A Lifecycle-Based Security Threat Model for FPGA in Safety-Critical Systems

Dongmin Kim[1], Sooyon Seo[1], Moohong Min[2], Aram Kim[3*]

[1]Department of Immersive Media Engineering, Sungkyunkwan University, Seoul 03063, Republic of Korea

[2]Department of Computer Education/Social Innovation Convergence Program, Sungkyunkwan University, Seoul 03063, Republic of Korea

[3]Department of Information Security, University of Suwon, Hwaseong-si 18323, Republic of Korea

Received: November 10, 2025; Revised: January 15, 2026; Accepted: February 18, 2026; Published: March 01, 2026

Abstract

Field-programmable gate arrays are increasingly adopted in safety-critical systems due to their deterministic execution, low latency, and suitability for rigorous verification and validation. However, prior studies largely focus on individual attack techniques or the operational phase, limiting their ability to capture how security threats are introduced, propagate, and remain dormant across the FPGA development lifecycle. To address this limitation, this paper proposes a lifecycle-based security threat analysis framework that integrates the IEEE Std 1012 verification and validation lifecycle with the FPGA development flow. The proposed framework supports systematic analysis by anchoring the assessment to key development artifacts spanning from design through bitstream generation and operation. By structuring FPGA security threats from a lifecycle perspective, this study provides a foundational analytical framework for systematically analyzing security risks across the FPGA development lifecycle in safety-critical systems.

Keywords: FPGA security, Safety-critical systems, Threat modeling, Hardware Trojans, Side-channel attacks, Lifecycle-based analysis.

1 Introduction

Field-programmable gate arrays (FPGAs) are reconfigurable digital devices that allow user-designed circuits to be implemented directly in hardware, providing low latency based on parallel processing and guaranteeing deterministic operation in which the relationship between inputs and outputs does not change. In addition, because the design methodology is not tied to a specific hardware circuit, FPGAs offer a high level of resilience against obsolescence.

Owing to these technical characteristics, FPGAs have been widely adopted in industrial control systems (ICS) where availability is prioritized and high reliability is required, such as in the automotive, aerospace, and defense sectors. Industrial control systems refer broadly to systems that control industrial processes; among them, systems whose failure can cause not only economic loss, but also safety-related accidents are referred to as safety-critical systems (SCS). In recent years, FPGAs have been actively discussed as a key device to replace programmable logic controllers (PLCs) traditionally used in microprocessor-based systems for such safety-

critical applications.

In safety-critical systems such as nuclear power plants (NPPs), aerospace systems, and automotive applications including autonomous driving systems, ensuring reliable operation that behaves exactly as designed while processing data rapidly is a core design requirement [9]. In this respect, FPGA-based systems, unlike PLCs, do not rely on software such as operating systems (OS), thereby reducing the likelihood of unintended behaviors beyond the original design and facilitating verification and validation (V&V) in ICS environments. In addition, due to their design flexibility, FPGAs are effective for implementing redundancy and mitigating common cause failures (CCF), thereby contributing to system safety and attracting attention as a semiconductor device capable of replacing PLCs. In next-generation nuclear facilities such as small modular reactors (SMRs), the adoption of FPGA-based safety systems is being considered to overcome the limitations of existing PLC-based safety systems. This trend can be observed through topical reports published by the U.S. Nuclear Regulatory Commission (NRC), which examine the applicability of various FPGA-based safety systems, including RadICS [24] and HFC-FPGA [24]. A list of FPGA-based safety systems discussed in relevant topical reports is summarized in Table 1.

Table 1. FPGA-Based Safety Systems published by NRC

Topical Report	FPGA-based Safety System	System Developer	Year
[25]	RadICS	Radity	2020
[24]	HFC-FPGA	Doosan	2018
[23]	Advanced Logic System (ALS)	Westinghouse	2013
[26]	Highly Integrated Protection System (HIPS)	Paragon Energy Solutions	2017

However, the replacement of PLC-based systems with FPGA-based systems does not guarantee inherent safety. FPGAs still present security risks when deployed in safety-critical systems. Until a bitstream—the file used to configure an empty FPGA with a user's design—is generated, the FPGA development process closely resembles a software development process. After the bitstream is loaded, however, the design becomes fixed and exhibits strong hardware characteristics. Owing to this dual nature, new vulnerabilities that do not exist in PLC-based systems may arise. Before bitstream generation, attacks targeting the electronic design automation (EDA) toolchain that supports the overall FPGA development process frequently occur [21], which may lead to the insertion of hardware trojans [15] containing trigger circuits that activate under specific conditions, or to side-channel attacks [12] that extract cryptographic keys by analyzing unintended physical signals of the FPGA. These threats may be activated during the development phase, but if they remain hidden until system operation begins, they can cause severe defects in actual safety-critical systems. Furthermore, because some side-channel attacks can be carried out by accessing physical signals from systems in operation, it can be concluded that multiple attack surfaces exist throughout the entire FPGA lifecycle.

As a result of these issues, a wide range of studies on attacks and countermeasures related to FPGA security have been conducted. However, most existing studies focus on analyzing individual attack techniques, making it difficult to apply them in the context of the overall FPGA development process and V&V activities encountered in real-world deployments. In safety-critical systems, clear definitions of artifacts generated at each development phase and corresponding V&V procedures are required; however, prior research centered on individual attack or defense techniques is difficult to directly associate with such artifacts and V&V processes, limiting its practical usefulness in industrial environments. Although some studies have examined the applicability of FPGAs to safety-critical systems, these works primarily adopt a reliability-focused approach, such as redundancy-based design or mitigation of internal signal instability caused by external factors like single event upset. While effective during the operational phase where accidental failures are likely, such approaches are insufficient to account for intentional attacks that may target FPGAs across the entire development lifecycle.

Therefore, this study develops a framework to support vulnerability analysis across FPGA development phases with the aim of improving practical applicability in real-world environments. To this end, the software V&V processes defined in the IEEE 1012 standard [13] are aligned with the FPGA development phases recommended in the Vivado Design Suite User Guide: Design Flows Overview (UG892) [3]. Based on this alignment, this study proposes a lifecycle-based framework that supports systematic security analysis by relating key development artifacts to corresponding phases of the FPGA development lifecycle, while providing a structured basis for examining V&V at each phase. By structuring security considerations around these artifacts, the proposed framework provides a methodological foundation that can be applied to safety-critical systems and readily integrated with existing V&V processes.

2 Background

2.1. FPGA-Based Safety-Critical Systems

In safety-critical systems, failures of individual components can lead to loss of life and operational shutdowns; therefore, the reliability of control logic is of paramount importance. Accordingly, each component constituting an SCS must undergo rigorous verification and validation processes throughout the entire lifecycle, from the initial design phase to operation, and any modification or functional extension of components is subject to strict control.

To satisfy such stringent verification requirements while maintaining design efficiency, the use of FPGAs has been increasing. Owing to their parallel processing architecture, FPGAs provide fast response times and deterministic execution characteristics, and their operation without reliance on an operating system (OS) reduces uncertainties that may arise from complex software layers. In addition, fault-tolerance techniques such as redundancy-based designs, including triple modular redundancy (TMR) [20], and scrubbing techniques [18] that recover errors in configuration memory can be directly incorporated at the circuit configuration phase, enabling high levels of fault tolerance. Furthermore, once design architecture and intellectual property (IP) have been verified, they can be reused in subsequent designs, which is considered a significant advantage of FPGAs in long-term operation scenarios where obsolescence is a concern.

Although FPGAs ultimately operate as hardware, their design and implementation processes exhibit characteristics similar to those of software development. As a result, errors or defects that may arise during the design phase and those occurring during operation require different mitigation approaches. For this reason, the safety-critical systems domain applies lifecycle-based verification frameworks, originally developed for software, to FPGA development. Among these, IEEE Std 1012 is widely used as a reference standard for establishing V&V processes across the entire development lifecycle.

IEEE Std 1012 is an international standard that systematically defines the objectives and deliverables of verification and validation activities to be performed at each phase of the software and system development lifecycle. By distinguishing verification targets and responsibilities across the requirements, design, implementation, integration, verification, and operation phases, the standard presents a lifecycle-based V&V approach to ensure that development artifacts satisfy their intended purposes at each phase. NRC recommends IEEE Std 1012 as a verification standard for safety-grade digital systems through Regulatory Guide (RG) 1.168 [22], and prior studies on FPGA-based safety systems and V&V design [1] have also reported cases in which this standard is referenced. These cases indicate that, although FPGAs are implemented as hardware platforms, software-oriented V&V approaches can be directly applied due to the nature of their development artifacts.

In this context, to reflect the lifecycle-based verification flow required in safety-critical systems, this study adopts IEEE Std 1012 as a primary reference when designing a security threat modeling approach for FPGAs.

3 Related works

3.1. Security Studies on FPGA

Security studies related to attacks on FPGAs have generally been categorized based on attack types or deployment environments. Duan et al. [10] classified security threats into major attack techniques such as side-channel attacks, fault injection, and covert channels, and systematically presented their subtypes and manifestation mechanisms. In the case of side-channel attacks, threats were categorized according to information leakage paths, including long-wire effects and electromagnetic (EM) emissions. Fault injection attacks were further classified by attack mechanisms, including timing violations induced by internal clock manipulation (clock glitches) and timing violations or denial-of-service (DoS) caused by power or voltage modulation. In addition, the authors analyzed the realistic threat levels of attacks that may arise depending on the deployment environment, such as FPGA-based accelerators, system-on-chip (SoC) FPGAs, and cloud environments (e.g., information leakage from cloud-deployed FPGAs).

Mirzargar et al. [16] focused on side-channel attacks with physical characteristics that extract information by observing power consumption or electromagnetic radiation, as well as covert channels that intentionally manipulate existing resources not originally intended for information transfer (e.g., power, delay, temperature) to establish unauthorized communication paths. For these attack categories, further classifications were performed based on attack execution methods, including power-based techniques utilizing ring oscillators and delay lines, and temperature-based techniques targeting heaters and temporal channels. For each identified threat, the authors also discussed the environments in which the attack could pose a risk (e.g., attacks conducted within the same FPGA or on other FPGAs located on the same printed circuit board) along with the required sophistication of the attack equipment.

These studies are effective in elucidating the detailed mechanisms of individual attack techniques. However, they primarily classify threats from a technical perspective, which limits their ability to capture when such threats are introduced during the overall development process and how they propagate or become activated across different phases of the development lifecycle. Accordingly, this study does not treat threats solely as isolated technical phenomena or confine them to a single development stage but instead proposes a lifecycle-based analytical framework that considers threats across the entire development lifecycle in a more realistic and integrated manner.

3.2. FPGA for Safety-Critical Systems

In the nuclear domain, FPGAs have been considered a major technology for overcoming the limitations of conventional microprocessor (PLC)-based systems, and accordingly, a wide range of studies has been conducted.

Farias et al. compared and analyzed design approaches based on different FPGA architectures, including SRAM, Flash, and Antifuse-based devices, and evaluated their fault tolerance and cost efficiency when applied to nuclear systems [11]. Based on this evaluation, they proposed a redundancy-based design that combines multiple architectures as a structural approach to cope with accidental faults. She et al. implemented a reactor shutdown system for CANDU reactors using FPGAs and conducted hardware-in-the-loop simulations to demonstrate the advantages of the proposed system [19]. Their study emphasized that FPGA-based systems provide faster response times and more stable deterministic behavior than legacy systems, thereby enhancing the ability to perform safety functions under emergency conditions. In addition, Wang et al. considered radiation environments in nuclear facilities and proposed an error recovery technique based on error-correcting codes (ECC) to protect FPGA block RAM [27]. To mitigate errors caused by single-event upset, which induces unintended bit flips, they applied a Bose–Chaudhuri–Hocquenghem (BCH) code-based design, aiming to compensate for the limitation of redundancy-based approaches that can detect errors but have difficulty providing error correction.

However, the overall trend of these studies is predominantly focused on reliability, which has traditionally

been a top priority in nuclear facilities. In safety-critical systems such as nuclear installations, ensuring system dependability requires not only performance-related aspects, such as signal processing speed, but also consideration of system failures caused by security vulnerabilities. Therefore, to complement existing reliability-centered approaches, this study proposes a framework for systematically identifying security threats that may arise throughout the entire FPGA development lifecycle and analyzes major threats accordingly.

4 Methodology

4.1. Design of Research Methodology

As the use of FPGAs in SCS continues to increase, ensuring the security of FPGAs deployed in SCS requires systematic verification not at a single implementation stage, but across development artifacts generated throughout the entire development lifecycle, from design to operation. To this end, this study adopts the software V&V processes defined in IEEE Std 1012 as the reference framework for threat modeling, as the standard structurally specifies verification requirements for outputs generated at each stage of the FPGA development process.

IEEE Std 1012 defines a lifecycle process consisting of Acquisition, Supply, Development, Operation, and Maintenance, and specifies the V&V activities to be performed at each stage. Among these, the Acquisition and Supply phases are primarily document-driven stages focused on definition and review, in which concrete implementation characteristics of FPGAs are only weakly reflected. In contrast, during the Development, Operation, Maintenance phases, actual hardware implementation, testing, and operational activities are performed, and concrete outputs are generated at each phase, resulting in a high degree of correspondence with the FPGA development process.

Accordingly, this study defines its analysis scope by focusing on the Development, Operation, Maintenance phases of the IEEE Std 1012's lifecycle, where the physical and implementation-specific characteristics of FPGAs are explicitly reflected. By mapping these phases to commonly adopted FPGA development guidelines (UG892), the proposed methodology identifies security threats that may arise at the boundaries where development outputs are transformed, and organizes the processes of threat introduction, propagation, and activation from a lifecycle-oriented perspective. In addition, this study considers representative testing activities typically performed during V&V at each phase, allowing such testing activities to be considered alongside development outputs.

A schematic overview of the IEEE Std 1012 lifecycle processes and associated V&V activities referenced in designing the proposed framework is shown in Figure 1.

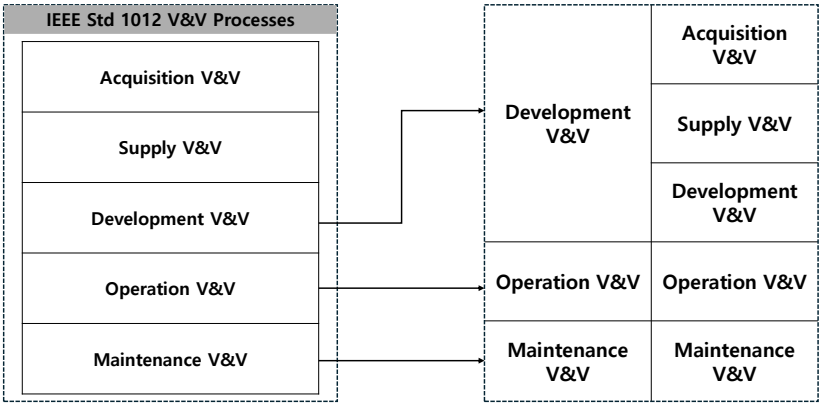

Figure 1. Overview of Referenced IEEE Std 1012 V&V Processes

4.2. FPGA-Specific Lifecycle Framework

This study establishes an FPGA-specific lifecycle framework by integrating the practical FPGA development flow referenced in UG892 with the lifecycle structure defined in IEEE Std 1012, focusing on key transformation points at which security threats may be introduced, modified, or activated. Figure 2 shows the overview of the FPGA development lifecycle framework. Based on data transformation boundaries and V&V application points, the framework is structured into two high-level levels: Implementation and Commissioning & Operation. These phases are further decomposed into six detailed phases spanning from HDL design to Operation & Maintenance.

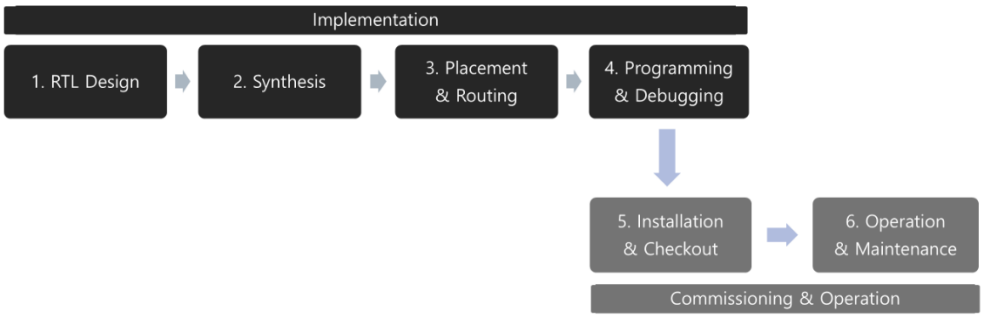

Figure 2. Overview of the FPGA development lifecycle framework

4.2.1. Implementation

This level represents the process in which design requirements are concretized into a bitstream file (.bit) that is finally loaded onto an empty FPGA device. Each phase receives the previous phase's output, performs V&V activities, and passes the verified output to the next level (Commissioning & Operation).

HDL Design This phase translates system functional requirements into logical structures and behaviors, which are described using HDLs such as Verilog or VHSIC HDL (VHDL). At this phase, designers define the hierarchical structure of top-level and submodules, as well as core system logic including clock domains and reset schemes. The resulting register-transfer level (RTL) code constitutes the most fundamental source artifact

defining the logical behavior of the hardware and serves as the input for subsequent synthesis and implementation phases.

Since the RTL code directly specifies the intended functional behavior of the system, testing activities at this stage primarily aim to validate the correctness and consistency of the described logic before it is transformed into lower-level representations. Representative testing activities commonly associated with the RTL design phase include the following:

- Functional simulation: Verifying that the RTL logic produces correct outputs for given input stimuli in accordance with the specified functional requirements [4].

- Code coverage analysis: Quantitatively evaluating how comprehensively the executed functional simulations exercise the HDL code by tracking the execution of lines, branches, and conditional paths, and generating coverage reports to assess verification completeness [4].

Synthesis In this phase, the RTL code is transformed into gate-level representations composed of actual hardware primitives available on the target FPGA device, such as LUTs and flip-flops. During synthesis, the EDA tool automatically performs logic optimization to satisfy designer-specified timing and area constraints, finally generating a gate-level netlist optimized for the specific FPGA architecture. The synthesized netlist serves as a key development output that reflects both the functional intent of the RTL design and the effects of synthesis-level optimizations.

Since synthesis involves structural transformations and logic optimizations, verification activities at this stage aim to ensure that these transformations do not alter the intended functionality of the original RTL design. Representative testing activities commonly associated with the synthesis phase include the following:

- Logical equivalence check: Verifying that the synthesized gate-level netlist is logically equivalent to the original RTL description, ensuring that logic optimization and technology mapping do not introduce unintended functional changes [5].

- Gate-level simulation: Re-simulating the synthesized gate-level netlist to identify synthesis-induced functional discrepancies that may not be observable at the RTL simulation level, such as differences in initialization behavior or logic restructuring effects [4].

Placement & Routing This phase assigns the synthesized logic elements to physical locations within the FPGA fabric (placement) and connects them using routing resources to establish the final hardware configuration. The resulting output is a fully mapped and routed design that includes complete physical information, such as detailed timing characteristics, resource utilization, power consumption estimates, and standard delay format (SDF) files representing actual signal propagation delays.

Because placement and routing determine the physical realization of the design, verification activities at this stage primarily focus on confirming that the implemented design satisfies timing and resource constraints under realistic operating conditions. Representative testing and analysis activities associated with this phase include the following:

- Static Timing Analysis (STA): Evaluating whether signal paths in the implemented design satisfy specified timing constraints by analyzing propagation delays and slack values after routing delays are reflected, without relying on dynamic simulation [7].

- Utilization and power analysis: Assessing resource utilization and power consumption based on the placed-and-routed design to verify feasibility with respect to system-level constraints such as power budgets and thermal limits [6].

Programming & Debugging In this phase, the finalized physical design data are converted into a binary bitstream that can be interpreted by the FPGA device. The bitstream is then loaded into the device configuration

memory to configure the FPGA hardware. This phase encompasses bitstream generation, device programming, and initial on-hardware debugging, ensuring that the implemented logic operates as intended on the physical device.

Because this phase directly transitions the design from an implementation to an operational hardware component, verification activities focus on confirming the correctness, integrity, and system-level behavior of the programmed FPGA. Representative testing activities associated with this phase include the following:

- Bitstream integrity and security verification: Confirming that the generated bitstream has not been modified and that intended security options, such as encryption and authentication, are correctly applied during device configuration [2, 8].

- System integration test: Evaluating the programmed FPGA within the target system to confirm that system-level requirements and interface constraints are satisfied under representative operational conditions.

4.2.2. Commissioning & Operation

This level covers the process by which verified FPGA modules are installed, integrated into real industrial systems, operated to perform safety functions, and maintained over their service lifetime.

Installation & Checkout This phase involves physically installing the developed and verified FPGA design into its final operational environment and integrating it with the overall system. Verification activities at this stage assess whether the system reliably satisfies functional and performance requirements under real-world conditions that are difficult to reproduce in laboratory environments, such as variations in power supply, temperature, and electromagnetic interference. Accordingly, the focus extends beyond revalidating individual functions to confirm environmental compatibility and operational stability.

Representative testing activities associated with the installation and checkout phase include the following:

- Installation configuration audit: Verifying that all installed hardware, software, configuration files, and documentation are consistent with the approved baseline. In accordance with the installation configuration audit procedures defined in IEEE Std 1012, this activity compares items such as bitstream version or hash data, applied security options, and device revisions to detect unauthorized modifications that may occur during delivery or installation, thereby ensuring system integrity and traceability.

- Installation checkout / Site acceptance test: Conducting on-site testing of the fully installed system to confirm that operational requirements and interoperability constraints are satisfied in the final deployment environment. Following procedures described in relevant standards such as IEC 62381 [14], this activity verifies system behavior under actual power-up, reset, and integration scenarios that cannot be completely validated in laboratory settings.

Operation & Maintenance During the operation & maintenance phase, the system is continuously operated in its target environment to perform its intended safety functions throughout the system lifecycle. This phase includes routine operation, continuous monitoring, and maintenance activities such as firmware or configuration updates to address discovered vulnerabilities or to improve system functionality.

Representative testing and verification activities associated with this phase include the following:

- Real-time monitoring: continuously observing system behavior using monitoring mechanisms such as network intrusion detection systems (NIDS), system health monitoring tools, and process supervision functions to detect anomalies including unauthorized access attempts, malicious code execution, or abnormal operational states. These activities support timely detection and response to security incidents and operational faults during runtime [17].

- Log analysis: collecting and analyzing operational logs, access records, and security event logs to verify normal system behavior during routine operation and to support forensic analysis in the event of security incidents or system failures. Log analysis provides evidence for tracing incident causes, attack paths, and potential impacts over extended operational periods.

The tasks performed and the corresponding outputs for each phase are summarized in Table 2. Based on the FPGA development lifecycle framework defined in this study, the threat characteristics that may arise at each lifecycle phase are systematically analyzed and described.

Table 2. FPGA Development Lifecycle in our study

Level	Phase	Output	Tasks
Implementation	RTL Design	HDL code	• Logic definition • IP integration • I/O planning
	Synthesis	Netlist	• Logic optimization • Constraint application • Gate mapping
	Placement & Routing	Mapped design	• Logic Placement • Routing optimization • Timing closure
	Programming & Debugging	Bitstream	• Bitstream generation • Device programming • Functional validation
Commissioning & Operation	Installation & Checkout	Installed system	• System integration • Site acceptance test • Configuration verification
	Operation & Maintenance	System log	• Continuous monitoring • Periodic testing • Log analysis

5 Discussion

This study contributes by proposing a systematic analytical framework for addressing security concerns in FPGA-based SCS, moving beyond conventional approaches that primarily focus on functional safety. In particular, the establishment of an analytical baseline by integrating the V&V lifecycle processes defined in the IEEE Std 1012—widely adopted in high-assurance domains such as nuclear and defense systems—with the FPGA industry-standard development guideline UG892 highlights the practical applicability of the proposed approach. Rather than limiting the analysis to individual attack techniques, this framework provides a structured method for identifying security threats in direct relation to development-phase outputs, thereby enabling developers to systematically understand how vulnerabilities may be introduced and propagated throughout the FPGA development lifecycle. Furthermore, because the proposed framework adopts a vendor- and system-agnostic structure, it can be readily extended to a wide range of safety-critical system domains in future applications.

This study presents the design of a lifecycle-oriented framework up to the level of defining development phases, key artifacts, and associated test perspectives. However, this work does not include detailed analyses of specific threat cases using the proposed framework. Representative threat scenarios—such as major instances of hardware trojans or side-channel attacks—are not examined to illustrate how the framework can be applied in practice. As a result, the applicability of the proposed framework within real-world development and verification workflows has not yet been demonstrated through concrete usage scenarios. Such applications

are left for future work, where the framework can be exercised against representative threat cases to assess its practical use in realistic FPGA development and V&V processes.

6 Conclusion and Future Work

This study proposes an analytical framework that integrates the V&V lifecycle defined in IEEE Std 1012 with the FPGA development process, with the aim of systematically identifying security threats affecting FPGAs deployed in SCS. Unlike prior studies that primarily focus on technical analyses of individual attack techniques or on accidental failures related to operational safety, this study emphasizes the structured identification of security vulnerabilities that may arise at each point where development outputs are transformed throughout the entire lifecycle, from early design stages to system operation.

Using the proposed framework, this study does not perform detailed analyses of specific attack cases across individual development phases. Instead, the framework is presented to illustrate how security considerations can be systematically organized and discussed across successive development phases, including RTL design, synthesis, placement and routing, and bitstream generation, without centering on concrete threat analyses. Accordingly, the discussion of potential threat behaviors is limited to a conceptual level, focusing on how such issues may be considered within the lifecycle rather than demonstrating analytical results derived from specific cases.

Building on the proposed framework, subsequent research will focus on applying the framework to representative threat scenarios to demonstrate its practical usage and applicability in real-world FPGA development and verification environments. Such studies may examine concrete threat cases and explore how the framework can be used to support security-related discussions and decision-making within V&V activities mandated by standards such as IEEE Std 1012, as well as in safety-critical domains including nuclear, automotive, and aerospace systems.

Funding Details

This work was supported by the Nuclear Safety Research Program through the Korea Foundation Of Nuclear Safety (KoFONS) using the financial resource granted by the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea. (No. RS-2024-00403596)

References

[1] Ahmed, I., Jung, J., & Heo, G. (2017). Design verification enhancement of field programmable gate array-based safety-critical I&C system of nuclear power plant. Nuclear Engineering and Design, 317, 232–241.
[2] AMD Xilinx. (2015). 7 series FPGAs configuration user guide (Report No. UG470, v1.10).
[3] AMD Xilinx. (2022). Vivado Design Suite user guide: Design flows overview (Report No. UG892, v2022.1).
[4] AMD Xilinx. (2022). Vivado Design Suite user guide: Logic simulation (Report No. UG900, v2022.1).
[5] AMD Xilinx. (2022). Vivado Design Suite user guide: Synthesis (Report No. UG901, v2022.2).
[6] AMD Xilinx. (2022). Vivado Design Suite user guide: Implementation (Report No. UG904, v2022.2).
[7] AMD Xilinx. (2022). Vivado Design Suite user guide: Design analysis and closure techniques (Report No. UG906, v2022.1).
[8] AMD Xilinx. (2023). UltraScale architecture configuration user guide (Report No. UG570, v1.17).
[9] Bernardeschi, C., Cassano, L., & Domenici, A. (2015). SRAM-based FPGA systems for safety-critical applications: A survey on design standards and proposed methodologies. Journal of Computer Science and Technology, 30(2), 373–390.
[10] Duan, S., Wang, W., Luo, Y., & Xu, X. (2021). A survey of recent attacks and mitigation on FPGA systems. 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 461–466.
[11] Farias, M. S., Nedjah, N., & de Carvalho, P. V. R. (2022). Active redundant hardware architecture for increased reliability in FPGA-based nuclear reactors critical systems. Microprocessors and Microsystems, 90,

104495.
[12] Gerometta, G., Conficconi, D., & Santambrogio, M. D. (2021). On how FPGAs are changing the computer security panorama: An educational survey. 2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI), 80–85.
[13] Institute of Electrical and Electronics Engineers. (2005). IEEE standard for software verification and validation (IEEE Std 1012-2004). IEEE Computer Society.
[14] International Electrotechnical Commission. (2024). Automation systems in the process industry—Factory acceptance test (FAT), site acceptance test (SAT), and site integration test (SIT) (IEC 62381:2024, Ed. 3.0).
[15] Mal-Sarkar, S., Karam, R., Narasimhan, S., Ghosh, A., Krishna, A., & Bhunia, S. (2016). Design and validation for FPGA trust under hardware Trojan attacks. IEEE Transactions on Multi-Scale Computing Systems, 2(3), 186–198.
[16] Mirzargar, S. S., & Stojilović, M. (2019). Physical side-channel attacks and covert communication on FPGAs: A survey. 2019 29th International Conference on Field Programmable Logic and Applications (FPL), 202–210.
[17] National Institute of Standards and Technology. (2024). Guide to operational technology (OT) security (NIST Special Publication 800-82, Rev. 3).
[18] Omidi Gosheblagh, R., & Mohammadi, K. (2013). Dynamic partial based single event upset (SEU) injection platform on FPGA. International Journal of Computer Applications, 76, 19–24.
[19] She, J., & Jiang, J. (2011). On the speed of response of an FPGA-based shutdown system in CANDU nuclear power plants. Nuclear Engineering and Design, 241(6), 2280–2287.
[20] Sterpone, L., & Violante, M. (2005). Analysis of the robustness of the TMR architecture in SRAM-based FPGAs. IEEE Transactions on Nuclear Science, 52(5), 1545–1549.
[21] Sunkavilli, S., Zhang, Z., & Yu, Q. (2021). New security threats on FPGAs from FPGA design tools perspective. 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 278–283.
[22] U.S. Nuclear Regulatory Commission. (2013). Verification, validation, reviews, and audits for digital computer software used in safety systems of nuclear power plants (Regulatory Guide 1.168, Rev. 2).
[23] U.S. Nuclear Regulatory Commission. (2013). Advanced Logic System topical report (Report No. 6002-00301-NP, Rev. 3).
[24] U.S. Nuclear Regulatory Commission. (2018). Amendment for HFC-FPGA system of HFC-6000 safety platform (Report No. RR901-107-10-NP, Rev. F-A).
[25] U.S. Nuclear Regulatory Commission. (2020). RadICS topical report (Report No. 2016-RPC003-TR-001, Rev. 2).
[26] U.S. Nuclear Regulatory Commission. (2017). Design of the highly integrated protection system platform (Report No. TR-1015-18653, Rev. 2).
[27] Wang, X., Holbert, K. E., & Clark, L. T. (2011). Single event upset mitigation techniques for FPGAs utilized in nuclear power plant digital instrumentation and control. Nuclear Engineering and Design, 241(8), 3317–3324.

Author's Biography

Dongmin Kim received the B.S. degree in Data Science from Sungkyunkwan University. He is currently pursuing a master's degree in Immersive Media Engineering at the same university. His research interests include cybersecurity, industrial control systems, and cyber-physical systems security.

Sooyon Seo received the B.S. degree in Global Business Administration from Sungkyunkwan University, Seoul, Republic of Korea, in 2023. She is currently pursuing the Ph.D. degree in Immersive Media Engineering at Sungkyunkwan University. Her research interests include generative artificial intelligence and cybersecurity.

Moohong Min received the B.S. and M.S. degrees from Sungkyunkwan University in 2010 and 2012, respectively, and the Ph.D. degree from Korea University, Seoul, Republic of Korea. He is currently an Assistant Professor at Sungkyunkwan University. His research interests include artificial intelligence and cybersecurity.

Aram Kim received the B.S. and M.S. degrees in Computer Science from Korea University in 2005 and 2008 respectively, and the Ph.D. degree from the School of Cybersecurity, Korea University, Seoul, Republic of Korea. Her research interests include cybersecurity, industrial control systems, insider threat detection, cyber-physical systems security, abnormal detection.