National Security Agency
Cybersecurity Technical Report

DoD Microelectronics: Third-Party IP Review Process for Level of Assurance 2

February 2023

U/OO/120909-23
PP-23-0200
Version 1.0

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

Notices and history

Document change history

Date	Version	Description
FEB 2023	1.0	Initial Publication

Disclaimer of warranties and endorsement

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

Publication information

Author(s)

National Security Agency
Cybersecurity Directorate
Joint Federated Assurance Center

Contact information

Joint Federated Assurance Center: https://jfac.navy.mil

Defense Industrial Base Inquiries and Cybersecurity Services: DIB_Defense@cyber.nsa.gov

Media inquiries / Press Desk: Media Relations, 443-634-0721, MediaRelations@nsa.gov

Purpose

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

Executive summary

This report outlines a methodology of design review sufficient to validate that third-party intellectual property (3PIP) is appropriate to incorporate into a Level of Assurance 2 (LoA2) system.

In this context, 3PIP refers to 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.

This process is intended to mitigate the Threat Description #5 “Adversary compromises third-party **soft** IP,” as described in the *FPGA Best Practices – Threat Catalog*. The process attempts to minimize the required level of effort while providing sufficient mitigation for use of 3PIP at LoA2. The review process, when correctly executed, establishes an independent organization that an adversary would have to compromise or deceive for an attack to succeed. This directly increases the level of ***access*** required to carry out an attack. The process assumes that technology and level of effort are not substantial barriers to an adversary in modifying the 3PIP design.

Additionally, at LoA2 only inherently targetable and high-utility attacks are of concern. Attacks that provide pre-positioning as part of a more complex attack to be executed in the future are out of scope. This restriction bounds the steps of review that are required. However, should issues outside the scope of highly targetable attacks be found during this review process, they must be reported as appropriate[1] and the intellectual property (IP) should not be approved for use in an LoA2 system.

This review process can be applied in two distinct ways. One is to approve a 3PIP in general. The second is to approve a 3PIP for use in a specific role within a specific system. For instance, this process can be used to review a memory controller IP for use in any application. In some circumstances, it may be less expensive to review that same

[1] Appropriate action will depend on the specific discovery. Coding errors that appear innocent may require cybersecurity reporting, and in general should be reported to the IP vendor. Apparently intentional backdoors should be reported to the Federal Bureau of Investigations (FBI) or the National Intellectual Property Rights Coordination Center. JFAC is available for consultation should the correct action be unclear.

IP, but only for use in a single application. In this case, information about the system around the IP might be used to avoid reviewing certain components in detail.

The JFAC *FPGA Level of Assurance 2 Best Practices* appendix specifies this flow as one option to assure 3PIP. In most cases that will correspond to review within a certain system. The other mitigation offered is to provide the 3PIP to the JFAC. When JFAC elects to review the 3PIP, it will be a review for any system context.

This review process is summarized below:

- Assemble review prerequisites
- Establish suitability of the 3PIP for review
- Partition the design
- Perform manual code review
- Perform test-driven code review
- Document and sign the review package

Contents

1. Introduction

This report outlines a methodology to perform a design review that is sufficient to validate a third-party intellectual property (3PIP) function for use within a Level of Assurance 2 (LoA2) system. In this context, 3PIP refers to 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.

2. Assemble review prerequisites

For a 3PIP review to be valid, the review itself must be thorough and auditable. Procedural elements related to the following must be in place for the duration of the review:

- A clear record of who is conducting the review, and
- The review process itself.

2.1 Reviewer documentation

A clear record of the individuals conducting the review must be maintained. This record must justify why the individuals are well suited to the task. The information required is similar to the information collected by a company during the hiring process, such as identification information and a resume. The following details must be included:

- **Security suitability** – For LoA2 3PIP reviews, the individuals doing the review must be U.S. persons. The organization performing the review must maintain documentation proving that status.
- **Qualifications** – Each reviewer must have a suitable degree and experience with the design language or circuit type they review. For instance, a reviewer of Verilog code must have an understanding of digital design typically acquired with a degree in computer or electrical engineering and experience writing Verilog. Likewise, a reviewer of Very High Speed Integrated Circuit Hardware Description Language (VHDL) must similarly have experience with VHDL. Furthermore, if the 3PIP incorporates hard IP blocks, specific to the Field Programmable Gate Array

(FPGA) platform, the reviewer must have experience with platforms from that vendor[2].

- **Independence** – No reviewer should have worked on the development of the 3PIP. No reviewer may be a current employee of, or contractor to, the company that designed or sells the 3PIP.

2.2 Record of review

The review process itself must be auditable such that each decision is traceable to at least two specific reviewers. In addition, time spent conducting the review must be recorded for each decision. The time expected to complete the review will vary greatly depending on block complexity. The specific information to gather and document is similar to the information collected automatically by many code review tools. Specifically, for each artifact reviewed, the following must be maintained:

- **Identity of the reviewer** – The name, or other identifier, of the individual who performed the review.
- **Times of review** – A record describing the day or days on which the review took place, as well as the duration of the review.
- **Specific material reviewed** – A record of the specific design files reviewed to make the determination.
- **Tests run during the review** – A record of any functional tests run by the reviewers, including test benches, scripts, and other inputs necessary to replicate them.
- **Summary** – A record explaining the decision made, and the facts that led to that determination.

In addition, this documentation must record how the IP was received, how it was stored, and how it was distributed to the reviewers. A process should be put in place to ensure that all reviewers receive the same versions of each file.

This record of review should be stored in the revision control system, but does not need to be distributed as part of the final signed review package.

[2] For example, if a 3PIP makes explicit use of a multiply accumulate block in an Intel FPGA the reviewer must have experience with Intel FPGAs, or previous related Altera FPGAs.

3. Establish suitability of the 3PIP for review

IP can only be reviewed if it is documented and developed using practices that cannot be exploited to hide malicious changes. That means well-documented hardware description language (HDL) with clean names and reasonable design. Human and simulation review of the IP are not effective outside of those conditions. This section introduces a set of criteria that determines whether such practices are employed.

This step determines if a review of the 3PIP is feasible in a way that will generate sufficient assurance. A single individual can conduct this process by reviewing the 3PIP and asking these guiding questions:

- Is the 3PIP distributed in an encrypted form that prevents the reviewers from viewing it? Is it distributed as a netlist[3]?
    - If yes, the 3PIP is not suitable for review. It is not recommended for this flow.
    - Viewing the unencrypted code, or pre-synthesis code, is only sufficient with additional steps to ensure equivalence of the distributed and reviewed files. If this is needed, contact JFAC for guidance.
- Is the 3PIP obfuscated[4]?
    - If yes, the 3PIP is not suitable for review. It is not recommended for this flow. JFAC can be contacted if the 3PIP is essential; however, the process to approve such a 3PIP will be expensive, time consuming, and likely to return a negative result.
- Is the 3PIP clearly organized, with distinct modules that are limited in scope with inputs/outputs (I/O) limited to those needed to achieve their scope?
    - If no, the 3PIP is not suitable for review. It is not recommend for this flow.
- Is the 3PIP well documented? This should include comments describing the purpose of modules; clear, human-readable names; and top-level documentation describing the interface.

[3] Whether a 3PIP is distributed as a netlist is not determined by the file format, but rather the contents. For example, a Verilog file is considered a netlist distribution when a substantial amount of it is implemented with primitive or gate-level elements. This could either be primitives drawn from a library, such as a vendor specific primitive library, or primitives that implement individual gates, such as the “and” and “or” elements in Verilog.

[4] 3PIP for which module names, signal names, or any other names, have been replaced with non-meaningful alphanumeric codes or random words is considered to be obfuscated. Similarly, HDL that makes repeated use of complex Boolean operators, where it could have used complex functional commands, such as if/then blocks, is considered obfuscated.

  - If no, the 3PIP is not suitable for review. It is not recommended for this flow.

- Does the 3PIP feature parameterized code and pre-compiler directives where parameters are used to explicitly generate new HDL[5] that is then added to the system? Is software used to write HDL based on designer input? This includes explicit code that writes code, uses terms like 'ifdef' in Verilog, and uses the term 'generate' in VHDL. This includes parameterized IPs that can be configured via GUI or script.
  - If yes, the 3PIP is still suitable for review. Be aware that the review process will be longer.
- Does the 3PIP have a substantial number of tests distributed with it?
  - If no, the 3PIP is still suitable for review. Be aware that the review process will be longer.

4. Partition the design

Once review suitability is established, the team reviews the overall design of the 3PIP and makes a plan to split the 3PIP into one or more functional units that can be reviewed and tested independently. This number can vary greatly depending on the scale of the 3PIP being reviewed. For some simple 3PIP modules, there may be no clear partitioning of the code into meaningful, functional modules. For others, there may be a clear top-level design that quickly yields multiple functional modules.

4.1 Threats of interest

Because this process is specifically intended to catch LoA2 threats, the review team should begin with analyzing targetability. At LoA2, this means high utility threat operations are executed in a way that provides straightforward means to understand and predict the effect of an attack and 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

[5] Generator code that exists solely to add additional logging for debug should not be counted as generator code for these purposes. Generator code added to make a simulation work must be reviewed carefully in the process below, but should not count as generator code for this purpose.

trigger, which therefore cannot be controlled or timed in a controlled way, does not fall in this category.

The focus of this review is to look for malicious modifications to 3PIP. Because 3PIP can have many origins, as well as many distribution methods, an adversary has many options. In particular, an adversary can modify one copy of 3PIP destined for a particular target. This means that an in-depth 3PIP review is required.

The JFAC best practice guides do not specify what compromises are acceptable; only the scope of search that is required. If any flaw, intentional or otherwise, is identified, it should be reported and may render a 3PIP unsuitable for use.

Note: While these guidelines considerably limit the scope of attacks that reviewers must look for, they do not limit the scope of attacks that must be reported if discovered.

However, there are some limits to the level of review required that enable specific sub-portions of a 3PIP design as “of interest” for review. These attacks are divided into two classes: controlled effects and persistent effects.

4.2 Controlled effects

A controlled effect is just that: a specific control mechanism, coupled to a specific effect. In more detail, such an attack contains both of the following:

- **A means of command and control** – The attack must be externally controlled through some pre-existing or additional control mechanism. Within LoA2, assume that such an effect must not occur during acceptance testing or typical use of the device. To achieve this, a trigger must either be sufficiently long and unstructured or represent an out-of-spec behavior in a defined protocol.
    - As a rule of thumb, unstructured triggers of interest must contain, at a minimum, 32 bits of unstructured data. Logic to recognize such a sequence could be implemented many ways, but any implementation would contain at least 5 bits of state, as well as substantial additional logic comprised of combinatorial and state elements.
    - Structured triggers of interest must occur within the context of a specific defined protocol, and cannot simply represent any out-of-spec behavior. For instance, an incorrect cyclic redundancy check (CRC) field is overly

broad, as many systems will encounter incorrect CRC fields. However, a specific bit error pattern within a CRC field could be sufficient.

- **A useful effect** – Within LoA2, useful effects can include specific actions on the part of the system, denial of service, leak of data, and so on. The presence of denial of service in this list means that most any functional area in a 3PIP design will cause a useful effect.

Example that meets the criteria of a controlled effect: ***A modification to a sub-block of a 3PIP Ethernet filter waits for a specific magic packet before deactivating the Ethernet entirely until the system is reset.***

This attack is controllable: the adversary knows the protocols parsed by the core at design time and they can understand its role in a specific system. It is useful: it would allow an adversary to deactivate an important function of the device on command. This is an example of a controlled effect that would be of interest.

Second example that meets the criteria of a controlled effect: ***A modification to a programmable filter (e.g., a finite impulse response filter) is added that waits for a specific input pattern, after which it causes the filter to output only 0.***

This attack is controllable if the adversary understands the role of the filter in the system to process external data. For example, in a sensor application where data is preprocessed by a filter before being passed on to other parts of the system. While an adversary cannot control the exact values of the inputs in many such applications, they can control patterns in those values that can be detected by their additional code. In a sensor application such as this, this modification could give an adversary the means to deactivate a sensor, which is a denial of service attack.

Example that meets the criteria of a controlled effect for LoA2: ***A modification to a microcontroller core deactivates memory protection when a specific sequence of instructions is run.***

This attack meets the criteria at LoA2, as this would be considered prepositioning for an attack. If this is found, it is assumed there is also a secondary trojan able to take advantage of the memory deactivation and execute alternative code. If this is discovered, it must be reported and would represent a negative finding.

4.3 Persistent effects

In contrast to a controlled effect, a persistent effect lacks an external activation mechanism. Instead, it is always or regularly present. Persistent attacks against FPGA 3PIP are constrained principally by testing performed during development and ordinary use of the device. To be useful, they must evade detection during functional testing conducted by the designer and cannot immediately interfere with use of the system. This is entirely feasible depending on the test methodology chosen by the system designers and the 3PIP block itself.

For this review, the reviewers may assume that two categories of tests are run on the 3PIP. The first is the set of any tests or exemplar use cases distributed with the 3PIP itself. The second is a system-level functional test performed to validate that a complete system using the 3PIP successfully achieves a system-level outcome. When reviewing a 3PIP block for a specific program, rather than for general use, the team may also take as an input any additional tests developed for the block itself. However, the team should not assume that the 3PIP developer ran additional tests not shared with the review team.

Example that meets the criteria: *A multipoint bus 3PIP provides direct access from the Joint Test Action Group (JTAG) port of the FPGA to the bus.*

This attack is specifically useful, giving the adversary access to arbitrary data on the bus, which could easily include critical or classified data. It would be expected to be quickly discovered if placed into a widely distributed piece of 3PIP as soon as a user tried to use the JTAG port themselves. However, the attack being mitigated by this approach is specifically looking at the possibility that a vendor receives a special version tailored just for them. An adversary in this scenario may have identified that JTAG is unused in a given system. This attack also could be detected by extensive testing and evaluation of the final system. As stated above, reviewers should assume that no test would have involved JTAG if the system was not implementing JTAG intentionally.

Example that <u>fails</u> to meet the criteria: *A modification to a PCI-E engine causes it to degrade system performance*.

This does not meet the criteria for LoA2. Such a modification has no controlled effect, but instead degrades the performance at unknown times that are not necessarily useful to the adversary. As a result, this is not a concern at this assurance level.

4.4 Defining a functional area

The remainder of this report discusses how to evaluate each functional area. To do this, the review team must determine the functional area boundaries they want to use. A functional area is a set of circuit designs that have a specific documented purpose. Each functional area should be reviewed and tested independently. The evaluators can set the exact scale of these functional areas. However, as is specified below, depending on the design practices employed, the process may necessitate that functional areas be combined into larger functional areas for evaluation, or generate considerably more work bouncing between regions.

The review team should make a first attempt at defining functional areas based on the file structure, module documentation, and past experience. In general, small functional areas will be easier to evaluate. Design practices that limit the connections to a functional area to those needed to perform its function will enable small functional areas. In practice, design best practices encourage limiting connectivity between modules and thus this process should be straightforward.

4.5 Criteria for functional areas of interest

Based on the effects described at LoA2, the following questions should be used to determine whether a functional area is inherently of interest and what level of review is required. For example, review for controlled effects is much simpler than review for persistent effects:

- Does the functional area have I/Os that an adversary will be able to influence? When evaluating a 3PIP for a specific system, system knowledge can be used. When evaluating a 3PIP block in general, almost all I/O to the 3PIP block meet this criteria.[6] Additionally, inputs that are indirectly connected to top-level I/O through other modules must be reviewed.
    - If so, it is a functional area of interest for controlled effects. All avenues from external stimuli that may affect the device during normal use should be noted for investigation.

6 In certain cases it is possible to rule out external connections and influence when clear best practices would prevent this behavior. Specifically, in the case of JTAG and other test infrastructure, the final write-up should simply note that these must not be exposed remotely to an adversary. They do not need to be reviewed as remote connections, as any system where an adversary could access them is compromised by definition.

- Does the functional area have internal behaviors that serve a well-defined role in the system, but could fail in ways that would not be apparent during normal use?[7]
    - If so, it is a functional area of interest for persistent effects. The specific "difficult to test" internal behaviors should be noted for investigation.
- Does the functional area have a well-constrained set of I/Os that limit it to information relevant to its intended function?
    - If not, it needs to be re-evaluated in combination with the other functional areas connected to it as a single, larger functional area.

5. Perform manual code review

Each region requires two kinds of reviews: manual and test-driven. These two reviews may influence each other and do not need to be completed in a particular sequence. A manual review is a traditional "code review" where knowledgeable designers read the code to evaluate it for correctness.

Before beginning manual code review, automatic tools should be used to generate lines of interest within the code. Both a lint tool and the synthesis process must be run. A lint tool can be run with settings derived from the apparent code standard used in development of the source. Any lines identified by the lint code should be marked for extra review during the review process. Additionally, the 3PIP should be synthesized with an appropriate FPGA synthesis tool. Any warnings generated should be marked for extra review.

Each functional area of interest must be reviewed independently. For functional areas that are inspected for a controlled effect, this review should focus on looking at the data path for unexpected behaviors or unexpected connections from the functional area to other functional areas. For persistent effects, the specific hard-to-test function should be evaluated in depth for correctness.

For assessing a controllable effect, the following factors must be evaluated:

- Where does the information from the I/O propagate?

7 An example of such a region would be encryption on an internal communications bus. If the bus is responsible for encryption at one end, and decryption on the other end, no normal user of the bus would necessarily notice an error.

    - If it propagates largely intact or is transformed in a simple way to other functional modules, then they must be added as functional areas of interest.

- What logic handles the I/O? The logic within the functional area that processes the I/O must be evaluated for signs of a trigger:
    - If the logic is simple, then accounting for the state elements can be sufficient to remove it from consideration. This is possible when many bits of previous input are not stored natively within the functional area. In these cases, that storage, or a finite state machine (FSM), would need to be added in order to add a trigger.
    - If the logic is complex, it will require additional investigation:
        - One phase of the investigation should focus on reviewing the code for an unexpected, but explicit trigger. That is to say, specific lines of HDL or specific groups of gates, waiting for a specific command.
        - The second phase of the investigation should focus on more subtle effects. In particular, this should look at each FSM state not covered by a test, and validate that it has a well-documented and meaningful purpose. Special attention is needed if multiple interacting FSMs are present. In those cases, additional in-depth consideration must be paid to the interactions between FSMs. Particular attention must be given to out-of-spec behavior in known protocols.

For investigating a persistent effect, the process is more complex. Potential persistent effects at LoA2 are rare. However, appropriate experts must determine whether each such block is properly implemented. They must focus on the correctness and suitability of the implementation, with the entire implementation being carefully reviewed. Review of persistent effects is more dependent on the test-driven code review.

6. Perform test-driven code review

As part of the review process, simulation tests must be used. The simulation testing can be performed at the top 3PIP level, the functional module level, or a mix of the two. With line-by-line code review, the test benches that were provided with the 3PIP may be used as part of these tests. JFAC will provide guidance for blocks too large to efficiently simulate.

For regions that are being investigated for a controlled effect, sufficient tests must be developed such that all major desired functions are executed. In some cases, this may correspond to a requirements document or to the functions listed in a specification. In addition to that, code coverage[8] must be recorded and any lines not included in the final code coverage must be reviewed in detail, with a specific note identifying why they are not of interest in the context of their functional module.

For regions that are being investigated for a persistent effect, sufficient tests must be developed such that all major desired functions are executed. In addition to this, constrained random testing must be performed on the 3PIP. Code coverage must be recorded. In some cases, such as redundant serial safety checks, it may be impossible to achieve 100% code coverage. For each line not covered in the final code coverage, a specific note must be recorded identifying why a test bench did not reach that line and why it is acceptable in the context of the larger system.

In both cases, formal methods that prove equivalency between another reference design that has been vetted through this process and a 3PIP or functional module can substitute for the use of testing as described above. Similarly reviewed software blocks can also be used as a reference in this formal verification process.

7. Document and sign the review package

A review package should summarize the results of the review and minimally provide the following items:

- The name of the vendor who sells the 3PIP;
- The name of the 3PIP;
- Any version information of the 3PIP;
- The hash of the 3PIP package delivered by the vendor[9];
- A list of any use restrictions that were identified in the review of the 3PIP;
- The hash of a summary document detailing the review process, including all auditable information specified in the prerequisites section of this document;
- The auditing organization that maintains this summary document;
- The portion of the summary document above not containing personally identifiable information;

[8] For the purposes of this specific document, code coverage is measured by statement and branch coverage, not signal or gate.

9 This review will be valid solely for 3PIP packages that match this hash.

- The hash of all test benches and scripts used in the review process;
- If allowed by the 3PIP vendor, the collection of the test benches and scripts used in the review process;
- A summary hash of all the listed elements above, including hashes of packages reviewed but not the package itself; and
- A cryptographic signature of the above information, tied to the reviewing organization.

This set of information is a tradeoff designed to produce a distributable report not constrained by agreements that can be validated back to the specific files. The hashes in this report must be validated against the 3PIP for it to be used, and the report must be maintained as a design artifact for future audits. For instance, explicitly include a hash of the 3PIP, but not the 3PIP itself, so that other users can determine if they have the same files without needing to include the files themselves in the document.

Should an issue be identified during the review process, it should be reported to the organization requesting the review. In the case of Department of Defense organizations, any identified issue must be shared with the JFAC.

8. Pre-compiler and other machine-generated code

Machine-generated code refers to any code that takes a set of parameters and uses them to construct or pre-process HDL before a synthesis process. This can be custom C code that generates Verilog. This can be compiler directives such as ‘ifdef’ in Verilog or ‘generate’ in VHDL.

Review of a 3PIP block containing machine-generated code can be substantially more challenging, and, depending on the complexity of the pre-compiler directives or scripts, may not be possible. Instead, it may require review with specific settings in place. Contact JFAC for additional guidance for handling machine-generated code.

When reviewing 3PIP for use in a specific system, it is acceptable to run the pre-compiler with the desired parameters and evaluate the result. In this case, the generated HDL must comply with the same standards as specified for any 3PIP block. In addition, the review package must note those precise settings as a use restriction on the 3PIP.

9. Summary

For review of a 3PIP in general, the following guidelines govern the review:

- Assemble review prerequisites:
  - In cases where software is used to write HDL, members of the review team that review the generator code must have experience in both the language that the generator is written in and the language that is generated. For instance, to review C code that generates VHDL, the reviewer must have expertise in both.
- Establish suitability of the 3PIP for review:
  - Are the purposes of all uses of pre-compiler directives or scripts clear and well documented?
    - If not, the code is not suitable for review.
  - Are the parameters of the IP well documented with clear constraints?
    - If not, the code is not suitable to review.
  - Is the function of the 3PIP limited to a specific function, rather than an arbitrary programmable set of functions[10]?
    - If so, the code is suitable to review.
- Partition the design:
  - Partitions must be made with specific regard to the boundaries of the pre-compiler directives. The I/O between partitions should not dynamically change based on parameters outside of width changes.
- Perform manual code review:
  - Manual code review must be performed with the pre-compiler directives in place. The generator itself must be reviewed.
- Perform test-driven code review:
  - Simulations must be done on a representative sample of parameterizations.
  - All conditional code blocks within ‘ifdef,’ ‘generate,’ or similar statements must be included in at least one of the samples simulated, though individual lines within a block may be reviewed as in the normal flow.
- Document and sign the review package:
  - List the specific details of all parameter sets used in the evaluation.

10 For example, a parameter specifying the length of a filter is not problematic. A parameter indicating whether a CPU has a JTAG port is not problematic. A parameter that specifies a complex equation or programmatic behavior to implement is problematic.

- Include this data in the final summary hash.

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.