When Linux Meets Radiation: Analyzing Soft Errors in Linux on COTS SoCs under Proton Irradiation

Saad Memon† ⓘ, Rafal Graczyk§ ⓘ, Tomasz Rajkowski‡ ⓘ, Jan Swakon* ⓘ, Damian Wrobel* ⓘ, Sebastian Kusyk* ⓘ, Seth Roffe+ ⓘ, Mike Papadakis† ⓘ, *Member, IEEE*

† Interdisciplinary Centre for Security Reliability and Trust (SnT), University of Luxembourg
§ Independent Researcher, ‡ National Centre for Nuclear Research (NCBJ), Poland
* Institute of Nuclear Physics Polish Academy of Sciences (IFJ PAN), PL-31342 Krakow, Poland
+ NASA Goddard Space Flight Center, 8800 Greenbelt Rd, Greenbelt, MD, USA 20771

Abstract—The increasing use of Linux on commercial off-the-shelf (COTS) system-on-chip (SoC) in spaceborne computing inherits COTS susceptibility to radiation-induced failures like soft errors. Modern SoCs exacerbate this issue as aggressive transistor scaling reduces critical charge thresholds to induce soft errors and increases radiation effects within densely packed transistors, degrading overall reliability. Linux's monolithic architecture amplifies these risks, as tightly coupled kernel subsystems propagate errors to critical components (e.g., memory management), while limited error-correcting code (ECC) offers minimal mitigation. Furthermore, the lack of public soft error data from irradiation tests on COTS SoCs running Linux hinders reliability improvements. This study evaluates proton irradiation effects (20–50 MeV) on Linux across three COTS SoC architectures: Raspberry Pi Zero 2 W (40 nm CMOS, Cortex-A53), NXP i.MX 8M Plus (14 nm FinFET, Cortex-A53), and OrangeCrab (40 nm FPGA, RISC-V). Irradiation results show the 14 nm FinFET NXP SoC achieved 2–3× longer Linux uptime without ECC memory versus both 40 nm CMOS counterparts, partially due to FinFET's reduced charge collection. Additionally, this work presents the first cross-architecture analysis of soft error-prone Linux kernel components in modern SoCs to develop targeted mitigations. The findings establish foundational data on Linux's soft error sensitivity in COTS SoCs, guiding mission readiness for space applications.

Index Terms—Commercial-off-the-shelf (COTS), Linux operating system, Operating system reliability, Raspberry Pi, RISC-V Processor, Single event functional interrupt (SEFI), Single event upset (SEU), Soft errors, Space OS, System-on-chip (SoC)

I. INTRODUCTION

THe Linux kernel, powering 96% of the top 1 million web servers [1] and over 3.3 billion Android devices [2], is increasingly utilized in spaceborne computers [3]–[7] using commercial-off-the-shelf (COTS) system-on-chip (SoC) platforms. The paradigm shift is due to cost-effectiveness and the rapid deployment cycle that Linux (software) and COTS SoCs (hardware) enable in the NewSpace era [8]–[11] to their radiation-hardened counterparts. NASA's 2020 Ingenuity Mars Helicopter ran Linux on a COTS Snapdragon 801 SoC, completing 72 flights in $\approx$ 33 Earth months, 14 times its initial 5-flight goal [12]. However, modern COTS SoCs, designed for low-radiation terrestrial environments, are inherently susceptible to radiation-induced soft errors (transient errors) [13]–[15]. These transient faults [16], caused by a single bit flip, can trigger immediate Linux crashes [17], such as those caused by NULL pointer dereference. Thus, investigating how the unmodified Linux kernel handles soft errors (without ECC) in COTS SoCs is crucial for identifying the kernel's inherent fault-handling weaknesses and guiding reliability improvements to complement ECCs in spaceborne computers.
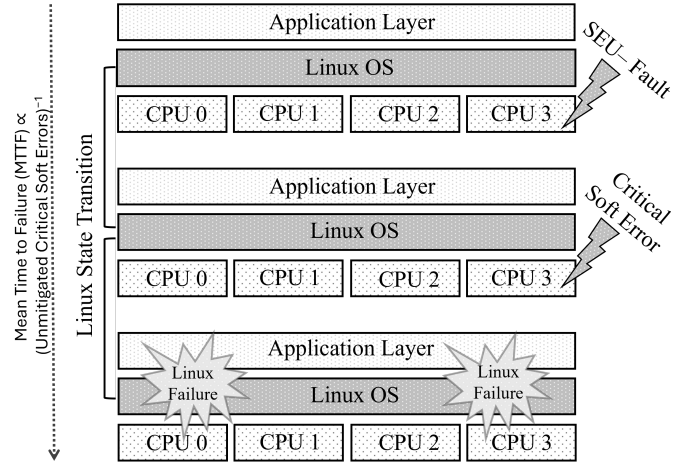
Fig. 1: Unmitigated radiation-induced critical soft error on CPU 3 in a Linux COTS SoC causing Linux crash

SpaceX's Starlink [18], a mega-constellation of 7,040 low-Earth orbit (LEO) satellites (2024) [19], with 4 million users [20], used 32,000 Linux-based COTS computers, including using Intel™ x86 processors, across its satellites by 2020 [21]. They mitigate soft errors by executing three identical Linux processes on triple modular redundancy (TMR) systems using x86 processors [22]. Although effective, TMR tolerates only one faulty processor in a three-node system, assuming no simultaneous faults impact the majority of TMR processors. Moreover, the probability of simultaneous TMR node failure in space varies with the Sun's Coronal Mass Ejection (CME) rates, as particle flux rises 10–100× during extreme solar cycles [23]. Furthermore, SpaceX uses Byzantine Fault Tolerance (BFT) algorithms to mitigate radiation-induced computational faults in TMR systems [24]. However, adopting TMR or higher-order N-modular redundancy (e.g., 4MR/7MR), aligned with BFT's $3f + 1$ threshold to tolerate (f) faulty nodes, does not improve overall reliability if radiation-induced faults simultaneously affect the majority of NMR nodes [25], defeating BFT's consensus protocols. These state-of-the-art approaches reveal limited Linux kernel reliability enhancements on COTS hardware for radiation-induced soft errors, that primarily rely on COTS's limited ECC memory for error masking.

Like Starlink, Linux usage on COTS hardware with similar reliability challenges is expected to grow with emerging satellite internet providers like OneWeb and Amazon's Project Kuiper. Together, they plan to deploy 6,372 and 3,236 satellites, respectively, by 2030, per US FCC filings [26], [27]. This stresses the need for thousands of reliable COTS processors, like SoCs, for spaceborne Linux computers in the coming decades, given current demand persists.

In space, radiation exposure in COTS electronics degrades electronic reliability through three primary phenomena: single-event effects (SEEs), total ionizing dose (TID), and displacement damage (DD). While TID and DD accumulate over time through distinct physical mechanisms, SEEs occur when energetic particles like protons, heavy ions, or neutrons, via direct or indirect ionization, deposit charges after penetrating a semiconductor substrate, potentially inducing transient soft errors or permanent hard errors. Soft errors occur when the collected charge (Q_{coll}) at a circuit's sensitive node exceeds the critical charge threshold (Q_{crit}), inducing transient faults that either change the output state or cause a transient spike that gradually dissipates. Unlike hard errors, which may cause permanent silicon damage, including unmitigated latch-up, soft errors are transient, nondestructive, and repairable. Soft errors, central to our study, include single-event upsets (SEUs) and single-event functional interrupts (SEFIs), which are:

1) SEUs: These include bit errors in memory cells (e.g., bit flips in DRAM, SRAM, or FPGA configuration memory) and logic circuits with state retention (e.g., flip-flops and latches). SEUs are typically mitigated using error-correcting codes (ECC) memory in hardware or automated software rewrites to repair corrupted data. If uncorrected, SEUs can propagate across memory hierarchies, potentially causing cascading failures, including silent data corruption (SDC), that degrades memory device's reliability.

2) SEFIs: They disrupt system operation by corrupting control logic in system components, leading to resets, hangs, or unintended state transitions. In memory devices, SEFIs often corrupt data across multiple rows and columns. They are mitigated through power cycling, watchdog timers, and redundancy-based fail-over systems (e.g. safe modes).

Notably, Ecoffet et al. [28] report a NASA study of 100 spacecraft anomalies (for both transient and permanent failures) from 1974 to 1994, attributing 45% of anomalous events to space radiation effects, with 80% of those being transient upsets (soft errors). Such soft errors are often the root cause of spacecraft software failures [29], exacerbated with increasing transistor densities in smaller nodes on modern COTS SoCs. However, the empirical data shows that per-bit SEU rates in DRAM have decreased by $5\times$ per generation, and DRAM soft error rates (SERs) have dropped by more than $1{,}000\times$ across five generations [30], [31]. This decline is primarily attributed to reduced charge collection efficiency (Q_{coll}) at the junction, despite plateaued operating voltages. However, the $\approx 2x - 4x$ increase in transistor density per generation coupled with the reduction in critical charge (Q_{crit}) required to toggle a DRAM bit has offset net reliability gains in smaller nodes [32].

Furthermore, Linux on Symmetric Multiprocessing (SMP) architecture (on COTS SoCs) exacerbates fault propagation risk, as multiple cores can access corrupted kernel data structures (e.g. NULL pointer dereference) without adequate soft error handling, potentially causing kernel crashes, as illustrated in Figure 1. Likewise, the strong inter-dependencies among Linux kernel components hinder full modularization of the kernel for limiting cross-module fault propagation, as shown by the Unikraft Project [33]. Corley et al. [34] discovered this vulnerability in their neutron irradiation experiments on a quad-core Raspberry Pi 3B+ SoC (ARM Cortex-A53) running Debian Linux. They observed radiation-induced faults propagated to inactive CPU cores in an SMP Linux environment. Notably, Kernel *Oops* messages were recorded in kernel logs for idle CPU cores (CPU1-CPU3), even though only CPU0 was actively running a test benchmark, indicating faults propagated across CPU cores. These SMP vulnerabilities in COTS multi-core processors show that SEE-induced Linux errors can bypass hardware-software isolation under irradiation, requiring careful mitigation to restrict cascading failures. Rodrigues et al. [35] reinforced that parallel applications running on Linux are more prone to errors than bare-metal implementations, as evidenced by tests using parallel APIs such as OpenMP and Pthreads. In contrast, legacy space-grade processors predominantly used single-core designs for deterministic behavior and simplified fault tolerance [36] for running software like Linux.

Modern COTS SoCs also utilize advanced transistor technologies like planar CMOS and FinFETs, with shrinking node sizes driven by semiconductor advances. Consequentially, differences in transistor technology affect soft error tolerance, resulting in varying reliability under identical exposure [37]. Despite several irradiation studies on COTS SoCs running Linux, the reliability of an unmodified Linux kernel across architecturally diverse SoCs under irradiation remains unexplored. Specifically, systematic studies to identify soft-error-led vulnerabilities in Linux kernel components such as corrupted data structures are lacking.

Previous studies have injected artificial faults (FIs) into specific Linux kernel areas, such as kernel data structures [17], to analyze fault cause-and-effect relationships. However, artificial FIs introduce test biases by targeting selective fault models, potentially overlooking critical vulnerabilities. This limitation is particularly critical for space missions, where in-orbit repairs are unfeasible. In contrast, irradiation tests offer a more realistic evaluation, as stochastic nuclear events induce random faults beyond artificial FI constraints. Additionally, irradiation tests ensure uniform radiation exposure across hardware and software layers, maintaining constant Linear Energy Transfer (LET) across ion penetration. In this study, we structure our work around two primary research questions (*RQ#s*) as follows:

1) *RQ#1: Which Linux kernel subsystems are at high risk of failure under SEEs in COTS SoCs?*

2) *RQ#2: How can targeted software-based mitigation on Linux be engineered to mitigate these vulnerabilities?*

We evaluate three architecturally distinct SoCs under irradiation as our Device Under Test (DUT): two hardcore and one soft-core SoC—Raspberry Pi Zero 2 W (40 nm CMOS, ARM Cortex-A53), NXP i.MX 8M Plus (14 nm FinFET, ARM Cortex-A53), and OrangeCrab v2 (40 nm CMOS FPGA, RISC-V). Notably, the Linux-capable RISC-V soft-core SoC was tested on an FPGA's programming fabric to evaluate Linux reliability on a soft-core under irradiation. Testing RISC-V soft cores running Linux provides greater flexibility for implementing targeted hardware-software co-design to improve fault tolerance, a flexibility absent in fixed architectures like ARM Cortex-A53 SoCs. In our test, each DUT ran Linux in its vendor-provided configuration, i.e., without software hardening or hardware shielding. Our key contributions to this paper are as follows:

1) *We demonstrate that the 14 nm FinFET-based NXP SoC (DUT) achieved 2–3× longer Linux OS uptime and 4× fewer bitflips (without ECC) to its 40 nm CMOS counterparts.*

2) *We present a comprehensive dataset on SEE-induced Linux failures, identify vulnerable kernel subsystems, and propose targeted software mitigations to improve Linux system reliability.*

II. Related Work

A. Radiation Testing of Linux on Modern SoCs

* Corley et al. [38] performed accelerated neutron irradiation on the Raspberry Pi 3B+ (ARM Cortex-A53 SoC) and Raspberry Pi 4 (ARM Cortex-A72 SoC), both running the Linux operating system during irradiation. In their study, kernel paging request errors and kernel "Oops" messages were observed, consistent with prior tests on the Pi 3B+. Custom signal handlers (SIGSEGV, SIGILL, and SIGFPE) were implemented to prevent the misclassification of "Oops" errors as standard SEFIs. Further analysis revealed that some events initially tagged as SEFIs were, in fact, SBUs affecting critical instructions, highlighting the need for proper classification of Linux SEFIs. The study concluded that, in the absence of ECC, both Raspberry Pi models exhibit moderate resilience to neutron-induced faults. However, the Linux OS's memory footprint increases susceptibility to SEUs, although it aids in error classification through software benchmark tests. Their findings underscore the need for improved fault tolerance in Linux design for radiation-prone environments.

* In a study by Guertin [39], Raspberry Pi 3B Plus and Pi Zero were subjected to SEE and TID analysis. For SEE analysis with heavy-ion exposure, ion energy around 20 MeV/amu was chosen, and the linear energy transfer (LET) were around $0.58-8.95$ MeV·cm^2/mg. The processors running Linux were reported to crash at all times under the beam with Linux exceptions or error messages. However, no such Linux error logs were collected, and no bit errors were examined. Their SEFI cross-section reveals an issue with the particle's range to traverse the flip-chip fully around $6-7$ MeV·cm^2/mg, although for LETs below 6 MeV·cm^2/mg, the SEFI cross-section rates appeared nearly comparable. Interestingly, an automatic reset caused by SEFI distorted the boot screen's video output. Notably, the Pi Zero series employs either a Package-on-Package (PoP) or System-in-Package (SiP) die-packaging setup, which vertically stacks the SoC and DRAM. This design presents a testing challenge that requires the test particles to traverse the stack to reach the flip-chip or impact-sensitive junction. Guertin introduced the Raspberry Pi for Space Guidelines [40] and the Board Level Proton Testing Book of Knowledge [41], which offers insights into proton irradiation and system reliability for Raspberry Pis.

* Miller et al., in their irradiation experiment [3], evaluated the reliability of their Space Operating Linux (SOL). SOL is a minimal Yocto-based OS for the Nvidia Jetson TX2i, designed for LEO orbits, featuring a TMR-verified boot process to mitigate radiation-induced corruption on critical boot files. Their fault-injection experiments confirmed that corrupt bits were found in the boot partition and were corrected with each reboot. However, without the TMR-boot procedure, the standard Linux for Tegra (L4T) on the same SoC failed to reboot without such a TMR mechanism, highlighting the efficacy of their method. Furthermore, they proposed RAM-based file systems to reduce the dependence on eMMC flash storage, which is known to be highly susceptible to radiation effects. The expected error rates for SOL devices were also reduced by roughly a third compared to L4T under solar minimum and solar maximum projections. They estimate less than four reboots per year of up-time on SOL for ISS orbits and assess the overall reliability of the TX2i SoC as adequate to operate under radiation effects. However, they suggest decoupling the chip from a few unreliable SoC components, such as the vulnerable embedded MultiMediaCard (eMMC), a non-volatile memory.

* Wyrwas [42], [43] analyzed Linux SEE responses on NVIDIA Jetson TX1 and TX2 SoCs (20 nm) under 200 MeV proton irradiation. For TX1, they employed particles and fluidsGL simulations to maintain low CPU usage and high GPU performance under radiation. Upsets occurred in all 16 test runs across varying proton fluences, fluxes, loads, and temperatures, with distinct Linux crash conditions, such as "Invalid Instruction", "Undefined Instruction", "Network failure", "Kernel paging request Error", "Level 2 translation fault", "Level 3 permission fault", or "Crash of Wi-Fi driver, bus failure, etc." Memory analysis revealed corrupted virtual addresses, including NULL pointer access (PGD NULL), likely due to page table faults. Some runs showed 8-bit flips, suggesting cascaded errors from a single upset. For TX2, all tests resulted in SEFI failures requiring power cycling. In their estimation, TX2 exhibited a smaller cross section than TX1, by at least an order of magnitude. Jetson SoCs are renowned for AI/ML acceleration in edge computing.

* Wali et al. investigated how running a Linux OS on an SRAM-based FPGA SoC with a RISC-V core influences the system's resilience to SBUs in the FPGA configuration memory [44]. In their fault injection experiment, they inject 10,000 random faults across the FPGA's configuration memory, particularly to track the occurrences of masked errors, SDCs, and SEFIs, and detect any architectural internal failures (AIFs). They found that employing OSes increases the SEFI rate by 3.6 times that of the bare-metal setup, highlighting the downside of the long reboot time of GP-OSes. Furthermore, they argue that configuration memory upsets are mainly independent of the workload nature, suggesting that upsets predominantly disrupt control logic rather than data paths.

III. Testing Methodology

This section describes the stress tools and beam parameters employed to test the DUTs, summarizes their hardware-software features, as briefly illustrated in Figure 2.

A. Significance of Stress Testing

Our test methodology hypothesizes that stress testing SoCs increases overall software events, increasing the probability of higher Linux soft error rates (SERs) compared to an idle state. To validate this hypothesis, we used the Linux perf tool on the i.MX 8M Plus (DUT) in a non-irradiated environment to assess the effectiveness of our stress configuration prior to irradiation testing. The perf analysis reveals that our stress test parameters, even without irradiation, caused an $\approx 2,352x$ increase in Linux page faults, as shown in Figure 4. The figure highlights key Linux performance metrics, demonstrating significant increases in system activity during stress testing, including CPU cycles, context switches, executed instructions, and branches. We expect these elevated software events to interact with SEEs, with some propagating to other subsystems and kernel components. Furthermore, stressing DUTs at peak performance could help validate COTS hardware and software reliability under prolonged complex workloads, particularly amid radiation-induced soft errors. This rigorous testing assesses workload-intensive tasks, such as AI/ML, in COTS-based space systems [3].

However, the error observation time window is short when performing accelerated radiation tests on sensitive software suites such as the Linux OS with multiple single points of failure. For instance, a single Linux-SEFI event in a critical control logic could result in a complete Linux crash or hang. During our test campaign, Linux's mean-time-to-failure (MTTF) consistently remained under 3 minutes across all DUTs, regardless
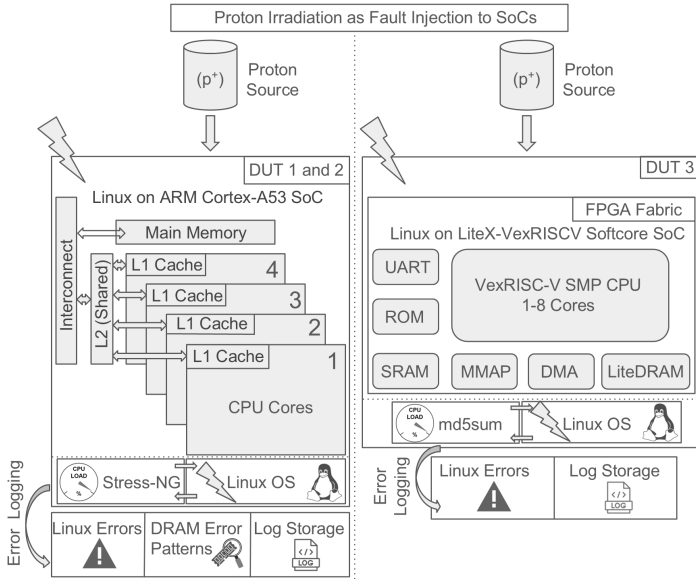
Fig. 2: Block diagram of ARM Cortex-A53 SoC (left) and VexRISC-V softcore (right), running Linux OS with kernel and memory errors collected under proton irradiation

of beam energies (20–58 MeV) and flux levels ranging from 3×10^6 to 1×10^8 cm^{-2}s^{-1}. We ran stress tests continuously while the DUT was under the beam until it eventually crashed, requiring a reboot to restore Linux operation or complete the expected test run time. However, during dry runs of DUTs under irradiation, crash times remained unchanged without stress tests, indicating that stressing did not directly affect Linux crash intervals. We set a 60-second test timeout for test runs, given the short observed MTTF under the beam, to ensure `stress-ng` complete its full scan.

B. Stress Testing Tool: Workload and Test Parameters

We employed the `stress-ng` utility [45], a widely used software stress tool, to expose vulnerable hardware and software components likely to fail under combined workload stress and radiation-induced failures. Using the `stress-ng` tool without Linux root privileges, we exercised all four CPU cores of the ARM Cortex-A53 SoC, including critical SoC components such as I/O peripherals, virtual memory (VM), and hard disk drive (HDD) storage. We used Fast Fourier Transforms (FFTs) as the choice of workload for computational stress on these SoCs, primarily due to the highly memory-bound nature of FFTs. Operationally, FFTs decompose a recursive or iterative data set into smaller components, requiring frequent memory access to move data between memory and processing units. We anticipated that intensive FFT workloads operating on corrupted data affected by soft errors could lead to elevated fault propagation. In Table VI, Table VII, and Table VIII, we present the Linux error data from the irradiation test, showcasing the effectiveness of our unique, exhaustive test parameters in capturing a diverse range of soft errors not reported in similar studies.

C. Significance of Proton Testing

We opted for a proton irradiation test with an energy range of 20–58 MeV for the experiment. Proton irradiation is a well-established method for the preliminary assessment of device susceptibility to SEEs. Although the Linear Energy Transfer (LET) of proton secondaries is lower than that of heavy ions and falls below LET qualification thresholds, protons remain valuable for inducing non-destructive errors, such as SEUs

and SEFIs, which are central to this study. Proton board-level testing is also used for screening out too-sensitive equipment, estimating in-orbit error rates, including validation of mitigation techniques. In this work, proton testing was instrumental in analyzing Linux's resilience to soft errors. LaBel et al. [46] recommend proton SEE testing for devices with a heavy ion Linear Energy Transfer (LET$_{th}$) below 37 MeV·cm^2/mg, where LET$_{th}$ is the threshold with no events at a test fluence of 1×10^7 particles/cm^2.
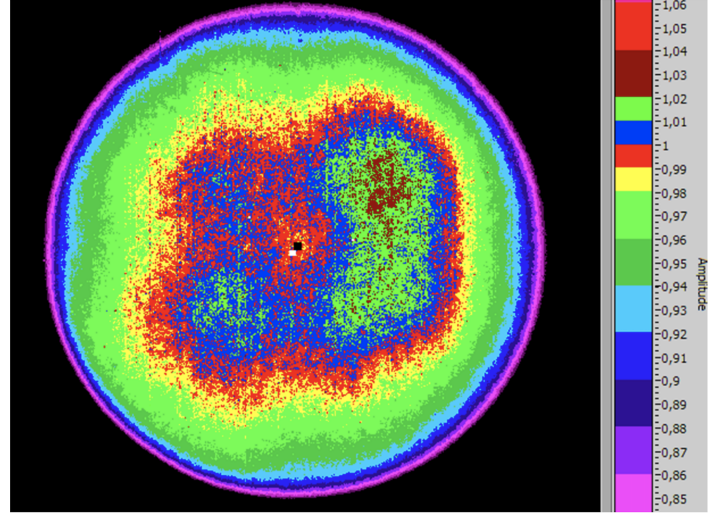
D. Beam Configuration

Fig. 3: 2D beam profile, normalised intensity

We conducted proton radiation experiments at the IFJ PAN in Krakow, Poland, using a 60 MeV proton beam produced by the AIC-144 isochronous cyclotron. Figure 5 illustrates the DUT setup in the radiation chamber. The irradiation was performed at a small-field irradiation station. The samples were placed in air, and a monoenergetic proton beam with a uniform field of 40 mm in diameter was used. The tests utilized beam energy configurations ranging from 20 to 58 MeV. The flux ranged from 3×10^6 p/cm^2·s to 1×10^8 p/cm^2·s, maintaining a flat lateral distribution, with an estimated fluence uncertainty of approximately 3%. Figure 3 illustrates an example of a proton beam field measured on the first day of a three-day test during QA of the irradiation facility. On subsequent measurement days, the beam lateral distributions were similar within statistical spread. Fluence and flux were determined based on dose measurements using a Markus ionization chamber placed in a PMMA phantom and a reference class electrometer PTW UNIDOS. Measurements were performed according to the IAEA TRS398 protocol [47]. In the calculations of fluence, active cross sections from the SRIM 2013 [48] were used. In this setup, a penetration range of 0.9–2.3 mm was sufficient to reach all sensitive regions of our DUTs in both planar and SiP configurations.

E. DUT Selection

Our DUT selection, particularly SoCs, prioritized SWaP attributes and cost-effectiveness for CubeSat and NanoSat missions, targeting LEOs with relatively low radiation levels compared to higher orbits. Additional factors, such as long-term product support availability (e.g., >5 years lifecycle) and an active development community, were also prioritized during DUT selection. Our test objectives required SoCs capable of running 32 or 64-bit Linux with adequate system resources.
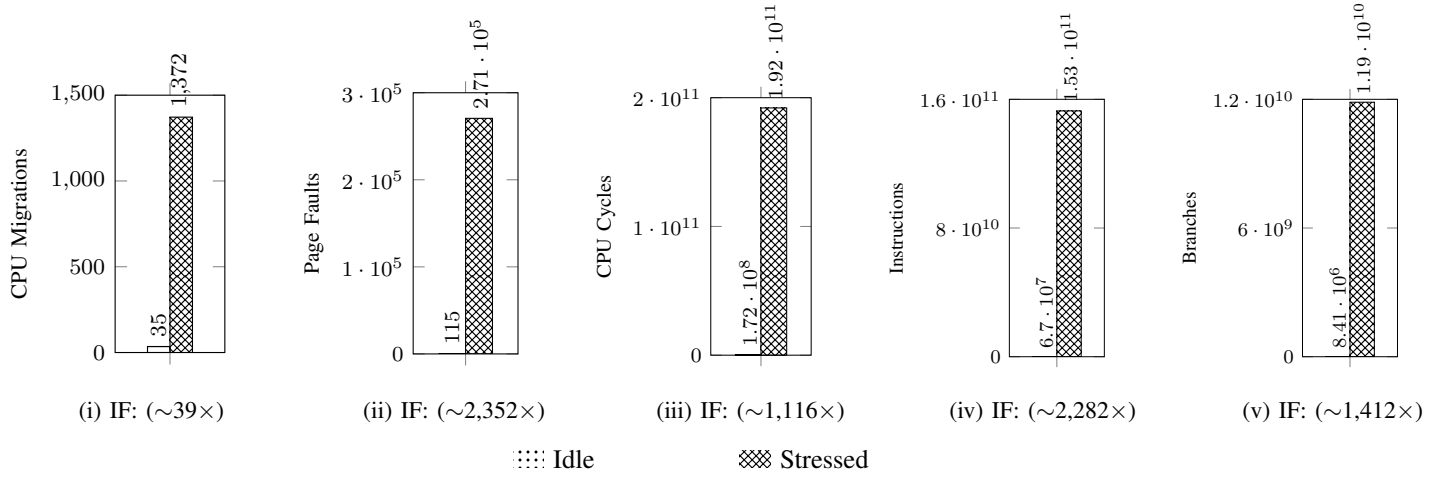
Fig. 4: Linux Kernel Performance Metrics: Idle vs. Stressed (`stress-ng`) via Perf Analysis
(*IF= Increased Factor*)

After evaluating all criteria, we selected the following devices: i) Raspberry Pi Zero 2 W, SoC on a Single-Board Computer (SBC); ii) NXP i.MX 8M Plus, SoC on a System-on-Module (SoM); and iii) OrangeCrab v2 ECP5 FPGA, SoC on a RISC-V softcore, all capable of running the Linux OS. In Table I, we briefly compare the hardware and software specifications of all three DUTs.

To power all the DUTs in the irradiation chamber, we employed an external programmable power supply (PPS). The PPS continuously monitors and logs all power-related signatures to an external dedicated computer outside the radiation chamber. This setup enables real-time detection of current latch-ups, including abnormal power consumption signatures during DUTs' irradiation. An internal Dynamic Host Configuration Protocol (DHCP) was configured to provide IPs to the DUT and an external monitoring machine through a wired router. Through these connections, we enabled SSH connections to control, monitor, and log the performance of the DUT. After each test objective was completed, all DUTs were replaced with identical copies of their respective compute modules. This precaution prevents bias in SEE analysis during prolonged DUT testing and cumulative effects like total ionizing doses (TIDs) accumulating charges. Similarly, displacement damage (DD) degrades the semiconductor lattice, leading to slow material displacement and influencing SEE analysis. The subsequent section comprehensively overviews each DUT's hardware and software specifications.

1) DUT 1: Raspberry Pi Zero 2 W

Hardware: The Raspberry Pi Zero 2W (rpi02w) builds on the legacy of its predecessor Raspberry Pi Zero [49], which served as the primary flight computer in NASA's GASPACS CubeSat mission [50]. During its mission, rpi0w's single-core SoC demonstrated its adequate mission-specific reliability by deploying an elongated boom system to stabilize and orient the CubeSat without active propulsion. This sparked our interest in its successor, the Raspberry Pi Zero 2 W (rpi02w), which offers significant computational throughput improvement with ARM™ Cortex-A53 (quad-core). Unlike its single-core predecessor, rpi02w supports 64-bit Linux, whereas the earlier version was limited to 32-bit Linux. The rpi02w board employs a RP3AO-AU System-in-Package (SiP) design, integrating a BCM2710A1 SoC with 512 MB LPDDR2 memory in a vertically stacked configuration, as shown in Figure 6. However, rpi02w lacks ECC memory protection, making both SoC and DRAM (LPDDR2) memory susceptible to radiation-induced memory failures. During irradiation tests, only the SiP (comprising the SoC and SDRAM) was exposed to the proton beam at a zero-degree angle. The microSD card, which stores the Linux filesystem, was positioned outside the beam's influence since testing the SoC and memory was our primary objective.

Software: The rpi02w operated on the vendor-supplied Raspbian Linux variant, configured with minimal adjustments to streamline testing and logging: (i) The Pi was set to auto-login as root on every reboot, ensuring testing tools ran with root privileges to prevent permission issues during tests. (ii) We ensured test logs were directed to an auto-mounting external USB flash drive, including real-time logging from the external monitoring unit. However, if the auto-mount of the USB fails at boot time, it can lead to a longer "boot delay" until a proper timeout is defined. A notable limitation of Pi's is the absence of an onboard Real-Time Clock (RTC), which tracks the current time and date, even when the system is powered off. This causes the system to lose track of time at startup if no time synchronization is configured from the NTP servers. If not configured, it complicates the timestamping of the log files critical for post-processing.

2) DUT 2: NXP i.MX 8M Plus

Hardware: The i.MX 8M Plus processor is powered by ARM64 quad-core Cortex-A53 cores, augmented with a dedicated Neural Processing Unit (NPU) capable of delivering performance up to 2.3 Tera Operations Per Second (TOPS). Fabricated on a 14 nm FinFET process, the processor is designed for low power consumption and high performance, making it particularly suited for edge computing and machine learning applications [51]. The 14 nm FinFET architecture is known for lower soft error rates (SER) due to a smaller sensitive volume, improved electrostatic control, and reduced leakage compared to older planar CMOS processes (e.g. 40 nm) used in the other two DUTs. In particular, its inline ECC memory (disabled by default) safeguards its 768 KB of L2 cache directly within the cache hardware, including support for its LPDDR4 memory. However, implementing ECC usage comes with performance penalties, including boot delays when protecting boot partitions. Overall, its promising hardware features position it as a strong candidate for evaluating the soft-error reliability of Linux on such SoCs.

Software: The i.MX 8M Plus board natively supports Portainer [52] for managing Docker images. In our test,
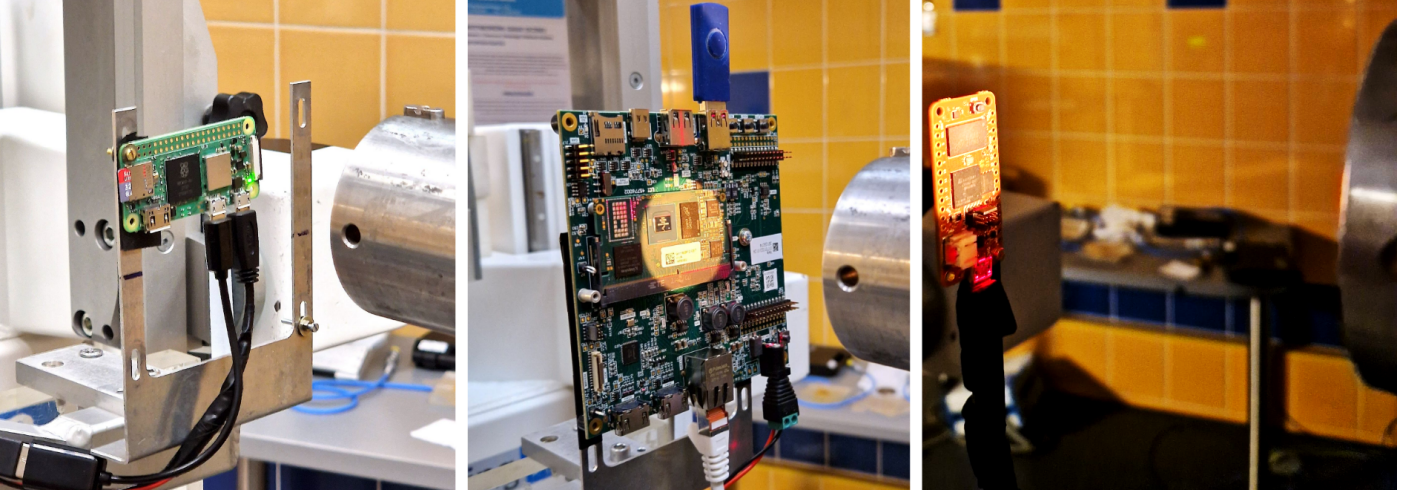
Fig. 5: DUTs in Proton Beam Testing: Raspberry Pi Zero 2W (Left), NXP i.MX 8M Plus (Center), OrangeCrab FPGA (Right)

`stress-ng` ran in a non-privileged Docker container, which, unlike virtual machines (VMs), uses the host Linux kernel for its functioning. Docker also uses settings like `namespaces` and `cgroups` to manage processes' requirements for CPU, memory, and I/O from the underlying hardware. However, despite Docker's process isolation, data show containerized applications perform identically to native (Linux) process execution, aside from a few seconds' initialization delays from the Docker daemon [66]. Likewise, Docker processes share the same memory management and Linux kernel's default Address Space Layout Randomization (ASLR) memory assignment for its processes as host user-space Linux processes. Thus, despite the isolation, the memory used by processes running within Docker remains equally exposed to SEEs unless it is explicitly confined to a specific memory region.
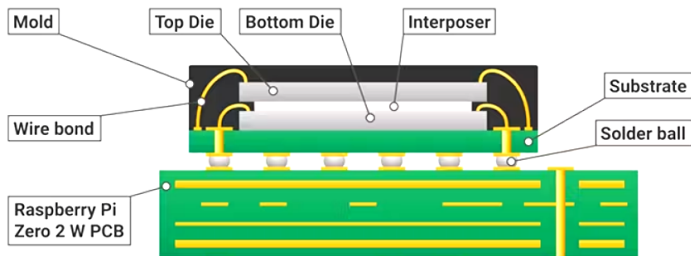

Fig. 6: RP3A0 utilizing a System-in-Package (SiP) configuration with Broadcom BCM2710A1 (bottom die) and 512 MB LPDDR2 DRAM (upper die) [53]

3) DUT 3: OrangeCrab v2

Hardware: The OrangeCrab FPGA, a pocket-sized development board, is of the Lattice ECP5 family [55], known for low power consumption and computational capabilities in a small form factor, ideal for resource-constrained designs. The FPGA is available in two configurations, 25,000 or 85,000 lookup tables. OrangeCrab's favorable SWaP profile makes it an appealing candidate for Linux deployment on soft-core processors in FPGA systems. With its 85,000-LUT variant, the FPGA can accommodate up to four RV32C(compressed 32-bit VeXRISC-V) cores as soft processors within a SoC. In our implementation, we opted for a variant with 25,000 LUTs to host a single VeXRISC-V CPU core running Linux in the SoC implementation.

Software: We used the Linux-on-Litex-VexRISCV tool [56] to configure a custom SoC implementation, enabling Linux

to run on a softcore processor in FPGAs. Running Linux on a softcore SoC in an FPGA's fabric accelerates software development with Linux's extensive libraries, unlike HDL/Verilog workflows that require complex hardware description for functionality implementations (e.g. implementing TCP/IP layers using Verilog/HDL). Notably, the LiteX tool's toolchains and frameworks are built entirely on open IP cores, open-source projects [57]. Furthermore, the VexRISCV CPU cores, central to our SoC component, are written in the Spinal Hardware Descriptive Language (HDL) that could be configured to support Linux with Memory Management Unit (MMU) support [56]. Heinz et al. [58] showed that the VexRISC-V CPU achieves $\sim$ 1.2 CoreMark per MHz with minimal resource usage, making it one of the fastest single-core RISC-V cores among seven compared implementations in their study.

A 32-bit Linux image was built to run on our SoC (DUT) using the Buildroot [59] tool, containing only essential libraries and test utilities. Linux boot files were loaded from an onboard microSD card, placed outside the beam's influence, which contained: i) Linux kernel image (Image), ii) a Device Tree Blob (rv32.dtb) file defining RISC-V hardware configuration, iii) an Open Supervisor Binary Interface (OpenSBI) file for an abstraction layer between the RISC-V hardware and Linux OS, iv) and a compressed root filesystem for Linux (rootfs.cpio). No ECC protection was implemented in FPGA, nor was any ECC memory present in the vendor-supplied configuration.

IV. EVALUATION I: ESTIMATING CROSS-SECTION OF LINUX CRASHES UNDER SEFIS

This section evaluates the Linux-SEFI-induced failure cross-section for three DUTs, highlighting architecture-specific Linux susceptibility on COTS SoCs.

A. Definition of Linux-SEFI

Our experiment defines Linux-SEFI terminology as an extension of JEDEC's SEFI definition, which specifies that SEFIs do not require device power cycling to restore device functionality [60]. However, in the Linux OS, SEFI manifestations vary in severity, including unrecoverable faults requiring an OS reboot, as repairing a corrupted kernel is often impractical compared to a simple reboot. For instance, Linux-SEFIs affecting control logic or critical registers can cause segmentation faults that may lead to Linux crashes, requiring a reboot. However, minor to moderate Linux-SEFI faults, such as offending Linux processes (e.g. Kernel Oops), are terminated by the Linux Kernel while

TABLE I: All DUTs compared by hardware architecture and key specs

DUT	Raspberry Pi Zero 2 W	NXP i.MX 8M Plus	OrangeCrab v2 ECP5 Lattice
Type	SBC (SoC)	SoM (SoC)	FPGA (SRAM-based)
Technology Node	40 nm CMOS	14 nm FinFET	40 nm FPGA Fabric
CPU	ARM64 Cortex-A53	ARM64 Cortex-A53	VexRISC-V (Synthesized in FPGA)
ISA	ARMv8-A (64-bit)	ARMv8-A (64-bit)	RV32I (32-bit)
Cores	4 (quad-core)	4 (quad-core)	(single-core) Configurable with 25K LUTs
Frequency	1.2 GHz	1.8 GHz	64 MHz (FPGA-Configured)
RAM	512 MB LPDDR2	4096 MB LPDDR4	1 Gbit DDR3L + 128 MBit Flash via QSPI
OS	Raspbian Lite (5.15.84, aarch64)	Yocto (5.15.77, aarch64)	Linux buildroot 5.0.13 (riscv32, VexRISCV)
Power	2.5–3W (Peak)	3–5W (Peak)	1.5–6.3W (Peak)
Package	65mm X 30mm	15mm X 15mm 0.5 pitch FC-BGA	10mm X 10mm 0.5 pitch (285-ball csfBGA)
Component Under Radiation	SiP: RP3A0-AU (Integrated 512 MB LPDDR2 + BCM2710A1 SoC)	SoC: NXP i.MX 8M Plus Application Processor + Micron's 4GB LPDDR4 External Memory	FPGA Fabric (Part: LFE5U-25F-8MG285C) with 25,000 LUTs

Linux remains operational, avoiding the need for a reboot. This terminology refines SEFIs' unique characteristics on Linux functionality failures, characterized by their crash severity. In our analysis, each Linux crash is counted as one Linux-SEFI event, which may result from one or more detected or undetected failures contributing to each crash.

B. Significance of Cross-Section Analysis in Estimating SEEs

In estimating DUT's reliability for radiation-induced failures, cross-section (σ) analysis quantifies a DUT's sensitivity to SEEs. The cross-section data represents an effective area (*per cm^2*) of the DUT exposed to a radiation beam to measure the hardware-software vulnerability to radiation effects. A large cross-section implies a higher probability of radiation-induced errors per unit fluence and is a key parameter in assessing the reliability of devices such as DRAMs, SRAMs, processor caches, registers, etc. The cross-section is reported as $\langle\sigma\rangle \pm \delta\sigma$, where $\langle\sigma\rangle$ is the sample mean from experimental observations, and $\pm\delta\sigma$ represents the standard error, reflecting the uncertainty in SEE measurement.

Throughout our irradiation campaign, the total events (each representing a single instance of Linux-SEFI crashes) consistently stayed less than 50 per DUT (Ne < 50), primarily because of Linux's high failure on COTS SoCs without ECC. Ideally, a large number of events (Ne > 50) would improve SEE cross-section estimation by reducing statistical uncertainty and narrowing confidence intervals. However, the frequent Linux crashes (with an MTTF < 3 minutes) required that the beam be switched off before starting each test run, as Linux often failed to reboot while the beam was on. The frequent beam switching during irradiation tests limited large event count, limiting Ne < 50. Despite this low count, Poisson statistics remain a valid approximation for small observed event counts (e.g. Ne < 50), as long as failures occur at a constant rate per unit fluence.

In this framework, a 95% confidence interval (CI) defines the range within which the true SEE cross-section is expected to fall with 95% probability, as detailed in [61]. We utilized Ricker's statistical limits [62], to derive 95% confidence bounds for low-event counts, making them suitable for SEE cross-section estimation where conventional Gaussian approximations are unsuitable for sparse data. For example, using Ricker's confidence limits, a single SEE at a fluence of 10^6 cm^2 yields a 95% confidence interval of 1.0×10^{-7} $cm^2 < \sigma < 5.6 \times 10^{-6}$ cm^2, addressing statistical uncertainty in small-sample cross-section estimates. Javanainen in [61] discusses methods to estimate the SEE cross-section ($\delta\sigma$), accounting for fluence delivery uncertainties, as formalized in Equation (1).

$$\delta\sigma = \sigma \cdot \sqrt{\left(\frac{\delta N_e}{N_e}\right)^2 + \left(\frac{\delta\Phi}{\Phi}\right)^2}, \quad \text{where} \quad \sigma = \frac{N_e}{\Phi}. \quad (1)$$

Notation:

N_e : Number of observed Linux crashes (Poisson-distributed).
δN_e : 95% CI for N_e (from Ricker's table).
Φ : Delivered fluence (particles cm^{-2}).
$\delta\Phi$: Uncertainty in fluence measurement.
σ : Linux-SEFI cross section (cm^2 per unit fluence).
$\delta\sigma$: Propagated uncertainty in cross-section estimation (σ).

C. Cross-Section Analysis of Linux-SEFI Crashes across DUTs

In this subsection, we calculate the Linux-SEFI cross-section for each DUT under irradiation using Equation (1). In Table II, we provide the total events (observed Linux-SEFI crashes) for each DUT at various proton energy levels (MeV), along with the delivered flux and cumulative fluence, which sums total fluence per run for each energy level. Additionally, Table III provides SEE cross-sectional data for Linux-SEFI crashes, calculated using Equation 1. Here, σ(Observed) represents the experimentally measured Linux-SEFI cross-section (*per cm^2*), defined as the number of Linux crashes (*f*) per unit fluence. Also, the 95% confidence interval bounds, $\delta\sigma$(Min) and $\delta\sigma$(Max), in Table II, are estimated using Ricker's confidence limits, incorporating fluence measurement uncertainty as described in Equation (1).

In Figure 7, we plot the observed cross-section (σ) of Linux-SEFI crashes for all DUTs under irradiation. Our analysis shows that, across the energy range, NXP's SoC consistently demonstrates the lowest SEE cross-section, indicating greater resilience to the other two DUTs.

1) DUT 1:Raspberry Pi Zero 2 W (40 nm CMOS)

The DUT 1 showed a Linux-SEFI cross-section σ(Observed) of $6.95^{+6.85}_{-3.99} \times 10^{-9}$ at 30 MeV, the highest among DUT 1 and 2, and $2.84^{+1.70}_{-1.18} \times 10^{-9}$ cm^2 (at 58 MeV) its lowest, as detailed in Table III(a)'s (95% confidence intervals), derived from Table II(a). At the upper bound, DUT 1's cross-section of $6.95^{+6.85}_{-3.99} \times 10^{-9}$ cm^2 is closely comparable to DUT 3's at $7.63^{+6.87}_{-4.23} \times 10^{-9}$ cm^2. Nearly all tests, except one at 25 MeV, resulted in Linux crashes, as summarized in Table II(a).

During irradiation, the proton beam targeted the SiP (RP3A0-AU), containing a quad-core ARM Cortex-A53 SoC and 4 Gbit Micron LPDDR2 RAM. SiPs are known for their cost-effective 3D die-stacking packaging, requiring particles with sufficient LET to penetrate deeper layers of the die and induce radiation-induced failures, as depicted in Figure 6 of the RP3A0-SiP structures.

2) DUT 2: NXP i.MX 8M Plus Processor (14 nm FinFET)

DUT 2 consistently exhibited the lowest Linux-SEFI cross-section, 5–14x lower than DUT 1 (which also features the ARM Cortex-A53 SoC) and DUT 3, as detailed in Table III(b). For instance, the highest σ was observed at 30 MeV ($1.41^{+1.88}_{-0.96} \times 10^{-9}$ cm^2) and lowest at 20 MeV ($1.98^{+1.30}_{-0.87} \times$

TABLE II: Overview of Proton Beam Logs for All DUTs

(a) DUT 1: Raspberry Pi Zero 2 W (SoC)

Energy	Total Linux Crashes (f)	Cumulative	
(MeV)	per Total Test Runs: (n)	Flux (p/cm²/sec)	Fluence (p/cm²) (Delivered)
20	7/7	3.51E+06	1.08E+09
25	7/8	3.75E+06	1.22E+09
30	8/8	3.43E+06	1.15E+09
40	5/5	3.07E+06	1.06E+09
50	5/5	2.93E+06	1.05E+09
58	17/17	6.17E+06	5.98E+09

(b) DUT 2: NXP i.MX 8M Plus (SoC)

Energy	Total Linux Crashes (f)	Cumulative	
(MeV)	per Total Test Runs: (n)	Flux (p/cm²/sec)	Fluence (p/cm²) (Delivered)
20	12/12	1.02E+07	7.59E+10
25	3/3	1.09E+07	3.56E+09
30	4/5	1.02E+07	3.55E+09
40	2/3	1.14E+07	3.52E+09
50	3/4	1.05E+07	3.50E+09
58	5/10	3.09E+07	1.61E+10

(c) DUT 3: OrangeCrab v2 (SoC on FPGA)

Energy	Total Linux Crashes (f)	Cumulative	
(MeV)	per Total Test Runs: (n)	Flux (p/cm²/sec)	Fluence (p/cm²) (Delivered)
25	5/5	3.15E+06	1.18E+09
30	8/9	3.12E+06	1.18E+09
40	6/7	3.25E+06	1.18E+09
50	3/4	3.38E+06	1.17E+09
58	26/27	3.83E+07	2.71E+10

TABLE III: Cross-Section Analysis of Linux-SEFI Crashes Across DUTs

(a) DUT 1: Raspberry Pi Zero 2W (SoC)

MeV	$\delta\sigma$ (Min) [cm²]	σ (Observed) [cm²]	$\delta\sigma$ (Max) [cm²]
20	2.59×10^{-9}	6.48×10^{-9}	1.33×10^{-8}
25	2.30×10^{-9}	5.73×10^{-9}	1.18×10^{-8}
30	2.96×10^{-9}	6.95×10^{-9}	1.38×10^{-8}
40	1.52×10^{-9}	4.71×10^{-9}	1.10×10^{-8}
50	1.53×10^{-9}	4.76×10^{-9}	1.11×10^{-8}
58	1.66×10^{-9}	2.84×10^{-9}	4.54×10^{-9}

(b) DUT 2: NXP i.MX 8M Plus (SoC)

MeV	$\delta\sigma$ (Min) [cm²]	σ (Observed) [cm²]	$\delta\sigma$ (Max) [cm²]
20	1.11×10^{-10}	1.98×10^{-10}	3.28×10^{-10}
25	1.70×10^{-10}	8.43×10^{-10}	2.47×10^{-9}
30	4.54×10^{-10}	1.41×10^{-9}	3.29×10^{-9}
40	1.72×10^{-10}	8.52×10^{-10}	2.50×10^{-9}
50	2.87×10^{-10}	1.14×10^{-10}	2.91×10^{-9}
58	2.93×10^{-10}	6.21×10^{-10}	1.14×10^{-9}

(c) DUT 3: OrangeCrab v2 (SoC on FPGA)

MeV	$\delta\sigma$ (Min) [cm²]	σ (Observed) [cm²]	$\delta\sigma$ (Max) [cm²]
25	1.36×10^{-9}	4.24×10^{-9}	9.93×10^{-9}
30	3.40×10^{-9}	7.63×10^{-9}	1.45×10^{-8}
40	2.38×10^{-9}	5.93×10^{-9}	1.22×10^{-8}
50	8.62×10^{-10}	3.42×10^{-9}	8.72×10^{-9}
58	8.05×10^{-10}	1.18×10^{-9}	8.05×10^{-9}

10^{-10} cm²), i.e. $14.4\times$ lower than DUT 1's lowest σ (Observed) $2.84^{+1.70}_{-1.18} \times 10^{-9}$ cm² at 58 MeV. Notably, 3 out of 37 Linux test runs completed without crashes in their intended beam run-time, as shown in Table II(b). The i.MX 8M Plus SoC, built on a 14 nm Low-Power Compact (LPC) FinFET process, is known for improved current control and reduced charge leakage with its 3D gate structure to bulk CMOS [63]. During irradiation, the proton beam targeted the SoC (from the top) and 4 GB LPDDR4 module. The SoC integrates a GPU, NPU, connectivity interfaces, and various I/O peripherals. Unlike SiPs, SoMs arrange components horizontally onto a shared PCB substrate, ensuring a relatively uniform exposure to the incident radiation beam. A Weibull fit for this DUT was estimated from its $\sigma_{min}-\sigma_{max}$ range, as shown in Table IV.

3) DUT 3: OrangeCrab v2 FPGA (40 nm CMOS)

DUT 3 exhibited the highest Linux-SEFI cross-section σ(Observed) of $7.63^{+6.87}_{-4.23} \times 10^{-9}$ cm² (at 30 MeV), approximately $5.4\times$ larger than DUT 2's highest cross-section, as shown in Table III(c). Its lowest σ(Observed) of $1.18^{+6.87}_{-0.38} \times 10^{-9}$ cm² is approximately 2.4x lower than DUT 1's lowest value of $2.84^{+1.70}_{-1.18} \times 10^{-9}$ cm². This analysis demonstrates that the Linux implementation on an FPGA fabric using RISC-V SoC has a comparable cross-section to its hardcore counterpart, the Raspberry Pi Zero 2 W, in our setup. Furthermore, it highlights the robustness of the emerging RISC-V ISAs as a viable alternative to established ISAs like ARM and x86 used in COTS processors. The OrangeCrab FPGA, based on the Lattice ECP5 FPGA (40 nm CMOS process, Ball Grid Array (BGA) package), was irradiated with the proton beam targeting its FPGA fabric.
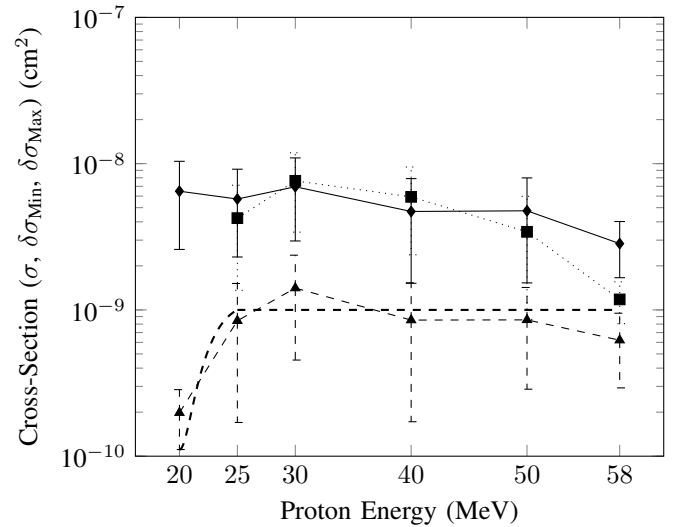
Fig. 7: Cross-sections of Linux crashes across all DUTs at σ_{min} and σ_{max} (derived from 95% confidence intervals)

TABLE IV: Fitted Weibull Parameters for DUT 2: NXP i.MX 8M Plus (SoC)

σ_{min}	$\Delta\sigma$	E_0 (MeV)	λ (MeV)	k
1.11×10^{-10}	1.00×10^{-9}	20	3.37	2

V. EVALUATION II: SEE-INDUCED FAILURES ON LINUX

This section addresses *RQ#1 and RQ#2* by identifying vulnerable Linux components across architecturally diverse COTS

Fig. 8: Vulnerability shares for Linux kernel components across different DUTs: Memory management (mm) dominates in DUT1 (39%) and DUT3 (67%), while filesystems (fs) account for 56% in DUT2.

SoCs and proposing mitigation strategies to enhance resilience against specific Linux failures.

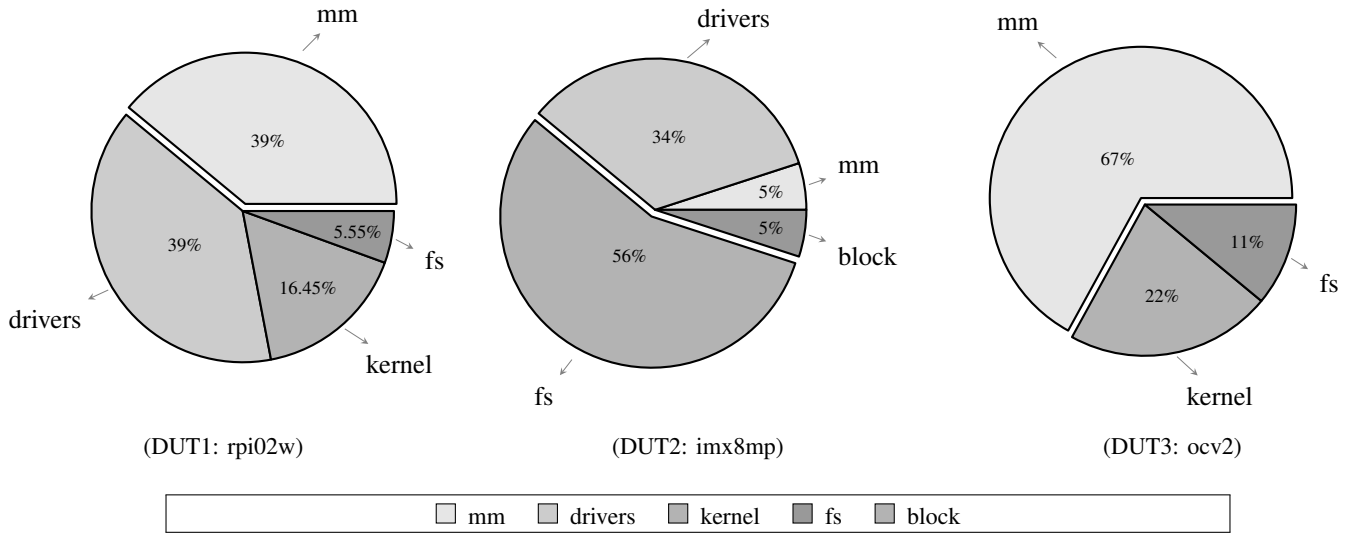Our extensive irradiation data is presented in Tables VI, VII, and VIII, while avoiding redundant counting of identical Linux errors (e.g., multiple occurrences of Kernel `Oops`) in our irradiation data, which are reported only once in the same table. We also traced these errors to their primary handlers in the Linux source code tree; for instance, page faults were linked to `mm/fault.c`. Additionally, we visualize the frequency of these errors using a pie chart in Figure 8. The pie chart highlights vulnerable components based on our observed data; however, it does not provide an exact estimation of specific vulnerable Linux kernel components, as certain faults may not have been immediately visible in Linux logs. Achieving an exact estimation would require comprehensive monitoring of the entire memory address space of the Linux kernel. Instead, our goal is to objectively illustrate the most frequently failing Linux kernel components observed during irradiation.

Based on the criticality of Linux errors, we categorize them into three groups in Tables VI, VII, and VIII, as follows:

- *Critical*: Unrecoverable system states necessitating a manual or automated OS reboot to restore functionality (e.g., segmentation fault).
- *High*: Partially recoverable; requires automated OS intervention, such as process checkpointing and restarts, without a full reboot to restore functionality.
- *Medium-to-Low*: Transient errors with localized impact and minimal reliability degradation when occurring in non-sensitive zones (e.g., ECC correcting bit flips).

Additionally, we observed critical-to-high Linux failures during irradiation that we categorized into two types:

- *Crashes*: Identified when the DUT became unresponsive over an active SSH connection, indicated by no response to ICMP ping requests.
- *Hangs*: Indicated partial Linux OS functionality, such as failed new SSH connections while an existing one remained active, or stress tools running while new terminal commands failed.

In the next subsection, we detail DUT-specific Linux errors based on irradiation data.

1) DUT 1: Raspberry Pi Zero 2 W

Our experimental data on Linux errors, outlined in Table VI, reports frequent low-level MMU corruptions, cascading failures across multimedia (`vc4-drm`), Bluetooth, firmware transaction timeouts, and memory errors to DUT 2 and 3. Under irradiation, DUT 1 experienced multiple reboot failures, likely due to disruptions in the boot process, including boot file corruption and power-related issues. Additionally, overcurrent protection (OCP) events were logged by the external PPS (power supply) unit, indicating that DUT 1's power rails are more sensitive to high-current incidents, possibly micro latch-ups. The DUT recovered after a soft reboot; however, most of the time, it failed to reboot under the beam. A total of 43% of Linux errors were classified as critical (non-recoverable), affecting core computational and power control functions. These included kernel oops, kernel paging errors, and firmware request failures.

Moderate faults accounted for approximately 29% of the total Linux faults, including transient clock signal disruption, I/O, and file system inconsistencies. These errors, while impactful, showed traits amenable to in-system recovery mechanisms such as retry mechanisms, process checkpointing, and restarts, as detailed in Table VI.

Low-impact errors represented 28% of all cases. They indicate errors with minimal reliability impact, including kernel taints. These faults had minimal effect on overall DUTs' reliability during irradiation. Overall, 78% of all detected Linux errors were attributed to the memory management (mm) and device driver (dd) subsystems, as illustrated in pie chart 8. The susceptibility of Linux drivers to frequent crashes aligns with prior studies showing their error rate is three to seven times higher than the rest of the kernel [54]. Jointly, these findings indicate that the CPU and memory subsystems were the most vulnerable components in DUT 1. These susceptible areas in the Linux kernel should be prioritized for software-based mitigation strategies, as proposed in Table VI.

2) DUT 2: NXP i.MX 8M Plus

DUT 2 demonstrated a 50% reduction in medium-severity errors related to CPU and memory, and a 2-3x greater Linux OS uptime under irradiation compared to DUT 1 and 3, as outlined in Table VII. This improvement can be attributed to its architectural advantages, including its 14 nm FinFET technology, compared to the 40 nm CMOS technology used in DUT 1 and 3. However, this improvement was offset by the emergence of various eMMC-related errors affecting filesystem issues (e.g., FAT-fs errors) during I/O operations. These eMMC errors occurred more frequently than other errors,

making eMMC the most vulnerable hardware component in DUT 2. Miller et al. in their irradiation experiment [3] also discovered eMMC's susceptibility on NVIDIA's Jetson TX2i. Their work proposed employing a RAM-based filesystem to reduce reliance on eMMC storage. In our analysis, eMMC errors manifested as ext4 extent failures, aborted journals, and JBD2 errors, highlighting the risk of heavy reliance on eMMC under radiation-induced upsets.

Further analysis revealed that Linux errors were predominantly concentrated in the memory and I/O subsystems. Filesystem (fs) and device driver (dd) errors collectively accounted for 90% of all detected instances, as illustrated in Table VII and Figure 8. However, incorporating DUT 2's inline ECC with SECDED capabilities could have significantly mitigated the observed bit-level errors. Furthermore, DUT 2's test was conducted on a Debian Linux running on top of an embedded Linux host. Observations of container logs during irradiation revealed identical SIGBUS errors in both Docker and host kernel logs, indicating uniform propagation of hardware faults through Linux layers. This shows that containerized software enclaves on a single Linux host can become a single point of failure when software isolation fails to block hardware-layer faults.

3) DUT 3: OrangeCrab v2 FPGA

Linux implementation on DUT 3 differed from the other two DUTs, running a softcore SoC on an FPGA fabric using LiteX's LiteDRAM IP core to handle memory management at the hardware level. Memory management (mm) errors comprised 67% of DUT 3's total errors, with kernel and file system (fs) errors at 22% and 11%, respectively, as illustrated in Figure 8 and Table VIII. For stress testing, Buildroot's inbuilt `md5sum`, which computes MD5 hash checksums, was employed instead of the stress-ng test used on DUTs 1 and 2. The *htop* command confirmed that *md5sum* effectively stressed Linux on the softcore SoC. Common errors frequently observed included NULL pointer dereferences, kernel paging failures, "Bad page map" states, segmentation faults (SIGSEGV) in user and dynamic linker libraries, often accompanied by CPU interrupt errors. These findings align with the typical inherent vulnerability of SRAM-based FPGA designs to radiation-induced SEEs.

VI. CONCLUSION

This work systematically evaluates Linux's resilience to soft errors on three architecturally diverse COTS SoCs under proton irradiation, using exhaustive stress configurations to enhance fault observability and, to some extent, fault propagation. Our preliminary results revealed that the i.MX 8M Plus (DUT 2), with 8× higher transistor density, exhibited 2–3× greater overall Linux OS uptime than the Raspberry Pi Zero 2 W, despite lacking ECC. This improvement is attributed to its 14 nm FinFET architecture and several hardware-software co-optimizations, leading to improved radiation tolerance. However, 56% of Linux failures on DUT 2 stemmed from eMMC vulnerabilities, underscoring the impact of peripheral components on the reliability of otherwise robust SoCs. Results from our irradiation experiments confirm that proton-induced soft errors propagate non-trivially to other Linux subsystems, critically affecting Linux reliability. This study concludes with a detailed analysis of radiation-induced Linux errors and mitigation measures to withstand or prevent these frequently emerging soft-error-induced faults in Linux on modern COTS SoCs.

VII. FUTURE WORK

We aim to create a Yocto-based minimal Linux that incorporates Linux hardening techniques, including periodic checkpoint-restart, TMR boot critical Linux processes, custom error handlers, software ECCs, and resilient file systems like Btrfs. Additionally, we aim to co-design the VeXRISC-V softcore SoC for improved hardware-level resilience to soft errors, including proactive Linux core rejuvenation within the COTS FPGA fabric.

TABLE V: ACRONYMS

Acronym	Full Form
API	Application Programming Interface
BFT	Byzantine Fault Tolerance
COTS	Commercial Off-The-Shelf
CME	Coronal Mass Ejection
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
DD	Displacement Damage
DUT	Device Under Test
FinFET	Fin Field-Effect Transistor
DRAM	Dynamic Random-Access Memory
EDAC	Error Detection and Correction
ECC	Error-Correcting Code
eMMC	Embedded MultiMediaCard
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
HDL	Hardware Description Language
NMR	N-modular redundancy
JBD2	Journaling Block Device 2
LPC	Low-Power Compact
LPDDR2	Low Power Double Data Rate 2
LPDDR4	Low Power Double Data Rate 4
MBU	Multiple-bit Upset
MMU	Memory Management Unit
MTTF	Mean Time To Failure
NPU	Neural Processing Unit
OS	Operating System
PPS	Programmable Power Supply
RISC-V	Reduced Instruction Set Computing V
RPI02W	Raspberry Pi Zero 2 W
SBC	Single-Board Computer
SBU	Singe-bit Upset
SECDED	Single Error Correction, Double Error Detection
SEE	Single Event Effect
SER	Soft Error Rate
SEFI	Single Event Functional Interrupt
SEU	Single Event Upset
SiP	System-in-Package
SMP	Symmetric Multiprocessing
SoC	System-on-Chip
SoM	System-on-Module
SDRAM	Synchronous Dynamic Random-Access Memory
SRAM	Static Random-Access Memory
SWaP	Size, Weight, and Power
TID	Total Ionizing Dose
TLB	Translation Lookaside Buffer
TMR	Triple Modular Redundancy
USB	Universal Serial Bus
VM	Virtual Memory
ZFS	Zettabyte File System

TABLE VI: Analyzing Soft Error-Induced Linux Failures on the Raspberry Pi Zero 2W: Classification and Resilience Strategies

No.	Component	Linux Error Description	Fault Type	Register/Component	Impacted Subsystems
		Critical			
	Memory (mm/)	Unable to handle kernel paging request at virtual address 0xffffff8086fe5ac0 Level-1 Translation Fault (page table error) Code: f9400393 (LDR x19, [x0]#0)	Kernel Paging Request	• pc: ret_to_user+0x0/0x5e8 • lr: el0t_64_sync+0x1a0/0x1a4	• vc4-drm, SoC:gpu, • bluetooth, • mm/fault.c, • do_page_fault(), • do_mem_abort()
1.1	Root Cause Hypothesis	An SEE-induced bit flip in a kernel pointer disrupted MMU address translation, leading to a failed page table lookup, data shows (PGD=0x1160000 → P4D=0x0 → PUD=0x0) and a subsequent null pointer exception during data access.			
	Cascading Linux Failures	• FAT-fs (sda): Volume was not properly unmounted. Some data may be corrupt. Please run fsck. • Internal error: Oops: 96000005 [#1] PREEMPT SMP; • CPU: 2 PID: 866 Comm: kworker/Tainted: G WC 5.15.84-v8+ #1613			
	Proposed Mitigation	• Store critical kernel structures (e.g., page tables) at redundant virtual addresses and periodically verify copy integrity, including use of stack canaries to detect unauthorized modifications. • Enable CONFIG_ARM64_PAN=y, CONFIG_HARDENED_USERCOPY=y, and Kernel Page Table Isolation (KPTI) to mitigate unprivileged memory corruptions.			
	Drivers (dd/) + Power Management	Firmware Transaction Timeout; rpi-firmware: Request 0x00030046 returned status 0x080000001)	Firmware Comm. Error	• pc: rpt_firmware_property_list +0x1dc/0x238 • pc: vc4_hdmi_connector _detect_ctx +0x2d8/0x348 [vc4] • lr: rpt_firmware_property_list +0x1dc/0x238 • lr: vc4_hdmi_connector _detect_ctx +0x48/0x348 [vc4]	• vc4-hdmi • firmware/raspberrypi.c • clk/bcm/clk-raspberrypi.c • hwmon/raspberrypi-hwmon.c • rpi_firmware_transaction() • rpi_firmware_property_list() • rpi_firmware_get_throttled() • raspberrypi_fw_set_rate()
1.2	Root Cause Hypothesis	An SEE-induced firmware corruption caused a timeout in a VideoCore Mailbox API request (offset +0x1DC) intended to power on the HDMI driver. Simultaneously, intensive mmap operations by Stress-ng and unstable voltage rails delayed time-sensitive firmware interactions and compromised firmware integrity.			
	Cascading Linux Failures	• Out of memory: Killed process @ PID 957 (stress-ng); • WARNING: CPU: 3 PID: 422 at drivers/gpu/drm/vc4/vc4_hdmi.c:464; and at drivers/firmware/raspberrypi.c; • hwmon: Failed to get throttled (-110); • raspberry-clk soc:firmware:clocks: Failed to change fw-clk-arm frequency: -110			
	Proposed Mitigation	Configure an external hardware watchdog to detect firmware hangs and trigger a reset (configure CONFIG_POWER_RESET=y in kernel configuration) via GPIO, for an immediate Linux reset; also implement a kernel hook to detect firmware errors with exponential backoff retries before triggering a hard reset.			
	Power	The Raspberry Pi Zero 2W failed to reboot during accelerated proton testing.	Reboot Failure	• boot/firmware, kernel/reboot.c • drivers/power/*, drivers/regulator/*	
1.3	Root Cause Hypothesis	Corruption in the boot ROM, firmware, or critical boot files destabilized system initialization, causing PMIC malfunctions, clock disruptions, and memory integrity violations.			
	Failure	Over-current protection (OCP) was triggered by an external power supply monitoring unit.			
	Proposed Mitigation	Prevent boot failures by implementing a TMR-led boot process with multiple SD card partitions (A, B, C) containing redundant copies of critical boot components (e.g., U-Boot, kernel image, dtb, initramfs) and verify file integrity with hash verification. Additionally, configure the external watchdog timer with a 30-second timeout for forceful hardware resets.			
		High			
	Drivers (dd/)	Error: dwc_otg_hcd_handle_rx_status _q _level_intr: Unable to get corresponding channel	USB OTG Error	• drivers/usb/host/dwc_otg/dwc_otg_hcd_intr.c; dwc_otg_hcd_handle_rx_status_q_level_intr()	
2.1	Root Cause Hypothesis	A bit-flip in the dwc_otg driver may corrupt critical data within the USB OTG controller's data structure, leading to invalid channel mappings and erroneous USB transactions.			
	Failure	FAT-fs (sda): Volume was not properly unmounted; some data may be corrupt; client_loop: send disconnect: Broken pipe.			
	Proposed Mitigation	Unload unused kernel drivers (e.g., USB_OTG) to reduce the memory attack surface. On recurring errors, employ exponential backoff for module reloads and trigger a system reset via a GPIO-activated external watchdog if error persists.			
	File System (fs/)	FAT-fs (sda): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.	Filesystem Mount Error	fs_fat/inode.c; fat_set_state()	
2.2	Root Cause Hypothesis	A bit-flip in RAM, CPU cache, or I/O operations when writing on SD card corrupts the filesystem metadata, affecting the boot sector or directory entries. Similar unexpected write interruptions also risk filesystem corruptions.			
	Proposed Mitigation	Enforce filesystem checks on reboot by appending fsck.mode=force to /boot/firmware/cmdline.txt. Use strict journaling or a self-repairing filesystem like Btrfs [67] with checksumming to ensure data integrity. Also, mount volatile data on temporary filesystem (tmpfs) to prevent corruption during interruptions.			
		Medium-to-Low			
	Kernel	CPU PID:(104,991) Comm:(kworker,stress-ng) tainted.	Kernel Taint	kernel/(panic.c, trap.c), kernel/core/taint.c, panic()	
3.1	Root Cause Hypothesis	A bit-flip induced corruption in kworker or stress-ng triggered kernel tainting, undermining system integrity though not immediately critical.			
	Proposed Mitigation	Employ process-level duplication using Linux's fork(), creating identical kernel stacks, thread_info, and task_struct for forked processes to isolated CPU cores via CPU affinity. Designate a voter on an independent core to perform majority voting on redundant data written to shared memory by the duplicated processes to mask discrepancies in computation.			

TABLE VII: Analyzing Soft Error-Induced Linux Failures on the i.MX 8M Plus EVK: Classification and Resilience Strategies

No.	Component	Linux Error Description	Fault Type	Register/Components	Impacted Subsystems
		Critical			
1.1	Memory (mm/)	Unable to handle kernel paging request at virtual address 0000800011006720	Kernel Paging Request	• arch/arm64/mm/fault.c, • kernel/(sched,signal.c) • mm/memory.c, • fs/coredump.c	• do_page_fault()
		systemd[1]: Caught <SEGV>, core dump failed (child 1733, code=killed, status=11/SEGV)	Segmentation Fault (SEGV)		
	Root Cause Hypothesis	Memory corruption in a child process (PID 1733), likely affected critical structures (e.g. stack pointer), leading to invalid memory access. This corruption increases the risk of dereferencing NULL pointers, triggering a segmentation fault (SIGSEGV).			
	Failures	FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.			
	Proposed Mitigation	• Periodically validate critical page table entries and software stacks while maintaining a redundant process state for recovery. • Protect critical processes with stack canaries, shadow stacks, and guard pages to mitigate execution failures. • Configure sigaction to handle signals like SIGSEGV, ensuring controlled process termination rather than a default core dump. • Enable CONFIG_RAS=y for ARM™ architecture to leverage built-in Reliability, Availability, and Serviceability (RAS) features. • Configure kexec to trigger a controlled reboot into a safe kernel after a panic timeout, preventing full system resets.			
1.2	Drivers (dd/)	watchdog: watchdog0: watchdog did not stop!	Watchdog Timer Failure	• drivers/watchdog/ • imx2_wdt.c	imx2_wdt_stop()
	Root Cause Hypothesis	Bit corruption may inadvertently modify the Watchdog Control Register (WDOGx_WCR) register, potentially triggering disabling of the watchdog timer during Linux runtime.			
	Proposed Mitigation	• Implement a custom handler to validate and restore the corrupted WDOGx_WCR register. • Use the i.MX8M Plus's dual Watchdog timer (WDOG1-WDOG2) for cross-verifying watchdog states from modifications.			
1.3	Drivers (dd/)	sdhci-esdhc-imx 30b6000.mmc: Card stuck in the wrong state! card_busy_detect status: 0xe00	eMMC Card Corruption	• pc:sdhci_send_command +0x4fc/0xeac • lr:sdhci_send_command _retry+0x4fc/0xeac	• drivers/mmc/host/sdhci-esdhc-imx.c • drivers/mmc/core/(block.c, mmc.c, core.c, mmc_ops.c) • __mmc_poll_for_busy(), • esdhc_wait_for_card_activity(), • _mmc_flush_cache(), • _mmc_hw_reset()
	Root Cause Hypothesis	A flipped bit in the eMMC controller's SDHCI_PRESENT_STATE register can falsely signal a busy status (e.g., CARD_BUSY, CMD_INHIBIT), potentially causing mmc_poll_for_busy() to deadlock and leading to card reset failures.			
	Cascading Linux Failures	• mmc2: Card stuck being busy! __mmc_poll_for_busy; • mmc2: cache flush error -110; • mmcblk2: recovery failed!; • mmc2: tried to HW reset card, got error -110; • EXT4-fs (mmcblk2): failed to convert unwritten extents to written extents – potential data loss! (inode 787934, error -30); • blk_update_request: I/O error, dev mmcblk2, sector 31348912 op 0x1: (WRITE) flags 0x4000 phys_seg 9 prio class 0.			
	Proposed Mitigation	• Set panic_on_oops=1 to configure the kernel to reboot on persistent eMMC errors (e.g., mmcblk2: recovery failed!) • Using a RAM-based temporary filesystem, such as tmpfs, can reduce eMMC reliance, also noted by Miller et al. [3].			
		High-to-Low			
2.1	Memory (mm/)	dockerd[783]: signal SIGBUS: bus error code=0x2 addr=0xaaaae17988c	Runtime Signal Interruption (e.g. SIGBUS)	• pc:0xaaaae046b64b, • mm/fault.c, • kernel/signal.c	• /usr/lib/go/src/runtime/panic.go, • handle_mm_fault()
		dockerd[783]: fatal error: unexpected signal during runtime execution			
	Root Cause Hypothesis	Invalid memory access by Docker daemon likely triggered SIGBUS. Additionally, corruption in kernel signal-handling structures (e.g. siginfo_t, or task_struct) may misroute signals, potentially affecting dockerd service.			
	Failures	Fatal error: unexpected signal during runtime execution panic during panic			
	Mitigation	Perform periodic memory scrubbing and regularly save process state for resumption after user-space fault occurrence.			
2.2	Filesystem (fs) + Drivers (dd/)	EXT4-fs error (device mmcblk2) in ext4_dirty_inode:5949: IO failure	eMMC Data Corruption & Journaling Failure	• fs/buffer.c • fs/ext4/ext2/ • jbd2.c • fs/ext4/extents.c • block/blk-core.c • fs/ext4/page-io.c • fs/ext4/super.c • fs/ext4/inode.c • fs/jbd2/journal.c • drivers/mmc/host/ sdhci-esdhc-imx.c • drivers/mmc/core/ mmc_blk.c	• ext4_dirty_inode() • ext4_journal_check_start() • blk_update_request(), • end_buffer_writesync(), • ext4_mark_inode_dirty(), • ext4_alloc_file_blocks(), • sdhci_request(), • jbd2_journal_update_sb, • _log_tail(), • ext4_end_bio(), • ext4_decode_error(), • ext4_commit_super(), • jbd2_write_superblock()
		blk_update_request: I/O error, dev mmcblk2, sector 31348912 op 0x0: (READ) flags 0x3000 phys_seg 1 prio class 0			
		JBD2: Error -5 detected when updating journal superblock for mmcblk2p1-8			
		ext4_end_bio:344: I/O error 10 writing to inode 787929 starting block 3118080			
		systemd[1]: docker.service: Main process exited, code=exited, status=1/FAILURE (and status=4/NOPERMISSION)			
		mpage_map_and_submit_extent: 2513: inode #395451: comm kworker/u8:2: mark_inode_dirty error			
		ext4_convert_unwritten_extents:4281: inode #385451: block 44592: len: 144: ext4_ext_map_blocks returned -30			
	Root Cause Hypothesis	These errors indicate that the eMMC hardware component is highly susceptible to radiation-induced failures, leading to corruption of various ext4 journal entries, superblocks, and the file system.			
	Proposed Mitigation	• Use a resilient, self-healing filesystem with in-built error correction, such as Btrfs, for enhanced data integrity alongside ECCs. • Harden the Linux kernel with the following kernel modules: • Error detection: (CONFIG_RAS, CONFIG_X86_MCE, CONFIG_MEMORY_FAILURE, CONFIG_EDAC) + recovery. • Mitigation: (CONFIG_CC_STACKPROTECTOR_STRONG, CONFIG_HARDENED_USERCOPY) for enhanced protection. • Recovery: (CONFIG_SOFTLOCKUP_DETECTOR, CONFIG_HARDLOCKUP_DETECTOR, and CONFIG_WATCHDOG).			

TABLE VIII: Analyzing Soft Error-Induced Linux Failures on the OrangeCrab v2 FPGA: Classification and Resilience Strategies

No.	Component	Linux Error Description	Fault Type	Kernel Component	Litex-SoC Component
		Critical			
1.1	Memory (mm/)	Unable to handle kernel NULL pointer dereference at virtual address 00000000	NULL Pointer Dereference	• sepc: 355b4e14 • ra:355b4e14 • sp:9ddd3ad0 • sstatus:00000100 • sbadaddr:00000000 • scause:0000000d	• litex/soc/cores /cpu/vexriscv/core.py • litex/soc/cores/ ecc.py (set ecc=True) • riscv/mm/fault.c • do_page_fault()
		Unable to handle kernel paging request at virtual address c00240c0	Kernel Paging Failure		
		Segmentation Fault (SIGSEGV)	Segmentation Fault		
	Root Cause Hypothesis	A flipped bit in RAM or CPU registers due to SEEs can corrupt a valid page table entry (PTE), potentially leading into NULL or an invalid address. When the kernel subsequently attempts to dereference the corrupted pointer, such as when accessing struct task_struct, it may result in a crash or a segmentation fault.			
	Failures	CPU: 0 PID: 843 Comm: sh Not tainted 5.0.13 #1			
	Proposed Mitigation	• Enable ECC functionality (like SECDED) in LiteX's LiteDRAM module [68]. This requires modifying the RTL generation process, including changes in litex/soc/cores/cpu/vexriscv/vexriscv.py, to incorporate ECC protection for critical registers within the VexRiscv CPU.			
		• For systems with higher resource availability and stringent reliability requirements, fault tolerance can be enhanced with double or triple replication (DMR/TMR) across the LiteX-based SoC. This includes replication of CPUs, memory controllers, and buses. Refer to Tables VI and VII for similar mitigation strategies.			
1.2	Kernel	Kernel panic – not syncing: Fatal exception in interrupt	Interrupt Handler	• kernel/traps.c • handle_arch_irq() • do_trap() • do_trap_irq()	• litex/soc/cores /cpu/vexriscv/core.py • litex/soc/cores/ watchdog.py
	Root Cause Hypothesis	A kernel panic in an interrupt context signifies a fatal exception, likely due to corruption of critical registers, such as those in the interrupt controller, leading the CPU to reference an invalid Interrupt Status Register (ISR) address and triggering a system crash.			
	Proposed Mitigation	• Implement ECC memory to safeguard critical interrupt controller registers, including the Platform-Level Interrupt Controller (PLIC), which manages external interrupts in RISC-V architectures.			
		• Utilize LiteX's watchdog timer to detect Linux system hangs and trigger a reset if the system remains unresponsive.			
		High-to-Low			
2.1	Memory (mm/)	BUG: Bad page map in process md5sum pte:10444cd7 pmd:11bb7001|flags:0x0()	Corrupted Page Map	Call Trace: walk_stackframe+0x0/0xfc show_stack+0x3c/0x50 dump_stack+0x2c/0x3c print_bad_pte+0x170/0x1c4 unmap_page_range+0x428/0x59 unmap_single_vma+0x40/0x60 unmap_vmas+0x50/0x84 exit_mmap+0xb0/0x174 mmput+0x2c/0x3c do_exit+0x1cc/0x7cc do_group_exit+0x38/0x94 _wake_up_parent+0x0/0x38 ret_from_syscall+0x0/0x10	• mm/(fault.c,memory.c pageattr.c, pgtable.c page_alloc.c) • kernel/traps.c • do_page_fault • set_pte_at() • set_pmd() • bad_page() • litex/soc/cores /cpu/vexriscv/core.py • litex/soc/cores/ watchdog.py • liteDRAM/core/ controller.py
		Page dumped because: bad pte/ nonzero mapcount • file:ld-2.29.so • fault:filemap_fault • mmap:generic_file_map			
		BUG: Bad page state in process kthread pfn:46f5a			
	Root Cause Hypothesis	The kernel detected an invalid PTE(0x10444cd7) and PMD:(0x11bb7001) during a page unmap operation, indicating potential corruption in the page table hierarchy. A nonzero mapcount suggests improper page reference tracking, possibly leading to unexpected page faults or kernel panic.			
	Proposed Mitigation	Maintain redundant copies of all critical registers storing PTE and PMD values, implement parity checks for PTEs/PMDs, and invalidate the TLB upon PTE repair.			
2.2	Kernel	Oops [#00001]	Kernel Oops	• kernel/traps.c • handle_trap()	• litex/soc/cores /cpu/vexriscv/core.py
	Root Cause Hypothesis	The Linux kernel triggers an Oops upon encountering an unrecoverable error, allowing continued Linux operation in a degraded state which may lead to instability without triggering a kernel panic.			
	Failures	Unable to handle kernel NULL pointer dereference at virtual address 0000			
	Proposed Mitigation	• At node-level, run TMRed Linux instances across three processors to mitigate erroneous outputs, similar to the architecture used in SpaceX's Falcon 9 [69]. • At the process level, implement DMR/TMR with lockstep execution for critical processes on multicore SoCs. Refer to Table VI, Section 3.1, *Proposed Mitigation* for a similar approach.			
2.3	Memory (mm/)	md5sum[1140]: unhandled signal 11 code 0x1 at 0x3ffcfb60 in ld-2.29.so	Segmentation Fault	• sstatus: 00000020 • scause: 0000000d (decimal 13) • sbadaddr: 9feb0bcf	• kernel/signal.c • mm/fault.c • do_trap_user() • do_page_fault() • litex/soc/ integration/soc_core.py • litex/soc/ interconnect/csr_bus.py
		find[123]: unhandled signal 11 code 0x1 at 0x9fe0bc0f in busybox[10000+cf000]			
		md5sum: can't read '/sys/kernel/slab/ kmalloc-8/(alloc_calls, free_calls)	SysFS Corruption	• mm/slub.c • alloc_calls_show() • sysfs_slab_add()	
	Root Cause Hypothesis	Signal 11, also known as SIGSEGV in Linux denotes a segmentation fault. The error code 0x1 further indicates SEGV_MAPERR, that fault resulted from a process trying to access an unmapped address (0x3ffcfb60).			
	Proposed Mitigation	To handle inconsistencies in shared libraries, perform manual unloading of affected modules using (dlclose()), reloading with (dlopen()), on failure to load, use a preset alternate library path using the LD_PRELOAD.			

REFERENCES

[1] Branka, "Linux statistics – 2024." Available: https://truelist.co/blog/linux-statistics/, 2024. Accessed: Feb. 23, 2025.

[2] Sam, "Android's latest statistics 2025: How many people have androids?." Available: https://avada.io/blog/how-many-people-have-androids/, 2025. Accessed: 2025-02-20.

[3] E. Miller, C. Heistand, and D. Mishra, "Space-operating linux: An operating system for computer vision on commercial-grade equipment in leo," in *2023 IEEE Aerospace Conference*, pp. 1–12, IEEE, 2023.

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

[5] T. Beck, *Evaluation and analysis of Linux-based applications on multicore processor in a space environment*. Ph.d. thesis, ISAE-Institut Supérieur de l'Aéronautique et de l'Espace, 2024.

[6] B. Ramesh, T. Bretschneider, and I. McLoughlin, "Embedded linux platform for a fault tolerant space based parallel computer," in *Proceedings of the Real-Time Linux Workshop*, pp. 39–46, 2004.

[7] S. S. Prieto, I. G. Tejedor, D. Meziat, and A. V. Sánchez, "Is linux ready for space applications," in *Open Source World Conference; Junta de Andalucia: Mlaga, Spain*, 2004.

[8] A. F. S. Command, "The future of space 2060 and implications for us strategy: Report on the space futures workshop (colorado springs, co: Afspc, september 5, 2019), 6–9."

[9] M. N. Sweeting, "Modern small satellites-changing the economics of space," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 343–361, 2018.

[10] W. Peeters, "The paradigm shift of newspace: New business models and growth of the space economy," *New Space*, vol. 12, no. 3, pp. 202–213, 2024.

[11] I. Sehovic, "The newspace race: the private space industry and its effect on public support for nasa funding," tech. rep., working paper, March 2018, 52 p, available at https://www. academia. edu . . . , 2018.

[12] B. Balaram, T. Canham, C. Duncan, H. F. Grip, W. Johnson, J. Maki, A. Quon, R. Stern, and D. Zhu, "Mars helicopter technology demonstrator," in *2018 AIAA atmospheric flight mechanics conference*, p. 0023, 2018.

[13] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and materials reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[14] J. Budroweit and H. Patscheider, "Risk assessment for the use of cots devices in space systems under consideration of radiation effects," *Electronics*, vol. 10, no. 9, p. 1008, 2021.

[15] J. Estela, "Cots and the newspace," *Radiation Effects on Integrated Circuits and Systems for Space Applications*, pp. 329–346, 2019.

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

[17] J. Freschl and D. Xue, "Swif-it: A tool for memory fault injection and protection," 2005.

[18] SpaceX, "Starlink satellite constellation." Online. Available: https://www.starlink.com/, 2025.

[19] J. McDowell, "Starlink statistics," 2025. Accessed: 2025-02-12.

[20] T. Verge, "Starlink surpasses 4 million users," September 2024.

[21] Z. Staff, "Spacex: We've launched 32,000 linux computers into space for starlink internet," 2024. Accessed: 2024-12-01.

[22] SpaceX, "Spacex uses linux and x86 processors in the falcon 9," *Aerospace Technology Review*, 2025. Accessed: 2025-02-19.

[23] S. Katz, U. Goldvais, and C. Price, "The connection between space weather and single event upsets in polar low earth orbit satellites," *Advances in Space Research*, vol. 67, no. 10, pp. 3237–3249, 2021.

[24] J. Edge, "Elc: Spacex lessons learned," *Linux Weekly News*.

[25] T. Krol, *A generalization of fault-tolerance based on masking*. PhD thesis, Technische Universiteit Eindhoven, 1991. PhD Thesis 1 (Research TU/e / Graduation TU/e), Electrical Engineering.

[26] Federal Communications Commission, "FCC Order for OneWeb Phase 2 Expansion to 6,372 Satellites." Federal Communications Commission, DA-22-970A1, 2022. [Online]. Available: urlhttps://docs.fcc.gov/public/attachments/DA-22-970A1.pdf.

[27] Federal Communications Commission, "FCC Authorization for Project Kuiper 3,236 Satellites." Federal Communications Commission, FCC-20-102A1, 2020. [Online]. Available: urlhttps://docs.fcc.gov/public/attachments/FCC-20-102A1.pdf.

[28] R. Ecoffet, "Overview of in-orbit radiation induced spacecraft anomalies," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1791–1815, 2013.

[29] R. Ecoffet *et al.*, "On-orbit anomalies: Investigations and root cause determination," in *IEEE NSREC*, vol. 1198, 2011.

[30] R. Baumann and K. Kruckmeyer, *Radiation handbook for electronics, available on TI.com (https://www.ti.com/applications/industrial/aerospace-defense/technical-documents.html)*. 01 2019.

[31] A. H. Johnston *et al.*, "Scaling and technology issues for soft error rates," in *Proc. of the 4th Annual Conf. on Reliability*, Citeseer, 2000.

[32] V. Pouget, "Single-event effects in advanced cmos technologies," in *European Conference on Radiation and its Effects on Components and Systems (RADECS) Short Course*, 2023.

[33] Unikraft Project, "Linux kernel components have strong inter-dependencies, making it difficult to remove or replace them (Figure 1)." https://unikraft.org/docs/concepts/design-principles. Accessed: 2025-01-17.

[34] C. J. Corley, H. M. Quinn, and E. E. Swartzlander, "Accelerated nuclear radiation effects on the raspberry pi 3b+," in *2022 IEEE Radiation Effects Data Workshop (REDW)(in conjunction with 2022 NSREC)*, pp. 1–8, IEEE, 2022.

[35] G. S. Rodrigues, F. Rosa, Á. B. de Oliveira, F. L. Kastensmidt, L. Ost, and R. Reis, "Analyzing the impact of fault-tolerance methods in arm processors under soft errors running linux and parallelization apis," *IEEE Transactions on Nuclear Science*, vol. 64, no. 8, pp. 2196–2203, 2017.

[36] R. Ginosar, "Survey of processors for space," *Data Systems in Aerospace (DASIA). Eurospace*, pp. 1–5, 2012.

[37] M. Riera, R. Canal, J. Abella, and A. Gonzalez, "A detailed methodology to compute soft error rates in advanced technologies," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 217–222, IEEE, 2016.

[38] C. J. Corley, H. M. Quinn, and E. E. Swartzlander, "Accelerated nuclear radiation effects on the raspberry pi 3b+ and pi 4," in *2023 IEEE Radiation Effects Data Workshop (REDW)(in conjunction with 2023 NSREC)*, pp. 1–9, IEEE, 2023.

[39] S. M. Guertin, S. Vartanian, and A. C. Daniel, "Raspberry pi zero and 3b+ see and tid test results," in *2022 IEEE Radiation Effects Data Workshop (REDW)(in conjunction with 2022 NSREC)*, pp. 1–5, IEEE, 2022.

[40] S. M. Guertin, "Raspberry pis for space guideline."

[41] S. M. Guertin, "Board level proton testing book of knowledge for nasa electronic parts and packaging program," 2017.

[42] E. J. Wyrwas, "Proton testing of nvidia jetson tx1," tech. rep., 2017.

[43] E. J. Wyrwas, "Proton testing of nvidia jetson tx2," tech. rep., 2019.

[44] I. Wali, A. Sánchez-Macián, A. Ramos, and J. A. Maestro, "Analyzing the impact of the operating system on the reliability of a risc-v fpga implementation," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, IEEE, 2020.

[45] "Colinianking/stress-ng: Stress-ng will stress test a computer system in various selectable ways. it was designed to exercise various physical subsystems of a computer as well as the various operating system kernel interfaces.." https://github.com/ColinIanKing/stress-ng. (Accessed on 08/26/2023).

[46] K. A. LaBel, "Proton single event effects (see) guideline," *Submitted for publication on the NASA Electronic Parts and Packaging (NEPP) Program web site*, 2009.

[47] *Absorbed Dose Determination in External Beam Radiotherapy*. No. 398 in Technical Reports Series, Vienna: INTERNATIONAL ATOMIC ENERGY AGENCY, 2001.

[48] J. F. Ziegler and J. P. Biersack, "The stopping and range of ions in matter," in *Treatise on heavy-ion science: volume 6: astrophysics, chemistry, and condensed matter*, pp. 93–129, Springer, 1985.

[49] "Raspberry Pi Zero 2 W." https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/. (Accessed on 08/03/2023).

[50] J. Mojica Decena, B. Wood, R. J. Martineau, M. Tayor, and J. Dennison, "Radiation damage threshold of satellite cots components: Raspberry pi zero for opal cubesat," in *Utah State University Student Research Symposium 2018*, 2018.

[51] "NXP i.MX 8M Plus - verdin system on module." https://www.toradex.com/computer-on-modules/verdin-arm-family/nxp-imx-8m-plus. (Accessed on 08/03/2023).

[52] "Portainer: Container management software for kubernetes and docker." https://www.portainer.io/. (Accessed on 02/01/2024).

[53] "Raspberry pi documentation - processors." https://www.raspberrypi.com/documentation/computers/processors.html. (Accessed on 08/26/2023).

[54] A. Chou, J. Yang, B. Chelf, S. Hallem, and D. Engler, "An empirical study of operating systems errors," in *Proc. 18th ACM Symp. on Operating Systems Principles*, pp. 73–88, 2001.

[55] "Orangecrab." https://orangecrab-fpga.github.io/orangecrab-hardware/r0.2/. (Accessed on 08/03/2023).

[56] "Github - spinalhdl/vexriscv: A fpga friendly 32 bit risc-v cpu implementation." https://github.com/SpinalHDL/VexRiscv. (Accessed on 08/03/2023).

[57] "Github - enjoy-digital/litex: Build your hardware, easily!." https://github.com/enjoy-digital/litex. (Accessed on 08/03/2023).

[58] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch, "A catalog and in-hardware evaluation of open-source drop-in compatible risc-v softcore processors," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–8, IEEE, 2019.

[59] "Buildroot - making embedded linux easy." https://buildroot.org/. (Accessed on 08/03/2023).

[60] JEDEC Solid State Technology Association, "Single Event Functional Interrupt (SEFI)," 2025. Accessed: 2025-01-14.

[61] A. Javanainen, "Fysv445 (ph1) radiation interactions in electronic materials, jss28, jyu, finland (2018)." http://research.jyu.fi/radef/JSS28/PH1/#1, 2018. (Accessed on 02/05/2024).

[62] W. E. Ricker, "The concept of confidence or fiducial limits applied to the poisson frequency distribution," *Journal of the American Statistical Association*, vol. 32, no. 198, pp. 349–356, 1937.

[63] NXP Semiconductors, "Nxp debuts i.mx neural processing." https://www.nxp.com/company/about-nxp/newsroom/NW-DEBUT-IMX-NEURAL-PROCESSING. Accessed: 2025-01-20.

[64] A. J. Tylka, J. H. Adams, P. R. Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, E. L. Petersen, M. A. Shea, D. F. Smart, and E. C. Smith, "Creme96: A revision of the cosmic ray effects on micro-electronics code," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2150–2160, 1997.

[65] Vanderbilt University, ISDE, "Creme-mc weibull distribution help," 2025. Accessed: 2025-02-21.

[66] L. Ferraz, "Performance Impact of Docker Containers on Linux." Available: https://www.torizon.io/blog/containers-in-linux-and-performance-impact, Dec. 13, 2022. Accessed: Apr. 3, 2025.

[67] C. Mason and the Btrfs Community, "Btrfs - a modern copy-on-write filesystem for linux." https://btrfs.wiki.kernel.org, 2007. Accessed: 2025-03-02.

[68] EnjoyDigital, "Litedram ecc module." https://github.com/enjoy-digital/litedram/blob/master/litedram/frontend/ecc.py, 2024. Accessed: 2024-10-23.

[69] N. Rajadurai, "Fault tolerant system design patterns for spacex falcon 9 flight software," *EmbeddedRelated*, 2020. Accessed: 2024-10-23.

Jan Swakoń Ph.D., D.Sc., professor of IFJ PAN, is Head of the Department of Radiation Research and Proton Radiotherapy at The H. Institute of Nuclear Physics, Polish Academy of Sciences in Krakow, Poland. He was responsible for building the proton eye radiotherapy facilities at the IFJ PAN and preparing the first proton ocular radiotherapy in Poland. His main scientific activity focuses on new methods of proton radiotherapy, dosimetry, medical and radiation physics including proton and gamma irradiations for space and high-energy physics applications.

Damian Wrobel graduated in technical physics from AGH University of Science and Technology. Currently, he is a PhD student at the Institute of Nuclear Physics Polish Academy of Sciences. His doctoral research is focused on dosimetry for proton FLASH radiotherapy. In particular, he is developing a proton beam forming system and applying radioluminescent crystals for dose determination in ultra-high dose rate proton irradiations.

Sebastian Kusyk is a PhD student in the Institute of Nuclear Physics, Polish Academy of Sciences since 2022. Subject of doctoral thesis: Proton beam dosimetry for electronics radiation hardness tests (development of proton irradiation station at AIC-144 cyclotron, beam forming, and dosimetry). Proton and gamma (Co-60) irradiations of electronics, new passive dosemeters (i.e., TLD, OSL), biological and other samples. Graduate of Advanced Materials and Nanotechnology at Jagiellonian University.

Saad Memon received the Erasmus Mundus European Double M.Sc. degree in Dependable Software Systems from the University of St. Andrews, UK, in 2019, with a specialization in Artificial Intelligence. He is currently pursuing a Ph.D. as a Doctoral Researcher with the SERVAL group at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. His research interests include hardware-software co-design for radiation-resilient space systems, focusing on leveraging COTS SoCs to develop robust computer architectures and operating systems for AI/ML applications.

Seth Roffe received a B.S. degree in physics, astronomy, and mathematics, and M.S. and Ph.D. degrees in electrical and computer engineering from the University of Pittsburgh, Pittsburgh, PA, USA. He is currently a Civil Servant with NASA Goddard Space Flight Center and a part of the Radiation Effects and Analysis Group, primarily performing research on the radiation effects of complex devices and machine-learning applications. His research interests involve resilience in sensor processing, including data reliability and error classification in novel sensors and processors or accelerators.

Rafal Graczyk received the B.Sc. and M.Sc. degrees in electronics and computer engineering and the Ph.D. degree in electronics from Warsaw University of Technology, in 2005, 2009, and 2017, respectively. He's got more than 15 years of experience in dependable systems research and development, with a focus on resilience and radiation effects in computer systems. He has been active in wide range of projects, conducted with the Space Research Center of the Polish Academy of Sciences, Interdisciplinary Centre for Security, Reliability and Trust at University of Luxembourg, and, as of recently, companies within European space sector, where he assumes technical leadership roles.

Mike Papadakis is an associate professor at the Interdisciplinary Center for Security, Reliability and Trust (SnT) of the University of Luxembourg. He received a Ph.D. degree in Computer Science from the Athens University of Economics and Business in 2011. His research interests include software testing, static analysis, prediction modeling, and search-based software engineering. He is best known for his work on Mutation Testing, for which he has been awarded the IEEE TCSE Rising Star Award 2020. He has been awarded several ACM SIGSOFT Distinguished Paper and Artifact Awards and a Facebook Research Award (2019). He is the general (co-)chair of the 37th IEEE International Conference on Software Maintenance and Evolution (ICSME 2021); he has been elected to the steering committees of the IEEE International Conference on Software Testing (ICST), the Symposium on Search-Based Software Engineering (SSBSE) and serves on the editorial and review boards of international Software Engineering journals (STVR, Empirical Software Engineering, ACM Transactions on Software Engineering and Methodology). He has (co-)authored more than 80 publications in international peer-reviewed conferences and journals. His work has been supported by Facebook, FNR, CETREL (SIX group company), BGL (BNP Paribas), Microsoft, and PayPal.

Tomasz Rajkowski graduated from the Electronics Department at Warsaw University of Technology. From 2011 to 2017, he worked at the Space Research Center of the Polish Academy of Sciences as an electronics test engineer (BRITE-PL project) and later as a hardware designer and project manager (ESA OPS-SAT project). Between 2018 and 2020, he completed his PhD at the University of Montpellier, conducting research at 3D-Plus on the radiation qualification of space systems. Currently, he is a postdoctoral researcher at the National Centre for Nuclear Research, specializing in radiation hardness assurance and serving as an engineer and consultant for several national and European space projects.