IEEE SA
STANDARDS ASSOCIATION

IEEE SA WHITE PAPER

IEEE P3164: SECURITY ANNOTATION FOR ELECTRONIC DESIGN INTEGRATION

ASSET IDENTIFICATION FOR ELECTRONIC DESIGN IP

Authored by

IEEE P3164 Security Annotation for Electronic Design Integration (SA-EDI) Working Group

IEEE

TRADEMARKS AND DISCLAIMERS

IEEE believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. IEEE is not responsible for any inadvertent errors.

The ideas and proposals in this specification are the respective author's views and do not represent the views of the affiliated organization.

ACKNOWLEDGMENTS

Special thanks are given to the following authors of this paper:

Brent Sherman, Chair, *Intel*
Kamran Haqqani, Vice Chair, *Accenture*
Adam Sherer, Secretary, *Cadence*

Sohrab Aftabjahani, *Intel*
Jessy Ayala, *UC Irvine*
Debojyoti Bhattacharya, *Arm*
Mike Borza, *Synopsys*
Akim Layachi Daineche, *Arm*
Trenton Grale, *Accenture*
Miltos Grammatikakis, *Hellenic Mediterranean University*
John Hallman, *Siemens*
Kathy Hayashi, *Qualcomm*
Pavani Jella, *Silicon Assurance*
Wayne Kohler, *Perforce Software*
Amit Kumar, *Microsoft*
Yann Le Floch, Self
Wenzhen Li, Self
Jean-Philippe Martin, *Intel*
Mehdi Mohtashemi, *Intel*
Jean-François Mousinho, *Synopsys*
Prasad Nandipati, *Qualcomm*
Anders Nordstrom, *Cycuity Inc.*
James Pangburn, *Cadence Design Systems*
Rafael Santos, *Arm*
Benjamin Tan, *University of Calgary*
Vivek Vedula, *Arm*

The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA

PDF: STDVA26891 979-8-8557-0661-1

NOTICE AND DISCLAIMER OF LIABILITY CONCERNING THE USE OF IEEE SA DOCUMENTS

This IEEE Standards Association ("IEEE SA") publication ("Work") is not a consensus standard document. Specifically, this document is NOT AN IEEE STANDARD. Information contained in this Work has been created by, or obtained from, sources believed to be reliable, and reviewed by members of the activity that produced this Work. IEEE and the IEEE P3164 expressly disclaim all warranties (express, implied, and statutory) related to this Work, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; quality, accuracy, effectiveness, currency, or completeness of the Work or content within the Work. In addition, IEEE and the IEEE P3164 disclaim any and all conditions relating to: results; and workmanlike effort. This document is supplied "AS IS" and "WITH ALL FAULTS."

Although the IEEE P3164 members who have created this Work believe that the information and guidance given in this Work serve as an enhancement to users, all persons must rely upon their own skill and judgment when making use of it. IN NO EVENT SHALL IEEE SA OR IEEE P3164 MEMBERS BE LIABLE FOR ANY ERRORS OR OMISSIONS OR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS WORK, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Further, information contained in this Work may be protected by intellectual property rights held by third parties or organizations, and the use of this information may require the user to negotiate with any such rights holders in order to legally acquire the rights to do so, and such rights holders may refuse to grant such rights. Attention is also called to the possibility that implementation of any or all of this Work may require use of subject matter covered by patent rights. By publication of this Work, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying patent rights for which a license may be required, or for conducting inquiries into the legal validity or scope of patent claims. Users are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. No commitment to grant licenses under patent rights on a reasonable or non-discriminatory basis has been sought or received from any rights holder.

This Work is published with the understanding that IEEE and the IEEE P3164 members are supplying information through this Work, not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought. IEEE is not responsible for the statements and opinions advanced in this Work.

TABLE OF CONTENTS

ASSET IDENTIFICATION FOR ELECTRONIC DESIGN IP

ABSTRACT

The Accellera Security Annotation for Electronic Design Integration (SA-EDI) standard [1][1] provides a framework for producing security assurance collateral for an IP. The root of the standard and its workflow is the identification of assets for a given IP. Once the assets are identified, the corresponding threats and attack surfaces can be determined to help an Integrator address risks in their integrated circuits (ICs). Therefore, if an asset was mistakenly identified, either a false positive or a false negative, the rest of the collateral would become invalid. Unfortunately, the standard provides little guidance on identifying assets and avoiding false positives/negatives. This paper proposes two methodologies for asset identification within an IP using conceptual and structural analysis (CSA) and points of influence and observation (PIO).

1. INTRODUCTION

The Security Annotation for Electronic Design Integration (SA-EDI) standard builds on the identification of assets within the IP. The Element and Attack Point Security Objective (APSO) objects are built using information from the Asset Definition objects. If the information in the Asset Definition objects is not accurate, then the collateral produced from it becomes inaccurate, thus leading to potential exploits in the integrator's integrated circuit (IC). Therefore, assets must be accurately identified and classified accordingly. Unfortunately, SA-EDI does not provide enough guidance in this domain to reduce false positives and false negatives. In addition, contextual IC information such as security requirements, use cases, and so on are not known to the IP developer, making the creation of SA-EDI collateral more difficult. This contextual information is only known well after the IP has been developed and delivered to the integrator.

[1] Numbers in brackets correspond to sources listed in Section 6.

This document proposes a new methodology and classification guidance to help identify assets in an IP. It walks through the methodology using several example IPs that identify and associate high-level assets to security objectives, resulting in the definition of structural assets in the Register-Transfer Level (RTL). These IPs are listed as follows:

- **Simple GPIO Pad**: Single GPIO with Direction Select.
- **Gaussian Noise Generator (GNG)** [2]: A GNG generates random noise that has a standard normal distribution and constant power spectral density. The core uses a 64-bit combined Tausworthe generator (LFSR and XOR) and further processes to transform random noise distribution to white Gaussian noise (GN) [with blocks split, mask, leading zero detector (LZD), swizzle, and address]. A polynomial approximation with specific coefficients is used. The generator also relies on an initial seed.
- **Advanced Encryption Standard (AES) Engine**: A high-level AES engine with separated data in/out paths. Configuration registers set the mode, encrypt/decrypt operation, key size, and start. Read-only status registers report errors, while the data out signal provides encrypted/decrypted output when it is available.
- **SRAM Controller** [5]: A simple static random-access memory (SRAM) controller that supports a sleep mode and memory built-in self-test (MBIST).
- **Generic CPU Core**: A basic core pipeline and execution unit with instruction and data caches.

Section 2 details the SA-EDI standard workflow at a higher level, examining how this methodology can be used to create security assurance collateral, mainly focusing on the Asset Definition objects. Section 3 details a new methodology called *conceptual and structural analysis* (CSA) to identify IP assets to create the Asset Definition objects. It also demonstrates the approach by applying it to several IP examples.

NOTE—Only the GNG and SRAM examples reference the actual code since the RTL is available in their respective repository.

Section 4 introduces another method specifically for complex IPs or IPs with multiple shared resources. This method is called *points of influence and observation* (PIO) and can be used in addition to CSA or as an alternative. Section 5 summarizes both approaches and use cases.

Lastly, this standalone document is intended to be used as supplemental guidance to v1.0 of the Accellera SA-EDI standard. Future versions of the standard may or may not impact the content of this document.

2. GENERAL

The SA-EDI standard focuses on creating security assurance collateral for proper integration of an IP. The standard defines four objects, as a progression, which are included in the IP bundle the IP provider creates. These objects and their associations are shown in Figure 5 of the SA-EDI standard [1]. At the root of the object creations are the Database and Asset Definition objects. These objects are the starting points for creating the collateral.

The Database object points to a security weakness knowledge base (e.g., MITRE CWE [6]), as defined in SA-EDI, which contains a database of known weaknesses. This object is optional, however, highly recommended, especially when using an industry-available database, such as CWE. The Asset Definition object points to assets within the IP. These assets may be a module, register, buffer, array, or any material defined by the RTL. Both objects, along with the RTL, are used to produce the Element objects as shown in Figure 3 of SA-EDI [1]. The Element object is used to associate IP top-level ports (i.e., inputs and outputs) to an asset. These ports are the attack surface for a threat to a security objective an asset may have. For example, a confidentiality security objective may be violated by a port if it can be used to observe an asset at the integration level of the IP. This information (i.e., assets, ports, security objectives, threats, etc.) is captured in Attack Surface Security Objective (APSO) objects, which is shown in Figure 3 of SA-EDI [1]. The APSO objects are the end results that are used by the integrator to help generate the threat model of the IC, as shown in Figure 4 of SA-EDI [1].

The Asset Definition objects form the foundation for creating the APSO objects. If they are not properly constructed, for example, if they are questionable in validity, then the result will be invalid APSO objects. Determining the validity of an Asset Definition object is subjective, meaning that what one developer deems as an asset may or may not be what another developer considers as an asset. This becomes the root of the problem; how can one objectively identify, with justification, what an IP asset is? This paper proposes a dual-approach to help increase the confidence that an asset has been properly identified, thus resulting in valid APSO objects for the integrator to consume. Both methodologies require the use of an architectural block diagram. As a disclaimer, this document does not eliminate subjectivity. It merely provides two structured approaches to help the IP provider apply a security mindset.

3. ASSET IDENTIFICATION

Identifying assets and their properties is a way to help discover risks associated with IP integration. Many use this approach as the first step in threat modeling, especially when dealing with a large interface or attack surface.

As with most approaches, the first step is often the critical piece since the rest of the process builds from it. Therefore, any inaccuracies, such as false positives or negatives, may produce erroneous results. In security, this can lead to exploitation. As an example, if the threat modeling process missed an asset (i.e., a false negative), then a threat could remain unidentified resulting in a missing mitigation. To help address this concern, the following two approaches are proposed: CSA and PIO.

The CSA approach focuses on security objectives to identify conceptual assets. These conceptual assets are used to identify their RTL representation, such as registers, modules, gates, and so on, which are the structural assets. These structural assets create the Asset Definition objects in the SA-EDI standard. The PIO method starts by looking at the interface of the IP to identify conceptual assets. Once identified, the focus becomes points of influence and observation on these assets. These are points through which an asset may be compromised. If a security objective can be compromised, then the RTL representation of these points would be a structural asset and thus require an Asset Definition object.

3.1. CONCEPTUAL AND STRUCTURAL ANALYSIS

The concept behind CSA is to use conceptual assets to identify structural assets that will be used to define the Asset Definition objects of the SA-EDI standard. A conceptual asset is a high-level asset associated with the use-case flows of the IP, such as data and system state, which involves a security objective such as confidentiality, integrity, or availability (CIA). An example of a conceptual asset is data that a user wants to identify as a secret. This data would, at least, require confidentiality as a security objective, thus making it an asset.

A structural asset is RTL material that physically supports a conceptual asset. Examples of structural assets may be registers, modules, flops/latches, gates, and so on. Using the plaintext data as the conceptual asset, a structural asset would be RTL material that supports its data flow. An example may be a buffer that temporarily stores the data as it is entered into the IP. Another example may be registers that contain details about this data, which should also be kept secret. These examples may be considered structural assets, thus requiring SA-EDI Asset Definition objects.

It is worth noting that the proposed methodology does not claim to eliminate ambiguity or subjectivity when determining what is an asset. Instead, the claim is that CSA is one of many approaches that an IP developer can use to help generate SA-EDI collateral and that the approach can be used independently or in addition to other methodologies. The recommendation is to use whatever approaches yield the best results for a particular IP.

3.1.1. CONCEPTUAL ANALYSIS

It is uncommon for IC security objectives to be known to an IP developer, simply because an IP is typically developed well before an integrator establishes requirements and use cases. To overcome this obstacle, IP developers must make certain assumptions to assess security risks in the IP. The conceptual approach is designed to assist in this process by focusing on asset identification and security objective associations. The approach assumes that the IP developer has zero contextual knowledge about the IC.

To start the methodology, the IP developer answers the following questions for each intended use case of the IP. An architectural diagram of the IP is helpful during this assessment.

1. Assume that the IP is to be integrated into an IC where confidentiality protections are required. Are there any elements in the IP that can leak or expose material that an integrator may deem confidential?
    - For example, is there any information, either as input or internally generated, that may be considered secret?
2. Assume that the IP is to be integrated into an IC where integrity protections are required. Are there any elements in the IP that can modify material an integrator may deem as sensitive?
    - For example, are there any state or configuration settings that need to be immutable during certain operations or modes?
3. Are there any elements in the IP that, if unavailable, would prohibit the operational behavior of the IP or IC?
    - For example, are there any elements that could gate an output port or the use of an input port? The focus should be on elements that may be impacted by a denial-of-service attack at the integration level.
4. Are there elements that could be impacted by behaviors at the integration level to undermine the functionality of the IP under normal operation?
    - For example, are there any privileged modes, overrides, bypass, test packet injection, and so on that can make the IP produce incorrect output? The focus should be on elements that may be compromised.

If the answer is “Yes” to any of the questions above, then the element in question may be considered a conceptual asset in the IP. The next step would be to move to the structural approach of the methodology. If

the answer is “No” to all the questions above, then it is probably safe to assume that the element is not an asset and does not require any associated SA-EDI objects.

3.1.2. STRUCTURAL ANALYSIS

Once the conceptual assets have been identified, the next step is to identify structurally where these assets are in the design. The intent is to look through the RTL and identify material that supports the conceptual assets. For example, during the conceptual phase, it was discovered that one of the use cases required confidentiality on “Data” being generated within the IP. The structural assets would be the RTL code that produces “Data” and stores and transports its value (e.g., reg and wire). These code parts are the structural assets and should have an associated SA-EDI Asset Definition object.

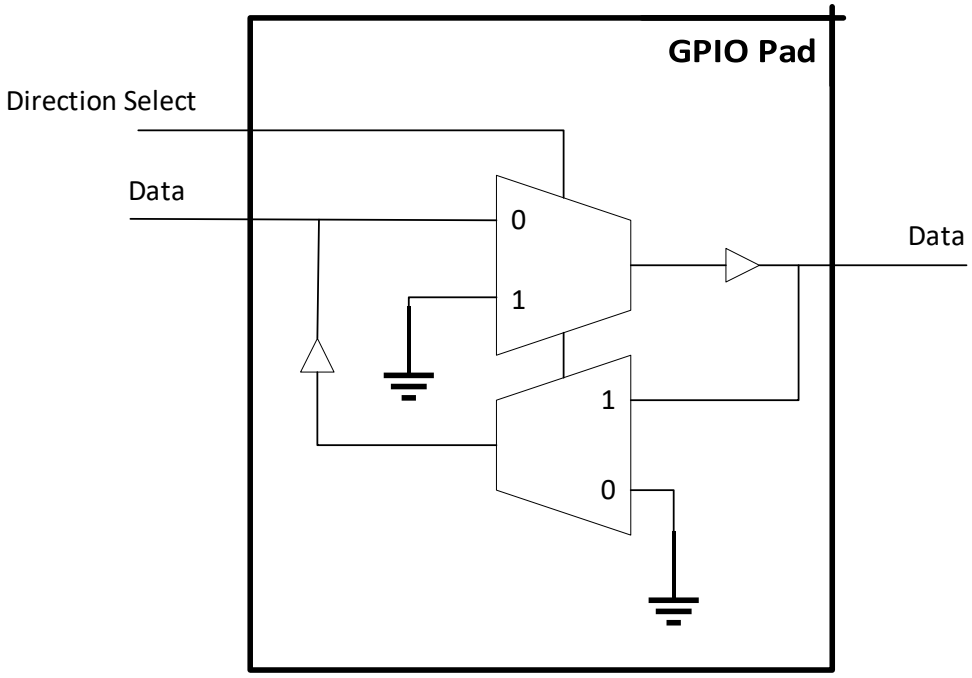
3.2. EXAMPLES

This section applies the CSA methodology to several IP examples by analyzing the architectural diagrams. Since the RTL is not provided for some of the IPs, the structural portion will be discussed without reference to the code. However, this should not take away from the learnings highlighted by the examples.

3.2.1. SIMPLE GPIO PAD

The architectural block diagram of the simple GPIO pad is shown in Figure 1. Two data ports and a Direction Select are the only interface ports. This IP was selected because of the minimal use cases and integration options that are supported. The intent is to start with a simple, noncomplex IP, so the methodology becomes the focus and not the IP itself. In later examples, the architecture becomes more complex.

FIGURE 1 Simple GPIO pad block diagram



Start by applying the conceptual phase of the analysis by answering the questions stated in Section 3.1.1.

1. ***Confidentiality****:* Are there any elements in the IP that can leak or expose material that may need confidentiality?
   - No. The gates inside the IP do not leak any information.
2. ***Integrity****:* Are there any elements in the IP that can modify material an integrator may deem as sensitive?
   - Yes. If the "Direction Select" was toggled during a runtime sample, the "Data" could also toggle in value, potentially producing an error. The mux gates can be considered conceptual assets.
3. ***Availability****:* Are there any elements in the IP that, if unavailable, can prohibit operational behavior?
   - Yes. The "Direction Select" can reverse the data flow on "Data" ports, which can be a denial of service. The elements impacted by this attack are the mux gates and should be considered conceptual assets.
4. ***Undermined Expected Behavior****:* Are there elements that could be impacted by behaviors at the integration level to undermine the functionality of the IP under normal operation?
   - No. The IP has no privilege or bypassing mechanisms that will alter its normal behavior.

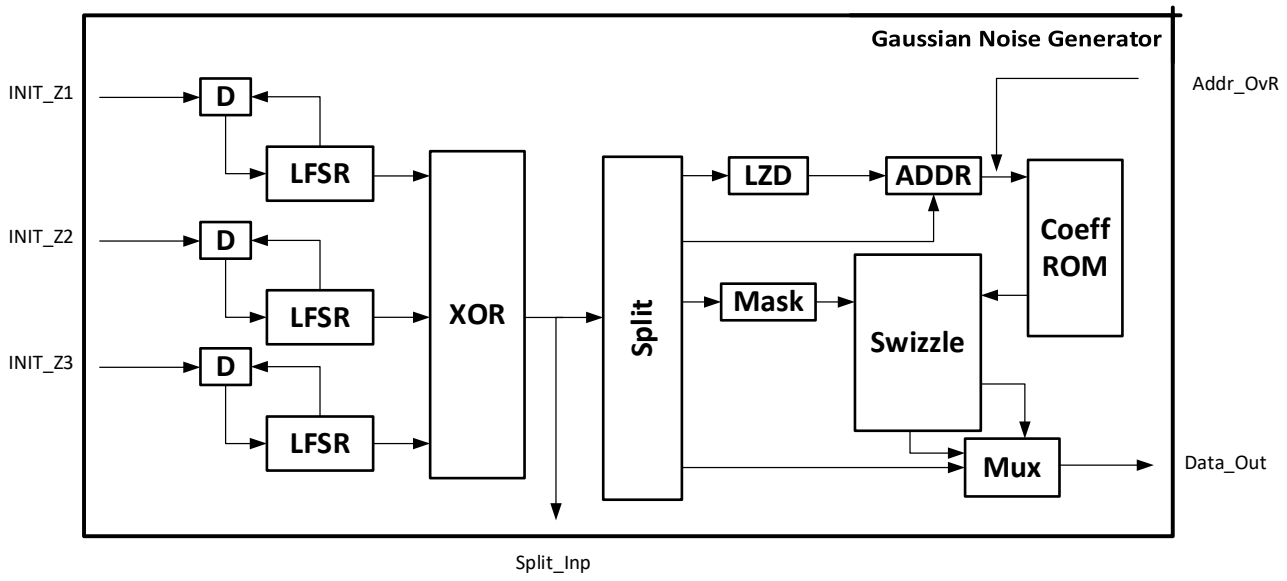
Since the assessment triggered questions #2 and #3, there should be at least one conceptual asset: the mux gates. The structural assets that create the Asset Definition objects would be the RTL of these gates. In addition, the "Direction Select" port can be used as the attack point to violate both objectives; however, attack points associated with SA-EDI objects are not the focus of this paper.

3.2.2. GAUSSIAN NOISE GENERATOR

The GNG is an IP that is available in OpenCores [3]. It can be used in deep-learning models to add randomness to the input data or weights to make the neural network more robust to data variations. It is frequently used in image detection and speech recognition.

Figure 2 is a modified GNG block diagram from the one in OpenCores. Inputs "INIT_Z" are ports instead of parameters and ports "Addr_OvR" and "Split_Inp" were added to challenge the methodology and add to the analysis. Another modification is the opaque "Swizzle" subsystem that simplifies the block diagram without impacting the assessment.

FIGURE 2 GNG block diagram



The IP has three data input ports labeled "INIT_Z" that are initialization vectors to prime the linear feedback shift registers (LFSRs). The "Split_Inp" is an output test port that is used to observe the randomness going into the split logic. The "Addr_OvR" port is used to override the address created by the LZD logic for testing the coefficient ROM. The "Data_Out" is the GN produced by the IP.

Start by applying the conceptual phase of the analysis by answering the questions stated in Section 3.1.1.

1. ***Confidentiality****:* Are there any elements in the IP that can leak or expose material that may need confidentiality?
    - Yes. The output value of the XOR block can be deemed as a seed and, if observed, may be used to predict the GN. This assumes that the IC considers GN as a secret. The conceptual asset would be the XOR and LFSR blocks.
2. ***Integrity****:* Are there any elements in the IP that can modify material an integrator may deem as sensitive?
    - Yes. The address into the Coeff ROM should not be modified once the "INIT_Z" inputs are set. Modifying the address to use a different coefficient than the one intended may reduce the randomness of the GN. Therefore, the conceptual assets would be the ADDR block and Coeff ROM.
3. ***Availability****:* Are there any elements in the IP that can become unavailable, prohibiting operational behavior?
    - No. There is no means at the integration level to disable or impede "Data_Out."
4. ***Undermined Expected Behavior****:* Are there elements that could be impacted by behaviors at the integration level to undermine the functionality of the IP under normal operation?
    - Yes. Input "Addr_OvR" can force the IP to select an unintended coefficient, which may produce an invalid GN on "Data_Out," pending the use case. Therefore, Coeff ROM is a conceptual asset.

All questions identified a conceptual asset except for #3. The assets identified are XOR, LFSR, ADDR, Coeff ROM, and the RTL that constructs these blocks will be the structural assets and, therefore, require an associated Asset Definition object. For example, the output value of Coeff ROM is located in the file gng_coef.v in line 51 shown in Figure 3.

FIGURE 3 Coeff ROM output

```
50    // Local variables
51    reg [52:0] d;    // {c0, c1, c2}
```

Therefore, the Asset Definition object for this asset may be defined as

```
{
  "Name" : "gng.gng_interp.gng_coef.d",
  "Description" : "Output from Coeff ROM",
  "Family" : ["Accelerator"],
  "Type" : ["Sensitive"],
  "Database_ID" : ["CWE VIEW: Hardware Design"]
}
```

3.2.3. AES ENGINE

The AES is a symmetric block cipher that is approved by NIST [4]. FIGURE 4 shows an architectural block diagram of an AES IP. The signal descriptions are detailed in Table 1.

FIGURE 4 AES engine

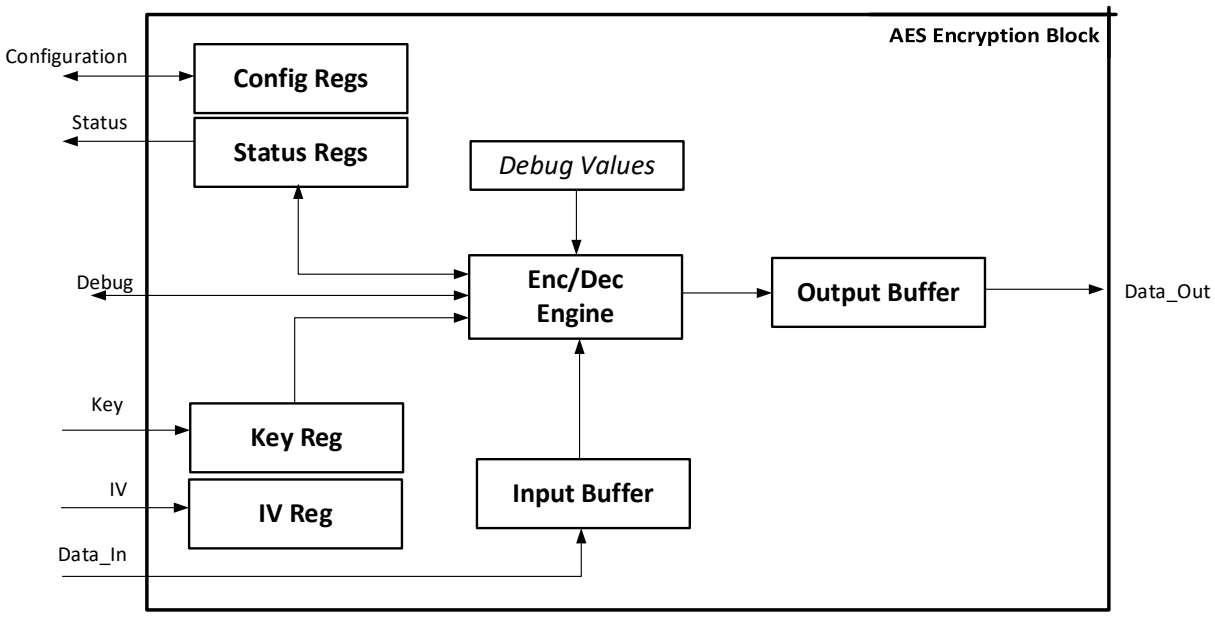


TABLE 1 AES signal descriptions

Name	Type	Description
Configuration	Read/Write	Access to the configuration registers for the Enc/Dec Engine such as key size, AES mode, operation, start/stop, and so on
Status	Read	Access to the status registers for error codes, state, completion, and so on
Debug	Read/Write	Signals used to debug the Enc/Dec Engine, including complete observability and control. When entering the debug mode, the IV and key values are replaced with debug values hardcoded in the RTL
Key	Write	Key value for encrypt/decrypt operation
IV	Write	Initialization vector for AES cryptography
Data_In	Write	Input data to be encrypted/decrypted
Data_Out	Read	Output data from the encrypt/decrypt engine

Start by applying the conceptual phase of the analysis by answering the questions stated in Section 3.1.1.

1. ***Confidentiality****:* Are there any elements in the IP that can leak or expose material that may need confidentiality?
    - Yes. Since this is a crypto IP, the plaintext data and key values are secrets. Therefore, any block in Figure 3 that supports these secrets will be a conceptual asset. These assets are Key Reg, Enc/Dec Engine, Input Buffer, and Output Buffer. In addition, the Status Regs may leak confidential information since it provides information about the Enc/Dec Engine. Therefore, this block may also be considered a conceptual asset.
2. ***Integrity****:* Are there any elements in the IP that can modify material an integrator may deem as sensitive?
    - Yes. When the Enc/Dec Engine is operating, the key, IV, input data, and its configuration should not be modified. Therefore, Key Reg, IV Reg, Input Buffer, and Config Regs are conceptual assets that require integrity.
3. ***Availability****:* Are there any elements in the IP that can become unavailable, prohibiting operational behavior?
    - Yes. The debug interface allows complete control of the Enc/Dec Engine. Therefore, “Data_Out”

can be blocked by this interface, thus making the Enc/Dec Engine a conceptual asset.

4. ***Undermined Expected Behavior****:* Are there elements that could be impacted by behaviors at the integration level to undermine the functionality of the IP under normal operation?

    - Yes. The debug interface allows the IP to encrypt/decrypt using the test key and IV values, which may result in a loss of security strength, making the Enc/Dec Engine a conceptual asset.

As identified in the questions above, every block in the IP, except the Debug Values block, can be considered a conceptual asset. Therefore, the RTL in these blocks would be the structural assets and require Asset Definition objects, which may be too numerous to comprehend. This is common for IPs that make security claims such as cryptography. One could assert that the entire IP is a structural asset, whereas the top RTL module would be the Asset Definition object. This would reduce the Asset Definition objects to just one, which simplifies the analysis. Another approach, which is a modification to the CSA methodology and is detailed in Section 4, may be used to analyze the IP from a vulnerability perspective to identify assets. This approach could help identify assets that are false positives. Both approaches are acceptable and should yield the same APSO [1] objects.

3.2.4. SRAM CONTROLLER

The SRAM controller is an IP that is available in OpenCores [3]. The modified architectural block diagram of an SRAM controller is shown in FIGURE 5. Modifications were made to simplify the architecture to highlight the CSA methodology. All inputs are both read/write and their descriptions are listed in Table 2.

FIGURE 5 SRAM controller

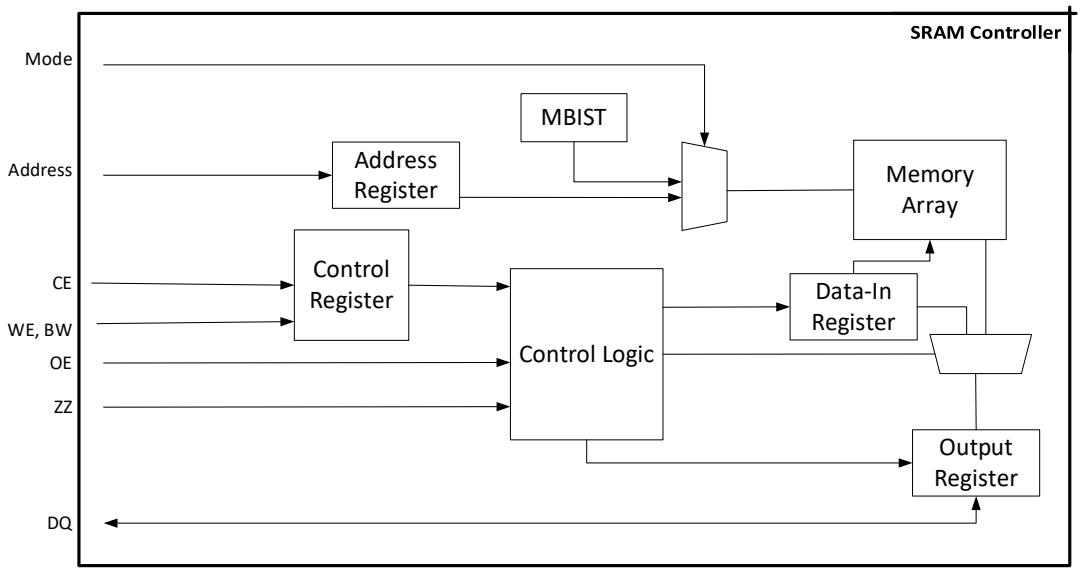


TABLE 2 SRAM controller signal descriptions

Name	Description
Mode	Used to select MBIST. This operation overwrites contents in the Memory Array and prevents the address signal from being utilized
Address	Address in the memory array to read from or write to
CE	Chip enable
WE	Write enable
BW	Synchronous byte write
OE	Output enable
ZZ	Power sleep mode
DQ	Data (input/output)

Start by applying the conceptual phase of the analysis by answering the questions stated in Section 3.1.1.

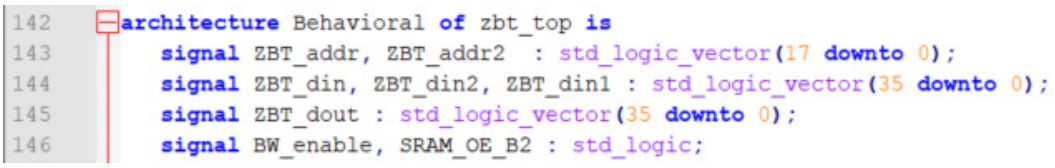
1. ***Confidentiality****:* Are there any elements in the IP that can leak or expose material that may need confidentiality?
    - Yes. If secret data is stored in the SRAM, then the Memory Array becomes an asset. In addition, the Data-In Register and Output Register may also contain secret information that is readable at the integration level. The address register might contain information that may be considered a secret, pending on the use case. Therefore, the conceptual assets are the Memory Array, Data-In Register, Output Register, and Address Register.
2. ***Integrity****:* Are there any elements in the IP that can modify material an integrator may deem as sensitive?
    - Yes. The integrator may want a certain address range to be read-only. Therefore, this range will need integrity protections, thus making the Memory Array a conceptual asset.
3. ***Availability****:* Are there any elements in the IP that can become unavailable, prohibiting operational behavior?
    - Yes. If MBIST is enabled, the address signal is no longer input into the Memory Array, thus preventing operational behavior. If the "ZZ" signal is asserted, the IP goes into sleep mode and prevents it from operating. Therefore, the Memory Array is a conceptual asset.

4. ***Undermined Expected Behavior****:* Are there elements that could be impacted by behaviors at the integration level to undermine the functionality of the IP under normal operation?

    - Yes. The MBIST operation makes the IP unusable while it is executing test patterns. Therefore, the Memory Array is a conceptual asset.

The SRAM controller triggered all four questions, resulting in the Memory Array, Data-In Register, Output Register, and Address Register as conceptual assets. The RTL that supports these blocks is considered a structural asset and should be associated with an Asset Definition object. For example, the address value in the Address Register is in file zbt_top.vhd in line 143 as shown in Figure 6.

FIGURE 6 SRAM address



The signals ZBT_addr and ZBT_addr2 represent the SRAM address and are considered structural assets. As for the Asset Definition object, the assets could be combined into a single object or left separated. In this case, the decision was to leave them separated and create two objects, so it is explicit.

```
{
  "Name" : "zbt_top.ZBT_addr",
  "Description" : "SRAM address that requires confidentiality protections",
  "Family" : ["Memories"],
  "Type" : ["Secret, Sensitive"],
  "Database_ID" : ["CWE VIEW: Hardware Design"]
}

{
  "Name" : "zbt_top.ZBT_addr2",
  "Description" : "SRAM address that requires confidentiality protections",
  "Family" : ["Memories"],
  "Type" : ["Secret, Sensitive"],
  "Database_ID" : ["CWE VIEW: Hardware Design"]
}
```

4. POINTS OF INFLUENCE AND OBSERVATION

The CSA methodology may get difficult to scale for complex IPs, such as a CPU core or an IP subsystem. An analysis may end up identifying all the internal blocks of an IP as structural assets which could result in false positives and/or too many Asset Definition objects for human comprehension. An alternative approach would be to level up the architecture block diagram to just focus on the interface of the IP. This should help reduce the complexity of the analysis. Next is to identify the information that goes into the IP and what information is produced as a result. This information may be the conceptual assets. Once the conceptual assets are identified, revert to the architectural block diagram and focus on the points of influence and observation of the IP, that is, blocks where the conceptual assets can be observed or influenced. Analyze the use cases at this observation point to identify structural assets in the RTL by answering the following questions, which align with the questions highlighted in Section 3.1.1:

1. Does the observation point expose any confidentiality of the conceptual asset?
2. Does the influence point allow any modification of the conceptual asset?
3. Can the observation and/or influence point prevent the conceptual asset from being available for functional operation?
4. Does the observation and/or influence point have any special behaviors that can prevent the conceptual asset from being available for normal operation?

The structural assets identified will be associated with an Asset Definition object in the SA-EDI standard. The following section will highlight this approach using a generic CPU core as an example.

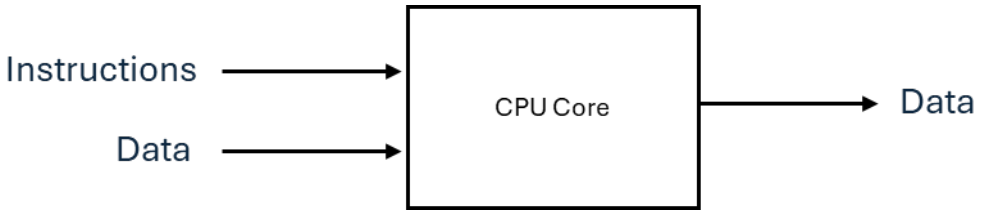
4.1. GENERIC CPU CORE

4.1.1. GENERAL

A CPU core is a complex IP consisting of a pipeline, shared structures, branch prediction (BP) capabilities, and so on. Due to this complexity, applying the CSA methodology to such an architectural diagram would not be as straightforward as shown in the previous examples. Therefore, the PIO methodology may be better applied to determine the structural assets.

The first step of the PIO methodology is to simplify the CPU core architecture to just the input and output signals, which is shown in Figure 7.

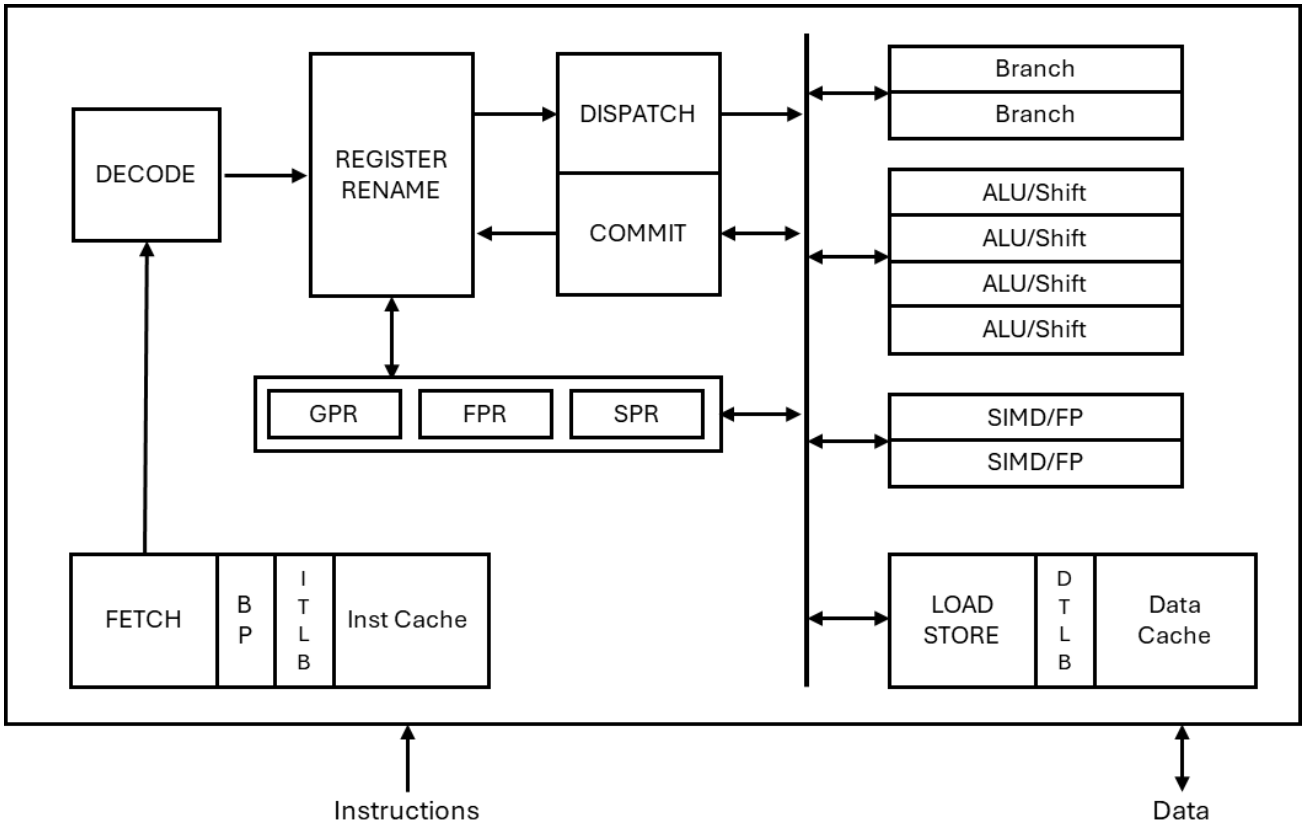
FIGURE 7 Simplified generic CPU core



A typical CPU core will execute instructions that act upon input data to produce some output data. By focusing on this use case, Instructions and Data can be identified as the conceptual assets. Next, use these assets to identify structural assets in a more detailed architectural diagram.

Figure 8 presents a more detailed block diagram showing the most important blocks within a generic CPU core. Note that the Data input and Data output from Figure 7 are shown as the same port in Figure 8.

FIGURE 8 Block diagram of a generic CPU IP



Since this is a generic CPU core, many of the common structures, such as power and system management, debug, and performance buses, have been abstracted away, so the focus is on the instructions and data as the main conceptual assets.

Instructions are stored in the instruction cache (ICache) and fetched according to the address stored in the program counter (PC), which thereafter increments to point to the next instruction to be fetched. The actual memory address is obtained once an address translation is performed by the instruction translation lookaside buffer (ITLB), which maintains information and permissions about the context of each process executed. If the instruction fetched is a control flow instruction (e.g., conditional branch, indirect jump, call/return, etc.), the next address to be fetched is not necessarily the next in memory. The BP stores the history of past executed control flow instructions to allow for predicting the value of the next PC in these cases.

Once fetched, an instruction will propagate to the Decode, Register Rename, and Dispatch units to finally be assigned to one of the execution units available (e.g., branch, ALU/shift, and SIMD/FP).

Data is stored in the data cache (DCache) and accessed through Load/Store instructions. The data translation lookaside buffer (DTLB) performs the address translation and permissions check. If permissions are granted, then the data is moved from the DCache into one of the architectural registers (GPR, FPR, and SPR) on a Load instruction and in the opposite direction on a Store instruction.

When instructions are dispatched to the execution units, their data is read from one or more of the architectural registers. The associated Execution Unit performs the appropriate transaction, and, if data is produced, it is stored back into the register file. Once completed, if no exception was raised during execution, instructions are sent to the Commit logic.

4.1.2. POINTS OF INFLUENCE AND OBSERVATION

Once the conceptual assets have been identified, the next step in PIO is to identify where these assets can be observed or influenced in the architecture. Using the block diagram in Figure 6, examples of such entities may be ICache, DCache, architecture registers, execution units, and so on. The key is to answer the questions listed in Section 4 for each point. For example, use the DCache as an observation and influence point for the conceptual asset data:

1. ***Confidentiality****:* Does DCache expose any confidentiality of data?—Yes. Caches are a shared resource that have been known to leak information under certain circumstances.

2. ***Integrity****:* Does DCache allow any modification of data?—No. The DCache by itself cannot modify data but it can replace when a store operation is requested. However, this is expected behavior and should not result in a "yes" to this question.
3. ***Availability****:* Can DCache prevent data from being available for functional operation?—Yes. Thrashing or exhausting the cache can prevent data from being available, in a timely fashion.
4. ***Undermined Expected Behavior****:* Does DCache have any special behaviors that can prevent data from being available for normal operation?—No. There are no features in the DCache that prevent data from being available.

Questions #1 and #3 raised the concern about confidentiality and availability. Therefore, the microarchitecture of the DCache can be considered structural assets such as the RTL logic for replacement policy, DCache contents, and internal state. These logic blocks would require an Asset Definition object based on the security objective of confidentiality. Table 3 lists more potential structural assets using the PIO approach. Note that this table is not comprehensive.

TABLE 3 Structural assets for the CPU core

Conceptual Asset: Instructions			
Observation/Influence Point	Rationale	Structural Asset(s)	Security Objective at Risk
ICache	Caches, if not protected, can be used as covert/side channels to exfiltrate data	ICache replacement policy, ICache contents, and ICache internal state	Confidentiality, availability
BP (Branch Predictor)	BP, if not protected, can be used to influence the flow of control. BP also allows speculative execution, which opens the possibility of exploiting transient execution attacks	Branch prediction history and target addresses	Integrity
ITLB	Caches, if not protected, can be used as covert/side channels to exfiltrate data	Memory mapping, ITLB contents, and ITLB replacement policy	Confidentiality, availability
Conceptual Asset: Data			
GPR, FPR	General-purpose registers are typically shared between multiple processes	Registers	Confidentiality
Functional Units (ALU/Shift, Branch, SIMD, etc.)	The processing time can reveal the data processed if directly dependent on the data itself	Source and data registers	Confidentiality

Conceptual Asset: Data *(continued)*			
DCache	Caches, if not protected, can be used as covert/side channels to exfiltrate data	DCache replacement policy, DCache contents, and DCache internal state	Confidentiality, availability
DTLB	Caches, if not protected, can be used as covert/side channels to exfiltrate data	Memory mapping, DTLB contents, and DTLB replacement policy	Confidentiality, availability

5. SUMMARY

This paper presents two methodologies to help IP developers identify assets in an IP: CSA and PIO. These methodologies are not mutually exclusive, and they are not the only means to identify assets within an IP. They are designed to simplify the application of the SA-EDI standard, and both require an architectural diagram. The CSA approach focuses on security objectives to identify conceptual assets. Once the conceptual assets are identified, the RTL representation of these assets, such as registers, modules, gates, and so on, are the structural assets. These structural assets create the Asset Definition objects in the SA-EDI standard.

The PIO approach still focuses on conceptual and structural assets but identifies them differently. This method starts by looking at the interface of the IP to identify conceptual assets. Once identified, the focus becomes points of influence and observation on these assets. These are points through which an asset may be compromised. If a security objective can be compromised, then the RTL representation of these points would be a structural asset and thus require an Asset Definition object.

Both methodologies can be applied to any IP, but PIO may be better suited for complex IP such as CPU cores or subsystems. This is because it starts with an abstraction of the architectural diagram to reduce complexity by focusing on the IP interface. The CSA approach does the opposite: it focuses on components within the architecture to identify how a security objective can be compromised. Both are effective and one must make a judgement call as to which one works best for the IP in question.

6. REFERENCES

The following sources have either been referenced within this paper or may be useful for additional reading:

[1] Accellera Systems Initiative. (2021). *Security Annotation for Electronic Design Integration Standard 1.0*. [Online]. Available: https://www.accellera.org/downloads/standards/ip-security-assurance

[2] G. Liu. (2014). *Gaussian Noise Generator*. OpenCores. [Online]. Available: https://opencores.org/projects/gng

[3] OpenCores©. 1999-2023. [Online]. Available: https://opencores.org/

[4] M. Dworkin et al. *Advanced Encryption Standard,* NIST FIPS Standard-197, Federal Inf. Process Stds. (FIPS), Gaithersburg, MD, USA, 2001.

[5] Integrated Silicon Solution Inc. (2008). *ZBT SRAM Controller*. OpenCores. [Online]. Available: https://opencores.org/projects/zbt_sram_controller

[6] MITRE. *Common Weakness Enumeration*. [Online]. Available: https://cwe.mitre.org/

APPENDIX A

ABBREVIATIONS, ACRONYMS, AND DEFINITIONS

Many of the terms, abbreviations, and acronyms in this paper are defined in the SA-EDI standard and, therefore, are not listed. Many of the acronyms in this paper are defined in line with their usage and not listed below. Those that are introduced and not defined are listed below.

- ALU Arithmetic logic unit
- ASIMD Single instruction, multiple data
- FP Floating point
- FPR Floating-point register
- IC Integrated circuit
- IP Intellectual property
- GPR General-purpose register
- RTL Register-Transfer Level
- SPR Special-purpose register

RAISING THE WORLD'S STANDARDS

3 Park Avenue, New York, NY 10016-5997 USA http://standards.ieee.org

Tel.+1732-981-0060 Fax+1732-562-1571