National Security Agency
Cybersecurity Technical Report

DoD Microelectronics:

Field Programmable Gate Array

Level of Assurance 1 Best Practices

May 2024

U/OO/156781-24
PP-24-1960
Version 1.1

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

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

Notices and history

Document change history

Date	Version	Description
December 2022	1.0	Initial Publication
May 2024	1.1	Minor updates throughout

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: JFAC_HWA@radium.ncsc.mil
General Cybersecurity Report 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 and Department of Defense information systems, 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 1 (LoA1). LoA1 captures the threats most likely to be exercised against a DoD system based upon their low cost and high value of return. At this level, these threats have the following characteristics:

- **Access** – Exploit a single available point of access,
- **Technology** – Use existing public technology,
- **Investment** – Require minimal investment of resources,
- **Value of effect** – Disable or subvert system capabilities, and
- **Targetability** – Are inherently targetable and controllable.

Organized by threat, this report provides multiple technical mitigations to choose from to mitigate each threat and allow the program the best fit for their program needs. The following table identifies the 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
TD 9	Adversary compromises single-board computing system (SBCS)

* TD 8 is not a threat at LoA1; however, JFAC requests that DoD programs even at LoA1 provide information about FPGA device use as described in JFAC Survey Request. This information relates to TD 8 mitigations.

Each subsection in this report contains mitigations described in detail to enable clear implementation guidelines for DoD systems. Secondary documents are referenced in cases where the suggested mitigation is highly detailed, specific to individual FPGA platforms, or subject to frequent change. Appendix E: Checklists and data/documentation requirements contains a quick reference list of threats and associated data requirements.

Once the DoD program has mitigated these threats, they have achieved an assurance level of LoA1.

Contents

Tables

1. Introduction

This document provides JFAC's recommended hardware assurance strategies for Field Programmable Gate Array (FPGA) devices used by DoD programs. The guidance outlined by this document provides hardware assurance to systems requiring Level of Assurance 1 (LoA1). 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.

The mitigations included within this document are the responsibility of the program, to include all subcontractors. These mitigations are not written for third-party providers to use for the purpose of becoming "LoA" compliant. Since the third-party provider represents potential malicious access points and the program has no positive control over them, work or products coming from them should be verified against threats. Third-party statements of mitigation resolution are not satisfactory; the program has the responsibility to verify the mitigation. Third-party entities could include third-party IP providers (3PIP), software vendors, and manufacturers.

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 contact JFAC_HWA@radium.ncsc.mil.

2. Terms

To aid the reader, a glossary of terms is at the end of this document in Appendix A. However, it will be helpful to highlight several terms before proceeding. Below are several important terms to understand:

- Application or application design – a DoD program's design that is programmed into an FPGA device. This can refer to the design in any of its various formats.

- Configuration file or data – also known as a bitstream, this is the data used to configure the FPGA including the application design and all other programming information.
- FPGA device or device – a specific individual FPGA packaged device.
- Hard IP – a hardware design, also called intellectual property or IP, which is represented by its physical layout format. In FPGAs, this is IP that is embedded by the FPGA vendor into the FPGA physical design.
- Soft IP – IP that is represented in a logical human readable format such as a hardware description language.
- Platform design – the design of an FPGA family of devices, not just an individual device.

3. Overview of Level of Assurance 1 threats and mitigations

LoA1 requires mitigations against FPGA assurance threats that have the following characteristics:

- **Access – A single point of access** to some portion of the FPGA supply chain. This point of access does not contain cleared personnel. This is defined by the following:
  - An Internet connected network, regardless of other security measures
  - Any single uncleared U.S. person
  - A group of associated foreign nationals within a U.S. organization, such as a corporate office operated in a foreign country
  - A foreign owned company servicing part of the supply chain
  - Any number of foreign nationals from a high threat country or its allies with access to some part of the FPGA supply chain

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 access or complex points of access. For example, the access mitigation could result in an attack needing access from differing areas of the supply chain or by multiple personnel in an area of the supply chain.

- **Technology – Existing public technology** means that an attack can be conducted using tools that are already available in the public or commercial domain, or are straightforward advances of public technology. Examples would include:
    - Development tools provided by FPGA vendors
    - Internal debugging features that are capable of changing device configuration
    - Lab equipment used as intended
    - Publicly available open source projects
    - Published academic research
    - Results of U.S. Government (USG) Research & Development (R&D) investment at the unclassified level, even when protected by International Traffic in Arms Regulations

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 can be found in the public domain. The most common method to achieve this mitigation is to apply cryptographic authentication based on USG standards.

- **Investment – Minimal investment of resources** means that an attack requires a team with existing FPGA knowledge and skills and individuals with domain knowledge in the technology area of the system being assured. Minimal resources are defined as any effort consisting of less than six person-years of FPGA/domain expertise focused solely on attacking the target of interest.

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 a complex interdisciplinary team comprised of a mix of specialties that are outside of the application design domain and FPGA technology to carry out an attack. Additionally, the mitigation would lengthen the time necessary to develop the attack to more than six person-years. Obfuscation of design data is not considered effective in this context. Common use of obfuscation has historically led to technological development in those additional disciplines that quickly nullified the obfuscation.

- **Value of Effect** – Attacks that **disable or subvert capabilities** enable an adversary to remove a capability from service or cause it to perform specific deleterious actions. When combined with high targetability, these represent the worst-case scenario for a failure of hardware assurance: enabling an adversary to take over or disable capabilities on command.

For a mitigation based upon value of effect to the adversary, it must constrain the severity of the outcome on the target to one of lesser effect.

- **Targetability - Inherently targetable and controllable** threat operations are executed in a way that provides straightforward means to understand and predict the effect of an attack, and also provide a mechanism to control or time the attack. For example, an adversary with the ability to introduce new code into a system design can implement a broad number of malicious functions. A denial of service attack falls in this category, if and only if, it is possible for the adversary to control when it takes effect after the device is fielded. However, a simple reduction in reliability not tied to any trigger, which therefore cannot be controlled or timed in a planned way, does not fall in this category.

For a mitigation based on targetability to be effective, it must prevent the attack from being controlled effectively. That is, it must prevent the FPGA device from being used to attack a specific target at a specific time with a controllable trigger. This can be accomplished by performing mitigations that specifically target opportunities for communication.

For a program to achieve Level of Assurance 1, it must provide mitigations against threats that fall within these characteristics. LoA1 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. However, all programs with radiation-hardened requirements are an exception and in almost all cases should mitigated at a Level of Assurance 2 or Level of Assurance 3.

The following table lists the nine FPGA threats that are addressed by LoA1. Each threat is explained and accompanied by examples in more detail within the JFAC *FPGA Best Practices – Threat Catalog.*

Table 1: LoA1 threats

#	Threat description (TD)
TD 1	Adversary utilizes a known FPGA platform vulnerability
TD 2	Adversary inserts malicious counterfeit
TD 3	Adversary compromises application design cycle
TD 4	Adversary compromises system assembly, keying, or provisioning
TD 5	Adversary compromises third-party soft intellectual property (IP)
TD 6	Adversary swaps configuration file on target
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 8 does not become a relevant threat until LoA2; however, JFAC requires LoA1 programs to answer an information query required by TD8 mitigations.

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 and 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).

3.1 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:

- National Institute of Standards and Technology (NIST) Federal Information Processing Standards Publication (FIPS) 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
- Committee on National Security Systems (CNSS) Policies
- 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.

3.2 Exclusions

This FPGA Level of Assurance 1 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 to 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).

3.3 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 should define its own protection methods as boards become integrated into subcomponents, components, and then final systems.

For LoA1 compliance, each program should perform each mitigation listed in the "TD # Mitigations" sections. The "Mitigation Descriptions" sections provide details for each mitigation. In some cases, the full description contains additional options that are required to be LoA1 compliant. An asterisk "*" next to any mitigation indicates additional options should be implemented.

When mitigations for all the threats listed under LoA1 are completed, that device can be said to have achieved LoA1. However, compliance with LoA1 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 the 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. It is recommended that changes in any of these areas should prompt a review of the most up-to-date assurance mitigations against the triggering event. If threats or vulnerabilities threaten the system, it is recommended that new mitigations should be implemented to remain compliant to LoA1. Absent any changes in these areas, the devices should be considered to have achieved LoA1.

3.4 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.
- When third party sources are required, 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.

4. Threat Descriptions (TD)

TD 1: Adversary utilizes a known FPGA platform vulnerability

In this attack, a foreign adversary utilizes a vulnerability in an FPGA platform or vendor development software package to initiate an attack. At LoA1, a vulnerability is an unclassified published weakness or vulnerability in the design of a specific FPGA platform or software program that would allow the attacker the ability to use it for malicious purposes.

Vulnerabilities could allow for leakage of sensitive information or keys; compromise of security or tamper detection functions; or unauthorized reconfiguration of the product. Unclassified and public vulnerabilities are published in various places, including vendor advisories, errata bulletins, and databases, such as those listed in the "*Research Vulnerabilities*" mitigation below. This threat can be introduced by a program not performing vulnerability research, an insider not disclosing the fact of the vulnerability

such that it may be used for nefarious purposes or adding/modifying design features for use with or triggering the vulnerability.

TD 1 mitigations

- Use caution when selecting tools or platforms. When possible do not select tools or platforms that are end-of-life or beta/initial releases. Also, ensure known vulnerabilities in tools/platforms have been adequately addressed in newer releases.
- *Research vulnerabilities affecting tools/platforms.
- Use a revision control/version management system that includes document/data control, document/data release, backups and archives, refresh of backup media, software, test equipment, and test environment.
- Enforce auditability of the application 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.
- Perform a vulnerability data review.
- Perform routine employment monitoring to unmask a compromised insider who might hide the existence of a vulnerability.
- *Take action to prevent a compromised insider from hiding a vulnerability.

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.

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

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
- Common Weakness Enumeration (CWE) – https://cwe.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, errata bulletins, 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 that meets the program requirements.

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

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 Government Industry Data Exchange Program (GIDEP) and to contact the vendor so they may correct it.

Use a revision control/version management system

To prevent vulnerable software from being loaded into the environment, it is important that robust configuration management and revision management systems are in place. Before any version is installed, research should be performed for any known

vulnerabilities, how they have been addressed, and their impacts and vulnerabilities. Only authorized changes are allowable. 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 only an authorized set of administrators; 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; introduce 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 should log every change made by individuals. This includes changes such as creating and deleting files and editing content. The history should 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

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.

Perform a vulnerability data review

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 people or a cleared individual. The original designer should not be the responsible party for performing the review. The reviewers should assess all vulnerability activities, including identification of vulnerabilities and the appropriateness of the mitigations.

Perform routine employment monitoring

Perform routine employment monitoring in which employee work patterns are observed. Patterns to look for include hostility toward other employees, late or excessive missing work, unexplained work outside normal work hours, and declining performance.

Prevent a compromised insider

To limit the potential effects of a compromised insider, select one of the following options:

Option 1: Ensure all critical activities are identified and documented. Independent third-party reviewers should assess all vulnerability activities, including identification of vulnerabilities and whether the appropriate mitigations are in place. **Note:** For LoA1, independent is defined as “not the originator”. The reviewer can be on the same team if necessary.

Option 2: Perform designated work using personnel with at least a Secret-level clearance.

TD 2: Adversary inserts malicious counterfeit

At LoA1, this threat assumes that the adversary already has access to an ***existing*** fabrication process for manufacturing counterfeits. Using this access, the adversary inserts additional logic in the FPGA die for their malicious purposes. In general, modifications to an FPGA design during the fabrication phase are considered to require a high effort. However, in cases where an adversary is known to have already counterfeited a device, that effort has already been invested and is thus reduced.

Explicitly, this threat is only of a counterfeit that contains functional differences that are security relevant.

While commercial (non-malicious) counterfeits, such as re-marked parts, may represent a reliability risk, they are not included under this level 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. The exception to this statement are parts that have radiation-hardening requirements. In all cases, these devices should comply with Level of Assurance 2 or Level of Assurance 3 in addition to the appropriate level of reliability testing to secure the operation of their design.

The insertion of counterfeit parts can happen during any part of a device's lifecycle. This includes prior to purchase, during transit, while in storage by the program, during assembly, and at distribution prior to fielding.

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

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

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

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

TD 2 mitigations

- For DoD system owners, purchase from DoD authorized vendors and distributors. The DoD program acquisition group can provide this information.
- Follow storage and shipping guidance for classified Secret or Trust Category I materials when storing and transferring FPGA devices between locations.
- Validate the authenticity of the FPGA device.

TD 2 mitigation descriptions

Purchase from DoD authorized vendors and distributors

For DoD system owners, utilize DoD authorized vendors and distributors for all purchases. Authorized vendors can be identified through the acquisition organization.

Follow storage and shipping guidance

The program should document, maintain, and enforce both device storage and shipping procedures. Minimally, the procedures should enforce the verification and acceptance testing of all devices upon receipt. In addition to device verification, storage and shipping processes should focus on preventing and detecting adversary access to the parts. Access controls should be designed to thwart those seeking to alter or replace the devices while in storage or shipping.

Storage protections should include the following characteristics:

- Production devices should be stored and maintained in a restricted area separate from non-production devices (i.e., ones for design, test, etc.).
- The restricted area should enforce access controls that limit access to only those personnel who require access to support direct job responsibilities. This should exclude all members of the design team.
- All accesses to the restricted area should be recorded and audited.
- The FPGA devices should be audited on a regular cadence to ensure they have not been removed or modified.

- 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 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. Accesses to the restricted area should be audited to include data containing who entered/exited the area, with a timestamp, and reason for entry.
- When possible, the FPGA devices should be stored within a tamper detection seal or tape to detect unauthorized access.

Shipping protections should include the following characteristics:

- Following acceptance testing, non-configured and non-keyed devices should be using a commercial carrier that has been approved by the appropriate Cognizant Security Agency (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.
- Utilize protections against pilferage, theft, and compromise, including:
    - Hardened containers
    - Tamper detection seals and tapes. The seals should be numbered, and the numbers indicated on all copies of the bill of lading (BL). When seals are used, the BL should 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.
- Closely track shipment times and depot stops. Packages that are late should be considered suspect.

Validate the authenticity of the FPGA device

To validate the authenticity of the FPGA devices, choose one of the following options (descriptions are below):

Option 1: Use an FPGA device that includes a cryptographically secure ID that can be vendor verified.

Option 2: Perform physical analysis on a random sampling of devices to detect counterfeit parts.

Cryptographically secure identifier

For LoA1, the program should utilize an FPGA device that incorporates a cryptographically protected ID that can be verified against information sent by the vendor (not the authorized distributor). The use of this type of device ID mitigates the sub-threat of counterfeit parts made in an existing, non-authorized fabrication facility. While the specifics of each FPGA vendor and platform vary, many newer FPGA platforms contain this type of anti-counterfeiting feature. When these features are sufficiently secure, such mechanisms provide an extremely cost-effective method to detect counterfeits both at acquisition and throughout the FPGA device's lifecycle in a system. The two biggest advantages of such techniques are the ability to validate a device remotely and the ability to non-destructively re-validate a device at any time.

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 through local access, such as a board header, or remotely. Depending on the exact mitigations selected, this potentially saves two distinct destructive steps: one at acquisition of the devices and one after assembly of the printed circuit board (PCB).

This kind of validation is where details matter, each FPGA vendor offers a unique approach, and each FPGA platform offers a unique variation. In no case is a fully readable ID acceptable. Instead, these schemes all detail cases where the device possesses a specific private cryptographic key. The device ID in this scheme can be cloned only if an adversary is able to get access to that private key. Regardless of the specific platform used, the public keys/identifiers of the devices being authenticated should be delivered and maintained in a secure way. For delivery, the vendor should

provide this information to the program using a NIST approved authentication algorithm to transmit the data. Examples would be an ECC-signed email with a verified certificate, or an https-based file distribution system using a verified certificate. Once received, the integrity of that list should be maintained by storing with protections appropriate to Critical Protected Information (CPI). This should include restricted role-based access on a network that is compliant with the contract defined Cybersecurity Maturity Model Certification (CMMC) level.

Specific criteria required for an appropriate device ID to support anti-counterfeiting are below:

- Cryptographically protected IDs utilize a private asymmetric key for which no read function exists. This should use a National Institute of Standards and Technology (NIST) approved asymmetric authentication algorithm.
- The provenance of the key should be understood in detail.
- The device should be able to authenticate a nonce using this key. Each device's ID should be authenticated by the vendor-provided public key through decryption of the nonce.

Physical analysis

Perform physical analysis on a sampling of random devices to detect counterfeit parts. This analysis applies specific, industry standard counterfeit inspection techniques, including package analysis, x-ray of the part, and examination of the die with comparisons against FPGA vendor-provided golden samples. This physical analysis is intended to catch parts that have been remarked or contain counterfeit die. The details of what steps to conduct in the analysis and recommendations on how to execute them are contained in the commercial standard document, *SAE Test Methods Standard; General Requirements, Suspect/Counterfeit, Electrical, Electronic, and Electromechanical Parts, AS6171*. These inspections should be carried out by cleared persons at a Secret level or higher or a lab independent of the program or its performers.

Physical analysis is a sequence of device analysis steps, from least destructive to most destructive, designed to ensure that the part in question is authentic. If a device fails a given step, it is not authentic, and there is no need to complete further steps. If all steps are completed and the device passes, it is likely authentic, with likelihood commensurate with the amount of effort it would take to get a counterfeit device to pass

these tests, subject to LoA1 criteria. Each AS6171 test is detailed in a separate document called a “slash sheet”. Listed below are the slash sheets that comprise the 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
AS6171/11	Techniques for Suspect/Counterfeit EEE Parts Detection by Design Recovery Test Methods

For the purposes of LoA1, the program should follow the lot sampling guidelines found in the AS6171 document and exercise the tests defined by slash sheets 1-10. The tests defined in slash sheet 11 are not necessary for LoA1 but will become more important at higher assurance levels. The waiving of slash sheet 11 testing for LoA1 does not supersede any other DoD standard that requires it.

Select sample parts

The selection of parts to be physically sampled should be handled in such a way that a compromised insider could not knowingly select only genuine parts to be sampled. Possible sampling options include the following:

Option 1: An independent party handles part selection before shipping. They should physically verify that the parts selected make it all the way to the physical inspection processes and verify upon receipt that the right parts were received.

Option 2: Use a non-human random selection automated process for sampling.

Option 3: Physical verification and sampling work should be conducted by personnel holding clearances of at least the Secret level and carried out in facilities cleared to at least the Secret level.

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, design constraints, or 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.

As TD 3 is comprised of numerous scenarios, this threat is broken into eight sub-threats or scenarios and addressed separately. Collectively, these scenarios describe the entire threat at TD 3 and each of the mitigations for each scenario should be implemented. The specific scenarios are as follows:

- Introduction of a compromised design into the application,

- Modification of test benches or plans to reduce coverage or hide Trojan code,
- Introduction of a Trojan into the application design during development,
- Introduction of compromised tooling or software into the environment,
- Network intrusion,
- Compromised employee,
- Modification of revision control that hides code or test bench modification (associated mitigations are captured in the “in all cases” section below), and
- Introduction of modified configuration data after generation (associated mitigations are captured in the “in all cases” section below).

TD 3 mitigations

The best practices presented here do not describe a standalone engineering FPGA design flow, but rather the assurance practices that 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.

TD 3 mitigations focus strongly on the insider threat. At LoA1, access is defined as singular with uncleared people. As such, this threat can be mitigated by following the guidance in the use cleared personnel in a cleared environment section. In the event that is not possible, the mitigations associated with each scenario should be incorporated.

Use cleared personnel in a cleared environment

Use cleared personnel with at least a Secret-level clearance in an environment suitable for a network cleared at the Secret level.

When the program selects not to use cleared personnel and a cleared environment, they should implement all TD 3 sub-threat mitigations, as there are multiple threats to be mitigated. The following mitigations 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 a revision control/version management system that meets 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 IP (3PIP)
- Utilized libraries
- Development files, code, software used for development, synthesis scripts, and tools
- Test benches, test plans, test procedures, and test reports
- Tool configuration settings
- Design documents

Enforce auditability

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

Use a revision control/version management system

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 only an authorized set of administrators; 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; introduce 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 should log every change made by individuals. This includes creation and deletion of files and content edits. The history should include the person who made the change, what was changed, the date, and written notes on the purpose of each change.
- Ensure the system stores a reliable copy of assets that are currently in production.
- Ensure the system stores reliable copies of previous production versions of assets, allowing for the complete retrieval of those versions.
- Enforce password best practices (password rotation, length, etc.). In lieu of a password, two-factor authentication can be used.

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

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

To accomplish this task, the adversary would have to compromise the revision management system. That compromise could allow the adversary to switch the verified configuration files, settings, hash or other pertinent information. To protect against this, the program should store and isolate the verified FPGA application configuration files, settings, and associated hash. Before the design is loaded for final deployment, the program should verify the hash to know the verified version is the same as what they want to deploy. For extra assurance, the program has all the necessary data to

reproduce the build which can be used to verify the stored versions against the reproduced version.

Mitigations

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

Descriptions

Isolate and store the application design

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

Perform a reproducible build

A reproducible build process is a methodology to verify the integrity of the FPGA synthesis and build software. 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 LoA1 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 electronic design automation (EDA) tools, and, in some cases, the same operating system version. This process will highlight the possession of modified software when 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, analysis to be performed, type of testing to be performed, and the methods used to accomplish the test.
- Validate and verify test processes which include design/test team separation, peer reviews, and use of automated tools where applicable.
- Ensure the test environment is maintained via configuration management as a critical system.
- Use a revision control/version management 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 captured in the FPGA specification.
- Explicitly list code coverage metrics, the type of testing that will be performed, and acceptable testing guidelines. Code coverage should state how much code is checked by the test bench, providing information about dead code in the design and holes in the test suites. Document the decision to use/not use other types of testing, such as directed test, constrained random stimulus, and assertion.
- Ensure code coverage includes statement coverage, branch coverage, Finite State Machine (FSM), condition, expression, and toggle coverage. Document any code that will not be covered and why. Ensure untested code is documented and reviewed through the review process. Use functional tests to verify the FPGA does what it is supposed to do. Any deviations should be documented and approved.
- Specify the verification environment which describes the tools, the software, and the equipment needed to perform the reviews, analysis, and tests. Each of these items should be maintained under revision control.

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

Validate and verify 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.

Ensure the test environment is maintained via configuration management

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

Use a revision control/version management system

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 only an authorized set of administrators; 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; introduce 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 should log every change made by individuals. This includes creation and deletion of files and content edits. The history should include the person who made the change, what was changed, the date, and written notes on the purpose of each change.

- Ensure the system stores a reliable copy of assets that are currently in production.
- Ensure the system stores reliable copies of previous production versions of assets, allowing for the complete retrieval of those versions.
- Enforce password best practices (password rotation, length, etc.). In lieu of a password, two-factor authentication can be used.

TD 3.3: 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

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

Descriptions

Ensure all design artifacts have a direct bi-directional link to approved requirements

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

Enforce peer review

Establish and enforce peer review practices with the following:

- The author and the reviewer should 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 agreed upon architecture.
- All black box portions of the design should be identified, justified, and approved.
- All scripts that produce design artifacts (HDL, netlist, etc.) should 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 program should consider assurance when creating and maintaining the test plan. The test plan and processes should at least:

- Provide a mechanism to verify all requirements captured in the FPGA specification.
- Explicitly list code coverage metrics, the type of testing that will be performed, and acceptable testing guidelines. Code coverage should state how much code is checked by the test bench, providing information about dead code in the

design and holes in the test suites. Document the decision to use/not use other types of testing, such as directed test, constrained random stimulus, and assertion.

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

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, JTAG settings, and key storage.

There are technical challenges associated with performing Logic Equivalence Check (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

- Accept only digitally signed software deliveries.
- Validate cryptographic hashes against vendor provided hashes.
- *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.
- Use a revision control/version management system 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.
- Utilize a reproducible build process to generate any deployable configuration files. The program should independently validate this reproducibility for each deployable version on a distinct computer system, with an independently acquired version of the same EDA tools.
- Select a formal “proof” process that can validate the equivalency of the hardware descriptor language (HDL) and final configuration file. For more information on “proof” tools, contact JFAC.

Descriptions

Accept only digitally signed software deliveries

Verify that the received software has a valid digital signature on all executables and scripts. The signature is used to confirm the software’s author and guarantee that the code has not been altered or corrupted since it was signed.

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. Platform and tool vulnerabilities can be found in databases such as:

- Common Vulnerabilities and Exposures (CVE) – https://cve.mitre.org
- Common Weakness Enumeration (CWE) – https://cwe.mitre.org
- NIST National Vulnerabilities Database (NVD) – https://nvd.nist.gov
- Government Industry Data Exchange Program (GIDEP) – https://www.gidep.org/products/products.htm
- DISA Security Technical Implementation Guides (STIGs) – https://public.cyber.mil/stigs/
- Searches for vendor advisories, errata bulletins, 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 FPGA platform device or software that does not have published vulnerabilities and that meets the program requirements.

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

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 Government Industry Data Exchange Program (GIDEP) and to contact the vendor so they may correct it.

Use a revision control/version management system

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 only an authorized set of administrators; 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; introduce role-based access control and privileges; and remove unused accounts.
- Remove any native system 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 should log every change made by individuals. This includes changes such as creating and deleting files and editing content. The history should 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.

Utilize a reproducible build process

A reproducible build process is a methodology to verify the integrity of the FPGA synthesis and build software. 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 LoA1 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 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 placed netlist is compared to the originating application HDL to demonstrate they are logically equivalent with no extraneous logic in the final format. In cases where commercial tools exist to extend the comparison to the bitstream itself, that comparison is recommended. 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 Logic Equivalence Check (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 in which they can act on their nefarious intentions, such as modifying tools, swap files, etc.

Mitigations

- Use a revision control/version management system 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.

- Assign privileges and accesses based on roles.
- Based on job requirements control and monitor access, including 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 vulnerability. Alternatively, perform a risk assessment and coordinate findings with JFAC.
- For DoD system owners, purchase from DoD authorized vendors and distributors. The DoD program acquisition group can provide this information.
- Use protected computing environments to protect from remote attack.

Descriptions

Use a revision control/version management system

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 only an authorized set of administrators; 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; introduce 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 should log every change made by individuals. This includes changes such as creating and deleting files and editing content. The history should 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.

Assign privileges and accesses based on 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 should 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 physical 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
- Common Weakness Enumeration (CWE) – https://cwe.mitre.org
- NIST National Vulnerabilities Database (NVD) – https://nvd.nist.gov
- Government Industry Data Exchange Program (GIDEP) – https://www.gidep.org/products/products.htm
- DISA Security Technical Implementation Guides (STIGs) – https://public.cyber.mil/stigs/
- Searches for vendor advisories, errata bulletins, publications, and academic papers detailing vulnerabilities in the device in question.

Purchase from DoD authorized vendors and distributors

For DoD system owners, utilize DoD authorized vendors and distributors for all purchases. Authorized vendors can be identified through the acquisition organization.

Use protected computing environments

Programs should select one of the protected computing platform 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.
- Track critical data in revision control.
- Adopt a structured application design process that is documented, adhered to, and organizationally approved.
- Identify, document, and review critical activities. These items should be reviewed by a cleared individual that is different than the original designer.
- Enforce reviewer criteria.

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 timestamp, why the change was requested, and who made the change with date timestamp.

Track critical data in revision control

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

- Third-party IP (3PIP)
- Utilized libraries
- Development files, code, software used for development, synthesis scripts, and tools

- Test benches, test plans, test procedures, and test reports
- Tool configuration settings
- Design documents

Adopt a structured application 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 peer reviewed. 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.

Enforce reviewer criteria

Enforce a formal review process using one of the following options:

Option 1: All critical activities are identified and documented. Independent third-party reviewers should assess all critical activities.

Option 2: Ensure reviews are performed by independent teams comprised of individuals who hold a Secret-level clearance.

Option 3: Perform all of the reviews with cleared personnel in an environment certified to handle classified information at the Secret level or higher by DSCA. This would also include design reviews.

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

In this threat, an adversary has carried out an attack on the system during 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 the PCBs.
- Shipping the PCBs to the next manufacturing stage.

As such, there are multiple scenarios in which mitigations need to be applied. These scenarios are as follows:

- **Scenario #1**: The program performs product assembly and keying/provisioning in a facility/process certified at a classified Secret level or higher.
- **Scenario #2**: The program performs one or more of the seven steps above outside of a classified facility.

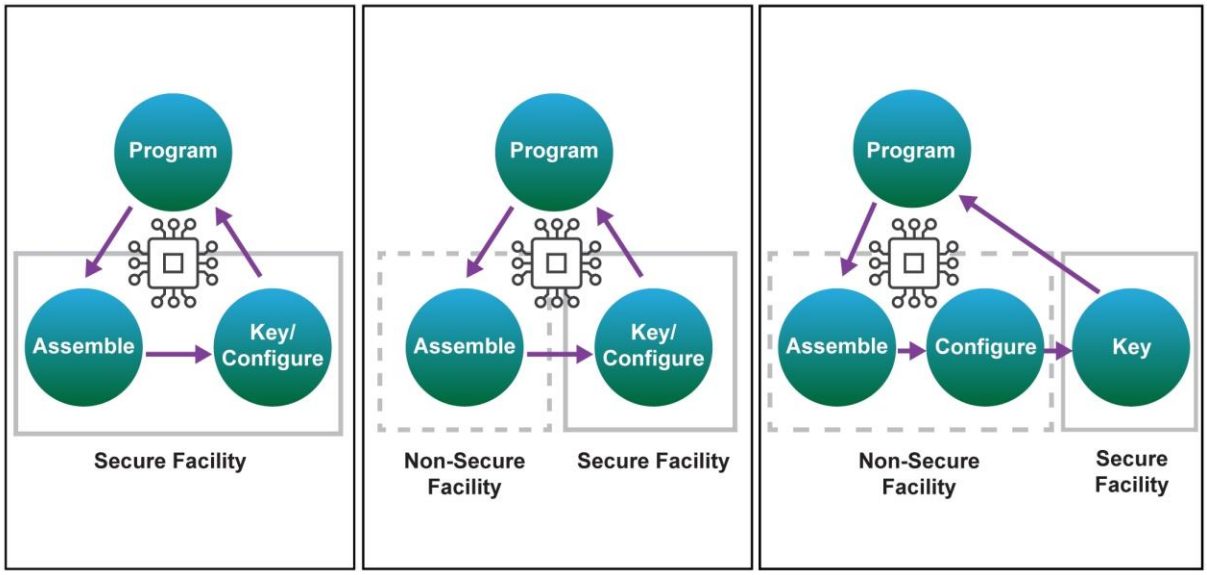

Figure 1: Secure and non-secure facility scenarios

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:

- For DoD system owners, purchase from DoD and vendor authorized distributors. The DoD program acquisition group can provide this information.
- Follow storage and shipping guidance for classified Secret or Trust Category I materials when storing and transferring FPGA devices between locations.
- Provide keys and configuration data to the provisioning house in digitally signed packages and with hashes. The assembly house should utilize these to verify the integrity of the contents.
- 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 FPGA from attack during assembly and provisioning.
- *Authenticate the FPGA device after being out of the control of the program.

TD 4 mitigation descriptions

Purchase from DoD and vendor authorized distributors

For DoD system owners, utilize distributors that are authorized by both DoD and the vendor 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 procedures should enforce the verification and acceptance testing of all devices upon receipt. In addition to device verification, storage and shipping processes should focus on preventing and detecting adversary access to the parts. Access controls should be designed to thwart those seeking to alter or replace the devices while in storage or shipping.

Storage protections should include the following characteristics:

- Production devices should be stored and maintained in a restricted area separate from non-production devices (i.e., ones for design, test, etc.).
- The restricted area should enforce access controls that limit access to only those personnel who require access to support direct job responsibilities. This should exclude all members of the design team.
- All accesses to the restricted area should be recorded and audited.
- The FPGA devices should be audited on a regular cadence to ensure they have not been removed or modified.
- 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 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. Accesses to the restricted area should be audited to include data containing who entered/exited the area, with a timestamp, and reason for entry.

- When possible, the FPGA devices should be stored within a tamper detection seal or tape to detect unauthorized access.

Shipping protections should include the following characteristics:

- Following acceptance testing, non-configured and non-keyed devices should be using a commercial carrier that has been approved by the appropriate Cognizant Security Agency (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.
- Utilize protections against pilferage, theft, and compromise, including:
    - Hardened containers
    - Tamper detection seals and tapes. The seals should be numbered, and the numbers indicated on all copies of the bill of lading (BL). When seals are used, the BL should 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.
- Closely track shipment times and depot stops. Packages that are late should be considered suspect.

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 AES algorithm with a key of at least 256-bit length. The assembly house should utilize these to verify the integrity of the contents.

Clear memory devices

Prior to provisioning, zeroize 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 FPGA from attack during assembly and provisioning

To protect the FPGA from attack during assembly and provisioning, the program can select from one of two options:

Option 1: Assemble and provision the system in a DSCA Classified Secret or Trust Category I certified facility. This can be a single secure facility or multiple secure facilities. The assembly work, as well as the keying and provisioning functions should be executed using equipment, processes, and personnel approved for Secret or Trust Category level efforts. This in and of itself is a mitigation against attacks focused on these portions of the manufacturing flow.

Option 2: Assemble and provision in an external unclassified facility. Programs that have chosen to perform some piece or all of the assembly, keying, and configuration file provisioning in an external unclassified facility should perform the following mitigations:

Those performing this validation should:

- **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. This verification should be performed by a team independent of the assembly and provisioning process. The verification should cover the entire contents of the memory device and not just the addresses containing the configuration data. It is recommended to program the entire memory space to disallow unused memory for nefarious purposes.
- **Verify** that the proper post-assembly keys have been loaded into the FPGA key storage elements. This verification should be performed by a team independent of the assembly and provisioning process. Some FPGA devices allow a hash of the keys to be read out for confirmation. Additionally, the program can create test

bitfiles to verify that the devices can properly utilize the keys and can reject actions utilizing 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 should be authenticated. The following guidance is intended to detect if the authentic verified FPGA devices were maliciously replaced by counterfeit or modified parts. 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 should contain a unique private asymmetric key for which no read function exists, and validation should involve the device signing a nonce. A National Institute of Standards and Technology (NIST)-approved asymmetric authentication algorithm should be used for this. The program should authenticate the FPGA devices utilizing this ID when they have been out of the positive control of the program.

Option 2: Verify the device on the PCB is an authentic and authorized device by performing physical counterfeit inspection with destructive sampling as described under Physical analysis for TD 2. 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 physical unclonable function (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 probable impacts include performance, power, or reliability. The mitigations to these attacks focus on verifying integrity of the delivery of the IP and reviews of its HDL code.

There are cases in which previously developed and/or evaluated IP can be re-used. See Appendix B IP reuse guidance for additional information.

TD 5 mitigations

- For DoD system owners, purchase from DoD authorized vendors and distributors. The DoD program acquisition group can provide this information.
- Only accept IP that is unobfuscated and distributed as functional code.
- Validate the cryptographic hash of the IP against the hash signed by the vendor.
- Check IP into revision control repository immediately upon receipt with the hashes used to authenticate the contents. Protection of the hash will allow for re-verification of the IP at a later date.
- Examine IP for malicious functions.

TD 5 mitigation descriptions

Purchase from DoD authorized vendors and distributors

For DoD system owners, utilize 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 unobfuscated and distributed as source code. The IP needs to be human readable and able to be reviewed.

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.

Check IP into revision control

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 unauthorized modifications.

Examine IP for malicious functions

To examine the IP for malicious functions, follow the guidance provided in *Third-Party IP Review Process for Level of Assurance 1* at https://www.nsa.gov/Press-Room/DoD-Microelectronics-Guidance/.

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 assurance of the operation of the hardware. 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 should involve authenticating the configuration file for both integrity and authorization. JFAC encourages programs to use device families that support configuration data authentication. For legacy devices that do not have

authentication functions, these guidelines will lean on external authentication functions or the use of symmetrically encrypted data files.

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

As of this publication (December 2022), all the major U.S. FPGA vendors provide built-in functionality to authenticate configuration files either at load into an internal memory or at configuration for at least one device family. The specifics of this authentication vary extremely. The exact details of key management and storage vary from device to device too. Some offer facilities to store many authentication keys, some use fuses, 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. As a rule, NSS programs must use the latest CNSS guidance. Non-NSS programs should use the latest NIST-approved asymmetric cryptographic algorithms at LoA1.

To achieve LoA1, the source of all boot/configuration images should be authenticated and the integrity of the data should be verified. That is, the device should validate that the file comes from an authorized provider and that the data has not been modified prior to loading. For LoA1, the recommended method for authenticating the data source is to use an asymmetric algorithm recommended by NIST. All of the FPGA commercial vendors have some device families with authentication schemes built in. Asymmetric algorithms are preferred because they do not require the protection of a secret key in the device, as implemented by the various vendors. For data integrity, a hashing algorithm such as secure hashing algorithm (SHA) is recommended.

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.

Many of the existing FPGA devices provide these functions for the user. However, less modern device families rely on symmetric encryption and the protection of a secret key by the device to provide source and data authentication. In general, LoA1 can be met with the following configuration authentication mechanisms, *in order of recommendation*:

- RSA/ECDSA with SHA-2 or SHA-3
- AES-GCM with SHA-2 or SHA-3
- Limited implementations of AES with HMAC with SHA-2 or SHA-3

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, as described in the latest approved version of FIPS 186, *Digital Signature Standard*, or FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC)*.
- Disable operation or use of test access pins in fielded products.
- When the program selects mechanisms that allow application modifications, ensure authentication is enabled for application modifications following the required NIST standards.
- Use a FIPS 140-2 compliant, Level 2 Hardware Security Module (HSM) that is program-controlled to generate and store all authentication keys.

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

NSS programs must use the latest approved CNSS algorithm and key length. Non-NSS programs should 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).

Disable operation or use of test access pins

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

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. In the case of JTAG, JFAC recommends disabling JTAG in full or limiting it to boundary scan for board testing. No access to the internals of the device should be left active. The program should verify that no unauthorized JTAG instructions remain active. JFAC recommends disabling this in non-volatile fuses when available.

Ensure authentication is enabled for application 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 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 should 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.

Authenticating reconfiguration data in the application itself

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

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

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

In some platforms, security settings can be programmed into both non-volatile storage in the device itself and as a setting in the configuration file loaded into the device. Settings should always be programmed in the non-volatile storage of the device. In those cases where use of security settings within the configuration file is acceptable, it should 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. Under these best practices, an application designer using this update technique should either validate that the technique in use applies the same authentication scheme described or perform authentication of this data in the application itself.

Many platforms support the ability to load different boot or configuration files from 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.

Use a FIPS 140-2 compliant, Level 2 HSM

Generate and store all authentication keys on a program-controlled, FIPS 140-2 compliant, Level 2 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. This is an LoA1 threat because it targets a specific program with a compromised executable. The adversary would have access commercially to the vendor software and, by reverse engineering, could modify the program 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 another threat and set of mitigations that applies at LoA2.

TD 7 mitigations

- For DoD system owners, 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.
- Use a protected computing environment.
- *Validate the cryptographic hash against the hash signed by the vendor.

TD 7 mitigation descriptions

Purchase from DoD authorized vendors and distributors

For DoD system owners, utilize 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 protected computing environment

The program can select the protected environment from the following options:

Option 1: Install and execute this software using one of the following computing platforms to protect from remote attack:

- A computer and network classified at the DSCA Secret level or above.
- 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. This is a computer with the vendor software installed by a network administrator. This admin should not be a designer working on the application design.

Option 2: Enforce compliance with Cybersecurity Maturity Model Certification, level 3 (https://www.acq.osd.mil/cmmc/).

Validate the cryptographic hash

Ensure the cryptographic hash 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.

In the event the hash is not provided or does not match the authorized version, the program can choose from the options below:

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

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

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

- The software used to mirror the program’s synthesis effort should 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 should utilize the same HDL code, IP, and synthesis scripts.
- The mirrored effort should 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, contact the software vendors for assistance.

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

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

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

- For DoD system owners, purchase from DoD authorized vendors and distributors. The DoD program acquisition group can provide this information.
- Authenticate the FPGA devices.
- Populate and inspect the SBCS.
- Document the steps taken to demonstrate compliance with TD 9.

TD 9 mitigation descriptions

Purchase from DoD authorized vendors and distributors

For DoD system owners, utilize DoD authorized vendors and distributors for all purchases. Authorized vendors can be identified through the acquisition organization.

Authenticate the FPGA devices

Authenticate the FPGA devices. In this mitigation, the program can either order SBCSs with unpopulated FPGA locations and perform the FPGA attachment, or the program can physically inspect a sampled set of pre-populated boards.

Populate and inspect the SBCS

The program should select one of the options below for the SBCS:

Option 1: Populate SBCSs in a classified or Trust Category 1 facility. If the program chooses to populate the SBCS boards with their own FPGA devices, they should adhere to the following caveats:

- Purchase an SBCS with empty FPGA location/bond-outs that can be populated by the program.
- Purchase the FPGA devices and perform the mitigations from *TD 2: Adversary inserts malicious counterfeit* for LoA1 on the FPGA devices to authenticate them.
- Populate the SBCS with the FPGA devices.
- The work should be conducted in a classified setting or Trust Category 1 facility.

Option 2: Populate SBCSs in an external facility with authentication in a classified facility. If the program chooses to populate the SBCS boards with FPGA devices in an external facility, they should adhere to the following caveats:

- Purchase an SBCS with empty FPGA location/bond-outs that can be populated by the program.
- Purchase the FPGA devices and perform the mitigations from *TD 2: Adversary inserts malicious counterfeit* for LoA1 on FPGA devices to authenticate them.
- Populate the SBCS with the FPGA devices.

Following SBCS population and return to the program, the FPGA devices should be authenticated in a classified facility prior to moving to the next stage of system integration. Authentication should be performed according to the guidelines found in the mitigations for *TD 2: Adversary inserts malicious counterfeit*.

Option 3: Purchase complete SBCS devices and physically inspect the FPGA devices contained on them. To perform physical counterfeit inspection with destructive

sampling, see guidance provided under *TD 2: Adversary inserts malicious counterfeit*. In addition verify the PCB connections by:

- Obtain and review the SBCS schematics for functional correctness, vulnerabilities, and security concerns as they relate to the FPGA configuration process and security connections.
- Post assembly, 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 SBCSs' FPGA configuration process by using SBCSs whose configuration process and board-level connections comply with the LoA1 mitigation requirements for *TD 6: Adversary swaps configuration file on target*. This includes, but is not limited to, requirements for:

- Strong authentication algorithms,
- Differential power analysis (DPA) resistant authentication,
- Strongly protected key storage,
- Strong anti-tamper detection and response,
- Freedom from known vulnerabilities in the configuration and security functions,
- Encryption and authentication key lengths compliant with the requirements outlined NIST SP 800-57, and
- The ability to disable FPGA test pins, such as JTAG.

Document the steps taken to demonstrate compliance

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

5. JFAC Survey Request

In support of these mitigations, JFAC asks all DoD 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 assist in identifying additional assurance gaps and mitigations. Refer to Appendix C: JFAC FPGA Reporting Template for the information a program should include in the email.

For more information on how this data will be used, contact JFAC_HWA@radium.nscs.mil

6. 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 8 threat descriptions, it can consider its FPGAs to have achieved LoA1.

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 JFAC_HWA@radium.ncsc.mil for assistance.

Appendix A: Standardized terminology

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Appendix B: IP reuse guidance

There are several situations in which a program or organization would like to reuse previously generated IP. This IP can be generated internally (i.e., from an authorized DoD program, but for a different program than originally developed) or externally (i.e., purchased IP). In either case, there are situations in which this IP will not need additional review. This appendix describes situations in which the IP should be evaluated and situations in which its use without an evaluation is permitted.

IP that was not generated or previously evaluated by a DoD program in conjunction with the DoD Microelectronics FPGA LoA1 Best Practices should be evaluated by the program before use. This includes cases in which vendors have had the IP evaluated by a third party. That review is not an acceptable alternative in accordance with FPGA Levels of Assurance Best Practices. Programs have the responsibility to perform or oversee all reviews.

Reuse Conditions

In general, at LoA1, IP reuse is acceptable as long as the following conditions have all been met:

- The IP has been developed internally for an LoA1 or higher compliant program or the IP was successfully internally evaluated at LoA1 or higher.
- All documentation associated with development or previous evaluation is cryptographically signed and stored within the configuration management system compliant with this DoD Microelectronics FPGA LoA1 Best Practices document. The documentation is provided to the new program in totality. The documentation remains unchanged since the time the evaluation was performed.
- The program cannot accept any IP in which the report has discrepancies from the version received. The name of the IP, version information, hash, and all other information matches exactly.
- After the initial development or evaluation, the IP remains unchanged. The IP should be stored in a configuration management system compliant with this DoD Microelectronics FPGA LoA1 Best Practices document. The hash of the IP is cryptographically signed and maintained separate from the actual IP. The hash was

maintained from the time of original usage through the time of reuse. The program verifies that the hash of the IP matches the stored hash value.

- In the event the IP was previously evaluated and assurance concerns were identified, these concerns should be documented and provided to the program that would like to reuse the IP. The program has the responsibility to accept or mitigate the risks based on individual program needs.

Reuse Scenarios

The following section describes several use cases that provide additional details of when IP can or cannot be reused at LoA1.

Scenarios in which LoA1 IP reuse is appropriate:

a. The program would like to reuse internally developed IP that was developed to be compliant with LoA1, LoA2, or LoA3, but not previously evaluated outside of the program use.

   In this scenario, the IP was developed internally using the processes described in the Levels of Assurance Best Practice documents. Therefore, the IP was previously proven compliant. To reuse the IP, the program should demonstrate compliance with the conditions outlined in the Reuse Conditions section.

b. The program would like to reuse internally developed IP that was developed to be compliant with LoA1, LoA2, or LoA3 and previously successfully evaluated to be compliant with LoA1, LoA2, or LoA3.

   In this scenario, the IP was developed and evaluated internally using the processes outlined in the Levels of Assurance Best Practice documents. Therefore, the IP was previously proven compliant. To reuse the IP, the program should demonstrate compliance with the conditions outlined in the Reuse Conditions section.

c. In this scenario the IP was developed by an external vendor. The 3PIP was previously verified internally at LoA1, LoA2, or LoA3. A different program would like to now use this IP for LoA1 purposes.

   In this scenario, the IP was evaluated internally using the processes outlined in the Levels of Assurance Best Practice documents. Therefore, the IP was previously proven compliant. To reuse the IP, the program should demonstrate compliance with the conditions outlined in the Reuse Conditions section.

Use Cases in which an LoA1 IP evaluation in accordance with Third-Party IP Review Process for Level of Assurance 1 document would be required:

a. The program would like to use internally developed IP that was not developed or evaluated to satisfy any level of assurance. The program would like to use this IP at LoA1.

   In this scenario, the program should treat the IP the same as unevaluated externally developed 3PIP. The program should follow the guidance provided within DoD Microelectronics FPGA LoA1 Best Practices to mitigate Threat Description 5: Adversary compromises third-party soft IP.

b. The program would like to use externally developed 3PIP version X.1. Version X.0 was previously verified to be LoA1 compliant.

   In this scenario, the IP has been modified. Due to the modification, the program should treat the IP the same as unevaluated externally developed 3PIP. The program should follow the guidance provided within DoD Microelectronics FPGA LoA1 Best Practices to mitigate Threat Description 5: Adversary compromises third-party soft IP.

c. At LoA1, the program would like to use externally developed 3PIP version X.0, which was verified by an independent third party at LoA1, LoA2, or LoA3.

   The program should treat the IP the same as not previously reviewed IP. The program should follow the guidance provided within DoD Microelectronics FPGA LoA1 Best Practices to mitigate Threat Description 5: Adversary compromises third-party soft IP.

d. At LoA1, the program would like to use 3PIP version X.1. Version X.0 was internally verified at LoA2 or LoA3.

   In this scenario, the IP to be used has been modified. Given the threats and level of access at LoA1, the program should treat the IP the same as unevaluated externally developed 3PIP. The program should follow the guidance provided within DoD Microelectronics FPGA LoA1 Best Practices to mitigate Threat Description 5: Adversary compromises third-party soft IP.

Appendix C: JFAC FPGA reporting template

Each DoD program is requested to provide the following information to JFAC. Multiple email addresses are provided to support a variety of classification levels; only one email to any of these is required. Please contact JFAC to obtain the appropriate email address at JFAC_HWA@radium.ncsc.mil.

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

==========

*** Please Portion Mark Appropriately ***

(U) POC Contact Info

(U) Name:

(U) Organization/Company:

(U) Email:

(U) Phone:

(U) Address:

(U) **Program Info**

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

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

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

(U) Estimated Number of Systems to be Built:

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

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

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

(U) FPGA Device Family:

(U) FPGA Device Part Number:

(U) FPGA Design Software Used and Version #:

(U) Description of Subsystem Containing FPGA Device:

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

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

(U) Source/seller of the FPGA devices:

(U) Date purchased:

(U) Anticipated Fielding date:

(U) LoA Level:

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

1.

2.

3.

==========

Example

==========

*** Please Portion Mark Appropriately ***

(U) POC Contact Info

(U) Name: **Jack Jackson**

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

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

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

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

(U) **Program Info**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(U) LoA Level**: 1**

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

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

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

==========

Appendix D: Guidance for embedded FPGA IP

LoA1 Introduction

While an embedded field programmable gate array (eFPGA) has many of the same functions and features of commercial FPGA devices, from an assurance stand point, it should be viewed as an application-specific integrated circuit (ASIC) third-party IP block. Assurance guidance for ASIC IP can be found in the JFAC ASIC/CIC Best Practices Guides and should be followed when considering the use of this macro in a custom design. However, there are some FPGA related assurance concerns that are inherent in this IP block and can addressed with guidance from this document. For any eFPGA use, the program should apply ASIC assurance best practices with the following FPGA additions.

The eFPGA is susceptible to a combination of ASIC and FPGA related threats. The table below illustrates what threat categories are relevant to eFPGA at LoA1 and which guidance to follow.

#	FPGA threat description (TD)	ASIC	FPGA
TD 3	Adversary compromises application design cycle	Yes	Yes
TD 4	Adversary compromises system assembly and keying and provisioning	NA	Yes
TD 5	Adversary compromises third-party soft IP	Yes	No
TD 6	Adversary swaps configuration file on target	No	Yes
TD 7	Adversary substitutes modified EDA software design suite	Yes	Not LoA1
TD 10	Adversary inserts a compromise during the ASIC fabrication process	Yes	No
TD 11	Adversary modifies FPGA software design suite	Not LoA1	Not LoA1

eFPGA Guidance

What follows is guidance provided to mitigate eFPGA-specific threats. This should be added to the steps already taken to assure the ASIC design and development flows.

TD 3: Adversary compromises application design cycle

The eFPGA IP block shares all the ASIC design cycle concerns with some additions from the FPGA space. The user is encouraged to reference the ASIC/CIC Best Practice Guidance for design cycle protections with the following FPGA enhancements. The

program should follow the FPGA guidance for the generation and protection of the configuration file and any associated cryptographic keys. The concerns here include an adversary compromising or stealing keys and modifying the configuration file while in storage. For this threat, it is recommended that the user follow the guidance found under TD 3 in this document.

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

The eFPGA macro shares the same requirement for the loading of cryptographic keys and configuration data as an FPGA. All of the FPGA threat surfaces in this area apply equally to the use of eFPGA IP. This includes the modification of the configuration file or keys, or their complete replacement during assembly and provisioning. The user should utilize the full guidance found under TD 4 of this document for the loading of their eFPGA data onto their system.

TD 5: Adversary compromises third-party soft IP

The eFPGA is a 3PIP hard macro and should be protected using guidance from the ASIC/CIC Best Practice guides for 3PIP. As this IP is delivered as a hard embedded layout macro, it is not able to be evaluated in the same manner as a soft HDL function. Thus, it should be evaluated as a hard macro according to the guidance provided by the CIC Best Practice Guide obtainable from JFAC.

TD 6: Adversary swaps configuration file on target

The eFPGA macro shares the same threat space as the FPGA with respect to the compromise of the configuration file after provisioning. The user should apply the full guidance found in TD 6 in this document. Unlike FPGAs, the function to perform authentication and store keys has to be designed and integrated into the ASIC by the program. This is not an easy task and does add cost and time to a development schedule. There are several means to achieve this protection and they are as follows:

- Integrate an authentication mechanism in the ASIC with key storage that follows the constraints for this type of cryptographic protections found under TD 6 of this document.
- Store the authenticated configuration file in memory on the ASIC itself. This protects the configuration file from being modified at rest. The ability to update or replace the configuration file memory space should be protected by authentication, using digital signatures or passwords.

- Store the authenticated configuration file in a stacked memory device in the ASIC package. This protects the configuration file from being modified at rest. The ability to update or replace the configuration file memory space should be cryptographically protected.

TD 7: Adversary substitutes modified FPGA software design suite

Unlike a commercial FPGA, the details of the interface between the eFPGA fabric and the ASIC logic is not available to the eFPGA vendor. Additionally, the eFPGA is custom built per the customer's required size and layout. Finally, the eFPGA does not have the plethora of additional functions embedded in a commercial FPGA, such as security, tamper, configurable IO, and SOC features, to compromise. An adversary would have to have specific knowledge of the program's application, its pinout, and the ASIC logic interfacing to the application. This would have to be combined with detailed knowledge of the vendor tools and an ability to swap a modified version for the authorized version. These facts would make it very difficult to coordinate an attack on the ASIC through the eFPGA software. Additionally, the access requirements would push this threat out of LoA1 into a higher level and therefore is not addressed here from an FPGA perspective.

TD 10: Adversary modifies FPGA software design suite

This threat does not meet the requirements for LoA1 for the same reasons as TD7. ASIC guidance should be followed for this threat at LoA1 as applicable.

Appendix E: Checklists and data/documentation requirements

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

TD 1 mitigations	**Data/Documentation requirement**
Use caution when selecting tools or platforms	The program should document the name of the person performing the research on tools/platforms, the date timestamp of the research, the research results, and 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.
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.
If vulnerabilities are found when researching vulnerabilities, choose one:	
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), the search results, the name of the person performing the search, and the date timestamp of when the search was performed.
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.

TD 1 mitigations	Data/Documentation requirement
Use a revision control/version management system	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.
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 timestamp, why the change was requested, and who made the change with date timestamp.
Perform a vulnerability data review	The program should ensure vulnerability data is reviewed by two distinctly independent individuals. The program should obtain the results of independent reviews to include: • Type and extent of verification performed, including evaluation objective, data used, sources, etc. • Findings, both positive and negative, for all evaluations performed • Risks identified by the review team (e.g., quality issues, vulnerability to threats, etc.) • Recommendations, if any

TD 1 mitigations	Data/Documentation requirement
Perform routine employment monitoring	Maintain employment records to include application, background checks, etc. Audit employee assessments, and discipline logs.
Prevent a compromised insider from hiding a vulnerability by choosing one of the following options:	
Option 1: Perform independent reviews	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, showing that the reviewers are independent from the team doing the design • Time/date stamp of when the review was performed
Option 2: Use personnel with Secret-level clearance	The program should keep a log of personnel assigned along with their clearance level.

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 the vendor is authorized.

TD 2 mitigations	Documentation requirements
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
Validate the authenticity of the FPGA device. Choose one:	
Option 1: Cryptographically secure ID	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.
Option 2: Physical 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 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.

Checklist for TD 3: Adversary compromises application design cycle

TD 3 mitigations	Documentation requirements
Use cleared personnel in a cleared environment	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, along with their clearance level. The program should maintain audit logs stating what each team member accessed. The program should maintain SSP documentation.

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, 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, what decision was made regarding the change, who made the decision with date

TD 3 mitigations	Documentation requirements
	timestamp, why the change was requested, and who made the change with date timestamp.
Use a revision control/version management system	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 a reproducible build	Document the reproducible build process and results validating that the separate builds produce the same binary and hash.
3.2 Mitigating the modification of test benches/plan to reduce coverage or hide Trojan code	

TD 3 mitigations	Documentation requirements
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, 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 should 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 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,

TD 3 mitigations	Documentation requirements
	etc. should be resolved via a change process utilize a change management system. The established processes should be documented, enforced, and audited.
Ensure the test environment is maintained via configuration management	The program should maintain the test environment based on the configuration management plan in accordance with requirements of CMMC Level 3 or *NIST SP 800-171 Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations* and *NIST SP 800-172 Enhanced Security Requirements for Protecting Controlled Unclassified Information*. The program should maintain the CMMC audit results or NIST SP 800-171 self-assessments.
Use a revision control/version management system	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.
3.3 Mitigating the introduction of Trojans into the application design during development	
Ensure all design artifacts have a direct bi-directional link to approved requirements	The program should document bi-directional traceability for all device requirements, including derived requirements.

TD 3 mitigations	Documentation 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 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, 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 should 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.

TD 3 mitigations	Documentation requirements
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, version 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 utilize 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. Document any functional discrepancies and how those discrepancies were resolved.
3.4 Mitigating the introduction of compromised tooling/software into the environment	
Accept only digitally signed software deliveries	The program should document the software author listed in the digital signature, the timestamp of the digital signature, and the timestamp of when the digital signature was validated.
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), 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
If vulnerabilities are found when researching vulnerabilities or tools, choose one:	
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), the search results, the name of the person performing the search, and the date timestamp of when the search was performed.
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.
Use a revision control/version management system	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.
Utilize a reproducible build process	Document the reproducible build process and results validating that the separate builds produce the same binary and hash.

TD 3 mitigations	Documentation requirements
Select a formal "proof" process	Document all code that was reviewed using LEC. Document any functional discrepancies and how those discrepancies were resolved.
3.5 Mitigating intrusion into the internal network	
Use a revision control/version management system	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.
Assign privileges and accesses based on 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 the vendor is authorized.
Choose one protected computing environment option:	
Option 1: DSCA Secret Level network	Maintain log of personnel with clearance information. Maintain all records in accordance with maintaining a DSCA Secret network as well as a documented and enforced SSP.
Option 2: DMEA Certified network	The program should maintain proof of DMEA certification.
Option 3: Network-Isolated Computer Enclave	The program should maintain documentation with the enclave layout, open ports, etc.
Option 4: Enforce CMMC level three requirements or implement the latest approved version of NIST SP 800-171 and NIST SP 800-172.	The program should maintain Cybersecurity Maturity Model Certification (CMMC) audit data. Until CMMC is the program requirement, the program should maintain a self-assessment demonstrating compliance with *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.*
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, what decision was made regarding the change, who made the decision with date timestamp, why the change was requested, and who made the change with date timestamp.

TD 3 mitigations	Documentation requirements

Track critical data in revision control	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, 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.

Adopt a structured application 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 design review with roles and responsibilities 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.

TD 3 mitigations	Documentation requirements
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
Enforce reviewer criteria with one of the following options:	
Option 1: Perform independent third-party reviews	In writing, the program should designate work that must be done by independent third-party reviewers or cleared personnel. The program should keep a log of personnel assigned to that work with their clearance level, the data reviewed, date timestamp and duration of the review, and the results of the review, in addition to audit logs demonstrating what each team member accessed.
Option 2: Perform independent reviews by Secret-cleared individuals	In writing, the program should designate work that must be done by independent third-party reviewers or cleared personnel. The program should keep a log of personnel assigned to that work with their clearance level, the data reviewed, date timestamp and duration of the review, and the results of the review, in addition to audit logs demonstrating what each team member accessed.
Option 3: Perform independent reviews by cleared individuals in a DSCA Secret-level facility	In writing, the program should designate work that must be done by independent third-party reviewers or cleared personnel. The program should keep a log of personnel assigned to that work with their clearance level, the data reviewed, date timestamp and duration of the review, and the results of the review, in addition to 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 and vendor authorized distributors	The program should document the name and location of the authorized vendor along with documentation demonstrating 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. This plan should specifically address DoD 5220.22-M, Chapter 5 Section 3. 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
Provide keys and configuration data	The program should document assembly house receipt of data packages and the hash value of the packages.
Clear memory devices	The program should document the company, location, individual, and method for clearing the contents along with the contents before and after clearing.

TD 4 mitigations	Documentation requirements
Provision private keys	The program should document: • 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
Select an option and verify to protect the FPGA from attack during assembly and provisioning:	
Option 1: Assemble and provision the system in a DSCA Classified Secret or Trust Category I certified facility	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 Cat I certification
Option 2: Assemble and provision in an external unclassified facility	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 CMMC audit
Verify assembly and provisioning 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

TD 4 mitigations	Documentation requirements
	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 FPGA device, where the authentication was performed, by whom, when, and the results of the verification.
Choose one option to authenticate the FPGA device:	
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 the vendor is authorized.
Only accept IP that is unobfuscated	The program should keep a copy of the clean unobfuscated code, along with the name and or ID of the person who received it.

TD 5 mitigations	Documentation requirements
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.
Check IP into revision control	The program should include the initial IP and hash check-in within the system.
Examine IP for malicious functions	The program should document all results in accordance with *Third-Party IP Review Process for Level of Assurance 1*. All interaction with JFAC regarding examining IP for malicious functions should be documented.

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 • The verification process used to test the authentication method
Prevent direct read back	The program should document the steps taken to prevent direct read back of private keys.
Use a CNSS/NIST-approved algorithm and key length	The program should document the key length being used along with the version number of the latest FIPS guidance and the approved key length in accordance with the guidance.
Disable operation or use of test access pins	The program should maintain documentation including the means by which the JTAG test pins were disabled.

TD 6 mitigations	Documentation requirements
Ensure authentication is enabled for application 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.
Use a 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 showing 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 the vendor is authorized.
Prevent automatic tool updates	The program should document, maintain, and follow the SSP.
Use a protected computing environment	The program should maintain documentation 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 all records in accordance with maintaining a DSCA Secret network. • 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. The documentation should include a log of personnel along with their clearance level and a documented SSP for all networks.
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.

TD 7 mitigations	Documentation requirement
If the hash is not provided or does not match, validate the tool output using one of the following options:	
Option 1: Perform logical equivalency check	Document all code that was reviewed using LEC. Document any functional discrepancies and how those discrepancies were resolved.
Option 2: Use a reproducible build process to validate the software	Document the reproducible build process and results validating that the separate builds produce the same binary and hash.

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

TD 9 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 the vendor is authorized.
Authenticate the FPGA devices	The program should document the physical inspection results for each slash sheet and UID for the device inspected.
Choose an option to populate and inspect the SBCS:	
Option 1: Populate SBCSs in a classified or Trust Category 1 facility.	Document the company name, responsible division, and physical address of the verification location, along with the associated work order, quote, and purchase order (PO).
Option 2: Populate SBCSs in an external facility with Authentication in Classified Facility.	Document the company name, responsible division, and physical address of the verification location, along with the associated work order, quote, and purchase order (PO).

TD 9 mitigations	Documentation requirement
Option 3: Purchase complete SBCS devices and physically inspect the FPGA devices contained on them.	Document the company name, responsible division, and physical address of the verification location, along with the associated work order, quote, and purchase order (PO).
Document the steps taken to demonstrate compliance	Document the steps taken to comply with these requirements. This includes hardware and software features.