IP Security Assurance Standard

Whitepaper

September 4, 2019

Authors

Brent Sherman, Intel Corporation

Mike Borza, Synopsys

James Pangburn, Cadence Design Systems, Inc.

Ambar Sarkar, NVIDIA Corporation

Wen Chen, NXP Semiconductors

Anders Nordstrom, Synopsys

Kathy Herring Hayashi, Qualcomm

Michael Munsey, Methodics

John Hallman, OneSpin Solutions

Alric Althoff, Leidos

Jonathan Valamehr, Tortuga Logic, Inc.

Adam Sherer, Cadence Design Systems, Inc.

Ireneusz Sobanski, Intel Corporation

Sohrab Aftabjahani, Intel Corporation

Sridhar Nimmagadda, Qualcomm, Inc.

Table of Contents

Abstract

A System on Chip (SoC) or Application Specific Integrated Circuit (ASIC) is comprised of multiple components referred to as Intellectual Property (IP) blocks or just IP. These blocks come from multiple sources such as internal development teams, IP suppliers, tool-generated IP, etc. Typically, the SoC/ASIC owner integrates multiple IPs from multiple sources, which raises concerns about security risk. How much risk is the Silicon owner (i.e., Integrator) inheriting? What potential security concerns exist that the Integrator must address to ensure the security objectives of the SoC/ASIC are upheld? This paper introduces an emerging new standard called IP Security Assurance (IPSA) to address these concerns in a manner that is low-overhead, non-disruptive, and scalable across IP families. The standard specifies an approach to highlight IP assets and associated entries in the Common IP Security Concerns Enumeration (CIPSCE) knowledge base for the mitigation implementer to address.

Introduction

This paper is Accellera's initial proposal to address the industry's security concerns involving IP integration. Since Integrators typically treat IP as a "black box", vulnerabilities may inadvertently be inserted into an SoC/ASIC. To highlight this, a study was performed [1] showing what could happen in this scenario; a real-case example [2] was discovered in 2018.

Table 1 shows the goals and corresponding methods of the IPSA standard. The stakeholders of the standard are Electronic Design Automation (EDA) vendors, IP suppliers, and IP Integrators. The standard also assumes the relationships between the stakeholders are trusted and there are no malicious actors. The standard does not address security concerns in the supply-chain flow between the stakeholders.

#	Goal	Method
1	Develop a low-overhead, non-disruptive standard that requires minimal experience in security assurance methodologies and practices for easy adoption	Survey the best known methodologies, tools, standards, and knowledge bases for security assurance across the entire IP lifecycle process
2	Develop a standard that is comprehensive, flexible, and scalable in order to support the complete ecosystem (existing and new technology) and avoid customization	Identify what is considered to be a sufficient security review for IP in terms of threat modeling and verification
3	Develop a standard that can be used to auto-generate collateral that can be verified for completeness, accuracy, and quality of compliance to the standard	Identify a reporting framework with existing tools that can be extended to help with generation and verification

Table 1. IPSA Standard Goals and Methods

The IPSA standard defines a specification that addresses security concerns for hardware IP and its associated components when integrated into a Silicon product. With the standard, IP vendors will be able to identify security concerns to either 1) mitigate themselves, or 2) acknowledge for the SoC/ASIC owner to address at the integration level. The standard is also defined outside of the IP definition. This maximizes flexibility by removing dependencies on existing standards, provides scalability for future capabilities, and allows applicability to existing or legacy designs.

This paper includes some ubiquitous terms that may have different definitions outside of the paper. When referenced, therefore, these terms have the following definitions:

- **IP** – Intellectual Property – The RTL or other design representation that is the subject of this paper
- **RTL** – Register-transfer level – A design abstraction that models a digital circuit
- **Asset** – Anything of value or importance that is used, produced, or protected within the IP
- **Threat (Attack)** – Anything that can potentially adversely affect an asset
- **Concern (Consequence)** – The potential harm that a threat poses to an asset
- **Vulnerability** – A weakness in the IP that could be exploited
- **Attack Surface** – The set of access points to which threats can be applied

It is important to note that the goal of this paper is to solicit feedback from the greater industry that is outside of the Accellera IPSA Working Group. Attributes, their associated fields, and definitions described here may be changed or appended in the initial release of IPSA standard.

This paper is sectioned in the following manner:

- **Methodology** details the overall concept and workflow along with the individual components, dependencies, and assumptions
- **CIPSCE Knowledge Base** lists potential IP security concerns
- **OpenCores Examples** uses open source cores to demonstrate the methodology
- **Summary and Outlook** captures the next steps required for public release of the standard

Methodology

The IPSA standard introduces new components to add to an IP's delivery collateral: 1) Asset Definition, 2) Attack Surface, and 3) Threat model informed by the CIPSCE knowledge base. The additions are highlighted in the conceptual workflow shown in Figure 1.

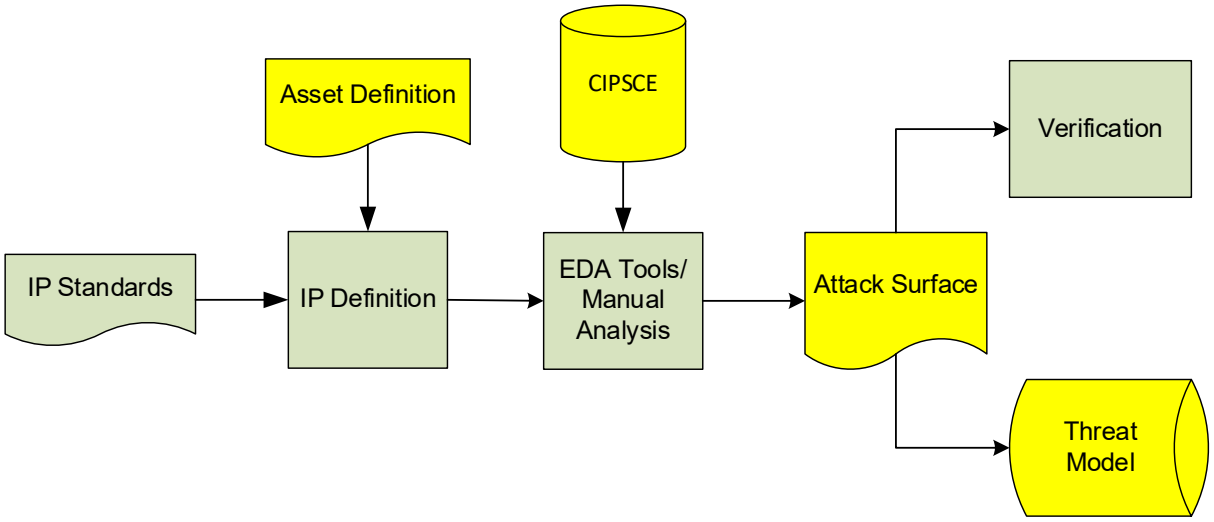

Figure 1. Conceptual Workflow

The workflow shows an addition (i.e., the asset definition defined by the IPSA standard) to the IP definition, which is formed today from existing standards such as SystemVerilog, Verilog, VHDL, SystemRDL, etc. The IPSA standard identifies assets within the IP and by using qualifying attributes

produces an attack surface over which to perform a security analysis. This security analysis process may be manual or may be augmented through the use of a generation tool (e.g., EDA tool). Similarly, security concerns to highlight for the Integrator may be extracted manually or using a generation tool. The CIPSCE database provides a standard nomenclature and cataloging of common security concerns to help simplify and classify the applicable weaknesses. The IPSA standard defines the relationship between the asset and CIPSCE database with an attribute association[1] as shown in Figure 2. In the following sections, the attributes that pertain to associations are in ***bold italic*** text.

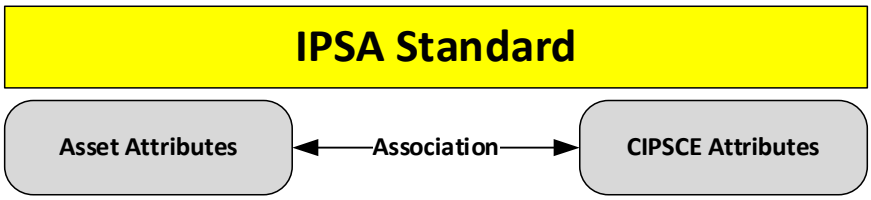

Figure 2. IPSA Attribute Association

Until the tools become available to help consume and digest the IPSA collateral, it is recommended to use a "divide and conquer" approach when applying the methodology to large or complex designs. The rationale is that for large and complex designs (e.g., processors, controllers, etc.), the collateral produced can be difficult for human comprehension. With "divide and conquer," complex IP can be broken down into basic blocks that are easier to analyze and verify. This approach aligns with other methodologies such as formal verification, functional modeling, etc. The use case example OR1200 (CPU) in the OpenCores Examples section shows how this approach is applied to a complex IP.

IPSA – Asset Attributes

An asset can be identified as a port, module, register, combination, and so on that is part of the design that the IP Developer deems important for the SoC/ASIC owner to consider during integration. Selecting the appropriate asset is a critical step in using the IPSA standard since this is a fundamental building block of the standard. The paper in [1] provides more information, along with examples, about how to identify assets within an IP. Once an asset is identified, its definition is comprised of the attributes defined in Table 2. Each asset will have its own asset attribute table. These attributes are to be provided by the IP Developer. The standard does not define which fields are inputs or outputs to be used by a tool for auto-generation.

Attribute	Required	Type	Definition
Name	Yes	Text (case-sensitive[2])	Name of the asset as defined in the RTL. If the asset is not a module, then the module name should be included to reduce naming collisions. Format: - module.asset
Instance Scope	No	Text (case-sensitive)	This should include every path where the asset has been instantiated. Format: - top_instance.instance_name

[1] The attribute association is inherently flawed since by definition the standard is meant to grow (e.g., as new threats are discovered). This means some associations will be missed or overlooked due to new values being added to the association. The IPSA Working Group is looking for feedback/solutions to help resolve this.
[2] Case-sensitivity may be dependent on the language of the RTL source.

Label	No	Text	A label that may be used as an identifier for a specified purpose (e.g., scripting, tool support, etc.). The value and format is user-defined.
Description	No	Text	Brief description about the asset (e.g., what makes it an asset, its purpose, etc.). This is not a required field, however it is strongly recommended since it provides useful information to the IP Integrator.
Family	***Yes***	***Enumeration***	***The family that describes the functionality of the IP (may be more than one). This field is used to help associate CIPSCE entries to the IP.*** ***1. Audio/Video*** ***2. Clock/Counter*** ***3. Communications*** ***4. Controllers*** ***5. DSP*** ***6. Memories*** ***7. Microcontroller*** ***8. Network-on-Chip*** ***9. Processors*** ***10. Security*** ***11. Subsystems*** ***12. Test/Debug*** ***13. Interface IP*** ***14. Bus IP*** ***15. Analog & Mixed-Signal IP*** ***16. Storage*** ***17. Other***
Functionality	No	Text	Additional information about the functionality of the IP that may be useful in identifying threats.
Security Objectives	Yes	List	Describes the security objectives required for the asset. Each objective may have a different attack surface (e.g., fan-in vs. fan-out). The definitions are defined in [6]. - Confidentiality - Integrity - Availability
Condition	No	SVA	SystemVerilog Assertion expression that, when true, defines one or more conditions that are required in order to support the “Security Objectives” attribute. An example may be a register setting or lock bit enabled in order to ensure integrity is met on an asset. The expression must include the module name in order to provide scope.
CIPSCE References	No	List	CIPSCE entries that are associated with the “Family” attribute. The format should follow the “Relationships” attribute detailed in Table 3.
Additional Security Concerns	No	CIPSCE	Used for security concerns that are not in the CIPSCE database. Use the CIPSCE format detailed in Table 3.
Attack Surface	Yes	Text (case-sensitive)	The attack surface should include the complete path of each access point to the asset. The access points should be part of the IP’s top module. Format: - top_module.port
Extension	No	User-defined	User-defined field for customization

Table 2. Asset Attributes

IPSA – CIPSCE Attributes

The CIPSCE knowledge base is an enumeration of security concerns or weaknesses that are associated with a particular IP family or functionality. The attributes of the entries are similar to [3] except with some modifications to address specific IP characteristics. The goal is to have Accellera host a public database that will support field queries and downloads for scalability. The content will be controlled by Accellera, but the greater community can contribute to the database by submitting entries to the IPSA Working Group for review and approval. This process is currently not defined; however the idea is to allow any contributor (i.e., commercial, researcher, government, etc.) to share knowledge in order to improve the security robustness of products. The attributes for each entry in the database are listed in Table 3.

Attribute	**Required**	**Type**	**Definition**
Reference Number	Yes	Alphanumeric	Reference number generated to uniquely identify the security concern in the repository. Format: YYYY.#
Title	Yes	Text	Title/Name of the security concern
Description	Yes	Text	Detailed overview of the security concern
Consequence	Yes	List	Level of risk and/or impact associated with the security concern. - Confidentiality - Integrity - Availability
Relationships	No	Text	Associations with other entries in the CIPSCE repository (Reference#, Title).
Applicability	***Yes***	***List***	***This attribute is associated with the "Family" field defined in Table 2. If the security concern applies to all IP families, then use "ALL".***
Modes of Introduction	Yes	List	Area(s) where the threat can be introduced into the product life cycle: - Architecture - Design - Implementation - Integration - Manufacturing - Provisioning
Examples	No	Text	Example(s) of how the threat can exist in the IP and/or SoC
Mitigations	No	List	Area(s) where a potential mitigation can be inserted: - Architecture - Design - Implementation - Integration - Manufacturing - Provisioning - Field or In-service updates

		Text	Description of the mitigation to minimize the risk of the security concern
Comments	No	Text	Used for additional and/or miscellaneous information associated with the security concern

Table 3. CIPSCE Attributes

IPSA – Conceptual Workflow

Figure 1 shows the conceptual workflow, however it lacks the details of how the standard can be applied. Below are the steps showing the roles of each stakeholder and the actions to be taken.

1. *EDA Vendor:*
   - Develop tool capabilities to parse the required information in Table 2 from RTL or companion source files to generate an attack surface and associate CIPSCE entries from the Accellera database based on IP family and/or functionality. The details of how a tool performs these capabilities are out of scope of the standard since it gets into the EDA vendor's specific implementation.

2. *IP Developer:*
   1. Identify the asset(s) in the IP design.
   2. Complete the attributes listed in Table 2 for each asset.
      a. Attributes "CIPSCE References" and "Attack Surface" may be left empty.
   3. Use an EDA tool or manual means to generate the attack surface and relevant CIPSCE associations. The resulting output is a threat model that incorporates the "CIPSCE References" and "Attack Surface" attributes for the asset.
   4. Add the completed Table 2 entry for each asset to the IP collateral to be delivered to the IP Integrator. This can be manual or part of the EDA tool processing.

3. *IP Integrator:*
   1. The Integrator has two options when receiving the IP from the vendor:
      1. Use the IPSA collateral "as is" to perform testing methods on the attack surface(s) and review which CIPSCE entries are in/out of scope for their product.
      2. Use an EDA tool to regenerate the attack surface(s) and CIPSCE associations for verifying the integrity of the IPSA collateral. Once verified, perform testing methods on the attack surface(s) and review which CIPSCE entries are in/out of scope for their product.
   2. Manually search through the CIPSCE knowledge base for entries that may be applicable that were not identified by the efforts of the IP Developer.

The workflow described implies that EDA tools provided by different vendors produce equivalent results. In practice, this may not be the case. Creation of a compliance test suite to verify that different tools produce comparable results is outside the scope of this standard.

CIPSCE Knowledge Base

At the time of this paper, content in the CIPSCE knowledge base does not exist. However, it is worth listing the topics the knowledge base may address. Table 4 shows a list of potential security concerns. This list is expected to be dynamic.

High-level Topic	Security Concerns
Debug	Bypassing of access control protections
	Escalation of privilege
	Alter functional behavior (e.g., flow control)
	Access to secrets or sensitive information
	Man-in-the-Middle (MITM) attacks
Test modes or pattern generators	Corruption of state
	Extraction of state
	Error injection attacks
Save/Restore state	Corruption of state
	Access to secrets or sensitive information
	Replay of state
	Porting of state to another system
	Denial of Service (DoS)
Locking controls	Incorrect order of locking
	Breaking locking dependencies
	Locking mechanism on a different power domain
Register exposure	Different interfaces with different access controls
	Duplicate registers with different access controls
	Duplicate registers with different values
Memory	Scraping
	DMA spoofing
	Aliasing or overlapping ranges
	Address translation collisions
	Execution of data
Shared resources	Caching/timing attacks
	Performance monitors (counters, timers, etc.)
Power	Power analysis (SPA/DPA)
	Voltage glitching
	Race condition attacks on protection mechanisms
	DoS
	Permanent damage
	Bypassing thermal protections
Compute	Untrusted execution (unsecure boot)
	Data execution
	Patching attacks (image rollback)
	Privilege escalation
	Control flow integrity
	Undocumented instructions/registers
Non-standard signals	Protection bypassing
	Permanent damage
	Option-out/disabling of protections
Non-volatile storage	Boot wear-out
	Bypassing read-only protections
Cryptography	Information leakage
	Key protections
	Weak security strength
	Standards compliance
All (Applies across all families)	Fault injection

Table 4. CIPSCE Topics

OpenCores Examples

To illustrate how the workflow and methodology apply to real IPs, two modules were selected from the OpenCores library: 1) Computer Operating Properly [4] and 2) OR1200, which is part of the Minisoc [5]. These blocks were chosen because of their contrast in complexity and functionality which demonstrates the agnostic applicability of the standard.

At the time of this paper, the CIPSCE database was just being created, so the entries associated with the assets are minimal. In a mature state there would be more entries associated. Regardless, the value remains and should not diminish the conceptual workflow.

Computer Operating Properly (COP)

This module is basically a watchdog timer that triggers a system reset if not regularly serviced. The block diagram is shown in Figure 3. One can easily argue that the "Watchdog Counter" block is critical to proper functionality since if it was to be controlled by an adversary that has put the system in a compromised state, a timeout (i.e., *cop_rst_o*) would never assert.

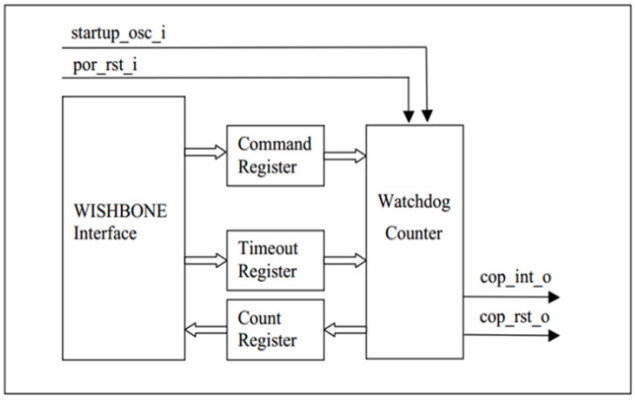

Figure 3. COP Block Diagram

Inside the counter block (cop_count.v), the register *cop_counter* holds the timeout value of the watchdog. This is what an adversary would want to control, thus making it an asset. The values for the IPSA asset table are shown in Table 5. It is also important to call out the expression in the "Condition" attribute. The COP has a write protect feature that prevents the counter from being disabled or reconfigured. Once the cwp bit is asserted, this ensures that the "Availability" security objective is upheld. This would be one of the conditions to verify during the integration phase of the IP. The COP has several conditions that should be accounted for in the asset table; for simplicity, only the write protect is highlighted. It is recommended that conditions that support different security objectives be separated into their own asset table.

Attribute	**Value**
Name	cop_count.cop_counter
Instance Scope	cop_top.counter
Label	COUNTER_IA_LABEL
Description	Modulo Counter value. Critical for proper operation.
Family	Clock/Counter, Test/Debug
Functionality	-
Security Objectives	Integrity, Availability
Condition	(cop_regs.cwp == 1)
CIPSCE References	-
Additional Security Concerns	-

Attack Surface	-

Table 5. COP Counter Asset Values

Once the asset table is complete, the IP Developer uses an EDA tool to produce the attack surface and CIPSCE entries associated to the IP family and functionality. This can be done manually if no such tool is available. For the COP, this is shown in Table 6.

Attribute	**Value**
Name	cop_count.cop_counter
Instance Scope	cop_top.counter
Label	COUNTER_IA_LABEL
Description	Modulo Counter value. Critical for proper operation.
Family	Clock/Counter, Test/Debug
Functionality	-
Security Objectives	Integrity, Availability
Condition	(cop_regs.cwp == 1)
CIPSCE References	2019.10, 2019.12
Additional Security Concerns	-
Attack Surface	cop_count.wb_clk_i cop_count.wb_rst_i cop_count.arst_i cop_count.wb_adr_i cop_count.wb_dat_i cop_count.wb_we_i cop_count.wb_stb_i cop_count.por_reset_i cop_count.startup_osc_i cop_count.stop_mode_i cop_count.wait_mode_i cop_count.debug_mode_i cop_count.scantestmode cop_count.cop_rst_o

Table 6. COP Counter Asset with Attack Surface Defined

Part of the attack surface is obvious to an Integrator (e.g., clock, reset, Wishbone bus), however what is interesting are the non-standard signals that can influence the counter (e.g., *stop_mode_i*, *debug_mode_i*, etc.). Furthermore, since availability is one of the security objectives, the *cop_rst_o* output should not be gated by an untrusted agent. These concerns should be called out as security concerns for the Integrator to consider. This can be done by manually adding to the "CIPSCE References" attribute if the EDA tool didn't make that association. Additionally, the IP Developer can use the "Additional Security Concerns" attribute to provide even more guidance.

The CIPSCE references listed in Table 6 are just examples because the database was not active at the time of the paper. However, to illustrate how such entries may look, Table 7 shows associated CIPSCE entries that would be applicable.

Attribute	**Value**
Reference Number	2019.10
Title	Corruption of state using test/debug modes
Description	Using a test/debug mode to change the state of the IP in order to corrupt an operation
Consequence	Integrity, Availability
Relationships	-
Applicability	Test/Debug, All

Modes of Introduction	Architecture, Design, Implementation, Manufacturing, Provisioning
Examples	Issuing a test/debug mode during a valid transaction in order to change that transaction into a different operation. An example may be error injection or changing the state of a Finite State Machine (FSM).
Mitigations	*Insertion*: Architecture, Design, Implementation, Manufacturing, Provisioning
	Description: Disable test/debug modes at runtime
	Insertion: Architecture, Design, Implementation, Manufacturing, Provisioning
	Description: Disable test/debug modes for critical operations
	Insertion: Integration
	Description: Add access protections so only trusted agents/components can invoke a test/debug mode
Comments	-
Attribute	**Value**
Reference Number	2019.12
Title	Defining non-standard signals outside of a standard interface
Description	Using non-standard signals can potentially bypass or undermine security protections defined within the standard interface. These signals may also be referred to as out of band or sideband signals.
Consequence	Confidentiality, Integrity, Availability
Relationships	
Applicability	All
Modes of Introduction	Implementation
Examples	Often non-standard signals are used by developers to simplify configuration and/or testing. However, these signals may bypass security protections or checks defined within a standard. For example, bypassing TrustZone protections on the ARM Peripheral Bus (APB).
Mitigations	*Insertion*: Implementation
	Description: Provide a disable strap for all non-standard signals
	Insertion: Implementation
	Description: Remove all non-standard signals before release of IP
	Insertion: Integration
	Description: Add access protection mechanisms on the non-standard signals
	Insertion: Integration
	Description: Tie off the non-standard signals to disable
Comments	-

Table 7. CIPSCEs for COP

OR1200 (CPU)

The Minisoc contains a RISC core named OpenRISC 1200 (OR1200). Below is a brief description about the core taken from the "OpenRISC 1200 IP Core" documentation.

> *The OR1200 is a 32-bit scalar RISC with Harvard microarchitecture, 5 stage integer pipeline, virtual memory support (MMU) and basic DSP capabilities. Default caches are 1-way direct-mapped 8KB data cache and 1-way direct-mapped 8KB instruction cache, each with 16-byte line size. Both caches are physically tagged. By default MMUs are implemented and they are constructed of 64-entry hash based 1-way direct-mapped data TLB and 64-entry hash based 1-way direct-mapped instruction TLB.*

There are many assets in this IP, however for purposes of this paper only one will be shown as an example. From the documentation, Figure 4 shows the block-level architecture for the core. Critical to the IP is performing proper data translations in the DMMU to ensure access protections are upheld.

This is implemented in the Verilog module *or1200_dmmu_tlb*. The concern would be if an adversary could influence or alter its behavior then maybe the adversary could extract sensitive data without the associated permission (i.e., privilege escalation). The point of this example is not to conduct a security review of the architecture or design, but to highlight how the IPSA standard can be used to identify potential vulnerabilities that can compromise an asset.

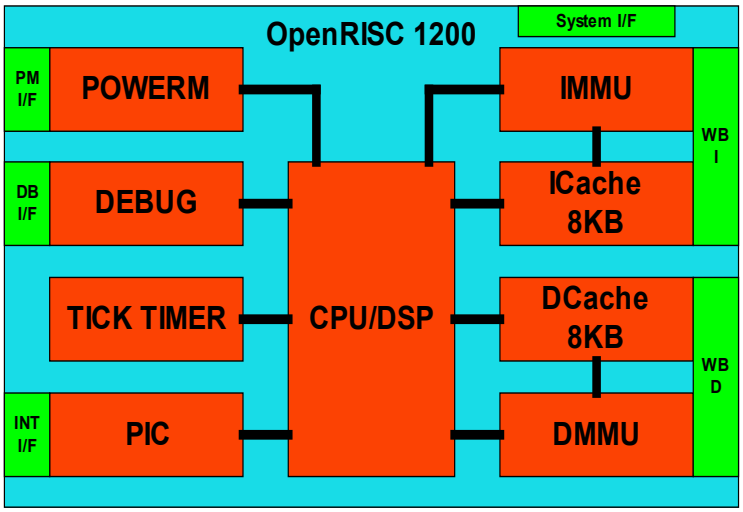

Figure 4. OR1200 Core Architecture

The asset table for the DMMU TLB is listed in Table 8. Once the asset table is complete, the IP Developer uses an EDA tool to produce the attack surface and the associated CIPSCE entries to the IP family and functionality. The attack surface for the DMMU TLB is shown in Table 9.

Attribute	**Value**
Name	or1200_dmmu_tlb
Instance Scope	or1200_dmmu_top
Label	DMMU_TLB_LABEL
Description	Memory Management Unit for Data Translation Lookaside Buffer. If translation is compromised, secrets in data could be exposed.
Family	Memories, Processors, Test/Debug
Functionality	Address translation
Security Objectives	Confidentiality, Integrity, Availability
Condition	-
CIPSCE References	-
Additional Security Concerns	-
Attack Surface	-

Table 8. OR1200 DMMU TLB Asset Values

Attribute	**Value**
Name	or1200_dmmu_tlb
Instance Scope	or1200_dmmu_top
Label	DMMU_TLB_LABEL
Description	Memory Management Unit for Data Translation Lookaside Buffer. If translation is compromised, secrets in data could be exposed.
Family	Memories, Processors, Test/Debug
Functionality	Address translation

Security Objectives	Confidentiality, Integrity, Availability
Condition	-
CIPSCE References	2019.12
Additional Security Concerns	-
Attack Surface	or1200_top.pic_ints_i or1200_top.clmode_i or1200_top.iwb_ack_i or1200_top.iwb_dat_i or1200_top.dwb_ack_i or1200_top.dwb_err_i or1200_top.dwb_dat_i or1200_top.dbg_stall_i or1200_top.dbg_stb_i or1200_top.dbg_we_i or1200_top.dbg_adr_i or1200_top.dbg_dat_i

Table 9. Complete Asset Values for DMMU TLB

Just like the COP example, the attack surface in the DMMU TLB asset table quickly shows there are debug signals (i.e., "dbg") that can influence the translation. Debug signals are always a concern when it pertains to security. In this case, since it was not specifically called out in the "Family" attribute, a tool probably would not associate such CIPSCE entries to the asset table. However, an Integrator would be able to easily recognize that these are debug signals and therefore manually scan the CIPSCE database for such associations.

Summary and Outlook

This paper has demonstrated how the emerging IPSA standard can be used to minimize risk when integrating IP into an SoC/ASIC. It has provided a methodology that identifies an attack surface to an asset within an IP and associates potential security concerns (i.e., CIPSCE). The methodology also provides the hooks for tools to auto-generate and auto-verify parts of the collateral.

There are still some areas that need more attention before a complete standard can be released. One of these is that the data format of IPSA attributes needs to be defined. The data format needs to be both human- and machine-readable. Currently the IPSA Working Group is favoring an XML schema, however it has not determined how that schema would look. Since there are not many attributes defined in the IPSA standard, a bare minimum XML schema would probably be sufficient. Anything more may be considered overhead and add unnecessary complexity. Another area that needs more focus is the CIPSCE database. How this knowledge base will be presented, queried, and maintained is still under discussion. In addition, there are security concerns that need to be taken into consideration when supporting a database with public access. The working group is currently working with Accellera to see what resources are required to replicate something similar to [3].

References

[1] Sherman, B., Borza, M., Rosenberg, B. et al. J Hardw Syst Secur (2017) 1: 38. https://doi.org/10.1007/s41635-017-0002-5

[2] Severe Security Advisory on AMD Processors, *https://safefirmware.com/amdflaws_whitepaper.pdf*

[3] Common Weakness Enumeration, CWE List Version 3.3, *https://cwe.mitre.org/data/index.html*

[4] OpenCores. COP. Retrieved from http://opencores.org/websvn,listing?repname=cop&path=%2Fcop%2F&rev=0

[5] OpenCores. Minisoc. Retrieved from https://opencores.org/projects/minsoc

[6] FIPS 199: Standards for Security Categorization of Federal Information and Information Systems, NIST, 2004, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.199.pdf

