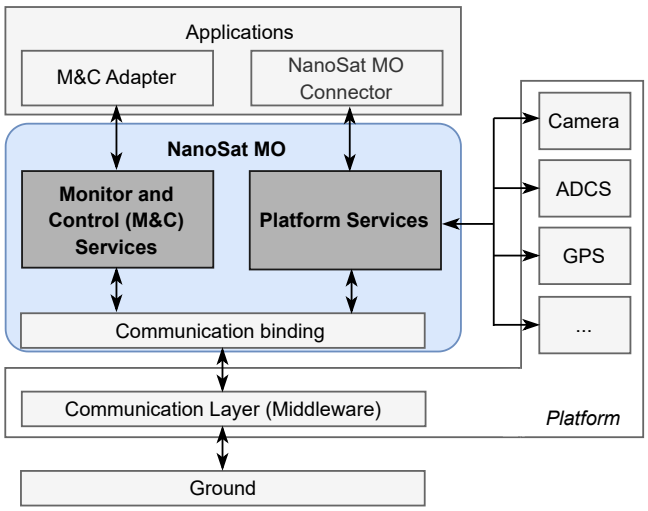

FIGURE 7. NanoSat MO high-level architecture [178].

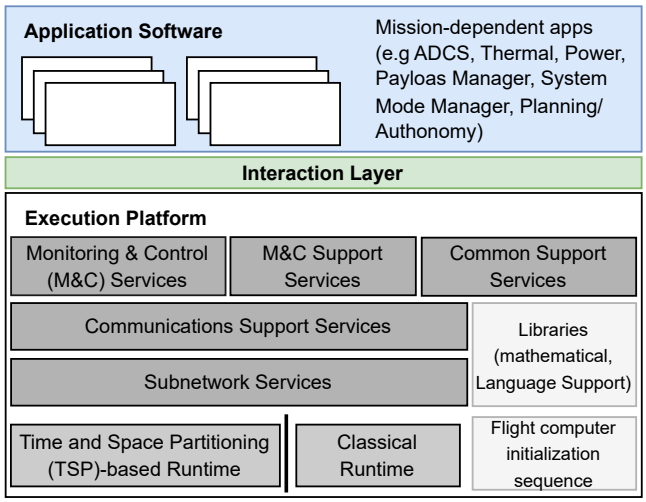

FIGURE 8. SAVOIR - CORDET FW reference architecture [26].

cepts [177]. NanoSat MO will also be implemented in ESA upcoming Φ-Sat-2 CubeSat, facilitating the development of AI applications used directly onboard the satellite for EO purposes [180].

4) SAVOIR - CORDET

The Component ORiented DEvelopment Techniques (CORDET) framework is a FW reference architecture developed as part of ESA's SAVOIR initiative ([26], §II-A) by the SAVOIR-FAIRE working group.

CORDET reference architecture is based on the segregation of two main components, the *Application Software* and *Execution Platform* (Fig. 8). The Application Software layer uses mission-dependent software components to manage the functionality of the satellite bus and its subsystems (e.g., FDIR management, autonomous task planning, and subsystem management). These components are developed independently of the execution environment. The Execution Platform layer then provides services for the execution of these software components and deploys the components from the higher abstraction layer to the chosen physical architecture. The mapping of components from the Application Software to the Execution Platform is accomplished using a set of design rules materialized in an interaction layer [181]. This bimodal architecture clearly separates the management of satellite functionality from computing issues (such as interapplication communication, on-board time, and libraries), allowing users flexible application development.

CORDET services comply with the European Cooperation for Space Standardization (ECSS) Packet Utilization Standard. There is an existing plan to conform such an architecture to the CCSDS standard [182]. Unlike the other frameworks analyzed in this paper, CORDET does not represent a particular FSW implementation and does not provide specific API rules [160], but simply outlines a generic architecture for service-oriented applications. However, there is a formal specification of CORDET implemented in the C language, developed by a company that is a member of the consortium founded by the ESA to specify CORDET (P&P Software GmbH). This specification is called the C2 Implementation and it is open source [183, 184].

The CORDET C2 implementation has a flight heritage on large satellites, successfully flying onboard the CHaracterising ExOPlanets Satellite (CHEOPS; launched in 2019, in progress). It will also fly on-board the Solar wind Magnetosphere Ionosphere Link Explorer (SMILE; 2025) and the Atmospheric Remote-sensing Infrared Exoplanet Large-survey (ARIEL; 2029). It has no flight heritage in CubeSats, although it is also considered an FW framework for nanosatellite applications [160].

B. FSW OPERATING SYSTEMS

The OS plays a crucial role in the FSW structure, as it manages hardware resources and provides essential services for the development of software applications. Selecting the appropriate OS can considerably simplify the development of FSW and its applications. As discussed in §IV-A, the FSW frameworks used in the CubeSat landscape offer their own optimized version for the OS or are compatible with different OSs. In recent years, the integration of real-time operating systems (RTOSs) has become an important advancement in the field of small satellite missions. An RTOS offers precise processing time management and efficient resource utilization. There is a wide range of RTOSs, ranging from open-source solutions, such as FreeRTOS, to licensed platforms, such as VxWorks.

All RTOS have a central element, the kernel, which serves as an abstraction layer connecting applications with the hardware device and handles services. These services include APIs that can be used to simplify the management of I/O devices, memories, timers and interrupts.

In the following sections, we provide an overview of the various OSs suitable for CubeSat applications.

1) FreeRTOS

FreeRTOS [185] is an open-source real-time OS specifically developed to run real-time applications on µCs and small µPs, supporting a wide range of processor architectures, such as ARM and RISC-based architectures. Support for a wide variety of hardware architectures is one of the strengths of this OS. The FreeRTOS kernel consists of only three files in the C language. FreeRTOS has a so-called *microkernel* architecture [186], which signifies that the kernel provides only a basic set of OS functionalities, such as the scheduling of processes, communication between them, and their synchronization. All other functionalities of the OS, including device drivers and system libraries, operate on programs separate from the kernel. The small footprint of the kernel is another strength of this OS, as it makes it flexible, scalable, and easily executable in various small embedded applications. FreeRTOS also features a small memory footprint, which makes it suitable for applications where the memory capacity is limited. All these features make it suitable for real-time applications that require critical tasks, such as sensor monitoring, while maintaining low power consumption. For this reason, numerous CubeSat OBCs are based on FreeRTOS (see Tab. 5). This OS, which offers the simplest implementation option, has been proposed in several projects for the development of FSWs for resource-constrained CubeSat missions [76, 187–189].

2) RTEMS

The Real-Time Executive for Multiprocessor Systems (RTEMS, [190]) is a widely used open-source RTOS with no kernel-space/user-space separation (*monolithic* architecture), designed to provide deterministic performance through low lantency of management of scheduling, threads and external interrupts [191, 192]. Its key feature is its high level of portability, as this OS currently supports 19 processor architectures (including all microprocessors developed for use in space, such as SPARC ERC32 and LEON, PowerPC, MIPS Mongoose-V), approximately 200 board support packages (BSPs) and various open API standards, including the Portable Operating System Interface (POSIX). To limit application complexity and reduce the size of the kernel, the RTEMS provides core services dedicated exclusively to task management, interrupts, synchronization and interthread communication. Other services, such as support for I/O and memory file systems, networks and programming languages, are provided as optional features [192]. Thus, RTEMS can be considered a simplified OS with optional attributes of memory and process management, resulting in an application-dependent size of the RTEMS kernel. RTEMS has been sponsored, deployed and used extensively in space missions, including NASA's MRO and ESA's ExoMars Trace Gas Orbiter. This OS is particularly well suited to the constrained resources of nanosatellites and runs in numerous CubeSat OBCs (Tab. 5).

3) RODOS

The Realtime Object-Oriented Distributed Operating System (RODOS, [193]) is an open-source RTOS specifically designed for the aerospace field, as it has a small footprint, which allows it to be suitable for all high-reliability applications. This OS was jointly developed by the DLR and the Department for Aerospace Information Technology at the University of Würzburg as part of the DLR's microsatellite program. RODOS is written mainly in the C++ programming language and supports a wide range of processor architectures, such as ARM Cortex-M, Atmel AVR, 32-bit STM3 and PowerPC architectures. It can be used as a stand-alone OS or as a guest OS on top of Linux and some RTOSs. The main feature of RODOS is that, in addition to the kernel (which provides support for the basic functionalities of the OS), it also integrates its own real-time middleware, which allows transparent data exchange between software applications. The RODOS kernel and middleware provide an integrated object-oriented framework that allows for optimized management of software resources and a communication infrastructure between applications, guaranteeing a high degree of modularity for their development. Software applications or modules can be integrated by the user as easily accessible and modifiable independent building blocks, providing flexibility in the development of FSW [194].

RODOS has flight heritage on board the two microsatellites forming the DLR's FireBird constellation, the Technology Experiment Carrier-1 (TET-1) and Bi-spectral Infrared Optical System (BIROS) satellites [195], which were decommissioned in 2019 and 2020, respectively. In particular, BIROS integrated two different software experiments in its RODOS-based payload processing unit, namely the Verification of IMage analysis Onboard a Spacecraft (VIMOS; optical image on-board processing software, [196]), and the Verification of Autonomous Mission planning Onboard a Spacecraft (VAMOS; on-board task scheduling software, [195]). RODOS was also used in the SOlutus NAno satteliTE (SONATE), a Cubesat mission developed by the University of Würzburg, to manage the software of the Autonomous Sensor And Planning (ASAP) instrument and to enable communication with the main OBC.

4) VxWorks

VxWorks [197] is a proprietary RTOS developed by the Wind River System company (US). Its two main attributes are its low latency (measurement of the time interval between the generation of an interrupt and the subsequent external response generated by the interrupt handler) and minimal jitter (random variation of latency values), which make it a fast, stable and reliable OS, suitable for highly complex, high-performance applications requiring critical reliability [198]. Due to its compatibility with a wide catalog of processors (32- and 64-bit processors based on Intel, Atmel, Power, RISC-V; LEON and other architectures) and the various programming languages that can be used for its software application development (C++, Python, Rust), it is used in various sec-

tors, including aerospace, defense, automotive, medical, and telecommunications. VxWorks has a layered architecture, in which the kernel is separated from the middleware, BSPs, applications, and other packages, and supports application development via docker containers. This architecture allows applications to be isolated from the rest of the system and provides the OS with modularity and easy upgradability, enabling easier bug fixing and testing of new functionalities. VxWorks is one of the most widely used OSs in space missions, particularly in NASA programs. It has been used by NASA for more than 30 years on major scientific missions, such as the Clementine spacecraft, the Juno space probe, and the JWST. It is also used in new-generation CubeSat applications. The FSW of the PEARL CubeSat bus infrastructure, an initiative developed by the Space Dynamics Laboratory (SDL) with the aim of migrating the CubeSat standard from the use of COTS components to the use of space-grade components, is based on the VxWorks OS [199].

5) Linux

Linux is a family of open-source general-purpose OSs. Since the Linux kernel source file is open-source, there are several Linux distributions, both commercial and purpose-built by embedded developers, with varying system and memory space requirements. This feature makes Linux a highly flexible OS suitable for many embedded applications and widely used in space systems (the more famous examples of space vehicles using Linux as their OS are SpaceX's Falcon and Dragon). Linux can simplify software application development due to its advantages, such as the support of a wide range of hardware devices (provides the widest support for hardware platforms of any OS, including LEON processors), the compatibility with numerous communication protocols and standards, the availability of standardized development tools and a large developer community, open source license conditions that allow supplier independence, and the low adoption cost. Furthermore, although it is not an RTOS, as it is designed to be a general-purpose OS with no real-time functionality, it can support the development of real-time applications, as patches for the Linux kernel implementing real-time computing functionalities (known as PREEMPT_RT) are available. The use of a Linux OS in avionics systems has the potential to simplify the FSW design, as there is a large catalog of existing Linux applications that can be readily integrated into an OBC (such as data compression, operations scheduling, security, etc.). Moreover, parts of the software can be developed and debugged on a desktop computer and easily transferred to an OBC [200].

Linux is used in several CubeSat projects, including the QuakeSat [201], UWE-1 and UWE-2 [202] and Aalto-1 [203] missions. In addition, there are also certain initiatives designed to demonstrate the use of smartphones with the Android OS, which is based on Linux, to build satellite OBCs, such as the NASA PhoneSat project [204] and the STRaND-1 mission [205].

V. FUTURE BREAKTHROUGHS

Approximately 20 years after their first launch and thousands of missions being launched [4], CubeSats have proven their potential in facilitating high-quality scientific research and enabling new mission concepts. They are attractive platforms for a wide range of mission objectives, including the demonstration of cutting-edge space technologies, telecommunications, space science and astrophysics, and EO [8]. They could be used as piggyback missions to enhance the scientific return of larger traditional spacecraft. Simultaneously, CubeSats could offer the possibility of building innovative space system architectures, which have hitherto not been feasible due to the high costs associated with larger satellites. Among these, distributed space systems, such as swarms, constellations, and arrays of nanosatellites [88, 209–211], provide unprecedented temporal and spatial coverage. To meet the needs of next-generation science and exploration, it is essential to develop avionics systems characterized by three key attributes [22, 212]: modularity, reconfigurability, and autonomy. Emerging trends that will contribute to enhancing the capabilities of CubeSat C&DH subsystems include the use of distributed hybrid fault-tolerant architectures (§III-A4) and advanced AI-based algorithms [206].

Distributed hybrid fault-tolerant architectures. As discussed in §III-A4, hybrid architectures, which combine various processing units (multicore µPs, FPGAs, GPUs, etc.) on a single board or chip (SoC), are revolutionizing onboard CubeSat computing, increasing its performance, computational speed, and parallel processing capability, with significant implications for improving data and image processing. Multicore architectures will play a key role in next-generation flight computing, enabling parallel processing on a single chip and facilitating the implementation of new fault-tolerance techniques, such as task rescheduling or swapping (§II-C3). The Proton 600k multi-core computer developed by Space Micro (US; [213]) is an example of such an architecture, featuring an 8-core ARM Cortex processor. The combined use of various processing units will enable the development of distributed avionics systems, reducing the processing load on each chip, allowing flight software tasks to be optimized by distributing algorithms and applications over several cores, and increasing overall system performance and reliability. Furthermore, according to the latest developments in hybrid architectures (§III-A4), future CubeSat avionics systems should combine the use of COTS and rad-hard components to exploit the advantages of both devices, such as the high performance, low power consumption, and low cost typical of COTS units and the high reliability and flight heritage of rad-hard components [24].

Advanced AI-based algorithms. AI, particularly ML models such as NNs, has the potential to address complex problems by utilizing the inherent information within the data. Specifically, convolutional NNs, which are suitable for image processing, can significantly improve the scientific

TABLE 2. Main topics covered by the paper and key references.

Section	Objective	Category	Key Papers and References
II. Overview on C&DH Subsystem: Functions, Components and Techniques for Radiation Effect Mitigation	It discusses the general functional architecture of a C&DH subsystem, the effects of radiation on space components and the mitigation techniques typically implemented.	C&DH subsystem functional architecture	Furano *et al.*, 2018 [3] Cutler *et al.*, 2021 [1] THAG (ESA), *Technology Harmonization Dossier*, 2021 [22]
		Effect of radiation on space components and hardware fault-tolerant techniques	TI, *Radiation Handbook for Electronics*, 2019 [41] Prinzie *et al.*, 2021 [32] Luza *et al.*, 2022 [55]
		Software fault-tolerant techniques	Feng *et al.*, 2017 [65] Hillier *et al.*, 2019 [59]
III. CubeSat OBCs State-of-the-Art	It surveys commercial and mission-specific OBCs, categorizing them according to the nature of their processing core and focusing on hybrid architectures. Synthesise design elements to be considered, including communication interfaces.	SmallSats avionics state-of-the-art	NASA, *State-of-the-Art of Small Spacecraft Technology*, 2024 [23]
		Early CubeSat avionics systems and more recent advances on COTS µCs-based systems	Guertin *et al.*, 2015 [72] De Melo *et al.*, 2020 [144] Lara *et al.*, 2023 [73]
		Reconfigurable, hybrid systems and recent advances in fault-tolerant hybrid architectures	George *et al.*, 2018 [24] Amorim *et al.*, 2021 [138] Geist *et al.*, 2023 [135] Lüdtke *et al.*, 2023 [139]
		Communication infrastructure	Bouwmeester *et al.*, 2017 [153] Speretta *et al.*, 2023 [145]
IV. Flight Software State-of-the-Art	It analyzes the FSW development frameworks and OSs currently used in CubeSats.	FSW frameworks for NanoSats	Miranda *et al.*, 2019 [160] Farges *et al.*, 2022 [162]
		OSs for small embedded devices	Qutqut *et al.*, 2018 [186]
		Cybersecurity topics	Manulis *et al.*, 2021 [163] Curbo *et al.*, 2023 [164] Willbold *et al.*, 2023 [168]
V. Future Breakthroughs	It examines future trends in flight computing for nanosatellites	Multi-core processing systems	Dobiáš *et al.*, 2021 [68]
		AI for on-board image processing, task scheduling, autonomous navigation and anomalies detection	Russo *et al.*, 2022 [206] Horne *et al.*, 2023 [207] Zeleke *et al.*, 2023 [208]

value of satellite data by reducing the need for extensive preprocessing and postprocessing, which are typically required by traditional onboard processing techniques, while optimizing downlink data transmission. The ESA Φ-Sat program [214] is the first experiment to demonstrate the extraction of image features directly onboard a CubeSat using convolutional NNs implemented in a Vision Processing Unit (VPU). Future CubeSat avionics systems could exploit NNs, and ML-based algorithms in general, not only for onboard image processing, but also for navigation and platform control [215], task scheduling [208] and for FDIR functions. The system could be trained to recognize early anomalies and malfunctions onboard the satellite based on telemetry data correlations, enabling early warning of potential failures [207, 216]. Furthermore, as demonstrated by the success of KP Labs Intuition-1 mission and its novel Leopard DPU, the use of FPGA SoCs as hardware accelerators (AI engines) could facilitate the implementation of AI algorithms in CubeSats. FPGA SoCs offer significant advantages for the acceleration of algorithms, as they enable efficient implementation of sequential components on processors and accelerate highly parallel or iterative operations on the FPGA with a low power consumption [132]. As a result, increasingly complex AI algorithms could be implemented, overcoming the challenges of computing power and memory resources associated with the use of ML models in CubeSats.

VI. CONCLUSIONS

This paper systematically reviews the development methodologies, radiation mitigation techniques, and hardware and software platforms that currently shape the state-of-the-art in the OBCs subsystems for CubeSats. On-board computing is a key factor when designing CubeSat missions, as high-performance OBCs are required to address the computational challenges of future missions. Consequently, it is essential to review currently available technologies to gain a full understanding of their performance and limitations, so that engineers can make the most appropriate choices when designing C&DH subsystems for their missions. Tab. 2 categorizes the main topics covered in our analysis and provides a comprehensive overview of the most important literature considered in our survey, summarizing key papers and references for each topic.

The main lesson learned from this analysis is that CubeSat OBCs use different options for their hardware implementation, aligning with advancements in embedded sys-

tems. The OBCs of CubeSat missions with reduced performance requirements and intended mainly for the LEO environment, tipically use low-power COTS microcontrollers (mainly based on the ARM architecture or the MSP430 series), which offer the optimum trade-off between power consumption and the performance required for basic tasks, such as monitoring the health status of the on-board subsystems and telemetry. However, as the C&DH subsystem progressively integrates more functions, such as complex ADCS calculations or payload data processing, the OBC computing requirements increase, leading to higher performance devices such as FPGA or GPU SoCs. These OBCs offer significantly higher computing capabilities, enabling advanced processing tasks, such as the executing of complex data processing algorithms directly onboard the satellite. The use of AI-based algorithms for real-time on-board data analysis drives many current and future CubeSat missions. Small boards integrated with hardware accelerators, such as GPUs, VPUs, and FPGA SoCs, are already available on the market. Among these, FPGAs, in particular, appear promising for DL model acceleration, offering a trade-off between high portability, configuration flexibility, performance, low cost, and low power consumption. Rad-hard versions of FPGAs are also available, potentially extending the market for DL-based solutions to long-term LEO or interplanetary exploration missions. Additionally, with the increasing interest in employing CubeSat platforms beyond LEO, careful consideration must be given to hardware component selection and fault-tolerance techniques. Following the latest developments, future systems could benefit from hybrid fault-tolerant architectures that combine COTS and rad-hard devices.

From a FW perspective, several frameworks can be used to facilitate CubeSat FSW development. The challenge for future missions lies in designing modular FSWs that simplify the debugging process and ensure their scalability and extensibility to different mission objectives. CubeSat FSWs shall include fault-tolerance software techniques based on the use of multicore processors (such as task rescheduling) and exploit the benefits of using OSs for real-time multiprocessing. It should also incorporate increasingly complex AI algorithms.

REFERENCES

[1] J. W. Cutler and J. Beningo, "On-Board Data Handling Systems," in *Cubesat Handbook* (C. Cappelletti, S. Battistini, and B. K. Malphrus, eds.), ch. 10, pp. 199–219, Academic Press, 2021.

[2] N. P. Fillery and D. Stanton, "Telemetry, Command, Data Handling and Processing," in *Spacecraft Systems Engineering*, ch. 13, pp. 439–466, John Wiley & Sons, Ltd, 2011.

[3] G. Furano and A. Menicucci, "Roadmap for On-Board Processing and Data Handling Systems in Space," in *Dependable Multicore Architectures at Nanoscale* (M. Ottavi, D. Gizopoulos, and S. Pontarelli, eds.), pp. 253–281, Cham: Springer International Publishing, 2018.

[4] E. Kulu, *Nanosats database*. Accessed: June 16, 2024. [Online]. Available: https://www.nanosats.eu/.

[5] H. Heidt, J. Puig-Suari, A. S. Moore, S. Nakasuka, and R. J. Twiggs, "CubeSat: A New Generation of Picosatellite for Education and Industry Low-Cost Space Experimentation," in *Proceedings of the 15th Annual/USU Conference on Small Satellites*, 2000.

[6] J. Puig-Suari, C. S. Turner, and R. J. Twiggs, "CubeSat: The Development and Launch Support Infrastructure for Eighteen Different Satellite Customers on One Launch," in *Proceedings of the 14th Annual AIAA/USU Conference on Small Satellites*, 2001.

[7] Cal Poly CubeSat Laboratory, *CubeSat Design Specification Revision 14.1*. Accessed: June 16, 2024. [Online]. Available: https://www.cubesat.org/cubesatinfo.

[8] A. Poghosyan and A. Golkar, "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, vol. 88, pp. 59–83, 2017.

[9] R. Hevner, W. Holemans, J. Puig-Suari, and R. J. Twiggs, "An Advanced Standard for CubeSats," in *Proceedings of the 25th Annual AIAA/USU Conference on Small Satellites*, 2011.

[10] J. Bouwmeester and J. Guo, "Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology," *Acta Astronautica*, vol. 67, no. 7, pp. 854–862, 2010.

[11] D. Selva and D. Krejci, "A survey and assessment of the capabilities of Cubesats for Earth observation," *Acta Astronautica*, vol. 74, pp. 50–68, 2012.

[12] E. Buchen, "Small Satellite Market Observations," in *Proceedings of the 29th Annual AIAA/USU Conference on Small Satellites*, 2015.

[13] A. Zeedan and T. Khattab, "A Critical Review of Baseband Architectures for CubeSats Communication Systems," 2022. *arXiv:2201.09748*.

[14] F. R. Davoli, C. I. Kourogiorgas, M. Marchese, A. D. Panagopoulos, and F. Patrone, "Small satellites and CubeSats: Survey of structures, architectures, and protocols," *International Journal of Satellite Communications and Networking*, vol. 37, pp. 343 – 359, 2018.

[15] J. Crusan and C. Galica, "NASA's CubeSat Launch Initiative: Enabling broad access to space," *Acta Astronautica*, vol. 157, pp. 51–60, Apr. 2019.

[16] R. Walker, D. Koschny, C. Bramanti, I. Carnelli, E. Team, *et al.*, "Miniaturised Asteroid Remote Geophysical Observer (M-ARGO): a stand-alone deep space CubeSat system for low-cost science and exploration missions," in *6th Interplanetary CubeSat Workshop, Cambridge, UK*, vol. 30, 2017.

[17] S. Speretta, A. Cervone, P. Sundaramoorthy, R. Noomen, S. Mestry, A. Cipriano, F. Topputo, J. Biggs, P. Di Lizia, M. Massari, *et al.*, "LUMIO: an autonomous CubeSat for lunar exploration," *Space operations: inspiring Humankind's future*,

pp. 103–134, 2019.

[18] P. Tortora and V. Di Tana, "LICIACube, the Italian witness of DART impact on Didymos," in *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pp. 314–317, IEEE, 2019.

[19] V. Di Tana, C. Fiori, S. Simonetti, and S. Pirrotta, "ArgoMoon, a multipurpose CubeSat platform for missions in Moon vicinity and orbit," in *European Planetary Science Congress*, pp. EPSC2018–1181, 2018.

[20] W. Ley, F. Merkle, J. Block, J. Kreuser, R. Röder, A. Kohlhase, R. Schlitt, H. D. Schmitz, C. Arbinger, B. Lübke-Ossenbeck, S. Montenegro, and P. Turner, "Subsystems of Spacecraft," in *Handbook of Space Technology*, ch. 4, pp. 200–398, John Wiley & Sons, Ltd, 2009.

[21] G. Lentaris, K. Maragos, I. Stratakos, L. Papadopoulos, O. Papanikolaou, D. Soudris, M. Lourakis, X. Zabulis, D. Gonzalez-Arjona, and G. Furano, "High-performance embedded computing in space: Evaluation of platforms for vision-based navigation," *Journal of Aerospace Information Systems*, vol. In press, 02 2018.

[22] Technology Harmonisation Advisory Group (THAG), *Technology Harmonization Dossier. On Board Computer, Data Handling Systems and Microelectronics*. European Space Agency, 2021.

[23] NASA, *State-of-the-Art of Small Spacecraft Technology*, 2024. Accessed: June 16, 2024. [Online]. Available: https://www.nasa.gov/smallsat-institute/sst-soa/.

[24] A. D. George and C. M. Wilson, "Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 458–470, 2018.

[25] M. Alam, A. Khamees, T. Aboelnaga, A. Amer, A. Harbi, M. Alamir, H. Alarwsh, and O. A. Elsayed, "Design and Implementation of an Onboard Computer and payload for Nano Satellite (CubeSat)," in *The International Undergraduate Research Conference*, vol. 5, pp. 361–364, The Military Technical College, 2021.

[26] ESA, *Space AVionics Open Interface aRchitecture (SAVOIR)*. Accessed: June 16, 2024. [Online]. Available: https://savoir.estec.esa.int/.

[27] S. Duzellier, "Radiation effects on electronic devices in space," *Aerospace Science and Technology*, vol. 9, no. 1, pp. 93–99, 2005.

[28] B. Todd and S. Uznanski, "Radiation Risks and Mitigation in Electronic Systems," in *CAS - CERN Accelerator School: Power Converters*, pp. 245–263, 2014.

[29] T. Kuwahara, Y. Tomioka, K. Fukuda, N. Sugimura, and Y. Sakamoto, "Radiation effect mitigation methods for electronic systems," *2012 IEEE/SICE International Symposium on System Integration (SII)*, pp. 307–312, 2012.

[30] D. P. Siewiorek and R. S. Swarz, "2 - Faults and their manifestations," in *Reliable Computer Systems (Second Edition)* (D. P. Siewiorek and R. S. Swarz, eds.), pp. 22–78, Boston: Digital Press, second edition ed., 1992.

[31] J. Srour and J. McGarrity, "Radiation effects on microelectronics in space," *Proceedings of the IEEE*, vol. 76, no. 11, pp. 1443–1469, 1988.

[32] J. Prinzie, F. M. Simanjuntak, P. Leroux, and T. Prodromakis, "Low-power electronic technologies for harsh radiation environments," *Nature Electronics*, vol. 4, pp. 243 – 253, 2021.

[33] R. H. Maurer, M. E. Fraeman, M. N. Martin, and D. R. Roth, "Harsh Environments: Space Radiation Environment, Effects, and Mitigation," *Johns Hopkins Apl Technical Digest*, vol. 28, pp. 17–29, 2008.

[34] J. Bouwmeester and J. Guo, "Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology," *Acta Astronautica*, vol. 67, no. 7, pp. 854–862, 2010.

[35] M. Barella, G. Sanca, F. G. Marlasca, W. R. Acevedo, D. Rubi, M. G. Inza, P. Levy, and F. Golmar, "Studying ReRAM devices at Low Earth Orbits using the LabOSat platform," *Radiation Physics and Chemistry*, vol. 154, pp. 85–90, 2019.

[36] "Space Radiation Effects on Electronic Components in Low-Earth Orbit," 1999. Accessed: June 16, 2024. [Online]. Available: https://llis.nasa.gov/lesson/824.

[37] S. Samwel, A. Hady, J. Mikhail, M. Ibrahim, and Y. Hanna, "Studying the Total Ionizing Dose and Displacement Damage Dose effects for various orbital trajectories," in *First Middle East-Africa Regional IAU Meeting*, pp. 55–58, 2008.

[38] J. Lipovetzky, M. Garcia-Inza, M. R. Cañete, G. Redin, S. Carbonetto, M. Echarri, F. Golmar, F. G. Marlasca, M. Barella, G. A. Sanca, P. Levy, and A. Faigón, "COTS MOS Dosimetry on the MeMOSat Board, Results After 2.5 Years in Orbit," 2020. *arXiv: 2007.00143*.

[39] M. Dowd, "How Rad Hard Do You Need? The Changing Approach To Space Parts Selection?," *Maxwell Technologies Microelectronics*, 2003.

[40] J. Srour, C. Marshall, and P. Marshall, "Review of Displacement Damage Effects in Silicon Devices," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 653–670, 2003.

[41] R. Baumann and K. Kruckmeyer, "Radiation Handbook for Electronics," 2019. Accessed: June 16, 2024. [Online]. Available: https://www.ti.com/applications/industrial/aerospace-defense/space/radiation-handbook-for-electronics.html.

[42] J. Wang, R. Katz, J. Sun, B. Cronquist, J. McCollum, T. Speers, and W. Plants, "SRAM based reprogrammable FPGA for space applications," *IEEE Transactions on Nuclear Science*, vol. 46, no. 6, pp. 1728–1735, 1999.

[43] T. C. Macleod, W. H. Sims, K. Varnavas, R. Sayyah, and F. D. Ho, "Satellite test of radiation impact on

Ramtron 512K FRAM," in *2009 10th Annual Non-Volatile Memory Technology Symposium (NVMTS)*, pp. 24–27, 2009.

[44] A. J. Tylka, W. F. Dietrich, P. R. Boberg, E. C. Smith, and J. H. Adams, "Single event upsets caused by solar energetic heavy ions," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2758–2766, 1996.

[45] D. W. Caldwell and D. A. Rennels, "A Minimalist Fault-Tolerant Microcontroller Design for Embedded Spacecraft Computing," *The Journal of Supercomputing*, vol. 16, pp. 7–25, 2000.

[46] C. S. M. Rao and A. Chavan, "Review on Radiation Hardness Assurance by Design, Process and NextGen Devices," *Journal of Physics: Conference Series*, vol. 1916, 2021.

[47] P. J. Botma, A. Barnard, and W. H. Steyn, "Low cost fault tolerant techniques for nano/pico-satellite applications," in *2013 Africon*, pp. 1–5, 2013.

[48] C. M. Fuchs, N. M. Murillo, A. Plaat, E. van der Kouwe, D. Harsono, and T. Stefanov, "Fault-Tolerant Nanosatellite Computing on a Budget," in *2018 18th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pp. 1–8, 2018.

[49] A. Burns and A. Wellings, "2 - Reliability and fault tolerance," in *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*, Addison-Wesley Educational Publishers Inc, 2009.

[50] I. Brandao Machado Matsuo, L. Zhao, and W.-J. Lee, "A Dual Modular Redundancy Scheme for CPU-FPGA Platform-Based Systems," *IEEE Transactions on Industry Applications*, vol. PP, pp. 1–1, 07 2018.

[51] GAUSS S.r.l, *ABACUS OBC*. Accessed: June 16, 2024. [Online]. Available: https://www.gaussteam.com/products/onboard-computer/abacus-2/.

[52] M. Sim and Y. Zhuang, "A Dual Lockstep Processor System-on-a-Chip for Fast Error Recovery in Safety-Critical Applications," in *IECON 2020 - The 46th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2231–2238, 10 2020.

[53] T. Kuwahara, *FPGA-based Reconfigurable On-board Computing Systems for Space Applications*. PhD thesis, Universität Stuttgart, 2010.

[54] Ingegneria Marketing Tecnologia (IMT), *Cubesat On-Board Computer*. Accessed: June 16, 2024. [Online]. Available: https://www.satcatalog.com/component/cubesat-on-board-computer/,.

[55] L. M. Luza, F. Wrobel, L. Entrena, and L. Dilillo, "Impact of Atmospheric and Space Radiation on Sensitive Electronic Devices," in *2022 IEEE European Test Symposium (ETS)*, pp. 1–10, 2022.

[56] P. Shirvani, N. Saxena, and E. McCluskey, "Software-implemented EDAC protection against SEUs," *IEEE Transactions on Reliability*, vol. 49, no. 3, pp. 273–284, 2000.

[57] R. W. Hamming, "Error Detecting and Error Correcting Codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[58] N. Mhatre and S. Aras, "A hybrid approach to radiation fault tolerance in small satellite applications," in *62nd International Astronautical Congress (IAC), Cape Town, South Africa*, vol. 11, 2011.

[59] C. Hillier, V. Balyan, *et al.*, "Error Detection and Correction On-Board Nanosatellites Using Hamming Codes," *Journal of Electrical and Computer Engineering*, vol. 2019, 2019.

[60] X. Zhang, "VLSI Architectures for Reed–Solomon Codes: Classic, Nested, Coupled, and Beyond," *IEEE Open Journal of Circuits and Systems*, vol. 1, pp. 157–169, 2020.

[61] D. Mavis, P. Eaton, M. Sibley, R. Lacoe, E. Smith, and K. Avery, "Multiple Bit Upsets and Error Mitigation in Ultra-Deep Submicron SRAMS," *IEEE Transactions on Nuclear Science*, vol. 55, no. 6, pp. 3288–3294, 2008.

[62] C. Sansoe and M. Tranchero, *Use of FRAM Memories in Spacecrafts*, pp. 213–230. InTech, 2011.

[63] J. Heidecker, "MRAM Technology Status," tech. rep., Jet Propulsion Laboratory, California Institute of Technology, 2013.

[64] F. Zahoor, T. Z. A. Zulkifli, and F. A. Khanday, "Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (mlc) Storage, Modeling, and Applications," *Nanoscale Research Letters*, vol. 15, 2020.

[65] Y. Feng, J. Gong, G. Hua, and M. Yang, *Software Fault-Tolerance Techniques*, ch. 5, pp. 151–178. John Wiley & Sons, Ltd, 2017.

[66] I. Sünter, A. Slavinskis, U. Kvell, A. Vahter, H. Kuuste, M. Noorma, J. Kutt, R. Vendt, K. Tarbe, M. Pajusalu, M. Veske, and T. Ilves, "Firmware Updating Systems for Nanosatellites," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 5, pp. 36–44, 2016.

[67] S. Ghosh, R. Melhem, and D. Mosse, "Fault-Tolerance Through Scheduling of Aperiodic Tasks in Hard Real-Time Multiprocessor Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 3, pp. 272–284, 1997.

[68] P. Dobiáš, E. Casseau, and O. Sinnen, "Improving the CubeSat reliability thanks to a multiprocessor system using fault tolerant online scheduling," *Microprocessors and Microsystems*, vol. 85, p. 104312, 2021.

[69] N. Murphy and M. Barr, "Watchdog timers," *Embedded Systems Programming*, vol. 14, no. 11, pp. 79–80, 2001.

[70] A. V. Dias, J. A. Pomilio, and S. Finco, "A Current Limiting Switch for Applications in Space Power Systems," in *2017 IEEE Southern Power Electronics Conference (SPEC)*, pp. 1–6, 2017.

[71] W. Sajjad, A. Shafique, U. B. Khalid, and R. Mahmood, "Design of Reliable, Low Power, and Enhanced

Performance Architecture of On-Board Computer for CubeSats," *IEEE Journal on Miniaturization for Air and Space Systems*, 2023.

[72] S. M. Guertin, M. Amrbar, and S. Vartanian, "Radiation Test Results for Common CubeSat Microcontrollers and Microprocessors," in *2015 IEEE Radiation Effects Data Workshop (REDW)*, pp. 1–9, 2015.

[73] C. A. Lara, M. Fragoso, L. M. Juárez, L. Barboni, R. Reyes, R. Vázquez, J. P. Acle, and S. de la Rosa, "Fault Tolerant Architecture Design of a CubeSat Command and Data Handling System," in *2023 IEEE 24th Latin American Test Symposium (LATS)*, pp. 1–6, 2023.

[74] K. V. C. K. de Souza, Y. Bouslimani, M. Ghribi, and T. Boutot, "On-Board Computer and Testing Platform for CubeSat Development," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 4, no. 2, pp. 199–211, 2023.

[75] L. Gagliardi, F. D. Nardo, G. A. Sanca, F. Izraelevitch, M. Cveczilberg, and F. Golmar, "Labosat-02: Hardware and firmware development of an on-board computer for small satellites," *IEEE Embedded Systems Letters*, pp. 1–1, 2023.

[76] H. Leppinen, A. Kestilä, P. Pihajoki, J. Jokelainen, and T. Haunia, "On-board Data Handling for Ambitious Nanosatellite Missions using Automotive-grade Lockstep Microcontrollers," in *Small Satellites Systems and Services-The 4S Symposium*, pp. 1–10, 2014.

[77] B. Sheela Rani, R. R. Santhosh, L. S. Prabhu, M. Federick, V. Kumar, and S. Santhosh, "A Survey to Select Microcontroller for Sathyabama Satellite's On Board Computer Subsystem," in *Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010)*, pp. 134–137, 2010.

[78] J. Praks, M. R. Mughal, R. Vainio, P. Janhunen, J. Envall, P. Oleynik, A. Näsilä, H. Leppinen, P. Niemelä, A. Slavinskis, J. Gieseler, P. Toivanen, T. Tikka, T. Peltola, A. Bosser, G. Schwarzkopf, N. Jovanovic, B. Riwanto, A. Kestilä, A. Punkkinen, R. Punkkinen, H.-P. Hedman, T. Säntti, J.-O. Lill, J. Slotte, H. Kettunen, and A. Virtanen, "Aalto-1, multi-payload CubeSat: Design, integration and launch," *Acta Astronautica*, vol. 187, pp. 370–383, 2021.

[79] J. H. Davies, "Chapter 2 - The Texas Instruments MSP430," in *MSP430 Microcontroller Basics* (J. H. Davies, ed.), pp. 21–42, Burlington: Newnes, 2008.

[80] J. Strnadel and P. Rajnoha, "Reflecting RTOS model during WCET timing analysis: MSP430/FreeRTOS case study," *Acta Electrotechnica et Informatica*, vol. 12, no. 4, p. 17, 2012.

[81] P. Villa, L. Slongo, J. Salamanca, V. Martins, F. Silva, S. Martinez, L. Mariga, B. Eiterer, I. Vidal, V. Menegon, *et al.*, "A complete Cubesat mission: the FloripaSat experience," in *1st IAA Latin American Cubesat Workshop*, vol. 2, pp. 307–314, 2014.

[82] Spacemanic, *Eddie Onboard Computer*. Accessed: June 16, 2024. [Online]. Available https://www.spacemanic.com/eddie-onboard-computer/.

[83] J. Mangan, D. Murphy, R. Dunwoody, M. Doyle, A. Ulyanov, L. Hanlon, B. Shortt, and S. McBreen, "Embedded Firmware Development for a Novel CubeSat Gamma-Ray Detector," in *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, pp. 14–22, 2021.

[84] G. A. Sanca, M. Barella, F. G. Marlasca, N. Alvarez, P. Levy, and F. Golmar, "LabOSat-01: a payload for in-orbit device characterization," *IEEE Embedded Systems Letters*, pp. 1–1, 2023.

[85] R. Berger, D. Bayles, R. Brown, S. Doyle, A. Kazemzadeh, K. Knowles, D. Moser, J. Rodgers, B. Saari, D. Stanley, and B. Grant, "The RAD750™ - A Radiation Hardened PowerPC™ Processor for High Performance Spaceborne Applications," in *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, vol. 5, pp. 2263–2272 vol.5, 2001.

[86] N. F. Haddad, R. D. Brown, R. Ferguson, A. T. Kelly, R. K. Lawrence, D. M. Pirkl, and J. C. Rodgers, "Second Generation (200MHz) RAD750 Microprocessor Radiation Evaluation," in *2011 12th European Conference on Radiation and Its Effects on Components and Systems*, pp. 877–880, 2011.

[87] BAE Systems, *Radiation-hardened electronics product guide*. Accessed: June 16, 2024. [Online]. Available: https://www.baesystems.com/en/product/radiation-hardened-electronics.

[88] P. Kelly and R. Bevilacqua, "The constellation for Mars position acquisition using small satellites: Cubesat design feasibility and challenges," in *Advances in the Astronautical Sciences 4th IAA Dynamics and Control of Space Systems Conference*, vol. 163, pp. 629–640, 2018.

[89] B. LaMeres, C. Delaney, M. Johnson, C. Julien, K. Zack, B. Cunningham, T. Kaiser, L. Springer, and D. Klumpar, "Next on the Pad: RadSat - A Radiation Tolerant Computer System," in *31th Annual AIAA/USU Conference on Small Satellites*, 2017.

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

[91] Argotech, *FERMI Deep Space On-board Computer*. Accessed: June 16, 2024. [Online]. Available: https://www.argotecgroup.com/products-2/.

[92] A. Hanafi, M. Karim, I. Latachi, T. Rachidi, S. Dahbi, and S. Zouggar, "FPGA-based Secondary On-Board Computer System for Low-Earth-Orbit Nano-satellite ," in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pp. 1–6, IEEE, 2017.

[93] K. Ngo, T. Mohammadat, and J. Öberg, "Towards a Single Event Upset Detector Based on COTS FPGA,"

in *2017 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–6, IEEE, 2017.

[94] M. Qasaimeh, K. Denolf, A. Khodamoradi, M. Blott, J. Lo, L. Halder, K. Vissers, J. Zambreno, and P. H. Jones, "Benchmarking Vision Kernels and Neural Network Inference Accelerators on Embedded Platforms," *Journal of Systems Architecture*, vol. 113, p. 101896, 2021.

[95] N. Hou, D. Zhang, G. Du, and Y. Song, "An FPGA-Based Multi-Core System for Synthetic Aperture Radar Data Processing," in *2014 International Conference on Anti-Counterfeiting, Security and Identification (ASID)*, pp. 1–4, 2014.

[96] D. Mandl, G. Crum, V. Ly, M. Handy, K. F. Huemmrich, L. Ong, B. Holt, and R. Maharaja, "Hyperspectral Cubesat Constellation for Natural Hazard Response (Follow-on)," in *30th Annual AIAA/USU Conference on Small Satellites*, 2016.

[97] B. Qi, H. Shi, Y. Zhuang, H. Chen, and L. Chen, "On-Board, Real-Time Preprocessing System for Optical Remote-Sensing Imagery," *Sensors*, vol. 18, p. 1328, 04 2018.

[98] J. A. Hogan, R. J. Weber, B. J. LaMeres, and T. Kaiser, "Network-on-Chip for a Partially Reconfigurable FPGA System," in *Proceedings of the 27th international ACM conference on International conference on supercomputing*, pp. 473–474, 2013.

[99] SkyLabs, *NANOhpm-obc*. Accessed: June 16, 2024. [Online]. Available: https://www.skylabs.si/products/nanohpm-obc/.

[100] AAC Clyde Space, *SIRIUS-OBC-LEON3FT*. Accessed: June 16, 2024. [Online]. Available: https://www.aac-clyde.space/what-we-do/space-products-components/command-data-handling/smallsat-sirius-obc.

[101] J.-J. Wang, B. Conquist, B. Sin, J. Moriarta, and R. B. Katz, "Antifuse FPGA for Space Applications," in *RADECS 97. Fourth European Conference on Radiation and its Effects on Components and Systems*, 1997.

[102] L. Rockett, D. Patel, S. Danziger, B. Cronquist, and J. Wang, "Radiation Hardened FPGA Technology for Space Applications," in *2007 IEEE Aerospace Conference*, pp. 1–7, IEEE, 2007.

[103] H. Quinn, "Radiation effects in reconfigurable FPGAs," *Semiconductor Science and Technology*, vol. 32, no. 4, p. 044001, 2017.

[104] Z. Liu, Z. Lu, L. Huang, Z. Yao, Z. Lu, and J. Zhang, "Recent Advances on Reliability of FPGAs in a Radiation Environment," *Microelectronics Journal*, vol. 148, p. 106176, 2024.

[105] NanoXplore, *NG-MEDIUM SPACE, NX1H35AS*. Accessed: June 16, 2024. [Online]. Available: https://nanoxplore.com/index.php/product/ng-medium/.

[106] L. M. Luza, C. A. Rigo, E. D. Tramontin, V. M. G. Martins, S. V. Martinez, L. K. Slongo, L. O. Seman, L. Dilillo, and E. A. Bezerra, "Enabling deep-space CubeSat missions through state-of-the-art radiation-hardened technologies," in *3rd IAA Latin American CubeSat Workshop (IAA-LACW 2018)*, 2018.

[107] J. Heiner, N. Collins, and M. Wirthlin, "Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing," in *2008 IEEE Aerospace Conference*, pp. 1–10, IEEE, 2008.

[108] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, "FPGA partial reconfiguration via configuration scrubbing," in *2009 International Conference on Field Programmable Logic and Applications*, pp. 99–104, IEEE, 2009.

[109] A. Jacobs, C. Conger, and A. D. George, "Multiparadigm Space Processing for Hyperspectral Imaging," in *2008 IEEE Aerospace Conference*, pp. 1–11, 2008.

[110] AMD-Xilinx, *AMD Zynq™ 7000 SoCs*. Accessed: June 16, 2024. [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/soc/zynq-7000.html.

[111] Microchip Technology, *SmartFusion® 2 SoC FPGAs*. Accessed: June 16, 2024. [Online]. Available: https://www.microchip.com/en-us/products/fpgas-and-plds/system-on-chip-fpgas/smartfusion-2-fpgas.

[112] AMD-Xilinx, *AMD Zynq™ UltraScale+™ MPSoCs*. Accessed: June 16, 2024. [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/soc/zynq-ultrascale-plus-mpsoc.html.

[113] Xiphos Technologies, *Q7s Processor*. Accessed: June 16, 2024. [Online]. Available: https://satsearch.co/products/xiphos-q7s-processor.

[114] Xiphos Technologies, *Q8s Processor*. Accessed: June 16, 2024. [Online]. Available: https://satsearch.co/products/xiphos-q8s-processor.

[115] AAC Clyde Space, *KRYTEN-M3*. Accessed: June 16, 2024. [Online]. Available: https://www.aac-clyde.space/what-we-do/space-products-components/command-data-handling/kryten-m3.

[116] Space Inventor, *OBC-P4*. Accessed: June 16, 2024. [Online]. Available: https://space-inventor.com/modules/on-board-computer?item=on-board-computer.

[117] Innoflight, *CFC-400*. Accessed: June 16, 2024. [Online]. Available: https://www.satcatalog.com/component/cfc-400/.

[118] KP Labs, *Leopard*. Accessed: June 16, 2024. [Online]. Available: https://kplabs.space/leopard/.

[119] L. Riha, J. Le Moigne, and T. El-Ghazawi, "Optimization of Selected Remote Sensing Algorithms for Many-Core Architectures," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 12, pp. 5576–5587, 2016.

[120] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pp. 779–788, 2016.

[121] Z. El Khatib, A. B. Mnaouer, S. Moussa, M. A. B. Abas, N. A. Ismail, F. Abdulgaleel, I. Elmasri, and L. Ashraf, "LoRa-enabled GPU-based CubeSat Yolo Object Detection with Hyperparameter Optimization," in *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–4, IEEE, 2022.

[122] A. Elshazly, A. Elliethy, and M. Elshafey, "Tactics Overview for Implementing High-Performance Computing on Embedded Platforms," in *IOP Conference Series: Materials Science and Engineering*, vol. 1172, p. 012034, IOP Publishing, 2021.

[123] R. Wu, B. Zhang, and M. Hsu, "Clustering billions of data points using GPUs," in *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, pp. 1–6, 2009.

[124] NVIDIA, *NVIDIA® Jetson™ TX2*. Accessed: June 16, 2024. [Online]. Available: https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-tx2/.

[125] NVIDIA, *NVIDIA® Jetson Xavier™ NX Series*. Accessed: June 16, 2024. [Online]. Available: https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-xavier-nx/.

[126] AMD, *1st and 2nd Generation AMD Embedded G-Series SoCs*. Accessed: June 16, 2024. [Online]. Available https://www.amd.com/system/files/2017-06/g-series-soc-product-brief.pdf.

[127] C. Adams, A. Spain, J. Parker, M. Hevert, J. Roach, and D. Cotten, "Towards an Integrated GPU Accelerated SoC as a Flight Computer for Small Satellites," in *2019 IEEE Aerospace Conference*, pp. 1–7, IEEE, 2019.

[128] C. Bonesteel, E. Tichenor, E. Miller, and A. Rodriguez, "Co-Operating Systems: A Technical Overview of Multiple Onboard Operating Systems," in *36th Annual Small Satellite Conference*, Utah State University, 2022.

[129] Spiral Blue, *SE-1. Space Edge One*. Accessed: June 16, 2024. [Online]. Available: https://www.spiralblue.space/edge-computing-for-space.

[130] Unibap, *SpaceCloud® iX5-106*. Accessed: June 16, 2024. [Online]. Available: https://unibap.com/solutions/spacecloud-hardware/ix5/.

[131] R. Wright, M. Nunes, P. Lucey, L. Flynn, T. George, S. Gunapala, D. Ting, S. Rafol, A. Soibel, C. Ferrari-Wong, A. Flom, J. Mecikalski, P. Thenkabail, and t. H. Team, "HyTI: Thermal Hyperspectral Imaging from A Cubesat Platform," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 4982–4985, 2019.

[132] A. Geist, C. Brewer, M. Davis, N. G. Franconi, S. Heyward, T. W. Wise, G. Crum, and D. Petrick, "Space-Cube v3.0 NASA Next-Generation High-Performance Processor for Science Applications," in *33rd Annual AIAA/USU Conference on Small Satellites*, 2019.

[133] AMD-Xilinx, *AMD Kintex™ UltraScale™ FPGAs*. Accessed: June 16, 2024. [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/kintex-ultrascale.html#!

[134] Microchip Technology, *RTAX™ Radiation-Tolerant FPGAs*. Accessed: June 16, 2024. [Online]. Available: https://www.microchip.com/en-us/products/fpgas-and-plds/radiation-tolerant-fpgas/rtax-s.

[135] A. Geist, G. Crum, C. Brewer, D. Afanasev, S. Sabogal, D. Wilson, J. Goodwill, J. Marshall, N. Perryman, N. Franconi, T. Chase, and T. t. Wise, "NASA Space-Cube Next-Generation Artificial-Intelligence Computing for STP-H9-SCENIC on ISS," in *37th Annual Small Satellite Conference*, 2023.

[136] Microchip Technology, *Radiation-Tolerant ProASIC® 3 FPGAs*. Accessed: June 16, 2024. [Online]. Available: https://www.microchip.com/en-us/products/fpgas-and-plds/radiation-tolerant-fpgas/rt-proasic-3.

[137] Thales Alenia Space Germany, *multiMIND Payload Processing Core*. Accessed: June 16, 2024. [Online]. Available: https://connectivity.esa.int/projects/multimind-eive.

[138] R. Amorim, R. Martins, M. Ghiglione, T. Helfers, and P. Harikrishnan, "Dependable MPSoC Framework For Mixed Criticaly Applications," in *2nd European Workshop on On-Board Data Processing (OBDP2021)*, 06 2021.

[139] D. Lüdtke, T. Firchau, C. G. Cortes, A. Lund, A. M. Nepal, M. M. Elbarrawy, Z. H. Hammadeh, J.-G. Meß, P. Kenny, F. Brömer, M. Mirzaagha, G. Saleip, H. Kirstein, C. Kirchhefer, and A. Gerndt, "ScOSA on the Way to Orbit: Reconfigurable High-Performance Computing for Spacecraft," in *2023 IEEE Space Computing Conference (SCC)*, pp. 34–44, 2023.

[140] L. Manoliu, B. Schoch, S. Haussmann, A. Tessmann, R. Henneberger, J. Freese, F. Steinmetz, D. Wrana, J. Wörmann, M. Koller, and I. Kallfass, "The technology platform of the EIVE CubeSat mission for high throughput downlinks at W-band," *Acta Astronautica*, vol. 205, pp. 80–93, 2023.

[141] VORAGO Technologies, *Arm® Cortex®-M0 MCUs*. Accessed: June 16, 2024. [Online]. Available: https://www.voragotech.com/arm-cortexm0-family.

[142] Microchip Technology, *PolarFire® FPGA Family*. Accessed: June 16, 2024. [Online]. Available: https://www.microchip.com/en-us/products/fpgas-and-plds/radiation-tolerant-fpgas/rt-polarfire-fpgas.

[143] A. Clements, *Computer Performance*. Accessed: June 16, 2024. [Online]. Available:http://alanclements.org/performance.html.

[144] A. C. C. P. de Melo, D. C. Café, and R. Alves Borges, "Assessing Power Efficiency and Performance in

Nanosatellite Onboard Computer for Control Applications," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 1, no. 2, pp. 110–116, 2020.

[145] S. Speretta, J. Bouwmeester, A. Menicucci, S. Di Mascio, and M. S. Uludag, "16 - Command and data handling systems," in *Next Generation CubeSats and SmallSats* (F. Branz, C. Cappelletti, A. J. Ricco, and J. W. Hines, eds.), pp. 369–399, Elsevier, Jan. 2023.

[146] F. G. H. Leite, R. B. B. Santos, N. H. Medina, V. A. P. Aguiar, R. C. Giacomini, N. Added, F. Aguirre, E. L. Macchione, F. Vargas, and M. A. G. da Silveira, "Ionizing Radiation Effects on a COTS Low-cost RISC Microcontroller," in *2017 18th IEEE Latin American Test Symposium (LATS)*, pp. 1–4, 2017.

[147] H. Akah, D. Elfiky, K. Shahin, E. Elemam, and A. Anwar, "Total Ionizing Dose Effects on Commercial ARM Microcontroller for Low Earth Orbit Satellite Subsystems," in *International Conference on Aerospace Sciences and Aviation Technology*, vol. 17, pp. 1–8, The Military Technical College, 2017.

[148] Z. Liu, Z. Lu, L. Huang, Z. Yao, Z. Lu, and J. Zhang, "Recent advances on reliability of FPGAs in a radiation environment," *Microelectronics Journal*, vol. 148, p. 106176, 2024.

[149] Microchip Technology, *RTG4™ Radiation-Tolerant FPGAs*. Accessed: June 16, 2024. [Online]. Available: https://www.microchip.com/en-us/products/fpgas-and-plds/radiation-tolerant-fpgas/rtg4-radiation-tolerant-fpgas.

[150] S. van der Linden, J. Bouwmeester, and A. Povolac, "Design and Validation of an Innovative Data Bus Architecture for CubeSats," *Proceedings of the Reinventing Space Conference 2016*, 2016.

[151] B. Grzesik, T. Baumann, T. Walter, F. Flederer, F. Sittner, E. Dilger, S. Gläsner, J.-L. Kirchler, M. Tedsen, S. Montenegro, and E. Stoll, "InnoCube—A Wireless Satellite Platform to Demonstrate Innovative Technologies," *Aerospace*, vol. 8, no. 5, 2021.

[152] J. Bouwmeester, S. P. van der Linden, A. Povalac, and E. K. Gill, "Towards an innovative electrical interface standard for PocketQubes and CubeSats," *Advances in Space Research*, vol. 62, no. 12, pp. 3423–3437, 2018.

[153] J. Bouwmeester, M. Langer, and E. Gill, "Survey on the implementation and reliability of CubeSat electrical bus interfaces," *CEAS Space Journal*, vol. 9, no. 2, pp. 163–173, 2017.

[154] A. Albalooshi, A.-H. M. Jallad, and P. R. Marpu, "Fault Analysis and Mitigation Techniques of the I2C Bus for Nanosatellite Missions," *IEEE Access*, vol. 11, pp. 34709–34717, 2023.

[155] L. Kepko, L. S. Soto, C. Clagett, B. Azimi, A. Cudmore, J. A. Marshall, D. L. Berry, and S. R. Starin, "Dellingr: Reliability Lessons Learned from On-Orbit," in *Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites*, 2018.

[156] J. Guo, J. Bouwmeester, and E. Gill, "In-orbit results of Delfi-n3Xt: Lessons learned and move forward," *Acta Astronautica*, vol. 121, pp. 39–50, 2016.

[157] M. Koller, L. Bötsch-Zavřel, M. Eggert, M. Fugmann, C. Holeczek, M. Kranz, M. Lengowski, T. Löffler, L.-M. Loidold, P. Maier, J. Meier, U. Mohr, R. Müller, A. Pahler, S. Pätschke, R. Schweigert, D. Starzmann, M. Steinert, M. Zietz, and S. Klinkner, "Lessons Learned and First Results of the E-Band CubeSat EIVE," 2023. preprint doi: 10.21203/rs.3.rs-3748010/v1.

[158] A. Scholz, T.-H. Hsiao, J.-N. Juang, and C. Cherciu, "Open source implementation of ECSS CAN bus protocol for CubeSats," *Advances in Space Research*, vol. 62, no. 12, pp. 3438–3448, 2018.

[159] S. C. Clancy, M. D. Chase, A. Yarlagadda, M. D. Starch, and J. P. Lux, "SpaceWire as a CubeSat Instrument Interface," in *Proceedings of the 8th International SpaceWire Conference*, 2018.

[160] D. Miranda, M. Ferreira, F. Kucinskis, and D. Mccomas, "A Comparative Survey on Flight Software Frameworks for 'New Space' Nanosatellite Missions," *Journal of Aerospace Technology and Management*, vol. 11, 10 2019.

[161] C. Gonzalez, C. Rojas, A. Bergel, and M. Diaz, "An Architecture-Tracking Approach to Evaluate a Modular and Extensible Flight Software for CubeSat Nanosatellites," *IEEE Access*, vol. PP, pp. 1–1, 07 2019.

[162] T. Farges and U. Levi-Cescutti, "Space Flight Software Systems: An approach to understanding their Open Source Framework Paradigm," tech. rep., SODERN Arianegroup, 2022.

[163] M. Manulis, C. P. Bridges, R. Harrison, V. Sekar, and A. Davis, "Cyber security in New Space: Analysis of threats, key enabling technologies and challenges," *Int. J. Inf. Secur.*, vol. 20, p. 287–311, jun 2021.

[164] J. Curbo and G. Falco, "A research agenda for space flight software security," in *2023 IEEE 9th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, pp. 68–77, 2023.

[165] OpenSSF, *Scorecard*. GitHub repository. Accessed: June 16, 2024. [Online]. Available: https://github.com/ossf/scorecard?tab=readme-ov-file#what-is-scorecard.

[166] T. Vladimirova, R. Banu, and M. N. Sweeting, "On-Board Security Services in Small Satellites," in *2005 MAPLD International Conference*, 2006.

[167] J. Pavur and I. Martinovic, "Building a launchpad for satellite cyber-security research: lessons from 60 years of spaceflight," *Journal of Cybersecurity*, vol. 8, p. tyac008, 06 2022.

[168] J. Willbold, M. Schloegel, M. Vögele, M. Gerhardt, T. Holz, and A. Abbasi, "Space Odyssey: An Experimental Software Security Analysis of Satellites," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1–19, 2023.

[169] NASA, *core Flight System*. Accessed: June 16, 2024. [Online]. Available: https://cfs.gsfc.nasa.gov/Introduction.html.

[170] NASA, *Core Flight System - BUNDLE*. GitHub repository. Accessed: June 16, 2024. [Online]. Available: https://github.com/nasa/cfs.

[171] A. Cudmore, "Porting the Core Flight System to the Dellingr Cubesat," in *Flight Software Workshop 2017*, 2017.

[172] eoPortal, *Dellingr CubeSat Demonstration Mission*. Accessed: June 16, 2024. [Online]. Available: https://www.eoportal.org/satellite-missions/dellingr#space-and-hardware-components.

[173] Kubos Corporation, *KubOS 1.21.0+13*. Accessed: June 16, 2024. [Online]. Available: https://kubos-preservation-group.github.io/kubos/index.html.

[174] Kubos Corporation, *kubOS*. GitHub repository. Accessed: June 16, 2024. [Online]. Available: https://github.com/kubos/kubos/.

[175] ESA, *NanoSat MO Framework*. Accessed: June 16, 2024. [Online]. Available: https://nanosat-mo-framework.readthedocs.io/en/latest/index.html#.

[176] CCSDS, *Mission Operations Services Concept, Green Book, CCSDS 520.0-G-3*, 2010. Accessed: June 16, 2024. [Online]. Available: https://public.ccsds.org/Pubs/520x0g3.pdf.

[177] C. Coelho, O. Koudelka, and M. Merri, "CCSDS Mission Operations Services on OPS-SAT," in *10th IAA Symposium on Small Satellites for Earth Observation*, 2015.

[178] C. Coelho, O. Koudelka, and M. Merri, "NanoSat MO Framework: Achieving On-board Software Portability," in *SpaceOps 2016 Conference*, 2016.

[179] ESA, *NanoSat MO Framework*. GitHub repository. Accessed: June 16, 2024. [Online]. Available: https://github.com/esa/nanosat-mo-framework.

[180] A. Marin, C. Coelho, F. Deconinck, I. Babkina, N. Longépé, and M. Pastena, "Φ-Sat-2: Onboard AI Apps for Earth Observation," in *Space and Artificial Intelligence 2021, Online Conference*, 2021.

[181] J.-L. Terraillon, "SAVOIR: Reusing specifications to improve the way we deliver avionics," in *Embedded Real Time Software and Systems (ERTS2012)*, (Toulouse, France), Feb 2012.

[182] CCSDS application support services working group, *SAVOIR as a CCSDS Onboard Reference Architecture*. Accessed: June 16, 2024. [Online]. Available: https://cwe.ccsds.org/fm/Lists/Projects/DispForm.aspx?ID=547.

[183] P&P Software GmbH, *The CORDET Framework*. Accessed: June 16, 2024. [Online]. Available: https://pnp-software.com/cordetfw/about.html.

[184] P&P Software GmbH, *CORDET Framework, C2 Implementation*. GitHub repository. Accessed: June 16, 2024. [Online]. Available: https://github.com/pnp-software/cordetfw.

[185] Amazon Web Services, *FreeRTOS Documentation*. Accessed: June 16, 2024. [Online]. Available: https://www.freertos.org/Documentation/RTOS_book.html.

[186] M. Qutqut, A. Al-Sakran, F. Almasalha, and H. Hassanein, "Comprehensive Survey of the IoT Open Source OSs," *IET Wireless Sensor Systems*, vol. 8, 10 2018.

[187] A. Yahyaabadi, M. Driedger, V. Parthasarathy, R. Sahani, A. Carvey, T. Rahman, V. Platero, J. Campos, and P. Ferguson, "ManitobaSat-1: Making Space for Innovation," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2019.

[188] I. Latachi, T.-e. Rachidi, M. Karim, and A. Hanafi, "Reusable and Reliable Flight-Control Software for a Fail-Safe and Cost-Efficient Cubesat Mission: Design and Implementation," *Aerospace*, vol. 7, p. 146, 10 2020.

[189] M. Doyle, A. Gloster, C. O'Toole, J. Mangan, D. Murphy, R. Dunwoody, M. Emam, J. Erkal, J. R. Flanaghan, G. Fontanesi, F. Okosun, R. R. Nair, J. Reilly, L. Salmon, D. Sherwin, J. W. Thompson, S. Walsh, D. de Faoite, U. Javaid, S. McBreen, D. J. McKeown, D. O'Callaghan, W. O'Connor, K. Stanton, A. Ulyanov, R. Wall, and L. Hanlon, "Flight Software Development for the EIRSAT-1 mission," *Proceedings of the 3rd Symposium on Space Educational Activities*, 2020.

[190] RTEMS Project, *RTEMS Documentation Project*. Accessed: June 16, 2024. [Online]. Available: https://docs.rtems.org/.

[191] F. Nicodemos, O. Saotome, and G. Lima, "RTEMS Core Analysis for Space Applications," in *2013 III Brazilian Symposium on Computing Systems Engineering*, pp. 125–130, 2013.

[192] G. Bloom and J. Sherrill, "Scheduling and thread management with RTEMS," *SIGBED Rev.*, vol. 11, pp. 20–25, 2014.

[193] S. Montenegro and F. Dannemann, "RODOS - Real Time Kernel Design for Dependability," *DASIA 2009 - DAta Systems in Aerospace*, vol. 669, p. 66, 2009.

[194] F. Dannemann and S. Montenegro, "Embedded Logging Framework for Spacecrafts," in *DASIA 2013 - DAta Systems In Aerospace* (L. Ouwehand, ed.), vol. 720 of *ESA Special Publication*, p. 52, Aug. 2013.

[195] C. Lenzen, M. Wörle, T. Göttfert, F. Mrowka, and M. Wickler, "Onboard Planning and Scheduling Autonomy within the Scope of the FireBird Mission," in *SpaceOps 2014 Conference*, 05 2014.

[196] K. Götz, K. Schwenk, and F. Huber, "VIMOS - Modular Commanding and Execution Framework for Onboard Remote Sensing Applications," in *Conference on Big Data from Space (BiDS'14)*, 01 2014.

[197] Wind River Systems, *VxWorks: Real-Time Operat-*

ing System for the Intelligent Edge. Accessed: June 16, 2024. [Online]. Available: https://www.windriver.com/products/vxworks.

[198] P. Hambarde, R. Varma, and S. Jha, "The Survey of Real Time Operating System: RTOS," in *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, pp. 34–39, 2014.

[199] eoPortal, *PEARL CubeSat Bus Initiative*. Accessed: June 16, 2024. [Online]. Available: https://www.eoportal.org/other-space-activities/pearl#pearlsoft-flight-software.

[200] H. Leppinen, "Current use of linux in spacecraft flight software," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 10, pp. 4–13, 2017.

[201] S. Flagg, T. Bleier, C. Dunson, J. Doering, L. DeMartini, P. A. Clarke, L. Franklin, J. Seelbach, J. Flagg, M. Klenk, V. Safradin, J. Cutler, A. Lorenz, and E. Tapio, "Using Nanosats as a Proof of Concept for Space Science Missions: QuakeSat as an Operational Example," in *18th Annual AIAA/USU Conference on Small Satellites*, 2004.

[202] M. Schmidt and K. Schilling, "An Extensible On-board Data Handling Software Platform for Pico Satellites," *Acta Astronautica*, vol. 63, pp. 1299–1304, 2008.

[203] H. Leppinen, P. Niemelä, N. Silva, H. Sanmark, H. Forstén, A. Yanes, R. Modrzewski, A. Kestilä, and J. Praks, "Developing a Linux-Based Nanosatellite On-Board Computer: Flight Results from the Aalto-1 Mission," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 1, pp. 4–14, 2019.

[204] A. Guillen, W. Attai, K. Oyadomari, C. Priscal, R. Shimmin, O. Gazulla, and J. Wolfe, "PhoneSat in-flight experience results," in *2014 Small Satellites Systems and Services Symposium*, 05 2014.

[205] S. Kenyon, C. Bridges, D. Liddle, R. Dyer, J. Parsons, D. Feltham, R. Taylor, D. Mellor, A. Schofield, and R. Linehan, "STRaND-1: Use of a 500 Smartphone as the Central Avionics of a Nanosatellite," in *62nd International Astronautical Congress, Cape Town*, 10 2011.

[206] A. Russo and G. Lax, "Using Artificial Intelligence for Space Challenges: A Survey," *Applied Sciences*, vol. 12, no. 10, 2022.

[207] R. Horne, S. Mauw, A. Mizera, A. Stemper, and J. Thoemel, "Anomaly Detection Using Deep Learning Respecting the Resources on Board a CubeSat," *Journal of Aerospace Information Systems*, vol. 20, no. 12, pp. 859–872, 2023.

[208] D. A. Zeleke and H.-D. Kim, "A new strategy of satellite autonomy with machine learning for efficient resource utilization of a standard performance cubesat," *Aerospace*, vol. 10, no. 1, 2023.

[209] E. Gill, P. Sundaramoorthy, J. Bouwmeester, B. Zandbergen, and R. Reinhard, "Formation flying within a constellation of nano-satellites: The QB50 mission," *Acta Astronautica*, vol. 82, no. 1, pp. 110–117, 2013. 6th International Workshop on Satellite Constellation and Formation Flying.

[210] A. Kak and I. F. Akyildiz, "Large-Scale Constellation Design for the Internet of Space Things/CubeSats," in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2019.

[211] NASA Jet Propulsion Laboratory, *Sun Radio Interferometer Space Experiment (SunRISE)*. Accessed: June 16, 2024. [Online]. Available: https://www.jpl.nasa.gov/missions/sun-radio-interferometer-space-experiment/.

[212] NASA, *2020 NASA Technology Taxonomy*, 2020. Accessed: June 16, 2024. [Online]. Available: https://www.nasa.gov/otps/2020-nasa-technology-taxonomy/.

[213] Space Micro, *Proton-600k^{TM} Multi-Core Computer*. Accessed: June 16, 2024. [Online]. Available: https://www.spacemicro.com/products/digital-systems.html.

[214] G. Giuffrida, L. Fanucci, G. Meoni, M. Batic, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, p. 3125567, Jan. 2022.

[215] I. V. Belokonov, A. V. Kramlikh, and M. E. Melnik, "Application of artificial intelligence technology in the nanosatellite attitude determination problem," *IOP Conference Series: Materials Science and Engineering*, vol. 984, p. 012036, nov 2020.

[216] S. Ibrahim, A. Ahmed, M. Zeidan, and I. Ziedan, "Machine Learning Techniques for Satellite Fault Diagnosis," *Ain Shams Engineering Journal*, vol. 11, pp. 45–56, 08 2019.

[217] X. He and R. E. Geer, "High Total-Dose Proton Radiation Tolerance in TiN/HfO2/TiN ReRAM Devices," *MRS Proceedings*, vol. 1430, 2012.

[218] D. Evans, "OPS-SAT: FDIR Design on a Mission that Expects Bugs - and Lots of Them," in *14th International Conference on Space Operations*, 05 2016.

[219] J. Willis, P. Walton, D. Wilde, and D. Long, "Miniaturized Solutions for CubeSat Servicing and Safety Requirements," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 1, no. 1, pp. 3–9, 2020.

[220] A. Utter, M. P. Zakrzewski, A. C. Keene, S. Dietrich, S. Lin, E. J. McDonald, N. Whitehair, and J. X. Zheng, "SatCat5: A Low-Power, Mixed-Media Ethernet Network for Smallsats," in *34th Annual Small Satellite Conference*, 2020.

[221] D. Ohlsson, H. Löfgren, E. Vinterhav, and S. Strålsjö, "Enabling advanced missions on small platforms through designing cost effective SpaceWire-based avionics solutions in the CubeSat form factor: SpaceWire missions and applications, short paper," in

2016 International SpaceWire Conference, pp. 1–5, 2016.

[222] Alén Space, *TRISKEL: OBC, TTC and OBSW in a single module*. Accessed: June 16, 2024. [Online]. Available: https://products.alen.space/products/triskel/.

[223] EnduroSat, *On Board Computer*. Accessed: June 16, 2024. [Online]. Available: https://www.endurosat.com/cubesat-store/cubesat-obc/onboard-computer-obc/.

[224] GomSpace, *NanoMind A3200*. Accessed: June 16, 2024. [Online]. Available: https://gomspace.com/shop/subsystems/command-and-data-handling/nanomind-a3200.aspx.

[225] NanoAvionics, *CubeSat OBC - Main Bus Unit SatBus 3C2*. Accessed: June 16, 2024. [Online]. Available: https://nanoavionics.com/cubesat-components/cubesat-on-board-computer-main-bus-unit-satbus-3c2/.

[226] Spacemanic, *Deep Thought On-board Computer*. Accessed: June 16, 2024. [Online]. Available https://www.spacemanic.com/deep-thought-onboard-computer/.

[227] I. Sünter, *Design and characterisation of subsystems and software for ESTCube-1 nanosatellite*. PhD thesis, Tartu University, 2019.

[228] T. Kuwahara, F. Böhringer, A. Falke, J. Eickhoff, F. Huber, and H.-P. Röser, "FPGA-based operational concept and payload data processing for the Flying Laptop satellite," *Acta Astronautica*, vol. 65, no. 11, pp. 1616–1627, 2009.

[229] P. Harikrishnan, "Deterministic COTS based OBC for high performance and mixed criticality applications," in *ADCSS 2022 Newton Conference Center, ESTEC*, 2022.

[230] eOPortal, *INTUITION-1 Mission*. Accessed: June 16, 2024. [Online]. Available: https://www.eoportal.org/satellite-missions/intuition-1#performance-specifications.

[231] KP Labs, *Antelope*. Accessed: June 16, 2024. [Online]. Available: https://kplabs.space/antelope/.

[232] Space Inventor, *Z7000-P4*. Accessed: June 16, 2024. [Online]. Available: https://space-inventor.com/modules/z7000.

[233] Space Micro, *Proton-400k™ Single Board Computer*. Accessed: June 16, 2024. [Online]. Available: https://www.spacemicro.com/products/digital-systems.html.

[234] eoPortal, *Lunar IceCube*. Accessed: June 16, 2024. [Online]. Available: https://www.eoportal.org/satellite-missions/lunar-icecube.

[235] A. Pawlitzki, F. Steinmetz, and T. A. S. Germany, "multiMIND – High Performance Processing System for Robust Newspace Payloads | Thales Alenia Space Germany," in *2nd European Workshop on On-Board Data Processing (OBDP2021)*, June 2021.

[236] NASA, *SpaceCube Flight Processor Card Family*. Accessed: June 16, 2024. [Online]. Available: https://spacecube.nasa.gov/.

[237] Ubotica Technologies, *CogniSAT-XE1*. Accessed: June 16, 2024. [Online]. Available: https://ubotica.com/product/cognisat-xe1-product-overview/.

ANGELA CRATERE was awarded a Master's degree in Astrophysics and Cosmology from the University of Bologna in 2021. Since 2022 she has been working as a PhD student at the Department of Electrical and Information Engineering at the Polytechnic University of Bari. Her project aims to design and develop C&DH subsystems for small satellites, in particular for CubeSat applications. Throughout her career, she has had various research interests, ranging from gravitational wave astrophysics to onboard microelectronic technologies for space systems. Currently, her main research interests and areas of investigation concern high-performance and high-reliability embedded systems for onboard data processing, with a particular emphasis on the use of ML techniques for nanosatellite onboard image processing.

LEANDRO GAGLIARDI is a doctoral candidate at National University of San Martin (UNSAM), focusing on Applied Sciences and Engineering. He holds a Bachelor's degree in Electronics Engineering from National University of La Matanza (UNLaM). Leandro's expertise includes developing electronics for satellite payloads. He has contributed to the development of small satellite's payloads and onboard computers for small satellites. Leandro's research interests lie in testing and validating electronic components for space applications. He has co-authored a publication in IEEE Embedded Systems Letters.

GABRIEL A. SANCA is currently the Head of the Electronic Engineering program and Professor at the School of Science and Technology (ECyT) at the National University of San Martin (UNSAM). He also works at the Nanoelectronic Integration Laboratory of the ECyT-UNSAM. His research interests include the design of integrated circuits, micro/nano devices, and applications in hostile environments, with a particular focus on space systems. Gabriel holds a Bachelor's degree in Electronic Engineering from the School of Engineering at the University of Buenos Aires (FIUBA), where he completed his thesis on "Development of an Interoperable Design Kit and a Set of Standard Open Cells for a Scalable CMOS Process" at the Microelectronics Laboratory. He also earned a Ph.D. in Applied Sciences and Engineering from the ECyT-UNSAM, with a thesis on "Integration of RS Devices with CMOS Technology for Hostile Environment Applications".

FEDERICO GOLMAR is currently the Dean and Professor at the School of Science and Technology (ECyT) at the National University of San Martin (UNSAM). He earned his Ph.D. in Engineering from the University of Buenos Aires, Argentina. His research spans the materials science, nanotechnology, spintronics, and microelectronics. Golmar has made contributions to understanding and manipulating material properties at the nanoscale. He has authored numerous publications and holds several positions in various scientific organizations. At ECyT-UNSAM, Golmar leads the Nanoelectronics Integration Laboratory.

FRANCESCO DELL'OLIO (Senior Member, IEEE) received the M.S. degree in electronic engineering and the Ph.D. degree from the Polytechnic University of Bari, Bari, Italy, in 2005 and 2010, respectively. In 2015, he joined the Department of Electrical and Information Engineering, Polytechnic University of Bari, as an Assistant Professor, where he was promoted to an Associate Professor in 2022. He is the coauthor of two books, published by Springer and World Scientific, and more than 60 articles in international peer-reviewed journals. He has been involved in several research projects funded by the Italian Ministry of University and Research, the European Space Agency, the Italian Space Agency, and industrial companies, some of which involved taking on the role of principal investigator. His research focuses on silicon photonics and nanophotonics, with particular regard to modeling, design, and characterization of devices and integrated circuits for telecommunications and sensing. He has of late branched into miniature sensor-based embedded systems.

Mr. Dell'Olio is a regular member of organizing committees and program committees for international conferences, including Conference on Lasers and Electro-Optics (CLEO), Society of Photo-Optical Instru mentation Engineers (SPIE) Photonics West, and the IEEE Photonics Conference.

TABLE 3. Comparison of memories used in CubeSat OBCs. Data and information from [23, 32, 63, 64, 71, 217].

Parameters	SRAM	DRAM	EEPROM	Flash	MRAM	FeRAM	RRAM	PCM
Technology	Data stored as electric charges in the nodes of bistable latching circuits (flip-flop) (single-bit storage)	Data stored as electric charges on gate capacitors (multibit storage)	Data stored as electric charges on a floating gate (multibit storage)		Data stored as magnetic charges in a magnetic tunnel junction (MTJ)	Data stored as electric charges in ferroelectric dielectric capacitors (multibit storage)	Data storage through resistance variation of a solid-state dielectric material induced by an external voltage pulse (multibit storage)	Data storage through heat-induced state changes of a chalcogenide glass (multibit storage)
Non-Volatile	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Supply Voltage	3.3 - 5 V	3.3 V	1.7 - 5.5 V	3.3 - 5 V	3.3 V	3.3 V	3.3 V	3.3 V
Power consumption	500 mW	300 mW	44 mW	30 mW	900 mW	270 mW	-	-
Read time	0.1 - 0.3 ns	< 10 ns	4 ms	50 ns (NAND) 10 ns (NOR)	20 ns	45-80 ns	10 ns	20-50 ns
Write time	0.1 - 0.3 ns	< 10 ns	5 ms	1 µs - 10 ms (NAND) 0.1 ms - 1 ms (NOR)	20 ns	50 ns	10 ns	20-50 ns
Endurance [a]	Infinite	Infinite	10^6 cycles	10^5 cycles	10^{13} cycles	10^{13} cycles	10^6-10^{12}cycles	10^{13} cycles
Data Retention (@70°C) [b]	-	-	10 years	10 years	10 years	10 years	10 years	10 years
Temperature range	MIL-STD [c]	Industrial [d]	Industrial	Commercial [e]	MIL-STD	MIL-STD	MIL-STD	MIL-STD
TID tolerance	50 krad - 1 Mrad	50 krad	NA	30 krad	1 Mrad	1 Mrad	up to 5 Grad	1 Mrad
SEU/SEL immunity	Low	Low	Mid-Low	Mid-Low	High	High	High	High
Cost/Mb ($)	5	0.75	1.5	0.0002 (NAND) - 0.01 (NOR)	1.5	10	-	0.05
OBC application	Working memory	Working memory	Boot memory; OS image storage; Safe-guard memory; Configuration storage	Boot memory - OS image storage (NOR); Mass memory for data storage (NAND)	External SRAM; Configuration/critical parameters storage; Mass memory for data storage	External SRAM; Configuration/critical parameters storage; Mass memory for data storage	External SRAM; Configuration/critical parameters storage; Mass memory for data storage	External SRAM; Configuration/critical parameters storage; Mass memory for data storage

[a] The number of times a memory device can perform the write/erase cycle before failing to read the correct data.

[b] The ability of a memory bit to keep the state of data for long periods of time, regardless of whether the device is switched on or off.

[c] -40°C to +125°C.

[d] -40° to 85° C.

[e] 0 to 70 C.

TABLE 4. Common communication interfaces in CubeSat OBCs. Data and information from [1, 145, 150, 152]

Bus interface	Communication Type	Topology	Data rate	Power consumption	Possible OBC applications
I2C	Synchronous Serial	Multipoint (half-duplex)	0.1 -0.4 Mbps	50 - 150 mW	System data bus for communication between spacecraft subsystems [156]. Low-speed, short-distance communication between the OBC, sensors and actuators [157, 171, 218].
CAN	Asynchronous Serial	Multipoint (half-duplex)	1 Mbps, up to 5 Mbps (CAN FD)	150 - 300 mW	System data bus for communication between spacecraft subsystems [157, 158].
RS-485	Asynchronous Serial	Multipoint (half-duplex, full-duplex)	up to 10 Mbps	10 - 100 mW	System data bus for communication between spacecraft subsystems [150, 152].
SPI	Synchronous Serial	Point-to-point (full-duplex)	1, up to 20 Mbps	10 - 100 mW	High-speed data transfer between the OBC, sensors and actuators [157].
USB	Asynchronous/ Synchronous Serial	Point-to-point (half-duplex)	1.5 - 480 Mbps, up to 20 Gbps (USB 3.2)	150 - 700 mW	High data rate bus to transfer payload data [150]. Miniaturized interface for servicing and safe spacecraft handling during preflight and launch operations [219].
Ethernet	Asynchronous Serial	Point-to-point (half-duplex, full-duplex)	100 - 1000 Mbps	150 mW - 1 W	High-speed data networks within the spacecraft, enabling fast and effective data exchange between the OBC, payload processing units, and scientific instruments [220].
SpaceWire	Asynchronous Serial	Point-to-point (full-duplex)	2 - 400 Mbps	10 mW - 1 W	OBC system data bus [100, 221]. Payload data interfaces [159]. Interconnection between the various OBC computing nodes, e.g. between central processing nodes and the external supervisor in both SoM [138] and motherboard-daughterboards [139] configurations.
RS-232/422	Asynchronous Serial	Point-to-point (full-duplex)	up to 10 Mbps	10 - 100 mW	Point-to-point payload data bus for demanding payloads [150, 157].

TABLE 5. Representative list of CubeSat OBCs covered in the analysis.

OBC	Processing Unit	Memories	Operating System	Power Consumption	Radiation Performance	Fault-tolerant techniques	Status	Application
µC-based architectures								
Commercial OBCs								
TRISKEL (OBC+TTC) by Alén Space [222]	32-bit ARM Cortex-M7 µC	2 MB Flash for program storage; EEPROM for configuration storage; 1.4 MB internal SRAM; 4 Mb MRAM as external RAM; 1 Gb NAND Flash for data storage; 2 x µSD card slot for massive storage	FreeRTOS	OBC peak: 6 W; TTC peak: 6 W	COTS	Hardware watchdog and reset circuit; shielding	Flight heritage	Platform control + TT&C management + optional internal GNSS module for LEO satellites
OBC by EnduroSat [223]	32-bit ARM Cortex-M7 µC (STMicroelectronics STM32)	2 MB program storage; 1 MB internal SRAM; 8 Mb MRAM as external RAM; 2 MB NAND Flash for data storage; 1 x µSD card slot	FreeRTOS	1.5 W (peak)	COTS	NA	Flight heritage	Platform control + ADCS + optional internal GNSS module for LEO satellites
NanoMind A3200 by GomSpace [224]	32-bit RISC µC (Atmel AVR32)	32 MB SRAM; 32 kB FRAM for configuration storage; 128 MB Flash for data storage	FreeRTOS	0.9 W	COTS	NA	Flight heritage since 2015 in GOMX-3/4/5, Dellingr, PRETTY, OPS-Sat	Platform control + ADCS for LEO satellites
OBC by IMT [54]	32-bit MIPS µC (Microchip Technology PIC32MZ)	16 MB SRAM; 64 MB NOR flash for housekeeping data + 8 GB NAND flash for for payload data	FreeROTS	300 mW (normal operation)	COTS (< 15 kRad; Si)	Watchdog; TMR for memories; anti latch-up circuits	TRL 9	Platform control + ACDS + On-Board Camera Interface for LEO EO missions
SatBus 3C2 by NanoAvionics [225]	32-bit ARM Cortex-M7 µC (STM32)	1 MB integrated RAM; 2 MB integrated flash memory; 256 MB external NOR flash memory for data storage; 2 × 512 kB FRAM for frequently changing data storage; µSD card support	NA	NA	COTS	NA	Flight heritage in dozen of missions since 2019	Platform control + ADCS + redundant UHF communication system for LEO satellites
OBC-P4 by Space Inventor [116]	32-bit ARM Cortex-M7 µC (Microchip Technology SAME70)	384 kB SRAM; 64 GB embedded MultiMediaCard (eMMC) for mass storage; 32 kB FRAM as application memory (for storage critical parameters)	FreeRTOS	NA	COTS	Hot/cold redundancy (four independent ARM Cortex-M7 modules, each with separate power supply, interfacing, and storage); shielding	TRL 9	Platform control + TT& C management + ACDS + GNSS module + payload data storage for LEO and GEO satellites
Eddie and **Deep Thought** by Spacemanic [82, 226]	**Eddie**: 16-bit RISC µC (TI MSP430) **Deep Thought**: 32- bit Cortex-M7 µC (Microchip Technology SAMV71)	**Eddie**: 256 Kb internal FRAM for code storage; 24 Mb external FRAM for data storage; **Deep Thought**: 2048 KB flash memory; 384 KB multi-port SRAM; 128 MB flash storage	FreeRTOS	100 mW (average)	COTS	Hardware watchdog; rad-tolerant data storage (FRAM and flash technologies); shielding	Flight heritage	Platform control + TT&C management for LEO satellites
Mission-specific OBCs								
ESTCube-1's OBC by University of Tartu [227]	Two 32-bit ARM Cortex-M7 µCs (STM32)	µC-embedded memories; 128 KiB FRAM; 5 x 256 KiB FRAMs; 3 NOR Flash memories	FreeRTOS	NA	COTS	µCs in cold redundancy; watchdog; ECCs; overcurrent protection; shielding	Flight heritage in ESTCube-1 in 2013	Platform control + TT&C management + ACDS calculations for LEO ESTCube-1 mission
FloripaSat's OBC by UFSC [a] and IFSC [b] [81]	16-bit RISC µC (MSP430)	60kB Flash and 2kB RAM µC internal memories + SD card	Not specified RTOS	NA	COTS	NA	Flight heritage in FloripaSat-1 in 2019	Housekeeping data management (storage + sending to ground stations) + telecommands decoding + platform control for LEO FloripaSat mission
Aalto-1's OBC by Aalto University [78]	Two 32-bit ARM Tumb µCs (Atmel AT91RM9200)	256 Mb SRAM; NOR Flash memory to store boot-loaders; NOR Flash to store kernel images; NAND Flash to store file systems; 32 GB µSD for data storage	Linux	200 mW	COTS	µC in cold redundancy; other information NA	Flight heritage in Aalto-1 in 2017	Platform control + TT&C and payload data managment for LEO Aalto-1 mission
LabOSat-02 by LabOSat [c] [75]	32-bit RISC ARM Cortex-R4F µC	4 Mb FRAMs	FreeRTOS	<200 mW	COTS	External Watchdog and Voltage Supervisor; backup copy of firmware; rad-tolerant memories	TRL 6	General purpose and payload controller

[a] Federal University of Santa Catarina.

[b] Federal Institute of Santa Catarina.

[c] Laboratory-on-a-satellite collaboration (Universidad de San Martín-CONICET)

OBC	Processing Unit	Memories	Operating System	Power Consumption	Radiation Performance	Fault-tolerant techniques	Status	Application
FPGA-based architectures								
Commercial OBCs								
Sirius OBC by AAC Clyde Space [100]	32-bit LEON3FT fault-tolerant softcore processor	64 MB SRAM; 16 kB non-volatile RAM; 2 GB NAND Flash for data storage	RTEMS	1.3 W (normal operation)	Rad-tolerant (qualified > 30 krad, Si)	Rad-tolerant processor; TMR; EDAC-protected memories	TRL 9	Platform control + TT&C management + housekeeping and payload data storage for LEO satellites
NANOhpm-OBC by Sky Labs [99]	32-bit RISC-V softcore processor (in Microchip Technology PolarFire FPGA)	256 MB DDR3 (SRAM); 2 GB NAND Flash for mass storage; 1 MB MRAM for TM storage	RTEMS, FreeRTOS, SafeRTOS	5 W	Rad-tolerant (> 30 krad, Si)	SEU-immune FPGA fabric; memory redundancy; EDAC-protected memories	TRL 9	Platform control + TM storage + Integrated GNSS module for LEO satellites
Mission-specific OBCs								
Flying Laptop's OBC by IRS [d] [228]	4 SRAM FPGA-based CPN [e] & 1 Flash FPGA-based CDV [f]	4 x SRAM for intermediate results; 2 x DDR SRAM for application data or as program memory; 15 Gb NAND Flash memory for mass storage	No OS; Handel-C-based control algorithm	20 W	COTS	Multi-module redundancy; TMR in FPGA combinational logic; SEL protection circuit	Flight heritage since 2017	TM generation + TC/TM management + ACDS + housekeeping routines for LEO satellites
CubeSpace v3.0 mini by NASA GSFC [135]	FPGA (Kintex UltraScale FPGA) as HPN [g] & rad-tolerant FPG (RT ProASIC 3 FPGA) as external supervisor	2 GB DDR3 SDRAM as volatile memory resources; 2 x 16 GB NAND flash memories for OS boot images storage and data storage	SpaceCube Linux	< 10 W	Rad-tolerant (> 50 krad)	Rad-tolerand FPGA for monitoring and scrubbing the HPN; ECC memories	Tested in the SCENIC experiment in 2023	Payload manager and data processing unit; adaptive processing system for AI applications, communication, navigation, and cybersecurity
hybrid and SoC-based architectures								
Commercial OBCs								
KRYTEN-M3 by AAC Clyde Space [115]	CPU+FPGA SoC (SmartFusion 2 SoC)	16 MB MRAM for code storage and execution; 4 GB flash memory for bulk data storage	RTEMS	400 mW - 1 W	COTS/ rad-tolerant (20 krad)	Latch-up protection; EDAC-protected memories; firmware recovery mechanisms	Fligh heritage since 2014	Platform control + optional GNNS module (Kryten-M3-PLUS) for LEO satellites
FERMI by Argotech [91]	SoC-based core board (incorporating a dual-core LEON-3FT SPARC-V8 processor) & FPGA-based aXelerator module	256 Mb SRAM; 25 Mb EEPROM; 16 GB ECC-corrected mass memory	RTEMS	5 W (tipical)	Rad-hard (100 krad Si)	Rad-hard processor; rad-hard FPGA; EDAC-protected memories; sheliding	Flight heritage in LiciaCube ArgoMoon	Platform control + TT&C + TM managment + FDIR services for deep space satellites (the FPGA board can be used to implement advanced payload management)
CHICS OBC by Evoleo Technologies and Airbus Defence and Space [138, 229]	CPU+FPGA+GPU SoC (Xilinx Zynq UltraScale+ XQ MPSoC) & PolarFire FPGA as external supervisor in a single board	32 GB NAND Flash for platform data storage; up to 1 TB NAND Flash for payload data storage	FreeRTOS, Linux	TBD	Rad-tolerant COTS	TMR for memories; external PolarFire FPGA supervisor for critical faults; LCLs; data exchange buffers; software FDIR services;	TRL 6 in 2022	Payload processing core for LEO satellites; AI for non-mission critical on-board data processing
ABACUS by Gauss [51]	16-bit RISC µC (MSP430) & FPGA (Xilinx Spartan-3E) in a single board	2 MB SRAM memory; 2 x 16 MB NOR Flash memories	FreeRTOS	50 mW (µC on; FPGA off)	COTS	DMR of processing cores (with varying implementation modes, i.e. Master/Slave or multi-Master); TMR of FPGA configuration codes	Flight heritage in UniCubeSat-GG, UniSat-5/6/7, TigriSat, Serpens	Platform monitoring + ACDS for LEO satellites
CFC-400 by Innoflight [117]	High-performance SoC-based board (Xilinx CPU + FPGA MPSoC CG) & High-reliability FPGA-based board (MicroSemi rad-tolerant FPGA with LEON3FT softcore CPU) as external supervisor	2 GB DDR4 as CPU volatile memory; 256 MB DDR3 as FPGA volatile memory; 16 GB NAND flash and 8 MB MRAM as non-volatile memories	Linux, VxWorks	0.6 - 12 W	Rad-tolerant (30 krad), SEL immune	Rad-tolerant FPGA and memories (MRAM); EDAC-protected memories; power supply protection circuit	Flight heritage since 2021	High-reliability C&DH platform + High-performance payload data processing unit + High-Assurance End Cryptographic Unit (ECU) with built-in cyber protection for LEO and GEO satellites
Leopard by KP Lab [118]	CPU+FPGA+GPU (Xilinx Zynq UltraScale+ MPSoC EG)	4-16 GiB DDR4; 4-16 GiB flash-based file system storage; 2x256 GiB flash-based data storage	Linux	7 W (static power consumption); 10-40 W (for deep learning inference)	COTS	EDAC for memories; optional redundant OBC configuration	Flight heritage onboard the Intuition-1 mission [230] in November 2023	Payload data on-board processing and deep learning algorithms acceleration for LEO EO missions

[d] Institut für RaumfahrtSysteme - Institute of Space Systems - University of Stuttgart.

[e] Central Processing Nodes.

[f] Command Decoder and Voter.

[g] High Performance Node.

OBC	Processing Unit	Memories	Operating System	Power Consumption	Radiation Performance	Fault-tolerant techniques	Status	Application
hybrid and SoC-based architectures								
Commercial OBCs								
Antelope OBC + DPU by KP Lab [231]	**OBC board**: 32-bit ARM Cortex-R5F µC (TI Herkules RM57); **DPU board**: CPU+FPGA+GPU SoC (Xilinx Zynq UltraScale+ MPSoC EG)	**OBC**: 8 or 16 MB MRAM; 64 MB NOR Flash; 4 or 8 GB NAND Flash for data storage; **DPU**: 8 GiB of DDR4; 4 GB of NAND Flash; Optional SSD	**OBC**: KP Lab's Software -Oryx; **DPU**: Linux	< 0.5 W (DPU off); up to 6 W (DPU on)	COTS	Redundancy and EDAC for memories; shielding	It will acquire flight heritage on PW-Sat3 satellite in late 2024	Platform and subsystem monitoring + communication handling + FDIR services for LEO satellites + Optional payload data processing
SE-1 by Spiral Blue [129]	CPU+GPU SoC (Nvidia Jetson Xavier NX)	8-16 GB embedded RAM; 1 TB non volatile memory; SSD support	Linux based	3 W (idle); 20 W (peak)	COTS	NA	Flight heritage on 2023	Payload on-board data processing; AI accelerator
Z700-P4 by Space Inventor [232]	CPU+FPGA SoC (Xilinx Zynq 7030)	256 KB on-chip memory; 1 GB RAM; 16 GB eMMC for mass storage	FreeRTOS, Linux	1.5 - 20 W	COTS	Shielding; other information NA	TRL 9	Payload data processing for LEO and GEO satellites
Proton-400k™ by Space Micro [233]	32-bit dual core processor (NXP P2020) & FPGA as module controller in a single board	2 GB DDR3; 4 Mb boot MRAM; 1 - 4 GB NAND Flash	Linux, VxWorks	8-12 W	Rad-tolerant; Rad-hard (100 krad, Si) option	EDAC for RAM memories; rad-hard flash memories; shielding; watchdog	Flight heritage since 2021; selected for Lunar IceCube mission [234]	Platform control + payload electronics + image/signal processing for LEO, MEO, GEO satellites
Proton-600k™ by Space Micro [213]	Multi-core processor (CPU with 8 fully-isolated ARM Cortex cores and dual-core GPU/VPU) & FPGA as module controller in a single board	2-8 GB DDR4; 128 MB NOR Flash; 64 GB eMMC	Linux, VxWorks, and others	12 W peak power	Rad-hard	ECC memories; Radiation-tolerant CPU building technologies; SEU immunity; Patented H-Core™ SEU and single event functional interrupts (SEFI) mitigation	Developed in 2020; flight heritage information NA	High performance computing payload module for LEO, MEO, GEO satellites
multiMIND processing unit (IRS's EIVE mission) by TAS Germany [137, 235]	CPU+FPGA+GPU SoC (Xilinx Zynq UltraScale+ MPSoC EG) & rad-hard ARM Cortex-M4 µC (Vorago VA41630) as external supervisor in a single board	2 x 128 MB NOR flash memories for code; 512 kB MRAM for configuration; 2 x 16 GB eMMC for storage	Linux	5-12 W (1 W in standby)	COTS + rad-hard supervisor	EDAC and scrubbing for memories; rad-hard watchdog and anti-latch-up supervisor circuit	Flight heritage on the EIVE mission in 2023 [140]	Payload OBC for LEO satellites
SpaceCloud™ iX5 by Unibap [130]	CPU+GPU SoC (AMD Embedded G-Series) & CPU+FPGA SoC (Microsemi SmartFusion2) as external supervisor Optional Myriad-X VPU-based borad.	2 GB DDR3 RAM; 2 x 120 GB SSD for data storage	Linux	10 - 30 W	Rad-tolerant	EDAC-protected memories; shielding; other information NA	Flight heritage on the Wild Ride mission in 2021 (it will fly in the NASA HyTI mission in 2024)	Intelligent payload data processing for EO and interplanetary exploration satellites
Q7s and **Q8s** by Xiphos [113, 114]	**Q7s**: CPU+FPGA SoC (Xilinx Zynq 7020); **Q8s**: CPU+FPGA+GPU SoC (Xilinx Zynq UltraScale+ MPSoC EG); Equipped with a Microsemi ProASIC3 FPGA as module controller in the same board	**Q7s**: 256 MB DDR2 RAM; 2 x 123 MB NOR flash; 2 x 32 GB µSD slots; **Q8s**: 4 GB DDR4 DRAM; 2 x 128 MB NOR flash; 2 x 128 GB eMMC for data storage	Linux, Robot Operating System (ROS)	**Q7s**: scalable, 2 W (typical); **Q8s**: 4 - 25 W	Rad-tolerant (> 25 krad)	Software watchdog; TMR of FPGA configuration codes; FPGA scrubbing; EDAC-protected RAM; Upset and multi-current monitoring; overcurrent protection	TRL 9	Platform control + housekeeping and payload data processing + FDIR services for LEO and GEO satellites

OBC	Processing Unit	Memories	Operating System	Power Consumption	Radiation Performance	Fault-tolerant techniques	Status	Application
hybrid and SoC-based architectures								
Mission-specific OBCs								
ScOSA by DLR [m] [139]	2 CPU+FPGA SoCs (Xilinx Zynq 7020) as HPNs [g] & rad-tolerant LEON3 softcore processor (FPGA-based) as RCN [h]	HPN: Embedded DDR3 RAM as volatile memory and 2 x NAND flash memories; RCN: SDRAM; MRAM to store the boot loader; 2 x NAND flash memories to store FSW application	RTEMS; Linux	NA	Rad-tolerant	Rad-tolerant component; software fault-tolerant techniques: reconfiguration as mitigation for node failures, redundancy for tasks	First flight planned on a 12U CubeSat in late 2024	Platform control + TC/TM management; FDIR services + payload processing unit for LEO satellites
SpaceCube v3.0 by NASA GSFC [132]	CPU+FPGA SoC (Zynq UltraScale+ MPSoC CG) as HPN [g] & FPGA (Kintex UltraScale FPGA) as additional HPN & rad-tolerant FPGA (RTAX FPGA) as external supervisor	3 x 2 GB DDR3 SDRAM as volatile memory resources; 3 x 16 GB NAND flash memories for OS boot images storage and data storage	NA	< 20 W	Rad-tolerant (> 50 krad)	Rad-tolerant FPGA for monitoring and scrubbing the HPNs; ECC memories; rad-hard and space-grade peripherals and power circuitry	Still undergoing development; its predecessors have flown on several NASA missions [236]	Platform control for autonomous operations/ robotic servicing and mission-critical computing; payload OBC (e.g., real-time instrument processing, data reduction and classification) for LEO satellite
AFC (MOCI mission OBC) by SSRL [127]	CPU + GPU SoC-based board (Nvidia Jetson TX2i) & CPU + FPGA SoC-based board (SmartFusion2) as external supervisor	8 GB embedded DDR4; 4 x 256 MB external DDR3; NOR Flash memory	TBD	TBD	COTS	CPU+FPGA SoC used as central control node; TMR for file systems on TX2i SoC; watchdog; SEU-protected memories; shielding; U-boot as boot-loader	Still undergoing development	Secondary OBC to interface with primary avionics and enable AI-based high-performance computing (LEO satellites)
Eye of Things (EoT) payload coprocessor (ϕ-Sat-1 mission) by Ubotica Technologies [214][i]	Intel Movidius Myriad 2 VPU SoC (2 RISC-V LEON softcore processor; 12 vector processors)	2 MB SoC-embedded DRAM	No OS	2 W	Rad-tolerant (50 krad)	Latch-up protection; shielding	Flight heritage in ϕ-Sat-1 in 2020	AI accelerator for EO data processing

[h] Reliable Computing Node.

[i] Now marketed as CogniSAT-XE1 [237]