Procedures for Developing Secure FPGA in Nuclear Power Plants

Minseok Hur[a*], Jiho Shin[b*], Moohong Min[c], and Aram Kim[d**]**
[a]Department of Immersive Media Engineering, Sungkyunkwan University, Seoul, Republic of Korea
[b]Department of Forensics, Sungkyunkwan University, Seoul, Republic of Korea
[c]Department of Computer Education/Social Innovation Convergence Program, Sungkyunkwan University, Seoul, Republic of Korea
[d]Department of Information Security, University of Suwon, Hwaseong-si, Republic of Korea
[*] These authors equally contributed to this paper.
[**] Corresponding Authors: Moohong Min, Aram Kim

Abstract: Field programmable gate arrays (FPGAs) are reconfigurable integrated circuits that can be configured to implement complex digital circuits using hardware description languages (HDLs). With features such as parallelism, timely responsiveness, reconfigurability, and fault tolerance, FPGAs are well-suited for applications in nuclear power plants. While FPGAs are generally secure during the operation phase, functioning like hardware, they have been found vulnerable to various cyber-attacks. In this study, we analyze FPGA vulnerabilities based on Common Vulnerabilities and Exposures (CVEs) categorizing them into five types, including inadequate access control and buffer management issues. To address these vulnerabilities, we propose a security-focused development procedure that follows the V-model and leverages Common Weakness Enumeration (CWE) to identify and mitigate weaknesses commonly found in both HDL and FPGA circuit development. The proposed framework enhances security, safety, and quality by addressing specific, concrete weaknesses within the development, verification, and validation stages, ultimately improving development efficiency.

Keywords: FPGA Software Security, Secure Development Lifecycle, Hardware Security

1. Introduction

1.1. Overview of FPGA Technology

Field programmable gate arrays (FPGAs) are extensively utilized across domains such as embedded systems, robotics, power electronics, and AI-driven industrial controls [1]. These highly versatile, user-programmable devices consist of an array of programmable logic cells interconnected within a matrix, facilitating the implementation of diverse logic functions. The FPGA design process encompasses stages like mapping, as well as placement and routing (PAR), which serve to optimize the hardware configuration [2]. Recognized for their ability to execute multiple operations simultaneously, FPGAs integrate with computer-aided design (CAD) tools to efficiently manage complex designs. Unlike traditional systems dependent on operating systems, FPGAs are dedicated solely to essential functions, thereby enhancing performance while minimizing system complexity. Although developed with a software-like approach, FPGAs operate as hardware during execution.

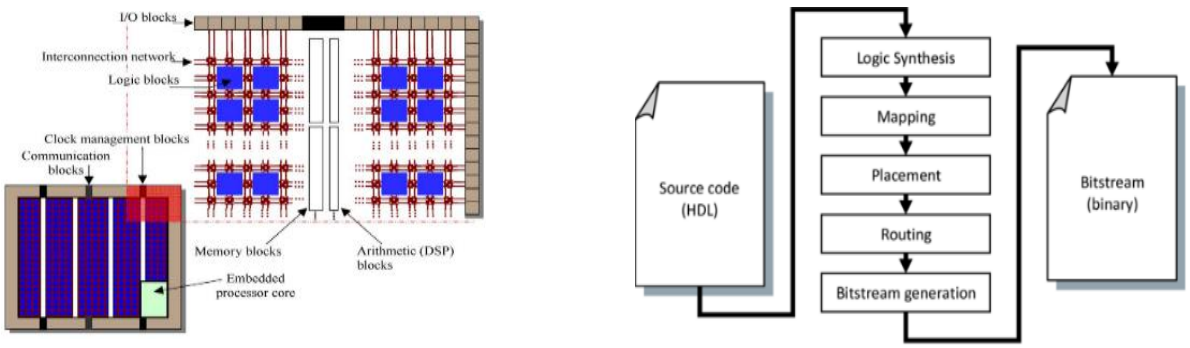

Figure 1. FPGA Architecture [1] and Bitstream Generation [4]: The left side illustrates the FPGA layout, highlighting the arrangement of logic blocks, interconnections, and I/O cells. The right side depicts the process of bitstream generation from HDL code.

FPGAs comprise programmable logic blocks, interconnects, and input/output (I/O) blocks that facilitate communication with external systems [3]. The logic blocks, constructed from look-up tables (LUTs), flip-flops (F/Fs), and multiplexers (MUXs), connect through interconnects that route channel data via paths of switches and routing channels. In order to implement a design, the hardware description language (HDL) code undergoes several critical processes, including synthesis, mapping, and placement and routing, ultimately producing a bitstream. This bitstream is then used to configure the FPGA components to achieve the intended functional objectives.

1.2. FPGA Applications in Nuclear Power Plants

FPGAs are increasingly prevalent in nuclear power plant instrumentation and control (I&C) systems, largely due to their reconfigurability, parallel processing capabilities, reliability, rapid response times, and regulatory compliance adaptability. As nuclear facilities seek to modernize aging infrastructure, FPGA technology offers an efficient solution for replacing outdated systems. This programmability and deterministic operation of FPGAs make them particularly well-suited for safety-critical environments, where predictable performance is essential. According to the International Atomic Energy Agency (IAEA) Nuclear Energy Series [5], these qualities are especially advantageous for the modernization of control systems in existing nuclear plants.

In existing plants, FPGA-based upgrades help address obsolescence and reliability challenges associated to legacy microprocessor-based control modules [6]. FPGAs enable system improvements without the need for extensive hardware replacements, allowing for faster response times, enhanced safety, and reduced system complexity. Their reconfigurable architecture consolidates multiple functions onto fewer modules, an advantage in space-constrained environments while maintaining high performance and reliability.

For new nuclear power plants, FPGAs offer a robust means of complying with stringent safety and regulatory standards [7]. The inherent parallel processing capabilities and absence of operating systems facilitate rapid and reliable execution of safety-critical functions. Furthermore, the obsolescence resistance and re-programmability of FPGAs extend the control systems' operational lifespan, lowering long-term maintenance costs and supporting adaptability to future regulatory requirements.

2. Development Process for FPGA

Several standards have been published for the design, development, and implementation of FPGAs across industries such as aerospace and military [9-11]. In the nuclear power sector, the International Electrotechnical Commission (IEC) has published IEC 62566 [10], a standard focused on development of FPGAs (referred to as hardware description language programmed devices within the standard) for use in systems performing Category A functions or safety functions in nuclear power plants. Category A refers to functions that play a primary role in maintaining safety by preventing design-based events from leading to unacceptable consequences [11]. Safety functions cover activities related to ensuring operational safety, such as controlling reactivity, removing heat from the core, containing radioactive materials, controlling operational discharges, and limiting accidental releases [12]. IEC 62566 also outlines procedures for modification and configuration control of FPGA-based systems, and the selection and application of tools used in FPGA development.

IEC 62566 provides an structured approach to the requirements for designing, implementing, integrating, and validating FPGA development with HDLs and associated software tools to meet these requirements. Verification and validation (V&V) processes are integral to assuring the safety and quality of systems, software, and hardware engineering [13]. These processes are critical for confirming that the developed product aligns with activity requirements and meets its intended use and user needs [13-16]. Figure 2 presents a V-shaped diagram that offers an overview of the development life cycle integrated with V&V processes.

2.1. Requirements Specification

This phase involves documenting the following: 1) functional requirements, 2) design specifications, 3) fault tolerance strategies, and 4) the use of tools at the Electronic System Level (ESL), which provides a high-level representation of an electronic system's component functionalities. The specification includes details on 1) the functions performed by the FPGA, 2) its operating modes, 3) its interfaces and interactions with the environments, 4) parameters that can be manually modified during operation and their roles, 5) FPGA's

performance in terms of response time, and 6) assumptions about the FPGA's operating environment, including electrical and temporal characteristics of input-output signals, power supplies, and specific profiles for power-on and cooling phases.

For design specification, the FPGA should adhere to a deterministic design, ensuring that any input sequence following specific electrical and temporal parameters consistently produces the same outputs. Additionally, requirements for defensive design should be established to address fault detection, such as handling erroneous data modification, and fault tolerance, detailing the expected behavior when a fault is detected.

To verify the requirements specification, a rigorous analysis is conducted to identify potential inconsistencies, omissions, and ambiguities. This analysis needs to cover all functional requirements and other requirements, performed by engineers and specialists with expertise in the FPGA's interface and functional areas.

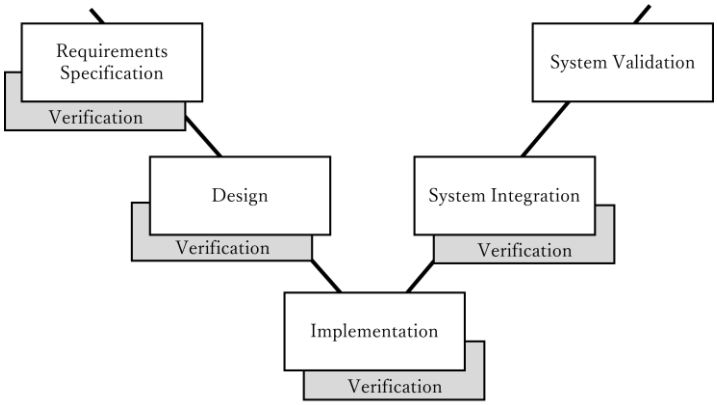

Figure 2. Development Life Cycle of an FPGA [10]

2.2. Design

IEC 62566 outlines nine key requirements for clear and verifiable design: 1) defensive design, 2) structure, 3) language and coding rules, 4) synchronous/asynchronous design, 5) power management, 6) initialization, 7) non-functional configurations, 8) testability, 9) design documentation.

For 1) defensive design, fault detection, and corresponding actions should be performed in accordance with the requirement specification, including the identification of micro-technologies such as native blocks and Pre-Developed Blocks (PDBs). For 2) structure, a top-down approach is preferred, and the structure should be based on the decomposition of modules. For 3) Language and coding rules, proven design methodology, strict design rules with the latest design knowledge, and enforcement for compliance with the rules should be used. For 4) synchronous/asynchronous design, a synchronous design should be used for stable and robust design, where the changes of the registers are simultaneous and follow the ticking of a clock signal. 5) Power management is needed to consider the electrical and temporal characteristics of the circuit during power-up, power-down, and sudden loss of power. Behavior of each pin must be documented. In 6) initialization, the state of output signals, registers, and finite state machine should be known according to the assertion and de-assertion of the initialization signal. 7) Non-functional configurations are needed for analysis and configuration on special pins and registers that are not specified in the requirements to exclude any adverse impact on functions. 8) Testability is related to the testability of each function implemented in the FPGA device by self-tests, periodic tests, or observable contribution to a higher-level function, which is itself submitted to self-tests or periodic tests. In 9) design documentation, documentation is needed to describe and justify the adequacy of the design decisions that fulfill the requirements. It should include all parameters needed to unambiguously configure and use all native blocks and PDBs. Estimated timings and electrical characteristics should be included.

After defining these requirements, a detailed FGPA description, known as a Register Transfer Level (RTL) model, is developed. The RTL model is a synchronous parallel representation of an electronic circuit, detailing FPGA functions with signals processed through combinational logic and transferred between registers via clock pulses. This model is typically written using HDLs such as Verilog (IEEE 1364 [17]) or VHDL (IEEE

1076 [18]). Importantly, the RTL content is independent of the actual hardware (i.e., the FPGA circuit), enhancing FPGA applications' resilience to hardware obsolescence [7].

2.3. Implementation

After completing the RTL design, the implementation phase is divided into two steps: 1) synthesis and 2) PAR. In the synthesis step, the circuit-independent description is translated into a circuit-dependent description known as a "netlist", which utilizes the resources of the selected FPGA. Testing is conducted through simulation using logic synthesis tools, which generate timing information, also known as "back-annotation," to precisely simulate the temporal behavior by accounting for delays associated with gates and wires. In the PAR phase, optimal physical positioning and mapping of FPGA resources are calculated to ensure uniform data distribution across the interconnection grid. This phase generates the physical description required to produce the FPGA, ultimately creating a programming file or a "bitstream."

Parameters and constraints, such as operating frequency and timing relationships between signals must be specified to execute synthesis and PAR operations. Additionally, a post-route analysis is performed to verify that the design meets the specified RTL description and timing constraints. Static timing analysis (STA) is performed and documented under both best- and worst-case scenarios to calculate timing margins and confirm the compatibility of each clocked block's frequency with all possible conditions.

The verification of the design and implementation includes series of tests and analyses to assess the design's adequacy, testability, understandability, modifiability, and correct implementation of safety requirements. Test benches can be developed for simulating and testing modules according to the requirement specifications. The V&V team, comprising software and hardware experts and engineers, performs these verification operations, ensuring that the FPGA design and implementation are complete, correct, and robust.

2.4. System Integration

During this phase, system integration and integrated system verification are performed. System integration involves assembling and interconnecting verified hardware and software components to build intermediate and final system targets. Integrated system verification ensures that these components meet their design specification, can operate together, and adhere to interface requirements. System integration follows the system integration plan, which is developed early in the project and outlines all integration requirements,as specified in Clause 6 of IEC 61513 [19]. The plan must specify 1) the sequences and timing of input signals to the system or subsystem tested, 2) the sequence and timings of expected outputs from the system or subsystem being tested, and 3) the acceptance criteria. Also, it should include requirements for procedures and control methods covering 1) system configuration control, 2) system integration, 3) integrated system verification, and 4) fault resolution.

Verification of the components and subsystems is needed to confirm their correct integration into the system and their performance as specified. The system should be as complete as possible at this phase, and test cases for system verification need to 1) exercise all FPGA interfaces and all basic operations, 2) exercise all interface characteristics in the requirement specification according to its protocols, sequences, timings, and electrical features, and 3) have sufficient coverage to demonstrate that the FPGA performs as required for all cases reachable in the system.

2.5. System Validation

In this phase, the final system undergoes testing according to the system validation plan, which is developed, beforehand, and the validation results are evaluated by individuals independent of the design and implementation phases. System validation is conducted using both static and dynamic input signals that simulate normal operations, expected events, and accident conditions. The tests must encompass all functions specified in the requirement specification, covering, to the extent possible, all signal and parameter ranges, all trip or protective signals in the final assembly configuration, required responses to specified failures, and any other functions impacting reactor safety. The system validation report should document the test results, detailing the hardware, software, configurations, tools, and any discrepancies identified during testing. It should also comply with Clause 6.4.6.2.b of IEC 61513 [19].

3. FPGA Security: Threats and CWE-Based Countermeasures

3.1. Analysis of FPGA Security Vulnerabilities Based on CVE

To gain a deeper understanding of cybersecurity threats affecting FPGA systems, we conducted a comprehensive analysis of publicly available security vulnerabilities, with a particular focus on those documented in the Common Vulnerabilities and Exposures (CVE) database. The CVE database, a widely recognized repository for cataloging security weaknesses, assigns each vulnerability a unique identifier and provides detailed descriptions of its potential impact and severity. By examining CVE entries related to FPGA devices, we aim to illuminate specific security issues identified across various manufacturers and models. This analysis builds on existing knowledge and offers valuable insights into the types of vulnerabilities to which FPGA systems are particularly susceptible, guiding future research and mitigation strategies. Table 1 summarizes key CVE entries, highlighting the range of vulnerabilities, their nature, and the severity scores that reflect their potential impact.

Table 1. Reported FPGA Vulnerabilities Based on CVE Data

CWE Number	Vulnerability	Vulnerability Score
CVE-2010-0928 [20]	Verify Signature	4.0
CVE-2014-2107 [21]	CSCug84789	7.1
CVE-2019-11165 [22]	Check for Inappropriate Condition	5.5
CVE-2019-14604 [23]	Dereferencing Null Pointers	5.5
CVE-2019-14625[24]	Inadequate Access Control	4.4
CVE-2019-14626 [25]	Inadequate Access Control	6.7
CVE-2019-1649 [26]	Logic Handling Access Control	6.7
CVE-2019-1700 [27]	Ingress Buffer Management	6.1
CVE-2020-0574 [28]	Improperly Configured Design	5.9
CVE-2020-8737 [29]	Improper Buffer Restriction	6.8
CVE-2020-12312 [30]	Improper Buffer Restriction	6.8
CVE-2020-24485 [31]	Inappropriate Condition	7.8
CVE-2020-8684 [32]	Inadequate Access Control	6.7
CVE-2020-8737 [33]	Improper Buffer Limit	6.8
CVE-2021-27208 [34]	Classic Buffer Overflow	6.8
CVE-2021-44850 [35]	Classic Buffer Overflow	6.8
CVE-2022-23822 [36]	Incorrect Authorization	6.8
CVE-2022-26512 [37]	Add-on's Uncontrolled Search Path Element	7.3
CVE-2022-34157 [38]	Inadequate Access Control	7.8
CVE-2022-38787 [39]	Validate Invalid Input	7.8

3.2. Common Categories of FPGA Vulnerabilities

FPGA vulnerabilities identified from the CVE database were analyzed to examine the specific security issues they presented. This analysis revealed common characteristics, leading to the classification of these vulnerabilities into five categories: inadequate access control, buffer management issues, improper configuration and condition handling, authorization and path management flaws, and denial of service (DoS) vulnerabilities. This classification was derived from a thorough review of the CVE data, where shared vulnerability factors were identified and grouped accordingly. Table 2 provides an overview of these categories, summarizing the types of vulnerabilities and their potential impact on FPGA systems.

3.3. CWE and FPGA Security

To effectively address security vulnerabilities in FPGA design, it is critical to reference and align with established frameworks such as the Common Weakness Enumeration (CWE) [38]. The CWE provides a comprehensive, categorized list of software and hardware vulnerabilities, developed collaboratively to clearly define each type of vulnerability. This framework aids developers and security professionals in understanding, identifying, and mitigating specific security risks in various environments, including FPGA systems, as discussed in further detail in Section 4. As technology

advances, the CWE list is regularly updated to capture new exploit mechanisms, vulnerabilities, and implementation details relevant to modern hardware and software systems, ensuring its applicability to emerging areas such as FPGA security.

Table 2. Analysis of FPGA Security Vulnerabilities

Category	Description	Potential Impact
Inadequate Access Control [24, 25, 26, 32, 38]	Failure to enforce access restrictions allows unauthorized access or changes to FPGA configurations	Unauthorized data access, privilege escalation
Buffer Management Issues [27, 29, 30, 33, 34, 35]	Poor buffer handling leads to overflows and possible arbitrary code execution	System crashes, arbitrary code execution
Improper Configuration and Condition Handling [20, 22, 23]	Misconfiguration or incorrect runtime condition handling causes instability or security bypass	System instability, privilege escalation
Authorization and Path Management Flaws [36, 37]	Insufficient permission checks or insecure path management leads to privilege escalation or unauthorized access	Unauthorized resource access, data breaches
Denial of Service (DoS) Vulnerabilities [21]	Resource overloads cause system unresponsiveness or malfunction due to excessive or malicious request	System malfunction, service disruption

The CWE list is structured into several hierarchical structures that categorize vulnerabilities based on context - software development, hardware design, or research concepts. This approach enables users to navigate vulnerabilities from their specific perspective. The software development representation organizes vulnerabilities by concepts frequently encountered in software engineering, while the hardware design representation focuses on vulnerabilities commonly found in hardware-based systems such as FPGAs. The research concept representation provides a more abstract view, supporting academic or exploratory studies of vulnerability types. These representations help align security requirements with industry needs and emerging threats.

Furthermore, CWE provides external mappings to key security standards and frameworks, such as ISA/IEC 62443 requirements for industrial automation, the OWASP Top Ten for common software vulnerabilities, and the CWE Top 25 list, which ranks the most critical security vulnerabilities. These mappings facilitate the application of CWE across diverse security practices emphasizing the most significant vulnerabilities for different sectors, including those critical to FPGA development.

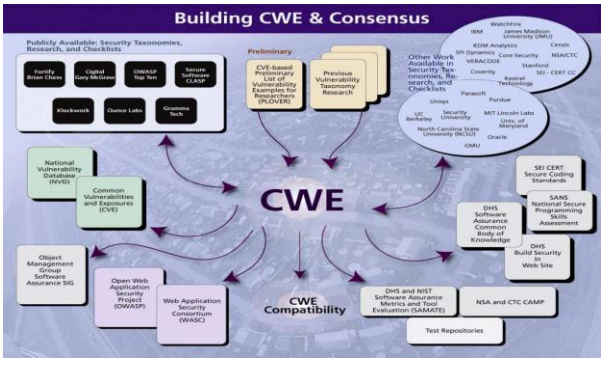

Figure 3. CWE Components

4. Proposed Development Process

As IEC 62566 is a standard focused on safety systems, it does not address security vulnerabilities that constantly emerge with the rapid evolution of digital technology. The V&V processes proposed in the standard primarily ensure product quality rather than specifically addressing safety or security concerns [13]. In this section, we propose a framework that expands the existing V-model to enhance FPGA product security by

incorporating CWE. Our framework aims to improve the efficiency of FPGA system development and enhance safety by implementing targeted mitigation strategies for identified weaknesses and vulnerabilities.

Using the standardized CWE system, our proposed security-focused framework addresses reported and known potential security vulnerabilities throughout the development stages, ensuring that these vulnerabilities are systematically mitigated during the design and development phases to prevent exploitation. Recognizing that vulnerabilities can also be exploited during deployment and operation, we ensure that the system undergoes continuous verification and validation throughout its operation phase.

Figure 4 illustrates the conceptual framework proposed in this paper. Given that FPGA development involves creating a bitstream through HDL and subsequently using this bitstream to configure the hardware circuits, we divided the development process into two phases: HDL Engineering and Hardware Engineering. This framework is designed to support the secure development of FPGA systems by following these two engineering phases and incorporating relevant CWE considerations.

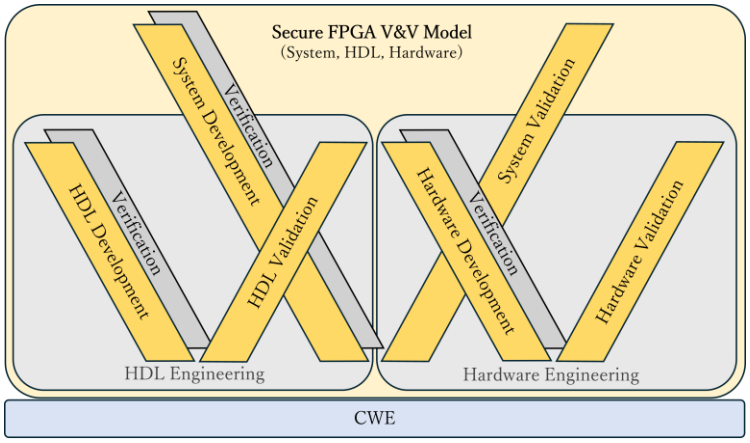

Figure 4. Conceptual V&V Model for Secure FPGA Systems

4.1. Development

The development of HDL and hardware should adhere to existing standards such as IEC 62566 [10] and DO-254/ED-80 [41, 42], which provide design assurance guidance for airborne electronic hardware.

Although FPGA circuits are fundamentally hardware, the use of software tools in HDL development is unavoidable. Therefore, we propose that CWEs related to both hardware and software be considered. As of the latest modification on June 29th, 2023, the CWE catalog includes 445 weaknesses in Software Development and 129 in Hardware Design (Table 3). Within Hardware Design, a category named "General Circuit and Logic Design Concerns" encompasses 14 weaknesses related to hardware-circuit design, logic, and HDLs such as Verilog or VHDL, as detailed in Table 4.

Each weakness entry in the CWE catalog includes detailed attributes such as a description, extended description, common consequences, potential mitigations, relationships, modes of introduction, applicable platforms, demonstrative examples, memberships, vulnerability mapping notes, related attack patterns, and content history as shown on the website (Figure 5). For instance, CWE-1209[1], "Failure to Disable Reserved Bits," describes a vulnerability where reserved bits are left enabled for debugging or control/configure purposes. If these bits remain enabled in a production environment, an adversary could induce unwanted behavior such as system reset or shutdown. A mitigation strategy for this vulnerability might involve designing a feature to disable reserved bits during the design phase or blocking write access to these bits during the integration phase. Similarly, other weaknesses should be systematically addressed during development, with appropriate mitigations integrated at each relevant phase.

[1] https://cwe.mitre.org/data/definitions/1209.html

Table 3. The Number of Software and Hardware Weaknesses

CWE View Name	Number of Weaknesses
Software Development	445
Hardware Design	129

Table 4. Weaknesses Related to Hardware-Circuit Design and Logic

Weakness Name	Weakness ID
Failure to Disable Reserved Bits	1209
Incorrect Register Defaults or Module Parameters	1221
Race Condition for Write-Once Attributes	1223
Improper Restriction of Write-Once Bit Fields	1224
Improper Prevention of Lock Bit Modification	1231
Improper Lock Behavior After Power State Transition	1232
Security-Sensitive Hardware Controls with Missing Lock Bit Protection	1233
Hardware Internal or Debug Modes Allow Override of Locks	1234
Improper Finite State Machines (FSMs) in Hardware Logic	1245
Improper Preservation of Consistency Between Independent Representations of Shared State	1250
Incorrect Selection of Fuse Values	1253
Incorrect Comparison Logic Granularity	1254
Improper Handling of Single Event Upsets	1261
Hardware Logic Contains Race Conditions	1298

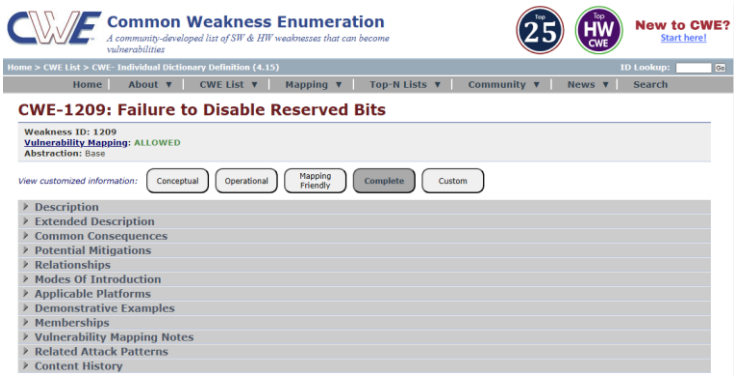

Figure 5. Features Listed on a Single Weakness in CWE

4.2. Verification and Validation

As development progresses, verification and validation should be performed in parallel to ensure that development aligns with requirements and that mitigations are implemented according to the CWE. According to IEEE 1012 [13], V&V processes should be applied to both HDL engineering and hardware engineering. Conceptually, we propose that meeting requirements and systematically addressing mitigations are essential for successful and robust FPGA system.

5. Conclusion

This paper proposes a conceptual framework for the development of FPGA systems in nuclear power plants where safety and security are critical. While FPGAs are generally considered less vulnerable to cyber-attacks compared to microprocessor-based systems, our study underscores the importance of integrating safety and security considerations in the development process. By analyzing the V&V process outlined in IEC 62566 and examining potential security vulnerabilities within FPGA development, we identified that traditional V&V processes prioritize quality over safety and security, and noted that various security threats, such as bitstream theft, side-channel attacks, and the insertion of malicious backdoors, pose potential risks.

To address these concerns, we propose leveraging CWE, a comprehensive framework providing descriptions and mitigation strategies for software and hardware vulnerabilities throughout the FPGA system development lifecycle. We believe that our proposed framework can aid in identifying common software and hardware weaknesses, enabling proactive prevention measures to ensure secure, regulation-compliant FPGA systems for the nuclear power industry. This approach also enhances development efficiency, as specific weaknesses are systematically addressed with clear mitigations, resulting in more robust and secure systems. Future work may focus on developing practical guidance for designing hardware circuits and coding functions or modules that address known vulnerabilities. Additionally, creating a checklist for streamlined development and V&V processes could further facilitate efficient and effective FPGA system development.

Acknowledgments
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] Monmasson, E., Idkhajine, L., Cirstea, M. N., Bahri, I., Tisan, A., Naouar, M. W.: FPGAs in industrial control applications. IEEE Transactions on Industrial Informatics 7(2), 224–243 (2011).
[2] Brown, S. D., Francis, R. J., Rose, J., Vranesic, Z. G.: Field-programmable gate arrays, vol. 180. Springer Science & Business Media, Heidelberg (2012).
[3] Farooq, U., Marrakchi, Z., & Mehrez, H.: Tree-based heterogeneous FPGA architectures: application specific exploration and optimization. Springer Science & Business Media. (2012).
[4] Arcas Abella, O.: Beehive: An FPGA-based multiprocessor architecture. Master's thesis, Universitat Politècnica de Catalunya (2009).
[5] Farias, M. S., Martins, R. H., Teixeira, P. I., & Carvalho, P. V. R. FPGA-based I&C systems in nuclear plants. *Chemical Engineering Transactions*, *52*, 1–6, (2016).
[6] INTERNATIONAL ATOMIC ENERGY AGENCY, Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants, IAEA Nuclear Energy Series No. NP-T-3.17, IAEA, Vienna (2016)
[7] Hayashi, T., Kojima, A., Miyazaki, T., Oda, N., Wakita, K., & Furusawa, T. (2014). Application of FPGA to nuclear power plant I&C systems. *Progress of Nuclear Safety for Symbiosis and Sustainability: Advanced Digital Instrumentation, Control and Information Systems for Nuclear Power Plants*, 41-47.
[8] AMD, "XQR Space-Grade KintexTM UltraScaleTM and Space Heritage", Product Brief, https://www.amd.com/content/dam/amd/en/documents/solutions/amd-space-solution-brief.pdf (2023)
[9] AMD, "Defense-Grade XQ UltraScale Architecture Portfolio", https://www.amd.com/content/dam/amd/en/documents/products/adaptive-socs-and-fpgas/product-briefs/xq-ultrascale-aerospace-defense-product-brief.pdf (2023)
[10] IEC 62566-2012: Nuclear power plants – Instrumentation and control important to safety – Development of HDL-programmed integrated circuits for systems performing category A functions (2012)
[11] Wood, R. T., Joseph III, R. A., Korsah, K., Muhlheim, M. D., & Mullens, J. A. Classification Approach for Digital I&C Systems at US Nuclear Power Plants. *Oak Ridge National Laboratory* (2012).
[12] INTERNATIONAL ATOMIC ENERGY AGENCY, IAEA Safety Standards, No. NS-R-1, IAEA, Vienna (2000)
[13] IEEE 1012-2016: IEEE Standard for System, Software, and Hardware Verification and Validation (2017)
[14] IEEE 1012-2004: Software Verification and Validation (2004)
[15] Wu, Y., Shui, X., Cai, Y., Zhou, J., Wu, Z., & Zheng, J. Development, verification and validation of an FPGA-based core heat removal protection system for a PWR. Nuclear Engineering and Design, 301, 311-319 (2016).
[16] Maerani, R., Mayaka, J. K., Jung, J. C.: Software verification process and methodology for the development of FPGA-based engineered safety features system. Nuclear Engineering and Design 330, 325–331 (2018)
[17] IEEE 1364-2005: IEEE Standard for Verilog Hardware Description Language (2005)
[18] IEEE 1076-2019: IEEE Standard Criteria for VHDL Language Reference Manual (2019)
[19] IEC 61513-2011: Nuclear power plants – Instrumentation and control important to safety – General requirements for systems (2011)

[20] NIST, “CVE-2010-0928”, https://nvd.nist.gov/vuln/detail/CVE-2010-0928
[21] NIST, “CVE-2014-2107”, https://nvd.nist.gov/vuln/detail/CVE-2014-2107
[22] NIST, “CVE-2019-11165”, https://nvd.nist.gov/vuln/detail/CVE-2019-11165
[23] NIST, “CVE-2019-14604”, https://nvd.nist.gov/vuln/detail/CVE-2019-14604
[24] NIST, “CVE-2019-14625”, https://nvd.nist.gov/vuln/detail/CVE-2019-14625
[25] NIST, “CVE-2019-14626”, https://nvd.nist.gov/vuln/detail/CVE-2019-14626
[26] NIST, “CVE-2019-1649”, https://nvd.nist.gov/vuln/detail/CVE-2019-1649
[27] NIST, “CVE-2019-1700”, https://nvd.nist.gov/vuln/detail/CVE-2019-1700
[28] NIST, “CVE-2020-0574”, https://nvd.nist.gov/vuln/detail/CVE-2020-0574
[29] NIST, “CVE-2020-8737”, https://nvd.nist.gov/vuln/detail/CVE-2020-8737
[30] NIST, “CVE-2020-12312”, https://nvd.nist.gov/vuln/detail/CVE-2020-12312
[31] NIST, “CVE-2020-24485”, https://nvd.nist.gov/vuln/detail/CVE-2020-24485
[32] NIST, “CVE-2020-8684”, https://nvd.nist.gov/vuln/detail/CVE-2020-8684
[33] NIST, “CVE-2020-8737”, https://nvd.nist.gov/vuln/detail/CVE-2020-8737
[34] NIST, “CVE-2021-27208”, https://nvd.nist.gov/vuln/detail/CVE-2021-27208
[35] NIST, “CVE-2021-44850”, https://nvd.nist.gov/vuln/detail/CVE-2021-44850
[36] NIST, “CVE-2022-23822 ”, https://nvd.nist.gov/vuln/detail/CVE-2022-23822
[37] NIST, “CVE-2022-26512 ”, https://nvd.nist.gov/vuln/detail/CVE-2022-26512
[38] NIST, “CVE-2022-341572 ”, https://nvd.nist.gov/vuln/detail/CVE-2022-34157
[39] NIST, “CVE-2022-38787”, https://nvd.nist.gov/vuln/detail/CVE-2022-38787
[40] CWE, “Common Weakness Enumeration “, https://cwe.mitre.org/ last accessed 2024/10/03.
[41] EUROCAE, RTCA. "DO-254/ED-80–Design assurance guidance for airborne electronic hardware." *European Organisation for Civil Aviation Equipment and Radio Technical Commission for Aeronautics* (2000).
[42] Vadgaonkar, Prashant S., and Ullas Janardhan. *DO-254/ED-80-An Application Guidelines to Redesign/Re-Engineering Airborne Electronic Hardware*. No. 2016-01-2039. SAE Technical Paper, 2016.