



National Security Agency
Cybersecurity Technical Report

**DoD Microelectronics:
Field Programmable Gate Array
Level of Assurance 3 Best Practices**

June 2023

U/OO/170671-23

PP-23-1734

Version 1.0



This document was created through collaboration with each of the JFAC labs: National Security Agency (NSA), Air Force Research Lab (AFRL) RYDT, Naval Surface Warfare Center (NSWC) Crane, and Army Development Command (DEVCOM)/AVMC.

For additional information, guidance, or assistance with this document, please contact the Joint Federated Assurance Center (JFAC) at JFAC_HWA@radium.ncsc.mil.



Notices and history

Document change history

Date	Version	Description
JUN 2023	1.0	Initial Publication

Disclaimer of warranties and endorsement

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the United States Government, and this guidance shall not be used for advertising or product endorsement purposes.

Publication information

Author(s)

National Security Agency
Cybersecurity Directorate
Joint Federated Assurance Center

Contact information

NSA Joint Federated Assurance Center: JFAC_HWA@radium.ncsc.mil
Cybersecurity Report Feedback / General Cybersecurity Inquiries: CybersecurityReports@nsa.gov
Defense Industrial Base Inquiries and Cybersecurity Services: DIB_Defense@cyber.nsa.gov
Media inquiries / Press Desk: Media Relations, 443-634-0721, MediaRelations@nsa.gov

Purpose

This document was developed in furtherance of NSA's cybersecurity missions. This includes its responsibilities to identify and disseminate threats to National Security Systems, Department of Defense information systems, and the Defense Industrial Base, and to develop and issue cybersecurity specifications and mitigations. This information may be shared broadly to reach all appropriate stakeholders.



Executive summary

In support of securing Field Programmable Gate Array (FPGA) based systems from adversary influence during the manufacturing process, this report outlines the categories of relevant threats and the best practices for mitigating them at Level of Assurance 3 (LoA3). LoA3 captures the threats that are technically feasible but have high cost to implement, in addition to all LoA1 and LoA2 threats. This level is defined as causing extremely grave harm to U.S. personnel, property, or interests if the systems fail. At this level, these threats have the following characteristics:

- **Access** – Exploit multiple points of difficult access in different areas of the custom microelectronic components (CMC) supply chain.
- **Technology** – Feasible threats for which existing research indicates the likelihood that technology could be developed with an investment that would be feasible for a known adversary.
- **Investment** – A nation-state scale directed priority requiring resources from many specialties and organizations across a wide scope to facilitate an attack.
- **Value of effect** – Fully or partially degrading a system or feature.
- **Targetability** – Affect only a subset of systems.

Organized by threat, this report provides multiple technical mitigations to choose from to mitigate each threat and to allow the program the best fit for their program needs. The following table identifies the ten threat descriptions (TD) addressed by this guidance.

#	Threat description (TD)
TD 1	Adversary utilizes a known FPGA platform vulnerability
TD 2	Adversary inserts malicious counterfeit
TD 3	Adversary compromises application design cycle
TD 4	Adversary compromises system assembly, keying, or provisioning
TD 5	Adversary compromises third-party soft intellectual property (IP)
TD 6	Adversary swaps configuration file on target
TD 7	Adversary substitutes modified FPGA software design suite



#	Threat description (TD)
TD 8	Adversary modifies FPGA platform family at design
TD 9	Adversary compromises single-board computing system (SBCS)
TD 10	Adversary modifies vendor FPGA software design suite during development

Each subsection in this report contains mitigations described in detail to enable clear implementation. Secondary documents are referenced in cases where the suggested mitigation is highly detailed, specific to individual FPGA platforms, or subject to frequent change. In some cases, one hundred percent threat mitigation is not possible. The provided guidance adds additional layers of protections to increase the difficulty of malicious action. Additionally, the risks posed by the threat are explained. Appendix D: Checklists and data/documentation requirements contains a quick reference list of threats and associated mitigations.

Once the program has mitigated these threats, they have achieved an assurance level of LoA3.



Contents

DoD Microelectronics: Field Programmable Gate Array Level of Assurance 3 Best Practices	i
Executive summary	iv
Contents	vi
1 Overview of Level of Assurance 3 threats and mitigations	1
1.1 Complementary standards and guidance.....	4
1.2 Exclusions	5
1.3 Document use.....	6
1.4 General comments on mitigations	7
2 Threat descriptions (TD)	7
TD 1: Adversary utilizes a known FPGA platform vulnerability	7
TD 1 mitigations.....	8
TD 1 mitigation descriptions	8
Use caution when selecting tools or platforms.....	8
Use cleared personnel	8
Research vulnerabilities.....	8
Use revision control/version management.....	9
Enforce auditability	10
Enforce the approved design process	10
TD 2: Adversary inserts malicious counterfeit	11
TD 2 mitigations.....	13
TD 2 mitigation descriptions	13
Purchase from DoD authorized vendors and distributors	13
Consult GIDEP	13
Follow storage and shipping guidance	14
Verify the FPGA cryptographically secure identifier.....	14
Perform physical inspection/analysis	17
Cleared insider	20
TD 3: Adversary compromises application design cycle	21
TD 3 mitigations.....	22
Use Secret level cleared personnel	23
Track critical data in a revision control system.....	23
Enforce auditability	23
Use revision control/version management.....	24
TD 3.1 Mitigating the introduction of a compromised design into the application.....	24
Isolate and store the application design.....	25
Perform reproducible build	25
TD 3.2 Mitigating the modification of test benches or plans to reduce coverage or hide Trojan code.....	26
Execute a documented test plan.....	26
Validate and verify the test processes.....	27
Maintain test environment via configuration management.....	27



TD 3.3 Mitigating the introduction of a Trojan into the application design during development.....	27
Maintain bi-directional links to approved requirements.....	27
Enforce peer review	28
Execute a documented test plan.....	28
Implement, validate, and verify test processes	29
Select a formal “proof” process.....	29
TD 3.4 Mitigating the introduction of compromised tooling or software into the environment ..	30
Validate cryptographic hashes.....	30
Research vulnerabilities.....	30
Validate tools.....	31
TD 3.5 Mitigating intrusion into the internal network.....	32
Assign roles	33
Control and monitor access	33
Research vulnerabilities.....	33
Use a secret or classified network.....	34
TD 3.6 Mitigating risk from a compromised employee.....	34
Enforce auditability	34
Enforce the approved design process	35
Review critical design activities	35
Use cleared personnel	35
TD 3.7 Mitigating risk associated with the compromise of device identifiers.....	35
Store device identifiers.....	36
Limit access to device identifier information.....	36
TD 4: Adversary compromises system assembly, keying, or provisioning.....	36
TD 4 mitigations.....	37
TD 4 mitigation descriptions	38
Purchase from DoD authorized vendors and distributors	38
Follow storage and shipping guidance	38
Provide keys and configuration data	39
Clear memory devices.....	39
Provision private keys.....	39
Protect the configuration data package.....	39
Perform verification activities.....	39
Authenticate the FPGA device.....	40
TD 5: Adversary compromises third-party soft IP	41
TD 5 mitigations.....	41
TD 5 mitigation descriptions	42
Purchase from DoD authorized vendors and distributors	42
Only accept IP that is unobfuscated.....	42
Ensure IP deliverable packages are digitally signed.....	42
Validate the cryptographic hash	42
Store IP in a revision control repository.....	42
Examine IP for malicious functions	43
TD 6: Adversary swaps configuration file on target.....	43



TD 6 mitigations.....	44
TD 6 mitigation descriptions	45
Incorporate cryptographic authentication	45
Authenticate configuration data each time the data is loaded	45
Prevent direct read back.....	45
Use a CNSS/NIST approved algorithm and key length.....	45
Use DoD evaluated authentication mechanisms.....	45
Disable test access pins.....	46
Ensure authentication for modifications	46
Use a FIPS 140-2 compliant, Level 2 HSM	48
TD 7: Adversary substitutes modified FPGA software design suite.....	48
TD 7 mitigations.....	49
TD 7 mitigation descriptions	49
Purchase from DoD authorized vendors and distributors	49
Prevent automatic tool updates	49
Use a trusted computing environment.....	49
Use cleared personnel	50
Validate the cryptographic hash.....	50
Validate the tool output	50
TD 8: Adversary modifies FPGA platform family at design.....	51
TD 8 mitigations.....	52
TD 8 mitigation description.....	52
Engage JFAC.....	52
TD 9: Adversary compromises single-board computing system (SBCS)	53
TD 9 mitigations.....	53
TD 9 mitigation descriptions	54
Engage a DoD vendor to build the SBCS	54
Verification and authentication.....	54
Authenticate the FPGA devices.....	54
Verify the SBCS configuration process	54
Test non-volatile memory.....	55
Document the steps	55
TD 10: Adversary modifies vendor FPGA software design suite during development	55
TD 10 mitigations	56
TD 10 mitigation descriptions.....	56
Perform logical equivalency checking.....	56
3 Summary.....	57
Appendix A: Standardized terminology.....	58
Appendix B: IP Reuse Guidance.....	61
Reuse conditions.....	61
Reuse scenarios.....	62
Appendix C: JFAC FPGA reporting template.....	65
Appendix D: Mitigations and data/documentation requirements.....	69
Checklist for TD 1: Adversary utilizes a known FPGA platform vulnerability.....	69



Checklist for TD 2: Adversary inserts malicious counterfeit.....71
 Checklist for TD 3: Adversary compromises application design cycle 74
 Checklist for TD 4: Adversary compromises system assembly, keying, or provisioning 83
 Checklist for TD 5: Adversary compromises third-party soft IP..... 85
 Ensure IP deliverable packages are digitally signed.....86
 Checklist for TD 6: Adversary swaps configuration file on target 87
 Checklist for TD 7: Adversary substitutes modified FPGA software design suite..... 88
 Checklist for TD 8: Adversary modifies FPGA platform family at design 90
 Checklist for TD 9: Adversary compromises single-board computing system (SBCS) 90
 Checklist for TD 10: Adversary modifies vendor FPGA software design suite during
 development..... 92

Tables

Table 1: LoA3 threats..... 3
Table 2: List of AS6171 slash sheets..... 17



1 Overview of Level of Assurance 3 threats and mitigations



This document provides JFAC's recommended hardware assurance strategies for Field Programmable Gate Array (FPGA) devices. The guidance outlined by this document provides hardware assurance to systems requiring Level of Assurance 3 (LoA3). Additionally, it provides the requisite strategies and details for implementing each threat mitigation. Secondary documents are referenced in cases where the suggested mitigation is highly detailed, specific to individual FPGA platforms, or subject to frequent change.

This guidance is meant to stand on its own and not require the participation of JFAC in the development process of a program's product, unless required by a specific mitigation. However, JFAC does remain at the ready to aid programs who seek to better understand this guidance, to incorporate a program specific mitigation or are seeking alternatives to the guidance contained herein. For further information or support, please visit the JFAC portal at <https://jfac.navy.mil>.

In addition, to threats and mitigations identified at LoA1 and LoA2, LoA3 requires mitigations against FPGA assurance threats that have the following characteristics:

Access – Multiple points of difficult access in different areas of the custom microelectronic components (CMC) supply chain.

- This could include multiple people working on different elements of the CMC or government design teams.
- This could include multiple people performing different functions in the fabrication process.
- This could include single or multiple cleared insiders working on the same or different parts of the supply chain.

For a mitigation based on access to be effective, it needs to make it considerably more difficult to carry out the attack. Examples include necessitating multiple points of difficult access via many cleared people in conjunction with attacking multiple areas of the supply chain such that actors will need coordination and communication amongst the group.



Technology – *Technologically feasible* threats for which existing research indicates the likelihood that technology could be developed with an investment that would be feasible for a known adversary. However, these threats may not be associated with existing and/or known tools and may not have associated reporting indicating adversary activity. Moreover, while all the threats are validated to be possible, it may be that there is no known or ongoing investment in the capability.

For a mitigation based on technological complexity to be effective, it must increase the level of technology needed to carry out the attack to that which is beyond what is recognized as technically feasible and practical. This includes areas for which there is no known research.

Investment – *A nation-state scale directed priority* refers to a substantive program conducted by a nation state that coordinates resources from many specialties and organizations across a wide scope to facilitate an attack.

For a mitigation based on investment of resources to be effective, it must force the attacker to expend greater resources that would be daunting even for a nation-state.

Value of Effect – *Degrade system performance* are those effects that reduce the behavior of a system without fully disabling any specific feature or reliably having a specific planned effect. Note, that the term degradation may be used in some domains in a different way. For instance, a communications link might be “degraded” in a way that prevents all communication. Such an attack would fall under disabling a capability for the purposes of this evaluation. In addition, this LoA must consider all higher value effects described in LoA1 and LoA2.

For a mitigation based upon value of effect to the adversary to be effective in LoA3, it must eliminate or substantially reduce the value to the attacker.

Targetability – *Blind attacks* refers to attacks that impact large numbers of parts, whole device families, or users in a way that has a significant likelihood of discovery without effort, but only to impact a specifically targeted part. Blind attacks are those where it is hard to predict the interaction between what adversaries do and the intended consequence of the attack. This could include attacks that are performed against far more targets than expected, or with an intelligent agent that acts without an outside trigger or without foreknowledge of the attack outcome that would inform the adversary



of its execution. These attacks can also include activation times that cannot be controlled once fielded; that is to say, a pre-determined time at which devices will fail.

For a mitigation based on targetability to be effective, it must remove the ability of the adversary to affect targeted systems and force the adversary to rely on general and blind attacks.

For a program to achieve Level of Assurance 3, it must provide mitigations against threats that possess these characteristics. Of prime importance in LoA3 is the assumed presence of one or more compromised cleared insiders and allowance for attacks that are not targeted, but broadly applied to the entire supply chain. These new conditions render classified facilities and cleared people ineffective as a “sole” means of mitigation. As such, many of the mitigations offered in this guide focus on nullifying this adversarial advantage using dual or independent teams. LoA3 addresses threats that originate from an adversary whose intent is malicious, but unlike the previous LoA levels also includes cases where reliability is also compromised. These threats should be addressed by the reliability testing of a program. For programs with stringent or specific reliability requirements, it is strongly recommended that the appropriate level of testing be conducted to ensure the proper operation of the product rather than relying on assurance mitigations.

The following table lists the ten FPGA threats that are addressed by LoA3. Each threat is explained and accompanied by examples in more detail within the JFAC *FPGA Best Practices – Threat Catalog*.

Table 1: LoA3 threats

#	Threat description (TD)
TD 1	Adversary utilizes a known FPGA platform vulnerability
TD 2	Adversary inserts malicious counterfeit
TD 3	Adversary compromises application design cycle
TD 4	Adversary compromises system assembly, keying, or provisioning
TD 5	Adversary compromises third-party soft intellectual property (IP)
TD 6	Adversary swaps configuration file on target



#	Threat description (TD)
TD 7	Adversary substitutes modified FPGA software design suite
TD 8	Adversary modifies FPGA platform family at design
TD 9	Adversary compromises single-board computing system (SBCS)
TD 10	Adversary modifies vendor FPGA software design suite during development

Each threat listed here has corresponding mitigations. These mitigations are derived from various commercial/government standards and existing best practices. The use of these standards/best practices should not preclude the use of any other standards or best practices. In particular, DoD projects identified as National Security Systems (NSS) should also utilize the appropriate guidance as required by the Committee on National Security Systems (CNSS) Policy 15 and other CNSS documents.

1.1 Complementary standards and guidance

Microelectronic quantifiable assurance (MQA) standards are intended to be complementary to other government and industry recognized risk management practices and standards. The following are standards for various mitigations:

- CNSS Policy on the use of Commercial Solutions to Protect National Security Systems Policy 7
- CNSS Cryptographic Key Protection Policy 30
- National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS) Publication 186 Digital Signature Standard
- NIST FIPS Publication 198 The Keyed-Hash Message Authentication Code (HMAC)
- NIST Special Publication (SP) 800-53 Security and Privacy Controls for Federal Information Systems and Organizations
- NIST SP 800-57 Recommendation for Key Management
- The Department of Defense Cybersecurity Maturity Model Certification (CMMC)
- The Configuration Management section of NIST SP 800-60 Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems



- NIST SP 800-171 Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations
- NIST SP 800-172 Enhanced Security Requirements for Protecting Controlled Unclassified Information.
- SAE International AS6171 Test Methods Standard; General Requirements, Suspect/Counterfeit, Electrical, Electronic and Electromechanical Parts
- Trusted Systems and Network (TSN) Analysis
- Defense Acquisition Guidebook Chapter Nine – Program Protection Plan
- JFAC FPGA Best Practices Documents – contact JFAC for available documents to support implementation practices for the FPGA standards in this guide

Program offices should review and adhere to the standards provided in each document, as applicable. The standards and guidance contained in this best practice guide do not supersede any other DoD acquisition requirement or other DoD mandate. Additionally, programs are encouraged to apply applicable standards in addition to the standards described in this document

1.2 Exclusions

This FPGA Level of Assurance 3 Best Practice guide does **not** address the following concerns:

- Non-malicious and profit driven reliability risks such as re-marked parts. Programs are responsible for establishing and enforcing system reliability requirements. This document will not include guidance on how to conduct reliability testing. However, compliance with SAE International AS6171 Test Methods Standard: General Requirements Suspect/Counterfeit, Electrical, Electronic and Electromechanical Parts as recommended by this report is an effective detection mechanism for these kinds of counterfeit parts.
- Threats to the confidentiality of the application design. The program application can be loaded apart from the manufacturing process and under the protection and oversight of the program. Confidentiality is preserved using existing engineering practices, bitstream encryption and other anti-tamper practices. For more guidance in this area, see the DoD's Anti-tamper Executive Agent (<https://at.dod.mil>).



1.3 Document use

These FPGA assurance best practices instruct programs on protecting manufacturing and provisioning processes from adversarial influence. Specifically, they apply to the manufacturing, acquisition, programming and first attachment of the FPGA devices. The program must define its own protection methods as boards become integrated into subcomponents, components, and then final systems.

For LoA3 compliance, each program should perform each mitigation listed in the “TD Mitigation” section. The “Mitigations Description” section provides details for each mitigation. Underlined text in a listing indicates that there is a following section providing full details for implementing the protection. In some cases, the full description contains multiple technical options for mitigating the threat to be LoA3 compliant. An asterisk “*” next to any mitigation indicates that multiple technical options exist. In those cases, at least one option must be implemented.

When mitigations for all the threats listed under LoA3 are completed, that device can be said to have achieved LoA3. However, compliance with LoA3 can be impacted by changes in several areas during the system’s life.

The Program Protection Plan (PPP) emphasizes the need to maintain and update protection measures throughout lifecycle of a program. It is strongly recommended that each program identify events that would trigger a review of the PPP and the hardware assurance practices after fielding. These events should include but not be limited to:

- Changes to the system
- Changes to the supplier of critical components including the FPGA devices
- Changes to the FPGA design software (new releases, fixes, etc.),
- Changes to the threat environment
- Revelations of new vulnerabilities to the FPGA devices

The PPP documents list resources with which the program can track the latest available intelligence on threats and supply chain vulnerabilities. Changes in any of these areas should trigger a review of the most up-to-date assurance mitigations against the triggering event. If threats or vulnerabilities threaten the system, new mitigations should be implemented to remain compliant to LoA3. Absent any changes in these areas, the devices should be considered to have achieved LoA3.



1.4 General comments on mitigations

- Programs are encouraged to own as much of the assembly process as possible and avoid third parties to the fullest extent possible.
- Programs are encouraged to diversify their supply sources to minimize malicious targeting.
- Programs are encouraged to use cleared personnel and classified resources to the fullest extent possible.
- Programs are encouraged to use verification of all manufacturing steps to the fullest extent possible. This applies to packaging and assembly.

2 Threat descriptions (TD)

TD 1: Adversary utilizes a known FPGA platform vulnerability

In this threat, an adversary utilizes a vulnerability in an FPGA platform or vendor development software package to initiate an attack. At LoA3, a vulnerability is defined as a weakness known to the adversary in the design of a specific FPGA platform or software program that would allow the ability to use it for malicious purposes. The vulnerability could be publicly or non-publicly known.

Vulnerabilities could allow for the leakage of sensitive information or keys, compromise of security or tamper detection functions, or unauthorized reconfiguration of the product. Unclassified and public vulnerabilities are published in databases, such as the DISA Vulnerability Management System (VMS), “Common Vulnerabilities and Exposures (CVE)”, and the “National Vulnerabilities Database (NVD)”, vendor advisories, errata bulletins, etc. Non-public vulnerabilities refer to ones that have been discovered by the adversary's research or known by vendors but not exposed to the public.

This threat can be realized when a program does not perform vulnerability research or an insider hides the fact of the vulnerability such that it may be used for nefarious purposes or by adding/modifying design features for use with or for triggering the vulnerability.



TD 1 mitigations

- Use caution when selecting tools or platforms. When possible do not select tools or platforms that are end-of-life or beta/initial releases. Also, ensure previously identified vulnerabilities in tools/platforms have been adequately addressed in newer releases.
- Use cleared personnel that possess at least a Secret level clearance.
- *Research vulnerabilities affecting tools/platforms.
- Use revision control/version management that includes document/data control, document/data release, backups, and archives, refresh of backup media, retention of tools and software, test equipment and test environment.
- Enforce auditability of the requirements, architecture, design, code, tests, bugs, and fixes. At a minimum, audit data should include what decisions were made, by whom, for what reason, and on what date.
- Adopt, document, and enforce the approved design process that is organizationally approved and with clear entry and exit criteria. Entry and exit criteria incorporate peer reviews and technical reviews with management approval to exit a phase.

TD 1 mitigation descriptions

Use caution when selecting tools or platforms

Consider the longevity of selected tools and FPGA platforms. Newly released devices may not yet have a vulnerability history. Programs should proceed with caution when using newly released devices or tools. End-of-life devices may not have support to mitigate vulnerabilities once identified.

Use cleared personnel

Use personnel with at least a Secret clearance to perform designated work. Designated work could include design reviews, peer reviews, vulnerability research, validation, and verification activities, etc.

Research vulnerabilities

Research the respective FPGA platform and software for existing vulnerabilities in databases such as:

- Common Vulnerabilities and Exposures (CVE) – <https://cve.mitre.org>



- NIST National Vulnerabilities Database (NVD) – <https://nvd.nist.gov>
- Government Industry Data Exchange Program (GIDEP) – <https://www.gidep.org/products/products.htm>
- DISA Security Technical Implementation Guides (STIGs) – <https://public.cyber.mil/stigs/>
- Search vendor advisories, errata, publications, and academic papers detailing vulnerabilities in the device in question.
- Contact the vendor field application engineer for unreleased or pre-release vulnerability reports.

If vulnerabilities are found in the FPGA device, **choose one** of the following options:

Option 1: Select a different FPGA platform device or software that does not have published vulnerabilities and that meets the program requirements.

Option 2: Use standard formal processes and procedures to work with the vendor to resolve the vulnerability. Once a fix is identified, only accept formal releases, do not accept custom beta fixes, custom patches, etc. for incorporation; or

Option 3: The program can internally determine the vulnerability poses no significant risk to their product. JFAC is available to provide assistance in assessing the risk that the vulnerability poses to the system and acquire recommended mitigations for a particular vulnerability.

Note: If a vulnerability is identified, JFAC recommends reporting it to DISA and to contact the vendor so they may correct it.

Use revision control/version management

To prevent vulnerable software from being loaded into the environment, it is important that robust configuration management and revision control systems are in place. All changes to the system and/or any artifacts should be documented, approved, and auditable.

These systems should fulfill the following requirements:

- Allow only authorized system administrators to make changes to the underlying revision control tool and underlying server.



- Use a backup system that syncs to the primary and is maintained by a separate administrator. Each system should be managed by separate system administrators.
- Enforce administrative restrictions; restrict privileged access to authorized personnel only; limit what users can do to the database; ensure all users are verified; encrypt database information—both in transit and at rest; enforce secure passwords; enforce role-based access control and privileges; and remove unused accounts.
- Remove any components or functions that are not necessary (for example, remove all sample files and default passwords).
- Ensure the system provides a complete and immutable, long-term change history of every file. The system must log every change made by individuals. This includes changes such as creating and deleting files and editing content. The history must identify the person who made the change, what was changed, the date of the change, and the purpose of the change.
- Ensure the system stores a reliable copy of assets that are currently in production.
- Ensure the system stores reliable copies of previous production versions of assets, allowing for the complete retrieval of those versions.
- Ensure password best practices (password rotation, length, etc.) are enforced. In lieu of a password, two-factor authentication can be utilized.
- Ensure the final application synthesis and bitstream generation configuration settings are captured and stored.
- All changes to the system and/or any artifacts should be documented, approved, and auditable.

Enforce auditability

Enforce auditability of the requirements, architecture, design, code, tests, bugs, and fixes. At a minimum, audit data includes what decisions were made, by whom, for what reason, and on what date. System audits and logs are required where applicable.

Enforce the approved design process

The design process should include the identification of all assurance critical activities and highlight how each activity will be reviewed. The design process should ensure the design is reviewed by multiple cleared individuals. The original designer should not be the responsible party for performing the review. The cleared reviewers should assess



the satisfaction of all requirements, ensure no extraneous design elements, and review all vulnerability activities, including identification of vulnerabilities and the appropriateness of the mitigations. Additionally, the design process should contain clear entry and exit criteria that incorporate peer reviews and technical reviews with management approval required to exit a phase.

TD 2: Adversary inserts malicious counterfeit

LoA1 addresses counterfeit parts made in an unauthorized fabrication facility and inserted into the supply chain. These parts mimic the behavior of the target device, but are manufactured in a process differing from the authorized one. Insertion of counterfeit parts can happen during any part of the device's lifecycle. This includes prior to purchase, in transit, while in storage by the program, during assembly, and at distribution prior to fielding.

In addition to the LoA1 threats, LoA2 addresses counterfeit parts made in an authorized fabrication facility through the malicious compromise of the manufacturing process. Such an attack could happen during any of the following phases of the process:

- Transfer of graphic design system 2 (GDSII) mask data
- Mask fabrication
- Mask storage
- Wafer manufacturing
- Wafer testing
- Wafer dicing and packaging
- Package testing
- Device personalization

LoA2 also includes counterfeit parts created in an adversary facility purposely built to mimic the authorized device manufacturing process, as well as the insertion of a malicious function into the package of an authentic device. This includes:

- insertion of a snooping die stacked in the package,
- introduction of a kill switch in the package, or
- alteration of the bond out to compromise some FPGA feature.

LoA3 adds counterfeit parts fabricated in the authorized fabrication facility using stolen authorized GDSII under a different product name. It can also include the introduction of a reliability and performance degradation due to an attack on the manufacturing process or the remarking of used devices. The modification of the original design to insert a malicious function could be considered a “counterfeit” part, but will be addressed



separately in TD 8: Adversary modifies FPGA platform family at design. The difference between this new threat in TD 2 in LoA3 and TD 8 is that, for this threat, there exists a golden, unaltered (known good) representation of the design in the GDSII. For TD 8, the malicious function is baked into the design and cannot be exposed by any comparison to a golden model.

The mitigations at LoA3 for this threat rely heavily on the physical inspection of the parts. This reliance requires the differentiation between counterfeits from the authorized fabrication facility and counterfeits from an unauthorized fabrication facility as different types of malicious counterfeits. Physical inspections are more intensive for detection of a counterfeit device from an authorized fabrication facility.

Commercial (non-malicious) counterfeits, such as re-marked parts, may represent a reliability risk. Programs with specific reliability requirements should plan for the appropriate level of testing to verify that their design and components meet those goals. This document will not provide the details for performing reliability testing as they differ for each program. The program should perform sampled reliability testing against the standards claimed by the manufacturer or those needed by the program.

Additionally, at LoA3 there is the assumption of the existence of one or more compromised cleared insiders in the program and the presumption of an adversary achieving difficult points of access. The insider(s) may be used by the adversary to introduce malicious features during any portion of the product manufacturing cycle and/or compromise a portion of the FPGA device verification process. Compromised cleared insiders may be used to introduce counterfeit parts into the program supply chain or to compromise the program's acceptance testing. Overlapping checks are therefore necessary for each threat commensurate to this level of assurance.

JFAC relies on substantial physical device inspection to address these threats because the program has no positive control over the fabrication facility or its processes. Most of the FPGA fabrication facilities are foreign owned and not controllable by the program or DoD. JFAC can identify numerous "technically" feasible attacks for all fabrication countermeasures considered. Overlapping personnel and multi-party review in the verification process along with cryptographically protected IDs and reliability testing of sampled devices provides additional assurance protections.



Guidelines for conducting physical inspection are provided by the SAE AS6171 counterfeit detection standard. These guidelines are organized into “slash sheets.” Each slash sheet is a description of a singular type of inspection process. For the purposes of this document, the slash sheets may be divided into several purposes:

- Slash sheets 2-10: describe physical inspections able to identify devices that were manufactured in an unauthorized fab.
- Slash sheets 11: describes physical inspections able to identify maliciously altered devices that were manufactured in an authorized fab.
- Slash sheets 3, 4, 6, and 10: describe physical inspections intended to uncover malicious alterations made to the package internals of an authentic device.

More details regarding the physical inspection process are outlined below in the mitigations.

TD 2 mitigations

- Purchase from DoD authorized vendors and distributors.
- Consult GIDEP and follow their guidance on counterfeit risk mitigation, including guidance on known counterfeit parts. The program should use this information to inform their physical analysis efforts.
- Follow storage and shipping guidance when storing and transferring FPGA devices between locations.
- Verify the FPGA cryptographically secure identifier (ID) against information sent by the vendor (not the authorized distributor).
- Using the latest approved version of AS6171 with associated slash sheets perform physical inspection/analysis on a sampling of random devices to detect counterfeit parts.
- Mitigate risk of a cleared insider involved in the physical inspection process.

TD 2 mitigation descriptions

Purchase from DoD authorized vendors and distributors

Ensure devices are purchased from vendors and distributors authorized by DoD.

Consult GIDEP

GIDEP provides technical data compiled by government and industry regarding counterfeit hardware devices to be used for system design, development, production,



and logistics support processes. This information contains counterfeit risk mitigations and physical analysis results.

Follow storage and shipping guidance

The program should document, maintain and enforce both device storage and shipping procedures. Minimally the plan should enforce the verification of all devices upon receipt. Once verification has taken place production devices should be stored and maintained in a restricted area separate from non-production devices (design, test, etc.). Production devices should be continuously tracked to include arrival of the device by unique identifier, interaction anyone has with the device, and exit of the device from inventory. The restricted area should enforce access control that limits access to only a minimum subset of people that require access to support direct job responsibilities and excludes all members of the design team. The restricted area should have a clearly defined perimeter, but physical barriers are not required. Personnel within the area are responsible for challenging all persons who may lack appropriate access authority. The restricted area access should be audited to include data containing who entered/exited the area, with a timestamp and reason for entry.

Shipping should be controlled and managed. JFAC recommends shipping material using a commercial carrier that has been approved by the CSA to transport Secret shipments, although the material is not Secret. Commercial carriers may be used only within and between the 48 contiguous States and the District of Columbia or wholly within Alaska, Hawaii, Puerto Rico, or a U.S. possession or trust territory. When shipping using a commercial carrier take efforts to afford additional protection against pilferage, theft, and compromise as follows. This includes using hardened containers unless specifically authorized otherwise and ensuring the packages are sealed. The seals should be numbered and the numbers indicated on all copies of the bill of lading (BL). When seals are used, the BL shall be annotated substantially as follows: DO NOT BREAK SEALS EXCEPT IN CASE OF EMERGENCY OR UPON PRIOR AUTHORITY OF THE CONSIGNOR OR CONSIGNEE. IF FOUND BROKEN OR IF BROKEN FOR EMERGENCY REASONS, APPLY CARRIER'S SEALS AS SOON AS POSSIBLE AND IMMEDIATELY NOTIFY BOTH THE CONSIGNOR AND THE CONSIGNEE.

Verify the FPGA cryptographically secure identifier

For LoA3, the program should utilize an FPGA device that incorporates a cryptographically protected ID that can be verified against information sent by the



vendor (not the authorized distributor). The use and verification of this type of device ID mitigates the counterfeit parts made in an existing, non-authorized fabrication facility sub-threat.

While the specifics of each FPGA vendor and platform vary, many newer FPGA platforms contain this type of anti-counterfeiting feature. When these features are sufficiently secure, such mechanisms provide an extremely cost-effective method to detect counterfeits both at acquisition and throughout the FPGA device's lifecycle in a system. The biggest two advantages of such techniques are the ability to validate a device remotely and the ability to non-destructively re-evaluate a device at any time.

By contrast to physical anti-counterfeiting techniques, properly implemented cryptographically secure identifiers do not require destructive analysis for verification. A typical scheme could validate such a device simply by placing it in a socket. A design can facilitate access to the identifier through local access, such as a board header, or remotely. Depending on the exact mitigations selected, this potentially saves two distinct destructive steps: one at acquisition of the devices and one after assembly of the PCB.

For device families that do not offer a cryptographically secure ID, a soft physically unclonable function (PUF) should be used by the program for device authentication throughout the manufacturing and lifecycle of the device. In this case, a soft PUF is configured into each device to produce a unique identifier. This ID is then stored in association with the physical package serial number. The PUF is then removed. It can be reconfigured into the device at any time to retrieve the ID to validate the authenticity of the device at any later date. The PUF should be added and the PUF ID recorded immediately after validating the device lot as authentic. The step can then be repeated after the component has been out of the control of the program to verify the devices. The program should not proceed to manufacture or field LoA3 devices without one of these ID services. Additionally, the PUF code must be protected as critical data in a Secret level repository with strong access controls. The PUF signature should be verified before each use with a hash value.

This kind of validation is where details matter. At the same time, each FPGA vendor offers a unique approach, and each FPGA platform offers a unique variation. In no case is a fully readable ID acceptable. Instead, these schemes all detail cases where the



device possesses a specific private cryptographic key. The device ID in this scheme can be cloned only if an adversary is able to get access to that private key. Regardless of the specific platform used, the public keys/identifiers of the devices being authenticated must be delivered and maintained in a secure way. For delivery, the vendor must provide this information to the program using a CNSS or NIST approved authentication algorithm to transmit the data. Examples would be an ECC-signed email with a verified certificate or an https-based file distribution system using a verified certificate. Once received, the integrity of that list must be maintained by storing it as critical data in a Secret level repository with strong access controls.

Remote attestation is an additional advantage enabled by a cryptographically secure ID. While remote attestation cannot be used during acquisition and assembly due to the potential introduction of additional vulnerabilities, it can be used throughout the rest of the lifecycle of the device. This provides the possibility of a future where devices and their configurations can be validated and monitored remotely. Capabilities for remote attestation of hardware, firmware, and software are currently being developed in the cybersecurity space as enterprise management tools. While their use is not yet fully widespread in hardware development, inclusion of these features is a potential growth area for the lifecycle hardware assurance of FPGA devices.

Remote attestation is a powerful and valuable technique and JFAC can consult on appropriate remote attestation schemes, potentially based on these same mechanisms. However, the initial counterfeit screening must be done locally, validating each specific device.

This section describes at the highest level the specific criteria that is required for an appropriate device ID to support anti-counterfeiting.

- Cryptographically protected IDs must utilize a CNSS Policy compliant private asymmetric key for which no read function exists. If CNSS is not a program requirement, the program should use a CNSS or NIST approved asymmetric authentication algorithm.
- The provenance of the key must be understood in detail.

The device must be able to authenticate a nonce using this key. Each device's ID must be authenticated by the public vendor-provided key through decryption of the nonce.



Perform physical inspection/analysis

Perform physical analysis on a sampling of random devices to detect counterfeit parts. This analysis applies specific, industry standard counterfeit inspection techniques, including package analysis, x-ray of the part, and examination of the die with comparisons against FPGA vendor provided golden samples. This physical analysis is intended to catch parts that have been remarked or contain counterfeit die. The details of what steps to conduct in the analysis and recommendations on how to execute them are contained in the commercial standard document, SAE AS6171. To reduce personnel threats, these inspections should be carried out by cleared personnel at a Secret level or higher.

At LoA2 and LoA3, there are additional attacks introduced under the counterfeit threat:

- Insertion of a malicious function into the package of an authentic device
- Counterfeit parts made in an authorized fabrication facility

It is due to these new threats that physical inspection is required in all cases. In LoA1, cryptographically secure IDs were sufficient to address the counterfeit threat. However, this would not be sufficient at LoA2 or LoA3 since these IDs would not preclude the insertion of a malicious function into the package of a device nor identify devices where malicious features were added to the die during manufacturing.

Physical analysis is a sequence of device analysis steps, from least destructive to most destructive, designed to ensure that the part in question is authentic. If a device fails a given step, it is not authentic and there is no need to complete further steps. If all steps are completed and the device passes, it is likely authentic, with likelihood commensurate with the amount of effort it would take to get a counterfeit device to pass these tests, and the fact that the device in question is subject to LoA3. Each AS6171 test is detailed in a separate document called a “slash sheet”. Listed below are the slash sheets that comprise the standard.

Table 2: List of AS6171 slash sheets

Test Number	Description
AS6171	Test Methods Standard; General Requirements, Suspect/Counterfeit, Electrical, Electronic, and Electromechanical Parts



Test Number	Description
AS6171/1	Suspect/Counterfeit Test Evaluation Method
AS6171/2	Techniques for Suspect/Counterfeit EEE Parts Detection by External Visual Inspection, Remarking and Resurfacing, and Surface Texture Analysis Test Methods
AS6171/3	Techniques for Suspect/Counterfeit EEE Parts Detection by X-ray Fluorescence Test Methods
AS6171/4	Techniques for Suspect/Counterfeit EEE Parts Detection by Delid/Decapsulation Physical Analysis Test Methods
AS6171/5	Techniques for Suspect/Counterfeit EEE Parts Detection by Radiological Test Methods
AS6171/6	Techniques for Suspect/Counterfeit EEE Parts Detection by Acoustic Microscopy (AM) Test Methods
AS6171/7	Techniques for Suspect/Counterfeit EEE Parts Detection by Electrical Test Methods
AS6171/8	Techniques for Suspect/Counterfeit EEE Parts Detection by Raman Spectroscopy Test Methods
AS6171/9	Techniques for Suspect/Counterfeit EEE Parts Detection by Fourier Transform Infrared Spectroscopy (FTIR) Test Methods
AS6171/10	Techniques for Suspect/Counterfeit EEE Parts Detection by Thermogravimetric Analysis (TGA) Test Methods
AS6171/11	Techniques for Suspect/Counterfeit EEE Parts Detection by Design Recovery Test Methods

For the purposes of LoA3, the program should follow the lot sampling guidelines found in the latest version of AS6171 and exercise the tests defined by slash sheets 1-11. Sheets 1-10 should uncover a counterfeit fabricated in an unauthorized fabrication facility or a malicious package insert. Sheet 11 should uncover a counterfeit fabricated in the authorized fabrication facility.

Here are the physical analysis steps that should be taken:

- If the device family possesses cryptographically protected IDs:
 - Perform slash sheets 2 and 3 that incorporate visual inspection and 3D x-ray. This effort focuses on analyzing the parts for a malicious additive



inserted inside the package. The number of parts sampled should be guided by the sampling standard found in slash sheet 1.

- If the device family does not possess cryptographically protected IDs:
 - Perform slash sheets 2-10 for the purposes of detecting an in-package malicious insert and a die manufactured in an unauthorized facility.
 - Perform the steps outlined below as they relate to slash sheet 11 to identify malicious functions added to the die during manufacture in an authorized facility. This test may be limited to a single device.

Sheet 11 is a set of instructions for performing a full delayering, imaging of die, and comparison against the vendor provided GDSII or an exemplar device. This analysis exposes the FPGA die manufactured layers for comparison against a golden model made up of either vendor provided images/GDSII or an exemplar part. An exemplar part is one that is obtained directly from the vendor and not from an authorized distributor. For LoA3, the program must perform the following reverse engineering comparison on a single part:

Full chip delayering – imaging and comparison of all layers to the exemplar when the state of the art capability allows it. This is the ideal option for detecting malicious changes. In the case of FPGA multi-chip modules (MCM), all the dies should be examined using this technique. Special care should be taken to validate the internal packaging connections.

Full backside delayering – imaging and comparison of layers active, poly, contact, and metal 1 (M1). This is the ideal option for detecting malicious changes when a state-of-the-art lab capability is not sufficient. In the case of FPGA multi-chip modules (MCM), all the dies should be examined using this technique. Special care should be taken to validate the internal packaging connections.

Forward the results of the examination to JFAC with information regarding the FPGA type and lot. The results should include a description of the verification method and the coordinates of the windows opened for evaluation. JFAC will compile this information over time to develop better insight into malicious attacks on the manufacturing process.

Contact JFAC for guidance when process geometries are beyond the state-of-the-art in reverse engineering.



Reliability testing should be conducted on sampled parts using the same sampling guidelines provided by SAE AS6171. This testing should ensure that the devices meet the reliability requirements of the program. Do not assume compliance based upon device datasheet information.

Cleared insider

To mitigate the risk of a cleared insider compromising the physical analysis process, programs should:

- Select sample parts bound for physical inspection in ways that specifically defeat insider compromise.
- Create cryptographically protected IDs post verification.
- *Verify independent lab work using overlapping personnel and multi-party review.
- Follow the mitigation guidance in *TD 4: Adversary compromises system assembly, keying, or provisioning.*

Select sample parts

The selection of parts to be physically sampled must be handled in such a way that a compromised cleared insider could not just select good parts to be sampled. Possible options include the following:

- Multiple independent parties handle part selection before shipping, and they should physically verify that the parts selected make it all the way to the physical inspection processes.
- An independent party verifies sampling before shipping, and multiple parties verify upon receipt that the right parts were received.
- Use a non-human random selection automated process for sampling.
- All physical verification and sampling work should be conducted by personnel holding clearances of at least the Secret level and carried out in facilities cleared to at least the Secret level.

Create cryptographically protected IDs post verification

Following physical verification above, JFAC recommends using soft PUFs to protect the authentic parts from being swapped out for modified devices during the subsequent program manufacturing process:

- Load the soft PUF into the fabric and generate a unique ID for the FPGA die.
- Record the device serial number and PUF ID.



- Erase the soft PUF.

At any time during the lifecycle of the FPGA device, the soft PUF can be reloaded and the unique ID can be extracted and compared against the expected value for confirmation of the authenticity of the part.

Verify independent lab work

There is a need to check the lab performing the physical verification for compromised results. If a compromised program insider is working with a compromised lab to pass counterfeit parts off as good, the compromised lab could throw away all the devices submitted for examination and simply create reports and photos of an exemplary device, or they could do all the work but falsify the reports.

This threat is not completely mitigated with the following steps, but these steps increase the difficulty of returning false reports:

- Insist on the return of sampled materials and detailed reports after evaluation. This serves as a check that the lab did the work and serves as an additional means to verify that sampling guidelines were followed.
- Require lock and key storage of all parts to be physically inspected and whether that inspection is done by the program or by independent lab(s).

Additionally, **choose one** of the following:

Option 1: Insert known bad parts into the samples to be physically verified. Track which parts those are using custom bad data and/or markings. If the independent lab does not report those parts as bad, then either they, or who they are reporting bad parts to, or both, may be compromised.

Option 2: Use two labs, use an independent expert observer, or both. This creates a check against the lab being compromised.

Option 3: Perform any physical inspections done by the program, rather than an independent lab, with two-person authentication, or duplicate them independently, or both.

TD 3: Adversary compromises application design cycle

In this threat, a compromised insider has access to the design process and data related to an FPGA application development effort. This insider can use their access to modify



design code or design constraints, change FPGA configuration settings, or swap in a distinct configuration file that is authenticated and built with the same tools and keys being used by the design team. The actor is in a particularly advantageous position because they can modify the product during any phase of the design process. This same threat surface may also be attacked via remote network intrusion. An attacker with network access may also be able to modify important design data in a way that introduces a Trojan or other nefarious functions.

At LoA3, it is assumed that multiple cleared and uncleared individuals may be the adversarial actor. The uncleared people can have different positions within the supply chain. The actors could be working independently or with each other. In this threat the compromised insider has access to the design process and data related to an FPGA application development effort.

TD 3 is comprised of several specific scenarios. These scenarios describe the entire threat at TD 3 and each of the mitigations for each scenario should be implemented. The specific scenarios are as follows:

- Introduction of a compromised design into the application,
- Modification of test benches or plans to reduce coverage or hide Trojan code,
- Introduction of a Trojan into the application design during development,
- Introduction of compromised tooling or software into the environment,
- Intrusion into the internal network,
- Compromised employee,
- Modification of the revision control system to hide malicious code or test bench modifications (associated mitigations are captured in the “in all cases” section below),
- Introduction of modified configuration data after generation (associated mitigations are captured in the “in all cases” section below),
- Compromise of device identifiers.

TD 3 mitigations

The best practices presented here do not constitute a standalone FPGA design flow, but rather should be integrated into the existing design procedures. These assurance practices incorporate industry accepted design best practices with emphasis on documented and approved design, review, and test procedures.



The following set of mitigations apply to all TD3 scenarios, in addition to mitigations identified in the individual scenario sections.

In all cases mitigations

- Use Secret level cleared personnel. If the program has higher level clearance requirements, the program's requirement should be followed.
- Track critical data in a revision control system.
- Enforce auditability of the requirements, architecture, design, code, tests, bugs, and fixes.
- Use revision control/version management that meets the requirements described later in this section.

Descriptions

Use Secret level cleared personnel

Use personnel with at least a Secret level clearance to perform designated work

Track critical data in a revision control system

The program should identify and document all data that is considered critical. Each critical data item should be stored and tracked in the revision control system. Minimally, the following documents, data artifacts, and tool configurations should be managed in the revision control system:

- Third-party IP (3PIP)
- Utilized libraries
- Development files, code, software used for development, synthesis scripts, and tools
- Test benches, test plans, test procedures, and test reports
- Tool configuration settings
- Design documents

Enforce auditability

Enforce auditability of the requirements, architecture, design, code, tests, bugs, and fixes. At a minimum, audit data should include what decisions were made, by whom, for what reason, and on what date.



Use revision control/version management

Revision control/version management systems should meet the following requirements:

- Allow only authorized system administrators to make changes to the underlying revision control tool and underlying server.
- Implement a backup system that mimics the primary system and is maintained by a separate administrator. Separate system administrators should manage each system.
- Enforce administrative restrictions; restrict privileged access to authorized personnel only; limit what users can do to the database; ensure all users are verified; encrypt database information—both in transit and at rest; enforce secure passwords; enforce role-based access control and privileges; and remove unused accounts.
- Remove any components or functions that are not needed; for example, remove all sample files and default passwords.
- Ensure the system provides a complete and immutable long-term change history of every file. The system must log every change made by individuals. This includes creation and deletion of files and content edits. The history must include the person who made the change, what was changed, the date, and written notes on the purpose of each change.
- Ensure the system stores a reliable copy of assets that are currently in production.
- Ensure the system stores reliable copies of previous production versions of assets, allowing for the complete retrieval of those versions.
- Enforce password best practices (password rotation, length, etc.). In lieu of a password, two-factor authentication can be used.

TD 3.1 Mitigating the introduction of a compromised design into the application

In this scenario, the adversary is able to insert a Trojan into the design after the design has been verified, but before the design is loaded for final deployment. Strict controls on the revision control system will help prevent the adversary from making unmonitored changes.



To accomplish this task the adversary would have to compromise the revision control system. That compromise could allow the adversary to change the verified configuration files, settings, hash, or other pertinent information. To protect against this, the program should store and isolate the verified configuration files, settings, and associated hashes. Before the design is loaded for final deployment, the program should verify the hash to ensure that the verified version is the same as what they are going to deploy. For extra assurance, the program has all the necessary data to reproduce the build and can verify the stored version against the reproduced version.

Mitigations

- Physically isolate and store the application design until it is delivered.
- Perform reproducible build of the application.

Descriptions

Isolate and store the application design

To protect the application design after verification but before deployment, the final configuration file and hash should be physically isolated and stored until it is delivered for provisioning. Ensure the file can only be accessed via authentication of two distinct parties. No single individual should be able to access the file. The limited set of people with access should have to follow access control procedures such that access is controlled, monitored, logged, and auditable.

Perform reproducible build

Use a reproducible build process to verify the integrity of the FPGA synthesis and build software. The reproducible build performs the synthesis process that takes in human readable HDL, and other human readable inputs, and consistently generates the same final configuration file (bitstream). It is expected that this process will, in most cases, require the use of the same version of the Electronic Design Automation (EDA) tools, and in some cases the same operating system version. This process will highlight the possession of modified software where there is a mismatch. Contact the FPGA software vendors or JFAC for more information on how to perform reproducible builds.



TD 3.2 Mitigating the modification of test benches or plans to reduce coverage or hide Trojan code

In this threat, the adversary makes changes to the test bench to hide malicious code, reduce coverage or reduce functionality.

Mitigations

- Create and execute a documented test plan that identifies the various test reviews that will take place, analysis to be performed, type of testing to be performed, and the methods used to accomplish the test.
- Validate and verify test processes which include design/test team separation, peer reviews, and use of automated tools where applicable.
- Maintain test environment via configuration management as a critical system.

Descriptions

Execute a documented test plan

The program should consider assurance when creating and maintaining the test plan. The test plan and processes should at least:

- Provide a mechanism to verify all the requirements captured in the FPGA application specification.
- Explicitly list code coverage metrics, the type of testing that will be performed, and acceptable testing guidelines. Code coverage should state how much code is checked by the test bench, providing information about dead code in the design and holes in the test suites. Document the decision to use/not use other types of testing, such as directed test, constrained random stimulus, and assertion.
- Ensure code coverage includes statement coverage, branch coverage, Finite State Machine (FSM), condition, expression, and toggle coverage. Document any code that will not be covered and why. Ensure untested code is documented and reviewed through the review process. Use functional tests to verify the FPGA does what it is supposed to do. Any deviations must be documented and approved.
- Specify the verification environment which describes the tools, the software, and the equipment needed to perform the reviews, analysis, and tests. Each of these items should be maintained under revision control.



- Document and analyze unexpected behavior and final implementation conclusions.
- Ensure all test discrepancies, bugs, etc., are resolved via a change process.

Validate and verify the test processes

The program should take care to ensure test processes consider assurance needs. This includes design/test team separation, peer reviews, and use of automated tools where applicable. All test discrepancies, bugs, etc., should be resolved via a change process utilizing a change management system. The established processes should be documented, enforced, and audited.

Maintain test environment via configuration management

The test environment should be treated as a critical system and maintained similarly to the production environment.

TD 3.3 Mitigating the introduction of a Trojan into the application design during development

In this scenario, malicious functionality is introduced into the application design during the development phase.

Mitigations

- Maintain bi-directional links to approved requirements. Tracing to design decisions is permitted in support of derived requirements.
- Enforce peer review best practices.
- Create and execute a documented test plan.
- Implement, validate, and verify test processes which include design/test team separation, peer reviews, and use of automated tools where applicable.
- Select a formal “proof” process that can validate the equivalency of the HDL and the final configuration file. For more information on “proof” tools, contact JFAC.

Descriptions

Maintain bi-directional links to approved requirements

All requirements should be documented and traced. Functionality that is not associated with a requirement should not be allowed.



Enforce peer review

Establish and enforce peer review processes with the following characteristics:

- The author and the reviewer must be different people.
- Ensure the design process has time allocated for code reviews.
- Code review should be done in parallel with development, reviewing small chunks at a time.
- Anyone reviewing the code should already be familiar with the approved architecture.
- All black box portions of the design must be identified, justified, and approved.
- All scripts that produce design artifacts (HDL, Netlist, etc.) must be reviewed and approved. Ensure there are no unexpected paths, filenames, or suppressed outputs.
- Ensure the code reviews, at a minimum, verify:
 - The code does what it is intended to do.
 - The code can be traced to requirements.
 - The code is not needlessly complex.
 - Coding standards are being utilized.
 - No extraneous code exists: the developer is not implementing unapproved items that may have future utility.
 - The code has appropriate unit tests.
 - Tests are well designed.
 - The developer used clear names for everything.
 - Comments are clear and useful and mostly explain “why” instead of “what”.

Execute a documented test plan

The program should consider assurance when creating and maintaining the test plan.

The test plan and processes should at least:

- Provide a mechanism to verify all requirements captured in the FPGA application specification.
- Explicitly list code coverage metrics, the type of testing that will be performed, and acceptable testing guidelines. Code coverage should state how much code is checked by the test bench, providing information about dead code in the design and holes in the test suites. Document the decision to use/not use other



types of testing such as directed test, constrained random stimulus, and assertion.

- Specify the verification environment which describes the tools, the software, and the equipment needed to perform the reviews, analysis, and tests. Each of these items should be maintained under revision control.
- Document and analyze unexpected behavior and final implementation conclusions.
- Ensure code coverage includes statement coverage, branch coverage, FSM, condition, expression, and toggle coverage. Document any code that will not be covered and why. Ensure untested code is documented and reviewed through the review process. Use functional tests to verify the FPGA does what it is supposed to do. Any deviations must be documented and approved.
- Ensure all test discrepancies, bugs, etc., are resolved via a change process.

Implement, validate, and verify test processes

The program should take care to ensure test processes consider assurance needs. This includes design/test team separation, peer reviews, and use of automated tools where applicable. All test discrepancies, bugs, etc., should be resolved via a change process utilizing a change management system. The established processes should be documented, enforced and audited

Select a formal “proof” process

Use logical equivalency checking to the greatest extent possible. Equivalency checking is used to prove the tools did not modify the logic or configuration settings. To do this the final bitstream is compared to the originating application HDL to demonstrate they are logically equivalent with no extraneous logic in the final format. This approach confirms Trojans were not inserted during the implementation steps. This check also confirms configuration settings are maintained and not altered. Configuration settings are those parameters included in the configuration file that affect the behavior of the FPGA device itself, but are not a part of the program application. Examples would include tamper settings, Joint Test Action Group (JTAG) settings, and key storage.

There are technical challenges associated with performing logical equivalency checking (LEC) on FPGA data. Contact JFAC for information on emerging industry tools that can assist in identifying configuration data in the FPGA formats or automate the creation of hints files.



TD 3.4 Mitigating the introduction of compromised tooling or software into the environment

In this scenario, the adversary introduces compromised tooling or software into the environment. This can be accomplished by an insider or through network intrusion.

Mitigations

- Validate cryptographic hashes against hashes signed by the vendor.
- *Research vulnerabilities affecting tools/platforms using commercial and JFAC provided resources. If vulnerabilities are found, use an alternate or newer version that does not have the vulnerability. Alternatively, perform a risk assessment and coordinate findings with JFAC.
- *Validate tools.

Validate cryptographic hashes

All parts of the software delivery should be authenticated by comparing the cryptographic hash of all received software against the hash signed by the vendor. This includes “install” macros and other support functions. Only accept certificates validated by reputable third parties. Only accept publicly released software and document the source of the hash signature and the hash itself.

Research vulnerabilities

Software and tooling vulnerabilities can be exploited for nefarious purposes. The program should actively monitor for vulnerabilities and perform risk assessment for any software or tools selected. Platforms and tool vulnerabilities can be found in databases such as:

- Common Vulnerabilities and Exposures (CVE) – <https://cve.mitre.org>
- National Vulnerabilities Database (NVD) – <https://nvd.nist.gov>
- Government Industry Data Exchange Program (GIDEP) – <https://www.gidep.org/products/products.htm>
- DISA Security Technical Implementation Guides (STIGs) – <https://public.cyber.mil/stigs/>
- Searches for vendor advisories, publications and academic papers detailing vulnerabilities in the device in question.



- Contact the vendor technical representative for unreleased or pre-release vulnerability reports.

If vulnerabilities are found in the software of tools

If vulnerabilities are found in the software or tools, **choose one** of the following options:

Option 1: Select a different tool or software that does not have published vulnerabilities and meets the program requirements.

Option 2: Use standard formal processes and procedures to work with the vendor to resolve the vulnerability. Once a fix is identified, accept only formal releases and do not accept custom beta fixes, custom patches, etc., for incorporation; or

Option 3: Internally determine the vulnerability poses no significant risk to the program.

Note: If a vulnerability is identified, it is recommended to report it to the Government Industry Data Exchange Program (GIDEP) and to contact the vendor so they may correct it.

Validate tools

Validate that the tool delivers the expected output by selecting from one of the options below:

Option 1: Select a formal “proof” process that can validate the equivalency of the HDL and final configuration file.

Option 2: Use a reproducible build process to generate any deployable configuration files, **AND** acquire EDA tools from at least two different distributors.

Use a formal “proof” process

Use logical equivalency checking (LEC) to the greatest extent possible. LEC is used to prove the tools did not modify the logic or configuration settings. To do this, the final bitstream is compared to the originating application HDL to demonstrate they are logically equivalent with no extraneous logic in the final format. This approach confirms Trojans were not inserted during the implementation steps. This check also confirms configuration settings are maintained and not altered. Configuration settings are those parameters included in the configuration file that affect the behavior of the FPGA device



itself, but are not a part of the program application. Examples would include tamper settings, JTAG settings, and key storage.

There are technical challenges associated with performing LEC on FPGA data. Contact JFAC for information on emerging industry tools that can assist in identifying configuration data in the FPGA formats or automate the creation of hints files.

Use a reproducible build process

A reproducible build process is a methodology to verify the integrity of the FPGA synthesis and build software. A reproducible build performs the synthesis process taking in human readable HDL, and other human readable inputs, and consistently generates the same final configuration file (bitstream).

Acquire EDA tools from at least two different distributors

At LoA3, reproducible builds should be performed using independently acquired software and installed independently on two distinct computers. It is expected that this process will, in most cases, require the use of the same version of the EDA tools, and in some cases the same operating system version. This process will highlight the possession of modified software where there is a mismatch. Contact the FPGA software vendors for more information on how to perform reproducible builds.

TD 3.5 Mitigating intrusion into the internal network

In this scenario, an adversary gains access to the internal network. With this access, the adversary can employ multiple methods to achieve nefarious goals, such as making modifications to tools, swapping files, etc.

Mitigations

- Assign roles.
- Control and monitor access, including physical and logical restrictions.
- Periodically research vulnerabilities using commercial and JFAC provided information. If vulnerabilities are found, use an alternate or newer version that does not have the vulnerability. Alternatively, perform a risk assessment and coordinate findings with JFAC.
- Use a secret or classified network to protect from remote attack.



Descriptions

Assign roles

Employees should be assigned a specified role with associated accesses and privileges based on the role. At a minimum, these roles should include design, test, network administration and system administration. Roles should also be defined and documented with no overlap. For example, the test engineer should not be the same person who wrote the requirements to be tested. Users should not have multiple roles.

Note: In many real-world flows, designers and testers will require **elevated privileges**. Some of these elevated privileges may be shared with system administrators. Some may have names ("local admin," "root," etc.) that imply system administration. For example, a member of the design team working on a software hardware interface may require local administrative privileges to install and debug their work. A member of the test team for an FPGA-based device connected to an IP network might require the ability to configure multiple network devices in the test environment, as well as to connect a computer in promiscuous mode to that same test environment. Those accesses represent a part of the design or test role. However, these must be based on the needs of the design or test process.

Elevated privileges on computers should be granted only as needed, and kept local to specific computers. Elevated privileges should never include administrative access to revision control servers, software installation, or other corporate infrastructure.

Elevated privileges on networks should be limited to distinct test networks, properly isolated from the design environment and the corporate network.

Control and monitor access

Employees should only have access to areas, equipment, data, and information necessary to meet the requirements of their assigned job. Entry/access to appropriate areas should be recorded, monitored, and logged for auditability.

Research vulnerabilities

Software and tooling vulnerabilities can be exploited for nefarious purposes. The program should actively monitor for vulnerabilities and perform risk assessment for any software or tools selected. Platforms and tool vulnerabilities can be found in databases, such as:



- Common Vulnerabilities and Exposures (CVE) – <https://cve.mitre.org>
- National Vulnerabilities Database (NVD) – <https://nvd.nist.gov>
- Government Industry Data Exchange Program (GIDEP) – <https://www.gidep.org/products/products.htm>
- DISA Security Technical Implementation Guides (STIGs) – <https://public.cyber.mil/stigs/>
- Searches for vendor advisories, publications, and academic papers detailing vulnerabilities in the device in question.

Use a secret or classified network

Programs should select a network classified at the Defense Security Cooperation Agency (DSCA) Secret level or above.

TD 3.6 Mitigating risk from a compromised employee

This scenario involves the compromise of an employee with access to the design, tools, or network being used for design or test.

Mitigations

- Enforce auditability of the requirements, architecture, design, code, tests, bugs, and fixes.
- Enforce the approved design process.
- Identify, document, and review critical design activities. These items should be reviewed by a cleared individual that is different than the original designer.
- Use cleared personnel in an environment certified to handle classified material at the Secret level or higher by DSCA. This also includes design centers certified for Trust Category I by DMEA.

Note: For this threat, independent is defined as "not the originator." The reviewer can be on the same team if necessary.

Descriptions

Enforce auditability

Enforce auditability of the requirements, architecture, design, code, tests, bugs, and fixes. At a minimum, audit data includes what decisions were made, by whom, for what reason, and on what date.



Enforce the approved design process

The design should include the identification of all assurance critical activities and highlight how each will be reviewed. The design process should ensure the design is reviewed by multiple cleared individuals. The original designer should not be the responsible party for performing the review. The cleared reviewers should assess the satisfaction of all requirements, ensure no extraneous design, and assess all vulnerability activities, including identification of vulnerabilities and the appropriateness of the mitigations. The design process should contain clear entry and exit criteria. Entry and exit criteria should incorporate peer reviews and technical reviews with management approval to exit a phase.

Review critical design activities

Ensure all critical activities are identified, documented, and the entire design is reviewed by multiple cleared individuals other than the original designer. Reviewers should assess all critical activities. Specific considerations include:

- Design source files in conjunction with behavioral simulations
- Design synthesis in conjunction with functional verification
- Design implementation in conjunction with static timing analysis
- Bitstream generation with reproducible build results
- Programming in conjunction with in-circuit verification

Ensure that the review teams do not include the original designers and each reviewer should hold a U.S. Secret security clearance.

Use cleared personnel

Use personnel with at least a Secret level clearance to perform designated work.

TD 3.7 Mitigating risk associated with the compromise of device identifiers

It is imperative to protect the device IDs, ensuring adversaries are not able to utilize this information to track devices, swap counterfeits into the stores, or manipulate device controls.



Mitigations

- Store device identifiers in a protected area utilizing access control. This should include physical or logical separation, and could be a safe, a classified network, or a sensitive compartmented information facility (SCIF).
- Limit access to device identifier information to those that need it for completion of job responsibilities.

Descriptions

Store device identifiers

Store devices in a protected area utilizing access control. This should include physical or logical separation, and could be a safe, a classified network, or a SCIF.

Limit access to device identifier information

Limit device identifier information to those that need it for completion of job responsibilities.

TD 4: Adversary compromises system assembly, keying, or provisioning

In this threat, an adversary has carried out an attack on the system during printed circuit board (PCB) assembly, key injection, or flash provisioning. This attack could include the assembly house acquiring counterfeit parts on behalf of the end customer, swapping out authentic FPGA parts for counterfeit ones, stealing or compromising configuration data, or stealing or modifying keys. Multiple parties can be involved during the system assembly phase. The following areas of the supply chain are included in this threat:

- Shipping devices to the PCB assembly facility.
- Transmitting keys, configuration data and FPGA part numbers to the assembly facility.
- Injecting keys into the FPGA devices.
- Provisioning the configuration storage devices.
- Attaching the FPGA devices to the PCB.
- Testing PCBs.
- Shipping the PCBs to the next manufacturing stage.



Of particular concern in this attack is the assumed existence of one or more cleared insiders working maliciously in some portion of this manufacturing process. At LoA3, this insider could be working alone or in partnership with an external party to influence the outcome. Additionally, in LoA3, attacks can also result in reliability or performance degradation. The following mitigations are built to address these premises:

- All assembly work requires after-the-fact validation by the program validation team in a cleared facility.
- The assembly work should be conducted in a facility minimally classified as Secret. The post-fab validation should be done by a verification team with Secret clearance and independent of those who conducted the assembly work. The duplication is necessary as cleared insiders working in conjunction can compromise the device and the validation process. The use of multiple cleared teams helps to reduce the risk of that scenario.

It is recommended that all mitigation steps be performed in a classified facility.

TD 4 mitigations

Regardless of where the work is performed, the program should implement the following list of mitigations in the assembly, keying, and provisioning process:

- Purchase from DoD authorized vendors and distributors. The DoD program acquisition group can provide this information.
- Follow storage and shipping guidance when storing or transferring FPGA devices between locations.
- Provide keys and configuration data to the provisioning house in digitally signed packages and with hashes.
- Prior to provisioning, clear memory devices that store configuration data.
- Provision private keys into the FPGA devices in a DSCA Classified Secret or Trust Category I certified facility after the assembly process.
- Protect the configuration data package by sending it separately to the assembly house and the validation team.
- Following assembly and provisioning, perform verification activities in a DSCA Classified Secret or Trust Category I certified facility.
- *Authenticate the FPGA device after being out of the control of the program.



TD 4 mitigation descriptions

Purchase from DoD authorized vendors and distributors

Use DoD authorized vendors for all purchases. Authorized vendors can be located through the acquisition organization.

Follow storage and shipping guidance

All devices should be verified upon receipt. Once verification has taken place, production devices should be stored and maintained in a restricted area separate from non-production devices (design, test, etc.). Production devices should be continuously tracked to include arrival of the device by unique identifier, interaction anyone has with the device, and exit of the device from inventory. The restricted area should enforce access control that limits access to only a minimum subset of people that require access to support direct job responsibilities and excludes all members of the design team. The restricted area should have a clearly defined perimeter, but physical barriers are not required. Personnel within the area are responsible for challenging all persons who may lack appropriate access authority. The restricted area access should be audited to include data containing who entered/exited the area with a timestamp and reason for entry.

Shipping should be controlled and managed. JFAC recommends shipping material using a commercial carrier that has been approved by the CSA to transport Secret shipments, although the material is not Secret. Commercial carriers may be used only within and between the 48 contiguous States and the District of Columbia or wholly within Alaska, Hawaii, Puerto Rico, or a U.S. possession or trust territory. When shipping using a commercial carrier take efforts to afford additional protection against pilferage, theft, and compromise as follows. This includes using hardened containers unless specifically authorized otherwise and ensuring the packages are sealed. The seals should be numbered and the numbers indicated on all copies of the bill of lading (BL). When seals are used, the BL shall be annotated substantially as follows: DO NOT BREAK SEALS EXCEPT IN CASE OF EMERGENCY OR UPON PRIOR AUTHORITY OF THE CONSIGNOR OR CONSIGNEE. IF FOUND BROKEN OR IF BROKEN FOR EMERGENCY REASONS, APPLY CARRIER'S SEALS AS SOON AS POSSIBLE AND IMMEDIATELY NOTIFY BOTH THE CONSIGNOR AND THE CONSIGNEE.



Provide keys and configuration data

Provide keys and configuration data to the provisioning house in digitally signed packages and with hashes. JFAC recommends that these data packages be encrypted using the Advanced Encryption Standard (AES) algorithm with a key of at least 256-bit length. The assembly house should utilize the signature and hash to verify the integrity of the contents.

Clear memory devices

Prior to provisioning, clear memory devices that store configuration data. This prevents an adversary from storing malicious configuration data in non-used areas of the memory device. These memory devices could include a discrete PCB component like a Flash or the on-chip FPGA non-volatile storage available on certain devices.

Provision private keys

Provision private keys into the FPGA devices in a DSCA Classified Secret or Trust Category I certified facility after the assembly process.

Protect the configuration data package

The program should ensure there are processes and procedures in place to ensure that the configuration data package is provided to the assembly house and the validation team in a manner that cannot be corrupted by a single individual. The data should be provided directly and independently to each destination. The assembly house should not be used to pass the data to the test facility. Ensure there is a golden copy provided to each functional area ensuring the same data is transmitted.

Perform verification activities

Following assembly and provisioning, perform all verification activities in a DSCA Classified Secret or Trust Category I certified facility.

At LoA3, there can be multiple compromised cleared insiders. To mitigate this threat, a team of people cleared at the Secret level and independent from the assembly and provisioning team should be utilized to conduct the validation.

Those performing this validation must:

- **Verify** the PCB traces related to the FPGA device, the configuration memory devices, and any other devices related to the authentication of the configuration



data. If needed, the program should rely on guidance from the JFAC PCB Executive Agent to perform this verification.

- **Verify** the authenticity of the configuration data loaded on the FPGA memory device following provisioning and assembly. The verification can be executed by a bit comparison or a hash. This verification must be performed by a team independent of the assembly and provisioning process. The verification should cover the entire contents of the memory device and not just the addresses containing the configuration data. It is recommended to program the entire memory space to disallow unused memory for nefarious purposes.
- **Verify** that the FPGA system can cryptographically authenticate all loaded configuration data as part of the system containing the FPGA upon load. The authentication methodology should verify both the source and contents.
- **Verify** that the proper post assembly keys have been loaded into the FPGA key storage elements. This verification must be performed by a team independent of the assembly and provisioning process. Some FPGA devices allow a hash of the keys to be read out for confirmation. Additionally, the program should create test bitstreams to verify that the devices can properly utilize the keys and can reject actions using wrong keys.
- **Verify** the authenticity of the FPGA device to rule out the introduction of a counterfeit part during assembly.

Authenticate the FPGA device

When the FPGA has been out of positive control of the program it must be authenticated. The program should select one of the options below:

Option 1: Verify the device on the PCB is an authentic and authorized device by validating that each device has a unique cryptographic ID signed by the vendor. Each device must contain a unique private asymmetric key for which no read function exists, and validation must involve the device signing a nonce. A NIST approved asymmetric authentication algorithm must be used for this. The program should authenticate the FPGA devices utilizing this ID when they have been out of the positive control of the program.

Option 2: Verify the device on the PCB is an authentic and authorized device by performing physical counterfeit inspection with destructive sampling as described under *Perform physical inspection/analysis*. This is primarily an SAE International AS6171



Test Methods Standard; General Requirements, Suspect/Counterfeit, Electrical, Electronic and Electromechanical Parts based evaluation, with requirements to obtain vendor information.

Option 3: Use a soft PUF. Verify the device on the PCB is an authentic and authorized device by utilizing a soft PUF to create unique IDs. The soft PUF is used to validate the integrity of the devices when they are outside of the program's control. The program should generate these IDs when FPGAs are in their control by loading the soft PUF into the FPGA fabric, use it to generate a unique ID for the respective device, and then delete the PUF. Following assembly, the program should repeat this process and ensure the ID matches, authenticating the device. If the soft PUF will be used to authenticate the device when it is outside the program control, it is recommended that the following be done:

- Prevent readout of the PUF output to the FPGA's external pins.
- Utilize the PUF to encrypt a nonce that can transmit outside the device.
- Utilize a public key based on the PUF value to decrypt the nonce and authenticate the device.

This approach can be used to support remote attestation when needed.

TD 5: Adversary compromises third-party soft IP

In this threat, an adversary compromises third-party soft IP intended for integration into the configuration of the FPGA. The compromise can occur during the IP's development cycle, during its delivery, or while it is at rest at the program's design center. In all scenarios, the compromised IP contains a malicious function that was inserted during its design and can be triggered through some input to the FPGA, or when a specific scenario occurs. In all cases, it is important to remember the purpose of the Trojan is unknown, but probable impacts include functional change, performance, power, or reliability. The mitigations to these attacks focus on verifying integrity of the delivery of the IP and reviews of its HDL code.

See *Appendix B: IP Reuse Guidance* for information describing parameters for reusing internally created or previously evaluated IP.

TD 5 mitigations

- Purchase from DoD authorized vendors and distributors.



- Only accept IP that is unobfuscated and distributed as source code.
- Ensure IP deliverable packages are digitally signed.
- Validate the cryptographic hash of the IP against the hash signed by the vendor.
- Store IP in a revision control repository immediately upon receipt with the hashes used to authenticate the contents. Protection of the hash will allow for re-verification of the IP at a later date.
- *Examine IP for malicious functions.

TD 5 mitigation descriptions

Purchase from DoD authorized vendors and distributors

Use DoD authorized vendors and distributors for all purchases. Authorized vendors can be identified through the acquisition organization.

Only accept IP that is unobfuscated

Only accept IP that is unencrypted, unobfuscated, and distributed as source code. IP must be human readable for review.

Ensure IP deliverable packages are digitally signed

The program should only accept digital signature certificates validated by reputable third parties. The program should be limited to publicly released software and not special or custom distributions of the software. The program should maintain documentation of the vendor provided signature and hash, and the actual software hash.

Validate the cryptographic hash

Ensure that the cryptographic hash of the IP is validated against the hash signed by the vendor. All parts of the software delivery should be authenticated in this manner including “install” macros and other support functions. The program should only accept certificates validated by reputable third parties. The program should be limited to publicly released software. The program should maintain documentation of the source of the hash and the actual software hash.

Store IP in a revision control repository

Immediately upon receipt, the IP with its associated hash should be checked into a version control repository. The hash of the IP should be verified at various stages to ensure there have been no modifications. The hash should be stored separately from the IP block and be made read-only to the development team.



Examine IP for malicious functions

To examine the IP for malicious functions, choose one of the following options:

Option 1: Have two cleared personnel review the IP, according to the JFAC guidance in Third-Party IP Review for Level of Assurance 3. JFAC can provide this document upon request.

Option 2: Contact JFAC to determine if an IP review of the complete IP package has been previously completed. If JFAC has not performed an IP review, option 1 must be selected.

TD 6: Adversary swaps configuration file on target

In this threat, an adversary obtains access to the system during or after assembly and can compromise the FPGA device's operation via a modification to the configuration data.

For assurance purposes, these guidelines are not concerned with the exposure of the configuration data or the confidentiality of the public keys, as they do not compromise the authentication of the data. However, programs with security requirements may need to protect this information and can choose to implement additional protections.

Technological mitigations exist publicly for this threat such as configuration data authentication. Mitigations must involve authenticating the configuration file for both integrity and provenance. JFAC encourages programs to use device families that support configuration data authentication.

Programs are discouraged from using devices that do not support configuration data authentication. In this scenario, authentication practices apply to all configuration file loads, including local loads, remote updates, multi-boot scenarios, configuration via software, and configuration via protocol where the configuration file is loaded into the FPGA. For devices that store the data internally in non-volatile memory (NVM), this requirement only applies to the initial loading.

As of October 2022, all the major U.S. FPGA vendors provide built-in functionality to authenticate configuration files either at load into internal memory or at configuration for at least one device family. The specifics of this authentication vary greatly. The exact details of key management and storage vary from device to device. Some offer facilities



to store many authentication keys, some use fuses, and others use independently powered random access memory (RAM). Further, there are public techniques to subvert the authentication, which have complex implications for the security of built-in authentication¹.

The result is that the exact security of each method is not apparent without a detailed evaluation. This report communicates the specific mechanisms that meet JFAC expectations, as well as caveats for their use. As a rule, the program must use CNSS or NIST approved asymmetric cryptographic algorithms at LoA3.

To achieve LoA3, all boot/configuration images must be authenticated with respect to their source and data integrity. That is, the device must validate that the file comes from an authorized provider and that the data has not been modified prior to loading. For LoA3, the recommended method for authenticating the data source is to use an asymmetric algorithm recommended by CNSS or NIST. Asymmetric algorithms are preferred because they do not require the protection of a secret key. For data integrity, a hashing algorithm, such as secure hashing algorithm (SHA), is recommended. Many of the existing FPGA devices provide these functions for the user.

TD 6 mitigations

These are the configuration file threat mitigations:

- Incorporate cryptographic authentication of all loaded configuration data as part of the system containing the FPGA.
- Design the system to authenticate configuration data each time the data is loaded into the FPGA device.
- Configure all production devices in a way that prevents direct read back of the private keys through electrical means.
- Use a CNSS/NIST approved algorithm and key length.
- Use DoD evaluated authentication mechanisms.
- Disable test access pins in fielded products.
- *When the program utilizes mechanisms that allow application updates, ensure authentication for modifications is supported

¹ The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series FPGAs. Usenix Security '20. Maik Ender, Amir Moradi, Christof Paar.



- Generate and store all authentication keys on a program controlled, FIPS 140-2 compliant, Level 2 hardware security module (HSM)

TD 6 mitigation descriptions

Incorporate cryptographic authentication

The program should enforce cryptographic authentication of the configuration file. In addition, the program should maintain documentation including the authentication methodology, its architecture, and its compliance with appropriate CNNS Policy if the project is identified as a National Security System. Otherwise, ensure compliance with appropriate NIST standards.

Authenticate configuration data each time the data is loaded

Design the system to authenticate configuration data each time the data is loaded into the FPGA device.

Prevent direct read back

Configure all production devices in a way that prevents direct read back of the private keys through electrical means.

Use a CNSS/NIST approved algorithm and key length

If the project is identified as an NSS, use a CNSS Policy approved algorithm and key length. Otherwise use a NIST approved algorithm and key length, as described in the latest approved version of FIPS 186, *Digital Signature Standard*, or FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC)*.

Use DoD evaluated authentication mechanisms

The program can either select an authentication mechanism with an existing evaluation or sponsor the evaluation itself. JFAC can perform evaluations and maintains best practices in using commercial technology for this purpose.

At a minimum, any evaluation must:

- Ensure compliance with the current version of FIPS 186, *Digital Signature Standard*.
- Authenticate all boot configuration data.
- Confirm its ability to verify data integrity using positive and negative testing.



- Confirm its ability to verify the authorized source using positive and negative testing.
- Ensure authentication is applied to all configuration data regardless of how it is stored or delivered prior to or in parallel to configuration.
- Verify the authentication mechanisms do not contain any known vulnerabilities.
- All keys must be generated and protected in accordance with FIPS 140-2 Level 2².
- The use and operation of application test access is disabled in fielded products.

Disable test access pins

All modern FPGA family devices have hardware test interfaces to support fabrication testing of the device and testing of the user product. These interfaces usually include Joint Test Action Group (JTAG) pins and dedicated test pins.

JTAG pins should be disabled in fielded products. It is a common practice to disable these access points prior to fielding the device. JFAC recommends disabling this in non-volatile fuses when available.

Ensure authentication for modifications

Many FPGA platforms contain mechanisms that allow the application to change or update itself. Some allow for true in-flight reprogramming, where some portion of the FPGA continues normal operation while another portion changes its behavior. Others allow for reprogramming via external storage. Ensure that the built-in application change technique applies authentication to *all* the reconfiguration data.

The names of these operations are system specific and include terms like “dynamic reconfiguration,” “partial reconfiguration,” “in-application programming,” etc. In practice, most FPGA device families **do not** provide the same degree of authentication that the primary programming mechanisms provide.

Authenticating reconfiguration data in the application itself

In this case, the program incorporates functions in the application to perform authentication on configuration data when the FPGA device cannot. When utilizing this option, the program should pay attention to the following considerations.

² FIPS 140-2 will be replaced at a future date with FIPS 140-3.



System-on-chip FPGAs (SoC FPGAs) incorporate central processing units (CPUs) as a component of a reconfigurable platform. The JFAC FPGA Best Practices do not seek to provide software assurance to the application running in the CPUs of a SoC FPGA. However, the best practices listed here will provide the same degree of assurance to the initial user code (sometimes called a bootloader) executed by the CPU.

From there, it is possible for a designer to extend the same authenticity to the user code if their system requires it. In cases where the program uses an interface between the FPGA fabric and the SoC in order to have one function load the other, it is vital that no path exists from this interface to the input/output (I/O). It is up to the program to ensure that only the application has access to it.

In some platforms, security settings can be programmed into both non-volatile storage in the device itself and as a setting in the configuration file loaded into the device. Settings should always be programmed in the non-volatile storage of the device. In those cases where use of security settings within the configuration file is acceptable, it must be explicitly noted.

Some platforms provide support for remotely updating the boot or configuration data on the FPGA device. This update is sent via a network, stored locally on the FPGA device, and then loaded into the device by the application.

An application designer using these operations should implement one of the following two options:

Option 1: Validate that the built-in application change technique being used fully applies authentication to **all** the reconfiguration data.

Option 2: Perform authentication of the reconfiguration data in the application itself. Many platforms support the ability to load different boot or configuration files from a local memory. This methodology involves the current application instructing the device to point to a new memory location for the boot/configuration information. In these cases, the device maintains a pointer to the original data if there is a load error with new file. It is necessary to ensure that all boot/configurations can be authenticated with respect to its source and data integrity in the same manner as the base load. Many devices leave this task to the application to perform.



Use a FIPS 140-2 compliant, Level 2 HSM

Generate and store all authentication keys on a program controlled, FIPS 140-2 compliant, Level 2 HSM with the HSM configured to enforce role-based restrictions on the use of the keys. Maintain an approved list of individuals who can access the keys.

It is worth noting that there are additional protections that can be applied to the FPGA configuration data when its fielded location is physically unguarded. These include:

- Configuration file encryption using a NIST or DoD approved algorithm.
- The use of split decryption keys to make key theft more difficult. This involves storing multiple keys throughout the system, concatenating them, and then using the hash of the concatenation as the decryption key.
- The use of PUFs for key generation or a combination of PUF output and stored key.
- Utilize any additional key protection mechanisms provided by the vendors.
- Utilize good physical access protections for the PCB.

TD 7: Adversary substitutes modified FPGA software design suite

In this threat, an adversary replaces the design suite an application designer uses with one modified to subvert the application during synthesis, place and route, or configuration data generation. In this threat, the adversary would have access to a modified version of commercial vendor software and would use the modified software to:

- Subvert the security features of an FPGA during configuration data generation.
- Insert a malicious function into the device during synthesis, place and route or configuration data generation.
- Insert a data leak or backdoor into the synthesized device during synthesis, place and route, or configuration data generation.

This subverted tool would then be entered into the program's design environment by a vendor insider, an adversary-in-the-middle technique, or through a network intrusion. This threat **does not** include the scenario where an FPGA vendor insider modifies the authorized software during development for malicious purposes, which is covered by *TD 10: Adversary modifies vendor FPGA software design suite during development.*



TD 7 mitigations

- Purchase from DoD authorized vendors and distributors. Both DoD and vendors have recommendations for the appropriate distributors of products. The DoD program acquisition group can provide this information.
- Prevent automatic tool updates by using an installation and update process that does not require Internet connectivity.
- Install and execute software using a trusted computing environment to protect from remote intrusions.
- Use cleared personnel with at least a Secret level clearance.
- Validate the cryptographic hash of the software against the hash signed by the vendor.
- *Validate the tool output has not modified the source design.

TD 7 mitigation descriptions

Purchase from DoD authorized vendors and distributors

Use DoD authorized vendors for all purchases. Authorized vendors can be identified through the acquisition organization.

Prevent automatic tool updates

Prevent automatic tool updates by using an installation and update process that does not require Internet connectivity.

Use a trusted computing environment

Programs should select one of the trusted computing environment options below, to protect from remote attack.

Option 1: A computer and network classified at the DSCA Secret level or above.

Option 2: A computer and network certified for use in a Trust Category 1 facility as defined by DMEA.

Option 3: A network-isolated computer enclave with limited and controlled access adhering to NIST and CMMC standards. This is a computer with the vendor software installed by a network administrator. This administrator should not be a designer working on the application design.



Use cleared personnel

Use personnel with at least a Secret level clearance to perform designated work.

Validate the cryptographic hash

Ensure the cryptographic hash of the software deliverables is validated against the hash signed by the vendor. All parts of the software delivery should be authenticated in this manner including “install” macros and other support functions. The program should only accept digital signature certificates validated by reputable third parties. The program should be limited to publicly released software and not special or custom distributions of the software. The program should maintain documentation of the vendor provided signature and hash, and the actual software hash.

Validate the tool output

Validate the tool has not inserted any Trojan by choosing one of the following options:

Option 1: Perform logical equivalency checking between the application HDL and the final configuration data. This effort should attempt to verify that the final bitstream and originating application HDL are logically equivalent with no extraneous logic in the final format. This action will confirm that no Trojans were inserted during the implementation steps.

Option 2: Use a reproducible build process to validate the software.

When using reproducible builds to validate software, enlist a third party to mirror the FPGA’s synthesis, place and route, and configuration file generation. If the mirroring is executed properly and independently, the outputs can be compared to verify that the vendor software package is unmodified or modified in a way that does not affect the application design. To ensure proper execution of this mitigation, the following must be observed:

- The software used to mirror the program’s synthesis effort must be procured in a manner to make it independent from the procurement of the original version.
- The reproducible build software should be loaded/installed by a different administrator than the administrator that performed the original install.
- This mitigation requires independent duplicative activities since the adversary could have knowledge about the project and how it obtains, loads, and controls its tools.



- The mirrored effort should utilize the same version of the software on the same operating system and version.
- The application development team's software and the mirroring software should possess matching hashes and size values.
- The mirrored effort must utilize the same HDL code, IP and synthesis scripts.
- The mirrored effort must utilize the same vendor tool settings.
- The output of the effort is an unencrypted, uncompressed configuration data file.

Contact the FPGA software vendor for more detailed guidance on creating reproducible builds. They have already performed work in this area and can assist with documented instructions.

Both the development effort and the mirror effort should execute the FPGA development flow from synthesis to configuration file output and then perform the following steps:

- Throughout the flow, output any intermediary files that can be used to compare results at various stages. This can include primitive netlists, synthesized netlists, physical netlists, and final configuration data files.
- Compare the final configuration files for size and content. They should match in all respects except for header information that may include timestamps and other property information.
- If the files are encrypted, take steps to ensure that any nonces, such as the initialization vector, used by both efforts are the same.

If discrepancies are found in the comparison, the following steps should be followed:

- Contact the software vendors for assistance.
- Contact JFAC for assistance in resolving the discrepancy.

If a software version does not match what was expected, JFAC recommends reporting it to the vendor for further analysis and correction.

TD 8: Adversary modifies FPGA platform family at design

In this threat, an adversary inserts a malicious function or preplaces a vulnerability for later use in an FPGA device during its hardware design phase. This attack involves a network intrusion or a compromised insider working for the vendor or one of its subcontractors. While this attack lacks the ability to target an individual program, it can



preposition a vulnerability for later use. Evaluation of manufactured hardware for built-in malicious functions or vulnerabilities is a very difficult, highly expensive, and near impossible task. As such, no practical amount of evaluation can guarantee the absence of any designed-in malicious function.

TD 8 mitigations

- Engage JFAC to evaluate the FPGA device family.

TD 8 mitigation description

Engage JFAC

JFAC recommends the program engage JFAC to evaluate the chosen FPGA device family or to acquire information garnered from previous evaluations. JFAC will then instruct the program on what steps to take to identify malicious code or weaknesses in their FPGA platform. Initially, the program may be asked to conduct a subset of the evaluation steps in partnership with JFAC. In parallel, JFAC may evaluate the FPGA device family for malicious behavior and operational weaknesses. In addition, JFAC has been evaluating commonly used FPGA device families proactively.

In support of this mitigation, JFAC asks all programs seeking LoA compliance at any level to provide JFAC with information regarding the FPGA devices they are using along with a brief summary of their use. This information will be compiled to create a picture of which FPGAs are of greatest interest to DoD and which ones might represent a vulnerability to multiple programs. This information will drive the decision-making behind which device families to proactively analyze for vulnerabilities.

JFAC communicates this information at a variety of classification levels. Please contact JFAC to obtain the appropriate email address at <https://jfac.navy.mil>.

Refer to *Appendix C: JFAC FPGA reporting template* for the information a program should include in the email.

As evaluations are completed, JFAC will document the findings for programs to use in their vulnerability research.

Finally, JFAC recommends that programs utilize newer and more modern device families when possible. These families possess more mature design architectures that encompass vulnerability fixes and advanced assurance features.



TD 9: Adversary compromises single-board computing system (SBCS)

In this threat, an adversary compromises a single-board computing system (SBCS) purchased by a program for use in a system. An SBCS is a commercial off-the-shelf product consisting of a PCB with FPGAs and computer processing resources. These boards are common throughout DoD systems as they are readily available in the marketplace. Under this threat, the program does not have control of the manufacturing process of the SBCS, forcing the program to rely upon a verification heavy approach to mitigating attacks. In this light, programs should work with existing DoD providers to build custom SBCS devices in compliance with LoA3 guidelines.

Of primary concern in this scenario are threats to:

- Authenticity of the FPGA devices
- PCB connections to the FPGA
- The configuration methodology
- Test interfaces

The following mitigations only address the hardware assurance concerns related to the manufacturing and operation of the FPGA device and do not consider other components of the SBCS.

TD 9 mitigations

- Programs should engage a DoD vendor to build the SBCS devices under the LoA3 constraints. This includes the use of cleared people and classified facilities, minimally at the Secret level.
- All verification and authentication steps in this section should be conducted by a team of people independent from the manufacturing team.
- Authenticate the FPGA devices.
- Verify the SBCS configuration process and that the board-level connections comply with the LoA3 mitigation requirements.
- Document the steps taken to comply with these requirements. This includes hardware and software features.
- Test nonvolatile memory verifying there are no conflicting prepopulated settings.



TD 9 mitigation descriptions

Engage a DoD vendor to build the SBCS

Programs should engage a DoD supplier to build the SBCS devices under the LoA3 constraints. This includes the use of people cleared at least at the Secret level working in Secret cleared environments.

Verification and authentication

All verification and authentication steps in this section should be conducted by a team of people independent from the manufacturing team. This team should obtain and review the SBCS schematics for functional correctness, vulnerabilities, and security concerns as they relate to the FPGA configuration process and security connections. Verify the PCB traces related to the FPGA device, the configuration memory devices, and any other devices related to the authentication of the configuration data. The program should rely on guidance from the JFAC PCB Executive Agent to perform this verification. This evaluation should be performed on all devices.

Authenticate the FPGA devices

In this mitigation, the program should authenticate the devices utilizing the recommendations found under *TD 2: Adversary inserts malicious counterfeit*. Then, the devices should be re-authenticated upon completion of the SBCS manufacture utilizing a cryptographically protected ID or through the use of a soft PUF.

Verify the SBCS configuration process

Utilize SBCSs whose configuration process and board level connections comply with the LoA3 mitigation requirements for *TD 6: Adversary swaps configuration file on target*. This includes, but is not limited to, requirements for:

- NIST compliant authentication algorithms
- Differential power analysis (DPA) resistant authentication
- Protected key storage
- Anti-tamper detection and response
- Being free of known vulnerabilities in the configuration and security functions
- All encryption and authentication keys lengths must be compliant with the requirements outlined NIST SP 800-57
- The ability to disable FPGA test pins, such as JTAG



If the configuration file memory storage device contains SBCS vendor code, the program should review and evaluate that code for malicious functions. The proprietary SBCS support for configuration must be fully understood and validated. If the SBCS configuration process cannot be fully evaluated, it should not be used at LoA3.

Once the SBCS's configuration design and implementation are evaluated to be free of malicious functions, the program should craft a set of tests and validation processes to verify that all the devices comply with the evaluation.

Test non-volatile memory

Poll the FPGA settings captured in non-volatile memory, such as fuses, to determine if the SBCS vendor has preprogrammed any settings in a manner conflicting with these assurance guidelines or that conflict with user application needs.

Document the steps

Document all steps taken to demonstrate compliance with TD 9. These steps and associated data artifacts should be auditable.

TD 10: Adversary modifies vendor FPGA software design suite during development

In this threat, an adversary modifies the vendor design suite during its development to subvert the DoD application during FPGA implementation. This subversion could include:

- Inserting a malicious function or vulnerability into the device during synthesis, place and route, or configuration data generation.
- Enabling the exfiltration of program application design data over a network connection.

This subverted tool would then be part of the authorized software delivered by the vendor and its distributors. In this light, delivery protections such as encryption, package signing, and hashes would have no mitigating value. Evaluating the vendor software and certifying it as Trojan free is a prohibitively intensive and costly venture that is not practical at the program level.

At present, the only approach to addressing this attack is to verify the results of the FPGA implementation steps. Rather than determine that the tool is Trojan free, the



approach is to verify that the tool suite did nothing malicious to the application design. Logical equivalence checking (LEC) is the tool used to perform this verification.

JFAC is currently investigating additional measures to detect and thwart compromised vendor tools. Pending new advances, JFAC can assist programs with overcoming the difficulties of performing LEC.

TD 10 mitigations

- To prevent exfiltration of data from a malicious FPGA EDA tool, perform all FPGA design work on an isolated network as recommended in the mitigations for *TD 3: Adversary compromises application design cycle*.
- Perform logical equivalency checking between the application HDL and the final configuration data.

TD 10 mitigation descriptions

Perform logical equivalency checking

To the greatest extent possible, LEC verifies that the vendor tools did not modify the logic or configuration settings. The goal is to verify that the final bitstream and originating application HDL are logically equivalent with no extraneous logic in the final format. This confirms that Trojans were not inserted during the implementation steps. The LEC also verifies that the configuration settings were maintained and not altered. Configuration settings are those parameters included in the configuration file that affect the behavior of the FPGA device itself but are not a part of the program application. Examples include tamper settings, JTAG settings, and key storage.

There are technical challenges associated with performing LEC on FPGA data. First, due to the proprietary nature of the configuration file format, including it in the LEC effort is difficult. Contact JFAC for information on commercial tools that can assist with this for several device families.

Additionally, many FPGA synthesis optimizations make it difficult to perform LEC. For this reason, the following are recommended:

- Perform LEC after each implementation step to limit the amount of change that must be accounted for by the tool. This includes synthesis, place and route, and configuration data generation.



- Use hints files to assist in matching difficult-to-correlate logic in the compared databases. Most LEC tools accept these files.
- Contact JFAC for information on emerging industry tools that can assist in identifying configuration data in the FPGA formats or automate the creation of hints files.

3 Summary

The mitigations in this report are intended to protect against adversarial threats to assurance on FPGA-based systems. Once a program incorporates the mitigations for these 10 threat descriptions, it can consider its FPGAs to have achieved LoA3.

If a program has developed alternate solutions for mitigating these threats, it can consult with JFAC to determine if the alternative mitigations are sufficient.

Finally, if a program has questions regarding this report or requires assistance, it should contact JFAC at <https://jfac.navy.mil/> for assistance.



Appendix A: Standardized terminology

The following terms are used in the Joint Federated Assurance Center Field Programmable Gate Array Best Practices documents. These terms are modified from Defense Acquisition University definitions to support common understanding.

Application design – The collection of schematics, constraints, hardware description language (HDL), and other implementation files developed to generate an FPGA configuration file for use on one or many FPGA platforms.

Application domain – This is the area of technology of the system itself, or a directly associated area of technology. For instance, the system technology domain of a radar system implemented using FPGAs would be "radar" or "electronic warfare."

Configuration file – The set of all data produced by the application design team and loaded into an FPGA to personalize it. Referred to by some designers as a "bitstream", the configuration file includes that information, as well as additional configuration settings and firmware, which some designers may not consider part of their "bitstream."

Controllable effect – Program-specific, triggerable function allowing the adversary to attack a specific target.

Device/FPGA device – A specific physical instantiation of an FPGA.

External facility – An unclassified facility that is out of the control of the program or contractor.

Field programmable gate array (FPGA) – In this context FPGA includes the full range of devices containing substantial reprogrammable digital logic. This includes devices marketed as FPGAs, complex programmable logic devices (CPLD), system-on-a-chip (SoC) FPGAs, as well as devices marketed as SoCs and containing reprogrammable digital logic capable of representing arbitrary functions. In addition, some FPGAs incorporate analog/mixed signal elements alongside substantial amounts of reprogrammable logic.

FPGA platform – An FPGA platform refers to a specific device type or family of devices from a vendor.



Hard IP – Hard IP is a hardware design captured as a physical layout, intended to be integrated into a hardware design in the layout process. Hard IP is most typically distributed as Graphic Design System II (GDSII). In some cases, Hard IP is provided by a fabrication company and the user of the IP does not have access to the full layout, but simply a size and the information needed to connect to it. Hard IP may be distributed with simulation hardware description language (HDL) and other soft components, but is defined by the fact that the portion that ends up in the final hardware was defined by a physical layout by the IP vendor.

Level of assurance (LoA) – A Level of Assurance is an established guideline that details the appropriate mitigations necessary for the implementation given the impact to national security associated with subversion of a specific system, without the need for system-by-system custom evaluation.

Physical unclonable function (PUF) – This function provides a random string of bits of a predetermined length. In the context of FPGAs, the randomness of the bitstring is based upon variations in the silicon of the device due to manufacturing. These bitstrings can be used for device IDs or keys.

Platform design – The platform design is the set of design information that specifies the FPGA platform, including physical layouts, code, etc.

Soft IP – Soft IP is a hardware design captured in hardware description language (HDL), intended to be integrated into a complete hardware design through a synthesis process. Soft IP can be distributed in a number of ways, as functional HDL or a netlist specified in HDL, encrypted or unencrypted.

System – An aggregation of system elements and enabling system elements to achieve a given purpose or provide a needed capability.

System design – System design is the set of information that defines the manufacturing, behavior, and programming of a system. It may include board designs, firmware, software, FPGA configuration files, etc.

Target – A target refers to a specific deployed instance of a given system, or a specific set of systems with a common design and function.



Targetability – The degree to which an attack may have an effect that only shows up in circumstances the adversary chooses. An attack that is poorly targetable would be more likely to be discovered accidentally, have unintended consequences, or be found in standard testing.

Third-party intellectual property (3PIP) – Functions whose development are not under the control of the designer. Use of the phrase “intellectual property”, IP, or 3PIP in outlining this methodology of design review does not refer to property rights, such as, for example, copyrights, patents, or trade secrets. It is the responsibility of the party seeking review and/or the reviewer to ensure that any rights needed to perform the review in accordance with the methodology outlined are obtained.

Threat category – A threat category refers to a part of the supply chain with a specific attack surface and set of common vulnerabilities against which many specific attacks may be possible.

Utility – The utility of an attack is the degree to which an effect has value to an adversarial operation. Higher utility effects may subvert a system or provide major denial of service effects. Lower utility attacks might degrade a capability to a limited extent.

Vulnerability – A flaw in a software, firmware, hardware, or service component resulting from a weakness that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or components.



Appendix B: IP Reuse Guidance

There are several situations in which a program/organization would like to reuse previously generated soft IP or 3PIP. This IP can be generated internally (i.e., by an authorized DoD program, but for a different program than the original use) or externally (i.e., purchased IP).

IP that was not generated for or previously evaluated by a DoD program in conjunction with LoA3 requirements should **not** be used without a program evaluation. This includes cases in which vendors have had the IP evaluated by a third party. That review is not acceptable according to the DoD Microelectronics: FPGA Overall Assurance Process. Programs have the sole responsibility to perform or oversee all reviews.

LoA3 introduces several new threat vectors, to include insiders, cleared and uncleared personnel working alone or in conjunction with others, and new technologies, along with funding at the nation-state level. Given the complexity of LoA3 and the types of components and systems that require LoA3, JFAC strongly recommends re-evaluation of all IP regardless of the source.

In situations where the program chooses not to re-review the previously evaluated IP, the program should ensure the following conditions are satisfied.

Reuse conditions

To reuse IP, the following conditions should be satisfied:

- a) The IP must have been developed internally (i.e., by a government funded and managed program) for an LoA3 program or the IP was successfully internally evaluated at LoA3.
- b) All documentation associated with the development and/or previous evaluation must be signed with a valid cryptographic signature and stored within the configuration management system compliant with the LoA3 requirements in this document. The documentation must be provided to the new program in its totality. The documentation should clearly state any known vulnerabilities or risk associated with the IP. The documentation must be proven to have remained unchanged since the time the evaluation was performed.
- c) A second copy with a different cryptographic signature of the evaluation report should be stored in a controlled environment separate from the IP. The best



storage mechanisms would include in a SCIF, and either in a safe certified at the Secret level or on a Secret network.

- d) The program should verify the data in the separately stored evaluation reports is the same as what the program is using.
- e) The program cannot accept any IP in which the report has discrepancies from the version received. For example, the name of the IP, version information, hash, etc.
- f) After the initial evaluation, the IP must remain maintained in a configuration management system compliant with the LoA3 requirements in this document. The hash of the IP must also be cryptographically signed and maintained in the configuration management system. Additionally, the hash should be stored and maintained with a second cryptographic signature. The program must verify the separately stored hashes match.
- g) In the event the IP was previously evaluated and there were areas of risk identified, the risk must be documented and provided to the program that would like to reuse the IP. The program has the responsibility to accept or mitigate the risk based on individual program needs.

Reuse scenarios

The following section describes several use cases that provide additional details of when IP can or cannot be reused at LoA3.

Scenarios in which LoA3 IP reuse is applicable:

- a) The program would like to reuse internally developed LoA3 compliant IP, but not previously evaluated outside of the initial program for use.

In this scenario, the IP was developed and stored internally using the processes described in this document. Therefore, the IP was previously shown to be compliant. The program has the responsibility to ensure that no modifications were made to the IP since the time of development. To reuse the IP, the program must demonstrate compliance with the conditions outlined in the *Reuse conditions* section above.

- b) The program would like to reuse internally developed LoA3 IP that was previously successfully evaluated to be compliant with LoA3.



In this scenario, the fact that the IP has been evaluated and deemed compliant to LoA3 makes the reuse viable provided the program can demonstrate compliance with the conditions outlined in the *Reuse conditions* section above.

- c) The program would like to use IP that was developed by an external vendor. The 3PIP was previously internally verified as compliant with LoA3 for a different program.

In this scenario, the IP was evaluated internally using the processes outlined in this document. Therefore, the IP was previously shown to be compliant. To reuse the IP, the program must demonstrate compliance with the conditions outlined in the *Reuse conditions* section above.

Use cases in which an LoA3 IP evaluation in accordance with Third-Party IP Review Process for Level of Assurance 3 document would be required:

- d) The program would like to use internally developed IP that was not developed or evaluated to satisfy any level of assurance. The program would like to use this IP at LoA3.
- e) In this scenario, the program should treat the IP the same as unevaluated externally developed 3PIP. The program should follow the guidance provided in TD 5: Adversary compromises third-party soft IP.
- f) The program would like to reuse internally developed IP that was developed to be compliant with LoA1 or LoA2.
- g) Based on the increased threat complexity at LoA3, the program should treat the IP the same as externally developed IP. The program should follow the guidance provided in TD 5: Adversary compromises third-party soft IP.
- h) The program would like to reuse internally developed IP that was developed to be compliant with LoA1 or LoA2 and previously successfully evaluated to be compliant with LoA1 or LoA2.
- i) Based on the increased threat complexity, the program should treat the IP the same as externally developed IP. The program should follow the guidance provided in TD 5: Adversary compromises third-party soft IP.



- j) An LoA3 program would like to use externally developed 3PIP (e.g., v1.1). A different version of the 3PIP (e.g., v1.0) was previously verified to be LoA1, LoA2, or LoA3 compliant.
- k) In this scenario, the IP has been modified. Due to the modification, the program should treat the IP the same as unevaluated externally developed 3PIP. The program should follow the guidance provided in TD 5: Adversary compromises third-party soft IP.
- l) At LoA3, the program would like to use externally developed 3PIP that was previously verified by an independent third party at LoA1, LoA2, or LoA3.
- m) Program independent third party reviews are not acceptable. The program should treat the IP the same as not previously reviewed IP. The program should follow the guidance provided in TD 5: Adversary compromises third-party soft IP.



Appendix C: JFAC FPGA reporting template

Each program is requested to provide the following information to JFAC. Multiple email addresses are provided to support a variety of classification levels; only one email to any of these is required. Please contact JFAC to obtain the appropriate email address at <https://jfac.navy.mil>.

The template and information to be included in the email are as follows:

=====

***** Please Portion Mark Appropriately *****

(U) POC Contact Info

(U) Name:

(U) Organization/Company:

(U) Email:

(U) Phone:

(U) Address:

(U) Program Info

(U) Program Name (top-level program, i.e. F35, M1 tank, etc.):

(U) US Govt Sponsor: (Air Force, Army, Marines, Navy, DOE, other)

(U) Do you want to be included in any future JFAC FPGA Assurance related bulletins in the future?

(U) Estimated Number of Systems to be Built:

(U) Program Description (1-3 sentences describing the top-level program in which the subsystem listed below is included):



(U) FPGA Info (for each FPGA part number used)

(U) FPGA Vendor: (Intel, Lattice, MicroChip, Xilinx, other)

(U) FPGA Device Family:

(U) FPGA Device Part Number:

(U) FPGA Design Software Used and Version #:

(U) Description of Subsystem Containing FPGA Device:

(U) Total Estimated Number of Subsystems to be Built:

(U) Operating Environment: (mil, ind, com, radiation, cryo)

(U) Source/seller of the FPGA devices:

(U) Date purchased:

(U) Anticipated Fielding date:

(U) LoA Level:

(U) Description of FPGA Role in Subsystem. If multiple instances of FPGA devices, number and describe the role of each.

1.

2.

3.

=====

Example

=====



*** Please Portion Mark Appropriately ***

(U) POC Contact Info

(U) Name: **Jack Jackson**

(U) Organization/Company: **Army Research Lab**

(U) Email: **jjackson@army_email.mil**

(U) Phone: **555-555-5555**

(U) Address: **10 Main St, Fort Murphy, Illinois 55555**

(U) Program Info

(U) Program Name (top-level program, i.e. F35, M1 tank, etc.): **Next Generation Combat Vehicle (NGCV)**

(U) US Govt Sponsor: (Air Force, Army, Marines, Navy, DOE, other) **Army**

(U) Do you want to be included in any future JFAC FPGA Assurance related bulletins in the future? : **Yes**

(U) Estimated Number of Systems to be Built: **1400**

(U) Program Description (1-3 sentences describing the top-level program in which the subsystem listed below is included):

The Next Generation Combat Vehicle – Future Decisive Lethality (NGCV-FDL) will have capabilities that are enabled by assured position, navigation and timing and resilient networks. This will enable future maneuver formations to execute semi-independent operations while conducting cross-domain maneuver against a peer adversary.

(U) FPGA Info (for each FPGA part number used)



(U) FPGA Vendor: (Xilinx, Intel, MicroChip, Lattice, other): **Acme MicroElectronics**

(U) FPGA Device Family: **Big Blue Iceberg**

(U) FPGA Device Part Number: **BBI-624L100K**

(U) FPGA Design Software Used and Version #: **IceBreaker V2021.15**

(U) Description of Subsystem Containing FPGA Device: **image processing for data originating from the cannon targeting sensor**

(U) Total Estimated Number of Subsystems to be Built: **3000**

(U) Operating Environment: (mil, ind, com, radiation, cryo): **mil**

(U) Source/seller of the FPGA devices: **Digikey, online**

(U) Date purchased: **2/25/2020**

(U) Anticipated Fielding date: **5/1/2022**

(U) LoA Level: **1**

(U) Description of FPGA Role in Subsystem. If there are multiple instances of FPGA devices, number and describe the role of each one.

1. FPGA #1 – is used to perform signal processing on raw image data coming in from the externally mounted cannon.

2. FPGA #2 – is used to perform signal processing on raw image data coming from the scout drone through the external antennae #2 and synchronized with GPS positioning data.

=====



Appendix D: Mitigations and data/documentation requirements

Checklist for TD 1: Adversary utilizes a known FPGA platform vulnerability

TD 1 mitigations	Data/Documentation requirement
Use caution when selecting tools or platforms	The program should document the name of the person performing the research, the date timestamp of the research, the research results, and the vendor provided end-of-life plan or release notes (if available). If beta/initial release is selected, the program should document the rationale behind the selection and contain the signature of the programmatic approval authority.
Use cleared personnel	In writing, the program should designate work that must be done by cleared individuals. The program should keep a log of personnel assigned to that work along with their clearance level. The program should maintain a list of the members comprising each team, with clearance level. The program should maintain audit logs demonstrating what each team member accessed.
Research vulnerabilities	The program should document each publication that was searched (including at a minimum those identified in this guidance), search results, the name of the person who performed the search, and date timestamp when the search was performed. The same information should be documented by the reviewer.
If a vulnerability is found, choose one of the following options:	
Option 1: Select a different FPGA platform, device, or software	The program should document each publication that was searched (minimally those identified in this guidance should be searched), the search results, the name of the person performing the search, and the



TD 1 mitigations	Data/Documentation requirement
	date and timestamp of when the search was performed.
Option 2: Work with the vendor	The program should work through the vendor process to formally notify the vendor of any vulnerabilities, and only accept fixes through formal, approved processes. The program should maintain documentation regarding the identified vulnerability, log communication with the vendor, and document the source and method of the received fix.
Option 3: Risk analysis	The program should maintain documentation identifying the risk, any mitigations, and the approval authority for accepting the residual risk.
Use revision control/version management	The program should document, maintain, and utilize a program configuration management (CM) plan. This plan should include details on how configuration data will be maintained for control and audit purposes. The system used for CM should be named, and implementation specific details should be documented. The program should document how the CM plan is compliant with NIST SP 800-171 Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations. If a classified system is used, the program should store a copy of the approved SSP. Audit logs should be reviewed with the results recorded.
Enforce auditability	The program should maintain audit logs on all design data, including requirements, architecture, design, code, tests, bugs, and fixes. The audit data minimally should document who requested the change with date and timestamp, the decision made regarding the change, who made the decision with date and



TD 1 mitigations	Data/Documentation requirement
	timestamp, why was the change requested, and who made the change with date and timestamp.
Enforce the approved design process	<p>The program should document program design milestones with clear entry and exit criteria. The entry and exit criteria should be specifically identified to include the peer review/code review and technical review processes. The entrance and exit criteria should be utilized throughout the program lifecycle. The documentation should contain artifacts demonstrating the gates were satisfied, with signed management approval.</p> <p>The program should obtain the results of independent reviews to include:</p> <ul style="list-style-type: none"> • Type and extent of verification performed, to include evaluation objective, methodology, and tools • Findings, both positive and negative, for all evaluations performed • Risks identified by the review team (e.g., quality issues, vulnerability to threats, etc.) • Recommendations to mitigate identified risks • Independent team should be separate from the team doing the design • Identification and credentials of each reviewer • Date and timestamp of when the review was performed

Checklist for TD 2: Adversary inserts malicious counterfeit

TD 2 mitigations	Documentation requirements
Purchase from DoD authorized vendors and distributors	The program should document the name and location of the authorized vendor along with documentation demonstrating that the vendor is authorized.



TD 2 mitigations	Documentation requirements
Consult Government-Industry Data Exchange Program (GIDEP)	The program should document the GIDEP search results, the name or ID of the person performing the search, and the date and timestamp of when the search was performed.
Follow storage and shipping guidance	<p>The program should document, maintain and enforce a transportation plan which supports the movement of bulky classified material. Minimally the plan should include:</p> <ul style="list-style-type: none">• Title of Plan• Date of movement• Authorization/Approval• Purpose• Description of consignment, to include unique ID when available• Identification of responsible government and/or company representatives• Identification of commercial entities to be involved in each shipment• Packaging of the consignment• Routing of the consignment• Couriers/escorts• Recipient responsibilities• Return of material procedures• Other information as required <p>The program should document, maintain, and enforce a storage plan which supports the storage of bulky material.</p>
Verify the FPGA cryptographically secure ID	The program should document and store the ID of each FPGA against the ID that was provided directly by the vendor.
Perform physical inspection/analysis	The program should document the results of the physical analysis test with each FPGA unique ID the test was performed on.



TD 2 mitigations	Documentation requirements
To mitigate risk of a cleared insider:	
Select sample parts	The program should document: <ul style="list-style-type: none"> • The process to secure the device and the results • All parties that touched the device with the reason for the interaction
Create cryptographically protected IDs post verification	The program should record the device serial number and PUF ID. Compare results anytime the programs compares the soft PUF and unique ID for confirmation of the authenticity of the part.
Verify independent lab work	The program should require: <ul style="list-style-type: none"> • The return of residual materials and detailed reports after evaluation • The approved storage plan to be utilized by the lab with acceptable evidence • Documentation that demonstrates the lab identified the known bad parts; the name, address, and division of the two independent labs; or results of physical inspection
In addition to verifying independent lab work above, choose one of the following options:	
Option 1: Insert known bad parts	Document the known bad parts, the problem with the part, and the results from the verification facility that performed the physical analysis.
Option 2: Use duplicate independent labs	Document the credentials of the lab observers, the findings, and conclusion. The conclusion should confirm if the lab results match or are different.
Option 3: Use duplicate persons assigned to the program	Document the credentials of the observers, the findings, and conclusion. The conclusion should confirm if the results match or are different.



TD 2 mitigations	Documentation requirements
Follow guidance for <i>TD 4: Adversary compromises system assembly, keying, or provisioning</i>	Provide all of the <i>TD4: Adversary compromises system assembly, keying, or provisioning</i> data requirements.

Checklist for TD 3: Adversary compromises application design cycle

TD 3 mitigations	Documentation requirements
Use Secret level cleared personnel	<p>In writing, the program should designate work that must be done by cleared individuals. The program should keep a log of personnel assigned to that work with their clearance level.</p> <p>The program should maintain a list of the members comprising each team, with clearance level. The program should maintain audit logs demonstrating what each team member accessed.</p>
Track critical data in a revision control system	<p>The program should ensure the following data items are tracked in revision control:</p> <ul style="list-style-type: none"> • Third-party IP (3PIP) • Utilized libraries • Development files, code, software used for development, synthesis scripts, and tools • Test Benches, Test Plans and Test Procedures, and Test Reports • Tool configuration settings • Design documents to include: <ul style="list-style-type: none"> • Critical documents, to minimally include requirements, design artifacts, test reports, test plans, and discrepancy reports. • Documentation with approval to proceed from organizationally defined reviews: code reviews, architecture reviews, technical design reviews, and verification and validation reviews.



TD 3 mitigations	Documentation requirements
	Each of these artifacts should be identified in the programs auditing strategy and the audit logs should minimally include decisions that were made, by whom, for what reason, and on what date.
Enforce auditability	The program should maintain audit logs on all design data, including requirements, architecture, design, code, tests, bugs, and fixes. The audit data minimally should document who requested the change with date and timestamp, the decision made regarding the change, who made the decision with date and timestamp, why was the change requested, and who made the change with date and timestamp.
Use revision control/version management	The program should maintain revision control documentation in accordance with requirements of CMMC level 3 or NIST 800-171 Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations and NIST 800-172 Enhanced Security Requirements for Protecting Controlled Unclassified Information. The program should maintain the CMMC audit results or NIST 800-171 self-assessments.
TD 3.1 Mitigating the introduction of a compromised design into the application	
Isolate and store the application design	The program should document the hash of the final configuration after the final design and verify the hash prior to provisioning. The program should maintain the configuration management audit logs.
Perform reproducible build	Document the reproducible build process and results validating that the two separate builds produce the same binary and hash.



TD 3 mitigations	Documentation requirements
TD 3.2 Mitigating the modification of test benches/plan to reduce coverage or hide Trojan code	
Execute a documented test plan	<p>The program should document and maintain a test plan that includes a mechanism to verify all requirements.</p> <ul style="list-style-type: none"> • The test plan should explicitly list code coverage metrics, the type of testing that will be performed, and acceptable testing guidelines. • Code coverage should state how much code is checked by the test bench, providing information about dead code in the design and holes in test suites. Ensure code coverage includes statement coverage, branch coverage, Finite State Machine (FSM), condition, expression, and toggle coverage. Document any code that will not be covered and why. Ensure untested code is documented and reviewed through the review process. Use functional tests to verify the FPGA does what it is supposed to do. Any deviations must be documented and approved. • The decision to use/not use other types of testing such as directed test, constrained random stimulus, and assertion should be documented. • Unexpected behavior should be documented and analyzed, with final implementation conclusions documented. • The test plan should specify the verification environment which describes the tools, the software, and the equipment needed to perform the reviews, analysis, and tests. Each of these items should be maintained under revision control. • Ensure all test discrepancies, bugs, etc. are resolved via a change process.
Validate and verify test processes	<p>The program should document, review, maintain, enforce, and archive the test plan. The test plan should include which tools will be used with names, version</p>



TD 3 mitigations	Documentation requirements
	<p>numbers, and the various test reviews that will take place, type of testing to be performed, and the methods used to accomplish the test.</p> <p>The program should maintain documentation of all testing performed, including members of each team and role, all documentation associated with peer reviews, configuration logs indicating all actions taken by whom and when, and use of automated tools where applicable. All test discrepancies, bugs, etc. should be resolved via a change process utilizing a change management system. The established processes should be documented, enforced, and audited.</p>
<p>Maintain test environment via configuration management</p>	<p>The program should maintain configuration management documentation in accordance with requirements of CMMC level 3 or NIST SP 800-171 Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations and NIST SP 800-172 Enhanced Security Requirements for Protecting Controlled Unclassified Information. The program should maintain the CMMC audit results or NIST SP 800-171 self-assessments.</p>
<p>TD 3.3 Mitigating the introduction of Trojans into the application design during development</p>	
<p>Maintain bi-directional link to approved requirements</p>	<p>The program should document bi-directional traceability for all device requirements, including derived requirements.</p>
<p>Enforce peer review</p>	<p>The program should document the results of each peer review to include:</p> <ul style="list-style-type: none"> • Entry criteria and status, • Roles and responsibilities with associated names, • Attendees, • Findings, including deviations or waivers and



TD 3 mitigations	Documentation requirements
	<p>associated rationale and approval,</p> <ul style="list-style-type: none"> • Exit criteria and status.
Execute a documented test plan	<p>The program should document and maintain a test plan that includes a mechanism to verify all requirements.</p> <ul style="list-style-type: none"> • The test plan should explicitly list code coverage metrics, the type of testing that will be performed, and acceptable testing guidelines. • Code coverage should state how much code is checked by the test bench, providing information about dead code in the design and holes in test suites. Ensure code coverage includes statement coverage, branch coverage, Finite State Machine (FSM), condition, expression, and toggle coverage. Document any code that will not be covered and why. Ensure untested code is documented and reviewed through the review process. Use functional tests to verify the FPGA does what it is supposed to do. Any deviations must be documented and approved. • The decision to use/not use other types of testing such as directed test, constrained random stimulus, and assertion should be documented. • Unexpected behavior should be documented and analyzed, with final implementation conclusions documented. • The test plan should specify the verification environment which describes the tools, the software, and the equipment needed to perform the reviews, analysis, and tests. Each of these items should be maintained under revision control. • Ensure all test discrepancies, bugs, etc. are resolved via a change process.
Implement, validate, and verify test processes	<p>The program should maintain documentation of all testing performed, including members of each team and their roles, all documentation associated with peer reviews, configuration logs indicating all actions taken</p>



TD 3 mitigations	Documentation requirements
	by whom and when, and use of automated tools where applicable. All test discrepancies, bugs, etc., should be resolved via a change process utilizing a change management system. The established processes should be documented, enforced, and audited
Select a formal “proof” process	Document all code that was reviewed using LEC, any functional discrepancies, and how those discrepancies were resolved.
TD 3.4 Mitigating the introduction of compromised tooling/software into the environment	
Validate cryptographic hashes	The program should document the value of the calculated cryptographic hash and the signed hash provided by the vendor along with the software name, version, and release number.
Research vulnerabilities	The program should document each publication that was searched, (including at minimum those identified in this guidance) search results, the name of the person performing the search and the date and timestamp when the search was performed.
If vulnerabilities are found in the software or tools, choose one of the following options:	



<p>Option 1: Select a different tool</p>	<p>The program should document each publication that was searched, (including at minimum those identified in this guidance) search results, the name of the person performing the search and the date and timestamp when the search was performed.</p>
<p>Option 2: Work with vendor</p>	<p>The program should maintain documentation regarding the identified vulnerability, log communication with the vendor, and document the source and method of the received fix.</p>
<p>Option 3: Risk analysis</p>	<p>The program should maintain documentation identifying risk, mitigations and approval authority.</p>
<p>To validate tools, choose one of the following options:</p>	
<p>Use a formal “proof” process</p>	<p>Document all code that was reviewed using LEC, document any functional discrepancies and how those discrepancies were resolved.</p>
<p>Use a reproducible build process</p>	<p>The program should document the reproducible build process and results validating the separate builds produce the same binary and hash.</p>
<p>TD 3.5 Mitigating intrusion into the internal network</p>	
<p>Assign Roles</p>	<p>The program should approve, document, and maintain all individuals, the roles they perform, and the access allowed by that role. At a minimum, these roles should include design, test, network administration, and system administration.</p>
<p>Control and monitor access</p>	<p>Entry/access to appropriate areas should be recorded, monitored, and logged for auditability.</p>
<p>Research vulnerabilities</p>	<p>The program should document each publication that was searched, the results of the search, vulnerabilities and/or mitigations if applicable, name of the person</p>



	performing the search, and the date and timestamp of the search.
Use a secret or classified network	The program should maintain documentation and audit data demonstrating a network classified at the DSCA Secret level or above. The documentation should include a log of personnel with clearance information, all records in accordance with a maintaining a DSCA Secret network, as well as a documented and SSP.
TD 3.6 Mitigating risk from compromised hire or employee	
Enforce auditability	The program should maintain audit logs on all design data, including requirements, architecture, design, code, tests, bugs, and fixes. The audit data minimally should document who requested the change with date and timestamp, the decision made regarding the change, who made the decision with date and timestamp, why was the change requested, and who made the change with date and timestamp.
Enforce the approved design process	<p>The program should document and utilize the entry and exit criteria of each stage of the design process. This includes documentation for each peer review and design review with roles and responsibilities along with associated names, attendees, and findings, including deviations or waivers and associated rationale and approvals.</p> <p>All design changes should be documented and approved, and testing should adhere to organizationally approved test standards.</p>



<p>Review critical design activities</p>	<p>The program should obtain the results of independent reviews to include:</p> <ul style="list-style-type: none"> • Type and extent of verification performed, to include evaluation objective, methodology, and tools • Findings, both positive and negative, for all evaluations performed • Risks identified by the review team (e.g., quality issues, vulnerability to threats, etc.) • Recommendations to mitigate identified risks • Independent team should be separate from the team doing the design • Identification and credentials of each reviewer • Date and timestamp of when the review was performed
<p>Use cleared personnel</p>	<p>In writing, the program should designate work that must be done by cleared Individuals. The program should keep a log of personnel assigned to that work along with their clearance level.</p> <p>The program should maintain a list of the members comprising each team with their clearance levels. The program should maintain audit logs demonstrating what each team member accessed.</p>
<p>TD 3.7 Mitigating risk associated with the compromise of device identifiers</p>	
<p>Store device identifiers</p>	<p>Maintain access control logs to include who has access to the device identifiers and who actually accesses the device identifiers.</p>
<p>Limit access to device identifier information</p>	<p>Maintain access control logs to include who has access to the device identifiers and who actually accesses the device identifiers</p>



Checklist for TD 4: Adversary compromises system assembly, keying, or provisioning

TD 4 mitigations	Documentation requirements
Purchase from DoD authorized vendors and distributors	The program should document the name and location of the authorized vendor along with documentation demonstrating that the vendor is authorized.
Follow storage and shipping guidance	The program should document, maintain, and enforce a transportation plan which supports the movement of bulky classified material. Minimally the plan should include: <ul style="list-style-type: none">• Title of Plan• Date of movement• Authorization/Approval• Purpose• Description of consignment, to include unique ID when available• Identification of responsible government and/or company representatives• Identification of commercial entities to be involved in each shipment• Packaging the consignment• Routing of the consignment• Couriers/escorts• Recipient responsibilities• Return of material procedures• Other information as required
Provide keys and configuration data	The program should document assembly house receipt of data packages and the hash value of the packages.
Clear memory devices	The program should document the company, location, individual, and method for clearing the contents along with the contents before and after clearing.



TD 4 mitigations	Documentation requirements
Provision private keys	The program should document: <ul style="list-style-type: none">• The company name, location, and date of provisioning• The number of provisioned devices and number of unique keys used• Proof of DSCA facility classification• Proof of DMEA Trust Category I certification
Protect the configuration data package	The program should maintain data receipt documentation from each of the assembly and test teams showing each team either collected the data from a central repository or received it from a trusted transfer mechanism.
Perform verification activities	<p>The program should maintain documentation including the procedures used to verify the PCB traces, where the work was performed, when it was performed, and the results of the verification.</p> <p>The program should maintain documentation including the procedures used to authenticate the configuration data, where the work was performed, who performed it, when it was performed, and the results of the verification.</p> <p>The program should maintain documentation including the authentication methodology, its architecture, and its compliance with appropriate NIST standards.</p> <p>The program should maintain documentation including the methodology used to verify the proper keys were loaded, where the work was performed, when it was performed, and who performed the work.</p> <p>The program should maintain documentation including the procedures used to authenticate the post assembly FPGA device, where the authentication was performed, by whom, when, and the results of the verification.</p>



TD 4 mitigations	Documentation requirements
Authenticate the FPGA device by choosing one option:	
Option 1: Verify the unique cryptographic ID	The program should document: <ul style="list-style-type: none"> • The authenticity verification method • The verification outcomes • The individual name or reference ID who performed the verification
Option 2: Verify the device on the PCB	The program should document: <ul style="list-style-type: none"> • The authenticity verification method • The verification outcomes • The individual name or reference ID who performed the verification
Option 3: Use a soft PUF	The program should document: <ul style="list-style-type: none"> • The authenticity verification method • The verification outcomes • The individual name or reference ID who performed the verification

Checklist for TD 5: Adversary compromises third-party soft IP

TD 5 mitigations	Documentation requirements
Purchase from DoD authorized vendors and distributors	The program should document the name and location of the authorized vendor along with documentation demonstrating that the vendor is authorized.
Only accept IP that is unobfuscated	The program should keep a copy of the clean unobfuscated code, along with the name and or ID of the person who received it.



TD 5 mitigations	Documentation requirements
Ensure IP deliverable packages are digitally signed	The program should maintain documentation of the vendor provided signature and hash, and the actual software hash.
Validate the cryptographic hash	The program should document the value of the calculated cryptographic hash and the signed hash provided by the vendor along with the software name, version, and release number.
Store IP in a revision control repository	The program should include the initial IP and hash check-in within the system.
Examine IP for malicious functions	<p>The program should document all results in accordance with <u>Third-Party IP Review Process for Level of Assurance 3</u>. This document is available upon request.</p> <p>All interaction with JFAC regarding IP for malicious functions should be documented.</p>
To examine the IP for malicious functions, chose one of the following options:	
Option 1: At least two cleared personnel review the IP	The program should maintain documentation specific to that identified in the <u>Third-Party IP Review Process for Level of Assurance 3</u> .
Option 2: Contact JFAC to determine if an IP review of the complete IP package has been previously completed	<p>The program should maintain documentation of correspondence between the program and JFAC. This should include information about the IP, system the IP is used in, and the role that IP serves within that system, along with proof of receipt from JFAC.</p> <p>The program should obtain and review evidence of IP verification, including requirements sign-off.</p> <p>Note: This activity is intended to both provide confidence that the 3PIP will meet program specifications and that functionality not utilized by the developer, including testability, is understood by the</p>



TD 5 mitigations	Documentation requirements
	program. Data should be created and collected by the IP developer.

Checklist for TD 6: Adversary swaps configuration file on target

TD 6 mitigations	Documentation requirements
Incorporate cryptographic authentication	The program should document: <ul style="list-style-type: none"> • The method used to authenticate the configuration file on load. • The verification process used to test the authentication method.
Authenticate configuration data each time the data is loaded	For each configuration load method used, the program should document the method used to authenticate the configuration file on load, and the verification process used to test the authentication method.
Prevent direct read back	The program should document the steps taken to prevent direct read back of private keys.
Use a CNSS/NIST approved algorithm and key length	The program should document the key length being used along with the version number of the latest CNSS or NIST FIPS guidance approved key length.
Use DoD evaluated authentication mechanisms	The program should maintain documentation from JFAC with the security evaluation results.
Disable test access pins	The program should maintain documentation including the means by which the JTAG test pins were disabled.
Ensure authentication for modifications	Document if the FPGA allows application changes, how the vendor states authentication will apply to all reconfiguration data, and test results indicating how



TD 6 mitigations	Documentation requirements
	authentication was actually applied to all reconfiguration data.
Always program security settings in non-volatile storage of the device	The program should maintain documentation including the means used to set security settings.
When a platform supports remote updates, chose one of the following options:	
Option 1: Validate that the built-in application change technique fully applies authentication to all the reconfiguration data	The program should maintain documentation including the test used to validate the application update methodology and the outcome.
Option 2: Perform authentication of the reconfiguration data in the application	The program should maintain documentation including the methodology used to perform authentication in the application using partial reconfiguration.
Use a FIPS compliant 140-2 Level 2 HSM	Document how the program utilizes FIPS 140-2. Document the HSM that is being used and the spec sheet demonstrating FIPS compliance.

Checklist for TD 7: Adversary substitutes modified FPGA software design suite

TD 7 mitigations	Documentation requirement
Purchase from DoD authorized vendors and distributors	The program should document the name and location of the authorized vendor along with documentation demonstrating that the vendor is authorized.
Prevent automatic tool updates	The program should document, maintain, and follow the SSP.
Use a trusted computing environment	The program should maintain documentation and audit data demonstrating one of the following computing environments was used: <ul style="list-style-type: none"> • A computer and network classified at the DSCA



TD 7 mitigations	Documentation requirement
	<p>Secret level or above. The documentation should include a log of personnel with clearance information, all records in accordance with a maintaining a DSCA Secret network, as well as a documented and SSP.</p> <ul style="list-style-type: none"> • A computer and network certified for use in a Trust Category 1 facility as defined by DMEA. • A network-isolated computer enclave with limited and controlled access, adhering to NIST and CMMC standards.
Use cleared personnel	<p>In writing, the program should designate work that must be done by cleared Individuals. The program should keep a log of personnel assigned to that work along with their clearance level.</p> <p>The program should maintain a list of the members comprising each team, with their clearance levels. The program should maintain audit logs demonstrating what each team member accessed.</p>
Validate the cryptographic hash	The program should maintain the value of the calculated hash and the hash that is provided by the vendor, along with the version/release number and date/timestamp.
To validate the tool output, choose one of the following options:	
Option 1: Perform a logical equivalency check	Document all code that was reviewed using LEC, any functional discrepancies, and how those discrepancies were resolved.
Option 2: Use a reproducible build process	Document the reproducible build process and results validating that the two separate builds produced the same binary and hash.



Checklist for TD 8: Adversary modifies FPGA platform family at design

TD 8 mitigations	Documentation requirements
Engage JFAC	The program should maintain a copy of the data sent to JFAC with a date/timestamp of when it was sent and an acknowledgement of when it was received.

Checklist for TD 9: Adversary compromises single-board computing system (SBCS)

TD 9 mitigations	Documentation requirement
Engage a DoD vendor to build the SBCS	The DoD vendor should provide functionality and product specifications.
Verification and authentication	The program should maintain a list of the members comprising the independent verification team, with their clearance levels. The program should maintain audit logs demonstrating what each team member accessed, when and what reviews were conducted, and each device that was verified.
Authenticate the FPGA devices	The program should document the physical inspection results for each slash sheet and unique ID for the device inspected.
Verify the SBCS configuration process	Document the SBCS configuration process and how it complies with the LoA3 mitigation requirements for <i>TD 6: Adversary swaps configuration file on target</i> . This includes, but is not limited to, requirements for: <ul style="list-style-type: none"> • NIST compliant authentication algorithms • Differential power analysis (DPA) resistant authentication • Protected key storage • Anti-tamper detection and response • Being free of known vulnerabilities in the configuration and security functions



TD 9 mitigations	Documentation requirement
	<ul style="list-style-type: none"> • All encryption and authentication keys lengths must be compliant with the requirements outlined NIST SP 800-57 • The ability to disable FPGA test pins, such as JTAG <p>If the configuration file memory storage device contains SBCS vendor code, the program should review and evaluate that code for malicious functions, and document how the review was conducted and any findings. The proprietary SBCS support for configuration must be fully understood and validated. If the SBCS configuration process cannot be fully evaluated, it should not be used at LoA3.</p> <p>Once the SBCS's configuration design and implementation are evaluated to be free of malicious functions, the program should craft a set of tests and validation processes to verify that all the devices comply with the evaluation. The program should document the tests and validation processes along with the validation of all devices.</p>
Test non-volatile memory	<p>The program should maintain documentation including the FPGA settings available in the given FPGA device, the methodology used to read them, where they were tested, by whom, when and the results.</p>
Document the steps	<p>Document the steps taken to comply with these requirements. This includes all hardware and software that were authenticated and verified. All associated data artifacts should be auditable.</p>



Checklist for TD 10: Adversary modifies vendor FPGA software design suite during development

TD 10 mitigations	Documentation requirement
Perform all FPGA design work on an isolated network	Provide documentation in alignment with <i>Checklist for TD 3: Adversary compromises application design cycle.</i>
Perform logical equivalency checking	The program should document any hints, all optimizations, and rationale for any logic that did not match the equivalency checker with managerial approval signature.