International Journal of Aeronautical and Space Sciences
https://doi.org/10.1007/s42405-026-01130-w

IJASS International Journal of Aeronautical and Space Sciences

ORIGINAL PAPER


# In-Orbit Validation of a Heterogeneous COTS Architecture for Onboard AI: System-Level Operational Vulnerabilities and Failure Pathways

**Dongshik Won[1,2] · Kyungwoo Hong[1,4] · Dongwoo Lee[2] · Heoncheol Lee[3] · Hyochoong Bang[2]**

Received: 13 August 2025 / Revised: 30 December 2025 / Accepted: 19 January 2026


**Abstract**
The adoption of Commercial-Off-The-Shelf (COTS) processors for in-orbit AI is limited by operational reliability risks with limited in-orbit data. This paper reports an in-orbit validation of a heterogeneous architecture combining a Microchip PolarFire SoC and an NVIDIA Jetson Orin NX. In this on-orbit dataset, many observed faults stem from system integration and operational protocol-driven degradation rather than from isolated radiation events. The PolarFire-based system instability aligned with a fragile I/O boot chain at the storage interface rather than the SoC silicon. The Jetson-based system exhibited an end-of-life cascade consistent with cumulative storage degradation under repeated abrupt power cycles. In this paper, the term vulnerability refers to operational reliability failure modes rather than cyber-security exploits. This paper does not assess cyber-security vulnerabilities in open-source or COTS software. The results support a heterogeneous architecture where a resilient Supervisory Node manages recovery actions for a high-performance accelerator.

**Keywords** Onboard computing · COTS · Heterogeneous · Architecture · System-Level · Vulnerability · In-Orbit Demonstration · NVIDIA Jetson

## 1 Introduction

The rapid growth and widespread adoption of CubeSats in the New Space Economy is undeniable. Industry tracking shows more than 2000 CubeSats had been launched by January 2024, with forecasts predicting nearly two thousand more in the subsequent four years alone [1]. At the heart of these missions, the On-Board Computer (OBC) plays a critical role, evolving from a simple command and data handling unit to a sophisticated processing hub for increasingly complex operations [2]. A comprehensive survey of this trend can be found in [3]. A pressing demand drives this evolution for high-performance, autonomous onboard computing to support applications that require a low-latency response. These include time-sensitive tasks such as real-time maritime monitoring for ship detection [4, 5] and anomaly detection for environmental threats like oil spills [6]. Another critical area is dynamic network routing in large LEO constellations, where onboard reinforcement learning models must adapt to topological changes in real-time, a task that places significant computational demands on the OBC [7]. Additionally, "Direct-to-device" (D2D) satellite communications reduce latency and infrastructure dependence, empowering applications such as disaster response, real-time sensor data

Communicated by Dong-Hyun Cho.

✉ Hyochoong Bang
hcbang@kaist.ac.kr

Dongshik Won
dwon@kaist.ac.kr

Kyungwoo Hong
ruddn2445@kaist.ac.kr

Dongwoo Lee
cin6474@kaist.ac.kr

Heoncheol Lee
hclee@kumoh.ac.kr

1 Future Innovation Research Team, TelePIX Co., Ltd., 17, Techno 4-ro, Yuseong-gu, Daejeon 34013, South Korea

2 Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Daejeon 34141, South Korea

3 Department of IT Convergence Engineering, School of Electronic Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi, Gyeongbuk 39177, South Korea

4 Present Address: Agency for Defense Development, Daejeon, South Korea

Published online: 24 February 2026

streaming, and remote monitoring [8] where immediate data processing in orbit is critical for effective action. The need for such capability is fueled by many factors: the sheer volume of data generated by modern sensors threatens to overwhelm traditional ground station pipelines, creating a significant "downlink bottleneck" [9], while the increasing frequency of climate-related events necessitates faster response times than a ground-in-the-loop architecture provides. This operational paradigm shift is explored in missions like SpIRIT [10] and is a key driver for constellations like CYGNSS [11].

To meet this computational demand, mission designers are increasingly turning to Commercial-Off-The-Shelf (COTS) components [12]. High-performance processors like the NVIDIA Jetson series are being adopted for AI acceleration in missions like MOCI and SpIRIT [10, 13], and are slated for use in next-generation commercial constellations such as Planet's Pelican fleet to enable real-time, on-board data analysis [14, 15]. Meanwhile, reliable, flash-based System-on-Chips (SoCs) like the Microchip PolarFire are favored for dependable supervision in proposed heterogeneous systems like CHICS [16]. Deploying these COTS components is non-trivial. The harsh space radiation environment presents a well-understood challenge, where cumulative effects like Total Ionizing Dose (TID) and transient Single-Event Effects (SEEs) can induce performance degradation and operational faults [17]. A primary contributor to this environment in low Earth orbit is the South Atlantic Anomaly (SAA), a region of unusually high radiation flux. However, beyond this environmental threat, deploying complex COTS systems introduces significant system-level and operational risks that remain poorly characterized by in-orbit data. These include integration faults in boot chains, software stack fragility, and vulnerabilities tied to power management protocols, which can ultimately be more detrimental to mission success than transient radiation events. In this paper, the term vulnerability refers to operational reliability failure modes, with a focus on boot chain fragility and storage corruption. It does not refer to cyber-security exploits. We did not perform a cyber-security assessment of the open-source or COTS software stack. Frameworks to address these dependability challenges are an area of active research [12, 18], as are methods for onboard anomaly detection [19–21] and predictive maintenance [22, 23].

Pioneering missions have successfully demonstrated the feasibility of deploying COTS-based accelerators for specific tasks. ESA's landmark Φ-Sat-1 mission, for instance, used an Intel Movidius Myriad 2 Vision Processing Unit (VPU) to perform onboard cloud detection, effectively filtering unusable data before downlink [24]. This mission is often cited alongside its sibling concept paper [25]. Similarly, the OPS-SAT mission has served as a flying laboratory for validating numerous software experiments on a flexible, COTS-based platform [26]. An overview of its extensive experimental campaigns can be found in [27]. Building on this foundation, development of more advanced systems continues. The Intuition-1 mission recently achieved the first in-orbit demonstration of a deep neural network on a COTS FPGA-based accelerator (a Xilinx Zynq UltraScale+ MPSoC) [28], while systems like NASA's SC-LEARN card, featuring a Google Coral Edge TPU, are being developed to bring advanced AI frameworks to CubeSats [29]. This follows a broader trend of adapting terrestrial AI hardware for space applications [30–33]. These efforts, along with many others summarized in Table 1, confirm a clear and growing trend towards deploying capable COTS accelerators in space.

While these missions prove the potential of individual COTS components, a critical gap remains in published, empirical in-orbit data. Specifically, there is a lack of reporting that characterizes the complete operational failure pathways of these complex COTS systems.

Existing studies often prioritize environmental resilience or architectural proposals, frequently overlooking system-level vulnerabilities like boot chain fragility or cumulative degradation from power cycling. These operational factors can dominate the risk profile in real-world missions. Without empirical data on these failure modes, the justification for heterogeneous architectures remains largely theoretical.

This paper addresses this specific gap by presenting the first comprehensive in-orbit performance results from a heterogeneous payload combining the Microchip PolarFire SoC and the NVIDIA Jetson Orin NX. We make the following contributions:

1. We identify and analyze a critical system-level vulnerability, a fragile I/O boot chain, as the primary driver of instability in a PolarFire SoC-based system, demonstrating how integration faults can override theoretical component reliability.
2. We document the end-of-life cascade of the Jetson Orin NX system, revealing a predictable degradation signature driven by cumulative data corruption from operational protocols, a failure mechanism distinct from transient environmental events.
3. Using this data, we provide the first direct, in-orbit evidence that a heterogeneous architecture is a comprehensive strategy for managing not just environmental threats, but the entire operational risk profile of COTS systems, including predictable degradation and integration faults.

The remainder of this paper is structured to present our findings in a linear narrative. Section 2 details the experimental hardware and software architecture. Section 3 describes the pre-launch and in-orbit test campaigns. Section 4 presents the in-orbit performance and stability results, while Sect. 5 provides a detailed analysis of the observed failure pathways.

**Table 1** summary of pioneering COTS-based onboard processing missions and systems

| Mission/system name | Key COTS Accelerator(s) | Stated objective/application | Key finding/status | References |
|---|---|---|---|---|
| *In-orbit demonstrations with published results* | | | | |
| Φ-Sat-1 | Intel Movidius Myriad 2 VPU | Onboard cloud detection to filter unusable hyperspectral data | Pioneering mission; successfully demonstrated the first in-orbit deep neural network on a VPU | [24] |
| Intuition-1 (Leopard DPU) | Xilinx Zynq UltraScale+ MPSoC | Onboard hyperspectral image processing and segmentation using deep neural networks | First in-orbit demonstration of a deep NN on a COTS FPGA-based accelerator; launched late 2023 | [28] |
| SpIRIT (Loris Payload) | NVIDIA Jetson Nano | Onboard computer vision experiments and innovative image compression techniques | In-orbit mission; highlights the use of COTS for cost-effectiveness but notes the associated reliability risks | [10] |
| OPS-SAT | FPGA-based Platform | Flying laboratory for validating numerous software-defined radio and processing experiments | Successfully validated a wide range of algorithms, proving the flexibility of a COTS-based reconfigurable platform | [26] |
| *Advanced hardware development and proposed systems* | | | | |
| Pelican Constellation | NVIDIA Jetson AGX Orin | Enhance imaging capabilities with real-time, onboard AI data processing and analytics | Major commercial adoption; announced for upcoming launches to provide faster insights to customers | [14, 15] |
| SC-LEARN | Google Coral Edge TPU | High-performance, power-efficient AI card for Earth science applications (e.g., vegetation classification) | Ground-based development and testing; designed to NASA's CubeSat standards for future flight demos | [29] |
| MOCI | NVIDIA Jetson TX2i | GPU-accelerated flight computer for an educational nanosatellite | Proposed architecture; one of the first CubeSats to incorporate a Jetson TX2i into its core design | [13] |
| HyTI (SpaceCloud iX5) | AMD G-Series SoC | Onboard science data product generation for a hyperspectral thermal imaging mission | Flown mission (launched 2024); system designed for onboard Big Data analytics and AI processing | [17] |
| *Proposed Heterogeneous Architectures (Lacking Comparative In-Orbit Data)* | | | | |
| SpaceCube V3.0 | Zynq UltraScale+ (HPN) & RTAX FPGA (RCN) | High-performance, fault-tolerant computing platform for NASA science applications | Architectural proposal and ground validation; combines COTS performance with rad-tolerant supervision | [34] |
| CHICS | Zynq UltraScale+ (HPN) & PolarFire FPGA (RCN) | Dependable framework for mixed-criticality applications, partitioning safe and non-safe functions | Architectural proposal and development; explicitly pairs a high-performance SoC with a flash-based supervisor | [16] |
| ScOSA | Zynq 7020 (HPN) & LEON3 Core on FPGA (RCN) | Modular, scalable architecture combining COTS performance with a reliable computing node | Architectural proposal and development at DLR, targeting reconfigurable high-performance computing | [35] |

**Table 2** hardware specification and design philosophy of onboard processing boards

| ID | Core processor | Memory | Design heritage | Primary role in experiment |
|---|---|---|---|---|
| DIB | Microchip PF SoC (MPFS250T) | 1 GB LPDDR4 (CPU) + 2 GB DDR4 (FPGA) | Custom, Application-Specific | Baseline Bare-Metal Implementation |
| APB | Microchip PF SoC (MPFS250T) | 1 GB LPDDR4 (CPU) | Reference-Derived | *Supervisory Node* (Linux on Micorhip PF SoC) |
| IPB | NVIDIA Jetson Orin NX 16GB | 16 GB LPDDR5 (Unified) | COTS Module Integration | *High-Throughput COTS Accelerator* (GPU-enabled) |

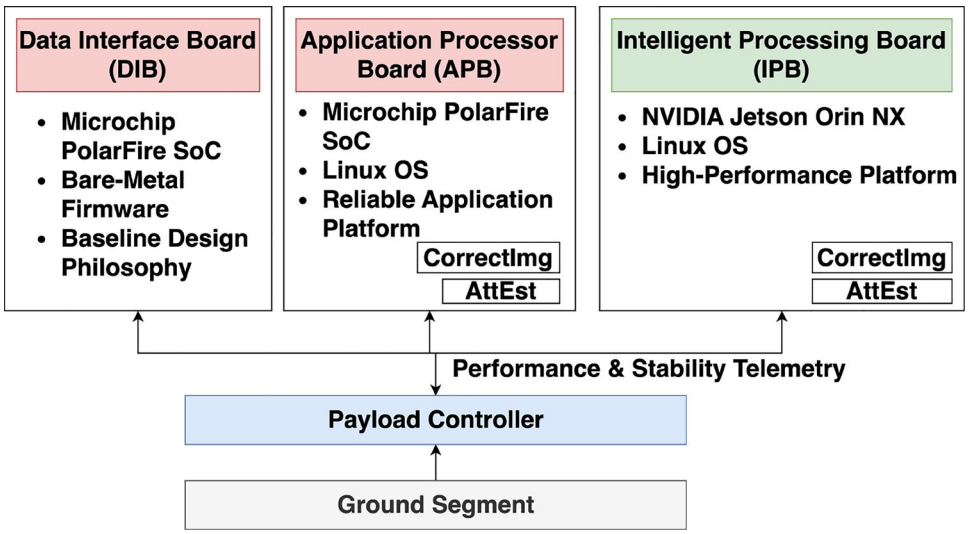



**Fig. 1** system architecture for the parallel in-orbit evaluation of three independent COTS processing modules. `CorrectImg` and `AttEst` are workload names executed on the nodes, not processor names.

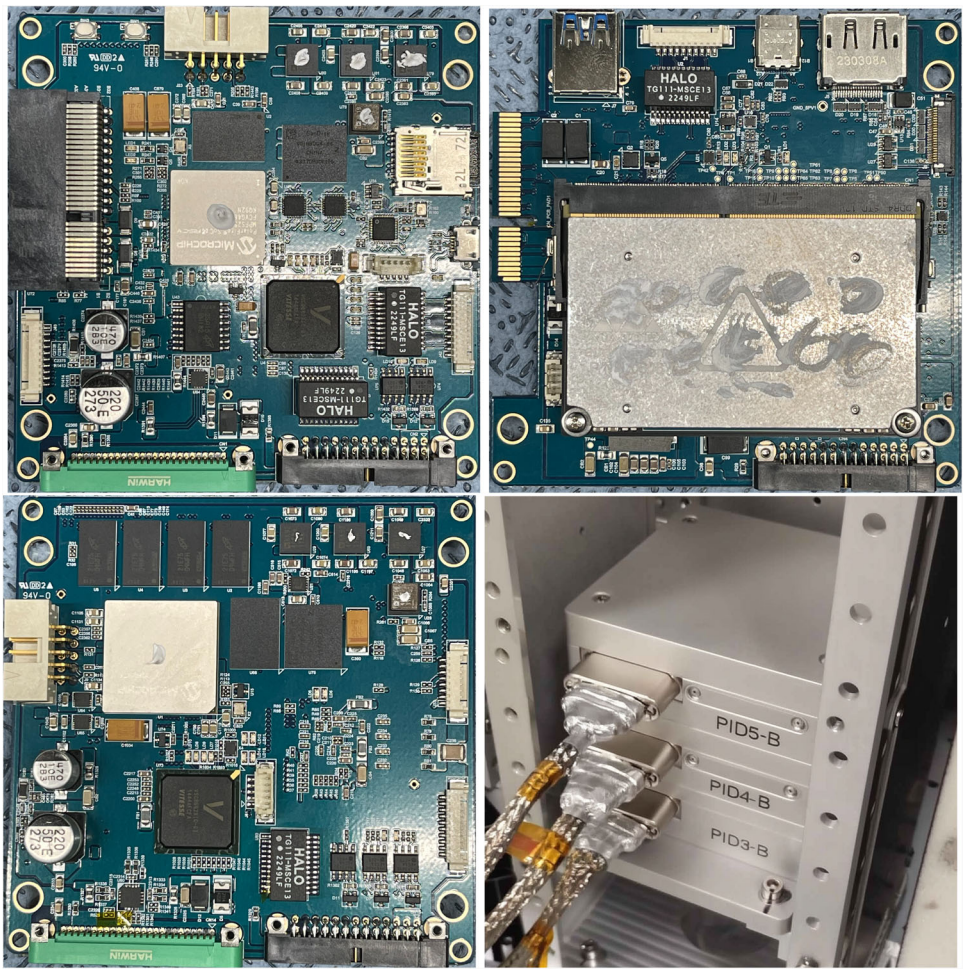



**Fig. 2** Flight hardware photographs

Finally, Sect. 6 discusses the strategic implications of these findings, and Sect. 7 summarizes the work and outlines future research directions.

## 2 System Architecture and Design Rationale

The core of our methodology is an architecture designed to evaluate three distinct, parallel philosophies for COTS-based processing in a single mission.

This experiment required three independent processing modules to deconstruct system-level failures and isolate variables. A single-board experiment would conflate failure modes, making it impossible to distinguish between silicon-level faults, I/O interface fragility, and software stack instability. Our three-board approach, depicted in Fig. 1, establishes this necessary separation: the Data Interface Board (DIB) provides a bare-metal baseline, the Application Processor Board (APB) represents a Supervisory Node, and the Intelligent Processing Board (IPB) acts as a high-throughput accelerator. This *parallel independence* was a deliberate design choice to enable the unambiguous attribution of performance data and failure signatures to their specific hardware and software origins.

### 2.1 Mission Context

The payload flew as a hosted unit onboard *Celestial Bliss* operated by *D-Orbit SpA (Italy)*. The spacecraft launched on August 16th, 2024, at 11:56 PDT (18:56 UTC) aboard a Falcon 9 rocket from the Space Launch Complex 4 East (SLC-4E) at Vandenberg Space Force Base in California, and deployed 2 h:35 m after liftoff into an approximately 597 km Sun-Synchronous Orbit and 97.8 deg inclination. The platform type is *ION SCV Magnificent Monica*. The primary mission objective is to carry several satellites and third-party hosted payloads. The on-orbit campaign reported in Table 8 spans August 21st, 2024 through July 15th, 2025. The operational phase during the campaign was routine. The spacecraft and hosted payload remained operational as of December 30th, 2025. As a hosted payload, the device under test operated under spacecraft constraints on power, command access, and telemetry rate (Tables 3 and 7).

#### 2.1.1 Hardware Components and Integration

The physical implementation of our three design philosophies is detailed in Table 2. Each board was designed with a distinct heritage to explore the trade-offs between optimization and risk. The DIB is a custom design, the APB is derived from a manufacturer reference design to ensure stability, and the IPB integrates a COTS NVIDIA Jetson module for maximum throughput. The flight hardware is shown in Fig. 2, and Table 3 details the standardized electrical interface.

**Table 3** payload electrical interface pinout (M80-482642M Connector)

| Pin number | Signal name | Description |
|---|---|---|
| 1, 2, 3 | 12V EXT PWR | Primary payload power input |
| 14, 15, 16 | GND | System ground |
| 20 | RS-422 TXD+ | RS-422 transmit (positive) |
| 19 | RS-422 TXD− | RS-422 transmit (negative) |
| 6 | RS-422 RXD+ | RS-422 receive (positive) |
| 7 | RS-422 RXD− | RS-422 receive (negative) |

### 2.2 Software Architecture

The software architecture for each board was tailored to its specific experimental role, as detailed in Table 4. The DIB was configured with a minimal bare-metal application, avoiding any operating system overhead. Its purpose was to isolate the PolarFire SoC and measure its intrinsic survivability, providing a "best-case" hardware baseline. In contrast, the APB and IPB were configured with nearly identical Linux-based application stacks to execute the same computational workload. The APB, using a Debian and Buildroot OS on the PolarFire SoC, was designed to represent the *Supervisory Node*, allowing us to characterize its performance and stability with a full but lightweight OS. The IPB, running a standard Ubuntu distribution on the Jetson Orin NX, was designed to represent the high-throughput COTS accelerator. This configuration enabled a direct, system-level comparison of performance, stability, and failure modes between the *Supervisory Node* and accelerator philosophies.

*Acronyms used in boot sources* eNVM denotes embedded non-volatile memory. NVMe denotes Non-Volatile Memory Express. SSD denotes solid-state drive.

*Heterogeneity and complexity management* The architecture separates roles across nodes. The APB is the Supervisory Node for power control, boot orchestration, and run-level health monitoring of the IPB over the payload serial interface. The IPB executes the high-throughput workload. This boundary limits cross-coupling between a complex accelerator software stack and the health manager logic.

*Redundancy scope* The payload is single-string per board and includes no cold-spare or hot-spare processors. The architecture provides a functional fallback. Workloads execute on the APB when the IPB is unavailable, at reduced throughput.

**Table 4** Software configuration and experimental role of onboard processing boards

| ID | Operating system | Boot source | Function in experiment |
|---|---|---|---|
| DIB | Bare-Metal (No OS) | Internal eNVM (128 KB) | Isolate and measure intrinsic SoC survivability |
| APB | Linux (Debian and Buildroot) | Micro SD Card | Establish a performance baseline for the *Supervisory Node* |
| IPB | Linux (Ubuntu) | External NVMe SSD | Characterize performance and operational risk of the *high-throughput COTS accelerator* |

## 3 Test Campaign and Validation

### 3.1 Pre-Launch Ground Testing

Our pre-launch validation strategy was pragmatically tailored to address the primary risk of COTS hardware: infant mortality. We implemented a two-phase approach based on Environmental Stress Screening (ESS) principles. First, a component-level thermal screening was conducted on all flight-candidate boards to precipitate latent defects and select the most robust units. The screening subjected the units to temperature extremes of −30°C and 70°C. A unit was classified as failing a specific test point if it required more than two manual reboots to become operational within a pre-defined test timeout period. The results, summarized in Table 5, revealed distinct thermal sensitivities. Based on this characterization, the unit from the PID-B batch was selected as the primary Flight Model (FM1), and the unit from the PID-C batch was designated the backup Flight Model (FM2).

Second, the integrated payload underwent a system-level acceptance campaign against the mechanical and thermal loads of launch. The key phases are summarized in Table 6. The thermal screening profile is shown in Fig. 3. The random vibration test profile, consistent with the launch provider's requirements [36], is shown in Fig. 4. The system-level thermal vacuum (TVAC) test profile, demonstrating functional performance at operational temperature plateaus, is shown in Fig. 5.

### 3.2 In-Orbit Campaign Design

The in-orbit campaign was conducted as a series of over 100 discrete, standardized test runs. The entire process was orchestrated by an On-Board Compute Procedure (OBCP), a compiled Python script running on the host spacecraft platform. The standardized sequence, shown in Fig. 6, ensured that every test was a repeatable, verifiable experiment. The OBCP initiates a test, triggers the payload's workload, captures the streaming telemetry with a CRC integrity check, and finalizes a log file upon receiving a termination signal. The logic is detailed in Algorithm 1, and the telemetry packet structure is defined in Table 7.

**Algorithm 1** On-board compute procedure (OBCP)

```
procedure ExecuteTestRun(target_board_id)
    serial ← InitializeSerialPort("/dev/tty...")
    log_file ← OpenLogFile(target_board_id, timestamp)
    SendCommand(serial, "START_COMMAND")
    eot_received ← false
    while not eot_received do
        if data available on serial then
            packet ← ReadPacket(serial)
            if VerifyCRC(packet) is true then
                WriteToLog(log_file, packet.data)
                if packet.type is EOT_PACKET then
                    eot_received ← true
            else
                WriteToLog(log_file, "ERROR: CRC mismatch")
    CloseLogFile(log_file)
    CloseSerialPort(serial)
```

### 3.3 Computational Workload and Algorithms

The computational workload executed on the APB and IPB was designed as a comprehensive benchmark of tasks common in Earth observation missions. It consisted of two primary C++ modules, `CorrectImg` and `AttEst`, selected to stress the processors with both high-throughput data processing and continuous floating-point arithmetic.

The first module, `CorrectImg`, emulates a standard radiometric correction pipeline, detailed in Algorithm 2. It ingests raw 12-bit image data and applies a two-step, per-pixel correction based on the mathematical formulation in Eqs. (1) through (5). The visual output of this process, showing the correction of raw image data, is presented in Fig. 7.

The second module, `AttEst`, provides a benchmark for real-time state estimation, implementing an Additive Multiplicative Extended Kalman Filter (AMEKF) as detailed in Algorithm 3 and Eqs. (6) through (14). The performance of

**Table 5** Summary of thermal screening and failure characterization

| Test condition | Unit ID | Reboot count | Disposition |
|---|---|---|---|
| Room temp. | PID-4-C | 1 | Operational |
| −30°C | PID-4-C | 1 | Operational |
| −30°C | PID-3-C | 4 | Disqualified for flight |
| −30°C | PID-3-B | 6 | Disqualified for flight |
| Room temp. (post-cold) | PID-4-C | 1 | Operational |
| 70°C | PID-4-B | 5 | Characterized failure mode |
| Room temp. (post-hot) | PID-4-C | 1 | Operational |

The unit from the PID-B batch was selected as the primary Flight Model (FM1)
The unit from the PID-C batch was designated the backup Flight Model (FM2)

**Table 6** Summary of the two-phase pre-launch validation campaign

| Phase | Test | Purpose | Standard and driver |
|---|---|---|---|
| Component screening | Thermal cycling (−30°C to +70°C) | Induce infant mortality failures; select most robust flight units | Internal procedure based on engineering best practice for COTS hardware |
| System acceptance | Random vibration | Verify survivability of the integrated payload against launch loads | SpaceX Rideshare User's Guide |
| System acceptance | Static load | Verify structural integrity of payload mounting interfaces | SpaceX Rideshare User's Guide |
| System acceptance | Thermal vacuum (TVAC) | Verify functional performance of the integrated system in a simulated space environment | Internal procedure simulating expected mission orbital temperatures |


**Fig. 3** Component-level thermal screening profile

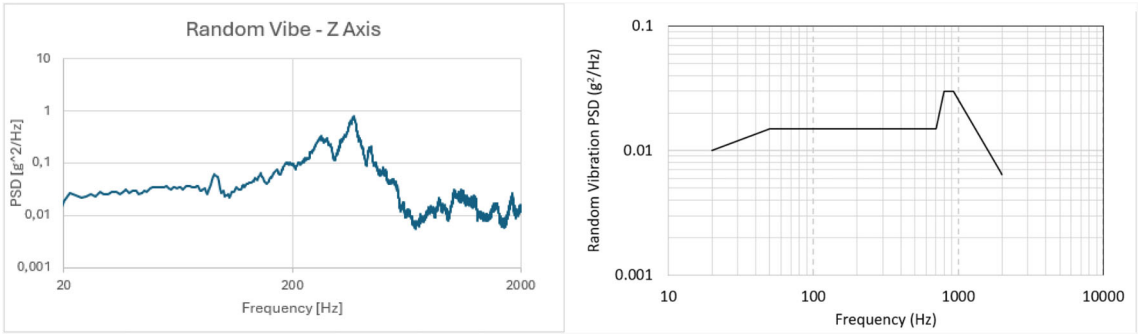



**Fig. 4** System-level random vibration test profile [36]

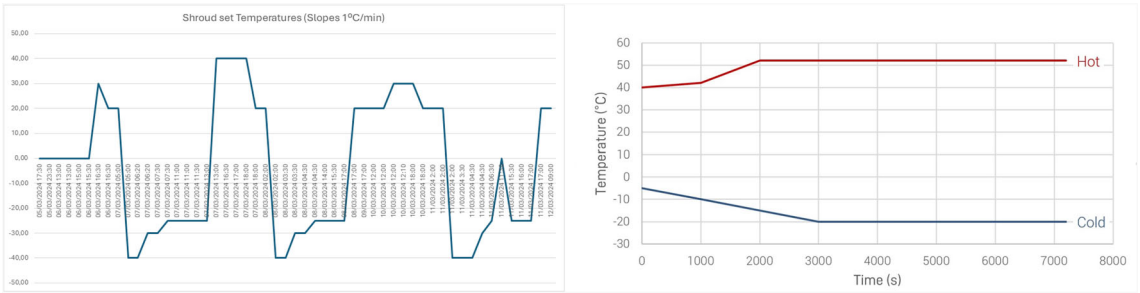
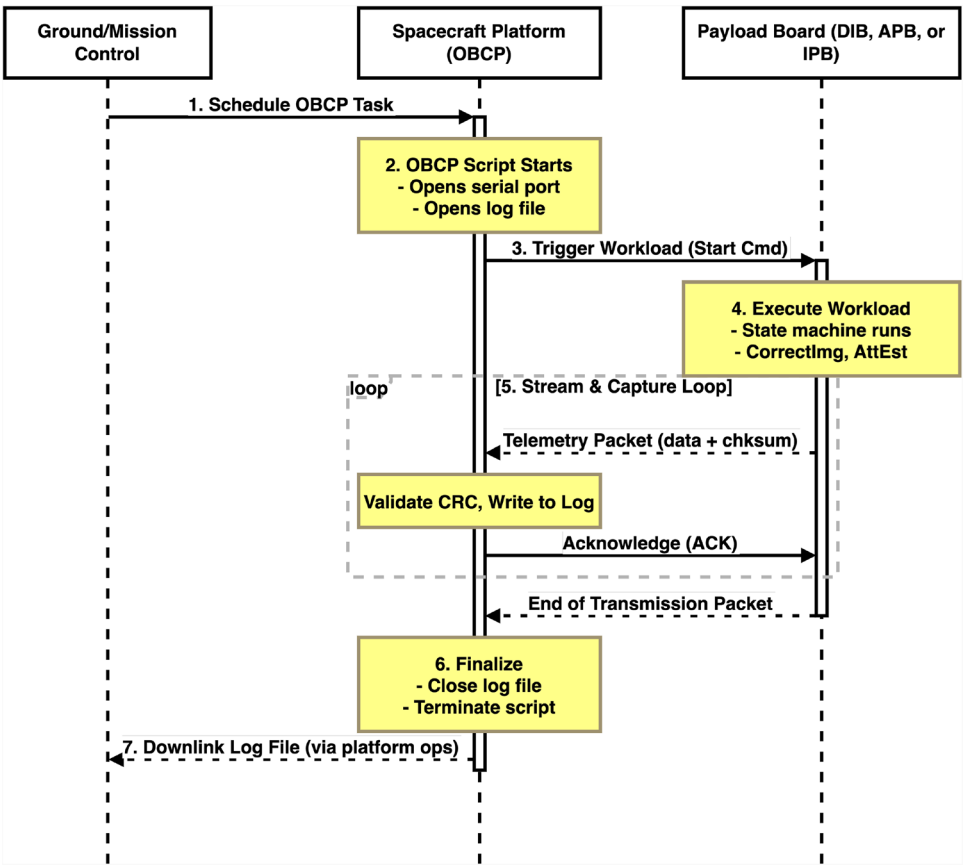



**Fig. 5** System-level thermal vacuum (TVAC) test profile [36]

**Fig. 6** In-orbit test sequence diagram



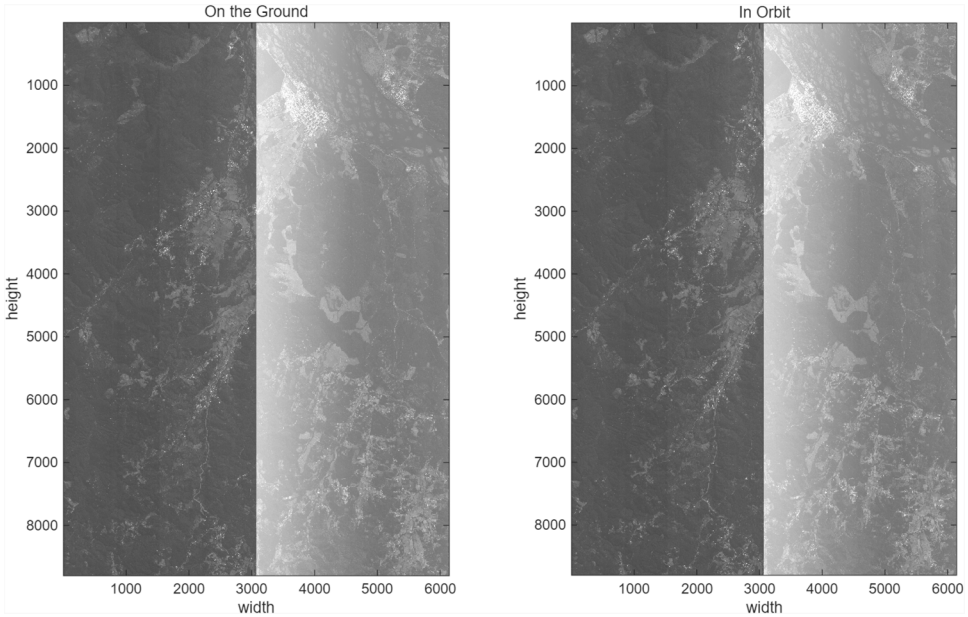



**Fig. 7** Visual result of the radiometric correction algorithm

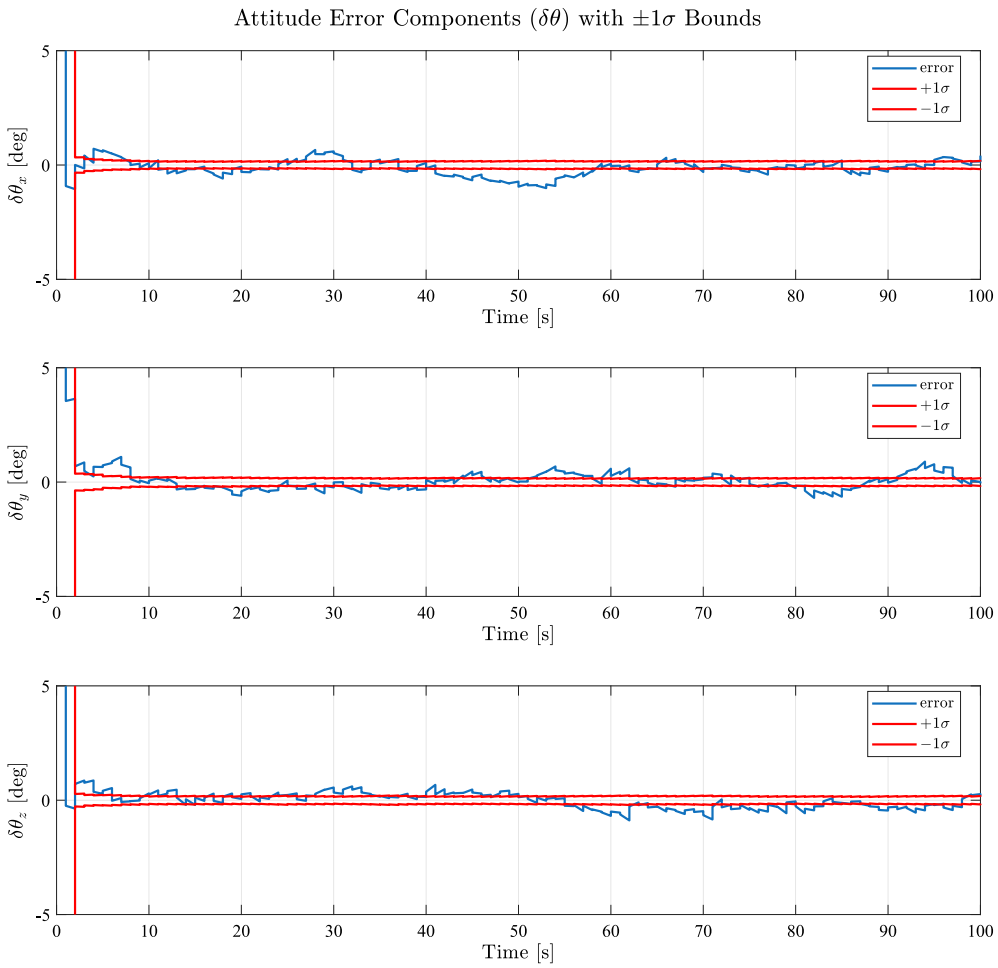



**Fig. 8** Attitude angle estimation error from the AttEst algorithm

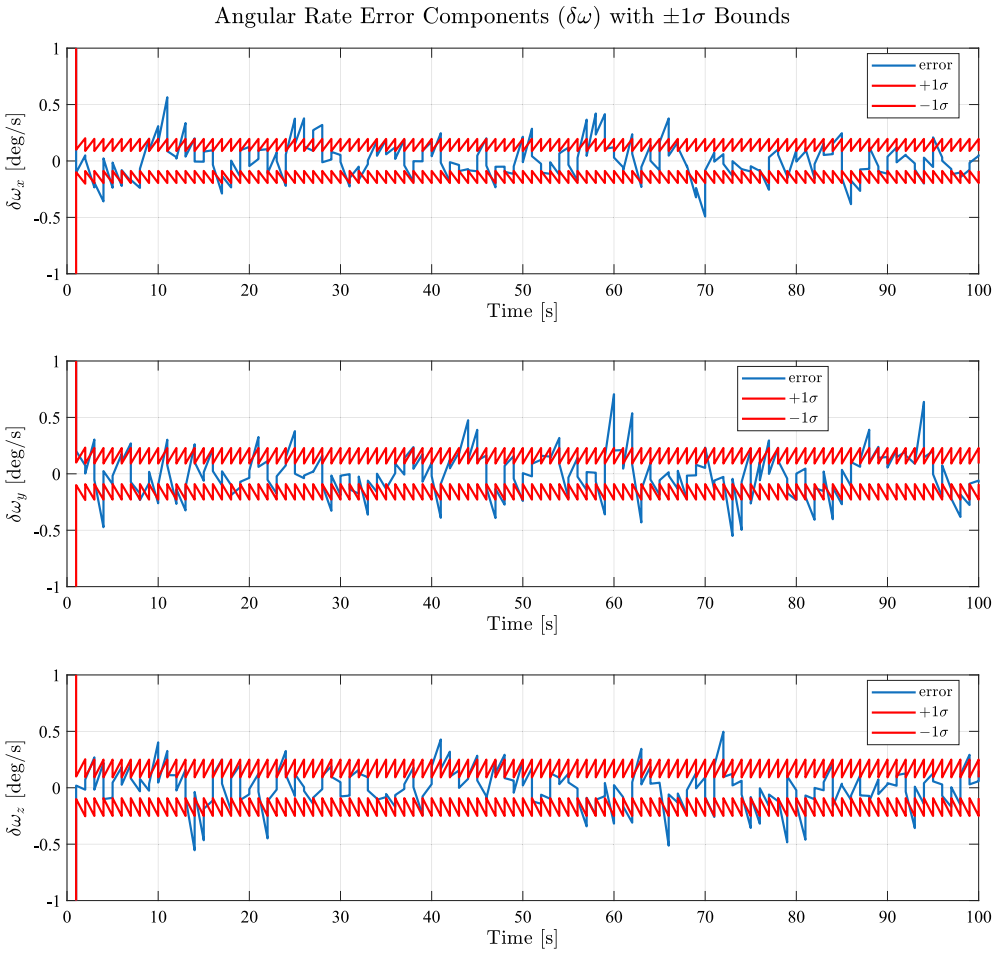



**Fig. 9** Angular rate estimation error from the AttEst algorithm

**Table 7** Decoded telemetry packet structure

| Field | Size (bytes) | Description |
|---|---|---|
| Timestamp | 4 | Mission elapsed time (seconds) |
| Board ID | 1 | Identifier for the source board (DIB, APB, or IPB) |
| Packet type | 1 | Defines the content type (e.g., log, status, EOT) |
| Payload | Variable | The core data content of the packet |
| CRC-32 | 4 | Cyclical redundancy check for data integrity |

the filter is shown in Figs. 8 and 9, which plot the estimation error for attitude and angular rate, respectively.

*Error definitions for Figs. 8 and 9* The attitude error $\delta\theta$ is the 3-axis small-angle representation derived from the quaternion error between the estimated attitude and the reference attitude used as truth for the benchmark. The angular rate error $\delta\omega$ is the component-wise difference between the estimated body rate and the reference body rate. In both plots, the red curves show the $\pm1\sigma$ bounds from the filter covariance terms for the corresponding state components. In Fig. 9, a drift remains because the AttEst filter estimates attitude and angular rate but omits gyro bias states. A standard reduction method is to augment the state vector with gyro bias states.

**Algorithm 2** Radiometric correction (`CorrectImg`)

```
procedure CorrectImage
    Initialize empty vector final_image_vector
    for mode from 0 to 3 do ▷ Loop over four spectral bands
        raw_files ← GetRawFileNames(mode)
        cal_files ← GetBinFileNames(mode)
        raw_image ← ReadImage(raw_files)
        (offset, gain, quartic) ← ReadCalFiles(cal_files)
                                        ▷ Apply two-stage correction
        sol_linear ← ApplyLinearCorrection(raw_image, offset, gain)
        sol_newton1 ← ApplyNewtonRaphson(sol_linear, quartic, raw_image)
        sol_newton2 ← ApplyNewtonRaphson(sol_newton1, quartic, raw_image)
        if any element in sol_newton2 is negative then
            final_solution ← sol_linear ▷ Revert if non-physical
        else
            final_solution ← sol_newton2
        downsampled_img ← DownsampleMatrix(final_solution, 100)
        Append(downsampled_img, final_image_vector)
    return final_image_vector
```

**Algorithm 3** Attitude estimation (`AttEst`)

```
procedure SimulateAttitudeFilter
    Initialize state vector xp and covariance matrix Pp
    Initialize noise matrices Qk and R
    Initialize empty vector results_vector
    for t from 0 to 99 do ▷ Loop over all measurement time steps
                                        ▷ Prediction Step
        x_predicted ← PropagateStateViaRK4(xp, dt)
        P_predicted ← PropagateCovariance(Pp, Qk)
                                        ▷ Correction Step
        measurement ← GetMeasurementAtTime(t)
        K_gain ← ComputeKalmanGain(P_predicted, R)
        xp ← UpdateState(x_predicted, K_gain, measurement)
        Pp ← UpdateCovariance(P_predicted, K_gain)
                                        ▷ Store results for this time step
        Append(xp, results_vector)
        Append(Pp, results_vector)
    return results_vector
```

### 3.4 Mathematical Formulation for Radiometric Correction

The radiometric correction process detailed in Algorithm 2 is based on the following mathematical steps, applied on a pixel-by-pixel basis. For a given pixel at location $(i, j)$:

*Offset correction* The raw measured digital number (DN), $I_{\text{measured}}$, is first corrected for dark current, $D(i)$, which is specific to each detector column $i$.

$$I_{\text{offset}}(i, j) = I_{\text{measured}}(i, j) - D(i) \tag{1}$$

*Linear correction (first solution)* A first approximation of the radiance, $L_{\text{linear}}$, is computed using a linear gain, $G(i)$, and offset, $O(i)$.

$$L_{\text{linear}}(i, j) = \frac{I_{\text{offset}}(i, j) - O(i)}{G(i)} \tag{2}$$

These coefficients correspond to the data loaded from the `*Linear16.bin` files into the `cal_gain` and `cal_offset` variables.

*Non-linear correction via Newton–Raphson* The final radiance, $L$, is the root of a fourth-order polynomial equation where the sensor's response equals the offset-corrected DN:

$$P(L) = aL^4 + bL^3 + cL^2 + dL + e = I_{\text{offset}}(i, j). \tag{3}$$

The coefficients $(a, b, c, d, e)$ are loaded from the `*Quartic.bin` files into the `cal_quartic` variable. This equation is solved iteratively using the Newton–Raphson method, with $L_{\text{linear}}$ as the initial guess. The update rule is:

$$L_{k+1} = L_k - \frac{P(L_k) - I_{\text{offset}}(i, j)}{P'(L_k)} \tag{4}$$

where $P'(L)$ is the derivative of the polynomial. This iterative update is applied twice to refine the solution.

*Fallback condition* If the non-linear correction results in a non-physical negative radiance value, the algorithm reverts to the more stable linear solution:

$$L_{\text{final}} = \begin{cases} L_{\text{linear}} & \text{if } L_{\text{newton2}} < 0 \\ L_{\text{newton2}} & \text{otherwise} \end{cases} \tag{5}$$

### 3.5 Mathematical Formulation for Attitude Estimation

The attitude estimation filter in Algorithm 3 is an Additive Multiplicative Extended Kalman Filter (AMEKF).

*State vector* The state vector $\mathbf{x}$ combines the attitude quaternion $\mathbf{q}$ and the angular velocity vector $\boldsymbol{\omega}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\omega} \end{bmatrix} \tag{6}$$

where $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ is the unit quaternion representing rotation from the inertial frame to the body frame, and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the body angular velocity.

*Prediction stage* The state is propagated forward in time using the continuous-time rigid-body dynamics, which are integrated numerically using a 4th-order Runge–Kutta (RK4)

method. The governing differential equations are:

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \quad \text{where} \quad \boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{7}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(-\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega})), \tag{8}$$

where $\mathbf{J}$ is the spacecraft's inertia matrix. The state covariance matrix $\mathbf{P}$ is propagated using the linearized state transition matrix $\mathbf{F}_k$ and the process noise covariance $\mathbf{Q}_k$:

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}. \tag{9}$$

*Correction stage* The predicted state is corrected using measurements. The AMEKF uses a multiplicative update for the quaternion to preserve its unit-norm constraint and an additive update for the angular velocity. The general EKF update equations are:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{10}$$

$$\delta\mathbf{x}_k = \mathbf{K}_k(\mathbf{y}_k - h(\mathbf{x}_{k|k-1})) \tag{11}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}, \tag{12}$$

where $\mathbf{y}_k$ is the measurement vector, $h(\mathbf{x})$ is the measurement model, $\mathbf{H}_k$ is its Jacobian, and $\mathbf{R}_k$ is the measurement noise covariance. The state is then updated in a hybrid manner:

$$\mathbf{q}_{k|k} = \delta\mathbf{q}(\delta\mathbf{x}_k) \otimes \mathbf{q}_{k|k-1} \quad \text{(Quaternion Multiplication)} \tag{13}$$

$$\boldsymbol{\omega}_{k|k} = \boldsymbol{\omega}_{k|k-1} + \delta\boldsymbol{\omega}(\delta\mathbf{x}_k) \quad \text{(Vector Addition)}, \tag{14}$$

where $\delta\mathbf{q}$ is the small error quaternion derived from the error state vector $\delta\mathbf{x}_k$.

## 4 In-Orbit Performance and Stability

The in-orbit demonstration yielded a comprehensive dataset that allows for a multi-layered analysis of COTS reliability. The full scope of the in-orbit campaign, encompassing over 500 individual test runs across 17 distinct test campaigns, is summarized in Table 8.

### 4.1 Baseline Performance in Nominal Conditions

Under nominal (non-SAA) operating conditions, the performance and reliability data establish a baseline contrast between the APB and IPB. The performance analysis in Table 9 shows that the IPB (mean 53.4 s) is approximately 7.6× faster than the APB (mean 405.6 s), with higher variance (16.6 s vs. 4.6 s standard deviation). The reliability analysis in Table 10 shows that the APB has a 38.0% run-level failure rate, nearly double that of the IPB at 19.3%. We classify a run with at least one reset as a failed run. The APB baseline is impacted by a known boot-chain constraint requiring Linux to boot from a microSD card rather than the intended eMMC path, a factor further analyzed in Sect. 5.

### 4.2 Impact of the South Atlantic Anomaly

The analysis of runs within the South Atlantic Anomaly (SAA), a region of unusually high radiation flux in low Earth orbit, provides key insights into the boards' environmental resilience. The performance analysis (Table 11) shows that mean execution times for both boards remain stable and consistent with their non-SAA performance. The reliability analysis (Table 12), however, reveals a significant divergence in behavior. The APB's reliability is largely unaffected by the SAA, maintaining a high failure rate of 37.5%. In an unexpected result, the IPB failure rate decreases to 9.1% in SAA-classified runs. Section 5.3 discusses why this should not be interpreted as improved radiation tolerance.

## 5 Analysis of Failure Pathways

The performance and reliability data reveal distinct failure signatures for the APB and IPB. In this dataset, the observed instability is dominated by system integration and operational protocol issues visible in the available telemetry. Table 13 lists the system-level operational vulnerabilities observed in flight data and the handling actions.

### 5.1 System Maturation and End-of-Life Analysis

The operational history of the payload, shown in Fig. 10, reveals a sequence of system performance under increasing stress, culminating in the end-of-life for the high-performance IPB. The initial test campaigns (LEOP through Test-011) established a baseline of nominal performance, characterized by a low number of reset events. A significant shift occurred starting with Test-012. During this phase, the IPB was deliberately subjected to more frequent operations within the SAA to assess its long-term resilience. The data shows a spike in instability, beginning with 9–10 resets during non-SAA runs in Test-012, and escalating to a mission-high of 16 resets during an SAA pass in Test-013. This period of heightened instability continued through Test-014 (9 resets, SAA) and ended with the final, unsuccessful run in Test-015, after which the IPB became permanently unresponsive. This sequence provides a data-rich case study of a COTS accelerator's failure cascade when subjected to cumulative

**Table 8** Summary of test campaigns

| Test ID | Timeframe | Total | Success | No resp. | Non-SAA | SAA |
|---|---|---|---|---|---|---|
| LEOP | 2024-08-21 to 2024-08-21 | 6 | 3 | 3 | 6 | 0 |
| Test-001 | 2024-09-02 to 2024-09-03 | 9 | 1 | 8 | 9 | 0 |
| Test-002 | 2024-09-03 to 2024-09-03 | 9 | 6 | 3 | 8 | 1 |
| Test-003 | 2024-09-23 to 2024-09-23 | 3 | 1 | 2 | 3 | 0 |
| Test-004 | 2024-09-30 to 2024-10-15 | 3 | 2 | 1 | 3 | 0 |
| Test-005 | 2024-10-11 to 2024-10-11 | 3 | 2 | 1 | 3 | 0 |
| Test-006 | 2024-10-15 to 2024-10-20 | 72 | 72 | 0 | 66 | 6 |
| Test-007 | 2024-10-22 to 2024-10-27 | 96 | 96 | 0 | 58 | 38 |
| Test-008 | 2024-10-29 to 2024-11-09 | 96 | 96 | 0 | 81 | 15 |
| Test-009 | 2024-11-12 to 2024-11-18 | 96 | 96 | 0 | 80 | 16 |
| Test-010 | 2024-11-18 to 2024-11-22 | 93 | 93 | 0 | 86 | 7 |
| Test-011 | 2025-02-12 to 2025-02-18 | 4 | 2 | 2 | 1 | 3 |
| Test-012 | 2025-03-10 to 2025-03-17 | 6 | 6 | 0 | 4 | 2 |
| Test-013 | 2025-04-11 to 2025-04-15 | 3 | 3 | 0 | 0 | 3 |
| Test-014 | 2025-05-16 to 2025-05-20 | 3 | 3 | 0 | 0 | 3 |
| Test-015 | 2025-06-13 to 2025-06-17 | 3 | 1 | 2 | 0 | 3 |
| Test-016 | 2025-07-11 to 2025-07-15 | 3 | 2 | 1 | 1 | 2 |

**Table 9** Performance analysis in non-SAA conditions

| Board ID | Mean (s) | Std dev (s) | Median (s) | Min (s) | Max (s) |
|---|---|---|---|---|---|
| APB | 405.6 | 4.6 | 406.0 | 385.0 | 427.0 |
| IPB | 53.4 | 16.6 | 51.5 | 20.0 | 77.0 |

**Table 10** Reliability analysis (non-SAA)

| board_id | Total runs | Runs w/zero resets | Runs w/ >=1 reset | Failure rate (%) | Mean resets | Total resets |
|---|---|---|---|---|---|---|
| APB | 129 | 80 | 49 | 38.0% | 0.660000 | 85 |
| IPB | 114 | 92 | 22 | 19.3% | 0.340000 | 39 |

*operational and environmental stresses*, moving from intermittent resets to catastrophic, non-recoverable failure.

## 5.2 APB Failure Mode: Fragile I/O Boot Chain Vulnerability

The APB shows a 38.0% run-level failure rate in non-SAA passes (Table 10). This behavior tracks the boot source. The APB boots Linux from a microSD card due to an issue with the intended eMMC boot path. The available serial console output and run termination patterns indicate stalled runs during boot or early root filesystem access, followed by recovery only after a hard power cycle. This supports a single-point operational vulnerability at the SD boot and root filesystem interface. The result is a measurement of boot-chain and storage-interface fragility rather than a measurement of PolarFire SoC silicon reliability.

## 5.3 IPB Failure Mode: Cumulative Corruption from Operational Protocol

The IPB shows an end-of-life cascade that starts with intermittent resets and ends with permanent non-response after Test-015 (Fig. 10). Reset counts rise sharply after Test-012 and remain elevated through Test-014. This signature matches cumulative degradation in the storage stack under

**Table 11** Performance analysis in SAA conditions

| Board ID | Mean (s) | Std dev (s) | Median (s) | Min (s) | Max (s) |
|---|---|---|---|---|---|
| APB | 405.9 | 5.9 | 406.0 | 383.0 | 427.0 |
| IPB | 57.2 | 16.3 | 67.0 | 15.1 | 68.0 |

**Table 12** Reliability analysis (SAA)

| board_id | Total runs | Runs with zero resets | Runs with >=1 reset | Failure rate (%) | Mean resets | Total resets |
|---|---|---|---|---|---|---|
| APB | 32 | 20 | 12 | 37.5% | 0.500000 | 16 |
| IPB | 44 | 40 | 4 | 9.1% | 0.610000 | 27 |

**Table 13** Observed system-level operational vulnerabilities and mission impact

| Item | Node | Operational vulnerability and observed signature | Mission impact and handling action |
|---|---|---|---|
| 1 | APB | Linux boot and root filesystem rely on a microSD card. The run-level data shows elevated reset rates and stalled runs that require hard power cycling for recovery | Loss of run data for the affected pass. Handling action in this campaign was hard power cycling. Design action is the removal of microSD from the boot chain or addition of a verified fallback image on internal non-volatile storage |
| 2 | IPB | Boot and root filesystem rely on an external NVMe SSD. Reset counts rise sharply in late campaigns and the node becomes non-responsive after Test-015 (Fig. 10) | Progressive loss of accelerator availability, ending in permanent loss of the node. Design action is the reduction of abrupt power loss events, the addition of storage health telemetry, and the addition of a supervisor-controlled reimage path |
| 3 | Hosted interface | Telemetry and logging are limited to serial console output and encoded result packets. Kernel-level logs and storage health counters are not available during anomalies | Root-cause confirmation is limited on orbit. Handling action relies on run-level signatures and recovery by power cycling. Design action is added instrumentation for persistent logs and health counters |

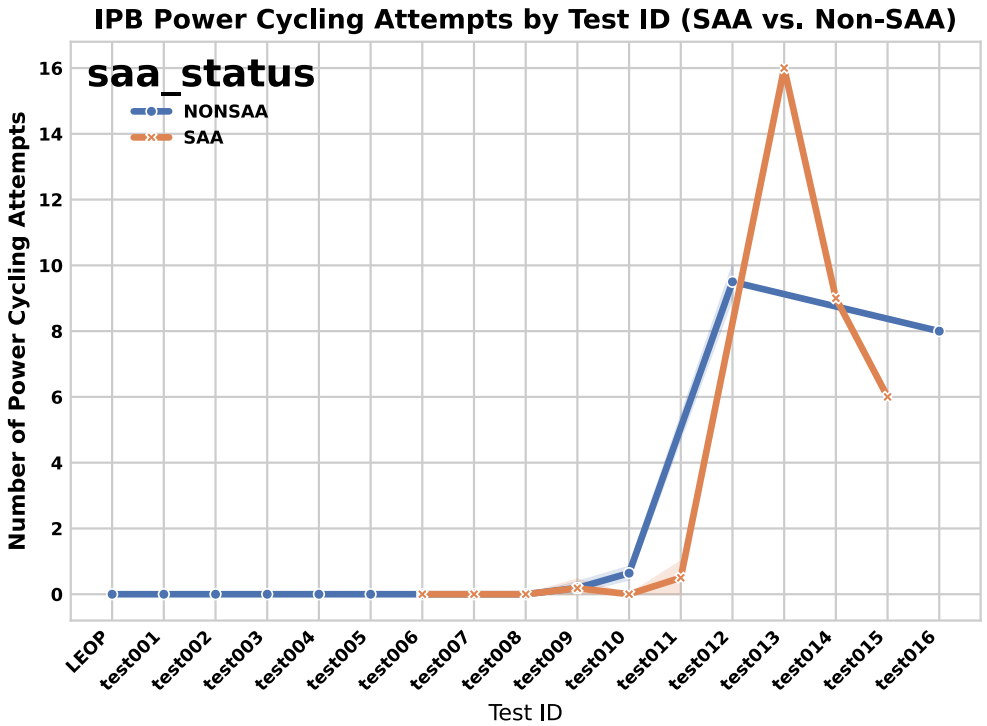



**Fig. 10** IPB power cycling counts over the full mission campaign

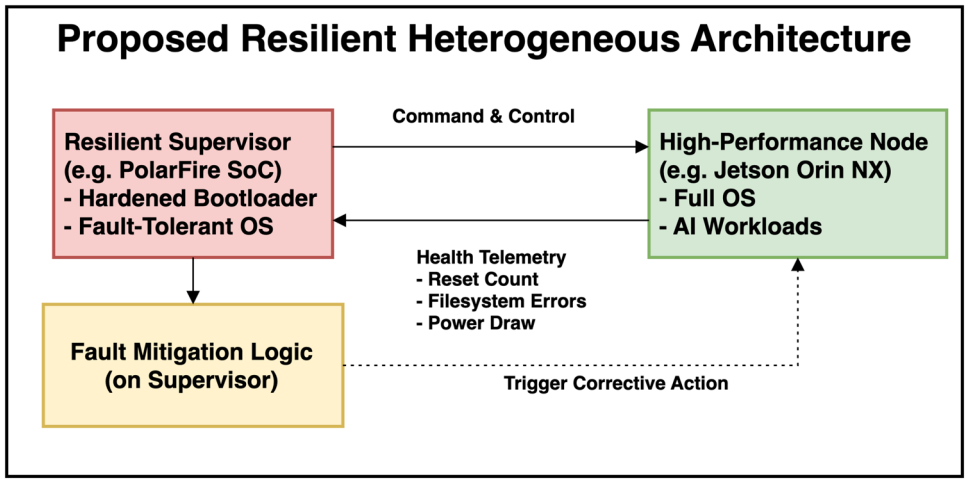



**Fig. 11** Conceptual architecture for future resilient systems

repeated abrupt power loss. The IPB boots Linux from an external NVMe SSD and uses a journaling filesystem. The test protocol relies on abrupt power cycling rather than graceful shutdown. These factors raise risk of filesystem inconsistency and storage wear. The on-orbit dataset does not include kernel-level filesystem diagnostics or NVMe health counters during anomalies, so storage corruption remains a root-cause inference rather than a directly logged event. The main empirical result is the observed reset escalation followed by loss of node availability under this operational protocol.

## 6 Discussion

The in-orbit results reported in this paper address the data gap identified in the introduction. Radiation-induced transient faults remain a mission risk for COTS processors. In this dataset, the dominant observed risks arise at the system integration and operational protocol level. The APB failure signature traces to a fragile storage boot chain. The IPB signature shows progressive degradation under the operational power-cycling protocol.

Our work demonstrates that a heterogeneous architecture is not merely a strategy for mitigating radiation. It is a comprehensive approach to total mission risk management. The APB's integration vulnerability (the fragile I/O boot chain) and the IPB's operational failure pathway (cumulative filesystem corruption) highlight that mission-ending threats can come from multiple, distinct vectors. A high-performance accelerator like the Jetson can deliver a significant computational speedup ( $7.6\times$ in our case), but its operational complexity introduces a predictable end-of-life degradation pathway. A theoretically reliable supervisor node like the PolarFire SoC can have its resilience completely negated by a single integration flaw.

Therefore, a successful architecture must account for both. The role of the resilient supervisor is not just to be a radiation-tolerant backup, but to serve as a system that can reliably overcome the integration faults and manage the predictable failure modes of the high-performance node. For mission designers, the key insight is that the escalating reset count we observed in the IPB serves as a vital, measurable precursor to total system loss. This highlights the absolute necessity for an autonomous, onboard health monitoring system, running on the resilient supervisor. Such a system can detect these early warning signs and trigger fault mitigation strategies, such as rerouting critical tasks or placing the high-performance node into a safe state, long before a failure becomes mission-ending. Our findings thus provide the data-driven foundation for designing these next-generation fault-tolerant systems.

## 7 Conclusion

This paper reported in-orbit performance data from a heterogeneous payload combining a Microchip PolarFire SoC and an NVIDIA Jetson Orin NX. In this on-orbit dataset, many observed instabilities arise from system integration faults in the boot and storage chain and from operational protocol-driven degradation. The results do not rule out radiation-induced faults, but the available telemetry did not show isolated radiation events as the dominant driver of the observed resets. These findings support a heterogeneous architecture where a resilient Supervisory Node manages recovery actions and degradation handling for a high-performance accelerator.

### 7.1 Limitations and Future Work

This study has the following constraints.

1. Telemetry and logging. On-orbit diagnostic data was limited to serial console output and encoded result packets. The dataset does not include persistent kernel logs, filesystem integrity logs, or NVMe health counters during anomalies.
2. Fault attribution. The failure pathway analysis relies on run-level signatures and the available serial output. In several cases, the data supports a narrow set of likely causes but does not provide direct confirmation at the driver or storage layer.
3. Baseline comparability. The DIB boots bare-metal code from internal eNVM, while the APB boots Linux from a microSD card and the IPB boots Linux from an external NVMe SSD. These differences limit direct comparison of filesystem robustness across nodes.
4. Radiation and silent data corruption. This campaign did not measure SEU rates, inference correctness under bit flips, or ECC event counters. The analysis focuses on run-level resets and node availability rather than on silent data corruption.
5. Cyber-security. This paper does not evaluate cyber-security properties of the open-source or COTS software stack, including secure boot configuration, patch state, supply-chain risk, or SBOM coverage.

These limitations define a clear path forward. Future work must focus on developing and validating the system concept shown in Fig. 11: a resilient Supervisory Node with a hardened bootloader and fault-tolerant OS. This Supervisory Node's primary role is to act as a health and mission manager, actively monitoring the specific degradation signatures we observed (e.g., escalating reset counts, filesystem errors) on the high-performance accelerator. Follow-on missions should add persistent health telemetry and storage diagnostics. Examples include kernel log persistence across reboot, filesystem integrity counters, and NVMe health telemetry. For silent data corruption, follow-on missions should add inference validity checks, redundant computation for safety-critical outputs, and supervisor-controlled rollback or reimage after integrity faults. This architecture enables the fault-mitigation strategies heterogeneous systems promise, transforming the high-performance COTS component from a liability into a managed, reliable asset. Ultimately, this research demonstrates the future of resilient space computing depends not only on hardening systems against the external environment, but on designing them to be resilient against their own internal complexities.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s42405-026-01130-w.

**Acknowledgements** This research, including all hardware procurement, testing, and launch services, was fully funded and supported by TelePIX Co., Ltd.. We extend our gratitude to D-Orbit for their collaboration on the launch contract and for providing the facilities for system-level testing as well as further mission management and support. We also thank the staff at Korea Reliability Technology Center for their support during component-level thermal screening, and EDMFG for their foundational work on the electronics schematics and initial FPGA design. The authors are grateful to Sangkwon Park for developing the firmware, Dongwook Koh for managing the environmental screening logistics and providing valuable business context, and Seonghui Kim for providing the radiometric correction datasets and algorithms that formed a core part of the computational workload. A special acknowledgment is due to Dr. Natnael Zewge. His insightful feedback on the manuscript's structure, the inclusion of detailed algorithms and equations, and the overall narrative strategy was instrumental in elevating the quality and impact of this work.

**Funding** Open Access funding enabled and organized by KAIST.

**Data Availability** The datasets generated and analyzed during the current study are not publicly available due to third-party mission and operational restrictions and cannot be shared.

## Declarations

**Conflict of interest** The authors declare no competing financial interests.

**Ethics approval** The research does not involve human participants, their data or biological material and it does not involve animals.

**Consent to participate** Not applicable.

**Consent for publication** Not applicable.



## References

1. Kulu E (2024) Cubesats & nanosatellites—2024 statistics, forecast and reliability. In: Proceedings of the international astronautical congress, IAC, vol. 2024
2. Cratere A, Gagliardi L, Sanca GA, Golmar F, Dell'Olio F (2024) On-board computer for Cubesats: state-of-the-art and future trends. IEEE Access 12:99537–99569. https://doi.org/10.1109/ACCESS.2024.3428388
3. Wang G, Wan G, Su Z, Wang Y, Jia Y, Li G, Liang S (2025) High-performance on-orbit intelligent computing and real-time services for remote sensing satellites based on large-scale computing power in space. IEEE Access

4. Ieracitano C, Mammone N, Spagnolo F, Frustaci F, Perri S, Corsonello P, Morabito FC (2024) An explainable embedded neural system for on-board ship detection from optical satellite imagery. Eng Appl Artif Intell 133:108517
5. Priyadharshini SD, Vadivazhagan K (2024) Enhanced vessel detection in maritime surveillance using multi-modal data integration and deep learning. In: 2024 8th international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). IEEE, pp 1090–1099
6. Goudemant T, Francesconi B, Aubrun M, Kervennic E, Grenet I, Bobichon Y, Bellizzi M (2024) Onboard anomaly detection for marine environmental protection. IEEE J Sel Top Appl Earth Obs Remote Sens 17:7918–7931
7. Kim D, Lee H, Won D, Han M (2024) Fpga-based inference parallelization for onboard rl-based routing in dynamic leo satellite networks. Int J Aeronaut Space Sci 25(3):1135–1145
8. Pasandi HB, Fraire JA, Ratnasamy S, Rivano H (2024) A survey on direct-to-device satellite communications: advances, challenges, and prospects. In: Proceedings of the 2nd international workshop on LEO networking and communication, pp 7–12
9. García LP, Furano G, Ghiglione M, Zancan V, Imbembo E, Ilioudis C, Clemente C, Trucco P (2024) Advancements in onboard processing of synthetic aperture radar (sar) data: enhancing efficiency and real-time capabilities. IEEE J Sel Top Appl Earth Obs Remote Sens 17:16625–16645
10. Del Castillo MO, Morgan J, McRobbie J, Therakam C, Joukhadar Z, Mearns R, Barraclough S, Sinnott R, Woods A, Bayliss C et al (2024) Mitigating challenges of the space environment for onboard artificial intelligence: Design overview of the imaging payload on spirit. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6789–6798
11. Ruf C, Gleason S (2025) Spaceborne gnss-r bistatic radar remote sensing, cygnss, and future missions. In: Proceedings of the IEEE
12. Nahiri L, Chanoui MA, Addaim A et al (2025) Firmware updating approach for fpga-based sdr nanosatellite payload. Int J Aeronaut Space Sci. https://doi.org/10.1007/s42405-025-01016-3
13. Adams C, Spain A, Parker J, Hevert M, Roach J, Cotten D (2019) Towards an integrated gpu accelerated soc as a flight computer for small satellites. In: 2019 IEEE aerospace conference. IEEE, pp 1–7
14. Machi V (2025) Planet to Use NVIDIA AI Processor Onboard Pelican Satellite. https://www.satellitetoday.com/technology/2024/06/10/planet-to-use-nvidia-ai-processor-onboard-pelican-satellite/. Accessed 12 July 2025
15. EOPortal (2025) Planet Pelican. https://www.eoportal.org/satellite-missions/planet-pelican. Accessed 12 July 2025
16. Amorim RC, Martins R, Harikrishnan P, Ghiglione M, Helfers T (2021) Dependable mpsoc framework for mixed criticality applications. In: European workshop on on-board data processing (OBDP)
17. Bruhn FC, Tsog N, Kunkel F, Flordal O, Troxel I (2020) Enabling radiation tolerant heterogeneous gpu-based onboard data processing in space. CEAS Space J 12(4):551–564. https://doi.org/10.1007/s12567-020-00321-9
18. Budroweit J, Patscheider H (2021) Risk assessment for the use of cots devices in space systems under consideration of radiation effects. Electronics 10(9):1008
19. Katsube S, Sahara H (2025) Towards an onboard anomaly detection and identification method for satellites. IEEE Access
20. Crotti E, Colagrossi A et al (2025) Machine learning approaches for data-driven self-diagnosis and fault detection in spacecraft systems. Appl Sci 15(14):1–25
21. Ibrahim SK, Ahmed A, Zeidan MAE, Ziedan IE (2020) Machine learning techniques for satellite fault diagnosis. Ain Shams Eng J 11(1):45–56
22. Weiss BM, Clarke B, Elnourani M, Öhrwall Rönnbäcka A, Laufer R, Macdonald M (2024) Leveraging smart maintenance for satellite health preservation. In: 75th international astronautical congress
23. Hedayati M, Barzegar A, Rahimi A (2024) Fault diagnosis and prognosis of satellites and unmanned aerial vehicles: a review. Appl Sci 14(20):9487
24. Giuffrida G, Fanucci L, Meoni G, Batič M, Buckley L, Dunne A, Dijk C, Esposito M, Hefele J, Vercruyssen N, Furano G, Pastena M, Aschbacher J (2022) The phi-Sat-1 mission: the first on-board deep neural network demonstrator for satellite earth observation. IEEE Trans Geosci Remote Sens 60:1–14. https://doi.org/10.1109/TGRS.2021.3125567
25. Guerrisi G, Del Frate F, Schiavon G (2023) Artificial intelligence based on-board image compression for the $\varphi$-sat-2 mission. IEEE J Sel Top Appl Earth Obs Remote Sens 16:8063–8075
26. Kacker S, Meredith A, Cahoy K, Labrèche G (2022) Machine learning image processing algorithms onboard OPS-SAT. In: Small satellite conference
27. Evans D, Merri M (2014) Ops-sat: a esa nanosatellite for accelerating innovation in satellite control. In: SpaceOps 2014 conference, p 1702
28. Leopard (2025) Advanced Data Processing Unit for Satellites—kplabs.space. https://www.kplabs.space/solutions/hardware/leopard. Accessed 12 July 2025]
29. Goodwill J, Crum G, MacKinnon J, Brewer C, Monaghan M, Wise T, Wilson C (2021) Nasa spacecube edge tpu smallsat card for autonomous operations and onboard science-data analysis. In: Proceedings of the small satellite conference. AIAA
30. Diana L, Dini P (2024) Review on hardware devices and software techniques enabling neural network inference onboard satellites. Remote Sens 16(21):3957
31. Bhattacharyya A, Nambiar SM, Ojha R, Gyaneshwar A, Chadha U, Srinivasan K (2023) Machine learning and deep learning powered satellite communications: enabling technologies, applications, open challenges, and future research directions. Int J Satel Commun Network 41(6):539–588
32. Furano G, Meoni G, Dunne A, Moloney D, Ferlet-Cavrois V, Tavoularis A, Byrne J, Buckley L, Psarakis M, Voss K-O et al (2020) Towards the use of artificial intelligence on the edge in space systems: challenges and opportunities. IEEE Aerosp Electron Syst Mag 35(12):44–56
33. Furano G, Tavoularis A, Rovatti M (2020) Ai in space: applications examples and challenges. In: 2020 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT). IEEE, pp 1–6
34. Geist A, Brewer C, Davis M, Franconi N, Heyward S, Wise T, Crum G, Petrick D, Ripley R, Wilson C et al (2019) Spacecube v3.0 nasa next-generation high-performance processor for science applications
35. Lüdtke D, Firchau T, Cortes CG, Lund A, Nepal AM, Elbarrawy MM, Hammadeh ZH, Meß J-G, Kenny P, Brömer F et al (2023) Scosa on the way to orbit: reconfigurable high-performance computing for spacecraft. In: 2023 IEEE space computing conference (SCC). IEEE, pp 34–44
36. Space Exploration Technologies Corp. (2022) Rideshare payload user's guide. Technical report, Space Exploration Technologies Corp