# AegisSat: Securing AI-Enabled SoC FPGA Satellite Platforms

Huimin Li†, Vusal Novruzov*, Nikhilesh Singh†, Lichao Wu†, Mohamadreza Rostami†, Ahmad-Reza Sadeghi†
Technische Universität Darmstadt, Germany
Email: †{huimin.li, nikhilesh.singh, lichao.wu, mohamadreza.rostami, ahmad.sadeghi}@trust.tu-darmstadt.de
Email: *{vusal.novruzov}@stud.tu-darmstadt.de

***Abstract*—The increasing adoption of System-on-Chip Field-Programmable Gate Arrays (SoC FPGAs) in AI-enabled satellite systems, valued for their reconfigurability and in-orbit update capabilities, introduces significant security challenges. Compromised updates can lead to performance degradation, service disruptions, or adversarial manipulation of mission outcomes. To address these risks, this paper proposes a comprehensive security framework, AegisSat. It ensures the integrity and resilience of satellite platforms by (i) integrating cryptographically-based secure boot mechanisms to establish a trusted computing base; (ii) enforcing strict runtime resource isolation; (iii) employing authenticated procedures for in-orbit reconfiguration and AI model updates to prevent unauthorized modifications; and (iv) providing robust rollback capabilities to recover from boot and update failures and maintain system stability. To further support our claims, we conducted experiments demonstrating the integration of these mechanisms on contemporary SoC FPGA devices. This defense-in-depth framework is crucial for space applications, where physical access is impossible and systems must operate reliably over extended periods, thereby enhancing the trustworthiness of SoC FPGA-based satellite systems and enabling secure and resilient AI operations in orbit.**

***Index Terms*—Satellite Security, SoC FPGA, AI, Reconfiguration, Multi-tenant.**

## I. Introduction

Satellites, which used to be static platforms with fixed configurations and limited adaptability, are evolving into intelligent, reconfigurable systems capable of performing sophisticated data processing tasks in orbit [1]–[4]. This transformation is driven by the increasing demand for enhanced autonomy, reduced dependence on ground-based control, rapid self-decision-making in mission-critical scenarios, and optimized use of constrained communication bandwidth [1], [3], [5]. On the other hand, the rapid advancement of Artificial Intelligence (AI) and edge computing technologies is revolutionizing the architecture and capabilities of modern space systems [1], [3], [6], [7]. AI enables critical onboard functions, including real-time data analytics, adaptive payload management, and autonomous decision-making under conditions of limited or delayed communication [6]. These capabilities enhance the resilience, efficiency, and operational independence of satellites across diverse mission profiles [1], [8], [9].

FPGAs have emerged as a preferable platform to meet the stringent performance requirements, adaptability, and energy efficiency in these missions [10]–[12]. Besides, their reconfigurability, deterministic latency, and parallel processing capabilities make them ideal for optimizing and accelerating AI workloads, such as neural network inference and real-time machine learning (ML) enhanced signal processing. Additionally, Commercial Off-The-Shelf (COTS) FPGAs are increasingly adopted in small satellites, thanks to their compact size, rapid prototyping, in-orbit updates, and flexible functionality, supporting various academic, military, and industrial applications [11], [13]–[16]. System-on-Chip (SoC) FPGAs further enhance these advantages by integrating general-purpose processors, referred to as the *Processing System* (PS), with programmable logic, known as the *Programmable Logic* (PL), fabricated together within a single silicon device [1], [17]. This hybrid architecture enables tight coupling between software control logic and hardware-accelerated AI engines, supporting complete AI processing pipelines with minimal latency and interconnect overhead.

Unfortunately, similar to other hardware devices, satellites have been targeted by real-world cyberattacks on multiple occasions. A prominent example is the ViaSat Cyberattack during the Russo-Ukrainian War, where attackers exploited a security vulnerability from the ground segment to achieve privilege escalation [18], [19]. Indeed, the flexibility of SoC FPGAs, while enabling powerful in-orbit reconfiguration and AI acceleration, also introduces significant security challenges. A compromised configuration change or an AI model update could lead to performance degradation, service disruption, or adversarial manipulation of mission objectives. The inability to physically access satellites after launch further amplifies these risks [19]. Despite these concerns, prior research has addressed only fragments about the security of AI-enabled SoC FPGA satellite platforms. For example, Li et al. proposed a partial reconfiguration method for satellite cryptographic devices [20]. Cotret et al. presented lightweight reconfiguration security services for SoC FPGAs [21], and Vallez et al. investigated onboard AI model updates while controlling their size and integrity [22]. Yet, no prior work has systematically integrated these aspects into a unified framework.

Satellites of the *multi-tenant model* paradigm enable multiple stakeholders, from commercial entities to government agencies, to share satellite resources such as payloads, sensors, and communication channels. By supporting diverse workloads concurrently, multi-tenant satellites improve utilization, reduce costs, and enable flexible Satellite-as-a-Service (SaaS) deployments [23], [24]. However, while cloud-FPGA literature

has discussed partitioning programmable logic into virtual FPGAs (vFPGAs) for isolation, this concept has not been adapted to multi-tenant satellite architectures, where preventing leakage or Trojan injection between tenants is critical. Recent works, such as [19], [25], have highlighted the potential of multi-tenant satellite platforms but stopped short of describing concrete implementations or security strategies.

Finally, as satellite-to-satellite communication becomes more prevalent, federated constellations will enable collaborative machine learning workflows, such as distributed training and in-orbit model sharing, as illustrated in Fig. 1. This evolution significantly broadens the attack surface, introducing threats that are unique to machine learning pipelines, including model injection, adversarial perturbations, and model exfiltration. These risks are further amplified when unverified updates propagate across interconnected platforms, potentially compromising the integrity and confidentiality of shared AI models [26], [27].

To address these challenges, this paper proposes a comprehensive security framework, named AegisSat[1] for AI-enabled SoC FPGA satellite platforms. Our approach establishes defense-in-depth by combining (i) cryptographically anchored secure boot, (ii) runtime isolation through TrustZone and hardware firewalls, (iii) authenticated reconfiguration and AI model updates, and (iv) robust rollback protection. Collectively, these mechanisms safeguard reconfigurable satellite platforms throughout their operational lifecycle in the inaccessible environment of space. Our contributions are as follows:

- We develop a systematic threat model that characterizes the attack surfaces resulting from the convergence of reconfigurable logic, AI acceleration pipelines, and multi-tenant mission architectures.
- We propose a layered security framework, AegisSat that unifies secure initialization, runtime protection, and lifecycle update mechanisms into an end-to-end defense architecture.
- We conduct experiments demonstrating the integration of these mechanisms on contemporary SoC FPGA devices.
- We outline a forward-looking research agenda highlighting critical open challenges to counter emerging threats.

## II. Background

### A. AI Acceleration and Design Methodologies on SoC FPGAs

AI acceleration on SoC FPGAs leverages the inherent parallelism of FPGA fabric to optimize the execution of machine learning algorithms. Advanced design methodologies, such as high-level synthesis (HLS) and hardware/software (HW/SW) co-design, facilitate the efficient implementation of AI workloads on these reconfigurable platforms. Neural network inference, particularly with quantized models, can be effectively mapped onto FPGA logic using HLS tools and dedicated AI model compilers [7], [11].

[1]Aegis was the shield of Zeus, meaning protection. Sat is the abbreviation for Satellite.

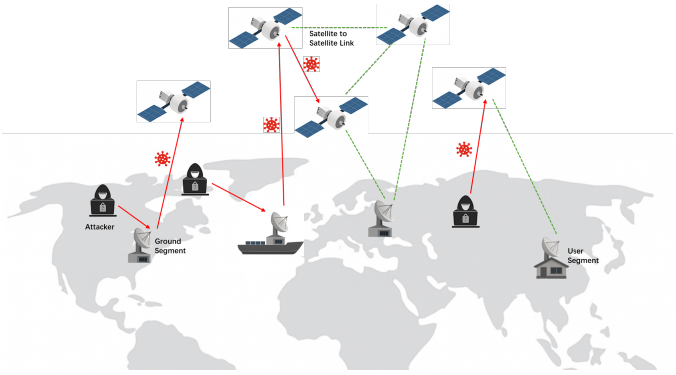


Fig. 1. Overview of Satellite System and Attack Vectors. The satellite system consists of space segment, the ground segment, and the user segment. Adversaries can target the ground infrastructure or attack satellites directly. Moreover, if a single satellite is compromised, the threat can propagate to other satellites through inter-satellite communication links, highlighting the systemic risks of federated constellations.

A range of toolchains supports this development process, including Xilinx Vitis AI[2], Intel's OpenVINO[3], hls4ml[4], FINN[5], and MATLAB's HDL Coder[6]. These tools enable systematic model partitioning, quantization, and deployment tailored to the underlying FPGA architecture. Moreover, partial reconfiguration (PR) facilitates dynamic updates to AI accelerators at runtime, ensuring uninterrupted system functionality. Using Isolation Design Flow (IDF), designers can maintain functional and security isolation between static and dynamic regions of the hardware, supporting safe and adaptive mission reconfiguration [39]. Collectively, these approaches reduce development cycles, enhance design flexibility, and improve the power efficiency of AI-enabled satellite systems.

### B. AI in Space Applications on SoC FPGA

Table I presents a comprehensive list of recent studies in satellite missions. Interestingly, these studies indicate a growing adoption of SoC FPGA platforms in space missions. Convolutional neural networks (CNN) are the most commonly deployed models, primarily due to their effectiveness in vision-based classification tasks. These are followed by autoencoders, spiking neural networks (SNNs), reinforcement learning algorithms, and graph neural networks (GNNs). Among the available platforms, AMD Xilinx solutions, particularly the Zynq UltraScale+ MPSoC, are the most prevalent, owing to their high reconfigurability, mature development ecosystem, and demonstrated reliability in space applications [44]. However, the combination of reconfigurable fabrics and their supporting software stacks, including FPGA bitstream managers, partial reconfiguration workflows, and AI runtime environments, introduces complex dependencies and attack surfaces that conventional software-only security measures are not ready to de-

[2]https://github.com/Xilinx/Vitis-AI
[3]https://github.com/openvinotoolkit/openvino
[4]https://github.com/fastmachinelearning/hls4ml
[5]https://finn.readthedocs.io/en/latest/
[6]https://www.mathworks.com/products/hdl-coder.html

TABLE I
SURVEY OF SOC FPGA-BASED AI IMPLEMENTATIONS FOR AEROSPACE APPLICATIONS.

| Authors | Year | AI Algorithms | SoC FPGA | Vendor | Applications |
|---|---|---|---|---|---|
| Pitsis et al. [28] | 2019 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Space data classification |
| Ma et al. [29], [30] | 2019 | Autoencoder NN | Zynq UltraScale+ MPSoC | AMD Xilinx | Feature extraction and anomaly detection |
| Sabogal et al. [31] | 2019 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Semantic segmentation of space imagery |
| Liu et al. [32] | 2019 | CNN | Arria 10 SX SoC | Intel Altera | Remote sensing image segmentation |
| Li et al. [33] | 2019 | SSD NN | Zynq 7000 | AMD Xilinx | Remote sensing imagery analysis |
| Reiter et al. [34] | 2020 | BNN | Zynq 7000 | AMD Xilinx | Real-time cloud detection |
| Lemaire et al. [35] | 2020 | BNN + CNN | Cyclone V | Intel Altera | Cloud classification and detection |
| Lent et al. [36] | 2020 | SNN | Zynq 7020 | AMD Xilinx | Routing in space networks |
| Cosmas et al. [37] | 2020 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Visual landmark recognition for navigation |
| Zhang et al. [38] | 2021 | YOLOv2 (CNN) | Zynq 7000 | AMD Xilinx | Optical object detection |
| Rapuano et al. [1] | 2021 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Cloud detection |
| Sabogal et al. [39] | 2021 | CNN | Zynq 7020, Zynq UltraScale+ MPSoC | AMD Xilinx | Semantic segmentation |
| Pacini et al. [40] | 2021 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Real-time image classification |
| Pitonak et al. [41] | 2022 | CNN | Zynq 7020 | AMD Xilinx | Cloud detection |
| Zhang et al. [42] | 2022 | GNN | Zynq UltraScale+ MPSoC | AMD Xilinx | SAR image classification |
| Abderrahmane et al. [43] | 2022 | SNN | Cyclone V | Intel Altera | Cloud detection |
| Papatheofanous et al. [44] | 2022 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Satellite image segmentation |
| Perryman et al. [45] | 2023 | MobileNetV1, ResNet-50, GoogLeNet | XCVC1902 (VCK190) | AMD Xilinx | Edge computing in space |
| Ekblad et al. [46] | 2023 | YOLOv4-based NN | Zynq UltraScale+ MPSoC | AMD Xilinx | Autonomous navigation |
| Gao et al. [47] | 2023 | CNN | Zynq 7000 | AMD Xilinx | CNN reliability evaluation |
| Carmeli et al. [48] | 2023 | SOM NN | Cyclone V | Intel Altera | Star pattern recognition |
| Coca et al. [49] | 2023 | ResNet | Zynq UltraScale+ MPSoC | AMD Xilinx | Burned area anomaly detection |
| Zhao et al. [50] | 2023 | YOLOv4-MobileNetv3 | Zynq UltraScale+ MPSoC | AMD Xilinx | Object detection in satellite images |
| Mazouz et al. [51] | 2024 | YOLOv3 | Zynq 7100 | AMD Xilinx | Streaming object detection |
| Kim et al. [52] | 2024 | Reinforcement Learning | Zynq 7000 | AMD Xilinx | Routing in LEO networks |
| Castelino et al. [53] | 2024 | Conv. Autoencoder | Zynq UltraScale+ MPSoC | AMD Xilinx | HSI artifact detection |
| Zhang et al. [54] | 2024 | Dehazing NN | Zynq 7000 | AMD Xilinx | Image dehazing |
| Cratere et al. [55] | 2024 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Cloud detection |
| Kim et al. [56] | 2024 | SqueezeNet | Zynq 7000 | AMD Xilinx | Cloud detection |
| Li et al. [57] | 2024 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Depth estimation in spacecraft |
| Upadhyay et al. [58] | 2024 | ResNetc | Zynq UltraScale+ MPSoC | AMD Xilinx | Cloud detection |
| Posso et al. [59] | 2024 | Mobile-URSONet | Zynq UltraScale+ MPSoC | AMD Xilinx | Pose estimation |
| Ciancarelli et al. [60] | 2024 | Autoencoders, CNNs | Xilinx ACAP | AMD Xilinx | Anomaly detection, SAR, RF |
| Leon et al. [61] | 2024 | UrsoNet, MobileNetV2, ResNet-50 | Zynq UltraScale+ MPSoC | AMD Xilinx | Pose estimation and benchmarking |
| Barnwal et al. [62] | 2024 | CNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Galaxy classification |
| Bai et al. [63] | 2024 | CNN | Zynq 7020 | AMD Xilinx | Particle identification |
| Jiang et al. [64] | 2024 | DNN | Zynq UltraScale+ MPSoC | AMD Xilinx | Hyperspectral anomaly detection |
| Shi et al. [65] | 2024 | CNN | Zynq-7000, UltraScale+ MPSoC | AMD Xilinx | Image classification |
| Justo et al. [66] | 2024 | CNN | Zynq 7030 | AMD Xilinx | Hyperspectral segmentation |
| Renaut et al. [67] | 2025 | DNN | Zynq 7000 | AMD Xilinx | Satellite pose estimation |
| Garcés-Socarrás et al. [68] | 2025 | CNN | VC190, Zynq UltraScale+ MPSoC | AMD Xilinx | Payload config and beamforming |
| Perryman et al. [69] | 2025 | CNN | XCVC1902 (VCK190), XCVE2802 (VEK280) | AMD Xilinx | Fault-tolerant AI acceleration |

fend. While prior research has focused primarily on improving inference throughput and energy efficiency, none of the works listed in Table I systematically address implementation-level platform security. This gap underscores the urgent need for cohesive, hardware-assisted security measures to ensure long-term system integrity and resilience in the unique operational context of space.

### C. Multi-Tenancy on SoC FPGA

In the context of SoC FPGAs, multi-tenancy refers to the partitioning of FPGA fabric into isolated regions, each dedicated to a different tenant [70]. As illustrated in Figure 2, the PL part is partitioned into multiple reconfigurable regions (e.g., $vFPGA_1$ and $vFPGA_2$), each capable of hosting isolated workloads from different tenants. These regions are managed and interfaced through a common *FPGA Shell*, which provides shared infrastructure and supports secure communication with PS. This architecture enables spatial isolation and dynamic partial reconfiguration of the programmable logic, allowing independent deployment and runtime updates of distinct AI or mission workloads within a single FPGA device, an essential capability for multi-payload and adaptive satellite missions. Leveraging PR, different applications can dynamically share FPGA resources without interference [71], [72]. This flexibility is essential for long-duration missions, enabling the hardware to adapt to evolving requirements such as new AI models, new instruments, or protocols over time [73].

Multi-tenancy enhances resource efficiency, which is especially critical for constrained platforms like CubeSats. However, it also introduces significant security challenges due to the shared nature of the hardware [74]. Potential risks include unauthorized access to a tenant's data or configuration, data leakage between tenants, and interference through side-channel attacks, such as those exploiting timing or power consumption [70]. These vulnerabilities are particularly critical in space-based systems, where adversarial environments and remote reconfiguration capabilities heighten the risk of attacks like bitstream tampering or hardware trojans [75].

### D. Security Challenges in Space-Based Systems

Space-based platforms, particularly those leveraging reconfigurable computing architectures such as SoC FPGAs, face complex and mission-critical security challenges. The distinctive constraints of the space environment, including radiation exposure, limited physical access, and intermittent communication with ground stations, exacerbate risks from both conventional and domain-specific cyber threats. The integration of AI accelerators, support for dynamic reconfiguration, and adoption of multi-tenant processing models further

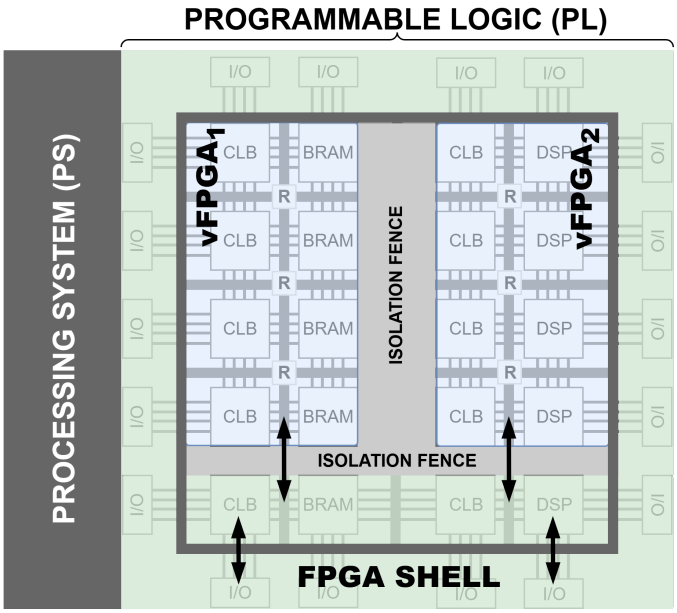


Fig. 2. Multi-Tenancy on SoC FPGA. CLB: Configurable Logic Block; R: Configurable Routing Block; BRAM: Block Random Access Memory; DSP: Digital Signal Processing; I/O: Input/Output Interface.

broaden the attack surface, demanding comprehensive end-to-end protection mechanisms.

This paper focuses on threats to platform integrity, especially those targeting the hardware reconfiguration pipeline, AI co-processing engines, and the overall system lifecycle. Key categories of threats include:

- **Fault-Induced Attacks:** Space radiation can cause Single-Event Upsets (SEUs) that alter configuration memory or computation logic. Deliberate fault injection (e.g., laser, EM pulses) can exploit similar vulnerabilities for code corruption or security bypass [76], [77].
- **Supply Chain Attacks:** Adversaries may insert malicious modifications during Integrated Circuit (IC) fabrication, IP core integration, or third-party toolchain use. Hardware Trojans, designed to remain dormant during functional verification, may activate under specific triggers to exfiltrate data or disrupt operations [75], [78].
- **Software Exploits:** Embedded Linux and Real-Time Operating System (RTOS) platforms on SoC FPGAs often include complex drivers, middleware, and AI toolchains. Vulnerabilities in these layers may enable unauthorized access to the programmable logic or privileged control interfaces [79], [80].
- **Reverse Engineering:** Through remote side channels, adversaries can extract proprietary bitstreams or reconstruct deployed AI model parameters, jeopardizing intellectual property and model confidentiality [81], [82].

Beyond general threat vectors, the secure operation of SoC FPGA-based satellites critically depends on defending against low-level, architecture-specific attacks. These vulnerabilities often arise at the intersection of software, firmware, and reconfigurable hardware, providing opportunities to compromise execution integrity, circumvent isolation mechanisms, or subvert critical AI functionalities. Addressing these foundational weaknesses is essential to maintain long-term trustworthiness in autonomous and remote satellite deployments.

- **Bitstream and AI Model Integrity:** Inadequate authentication of partial reconfiguration (PR) modules or AI model updates can allow injection of unauthorized logic. Adversaries may exploit this to embed covert computation units or poison AI inference results [83].
- **Hardware Trojanized Logic:** Third-party IP cores, if unchecked, may introduce hardware Trojans that evade static analysis and activate under rare conditions. These can undermine the system's trust boundary and persist undetected in deployed platforms [84]–[86].
- **Software Stack Exploits:** Improper memory protection or weak isolation between software layers may lead to privilege escalation, control flow hijacking, or logic reprogramming. This is particularly severe in embedded SoCs where software-hardware coupling is tight [87].

This work comprehensively examines all the above threat landscapes, with an emphasis on the interdependencies between reconfigurable hardware, embedded software, and AI workloads in space-based systems.

### *E. Cross-Layer and Lifecycle Security Requirements*

Ensuring the secure operation of AI-enabled SoC FPGA-based satellites requires a cross-layer security architecture covering the entire system lifecycle, from fabrication and provisioning to in-orbit updates and decommissioning. These systems face adversarial and resource-constrained environments, demanding tightly integrated hardware-software protections beyond standard embedded security.

A foundational element is the secure boot process, which uses cryptographic signatures to ensure firmware authenticity and integrity at startup, establishing a hardware-rooted chain of trust [88]. Runtime protections depend on isolation technologies such as ARM TrustZone, AXI firewalls, and Memory Protection Units (MPUs), which segment system components and restrict privilege escalation [88]. To support long-term missions, in-orbit platforms require secure firmware and AI model updates with cryptographic signing and rollback prevention, especially under connectivity delays [89], [90]. Meanwhile, physical-layer threats like radiation-induced Single Event Upsets (SEUs) necessitate resilience mechanisms such as Error-Correcting Codes (ECC), Triple Modular Redundancy (TMR), or dynamic reconfiguration to preserve system reliability [91]–[93].

## III. Threat Model

This paper adopts a comprehensive threat model reflecting the risks faced by AI-enabled SoC FPGA-based satellites in shared and adversarial environments. We assume the attacker has full knowledge of the satellite's hardware architecture, software stack, and operational workflows. Furthermore, the attacker is granted legitimate access via satellite-as-a-service

(SaaS) interfaces, allowing them to upload custom FPGA bitstreams or AI models onto shared computational payloads. Although other satellite applications are assumed benign, they are not trusted by default. Exploitation may arise from software bugs, weak isolation between the PS and PL, or insufficient runtime authentication. The attacker's objective may include unauthorized data access, disruption of mission-critical operations, or manipulation of AI inference outcomes.

## IV. Security Framework

Our security framework, AegisSat, adopts a layered architecture that integrates secure initialization, continuous runtime protection, and authenticated lifecycle management into a cohesive system. Rather than treating these components as isolated point solutions, our approach explicitly connects each stage of the platform's operation: secure boot establishes a hardware-rooted chain of trust that provisions cryptographic keys and baseline integrity measurements (IV-A); trusted execution environments maintain this trust throughout runtime by isolating sensitive assets and enforcing least-privilege execution (IV-B); and secure update workflows extend trust into the system's evolution by validating new configurations and AI models before activation (IV-C). During failed boot or updates, AegisSat provides robust fallback mechanisms that restore a known-good state from a golden image, ensuring service continuity and preventing persistent compromise. Together, these pillars form a defense-in-depth strategy, in which each layer reinforces and complements the others to ensure resilience against compromise, even in the absence of physical access or timely intervention.

### A. Secure Boot and Root of Trust for SoC FPGA Satellites

In space applications, where systems must operate reliably for extended periods without physical intervention, a robust, secure boot is essential to ensure that only authenticated and integrity-verified firmware, bitstreams, and AI models are executed onboard. This safeguards mission-critical operations against unauthorized modifications and cyber threats [94].

*1) Boot Sequence and Chain of Trust:* Secure boot in SoC FPGAs follows a hierarchical chain-of-trust model. The process begins with an immutable Boot ROM embedded in the silicon, which authenticates the first-stage bootloader (FSBL) using cryptographic algorithms such as RSA-4096 and SHA-3/384 [95], [96]. The FSBL initializes system components and verifies each subsequent stage, including loading the operating system and configuring the programmable logic fabric [97]. Cryptographic signatures and optional encryption are applied to software and bitstreams to ensure integrity and prevent tampering.

*2) Cryptographic Key Management:* Robust key management is essential for maintaining the integrity of the secure boot process. SoC FPGAs like the Zynq UltraScale+ MPSoC integrate mechanisms such as eFUSE arrays for storing public key hashes [98], [99]. eFUSE arrays are one-time programmable memory for permanent storage of sensitive data. Battery-backed RAM (BBRAM) is used for volatile key storage, which can be cleared in response to tampering. Physical Unclonable Functions (PUFs) enhance security by deriving keys from intrinsic hardware properties, eliminating the need for persistent storage [100]. Tamper detection circuits erase sensitive keys if anomalies in environmental parameters are detected, ensuring security throughout the satellite's lifecycle.

*3) Bitstream Authentication and Encryption:* To protect FPGA configurations, bitstreams are encrypted with AES-256 and authenticated using RSA or ECDSA signatures [101], [102]. AES-256 ensures strong symmetric encryption, while RSA and ECDSA verify digital signatures. In Xilinx Zynq UltraScale+ MPSoCs, the Configuration Security Unit (CSU) manages decryption and authentication during secure boot [103]. Intel's Stratix 10 and Agilex devices use a Secure Device Manager (SDM) with dedicated hardware engines for similar protections [104], [105]. To prevent replay attacks and unauthorized downgrades, modern FPGAs implement version control and anti-rollback mechanisms that compare stored version numbers against incoming bitstreams [106].

*4) Failure Handling and Recovery:* In case of boot failure, SoC FPGAs employ fallback mechanisms such as a non-overwritable golden image, retry logic for alternate configurations or safe mode, and watchdog timers for system recovery. A golden image is a pre-validated, secure configuration stored in protected memory. These methods are essential for space applications, where real-time intervention is constrained and environmental variability is high [107]–[109].

### B. Trusted Execution and Hardware Isolation

While secure boot establishes trust at system startup, the dynamic nature of AI-enabled satellite platforms requires continuous runtime protection [93], [110]. This section examines the runtime threat landscape in SoC FPGA-based satellites and presents key isolation mechanisms for least-privilege execution.

*1) Runtime Threat Landscape:* During in-orbit operation, vulnerabilities may arise beyond secure boot protections. Attackers could exploit flaws in AI inference engines [93], [111]–[113], OS services [114], or communication protocols [115] to execute arbitrary code or escalate privileges. Malicious bitstreams loaded during runtime reconfiguration might include unauthorized logic capable of memory access or bus manipulation [116]. The tight coupling between PS and PL could allow adversaries to traverse domains, causing data leakage, denial-of-service (DOS), or logic corruption [117]–[120].

*2) Execution Isolation in the Processing System:* The ARM TrustZone architecture partitions execution into Secure and Normal Worlds [121]. TrustZone is a security extension that enables secure execution of critical tasks (e.g., key management, firmware validation) in the Secure World, while general-purpose processes (e.g., AI inference, data handling) run in the Normal World [122]. TrustZone ensures memory and peripheral isolation and supports context switching via Secure Monitor Calls (SMCs) [123], [124]. Memory Protection Units (MPUs) and Memory Management Units (MMUs) provide

process-level access control, protecting AI models, telemetry data, and kernel-space buffers [125].

*3) Isolation of Programmable Logic:* The PL fabric interfaces with the PS often through Advanced eXtensible Interface (AXI) buses, playing a critical role in system-level communication. However, if compromised, the PL can become a conduit for unauthorized access to protected memory regions or peripheral devices. To mitigate such risks, platforms such as Xilinx offer hardware enforcements like the AXI Firewall IP [126] and the System Memory Management Unit (SMMU) [127], [128]. Other vendors, such as Intel, integrate similar security primitives in their FPGA SoCs, including configurable memory protection controllers and isolation-enabled bus interconnects, to ensure secure communication boundaries between hardware and software domains [102], [105], [129]. A robust security design also necessitates validating interrupt lines originating from the PL to the PS, especially when interfacing with high-risk modules like non-volatile storage or command subsystems [130], [131].

*4) Secure AI Co-Processing:* AI workloads deployed on SoC FPGAs often span both the PS and PL domains, requiring secure data exchange. Secure co-processing frameworks adopt cryptographic techniques such as Secure Hash Algorithms (e.g., SHA-256) to validate the integrity of AI computations. These hashes help ensure that data or intermediate results have not been tampered with during execution [113], [132]. Furthermore, runtime security monitors and HW/SW co-designed attestation schemes are employed to detect anomalous behaviors such as timing violations, unexpected control flow changes, or unauthorized access attempts [133]. These mechanisms form the backbone of a zero-trust execution environment, wherein no component, whether in the PS or PL, is inherently trusted without continuous verification.

*5) Privilege Separation and Containment:* Privilege separation limits the impact of compromises. In SoC FPGAs, the bootloader validates the launch chain, TrustZone manages cryptographic assets, and AI processes run in sandboxed environments. PL modules are restricted by memory and interrupt controls [128]. Localized resets or reconfigurations can preserve mission continuity if a subsystem (e.g., AI accelerator) fails, aided by watchdog timers and fault monitors [134].

### *C. Secure Update, Partial Reconfiguration, and AI Model Updates*

While runtime isolation ensures that malicious code cannot compromise critical functions, secure and resilient in-orbit updates are essential for the adaptability and longevity of AI-driven satellite platforms. While enabling on-the-fly improvements to mission logic and AI models [22], these capabilities also expose critical attack surfaces [135], [136]. The following section describes how authenticated update workflows extend and preserve trust throughout the system's lifecycle.

*1) Secure Update Workflow and Threat Mitigation:* Update commands, including logic bitstreams and AI model parameters, are transmitted via mutually authenticated and encrypted Telemetry, Tracking, and Command (TT&C) links [137]. TT&C links are the communication channels between ground stations and satellites. Commands include cryptographic signatures, sequence numbers, and timestamps and etc. to ensure integrity, prevent replay, and validate freshness [9], [138]. Upon receipt, updates are stored in protected memory and cryptographically validated before deployment [139]. Validation failures trigger automatic discards and logged alerts. Acknowledgment protocols and retry mechanisms address transient transmission failures [140]. This workflow mitigates threats such as bitstream injection, rollback attacks, and logic hijacking by enforcing staged validation and strict access control.

*2) Redundancy and Recovery Mechanisms:* To prevent mission degradation from failed updates, satellites retain a golden image, an immutable configuration stored in secure non-volatile memory [141], [142]. Watchdog timers monitor update behavior, reverting to the golden image upon detecting instability [143], [144]. Some architectures use dual-image buffers or redundant logic blocks for test deployments. Post-update, built-in self-tests (BIST) and output verification against golden baselines confirm operational correctness [145].

*3) Runtime Detection of Malicious Logic:* Runtime detection of malicious logic is critical for maintaining satellite security. Techniques include monitoring system behavior for anomalies, using intrusion detection systems, and leveraging hardware-based security features [146], [147]. Anomaly detection algorithms identify deviations from normal operation [148], [149], while hardware security modules (HSMs) protect sensitive operations. These measures ensure timely detection and mitigation of unauthorized logic or tampering [150].

*4) Partial Reconfiguration of FPGA Logic:* Partial Reconfiguration (PR) allows selective updates to FPGA logic without interrupting the rest of the system [151], [152]. For AI workloads, PR enables modular accelerator upgrades without full reboots [153]. Static and dynamic regions are defined using tools like Xilinx's Isolation Design Flow, enforcing strict boundaries. PR bitstreams are authenticated and decrypted using AES-256 and RSA/ECDSA before activation [154]. PR modules are sandboxed upon load, with behavior constrained by memory access restrictions and interface isolation. Validation with test vectors and integrity checks precede operational use [155]. A trusted controller in the static region orchestrates PR operations, ensuring traceability and mitigating logic-level attacks.

*5) AI Model Lifecycle and Security:* AI models must adapt to evolving tasks and environments. Software-based models are stored in encrypted memory and verified during runtime load [22]. Hardware-accelerated models (e.g., quantized neural networks) receive updates via PR or secure memory transactions. To prevent model poisoning [156], [157], updates are cryptographically signed and checked against expected behavior. Secure loading is complemented by functional testing, and in federated constellations, model propagation is governed by consensus or multi-signature authorization [26], [27].

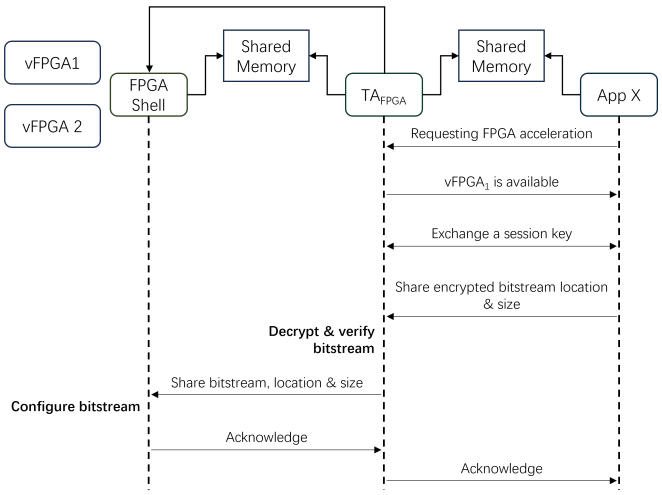


Fig. 3. Secure Reconfiguration Workflow.

*6) Operational and Scheduling Considerations:* In-orbit updates must align with operational safety and mission constraints. Updates are scheduled during idle periods with stable thermal and power profiles [158]. Configuration clocks must remain within tolerance, and critical maneuvers are paused during reconfiguration [159]. Failure handling includes retry logic, Cyclic Redundancy Check (CRC) validation, and rollback triggers [160]. A hardened PR controller ensures deterministic sequencing [161], while event logs support diagnostics and forensics [162].

## V. Implementation

To demonstrate the feasibility of the proposed security mechanisms, we implemented a proof-of-concept on the Xilinx ZCU102 development board. This platform integrates the Xilinx Zynq UltraScale+ MPSoC, which combines a quad-core ARM Cortex-A53 (with TrustZone), dual Cortex-R5 real-time cores, and FPGA programmable logic within a single chip. The hardware setup was configured and managed using Xilinx Vitis 2023.1. The device features dedicated secure boot capabilities via the Configuration Security Unit (CSU), hardware support for AES-256 encrypted bitstreams and RSA authentication, and TrustZone-based isolation for secure processing. During system provisioning, the eFUSE arrays were programmed with an RSA public key hash to enable cryptographic verification of all boot components. Secure boot was activated to ensure only authenticated firmware and FPGA bitstreams can be loaded.

The PL part was partitioned into three fixed regions: one static region hosting the FPGA shell and two dynamically reconfigurable virtual FPGA (vFPGA) partitions allocated to user applications. Each vFPGA was assigned dedicated I/O interfaces and unique memory address ranges. This setup allows independent management, configuration, and isolation of workloads. A trusted application on the PS part, referred to as the FPGA Trust Anchor ($TA_{\text{FPGA}}$), was implemented to control bitstream decryption, validation, and reconfiguration. The $TA_{\text{FPGA}}$ application executes within the secure world of TrustZone, while user applications ($AppX$) operate in the non-secure world. Inter-core communication was achieved using Xilinx Inter-Processor Interrupts (IPI) and shared memory. The secure reconfiguration workflow proceeds through the following coordinated steps among the system components:

1) *Request Initiation:* The $AppX$ application issues a request for FPGA acceleration to $TA_{\text{FPGA}}$.
2) *Resource Allocation:* $TA_{\text{FPGA}}$ checks the availability of reconfigurable regions and confirms that the required vFPGA is available for allocation.
3) *Session Key Exchange:* To establish a secure communication channel, $TA_{\text{FPGA}}$ and $AppX$ exchange a session key. This key is used to protect the confidentiality and integrity of the partial bitstream. Here, AES is exchanged, and RSA is used for encapsulating the AES session keys.
4) *Bitstream Preparation:* $AppX$ encrypts the partial bitstream through AES.
5) *Bitstream Transfer:* $AppX$ sends the encrypted partial bitstream and notifies $TA_{\text{FPGA}}$ of the bitstream's metadata, such as the intended location (vFPGA1 or vFPGA2) and its size over the secure channel.
6) *Decryption and Verification:* $TA_{\text{FPGA}}$ retrieves the encrypted partial bitstream from shared memory, decrypts, and verifies it.
7) *Configuration Instruction:* Once verified, $TA_{\text{FPGA}}$ provides the FPGA Shell with the decrypted partial bitstream, together with its intended location and size.
8) *Partial Reconfiguration:* The FPGA Shell configures the designated vFPGA region by streaming the partial bitstream through the internal configuration port (ICAP).
9) *Acknowledgment:* After successful configuration, the FPGA Shell acknowledges completion to $TA_{\text{FPGA}}$, which in turn issues a final acknowledgment to $AppX$.

This workflow ensures end-to-end protection of partial bitstream confidentiality and integrity while maintaining isolation between the control logic and user applications.

To illustrate practical usage, we implemented vFPGAs on the same FPGA. The statistics for the allocated size of blocks provide an estimate of resource usage for the two implemented applications on the FPGA, based on Vivado's floorplanning estimator. (1) *vFPGA1:* A lightweight CNN accelerator for image classification, comprising a 3×3 convolution, quantization, ReLU activation, and pooling, processing a 6×6 input feature map. Resource usage includes 30 CLB LUTs, 30 LUTs as Logic, 32 CLB Registers, 32 Registers as flip-flops, and one F7 Muxes. (2) *vFPGA2:* A configurable shift circuit supporting left and right shift operations. Resource usage includes 2 CLB LUTs, 2 LUTs as Logic, 35 CLB Registers, 35 Registers as Flip Flops, 5 CARRY8s, one Block RAM Tile, and RAMB36/FIFO.

The system was tested end-to-end by simulating a secure *ground station to satellite* reconfiguration scenario. $AppX$ acted as the ground station, transmitting a partial bitstream and emulating a radio link. $TA_{\text{FPGA}}$ works as the receiver

TABLE II
CONFIGURATION TIMES FOR DIFFERENT PARTIAL BITSTREAMS.

| Module | Mean Up. Time (ms) | Std. Dev. (ms) |
|---|---|---|
| vFPGA1 | 495.21 | 8.64 |
| vFPGA2 | 528.21 | 0.27 |

of the satellite. The update triggered secure boot procedures, cryptographic validation, and partial reconfiguration on the FPGA. Pre-emptive detection of hardware Trojans, especially those incorporating sensor or power-draining circuits, is essential prior to deploying configurations on the FPGA. Here we adopted the tool from [163] to analyze and identify malicious circuits. Despite minimal cryptographic overhead and multi-core coordination, the reconfiguration latency remained acceptable, demonstrating the approach's suitability for in-orbit reconfiguration scenarios. The reconfiguration time for each vFPGA is presented in Tables II. The configuration times were measured by performing 25 trials for each partial bitstream. The sample standard deviation reflects the variability within these limited trials. While the example CNN is lightweight, the same methodology supports more complex accelerators.

## VI. RESEARCH OUTLOOK

As future missions demand greater adaptability, resilience, and assurance in adversarial or disconnected environments, several technical and systemic challenges persist.

### A. Post-Quantum Cryptography

Current secure boot and update mechanisms rely on classical public-key cryptography, such as RSA and ECDSA, which are increasingly vulnerable to quantum attacks [164]. Given satellites' long service lifetimes, integrating post-quantum cryptographic (PQC) schemes into SoC FPGA toolchains is essential. Algorithms like CRYSTALS-Dilithium and SPHINCS+ are advancing toward NIST standardization [165]. However, practical implementation is still a major challenge. While proof-of-concept secure-boot demonstrations exist on terrestrial hardware, designing constant-time, radiation-tolerant PQC engines for space-grade, resource-constrained FPGAs remains largely unexplored. Optimizing PQC implementations for predictable timing and minimal memory use is also crucial to avoid side-channel vulnerabilities and ensure reliability in orbit.

### B. Autonomous Attestation in Disconnected Missions

While secure boot establishes a root of trust at startup, long-duration deep-space missions require continuous trust evaluation without real-time ground oversight. Autonomous attestation mechanisms are needed to detect unexpected behavior, validate reconfiguration events, and respond to tampering evidence. Notably, the development of in-orbit autonomous attesters capable of continual self-measurement of the PL fabric under severe communication delays remains an open field. Future work should explore lightweight, self-verifying architectures to provide resilience in disconnected scenarios.

### C. Energy-Constrained Isolation Mechanisms

Isolation mechanisms such as ARM TrustZone, AXI Firewalls, and SMMUs are critical for domain separation but may introduce latency and power overhead in energy-constrained platforms like CubeSats. Optimized isolation techniques at RTL and fabric level are needed, including logic-privilege tags, dynamic bus gating, and low-overhead privilege-checking circuits. Additionally, energy-adaptive security policies, wherein the isolation strength dynamically scales according to battery state and mission phase, remain largely unexplored.

### D. Cross-Domain Hardware-Software Co-Design

Securing AI-driven SoC FPGA satellites demands a holistic approach across the AI model lifecycle, from development to deployment and updates. Integrated toolchains must address FPGA synthesis, AI compiler output validation, secure key provisioning, and post-deployment telemetry, while accounting for aerospace constraints such as radiation, communication windows, and thermal budgets [166], [167]. Open-source toolchains could balance security and accessibility [168], [169]. Interdisciplinary collaboration is essential to develop standardized frameworks that meet mission requirements. A critical yet underexplored area is the integration of hardware-assisted side-channel monitors within the FPGA fabric itself, such as embedded glitch sensors and power anomaly detectors, to detect physical or electromagnetic probing attempts in orbit.

### E. Federated Learning Security in Satellite Swarms

Finally, emerging mission concepts envision large constellations of cooperating small satellites performing federated AI training and inference. Although secure aggregation protocols have been explored in terrestrial settings, federated learning security against poisoning and Byzantine faults across hundreds of Low Earth Orbit nodes remains unaddressed in hardware-rooted frameworks. Developing scalable, resilient architectures to enable trustworthy federated learning in space represents a promising frontier [113], [157], [170].

## VII. CONCLUSION

This paper addresses the pressing need for secure AI-enabled satellite platforms by presenting a comprehensive security framework, AegisSat tailored for SoC FPGAs. Recognizing the unique challenges posed by reconfigurable computing in space,AegisSat integrates multiple layers of protection. This holistic approach establishes a trusted chain from system initialization to ongoing operations, critical for maintaining mission integrity. By laying this security foundation, our work supports the safe and effective deployment of AI in space, enabling the next generation of intelligent satellite systems. As satellite technology evolves, future research must focus on adapting to new threats and constraints.

## ACKNOWLEDGEMENT

Our research work was partially funded by Intel's Scalable Assurance Program, Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297, the European Union under

Horizon Europe Programme – Grant Agreement 101070537 – CrossCon, and the European Research Council under the ERC Programme - Grant 101055025 - HYDRANOS. This work does not in any way constitute an Intel endorsement of a product or supplier. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect those of Intel, the European Union, and the European Research Council.

## REFERENCES


[1] E. Rapuano, G. Meoni, T. Pacini, G. Dinelli, G. Furano, G. Giuffrida, and L. Fanucci, "An fpga-based hardware accelerator for cnns inference on board satellites: benchmarking with myriad 2-based solution for the cloudscout case study," *Remote Sensing*, vol. 13, no. 8, p. 1518, 2021.

[2] A. D. George and C. M. Wilson, "Onboard processing with hybrid and reconfigurable computing on small satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 458–470, 2018.

[3] M. M. I. Mohamed, "Development and qualification of an fpga-based multi-processor system-on-chip on-board computer for leo satellites," 2014.

[4] C. M. Fuchs, T. P. Stefanov, N. M. Murillo, and A. Plaat, "Bringing fault-tolerant gigahertz-computing to space: A multi-stage software-side fault-tolerance approach for miniaturized spacecraft," in *2017 IEEE 26th Asian Test Symposium (ATS)*. IEEE, 2017, pp. 100–107.

[5] Z. Zhao, "Reconfiguration control networks for fpga-based space applications," Ph.D. dissertation, UNSW Sydney, 2016.

[6] M. Petry, A. Koch, and M. Werner, "Zero-copy ai-augmented signal processing pipeline on heterogeneous satellite processors," in *2024 58th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2024, pp. 354–361.

[7] M. Petry, P. Gest, A. Koch, M. Ghiglione, and M. Werner, "Accelerated deep-learning inference on fpgas in the space domain," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, 2023, pp. 222–228.

[8] J. N. Pelton and S. Madry, "Handbook of small satellites," *Cham: Springer International Publishing*, 2020.

[9] S. Vasudevan, "Design and development of a cubesat hardware architecture with cots mpsoc using radiation mitigation techniques," 2020.

[10] T. Kuwahara, "Fpga-based reconfigurable on-board computing systems for space applications," 2010.

[11] C. Léonard, D. Stober, and M. Schulz, "Fpga-enabled machine learning applications in earth observation: A systematic review," *arXiv preprint arXiv:2506.03938*, 2025.

[12] A. Raoofy, G. Dax, M. Ghiglione, V. Serra, M. Werner, and C. Trinitis, "Invited Talk: Benchmarking and Feasibility Aspects of Machine Learning in Space Systems."

[13] D. D. Langer, M. Orlandić, S. Bakken, R. Birkeland, J. L. Garrett, T. A. Johansen, and A. J. Sørensen, "Robust and reconfigurable on-board processing for a hyperspectral imaging small satellite," *Remote Sensing*, vol. 15, no. 15, p. 3756, 2023.

[14] F. Viel and C. A. Zeferino, "A module for remote reconfiguration of fpgas in satellites," in *IBERCHIP workshop*, 2017, pp. 50–53.

[15] U. Legat, A. Biasizzo, and F. Novak, "Seu recovery mechanism for sram-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 59, no. 5, pp. 2562–2571, 2012.

[16] S. L. Eine, D. D. Langerb, R. Birkeland, and M. Orlandicc, "Reconfiguration of fpga during operation of small satellite for flexible hyperspectral data compression," 2024.

[17] C. R. M. Kiruki, "Study on fpga-based on-board inferencing for spacecrafts in autonomous operations," 2021.

[18] N. Boschetti, N. G. Gordon, and G. Falco, "Space cybersecurity lessons learned from the viasat cyberattack," in *ASCEND 2022*, 2022, p. 4380.

[19] N. Yadav, F. Vollmer, A.-R. Sadeghi, G. Smaragdakis, and A. Voulimeneas, "Orbital shield: Rethinking satellite security in the commercial off-the-shelf era," in *2024 Security for Space Systems (3S)*. IEEE, 2024, pp. 1–11.

[20] J. Li, N. Song, X. Jia, and L. Wang, "An fpga partial reconfiguration method for satellite cryptographic device verification," in *International Conference on Cryptography, Network Security, and Communication Technology (CNSCT 2023)*, vol. 12641. SPIE, 2023, pp. 46–52.

[21] P. Cotret, G. Gogniat, J.-P. Diguet, and J. Crenne, "Lightweight reconfiguration security services for axi-based mpsocs," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2012, pp. 655–658.

[22] N. Vallez, R. Rodriguez-Bobada, A. Dunne, and J. L. Espinosa-Aranda, "Efficient in-orbit cnn updates," in *2023 European Data Handling & Data Processing Conference (EDHPC)*. IEEE, 2023, pp. 1–5.

[23] M. Swartwout, "The first one hundred cubesats: A statistical look," *Journal of Small Satellites*, vol. 2, no. 2, pp. 213–233, 2016.

[24] M. Handley, "Delay is not an option: Low latency routing in space," *Proceedings of the ACM SIGCOMM*, pp. 411–426, 2019.

[25] D. Zelenyi, ""Satellite-as-a-Service": A New Approach for Space Industry," Exodus Orbitals, Tech. Rep., 2020, https://www.exodusorbitals.com/files/whitepaper.pdf.

[26] M. A. Ferrag, B. Kantarci, L. C. Cordeiro, M. Debbah, and K.-K. R. Choo, "Poisoning attacks in federated edge learning for digital twin 6g-enabled iots: An anticipatory study," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2023, pp. 1253–1258.

[27] P. H. Barros and H. S. Ramos, "A novel aggregation method to promote safety security for poisoning attacks in federated learning," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 3869–3874.

[28] G. Pitsis, G. Tsagkatakis, C. Kozanitis, I. Kalomoiris, A. Ioannou, A. Dollas, M. G. Katevenis, and P. Tsakalides, "Efficient convolutional neural network weight compression for space data classification on multi-fpga platforms," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3917–3921.

[29] N. Ma, X. Yu, Y. Peng, and S. Wang, "A lightweight hyperspectral image anomaly detector for real-time mission," *Remote Sensing*, vol. 11, no. 13, p. 1622, 2019.

[30] P. Antunes and A. Podobas, "Fpga-based neural network accelerators for space applications: A survey," *arXiv preprint arXiv:2504.16173*, 2025.

[31] S. Sabogal, A. George, and G. Crum, "Recon: A reconfigurable cnn acceleration framework for hybrid semantic segmentation on hybrid socs for space applications," in *2019 IEEE Space Computing Conference (SCC)*. IEEE, 2019, pp. 41–52.

[32] S. Liu and W. Luk, "Towards an efficient accelerator for dnn-based remote sensing image segmentation on fpgas," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019, pp. 187–193.

[33] L. Li, S. Zhang, and J. Wu, "Efficient object detection framework and hardware architecture for remote sensing images," *Remote Sensing*, vol. 11, no. 20, p. 2376, 2019.

[34] P. Reiter, P. Karagiannakis, M. Ireland, S. Greenland, and L. Crockett, "Fpga acceleration of a quantized neural network for remote-sensed cloud detection," in *7th International Workshop on On-Board Payload Data Compression*, 2020.

[35] E. Lemaire, M. Moretti, L. Daniel, B. Miramond, P. Millet, F. Feresin, and S. Bilavarn, "An fpga-based hybrid neural network accelerator for embedded satellite image classification," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[36] R. Lent, "Evaluating the cognitive network controller with an snn on fpga," in *2020 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. IEEE, 2020, pp. 106–111.

[37] K. Cosmas and A. Kenichi, "Utilization of fpga for onboard inference of landmark localization in cnn-based spacecraft pose estimation," *Aerospace*, vol. 7, no. 11, p. 159, 2020.

[38] N. Zhang, X. Wei, H. Chen, and W. Liu, "Fpga implementation for cnn-based optical remote sensing object detection," *Electronics*, vol. 10, no. 3, p. 282, 2021.

[39] S. Sabogal, A. George, and G. Crum, "Reconfigurable framework for resilient semantic segmentation for space applications," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 14, no. 4, pp. 1–32, 2021.

[40] T. Pacini, E. Rapuano, G. Dinelli, and L. Fanucci, "A multi-cache system for on-chip memory optimization in fpga-based cnn accelerators," *Electronics*, vol. 10, no. 20, p. 2514, 2021.

[41] R. Pitonak, J. Mucha, L. Dobis, M. Javorka, and M. Marusin, "Cloudsatnet-1: Fpga-based hardware-accelerated quantized cnn for satellite on-board cloud coverage classification," *Remote Sensing*, vol. 14, no. 13, p. 3180, 2022.

[42] B. Zhang, R. Kannan, V. Prasanna, and C. Busart, "Accurate, low-latency, efficient sar automatic target recognition on fpga," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2022, pp. 1–8.

[43] N. Abderrahmane, B. Miramond, E. Kervennic, and A. Girard, "Spleat: Spiking low-power event-based architecture for in-orbit processing of satellite imagery," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–10.

[44] E.-A. Papatheofanous, P. Tziolos, V. Kalekis, T. Amrou, G. Konstantoulakis, G. Venitourakis, and D. Reisis, "Soc fpga acceleration for semantic segmentation of clouds in satellite images," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2022, pp. 1–4.

[45] N. Perryman, C. Wilson, and A. George, "Evaluation of xilinx versal architecture for next-gen edge computing in space," in *2023 IEEE aerospace conference*. IEEE, 2023, pp. 1–11.

[46] A. Ekblad, T. Mahendrakar, R. White, M. Wilde, I. Silver, and B. Wheeler, "Resource-constrained fpga design for satellite component feature extraction," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–9.

[47] Z. Gao, S. Gao, Y. Yao, Q. Liu, S. Zeng, G. Ge, Y. Wang, A. Ullah, and P. Reviriego, "Systematic reliability evaluation of fpga implemented cnn accelerators," *IEEE Transactions on Device and Materials Reliability*, vol. 23, no. 1, pp. 116–126, 2023.

[48] G. Carmeli and B. Ben-Moshe, "Ai-based real-time star tracker," *Electronics*, vol. 12, no. 9, p. 2084, 2023.

[49] M. Coca and M. Datcu, "Fpga accelerator for meta-recognition anomaly detection: Case of burned area detection," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 5247–5259, 2023.

[50] Y. Zhao, Y. Lv, and C. Li, "Hardware acceleration of satellite remote sensing image object detection based on channel pruning," *Applied Sciences*, vol. 13, no. 18, p. 10111, 2023.

[51] A. E. Mazouz and V.-T. Nguyen, "Online continual streaming learning for embedded space applications," *Journal of Real-Time Image Processing*, vol. 21, no. 3, p. 68, 2024.

[52] H. Kim, J. Park, H. Lee, D. Won, and M. Han, "An fpga-accelerated cnn with parallelized sum pooling for onboard realtime routing in dynamic low-orbit satellite networks," *Electronics*, vol. 13, no. 12, p. 2280, 2024.

[53] C. Castelino, S. Khandelwal, S. Shreejith, and S. V. Bogaraju, "An energy-efficient artefact detection accelerator on fpgas for hyperspectral satellite imagery," in *2024 27th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2024, pp. 551–558.

[54] Z. Zhang, G. Du, Z. Li, Q. Kang, W. Zhao, and X. Wang, "An energy-efficient dehazing neural network accelerator based on e 2 aod-net," *Journal of Real-Time Image Processing*, vol. 21, no. 6, p. 197, 2024.

[55] A. Cratere, M. S. Farissi, A. Carbone, M. Asciolla, M. Rizzi, F. Dell'Olio, A. Nascetti, and D. Spiller, "Efficient fpga-accelerated convolutional neural networks for cloud detection on cubesats," *IEEE Journal on Miniaturization for Air and Space Systems*, 2025.

[56] J.-H. Kim, Y. Kim, D.-H. Cho, and S.-M. Kim, "On-orbit ai: Cloud detection technique for resource-limited nanosatellite," *International Journal of Aeronautical and Space Sciences*, pp. 1–14, 2024.

[57] J. Li, C. Zhang, W. Yang, H. Li, X. Wang, C. Zhao, S. Du, and Y. Liu, "Fpga-based low-bit and lightweight fast light field depth estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2024.

[58] G. Upadhyay, S. Ghosal, S. Kar, K. Jain, S. SH, S. A. Balwantrao *et al.*, "Design and implementation of cnn-based custom net architecture with improved inference time for realtime remote sensing application," in *2024 IEEE Space, Aerospace and Defence Conference (SPACE)*. IEEE, 2024, pp. 647–651.

[59] J. Posso, G. Bois, and Y. Savaria, "Real-time spacecraft pose estimation using mixed-precision quantized neural network on cots reconfigurable mpsoc," in *2024 22nd IEEE Interregional NEWCAS Conference (NEWCAS)*. IEEE, 2024, pp. 358–362.

[60] C. Ciancarelli, D. di Ienno, R. Trois, L. Scandelli, C. de Biase, P. Serri, A. Leboffe, D. Pascucci, D. Steenari, and G. Furano, "Special session: Exploring the potential of versal acap: Advancing onboard edge ai for spacecraft," in *2024 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2024, pp. 1–5.

[61] V. Leon, P. Minaidis, D. Soudris, and G. Lentaris, "Mpai: A co-processing architecture with mpsoc & ai accelerators for vision applications in space," in *2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2024, pp. 1–2.

[62] R. Barnwal and S. Kala, "Morphological galaxy classification using convolutional neural networks on fpga," in *2024 IEEE Space, Aerospace and Defence Conference (SPACE)*. IEEE, 2024, pp. 190–193.

[63] C. Bai, X. Zhang, S. Zhang, Y. Sun, X. Zhang, Z. Wang, and S. Zhang, "Design and application of an onboard particle identification platform based on convolutional neural networks," *Applied Sciences*, vol. 14, no. 15, p. 6628, 2024.

[64] Y. Jiang, A. Vaicaitis, J. Dooley, and M. Leeser, "Efficient neural networks on the edge with fpgas by optimizing an adaptive activation function," *Sensors*, vol. 24, no. 6, p. 1829, 2024.

[65] D. Shi, J. Gao, and Y. Zhang, "Aircraft image management based on ai typical mainland zynq," in *2024 2nd International Conference on Machine Vision, Image Processing & Imaging Technology (MVIPIT)*. IEEE, 2024, pp. 187–192.

[66] J. A. Justo, D. D. Langer, S. Berg, J. Nieke, R. T. Ionescu, P. G. Kjeldsberg, and T. A. Johansen, "Hyperspectral image segmentation for optimal satellite operations: In-orbit deployment of 1d-cnn," 2024.

[67] L. Renaut, H. Frei, and A. Nüchter, "Deep learning on 3d point clouds for fast pose estimation during satellite rendezvous," *Acta Astronautica*, vol. 232, pp. 231–243, 2025.

[68] L. M. Garcés-Socarrás, A. Nik, F. Ortiz, J. A. Vásquez-Peralvo, J. L. G. Rios, M. Chehailty, M. Kuhfuss, E. Lagunas, J. Thoemel, S. Kumar *et al.*, "Artificial intelligence implementation of onboard flexible payload and adaptive beamforming using commercial off-the-shelf devices," *arXiv preprint arXiv:2505.01853*, 2025.

[69] N. Perryman, S. Sabogal, C. Wilson, and A. George, "Dependable dpu architectures on amd-xilinx versal adaptive socs for space applications," *IEEE Transactions on Aerospace and Electronic Systems*, 2025.

[70] G. Dessouky, A.-R. Sadeghi, and S. Zeitouni, "Sok: Secure fpga multi-tenancy in the cloud: Challenges and opportunities," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 487–506.

[71] D. Koch, J. Torresen, C. Beckhoff, D. Ziener, C. Dennl, V. Breuer, J. Teich, M. Feilen, and W. Stechele, "Partial reconfiguration on fpgas in practice—tools and applications," in *ARCS 2012*. IEEE, 2012, pp. 1–12.

[72] A. Karteris, E. Tsigkanos, M. Bernou, A. Chatzistylianos, and G. Lentaris, "Towards ai onboard eo satellites: Assessment of virtualization techniques for extreme edge computing," in *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2024, pp. 1727–1732.

[73] European Space Agency, "The use of reprogrammable fpgas in space," https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Microelectronics/The_use_of_reprogrammable_FPGAs_in_space, 2023, accessed: 2025-06-18.

[74] S. Meda and S. Morapally, "Secure multi-tenant architectures for cloud-based space mission operations: Mitigating risks in shared environments," *International Journal of Communication Networks and Information Security*, vol. 14, 2022.

[75] R. Elnaggar, R. Karri, and K. Chakrabarty, "Multi-tenant fpga-based reconfigurable systems: Attacks and defenses," in *2019 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2019, pp. 7–12.

[76] M. J. Campola and J. A. Pellish, "Radiation hardness assurance: Evolving for newspace," in *European Conference on Radiation Effects on Components and Systems (RADECS 2019)*, no. GSFC-E-DAA-TN72757, 2019.

[77] Y. Ren, M. Zhu, D. Xu, M. Liu, X. Dai, S. Wang, and L. Li, "Overview on radiation damage effects and protection techniques in microelectronic devices," *Science and Technology of Nuclear Installations*, vol. 2024, no. 1, p. 3616902, 2024.

[78] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[79] M. H. Rahman, "A comprehensive survey on hardware-software co-protection against invasive, non-invasive and interactive security threats," *Cryptology ePrint Archive*, 2024.

[80] S. Jero, J. Furgala, M. A. Heller, B. Nahill, S. Mergendahl, and R. Skowyra, "Securing the satellite software stack," in *Proceedings 2024 Workshop on Security of Space and Satellite Systems, San Diego, CA, USA: Internet Society*, 2024.

[81] J. Zhang and G. Qu, "Recent attacks and defenses on fpga-based systems," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–24, 2019.

[82] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.

[83] W. Xing, M. Li, M. Li, and M. Han, "Towards robust and secure embodied ai: A survey on vulnerabilities and attacks," *arXiv preprint arXiv:2502.13175*, 2025.

[84] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1010–1038, 2020.

[85] T. Hoque, "Ip trust assurance through design and runtime solutions," Ph.D. dissertation, University of Florida, 2020.

[86] D. Giuffrida, "A foss-based toolchain for automated hardware trojan injection in risc-v architectures," Ph.D. dissertation, Politecnico di Torino, 2024.

[87] S. R. Rajendran, N. F. Dipu, S. Tarek, H. M. Kamali, F. Farahmandi, and M. Tehranipoor, "Exploring the abyss? unveiling systems-on-chip hardware vulnerabilities beneath software," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3914–3926, 2024.

[88] J. Amacher and V. Schiavoni, "On the performance of arm trustzone: (practical experience report)," in *Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings 19*. Springer, 2019, pp. 133–151.

[89] I. Sünter, A. Slavinskis, U. Kvell, A. Vahter, H. Kuuste, M. Noorma, J. Kutt, R. Vendt, K. Tarbe, M. Pajusalu *et al.*, "Firmware updating systems for nanosatellites," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 5, pp. 36–44, 2016.

[90] A. Marchand, Y. Imine, H. Ouarnoughi, T. Tarridec, and A. Gallais, "Firmware integrity protection: A survey," *IEEE Access*, vol. 11, pp. 77 952–77 979, 2023.

[91] P. Yue, J. An, J. Zhang, G. Pan, S. Wang, P. Xiao, and L. Hanzo, "On the security of leo satellite communication systems: Vulnerabilities, countermeasures, and future trends," *Authorea Preprints*, 2022.

[92] R. R. Shrivastwa, "Enhancements in embedded systems security using machine learning," Ph.D. dissertation, Institut Polytechnique de Paris, 2023.

[93] L. Diana and P. Dini, "Review on hardware devices and software techniques enabling neural network inference onboard satellites," *Remote Sensing*, vol. 16, no. 21, p. 3957, 2024.

[94] AMD Xilinx, "Zynq ultrascale+ mpsoc security features," https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841708/Zynq%2BUltrascale%2BMPSoC%2BSecurity%2BFeatures, 2020, accessed: 2025-06-18.

[95] Intel Altera, "Intel soc fpgas secure boot user guide," https://www.intel.com/content/www/us/en/docs/programmable/683735/current/arria-10-soc-boot-user-guide.html, Intel Corporation, Tech. Rep., 2020, accessed: 2025-06-18.

[96] AMD Xilinx, "Secure boot and bitstream authentication on zynq-7000 soc," https://docs.amd.com/v/u/en-US/xapp1175_zynq_secure_boot, Xilinx Application Note XAPP1175, Tech. Rep., 2015, accessed: 2025-06-18.

[97] Microsemi, Inc, "Overview of secure boot with microsemi igloo2 fpgas," 2013, accessed: 2025-06-18.

[98] AMD Xilinx, "Zynq-7000 ap soc security," https://xilinxwiki.atlassian.net/wiki/spaces/A/pages/18841646/Zynq7000+AP+SoC+Security, Sep. 2018, accessed: 2025-06-22; Last updated: 24 Sept 2018.

[99] Intel Altera, "Intel stratix 10 configuration user guide," https://cdrdv2.intel.com/v1/dl/getContent/849885?fileName=ug-s10-config-683762-849885.pdf, User Guide UG-S10CONFIG, revision as of 2025-04-07.

[100] Microsemi, "Physically unclonable functions in smartfusion2," https://www.microsemi.com/products/fpga-soc/security/secure-boot, accessed: 2025-06-18.

[101] AMD Xilinx, *Zynq UltraScale+ MPSoC Technical Reference Manual*, https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug1137-zynq-ultrascale-mpsoc-swdev.pdf, 2022, accessed: 2025-06-18.

[102] Intel Altera, "Intel stratix 10 device security user guide," Santa Clara, CA, USA, User Guide UG-S10SECURITY (683642), Jul. 2023. [Online]. Available: https://www.intel.com/content/www/us/en/docs/programmable/683642.html

[103] AMD Xilinx, "Zynq ultrascale+ mpsoc: Software developer's guide," 2020, https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug1137-zynq-ultrascale-mpsoc-swdev.pdf.

[104] Intel Altera, "Agilex 7 fpgas and socs device overview," Device Overview 683458, Mar. 2025. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/666707?fileName=ag-overview-683458-666707.pdf

[105] T. Lu, R. Kenny, and S. Atsatt, "Secure device manager for intel stratix 10 devices provides fpga and soc security," Intel Corporation, White Paper WP-01252-1.2, 2023, describes SDM-based FPGA/SoC configuration, sector-level security, PUF, and encryption features.

[106] Intel Altera, "Security user guide: Intel fpga programmable acceleration card d5005," User Guide D5005 (683877), Aug. 2019, covers Root of Trust (RoT), AFU signing, FIM/BMC security features. [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/679776?fileName=ug-pac-security-d5005-683877-679776.pdf

[107] N. Dzemaili, "A reliable booting system for zynq ultrascale+ mpsoc devices," Ph.D. dissertation, CERN, 2021.

[108] F. Siegle, "Fault detection, isolation and recovery schemes for space-borne reconfigurable fpga-based systems," Ph.D. dissertation, University of Leicester, 2016.

[109] S. G. La Greca, "Study and development of fault tolerant operating systems on fpga for aerospace applications," Ph.D. dissertation, Politecnico di Torino, 2022.

[110] M. Gross, N. Jacob, A. Zankl, and G. Sigl, "Breaking trustzone memory isolation and secure boot through malicious hardware on a modern fpga-soc," *Journal of Cryptographic Engineering*, vol. 12, no. 2, pp. 181–196, 2022.

[111] P. Rech, "Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions," *IEEE Transactions on Nuclear Science*, vol. 71, no. 4, pp. 377–404, 2024.

[112] M. Wang, H. Qiu, L. Xu, D. Wang, Y. Li, T. Zhang, J. Liu, and H. Li, "A case for application-aware space radiation tolerance in orbital computing," *arXiv preprint arXiv:2407.11853*, 2024.

[113] H. Li, P. Rieger, S. Zeitouni, S. Picek, and A.-R. Sadeghi, "Flairs: Fpga-accelerated inference-resistant & secure federated learning," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2023, pp. 271–276.

[114] A. Cratere, L. Gagliardi, G. A. Sanca, F. Golmar, and F. Dell'Olio, "On-board computer for cubesats: State-of-the-art and future trends," *IEEE Access*, 2024.

[115] P. Yue, J. An, J. Zhang, J. Ye, G. Pan, S. Wang, P. Xiao, and L. Hanzo, "Low earth orbit satellite security and reliability: Issues, solutions, and the road ahead," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1604–1652, 2023.

[116] P. A. Oche, G. A. Ewa, and N. Ibekwe, "Applications and challenges of artificial intelligence in space missions," *IEEE Access*, vol. 12, pp. 44 481–44 509, 2021.

[117] A. Diro, "Space systems and malware: Potential threats," in *Ransomware Evolution*. CRC Press, 2025, pp. 208–232.

[118] M. M. Utsash, "Implementing zero-trust for securing spacecraft," Master's thesis, NTNU, 2024.

[119] C. Wu, Y. Li, M. Xu, C. Guo, Z. Yin, W. Gao, and C. Chi, "A comprehensive survey on orbital edge computing: Systems, applications, and algorithms," *arXiv preprint arXiv:2306.00275*, 2023.

[120] A. Geist, G. Crum, C. Brewer, D. Afanasev, S. Sabogal, D. Wilson, J. Goodwill, J. Marshall, N. Perryman, N. Franconi *et al.*, "Nasa spacecube next-generation artificial-intelligence computing for stp-h9-scenic on iss," 2023.

[121] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM computing surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[122] Z. Zhang, "Enhancing iot security through trusted execution environments," in *2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*. Atlantis Press, 2024, pp. 580–589.

[123] M. S. Islam, "Confidential computing with trusted execution environments," Ph.D. dissertation, 2024.

[124] Z. Jian, X. Liu, Q. Dong, L. Cheng, X. Xie, and T. Li, "Smartzone: Runtime support for secure and efficient on-device inference on arm trustzone," *IEEE Transactions on Computers*, 2025.

[125] H. Huang, F. Zhang, S. Yan, T. Wei, and Z. He, "Sok: A comparison study of arm trustzone and cca," in *2024 International Symposium on Secure and Private Execution Environment Design (SEED)*. IEEE, 2024, pp. 107–118.

[126] AMD Xilinx, "Axi firewall ip documentation," https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug1137-zynq-ultrascale-mpsoc-swdev.pdf, 2022, accessed: 2025-06-18.

[127] M. Gross, N. Jacob, A. Zankl, and G. Sigl, "Breaking trustzone memory isolation through malicious hardware on a modern fpga-soc," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, 2019, pp. 3–12.

[128] I. Yarza, I. Agirre, I. Mugarza, and J. P. Cerrolaza, "Safety and security collaborative analysis framework for high-performance embedded computing devices," *Microprocessors and Microsystems*, vol. 93, p. 104572, 2022.

[129] A. Proulx, J.-Y. Chouinard, P. Fortier, and A. Miled, "A survey on fpga cybersecurity design strategies," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 2, pp. 1–33, 2023.

[130] I. Christoforakis, "Protection and safety framework for on-chip communications and mixed-critical cyber-physical systems," 2020.

[131] A. Silitonga, H. Gassoumi, and J. Becker, "Mites: Software-based microarchitectural attacks and countermeasures in networked ap soc platforms," in *2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*. IEEE, 2020, pp. 65–71.

[132] "Sha-256 standard," https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf, NIST, 2015.

[133] D. Solet, S. Pillement, J.-L. Béchennec, M. Briday, and S. Faucou, "Hw-based architecture for runtime verification of embedded software on sopc systems," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, 2018, pp. 249–256.

[134] F. Fons, M. Fons, P. Olivier, and A. Weimerskirch, "A modular reconfigurable and updateable embedded cyber security hardware solution for automotive," in *Proc. Embedded World Conf.*, 2017, pp. 1–10.

[135] S. P. Singh, S. Verma, P. Pali, H. Tiwari, and S. Kanojiya, "Application of artificial intelligence (ai) to enhance satellite security," *International Journal of Innovative Research in Computer and Communication Engineering*, 2023.

[136] K. Thangavel, R. Sabatini, A. Gardi, K. Ranasinghe, S. Hilton, P. Servidia, and D. Spiller, "Artificial intelligence for trusted autonomous satellite operations," *Progress in Aerospace Sciences*, vol. 144, p. 100960, 2024.

[137] C. Bader, "On requirements & concepts for tt&c link key management," in *2nd Workshop on the Security of Space and Satellite Systems (SpaceSec)*, 2024.

[138] M. I. Morrison, "Method and system for providing secure software updates via satellite transmission systems," Patent WO2 006 021 829A1, Mar. 2, 2006, international Publication Number: WO2006021829A1. [Online]. Available: https://patents.google.com/patent/WO2006021829A1/en

[139] A. Mody, E. Gonzalez, and T. Underwood, "Satellite tt&c," Patent WO2 019 234 406A1, Dec. 12, 2019, international Publication Number: WO2019234406A1. [Online]. Available: https://patents.google.com/patent/WO2019234406A1/en

[140] A. Toubi and A. Hajami, "Vulnerability assessment and mitigation strategies for satellite communication systems under ddos attacks," in *2024 International Conference on Global Aeronautical Engineering and Satellite Technology (GAST)*. IEEE, 2024, pp. 1–8.

[141] M. Pinchas, E. Grunberg, and R. Hillel, "Satellite redundancy for critical applications," Aug. 20 2009, uS Patent App. 12/388,148.

[142] C. Poivey, "Radiation effects on space electronics," https://ntrs.nasa.gov/api/citations/20170002014/downloads/20170002014.pdf, NASA, Tech. Rep., 2017, accessed: 2025-06-18.

[143] J. W. Cutler and J. Beningo, "Watchdog-based fault recovery on nanosatellites," *Journal of Aerospace Information Systems*, vol. 19, no. 8, pp. 522–529, 2022.

[144] L. Dong, "Design of a five-level protection system based on watch-dog for sepacecraft computer," 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:111767946

[145] A. M. El-Attar and G. Fahmy, "An improved watchdog timer to enhance imaging system reliability in the presence of soft errors," in *2007 IEEE international symposium on signal processing and information technology*. IEEE, 2007, pp. 1100–1104.

[146] N. Wiatrek, K. Burnett, S.-L. Lin, S. Liu, and P. Saenz, "Advancing spacecraft security through anomaly detection," in *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*. IEEE, 2024, pp. 560–565.

[147] M. Rahmatian, H. Kooti, I. G. Harris, and E. Bozorgzadeh, "Adaptable intrusion detection using partial runtime reconfiguration," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 2012, pp. 147–152.

[148] S. Lu and R. Lysecky, "Analysis of control flow events for timing-based runtime anomaly detection," in *Proceedings of the WESS'15: Workshop on Embedded Systems Security*, 2015, pp. 1–8.

[149] M. F. B. Abbas, A. Prakash, and T. Srikanthan, "Power profile based runtime anomaly detection," in *2017 TRON Symposium (TRONSHOW)*. IEEE, 2017, pp. 1–9.

[150] X. Bailin, Y. Shunzheng, and W. Tao, "Application layer anomaly detection based on hsmm," in *2010 International Forum on Information Technology and Applications*, vol. 2. IEEE, 2010, pp. 411–414.

[151] S. Wankhade and R. Mahajan, "Dynamic partial reconfiguration implementation of aes algorithm," *International Journal of Computer Applications*, vol. 97, no. 3, 2014.

[152] S. Burman, P. Rangababu, and K. Datta, "Development of dynamic reconfiguration implementation of aes on fpga platform," in *2017 Devices for Integrated Circuit (DevIC)*. IEEE, 2017, pp. 247–251.

[153] B. Manjith, J. Kokila, and R. Natarajan, "Adaptive dynamic partial reconfigurable security system," in *International Conference on Next Generation Computing Technologies*. Springer, 2017, pp. 430–439.

[154] F. Unterstein, T. Sel, T. Zeschg, N. Jacob, M. Tempelmeier, M. Pehl, and F. De Santis, "Secure update of fpga-based secure elements using partial reconfiguration," *Cryptology ePrint Archive*, 2020.

[155] S. Wankhade and R. Mahajan, "Performance enhancement of aes algorithm using dynamic partial reconfiguration," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, no. 4, 2014.

[156] F. Anan, M. S. Kamal, K. S. Mamun, N. Ahsan, M. T. Reza, and M. I. Hossain, "Securing federated learning: A defense mechanism against model poisoning threats," in *2024 IEEE International Conference on Computing, Applications and Systems (COMPAS)*. IEEE, 2024, pp. 1–6.

[157] T. Gu *et al.*, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv*, 2017.

[158] S. Sakib, D. Faizullin, Y. Koga, M. Uetsuhara, and S. Onishi, "In-orbit fpga reprogramming device for small satellites," *Advances in Space Research*, vol. 71, no. 11, pp. 4549–4556, 2023.

[159] H. Ren, H. Liu, Q. Xue, X. Ma, T. Jiang, and B. Sun, "On-orbit maintenance and reconfiguration of dsp based on fpga," in *Second International Symposium on Computer Technology and Information Science (ISCTIS 2022)*, vol. 12474. SPIE, 2022, pp. 89–95.

[160] A. Hanafi, M. Karim, T. Rachidi, and I. Latachi, "Fail-safe remote update method for an fpga-based on-board computer system," in *WITS 2020: Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems*. Springer, 2022, pp. 273–283.

[161] M. Cao, T. Jin, Y. Zhao, G. Wang, X. Wen, J. Xia, X. Ma, Y. Li, and W. Lu, "On-orbit update and scrubbing design of sram fpga for spacecraft," in *Journal of Physics: Conference Series*, vol. 2870, no. 1. IOP Publishing, 2024, p. 012017.

[162] Y. Zhao, L. Li, Y. Song, L. Wu, D. Luo, and Y. Wang, "Intelligent fault diagnosis and health monitoring system for on-orbit fpga critical components," in *2024 8th International Conference on Imaging, Signal Processing and Communications (ICISPC)*. IEEE, 2024, pp. 173–177.

[163] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale+ fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 3, pp. 1–31, 2020.

[164] V. B. Kumar, N. Gupta, A. Chattopadhyay, M. Kasper, C. Krauß, and R. Niederhagen, "Post-quantum secure boot," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1582–1585.

[165] National Institute of Standards and Technology (NIST), "Post-quantum cryptography standardization," https://csrc.nist.gov/projects/post-quantum-cryptography, 2023, accessed: 2025-06-18.

[166] S. Popa, A. Kazak, A. Dinu, M. Ivanovici, N. Secrieru, V. Carbune, and V. Melnic, "Architecture and design choices for an ai-enabled fpga-based cosmic radiation sensor," in *2024 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, 2024, pp. 1–4.

[167] A. M. Cabrera, Y. A. Yucesan, F. Y. Liu, W. Blokland, and J. S. Vetter, "Errant beam detection using the amd versal acap and vitis ai," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2023, pp. 1–6.

[168] G. Özdil and B. Örs, "Model-based fpga ai accelerator design using vitis ai with in-depth performance and energy efficiency analysis," in *2024 32nd Telecommunications Forum (TELFOR)*. IEEE, 2024, pp. 1–4.

[169] N. U. Pintos, H. Lacomi, and M. Lavorato, "Comparación de vitis-ai y finn para implementar redes neuronales convolucionales en fpga," *Revista Elektrón*, vol. 8, no. 2, p. 200, 2024.

[170] S. P. Singh, S. Verma, P. Pali, H. Tiwari, and S. Kanojiya, "Application of artificial intelligence (ai) to enhance satellite security," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 11, no. 4, p. 284, 2023.