National Security Agency
Cybersecurity Technical Report

DoD Microelectronics:

Field Programmable Gate Array

Level of Assurance 2 Best Practices

February 2023

U/OO/120910-23
PP-23-0199
Version 1.0

For additional information, guidance, or assistance with this document, please contact the Joint Federated Assurance Center (JFAC) at https://jfac.navy.mil.

Notices and history

Document change history

Date	Version	Description
FEB 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

Joint Federated Assurance Center: https://jfac.navy.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 2 (LoA2). LoA2 captures the threats most likely to be exercised against a DoD system based upon their cost and high value of return. This level is defined as causing serious harm to U.S. personnel, property, or interests if the systems fails. At this level, these threats have the following characteristics:

- **Access** – Exploit a difficult point of access or multiple points of access. Difficult includes a single cleared insider.
- **Technology** – Use of technology with low risks from implementation. Technology may not be accessible commercially but proven technology.
- **Investment** – Require a large multidisciplinary team.
- **Value of effect** – Establish vulnerabilities for future exploitation.
- **Targetability** – Affect only a subset of systems.

Organized by threat, this report provides multiple technical mitigations to choose from to address each threat and to allow the user 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
TD 8	Adversary modifies FPGA platform family at design

#	Threat description (TD)
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. Appendix C, “Checklists and Data Requirements” contains a quick reference list of threats, associated mitigations, and associated documentation.

Once the program has mitigated these threats, they have achieved an assurance level of LoA2.

Contents

Tables

1 Overview of Level of Assurance 2 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 2 (LoA2). 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, LoA2 requires mitigations against FPGA assurance threats that have the following characteristics:

- **Access – A difficult point of access** requires the adversary to compromise a system or an individual to circumvent extensive practices taken to protect that access. This is defined by the following list:
- A single air-gapped computer network
- A single cleared U.S. person
- A group of uncleared U.S. persons, such as a small corporate office operated in the U.S.
- Shipping practices that are documented and program approved
- Threats may take advantage of multiple available points of access.

For a mitigation based on access to be effective, it needs to raise the access required to carry out the attack to one necessitating multiple points of difficult access, either in differing areas of the supply chain or of multiple personnel in the supply chain.

- **Technology – Low implementation risk technology** includes any technology which, while not publicly available now, could be implemented with sufficient effort and minimal risk of outright failure. These include:
- Capabilities that public academic research has identified, or internal U.S. Government research and development has shown to be practical.
- Techniques that have, according to substantial amounts of open research, been performed successfully, but for which commercial tools are not available.
- In addition, these threats may take advantage of existing public technology.

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 large multidisciplinary team** indicates that the team conducting the operation may draw on multiple skills not necessarily directly associated with the Custom Microelectronic Component. A large multidisciplinary team is defined as any effort consisting of roughly fifty person-years of a wide range of technical expertise, focused solely on attacking the device of interest. For example, physics and materials science experts may have suitable technical skills. This level also accounts for a more substantial amount of effort, potentially a sizeable organization working on the attack for a year or more.

For a mitigation based on investment of resources to be effective, it must force the attacker to expend greater resources in the form of engaging the resources of a nation across a wide scope to facilitate an attack.

- **Value of Effect – Establish vulnerabilities** pre-positions a change that is not by itself a complete attack and requires additional access and technical development to complete an attack. Additionally, attacks that disable or subvert capabilities are included at LoA2.

To be effective, a mitigation based upon value of effect to the adversary must constrain the severity of the outcome on the target to one of lesser effect. In this case, the mitigation must limit the LoA2 attack to degradation of performance of the device in a general non-targeted manner.

- **Targetability – The attack affects only a subset of systems**, and the adversary has no control over that subset. For systems that are affected, the behavior must still be inherently targetable and controllable.
- In addition, any inherently targetable and controllable attacks from LoA1 are relevant at this level.

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

For a program to achieve Level of Assurance 2, it must provide mitigations against threats that possess these characteristics. Of prime importance in LoA2 is the assumed presence of a compromised cleared insider. This new condition renders 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 through the use of dual or independent teams. LoA2 addresses threats that originate from an adversary whose intent is malicious and does not cover commercial assurance risks such as re-marked parts. Economically motivated assurance threats have reliability risks associated with them. 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 LoA2. Each threat is explained and accompanied by examples in more detail within the JFAC *FPGA Best Practices – Threat Catalog.*

Table 1: LoA2 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)

#	Threat description (TD)
TD 6	Adversary swaps configuration file on target
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 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 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 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
- DoD guidance for storage of Secret materials can be found in DODM 5200.01-V3
- 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. 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 2 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. 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 LoA2 compliance, each program should perform each mitigation listed in the “TD # mitigations” section. The “TD # mitigation descriptions” section provides details for each mitigation. In some cases, the full description contains additional options that are required to be LoA2 compliant. An asterisk “*” next to any mitigation indicates additional options must be implemented.

When mitigations for all the threats listed under LoA2 are completed, that device can be said to have achieved LoA2. However, compliance with LoA2 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 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, and
- 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 LoA2. Absent any changes in these areas, the devices should be considered to have achieved LoA2.

1.4 General Comments on Mitigations

- Programs are encouraged to own as much of the fabrication 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 utilize 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.

2 Threat descriptions (TD)

TD 1: Adversary utilizes a known FPGA platform vulnerability

In this threat, an adversary uses a known vulnerability in an FPGA platform or vendor development software package to initiate an attack that is not specific to a program or system. A known vulnerability is an unclassified published weakness in the design of a specific FPGA platform or software program that would allow an adversary the ability to use it for malicious purposes. This threat does not focus on a particular vulnerability but is any weakness in the FPGA device. These vulnerabilities are published in public databases, such as the “Common Vulnerabilities and Exposures (CVE)” and the “National Vulnerabilities Database (NVD)”, vendor advisories, errata bulletins, etc. Such vulnerabilities could allow for leakage of sensitive information or keys; allow for the compromise of security or tamper detection functions; or allow the unauthorized reconfiguration of the product. This threat can be introduced by a program not performing vulnerability research, an insider not disclosing the fact of the vulnerability, modifying data to cover up the existence of a vulnerability, or adding/modifying design features for use with 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 existing in previous 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.

- Enforce approved design process that has clear and 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. End-of-life devices may not have support to mitigate vulnerabilities once identified.

Use cleared personnel

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

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 Vulnerability 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.

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 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.

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 recommend mitigations for a particular vulnerability.

Note: If a vulnerability is identified, a program should report it to the Government Industry Data Exchange Program (GIDEP) and 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 management systems are in place. All changes to the system or 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.
- All changes to the system or artifacts should be documented, approved, and auditable.

Enforce auditability

The program should maintain audit logs on all design data to include requirements, architecture, design, code, tests, bugs, and fixes. The audit data minimally should document who requested the change with date timestamp, what decision was made regarding the change, who made the decision with date and timestamp, why the change was requested, and who made the change with date timestamp.

Enforce approved design process

To prevent a compromised insider from hiding a vulnerability ensure all critical activities are identified and documented. Ensure the entire 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 all vulnerability activities, including identification of vulnerabilities and the appropriateness of the mitigations.

TD 2: Adversary inserts malicious counterfeit

In this threat, an adversary with access to a fabrication process for manufacturing counterfeits inserts additional logic in the FPGA die for their malicious purposes. This includes counterfeit parts made in an unauthorized fabrication facility and inserted into the supply chain, as well as, counterfeit parts made in an authorized fabrication facility through the compromise of the manufacturing process. In either scenario the adversarial intent is to insert a malicious function into the package of an authentic device.

Additionally, this threat assumes that there is a compromised cleared insider in the program. Adversaries may use cleared insiders to introduce malicious counterfeit parts at any point in the product manufacturing cycle or may compromise a piece of the FPGA device verification process. Overlapping checks are therefore necessary for each threat commensurate to this level of assurance.

Insertion of counterfeit parts into the supply chain can happen at any point in 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.

Compromising the manufacturing process in an authorized facility in order to make and insert counterfeit parts could happen during any of the following phases of the manufacturing 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

Insertion of a malicious function into the package of an authentic device 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.

These challenges are addressed with a combination of:

- Physical device inspection,
- Overlapping personnel and multi-party review in the verification process, and
- Cryptographically protected IDs.

The LoA2 mitigations for this threat rely heavily on a physical inspection of the parts. Physical inspections are an extensive counterfeit detection approach used to identify a counterfeit device from an unauthorized fabrication facility versus a device from an authorized fabrication facility. JFAC relies on substantial physical analysis to address these threats for the following reasons:

- The program has no positive control over the fabrication facility or its processes,
- JFAC can identify numerous technically feasible attacks for all fabrication countermeasures considered, and
- Most FPGA fabrication facilities are foreign owned. They are not loyal to the goals of the United States, and are not controllable by the program or DoD.

While commercial (non-malicious) counterfeits such as re-marked parts may represent a reliability risk, they are not included under levels of assurance. Those counterfeits are not malicious by design, not controllable/targetable, and are economic in nature. Programs with specific reliability requirements should plan for the appropriate level of testing to verify that their design and components meet those goals.

TD 2 mitigations

- Purchase from DoD-authorized vendors and distributors. The DoD program acquisition group can provide this information.
- 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 for FPGA devices.
- Verify FPGA cryptographically secure identifiers (IDs) against information sent by the vendor (not the authorized distributor).
- Using AS6171, 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

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

Consult GIDEP

GIDEP provides technical data compiled by government and industry 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 should be 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 FPGA cryptographically secure identifiers

The use of this type of device ID mitigates the threat from counterfeit parts made in an existing, unauthorized fabrication facility as described at the beginning of this threat description. 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, they provide an extremely cost-effective method to detect counterfeits both at acquisition and throughout the FPGA device’s lifecycle in a system. The two biggest advantages of such mechanisms are the ability to validate a device remotely and the ability to non-destructively re-validate a device at any time.

While remote attestation cannot be used during acquisition and assembly, it can be used throughout the lifecycle of the device. This provides the possibility of devices and their configurations being 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.

Multiple devices currently provide a mechanism for remote attestation; however, remote attestation is not approved for initial counterfeit screening. 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, verifying that each specific device is the one connected and cryptographically verified.

In contrast to physical anti-counterfeiting techniques, properly implemented cryptographic 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 either remotely or through local access, such as a board header. 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 board.

Implementation details matter when validating the authenticity of an FPGA device. Each FPGA vendor offers their own authentication approach and each FPGA platform offers a unique variation. In no case is a fully readable ID acceptable. Instead, cryptographically protected IDs include cases where the device possesses a specific private cryptographic key. The device ID in these cases 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.

The following points describe at the highest level the specific criteria required for an appropriate device ID to support anti-counterfeiting:

- Cryptographically protected IDs must utilize a private asymmetric key for which no read function exists. This must use a CNSS Policy compliant algorithm if it is a National Security System. If CNSS is not a program requirement, the program should use a National Institute of Standards and Technology (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 vendor-provided public key through decryption of the nonce.

Protect identifier delivery data

For delivery, the vendor must provide this information to the program using a NIST-approved authentication algorithm to transmit the data. Examples would be an elliptic-curve cryptography (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 with protections appropriate to critical protected information. This should include restricted role-based access on a network that is compliant with NIST SP 800-171, *Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations.*

Perform physical inspection/analysis

In LoA2, two new attacks are introduced under TD 2:

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

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

Physical 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 (known good) 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 physical analysis and recommendations on how to execute them are contained in the commercial standard document, SAE International AS6171 *Test Methods Standard; General Requirements, Suspect/Counterfeit, Electrical, Electronic and Electromechanical (EEE) Parts*.

Cleared persons or a lab independent of the program or its performers should carry out these inspections.

The steps of a physical analysis are conducted from least destructive to most destructive and determine if 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.

Each AS6171 test is detailed in a separate document called a “slash sheet.” The following table lists the slash sheets that comprise the AS6171 standard. Users should ensure to use the latest version of AS6171 and associated slash sheets.

Table 2: List of AS6171 slash sheets

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

Test Number	Description
AS6171/11	Techniques for Suspect/Counterfeit EEE Parts Detection by Design Recovery Test Methods

For LoA2, the program should follow the lot sampling guidelines found in the AS6171 document and perform the tests defined by slash sheets 1 through 11.

Sheets 1 through 10 should uncover a counterfeit fabricated in an unauthorized facility or a malicious package insert. Sheet 11 should uncover a counterfeit fabricated in the authorized fabrication (fab) facility.

If the device family possesses cryptographically protected IDs:

- Perform slash sheets 2 and 3 that incorporate a visual inspection and 3D x-ray. This effort analyzes the parts for an additive malicious manufacturing insert 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 through 10 to detect an in-package malicious insert and a die manufactured in an unauthorized fab.
- 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 fab. This test may be limited to a single device.

Slash sheet 11 provides instructions for performing a full delayering, imaging of die, and comparison against a vendor-provided GDSII or an exemplar device. Ideally, an exemplar part is one provided by the FPGA vendor directly to the program for this specific purpose. However, if that is not possible, a device purchased from a vendor-authorized distributor would be acceptable. Preferably, the distributor is a different one from where the product lots were obtained. However, utilize DoD-authorized vendors for all purchases. At LoA2, a full tear down is not necessary, nor even possible in some cases. Instead, JFAC recommends the following procedures for carrying out slash sheet 11.

When process geometries are beyond the state of the art in reverse engineering, contact JFAC for guidance.

Full backside delayering

This includes imaging and comparison of layers active, poly, contact, and metal 1 (M1). This is the ideal option for detecting malicious changes.

Windowing

When full delayering is not an option, the program should:

- Use a plasma focused ion beam (FIB) to expose windows on the backside to examine areas of interest on layers active, poly, contact, and M1, or
- Expose windows on the die front side to examine areas of interest on all layers from passivation down to M1. The program should target windows of approximately 200x200µm.

At each layer, the structures should be compared with information from the FPGA vendor or with an exemplar device.

The windows should focus on non-fabric areas of the die. JFAC recommends that the windows target areas of interest to an adversary including, but not limited to:

- Cryptographic functions
- Microprocessors
- Transceiver logic
- Tamper sense and response logic
- Configuration logic
- Input/output (I/O) programming logic
- Key storage elements

If the program is unable to determine the physical location of these sensitive areas of interest, it should target every feature type. In this case, a visible examination is made of the die, noting areas of different structures and macro types. These areas could include various logic areas, memory macros, hard analog cells, I/Os, fabric, etc. Each one should be examined using a plasma FIB window and compared against the same location on an exemplar device.

- The windows should comprise no less than 1% of the die surface area.

In the case of FPGA multi-chip modules,

- All the dies should be examined using this technique, and
- 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.

Mitigate risk of a cleared insider

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

- *Select sample parts bound for physical inspection in ways that specifically defeat insider compromise.
- *Verify independent lab work using overlapping personnel and multi-party review.
- Create cryptographically protected IDs post verification.
- 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:

Option 1: Multiple independent parties handle part selection before shipping. They should physically verify that the parts selected make it all the way to the physical inspection processes.

Option 2: An independent party verifies sampling before shipping, and multiple parties verify upon receipt that the right parts were received.

Option 3: Use a non-human random selection automated process for sampling.

Verify independent lab work

If a 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 residual 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, 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.

Create cryptographically protected IDs post verification

Following the physical verification above, JFAC recommends using soft physically unclonable functions (PUF) to protect the authentic parts from being swapped out 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.

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 function.

At LoA2, it is assumed that multiple uncleared persons may be working in conjunction with the adversary. The uncleared persons may hold different positions within the supply chain. The actors could be working independently, with each other, or with a cleared insider. Using cleared personnel does not completely mitigate this threat, as the insider may be a single cleared person.

This section describes mitigations that should be taken for the overall threat, as well as mitigations that are associated with the following specific scenarios:

- 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 revision control that hides code or test bench modification (associated mitigations are captured in the “TD 3 mitigations” section below for all sub-threats), and
- Introduction of modified configuration data after generation (associated mitigations are captured in the “TD 3 mitigations” section below for all sub-threats).

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 best practices should be used to assure an FPGA application design at LoA2 and are applicable to all the sub-threats identified in this section:

- Track critical data in a revision control system.
- 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.
- Use revision control and version management tools that meet the requirements described later in this section.

TD 3 mitigation descriptions

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 intellectual property (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

The program should maintain audit logs on all design data to include requirements, architecture, design, code, tests, bugs, and fixes. The audit data minimally should document who requested the change with date timestamp, what decision was made regarding the change, who made the decision with date and timestamp, why the change was requested, and who made the change with date timestamp.

Use revision control and version management tools

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 utilized.
- All changes to the system or artifacts should be documented, approved, and auditable.

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 both configuration management and the revision control system will help prevent the adversary from making unmonitored changes.

At LoA2, programs should ensure no changes have been made to the application design. This can be done by isolating the deployable version with the associated hash

and verifying the hash before loading. Alternatively, this can be done by performing a reproducible build and verifying the results of the verified version against the results of the "to be loaded" version. The verification should produce the same results.

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 files can only be accessed via authentication of two distinct parties. No single individual should be able to access the configuration file and the stored value of the hash. The limited set of people with access should have to follow access control procedures such that access is controlled, monitored, logged, and auditable. Before the file is loaded, the hash should be recalculated and compared against the stored hash.

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 hardware descriptor language (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 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, analyses 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, similar to that of 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 requirements.
- 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, analyses, 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, 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.
- Ensure all test discrepancies, bugs, etc., are resolved via a change process.

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.

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 link to 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 link to 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 practices with the following:

- 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 code uses clear names for everything.
    - Comments are clear and useful, and mostly explain “why” instead of “what”.

Execute a documented test plan

The programs 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.
- 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, analyses, 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, ensuring they deliver the expected output.

Descriptions

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.

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, only accept formal releases, do not accept custom beta fixes, custom patches, etc. for incorporation.

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, 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: Utilize a reproducible build process to generate any deployable configuration files.

Option 2: Select a formal "proof" process that can validate the equivalency of the HDL and final configuration file.

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). At LoA2 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.

Select a formal "proof" process

Use logical equivalency checking (LEC) to the greatest extent possible. LEC 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, JTAG settings, and key storage.

There are technical challenges associated with performing logical 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 3.5 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 intentions, such as modifying tools, swap files, etc.

Mitigations

- Assign roles.
- Control and monitor access based on job requirements including use of physical 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 vulnerabilities. Alternatively, perform a risk assessment and coordinate findings with JFAC.
- Purchase from DoD-authorized vendors and distributors per DoD guidance.
- *Use trusted computing environments 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. 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.

Purchase from DoD-authorized vendors and distributors

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

Use trusted computing environments

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 Defense Security Cooperation Agency (DSCA) Secret level or above.

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

Option 3: A network-isolated computer enclave with limited and controlled access. 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.

Option 4: An infrastructure minimally compliant with NIST SP 800-171 and NIST SP 800-172, preferably compliant with Cybersecurity Maturity Model Certification (CMMC).

TD 3.6 Mitigating risk from a compromised hire or 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. At a minimum, audit data includes what decisions were made, by whom, for what reason, and on what date.
- Adhere to an approved design process.
- Identify, document, and review critical activities. These items should be reviewed by a cleared individual that is different than the original designer.
- Use cleared personnel to perform the design work in an environment certified to handle classified material at the Secret level or higher by DSCA. This would also include design centers certified for Trust Category I by the DMEA.

Descriptions

Enforce auditability

The program should maintain audit logs on all design data to include requirements, architecture, design, code, tests, bugs, and fixes. The audit data minimally should document who requested the change with date timestamp, what decision was made regarding the change, who made the decision with date and timestamp, why the change was requested, and who made the change with date timestamp.

Adhere to an approved design process

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 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. 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 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 a cleared insider working maliciously in some portion of this manufacturing process. At LoA2, this insider could be working alone or in partnership with an external party to influence the outcome. The mitigations for this threat are built upon the following premises:

- All assembly work requires after-the-fact validation in a cleared facility.
- If assembly work is conducted in a classified facility, a single verification team independent of those who conducted the assembly work, can do the post-fab validation. In this case, the cleared insider can compromise the device, or the validation process, but not both, which would be necessary to enable the threat.
- If assembly work is conducted in an external (i.e., third-party unclassified) facility, then two independent cleared teams made up of different individuals can do the validation process. In this scenario, the cleared insider could compromise the validation process to allow compromise while the parts were outside the control of the program. However, the dual nature of the validation prevents this from happening.

Therefore, this document recommends mitigations for the two paths: assembly in a classified facility or inclusion of an external 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 for FPGA devices.

- 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 for 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 and distributors for all purchases. Authorized vendors can be located through the acquisition organization.

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 should be 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 verify 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 in place to provide the configuration data to each entity. 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.

If all the work is conducted in a classified setting, a group independent from the assembly and provisioning team should be utilized to conduct the validation. As LoA2 assumes a single compromised insider, this mitigation strategy means that the insider can only be in either the assembly team or the validation team, but not both.

If any of the work is done in an external facility, use two separate groups independent from the assembly process to conduct the validation. As LoA2 assumes a single compromised insider, this mitigation strategy means that the insider can be in only one of the validation teams.

No matter where the work is performed the following actions must be completed:

- 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.
- 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. A team independent of the assembly and provisioning process must perform this verification. 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 using cryptographic authentication of all loaded configuration data as part of the system containing the FPGA. 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. A team independent of the assembly and provisioning process must perform this verification. Some FPGA devices allow a hash of the keys to be read out for confirmation. Additionally, the program can create test bitfiles to

verify that the devices can properly use the keys and can reject actions using the 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 in Perform physical inspection/analysis under TD 2: Adversary inserts malicious counterfeit. 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 vendor's development cycle, during its delivery or while 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 probably impacts include performance, power, and reliability. The mitigations to these attacks focus on verifying integrity of the delivery of the IP and reviews of its HDL code.

TD 5 mitigations

- Purchase from DoD-authorized vendors and distributors.
- Only accept IP that is unobfuscated and distributed as source code.
- 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 hashes 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.

Validate the cryptographic hash of the IP

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 version control. The hash of the IP should be verified at various stages to ensure there have been no modifications.

Examine IP for malicious functions

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

Option 1: Using JFAC guidance captured in *Third-Party IP Review Process for Level of Assurance 2,* two cleared personnel review the IP. JFAC can provide this document upon request.

Option 2: Contact JFAC to determine if an IP review of the complete IP package has been successfully completed.

Note: Although the results may not be perfect, it is good practice to utilize Trojan detection tools. Contact JFAC for more information.

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 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.

For legacy devices that do **not** have authentication functions, mitigations lean on external authentication functions or the use of symmetrically encrypted data files. In this

case, 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, 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. The specifics of this authentication vary greatly, with some older devices relying on symmetric cryptography rather than asymmetric cryptography. Older devices supported encryption but not authentication, leading some organizations to confuse the two.

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 authentication[1].

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.

At LoA2, the program must use NIST-approved asymmetric cryptographic algorithms.

To achieve LoA2, 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 LoA2, the recommended method for authenticating the data source is to use an asymmetric algorithm recommended by NIST. Asymmetric algorithms are preferred because they do not require the protection of a secret key. For data integrity, a hashing algorithm, such as the secure hashing algorithm (SHA), is recommended. Many of the existing FPGA devices provide these functions for the user.

This level of assurance addresses the compromise of a single cleared insider. That insider can introduce malicious counterfeit parts at any part of the device creation lifecycle or can compromise any single part of the validation process.

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.

TD 6 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 to prevent direct read back of the private keys through electrical means.
- Use a CNSS/NIST approved algorithm and key length. CNSS for National Security Systems; 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 security-evaluated authentication mechanisms.
- Disable operation or use of test access pins in fielded products.
- *When the program selects mechanisms that allow application modifications, ensure authentication for modifications is enabled following the required NIST standards.
- 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. In addition, the program should maintain documentation including the authentication methodology, its architecture, and 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 security-evaluated authentication

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 verified for 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 2[2].
- The use and operation of application test access is disabled in fielded products.

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 JTAG pins and dedicated test pins.

JFAC recommends disabling operation or use of these test access pins 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 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

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

reprogramming via external storage. Verify 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, these mechanisms **do not** provide the same degree of authentication that the primary programming mechanisms provide. Under these best practices, an application designer using these techniques must either validate that the technique they use applies the authentication scheme described below to all the configuration data or perform authentication of this data in the application itself.

Authenticate 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.

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 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 to the application to perform.

Generate and store all authentication keys on a program-controlled, FIPS 140-2 compliant, Level 2 HSM

Generate and store all authentication keys on a program-controlled, FIPS 140-2 compliant, Level 2 Hardware Security Module (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. To accomplish this, 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 for malicious purposes. That is covered by *TD 10: Adversary modifies vendor FPGA software design suite during development.*

TD 7 mitigations

- Purchase from DoD-authorized vendors and distributors. 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.
- Use cleared personnel with at least a Secret level clearance.
- *Validate the cryptographic hash 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 with the source of the vendor-provided hash and the actual software hash.

TD 7 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.

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 Defense Security Cooperation Agency (DSCA) Secret level or above.

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

Option 3: A network-isolated computer enclave with limited and controlled access. 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.

Option 4: An infrastructure minimally compliant with NIST SP 800-171 and NIST SP 800-172, preferably compliant with Cybersecurity Maturity Model Certification (CMMC).

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 deliverable 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 with the source of the vendor provided hash and the actual software hash.

When the hash of the software deliverable does not match the expected authorized hash value, work with the vendor to understand what caused the mismatch and take corrective action. If a resolution cannot be found, use a different software package in which the hash can be validated. All non-hash-validated software is a risk.

When the hash is not provided **choose one** of the following options:

Option 1: Independently validate the hash, ensuring at least two cleared individuals separately validate the cryptographic hash of two separately purchased versions of the tool.

Option 2: Select a formal "proof" process to perform logical equivalency checking between the application HDL and final configuration data.

Option 3: Use a reproducible build process to validate the software.

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, 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.

Use a reproducible build process

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.

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 of choice 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. 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 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 B: JFAC FPGA Reporting Template for the information a program should include in the e-mail.

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 an 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. Additionally, their relative technical simplicity and low expense do not justify the fabrication of custom solutions by programs.

In this scenario, since the program does not have control of the manufacturing process of the SBCS it is forced to rely on a verification-heavy approach to mitigating attacks. 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

- Purchase from DoD-authorized vendors and distributors. The DoD program acquisition group can provide this information.
- Verify and authenticate with independent teams.
- Authenticate the FPGA devices.

- Verify PCB connections using the SBSC's schematic.
- Verify the SBCS configuration process and board-level connections comply with LoA2 mitigation requirements.
- If the configuration file memory storage device contains SBCS vendor code, the program should review and evaluate SBCS vendor code for malicious functions.
- Poll FPGA settings captured in non-volatile memory, such as fuses, to determine if the SBC vendor has preprogrammed any settings in a manner conflicting with these assurance guidelines or that conflict with the user application needs.
- Document compliance steps taken to comply with these requirements. This includes hardware and software features.

TD 9 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.

Verify and authenticate with independent teams

All verification and authentication steps should be conducted by two independent teams. Personnel cannot overlap between teams and each team should be comprised of personnel cleared to the Secret level at a minimum. These teams are in place to eliminate the influence of a cleared insider working with an outside adversary.

Authenticate the FPGA devices

In this mitigation, the program should authenticate the devices utilizing the recommendations found under *TD 2: Adversary inserts malicious counterfeit*.

Verify PCB connections

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. Once the SBCS has been evaluated for PCB connections and correct configuration, the program should craft a set of tests that verify all the devices comply with the evaluation. This set of tests should be maintained and stored in the revision management system.

Verify the SBCS configuration process

The SCBS should be compliant with LoA2. This includes, but is not limited to, requirements for:

- NIST-compliant authentication algorithms
- Differential power analysis 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 Cryptographic Key Management Guidelines*
- The ability to disable FPGA test pins, such as JTAG

Review and evaluate SBCS vendor code

The program should review and evaluate all vendor code contained within the configuration file memory for malicious functions. While reviewing the code, the reviewer should ensure any proprietary SBCS vendor methodology for configuration is fully understood and validated. SBCS configuration processes that cannot be fully evaluated should not be used at LoA2.

Poll FPGA settings captured in 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 compliance 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 large 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 Isolate and store the application design mitigation under TD 3.
- 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 LoA2.

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: JFAC FPGA reporting template

Each program is requested to provide the following information to JFAC. Multiple e-mail addresses are provided to support a variety of classification levels; only one e-mail 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 C: Mitigations with 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, the vendor-provided end-of-life plan or release notes (if available). If a 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 their clearance levels. The program should maintain audit logs demonstrating what each team member accessed.
Research vulnerabilities	The program should document each publication that was searched, (minimally those identified in this guidance should be searched) search results, the name of the person performing 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	For the different tool, the program should document each publication that was searched, (minimally those identified in this guidance should be searched) search

TD 1 mitigations	Data/Documentation requirement
	results, the name of the person performing the search, and date timestamp 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 and enforce a configuration management (CM) plan that is compliant with 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 document how the CM plan is compliant with the requirements. The configuration management plan should include details on how configuration data will be maintained for control and audit purposes. It should include management of document/data, releases, backups and archives, refresh of backup media, retention of tools and software, test equipment, and the test environment. Audit logs should be reviewed with the results recorded.

TD 1 mitigations	Data/Documentation requirement
Enforce auditability	The program should maintain audit logs on all design data to include requirements, architecture, design, code, tests, bugs and fixes. The audit data minimally should document who requested the change with date 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 approved design process	The program should document program design milestones with clear entry and exit criteria. The entry/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 that the gates were satisfied, with signed management approval. The program should obtain the results of independent reviews to include: • 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 • Identification and credentials of each reviewer • Time/date stamp of when the review was performed The independent review team must be separate from the team doing the design.

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.
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 timestamp when the search was performed.
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: • 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 The program should document, maintain, and enforce a storage plan which supports the storage of bulky material.

TD 2 mitigations	Documentation requirements
Verify FPGA cryptographically secure IDs	The program should document and store the ID of each FPGA, the ID that was provided directly by the vendor, the date timestamp of when the ID was validated cryptographically, and who performed the validation.
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, the date timestamp of the analysis, and who performed the analysis. The program should maintain documentation on the sample selection process. This could be from the use of a non-human selection process, cleared personnel, or an independent party. The program should maintain the list of devices used for samples. Document the process to secure the device and the results, as well as, documentation of all parties that touched the device with the reason for the interaction.
To mitigate risk of a cleared insider:	
Select sample parts	The program should document the process used to collect the samples, secure the device and the results, as well as, all parties that touched the device with the reason for the interaction.
Create cryptographically protected IDs	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.

TD 2 mitigations	Documentation requirements
Verify independent lab work	The program should require: • 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 the item above, to verify the independent lab work also 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 observer, the findings, and the 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.
Follow guidance for “TD 4: Adversary compromises system assembly, keying, or provisioning”	Provide all of the “TD4: Adversary compromises system assembly, keying, or provisioning” data requirements.

Checklist for TD 3: Adversary compromises application design cycle

TD 3 mitigations	Documentation requirements

Track critical data in a revision control system	The program should ensure the following data items are tracked in revision control:
	• 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:
	  • 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
	
	Each of the artifacts should be identified in the program's 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 to include requirements, architecture, design, code, tests, bugs and fixes. The audit data minimally should document who requested the change with date 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 timestamp.

TD 3 mitigations	Documentation requirements
Use revision control and version management tools	The program should document and enforce a configuration management (CM) plan that is compliant with 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 document how the CM plan is compliant with the requirements. The configuration management plan should include details on how configuration data will be maintained for control and audit purposes. It should include management of document/data, releases, backups and archives, refresh of backup media, retention of tools and software, test equipment, and the test environment. Audit logs should be reviewed with the results recorded.
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 separate builds produce the same binary and hash.

TD 3 mitigations	Documentation requirements
3.2 Mitigating the modification of test benches/plan to reduce coverage or hide Trojan code	
Execute a documented test plan	The program should document and maintain a test plan that includes a mechanism to verify all requirements. • 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, 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	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

TD 3 mitigations	Documentation requirements
	numbers, and the various test reviews that will take place, type of testing to be performed, and the methods used to accomplish the test. 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.
Maintain test environment via configuration management	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 CMMC audit results or NIST SP 800-171 self-assessments.
3.3 Mitigating the introduction of Trojans into the application design during development	
Maintain bi-directional link to requirements	The program should document bi-directional traceability for all device requirements, including derived requirements.
Enforce peer review	The program should document the results of each peer review to include: • Entry criteria and status • Roles and responsibilities with associated names • Attendees • Findings, including deviations or waivers, and

TD 3 mitigations	Documentation requirements
	associated rationale and approval • Exit criteria and status
Execute a documented test plan	The program should document and maintain a test plan that includes a mechanism to verify all requirements. • 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, 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	The program should document, review, maintain, enforce, and archive the test plan. The test plan should include which tools will be used with names and version numbers, the various test reviews that will take

TD 3 mitigations	Documentation requirements
	place, the types of testing to be performed, and the methods used to accomplish each test. 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, 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.
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, (minimally those identified in this guidance should be searched) search results, the name of the person performing the search, and the date timestamp when the search was performed.
If vulnerabilities are found in the software or tools, choose one of the following options:	
Option 1: Select a different tool	For the different tool, the program should document each publication that was searched, (minimally those identified in this guidance should be searched) search results, the name of the person performing the search, and date timestamp when the search was performed.

TD 3 mitigations	Documentation requirements
Option 2: Work with vendor	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.
To perform tool validation, choose one of the following options:	
Option 1: Reproducible build process	Document the reproducible build process and results validating that the separate builds produce the same binary and hash.
Option 2: Select a formal "proof" process	Document all code that was reviewed using LEC, any functional discrepancies, and how those discrepancies were resolved.
3.5 Mitigating intrusion into the internal network	
Assign Roles	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.
Control and monitor access	Entry/access to appropriate areas should be recorded, monitored, and logged for auditability.
Research vulnerabilities	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 date timestamp of when the search was performed.

TD 3 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.
Use trusted computing environments	The program should maintain documentation and audit data demonstrating one of the following computing platforms were utilized: • A computer and 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. • 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. • An infrastructure compliant with NIST SP 800-171 and NIST SP 800-172, preferably compliant with CMMC level three. If CMMC is a program requirement, the program should maintain CMMC audit data. Until CMMC is a program requirement, the program should maintain a self-assessment demonstrating compliance.
3.6 Mitigating risk from compromised hire or employee	
Enforce auditability	The program should maintain audit logs on all design data to include requirements, architecture, design, code, tests, bugs and fixes. The audit data minimally should document who requested the change with date 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 timestamp.
Adhere to an approved design process	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

TD 3 mitigations	Documentation requirements
	design review with roles and responsibilities along with associated names, attendees, and findings, including deviations or waivers and associated rationale and approvals. All design changes should be documented and approved, and testing should adhere to organizationally approved test standards.
Review critical activities	The program should obtain the results of independent reviews to include: • 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 • Time/date stamp of when the review was performed
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 their clearance levels. The program should maintain audit logs demonstrating what each team member accessed.

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: • 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 The program should document, maintain and enforce a storage plan which supports the storage of bulky material.
Provide keys and configuration data	The program should document assembly house receipt of data packages and the hash value of the packages.

TD 4 mitigations	Documentation requirements
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.
Provision private keys	The program should document: • 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	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. 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. The program should maintain documentation including the authentication methodology, its architecture and compliance with appropriate NIST standards. 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. The program should maintain documentation including the procedures used to authenticate the post assembly

TD 4 mitigations	Documentation requirements
	FPGA device, where the authentication was performed, by whom, when and the results of the verification.
To authenticate the FPGA device, choose one of the following options:	
Option 1: Verify the unique cryptographic ID	The program should document: • 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: • 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: • 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 only 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
Validate the cryptographic hash of the IP	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.
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 document the reviews and all results in accordance with the *Third-Party IP Review Process for Level of Assurance 2*
Option 2: Contact JFAC to determine if an IP review of the complete IP package has been successfully completed	The program should maintain documentation and provide it to JFAC with IP identification information, what program the IP is used in, and the role that IP serves within the system. The program should document proof of receipt from JFAC along with all interactions with JFAC. The program should obtain and review evidence of IP verification, including requirements sign-off. **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 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: • 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 algorithm and key length being used along with the version number of the latest guidance and the approved key length in accordance with the guidance.
Use security-evaluated authentication	The program should maintain documentation from JFAC with the security evaluation results.
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 authentication was actually applied to all reconfiguration data.
As part of authenticating application modifications, in cases where security settings are set within the configuration file:	

TD 6 mitigations	Documentation requirements
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.
Generate and store all authentication keys on a program-controlled, FIPS 140-2 compliant, 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 platforms were utilized: • A computer and network classified at the DSCA Secret level or above. The documentation should include a log of personnel with clearance information,

TD 7 mitigations	Documentation requirement
	all records in accordance with a maintaining a DSCA Secret network, as well as a documented and SSP. • 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. • An infrastructure compliant with NIST SP 800-171 and NIST SP 800-172, preferably compliant with CMMC level three. If CMMC is a program requirement, the program should maintain CMMC audit data. Until CMMC is a program requirement, the program should maintain a self-assessment demonstrating compliance.
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 their clearance levels. The program should maintain audit logs demonstrating what each team member accessed.
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: Independently validate the hash	Document the separate purchases of the tool, the hashes that were validated and when, and the two cleared individuals who validated them.
Option 2: Select a formal “proof” process	Document all code that was reviewed using LEC, any functional discrepancies, and how those discrepancies were resolved.

TD 7 mitigations	Documentation requirement
Option 3: Use a reproducible build process	Document the reproducible build process and results validating that the separate builds produce 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 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.
Verify and authenticate with independent teams	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.
Authenticate the FPGA devices	The program should document the physical inspection results for each slash sheet and unique identifier for the device inspected.
Verify PCB connections	Document the review of the SBCS schematics and PCB traces and any findings, along with who performed the review and the date timestamp of the review. In addition the program should maintain the tests that were performed for the evaluation along with

TD 9 mitigations	Documentation requirements
	the evaluation results, timestamp, and name of evaluator in the revision management system.
Verify the SBCS configuration process	The DoD supplier should provide the document demonstrating compliance
Review and evaluate SBCS vendor code	The program should document the review of all vendor code contained within the configuration file. The vendor should provide and the program should maintain documentation about all proprietary capabilities. Record the vendor documentation and the evaluation of the proprietary methodology.
Poll FOGA settings captured in non-volatile memory	The program should maintain documentation that includes the FPGA settings available in the given FPGA device, the methodology used to read them, where and when they were tested, by whom, and the results.
Document the steps	Document the steps taken to comply with these requirements. These steps and associated data artifacts should be auditable.

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

TD 10 mitigations	Documentation requirement
Isolate and store the application design	The program should document the storage plan, who has access to the design, when, why and by whom the accessed the design. In addition, the hash of the final configuration after the final design should be stored and verified prior to provisioning. The program should maintain the configuration management audit logs.
Perform logical equivalency checking	The program should document any hints, all optimizations, and rationale for any logic that did not

TD 10 mitigations	Documentation requirement
	match the equivalency checker with managerial approval signature.