Article

Not peer-reviewed version

A Scalable and Power-Aware Security Health Monitor for FPGAs Using Lightweight Sensors and ML-Based Inference

Raj Parikh * and Khushi Parikh

Posted Date: 26 April 2025

doi: 10.20944/preprints202504.2188.v1

Keywords: FPGA security; hardware Trojan detection; on-chip sensors; PUF; ring oscillator; glitch detection; machine learning; anomaly detection; One-Class SVM; Isolation Forest; security health monitoring; side-channel analysis; runtime verification; distributed monitoring architecture; secure reconfiguration; Vivado; Quartus; local monitor agents; global security manager; bitstream integrity; JTAG log monitoring

 doi:10.20944/preprints202504.2188.v1

Article

A Scalable and Power-Aware Security Health Monitor for FPGAs Using Lightweight Sensors and ML-Based Inference

Raj Parikh [1,*] and Khushi Parikh [2]

[1] Altera Corporation

[2] California State University, Northridge; Khushi.parikh.877@my.csun.edu

* Correspondence: rparikh356@gmail.com

Abstract: The flexibility and performance of Field-Programmable Gate Arrays (FPGAs) have led to their popularity in safety and mission-critical software systems in aerospace, automotive, defense, and cloud computing. However, this reconfigurability comes with distinctive security challenges, such as the threat from hardware trojans, side-channel leakage, and malicious reconfiguration. In this paper, we propose a power-aware, scalable, and comprehensive architecture for the real-time policing security posture of FPGA devices. We combine lightweight on-chip sensors (like ring oscillators, PUFs, glitch detectors, and signal probes) with machine learning-based anomaly detectors, such as One-Class SVMs, Isolation Forests, and shallow neural networks. These elements are coordinated using a hierarchical structure of Local Monitor Agents (LMAs) and a Global Security Manager (GSM), allowing for localized threat detection and response. Leveraging recent case studies and side-channel attack databases, we evaluate the viability of our architecture for computing. Our system exhibits a detection resolution as high as possible with minimum area and power overhead and supports in-field mode update, calibration, and formal verification of the monitoring logic. The proposed platform allows passive FPGAs to be reprogrammed into active and security-aware systems, achieving immune system-like capabilities against a broad range of attacks. This work leverages recent developments in hardware security monitoring, thus paving the way for new developments in federated threat learning, secure debug interface monitoring, and unified firmware–bitstream attestation.

Keywords: FPGA security; hardware Trojan detection; on-chip sensors; PUF; ring oscillator; glitch detection; machine learning; anomaly detection; One-Class SVM; Isolation Forest; security health monitoring; side-channel analysis; runtime verification; distributed monitoring architecture; secure reconfiguration; Vivado; Quartus; local monitor agents; global security manager; bitstream integrity; JTAG log monitoring

1. Introduction

Field-Programmable Gate Arrays (FPGAs) are reprogrammable integrated circuits deployed in numerous critical systems, from cloud computing accelerators to defense electronics. Their reprogrammability provides significant flexibility, but this also introduces unique security challenges. The term "security health" of an FPGA refers to the integrity of its configuration and ongoing operation—ensuring that the device remains protocol-compliant without malicious modifications, information leakage (e.g., via side-channel attacks), or tampering [1,2].

Recent incidents underscore the necessity for FPGA security health monitoring. One notable example is the Starbleed vulnerability (2020), which enabled researchers to decrypt and modify encrypted bitstreams on FPGAs, granting complete control over the device and enabling the insertion of hardware Trojans [3]. This case illustrates how the dynamic reconfigurability of FPGAs can become a security liability without continuous security-health monitoring.

 doi:10.20944/preprints202504.2188.v1

This paper explores existing mechanisms for monitoring FPGA security health and proposes architectural enhancements to improve in-situ runtime monitoring. We examine recent literature (2021–2025) and real-world case studies featuring both successful defenses and high-profile security failures [4], [5]. Building upon the Secure FPGA Health Monitor concept introduced by Baykova et al. [6], we present multiple design augmentations. These include scalable, low-power, and distributed monitoring architectures that leverage on-chip sensors such as ring oscillators, physical unclonable functions (PUFs), and glitch detectors for continuous threat detection [7,8].

We also highlight the integration of machine learning models—including One-Class SVMs, Isolation Forests, and lightweight CNNs—to facilitate anomaly detection during runtime [9]. Our proposal emphasizes in-system integration with industry-standard FPGA CAD tools (e.g., Vivado, Quartus), embedding security monitoring directly into the design workflow to reduce human error and enhance adoption [10].

2. Background and Threat Landscape for FPGAs

An FPGA consists of programmable logic blocks and configurable interconnects, which are governed by a bitstream file. Unlike ASICs, FPGAs can be reconfigured post-deployment, which widens the attack surface [1,2]. Various lifecycle stages—including IP integration, synthesis, bitstream generation, and deployment—are susceptible to threats. For example, malicious third-party IPs or tampered bitstreams may introduce hardware Trojans (HTs) [11,12].

An adversary with access to bitstream or its deployed system may attempt to reverse-engineer or alter it, circumventing bitstream encryption. The Starbleed attack is a prominent case where attackers exploited the reconfiguration pathway to bypass encryption and compromise the entire FPGA [3].

Side-channel attacks pose another severe risk. Cryptographic keys and sensitive data can leak through power consumption or electromagnetic (EM) emissions, especially in multi-tenant FPGAs used in cloud environments, where shared infrastructure enables cross-design leakage [13,14]. Similarly, fault-injection attacks, e.g., using voltage or clock glitches—can disrupt normal operation, potentially bypassing security logic [15].

These varied attack vectors emphasize that periodic configuration checks are insufficient. A comprehensive and continuous runtime monitoring approach is essential to ensure FPGA devices remain secure throughout their operational lifecycle [5,6].

2.1. Hardware Trojans and Malicious Reconfiguration

Hardware Trojans (HTs) are stealthy modifications of a circuit that remain inactive under normal operation but are triggered under rare, specific conditions to perform malicious actions, such as leaking cryptographic keys or disabling functionality [1,2]. In FPGAs, HTs can be injected by manipulating the bitstream, either during initial configuration or via partial reconfiguration in the field [3]. Due to their dormant behavior, HTs often evade traditional design-time and test-time detection techniques and may only activate at runtime, potentially long after deployment [4].

This lifecycle vulnerability has prompted the development of runtime HT detection techniques. The security community now widely agrees that post-configuration monitoring is a critical requirement [5]. When Trojans evade early testing, on-chip monitoring systems offer a last line of defense by detecting HT activity during its activation phase.

Malicious reconfiguration is no longer a theoretical concern. Beyond Starbleed [6], multiple studies have demonstrated runtime reconfiguration attacks in which adversaries insert malicious circuits into unprotected or weakly encrypted bitstreams. For instance, a study introduced a malevolent ring oscillator payload into an unencrypted Xilinx bitstream, resulting in significantly increased power consumption and eventual device failure [7]. Such cases reinforce the need for robust in-situ security monitoring frameworks to detect and neutralize threats introduced to post-deployment.

 doi:10.20944/preprints202504.2188.v1

2.2. Security Health Monitoring Concept

Analogous to a system health monitor that tracks temperature, voltage, and error signals, a security health monitor for FPGAs inspects architectural and physical signals for signs of intrusion or system compromise [8]. Potential indicators include:

- Side-channel signature deviations (e.g., power, timing, or electromagnetic emissions),
- Abnormal cross-domain transitions, and
- Sensor anomalies that suggest tampering or environmental manipulation [9].

Some of these features already exist in modern high-assurance FPGAs. Although built-in monitors provide an initial defense, they are typically coarse-grained. Consequently, recent research has focused on enhancing detection precision through:

- Custom on-chip sensor arrays, and
- Machine learning-based anomaly detectors operating in parallel with the user's design [12,13].

These innovations aim to catch subtle intrusion patterns that traditional methods might overlook. In the subsequent sections, we examine these approaches in detail and propose new architectural enhancements to advance the state-of-the-art in FPGA security health monitoring.

3. Literature Review of Security Health Monitoring in FPGAs

3.1. On-Chip Sensors for Runtime Security Monitoring

Cryptographic bitstream integrity, anti-spoofing mechanisms, and the clock tree network are critical components of FPGA security-health systems. These features are supported by a distributed network of on-chip sensors that continuously monitor physical and logical parameters for anomalies. Prior research has shown that small hardware sensors can be deployed throughout the FPGA fabric to detect Trojan activity and tampering attempts at runtime [1].

One prominent approach involves deploying ring oscillators (ROs) in a grid formation across the FPGA. Any localized change in power consumption or switching activity—such as a Trojan activating in one region—can be detected as a frequency shift in nearby ROs. Mal-Sarkar et al. and Liu et al. pioneered the use of "order-variable" ring oscillators for this purpose, demonstrating that Trojan switching causes disturbances that are distinguishable from normal process variations [2], [3]. These RO-based sensor networks incur minimal area overhead (e.g., a 3-stage RO plus a counter occupies only a few LUTs) while providing continuous side-channel surveillance of FPGA internal activity.

A second monitoring strategy leverages built-in on-chip monitors. Most commercial FPGAs include analog-digital converters for voltage and temperature monitoring.

Forte et al. (2013) proposed using existing thermal sensors for Trojan detection, observing that active Trojans generate localized heat. Their system detected abnormal thermal profiles caused by Trojan activation, offering a low-overhead detection method by reusing existing infrastructure [4].

Other sensors target fault injection attacks, such as glitch detection circuits that monitor supply voltage and clock networks for abnormal fluctuations. A voltage-droop sensor, for instance, may consist of a reference voltage source and a fast comparator that triggers an alert if the voltage drops below a certain threshold. Voltage Glitch Detector IP blocks are now commercially available and can raise alarms when adversaries attempt to under-power or overclock the FPGA [5]. These systems may use dual comparators (positive and negative voltage spike detectors) along with latch logic to store the detection signal. Additionally, clock frequency monitors can identify attempts to manipulate clock timing by injecting extra pulses or unexpected slowdowns. Academia has proposed similar lightweight mechanisms for use in commodity FPGAs, such as distributed physically unclonable function (PUF) networks that can simultaneously serve as secure key storage and tamper detection systems [6]. Variants like SRAM PUFs and ring-oscillator PUFs can generate unique fingerprints and exhibit measurable drift under invasive or environmental stress.

 doi:10.20944/preprints202504.2188.v1

3.2. Sensor Fusion

The combination of multiple signals for cross-validation improves the robustness of detection frameworks. Apostolos Fournaris et al. proposed a multi-parametric Trojan detection method that integrates power, timing, and thermal signals from distributed sensors to generate a composite health signature of the FPGA [7]. While single-parameter drift may fall within noise margins, correlated deviation across multiple parameters is a clear indication of malicious activity.

TrustGuard (2023) is a standalone monitoring framework that applies this principle by simultaneously analyzing power side-channel signals via on-chip ADCs and observing digital logic behavior. The collected data feeds into an anomaly detection engine. TrustGuard demonstrated over 90% malware detection accuracy in experimental settings by monitoring FPGA behavior across multiple runtime dimensions [8].

These results reinforce the idea that a robust on-chip sensor suits the foundation of any FPGA security health monitor. These sensors provide raw, high-fidelity data that higher-level detection systems—such as machine learning classifiers or rule-based engines—can use to infer and react to threats in real time.

4. Machine Learning for Runtime Anomaly Detection

Raw sensor data alone is insufficient for robust security health monitoring. A well-designed system must distinguish normal behavior from attack conditions, typically through a training dataset that helps define operational baselines. Machine learning (ML) has emerged as a critical tool in achieving this objective. FPGA runtime data—including power traces, timing metrics, and electromagnetic emissions—are often high-dimensional and context-dependent, which makes traditional rule-based anomaly detection ineffective. Consequently, researchers have turned to ML models that learn the normal behavior of an FPGA design and raise alerts when deviations suggest the presence of hardware Trojans or other threats [1,2]. A fundamental challenge in hardware security is the lack of labeled attack data, since it is infeasible to anticipate every possible Trojan or side-channel exploit. As a result, unsupervised learning and one-class classification methods are widely favored. One popular approach is the One-Class Support Vector Machine (OC-SVM), which only requires benign data during training and flags outliers during runtime. The SPECTRE framework, for example, leverages OC-SVMs to model the normal distribution of a known FPGA power profile. When runtime power or EM signals deviate from this learned distribution, the model flags an anomaly. This method proved more resilient to chip-to-chip process variation than template-based techniques and eliminated the need for a "golden" reference chip [3,4]. OC-SVM also adapts well to device-specific characteristics, making it ideal for field deployments where each FPGA may behave slightly differently due to manufacturing variation.

4.1. Isolation Forest

Another powerful unsupervised approach is the Isolation Forest, a tree-based ensemble model designed for high-dimensional anomaly detection. It operates by randomly partitioning the dataset and isolating data points that require fewer splits—presumably anomalies [5]. In hardware security, Gurram et al. demonstrated the efficacy of Isolation Forests in monitoring side-channel data from FPGA designs, detecting behavioral anomalies linked to Trojan activation. Due to its randomized structure, this method is robust to novel attacks and yields an anomaly score for each observation. Chen et al. observed that, while threshold-based systems often miss subtle or partial Trojan activity, their Isolation Forest implementation successfully identified insertions while adapting to workload variations, reducing false positives [6]. Techniques like Local Outlier Factor (LOF) and clustering are also emerging in this space, but Isolation Forests stand out for their scalability and interpretability in unlabeled environments.

 doi:10.20944/preprints202504.2188.v1

4.2. Deep Learning Models

Where labeled attack data is available—or when finer feature extraction is needed, deep learning techniques have shown promising results. Models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can learn complex patterns from side-channel traces or bitstream data. For example, researchers have trained LSTM (Long Short-Term Memory) networks on sequential bitstream data to detect malicious reconfiguration. By modeling the "grammar" of a valid bitstream, the network could identify anomalies such as Trojan insertions. One study showed that an LSTM achieved a 93.5% detection rate, outperforming standard RNNs by 5% [7]. Further, Nasr et al. (2024) introduced a Siamese Neural Network model for Trojan detection using side-channel traces. The Siamese network learned to maximize the difference between Trojan-free and Trojan-infected signals without requiring detailed labels for every attack variant. Their Siamese LSTM architecture achieved 86.78% accuracy, outperforming CNN and GRU baselines [8].

4.3. Lightweight CNNs and TinyML

Lightweight architectures are particularly attractive for resource-constrained environments like IoT. A 2021 study by Hu et al. deployed a shallow 3-layer CNN on the FPGA fabric to classify power transient patterns in near-real-time. By leveraging DSP slices, the CNN operated entirely on-chip, enabling runtime anomaly detection without an external CPU [9]. This introduces the concept of an on-chip AI agent for FPGA security. However, deep models must be pruned or quantified to meet the power and area constraints of FPGA fabrics. Research in TinyML for FPGA security is actively exploring model compression strategies. For example, combining an Isolation Forest for coarse-grained anomaly detection with a miniature CNN for classification yields a favorable accuracy–overhead trade-off [10].

In short, ML-driven runtime anomaly detection has become a cornerstone of modern FPGA security health systems. OC-SVM and Isolation Forests provide robustness against novel attacks through unsupervised learning, while deep neural models (CNNs, LSTMs, Siamese networks) exploit fine-grained side-channel or bitstream patterns for precise detection. The most recent academic literature (2021–2024) suggests a hybrid approach, where unsupervised methods flag suspicious patterns that are subsequently analyzed by neural networks—in real time, during normal FPGA operation.

The key challenges moving forward include securing ML models against adversarial perturbations and ensuring that anomaly detectors are lightweight enough for real-world FPGA deployment. The next section discusses architectural frameworks that fuse sensors and ML models, addressing power, scalability, and integration challenges.

5. Architecture and Case Studies of Existing Security Health Monitor

Several architectural frameworks have been introduced in the literature to monitor the security health of FPGA systems. Below, we highlight key examples that represent centralized, distributed, and hybrid approaches to runtime monitoring.

5.1. SecureWorks Monitor (Hypothetical Prior Art)

The SecureFPGA Health Monitor is a conceptual architecture based on a centralized security controller implemented as either a soft-core processor or finite state machine. This controller aggregates data from all on-chip sensors—such as RO frequencies and voltage monitors—and compares them against predefined reference values. A typical implementation might use a MicroBlaze soft processor, executing lightweight security firmware that continuously polls for anomalies such as unauthorized reconfiguration events or missing configuration checksums. These may be accessed through the Internal Configuration Access Port (ICAP) for dynamic bitstream readback [1]. While this centralized model is feasible for small- to medium-scale FPGAs, it poses

 doi:10.20944/preprints202504.2188.v1

scalability limitations. In larger devices, localized events may be missed or delayed due to polling latencies, making the design less responsive under fast-evolving attacks.

5.2. TrustGuard (2023)

As discussed earlier, TrustGuard is a notable example of a real-world FPGA-based security co-processor that performs side-channel monitoring using on-chip ADCs or built-in sensors connected to power rails [TrustGuard-2023]. It operates adjacent to the main logic fabric—either on-chip or in a companion FPGA—and employs a machine learning engine to detect malicious activity in real time. One of TrustGuard's key innovations is its independence from external processors: all detection and decision-making occur within the FPGA. In one case study, TrustGuard successfully identified a stealthy hardware Trojan attempting to leak data through fine-grained power fluctuations. These anomalies were detected early enough to halt data exfiltration, validating the effectiveness of high sampling rates and FPGA-optimized ML classifiers (e.g., lightweight decision trees or small neural networks).

5.3. Drone during Runtime (2022) – Distributed Monitors

In 2022, Kanu et al. introduced a distributed security monitoring framework, codenamed *Drone*, that deploys multiple mini-monitors ("drones") across the FPGA fabric [Kanu-2022]. Each drone is a small state machine that observes local activity—such as signal transitions or protocol adherence—at the interface of potentially untrusted third-party IP blocks. These mini monitors report back to a centralized analyzer, which raises alarms if inconsistencies are detected. A case study involving a RISC-V soft processor with an embedded Trojan demonstrated this method's effectiveness. The Trojan was designed to leak a private key following a specific trigger pattern. However, the surrounding monitor shell detected the anomalous sequence and raised an alert, allowing the system to reset before the data leakage completed. This approach illustrates a hybrid strategy combining functional correctness checks with security analysis. The trade-off is the resource overhead associated with instrumenting each IP core, but it provides scalable protection for complex designs.

6. Proposed Security Health Monitoring Architectures

To address growing runtime threats and system complexity in modern FPGAs, we propose architectural improvements that extend the SecureFPGA Health Monitor model. These enhancements prioritize scalability, low power, intelligent threat detection, seamless toolchain integration, and robust response capability. Below is a detailed overview of the five major architectural innovations.

6.1. Hierarchical Distributed Monitor Network

The first proposed enhancement is a hierarchical architecture comprising multiple **Local Monitor Agents (LMAs)** distributed across the FPGA fabric, each reporting to a **Global Security Manager (GSM)**. LMAs are lightweight modules deployed in different regions of the chip, such as per clock region or across several logic columns. Each LMA is equipped with configurable sensors, including ring oscillators for dynamic power detection, signal probes, performance counters, or simple CRC checks on configuration frames. They also contain local anomaly detection logic, which could be implemented using basic counters and comparators or more advanced one-class classifiers.

Each LMA functions independently to detect anomalies in its region, such as sudden spikes in switching activity or configuration CRC failures. These agents communicate with the GSM via a shared on-chip bus or security-dedicated interconnect. The GSM, implemented as a soft-core processor or a dedicated controller, aggregates reports and performs global analysis to coordinate appropriate system-level responses. The distributed architecture allows localized events to be identified by nearby LMAs with minimal latency, reducing reliance on centralized polling. For example, if a Trojan activates in the DSP region, the nearest LMA can detect abnormal signal activity

 doi:10.20944/preprints202504.2188.v1

or local power spikes and raise an alert, while LMAs in unaffected regions remain idle—thus reducing false positives.

LMAs are designed to be passive most of the time, consuming minimal power and only waking periodically or upon trigger conditions. Their simplicity ensures they can be duplicated across the FPGA fabric without excessive resource overhead, with each LMA estimated to occupy approximately 50 LUTs and a few flip-flops. A hierarchical structure enables the architecture to scale with FPGA size, and the repeated instantiation of LMAs improves design reuse and HDL modularity. This monitoring strategy parallels the concept of thermal sensor grids used in CPUs, offering a spatially aware and localized security mechanism within the FPGA.

6.2. Integrating Low-Power Sensors and Isolation of Power Domains Better

To reduce power overhead, the architecture proposes reusing existing FPGA infrastructure rather than deploying redundant sensing logic. For example, instead of continually hashing entire configuration frames for integrity verification, the internal configuration access port (ICAP) can be used to read small, random samples of configuration bits at intervals. This technique, referred to as **integrity sampling**, consumes minimal bandwidth and power while maintaining high confidence in bitstream integrity. Additionally, the design reuses SYSMON ADCs already present in many FPGA families for monitoring temperature and voltage. Custom digital logic is added to flag transient behaviors that may indicate tampering. Wherever possible, the security infrastructure is placed on separate power rails. High-assurance devices such as those from Microsemi/Microchip implement similar techniques by placing their secure subsystems in always-on voltage domains. Following that model, the proposed architecture places LMAs in the main core voltage domain—ensuring accurate sensing of localized effects—while the GSM and critical logic, such as model parameters and reference memory, are located in isolated auxiliary domains. These domains remain functional even during core-level voltage manipulation attacks. LMAs may be constructed using asynchronous logic or threshold-based comparators that only activate on meaningful events, further improving energy efficiency. This structure makes use of already available power isolation features and significantly improves fault tolerance without requiring substantial redesign of the base FPGA fabric.

6.3. Security Co-Processor (Machine Learning Accelerator for Security)

The third enhancement introduces a dedicated security co-processor, embedded within the Global Security Manager, to support real-time intelligent anomaly detection. With the increasing complexity of side-channel signals, power traces, and configuration patterns, machine learning (ML) models are well-suited to identify threats based on deviations from learned normal behavior. The proposed ML engine executes models such as one-class SVMs, principal component analysis, or lightweight CNNs, with support for high-level synthesis (HLS) implementation. This engine receives summarized feature inputs from the LMAs—such as frequency deviations, power surge magnitude, or configuration mismatches—and outputs either anomaly scores or categorized alerts. Because the co-processor is integrated directly into the fabric, it can analyze streaming data at line speed. The engine is programmable and reconfigurable, allowing for model updates without hardware redesign. For instance, if a new side-channel attack is discovered, a retrained model can be uploaded to the co-processor while reusing the same sensor network. The ML engine can operate in an event-driven or semi-scheduled manner. It only activates when sufficient data has accumulated or when LMAs raise a suspicious flag. This hybrid detection architecture—fast thresholds in LMAs combined with deep analysis in the GSM—balances power, latency, and detection accuracy. With modern FPGAs featuring hardened DSPs and AI processing units (e.g., Versal AI Engine), such machine learning functions can be mapped efficiently with minimal impact on the main application logic.

 doi:10.20944/preprints202504.2188.v1

6.4. FPGA Toolchain Integration and Design Flow

For widespread adoption, the proposed architecture integrates into the standard FPGA design and implementation workflow through a tool called the SecMon Integrator. This tool allows users to specify high-level security constraints such as critical regions, power domains, and IP modules that require monitoring. It automatically inserts parameterized LMAs and GSMs into the design netlist and places them within reserved fabric locations during layout. The tool instruments selected signals—such as data paths, control signals, or handshake flags—by tapping into them for anomaly detection logic. It can also annotate and verify security behaviors using property specification languages such as PSL or SVA. For example, the user may define properties like: "if config_interface.busy flips unexpectedly, the security system must raise an alert within 10 cycles."At the bitstream stage, the tool can inject cryptographic hashes or HMAC keys into secure memory blocks. This enables runtime attestation, allowing the GSM to verify configuration integrity using ICAP without requiring off-chip support. On Intel FPGAs, secure NVM can be used to store golden hashes; in Xilinx FPGAs, SYSMON outputs can be routed to the ML engine for real-time interpretation. Verification tools accompanying the integrator can simulate security threats like voltage glitches or bitstream corruption and validate appropriate responses. These simulation-based testbenches are conceptually similar to ECC memory testing, which involves controlled fault injection and detection validation. By embedding monitoring logic directly into the CAD flow, this architecture ensures that runtime security becomes a fundamental part of the FPGA lifecycle—from design through deployment. An AI-enhanced approach to real-time security monitoring proposes integrating JTAG access logs as a feature source for anomaly detection models deployed on the FPGA itself [22]. By streaming JTAG interaction sequences to a lightweight on-chip LSTM engine, it becomes feasible to recognize suspicious sequences that deviate from normal configuration/debug workflows—thereby extending the scope of runtime health monitoring beyond physical signals to include behavioral log semantics [22].

6.5. Decentralized Response and Recovery Systems

Finally, the architecture includes a programmable, tiered response system that reacts to detected anomalies based on severity. While some events may only require logging or alerting an external controller, others may necessitate partial reconfiguration, regional clock gating, or even complete I/O isolation. This system transforms the monitor from a passive diagnostic tool into an active defender. If an LMA detects abnormal power consumption—potentially caused by a crypto circuit under a Trojan's control—it can immediately halt that region's operation by gating the local clock. If the GSM determines that the anomaly is critical, it can trigger a full-chip reset or reconfigure the device to a safe state using ICAP. Sensitive data, such as decryption keys stored in battery-backed memory, can be erased as a precaution. These capabilities mimic tamper-response systems found in secure microcontrollers and smart cards, where keys are wiped upon physical intrusion. The monitor network interfaces with existing FPGA primitives to enable this functionality. It can drive ICAP for configuration reloads, control MMCM or PLLs for clock adjustments, and manage I/O buffers to tri-state external connections. A response library defines these behaviors, and simulation infrastructure verifies the system's performance under different anomaly scenarios. Thresholds for ML-based detection are configurable to minimize false positives while preserving sensitivity to genuine threats. Including an active response mechanism ensures that threats are not only detected but also mitigated in real time. This shifts the role of the security monitor from observation to dynamic defense, enabling the FPGA to maintain operational integrity or safely degrade functionality depending on the criticality of the attack.

7. On-Chip Sensor Types and Deployment

The effectiveness of a security health-monitoring framework hinges on the strategic deployment of diverse on-chip sensors. Our proposed design incorporates a variety of sensors tailored to detect

 doi:10.20944/preprints202504.2188.v1

anomalies in power consumption, timing, and functional behavior, serving as the sensory backbone for the Local Monitor Agents (LMAs).

A fundamental sensor integrated into each LMA is the ring oscillator (RO)-based frequency sensor. The sensitivity of this sensor can be tuned by adjusting the number of inverter stages in the RO. Longer ROs are more sensitive to subtle voltage fluctuations but exhibit slower response times, whereas shorter ROs respond more quickly but are more susceptible to high-frequency noise. On FPGAs, ROs can be implemented by looping inverter-based LUTs, with placement constraints to prevent synthesis tools from optimizing them away. Each LMA includes a counter that tracks RO oscillation cycles within a defined time window. This count is compared to a reference value established during the initial calibration phase when the FPGA is operating in a known-good state. A deviation outside predefined temperature-compensated boundaries triggers an anomaly flag. Despite its simplicity, this method has been shown to detect power fluctuations of only a few milliwatts, often associated with Trojan activation. Complementing ROs, our design employs Physically Unclonable Functions (PUFs), particularly SRAM PUFs implemented within unused block RAMs. These modules are initialized at power-up and provide a unique fingerprint for each FPGA instance. During the secure boot or calibration phase, each PUF's initial response is stored in the Global Security Manager as a reference baseline. At runtime, the PUF can be re-evaluated periodically to detect changes that may indicate environmental degradation, aging, radiation exposure, or invasive tampering, such as a focused ion beam attack. While PUF responses can vary slightly with temperature, cross-referencing them with real-time thermal data ensures that anomalies are attributed to genuine threats rather than benign operating conditions. Though not suited for direct Trojan detection, PUFs enhance overall platform integrity by identifying persistent changes in the device's physical state. Recent surveys emphasize that while PUFs were initially proposed for cryptographic key generation, their evolving role in FPGA security extends into tamper detection and aging-induced degradation checks [21]. By observing entropy drift in SRAM-based PUFs during re-evaluation cycles, developers can detect not only process anomalies but also focused ion beam or environmental stress attacks, offering a non-invasive method to validate FPGA integrity post-deployment [21].

The architecture also includes glitch detectors positioned at strategic supply and clock points. These detectors are realized in two forms: analog comparators where available and high-speed digital samplers using carry-chain logic to capture ultra-short transients. For example, voltage glitch sensors monitor core and auxiliary rails, while clock glitch detectors validate external clock inputs. In FPGAs without analog comparator access, high-frequency sampling clocks derived from PLLs can be used to detect glitches by re-sampling the signal into itself.

In addition to physical sensors, internal signal probes are synthesized into the user design via toolchain automation. These include checkers for specific functional behaviors that should not occur during normal operation. For instance, a FIFO that should never overflow under standard workloads can be monitored for overflow events that might indicate denial-of-service conditions or internal faults. Likewise, heartbeat signals—regular toggles from user logic that confirm liveness—are monitored by nearby LMAs. If the heartbeat fails to pulse within an expected interval, this may indicate a stalled or compromised design region, possibly due to deadlock or a Trojan-induced fault. These functional monitors are optional but add significant value, especially in high-assurance deployments.

8. Machine Learning Model Deployment on FPGA

We have explored a hardware-optimized implementation of One-Class SVMs as the primary anomaly detection mechanism. These models are typically trained offline using simulation traces or post-silicon calibration and deployed by embedding support vectors and thresholds into on-chip memory blocks such as BRAM or distributed LUT RAM. For example, applying a linear or polynomial kernel to 8 support vectors in $\mathbb{R}^{10}$, where each feature vector includes metrics like maximum RO frequency deviation, average dynamic power, or configuration error counts, allows

 doi:10.20944/preprints202504.2188.v1

inference with minimal arithmetic operations. Using parallel DSP multipliers, dot products are efficiently computed and aggregated to form a novelty score. This score, compared against a stored threshold, provides real-time classification within a few microseconds—well within the window of typical Trojan activation or side-channel exploitation timescales.

Implementing Isolation Forests directly in hardware is more challenging due to their tree-traversal nature, though ensemble SVMs may approximate their behavior while maintaining simplicity. Alternatively, we could deploy a compact 3-layer MLP (multi-layer perceptron), which—when unrolled—can complete inference within a few clock cycles on mid-range FPGAs. While neural networks offer the advantage of learning complex, nonlinear relationships from sensor data, they are often considered less interpretable compared to traditional models such as SVMs or decision trees. To bridge this, an unsupervised autoencoder may serve as a model-agnostic anomaly detector using reconstruction loss as the decision metric.

A critical design consideration is model adaptability in the presence of concept drift—natural changes in workload or aging effects—which may require in-field model updates. Our architecture supports such adaptation via reconfiguration or memory-mapped updates to BRAM. Initial deployment may use conservative thresholds, gradually tuned based on operational data. A calibration mode supports statistics gathering, pre-deployment training, and parameter updates through embedded CPUs or host-controlled communication protocols. This form of runtime model refinement is highlighted as essential in recent literature on side-channel-aware monitoring in secure embedded systems [12].

9. Conclusions and Future Directions

In this work, we present an update on Security Health in FPGA Devices based on recent literature and actual cases to reflect on how FPGAs can be protected during operation. We described using on-chip sensors (e.g. ring oscillators, PUFs, glitch detectors, etc.) and machine learning-based anomaly detectors (e.g., One-Class SVM, Isolation Forest, CNN/LSTM models) to enable runtime hardware Trojan detection and tamper monitoring. Based on these observations, we suggested architecture augmentations for FPGA security health monitoring. Key contributions include a means for scalability as a distributed network of monitor agents, low-power requalification of existing FPGA sensors, segmented on-chip ML co-processors for real-time anomaly detection, and a close coupling with FPGA design tool flows to enable automation of security health features. Our proposed architecture is architecture-friendly for FPGA families with reasonable overhead suitable for practical deployment in high-security applications. We contended that such a system could immensely bolster the resilience of FPGAs to many threats — from stealthy hardware Trojans to active fault attacks — through on-the-fly "health check-ups" and fast protective responses akin to an immune system.

Several future directions arise looking ahead. One is tuning ML models for increasingly high levels of confidence — for example, federating across multiple FPGAs deployed in the field to learn together what attack vectors look like without revealing any one device's private data. Another avenue is formal verification of the security monitor itself, ensuring that it can't be circumvented or disabled by an attacker's design. This may include using FPGA isolation primitives (so that user logic cannot route to the monitor logic or control it) and proving security properties about the monitor's state machine. Moreover, if FPGAs are becoming elements of heterogeneous systems (for example, SoC FPGAs with hard processors), a powerful unified security health monitor encompassing integrated processors as well as FPGA fabric circuitry would be desirable — firmware attestation with FPGA bitstream attestation.

Remember that maintaining the security \ health of FPGA devices is an evolving challenge that requires a multi-faceted approach. By combining circuit-level sensors, intelligent algorithms, and solid architectural design, we can work towards secure FPGAs upon initial deployment and remain safe for their entire operational lifecycle. A foundation towards achieving this is a comprehensive review and proposal.

 doi:10.20944/preprints202504.2188.v1

References

1. Zhang, T., Tehranipoor, M., & Farahmandi, F. (2023). *TrustGuard: Standalone FPGA-Based Security Monitoring Through Power Side-Channel*. IEEE TVLSI. (Detected >90% of malicious circuits via power/ML analysis). https://doi.org/10.1109/TVLSI.2023.3335876
2. Dofe, J., Danesh, W., More, V., & Chaudhari, A. (2023). *Natural Language Processing for Hardware Security: Case of Hardware Trojan Detection in FPGAs*. *Electronics*, 8(3), 36. (Used RNN/LSTM to detect Trojans in FPGA bitstreams with 93.5% accuracy). https://doi.org/10.3390/cryptography8030036
3. Jiang, Z., & Ding, Q. (2024). *A framework for hardware trojan detection based on contrastive learning*. *Scientific Reports*, 14, 30847. (Contrastive learning on power traces for unsupervised HT detection). https://doi.org/10.1038/s41598-024-81473-0
4. Nasr, A., Mohamed, K., Elshenawy, A., & Zaki, M. (2024). *A Siamese deep learning framework for efficient hardware Trojan detection using power side-channel data*. *Scientific Reports*, 14, 13013. (Siamese NN – LSTM achieved ~86.8% HT detection accuracy, outperforming CNN/GRU). https://doi.org/10.1038/s41598-024-62744-2
5. Jap, D., He, W., & Bhasin, S. (2017). *Supervised and Unsupervised Machine Learning for Side-Channel based Trojan Detection*. (One-Class SVM based approach robust against variations; demonstrated on AES core in Xilinx FPGA). https://doi.org/10.1109/ASAP.2016.7897095
6. Hu, T., Wu, L., Zhang, X., & Liao, Z. (2023). *Hardware Trojan Detection Combines with Machine Learning: an Isolation Forest-based Detection Method*. (Proposed isolation forest on power side-channel; unsupervised detection of unknown Trojans). https://doi.org/10.1109/BigDataSE50710.2020.00021
7. Forte, D., Bao, C., & Srivastava, A. (2013). *Temperature Tracking: An Innovative Run-Time Approach for Hardware Trojan Detection*. *ICCAD 2013*. (Used on-chip thermal sensors to detect power/thermal deviations from Trojan activation). https://doi.org/10.1109/ICCAD.2013.6691170
8. Intel Corporation. (2021). *Physical Anti-Tamper Monitoring – Intel® FPGA Security Overview*. (Stratix 10/Agilex SDM monitors configuration clock, power rails, temperature; can notify logic of tamper events). https://doi.org/10.1109/HOST53854.2022.9839411
9. Microchip Technology. (2022). *PolarFire FPGA Security User Guide*. (PolarFire includes voltage, frequency, temperature monitors; tamper event triggers fabric signals and allows user-defined responses) (PolarFire Family FPGA Security User Guide) (PolarFire Family FPGA Security User Guide)
10. Ruhr Univ. Bochum & MPI. (2020). *Critical "Starbleed" vulnerability in FPGA chips identified* (Press Release). (Describes Starbleed: attackers can decrypt bitstream, gain full control, insert Trojans; unpatchable in hardware) (Critical "Starbleed" vulnerability in FPGA chips identified | Newsportal - Ruhr-Universität Bochum) (Critical "Starbleed" vulnerability in FPGA chips identified | Newsportal - Ruhr-Universität Bochum)
11. Mal-Sarkar, S., Karam, R., & Narasimhan, S. (2016). *Design and validation for FPGA trust under hardware Trojan attacks*. *IEEE Trans. Multi-Scale Computing Systems, 2*(3), 186–198. (Early work proposing ring oscillators for Trojan detection in FPGAs) (A framework for hardware trojan detection based on contrastive learning | Scientific Reports)
12. Kanu, S. et al. (2022). *Run-time Hardware Trojan Detection and Recovery for Third-Party IPs in FPGAs*. *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*. (Inserts monitoring "shells" around IP cores; demonstrates detection of Trojans in runtime on FPGA)
13. Azure Team – Microsoft. (2019). *Securing Cloud FPGAs in Multi-tenant Environment* (Blog/whitepaper). (Discusses need for FPGA attestation, monitoring in Azure; outlines using secure enclaves for FPGA

 doi:10.20944/preprints202504.2188.v1

bitstreams, and detecting abnormal behavior in network traffic). (A framework for hardware trojan detection based on contrastive learning | Scientific Reports)

14. Xilinx (AMD). (2021). *Tamper Monitoring and Response in Versal ACAP TRM (AM011)*. (Versal devices detect voltage/temperature out-of-range, debug port activity, etc., and can trigger user-defined tamper responses) (Tamper Monitoring and Response - AM011) (Tamper Monitoring and Response - AM011)
15. Gurram, P. et al. (2021). *fSEAD: a Composable FPGA-based Streaming Ensemble Anomaly Detector*. *ACM/SIGDA FPGA'21*. (Implements ensemble of anomaly detection algorithms on FPGA for high-speed data streams; relevant to security monitoring concept) (fSEAD: a Composable FPGA-based Streaming Ensemble Anomaly ...)
16. Zhao, Y., et al. (2018). *FPGA-Based Remote Power Side-Channel Attacks*. *IEEE Symposium on Security and Privacy*. (Showed how one user's FPGA design can remotely sense another's power usage to extract cryptographic keys, underscoring need for isolation and monitoring in cloud FPGAs).
17. Ender, M., et al. (2020). *The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series FPGAs*. *USENIX Security 2020*. (Technical paper corresponding to Starbleed, detailing the attack exploiting WBSTAR register to read out bitstream)
18. Xilinx (AMD). (2020). *Design Advisory: Defeating Bitstream Encryption in 7-Series and Virtex-6 FPGAs*. (Xilinx's guidelines after Starbleed, recommending physical security and key management changes).
19. Shende, S. et al. (2016). *Hardware Trojan detection using side-channel power analysis with PCA and LDA*. (Demonstrated detecting Trojans via dimensionality reduction of power trace data; early ML approach on FPGA) (A framework for hardware trojan detection based on contrastive learning | Scientific Reports) (A framework for hardware trojan detection based on contrastive learning | Scientific Reports)
20. Lattice Semiconductor. (2021). *Protecting FPGAs in IoT Applications*. (Notes lightweight security features in small FPGAs and importance of active monitoring for IoT) – (Provides industry perspective on balancing security and resources in tiny FPGAs).
21. Parikh, R., & Parikh, K. (2025). Survey on Hardware Security: PUFs, Trojans, and Side-Channel Attacks. DOI:10.20944/preprints202501.1559.v1
22. Parikh, R., & Parikh, K. (2025). AI-Driven JTAG Log Monitoring System for FPGA. https://doi.org/10.20944/preprints202503.1144.v