Specification
DI Sign-HEP - An Open and Reliable Hardware Security Module
V0.9

Steffen Reith
steffen.reith@hs-rm.de

Sergei Andreev
andreev@ihp-microelectronics.com

Fabian Buschkowski
fabian.buschkowski@rub.de

Milan Funck
milan.funck@dfki.de

Markus Fritscher
fritscher@ihp-microelectronics.com

Torsten Grawunder
torsten.grawunder@swissbit.com

Tim Henkes
tim.henkes@hs-rm.de

Norbert Herfurth
herfurth@ihp-microelectronics.com

René Rathfelder
rene.rathfelder@iav.de

Marc Stöttinger
Marc.Stoettinger@hs-rm.de

Julian Wälde
julian.waelde@hs-rm.de

Arnd Weber
arndwweber@gmail.com

August 22, 2025

Hardware Security Modules (HSMs) are fundamental components in the realm of IT security, serving as specialized devices for managing and protecting digital keys, carrying out cryptographic operations, and safeguarding sensitive data. Their significance extends beyond conventional security roles, as they play a pivotal role in promoting digital sovereignty.

To enhance security, it is crucial that all design artifacts are open and accessible for validation by third parties. In this context, Sign-HEP is committed to exploring innovative methods to build trustworthy, powerful, and reliable HSMs through the application of open-source principles. This approach involves using an open Process Design Kit (PDK), open development tools, and

open hardware descriptions to achieve a transparent HSM design.

Sign-HEP aims to demonstrate that the closed-source methods currently used in HSM products are not the only viable path to successfully creating essential hardware. Adopting an open-source framework, Sign-HEP hopes to encourage cooperation, enhance security through community involvement, and open the door to a fresh approach in HSM creation. This initiative not only challenges existing paradigms but also seeks to establish a foundation for more secure and resilient hardware solutions, thereby contributing to the broader goals of digital sovereignty and trust in IT infrastructures.

This document contains the specification the research consortium intends to implement, including the specifications for our first tape-out, which contain some possibly non-open black boxes for components which are not yet ready. These black boxes can contain proprietary components to accelerate the development process. The necessary interfaces are designed in such a way that these non-open components can be easily exchanged in the future.

Contents

1 Introduction

HSMs are designed to manage and safeguard cryptographic keys and execute various cryptographic operations. Therefore, these modules play a crucial role in ensuring the confidentiality, integrity, and authenticity of all kind of data within various applications, including electronic payments, secure communications, and data protection. Moreover, such a secure hardware environment generates cryptographic keys in a trustworthy and safe way.

Kerckhoffs' principle is an essential in cryptography, stating that a system's security should depend on the secrecy of the key rather than the obscurity of its design. This principle promotes IT security on the software side, as it enables transparency and validation of cryptographic techniques. In contrast, hardware production is based on secrets, such as proprietary designs and nondisclosure agreements, which strongly contradicts this principle. Long-term experience shows that all secretive approaches create vulnerabilities, reduce trust, limit community collaboration, and may lead to obsolescence as security threats evolve. In addition to the technical difficulties, communities are also important for the long-term development of those projects. The closed approach, however, is not conducive to such a community-driven project and even makes this method impossible to apply. The growth and added value generated by the Internet services demonstrates both the disadvantages of the closed development process and the opportunities for open methods.

For these reasons, attempts have been made for some time to develop HSMs using open development methods. The most prominent examples are the OpenTitan project[1] and its further development, Caliptra[2] (cf. Figure 1). The minimalist Caliptra silicon Root of Trust (RoT) aims to maximize agility and applicability and aims to help the industry to quickly adopt security features. Additionally, this transparency fosters trust among users and developers, as vulnerabilities can be identified and addressed collaboratively. A Caliptra-module supports key generation, encryption, and secure boot functions. It is designed to be modular, allowing it to be integrated into a wide range of applications. By adhering to industry standards, it helps to fulfill regulatory requirements and best practices for data protection. Hence, Caliptra promises a robust and open approach for safeguarding information and establishing trust in electronic platforms.

Outstanding features of Caliptra:

1. High-Level Security Services: Caliptra provides a comprehensive suite of security services, including identity, measured boot, and attestation capabilities. These features ensure that the integrity of the system is maintained from the moment it is powered on.

[1] https://opentitan.org/
[2] https://github.com/chipsalliance/Caliptra

2. Modular Design: The modular nature of Caliptra allows it to be easily integrated into various types of System on Chips (SoCs) such as CPUs, GPUs, DPUs, TPUs and NAND Flash Controller (SSD). This flexibility makes it suitable for a wide range of applications, from datacenters to edge devices.

3. Compliance with Industry Standards: Caliptra adheres to several industry standards and specifications, including those from the Trusted Computing Group (TCG) and the National Institute of Standards and Technology (NIST). This compliance helps organizations meet regulatory requirements and implement best practices for data protection.

4. Post-Quantum Cryptography (PQC) support: Caliptra includes support for post-quantum cryptography, ensuring that it remains secure against future threats posed by quantum computing.

5. Physical Attack Countermeasures: Caliptra incorporates various countermeasures to protect against physical attacks, enhancing the overall security of the system.

6. Open-Source and Collaborative Development: The development of Caliptra is open and collaborative, with contributions from major industry players like Microsoft, Google, AMD and Nvidia. This openness fosters innovation and allows for continuous improvement of the technology.

As an open standard [3] widely supported by the industry, Caliptra offers an ideal starting point for the DI-Sign HEP project, as the usability of the results is improved by following this standard. The aim of DI-Sign HEP is therefore a potentially modified, but functionally Caliptra-compatible implementation. "DI" stands for "Design Initiative". We often truncate its name to Sign HEP.

The open approach enables validation by independent third parties to expose the security features and vulnerabilities of a device. Therefore, independent auditors can provide ideas for security practices that may not be known or apparent to developers or manufacturers. Hence, open-source software enhances credibility, which is crucial for stakeholders such as customers, regulatory bodies, and partners. Openly sharing their findings and recommendations can foster a culture of transparency. As a result, stronger security practices and improved device security can be achieved through collaborative efforts between manufacturers and independent evaluators. As security threats evolve, an ongoing validation processes can help adapt and enhance security measures in response to emerging risks.

The tools used to design chips are integral to the success and viability of open-source hardware projects. These tools facilitate design, verification, and implementation processes, enabling developers to create robust hardware solutions that adhere to open-source principles. Moreover, open-source Electronic design automation (EDA)-tools open a project to a broad community, since they provide free or low-cost access to essential functionalities. This democratizes hardware design, allowing a broader range of individuals and organizations to contribute to and benefit from open-source projects. In retrospect, much of the success of open-source software (e.g. the GNU-tools and the Linux kernel) can be attributed to the development of the GCC compiler infrastructure. For this reason, open EDA-tools have the potential to play a comparable role in the successful establishment of open-source hardware.

[3] https://github.com/chipsalliance/Caliptra

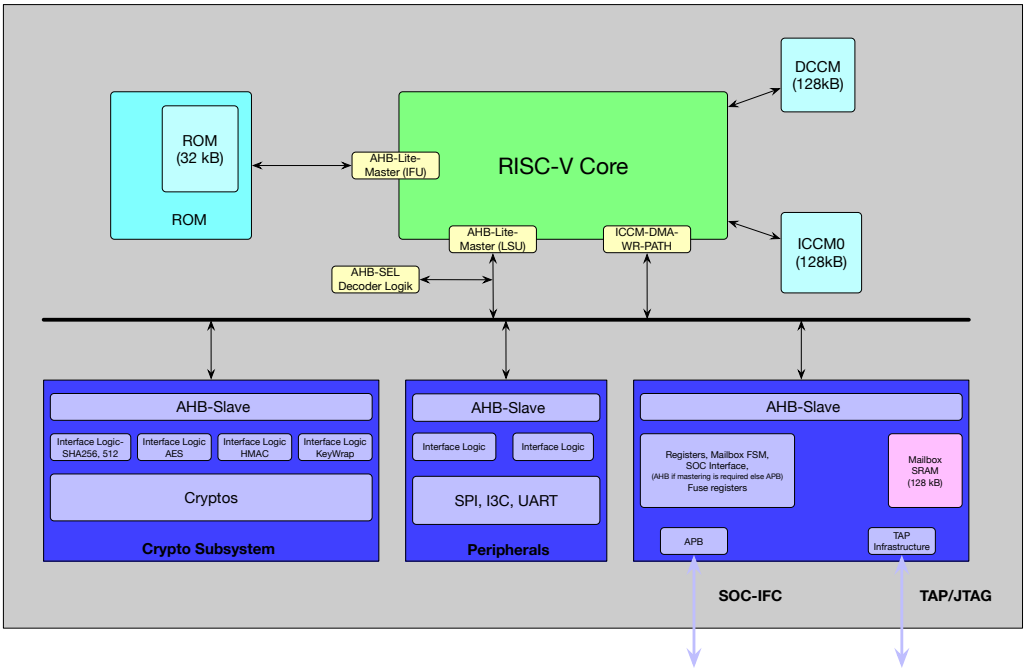

Figure 1: A Caliptra Root of Trust subsystem (cf. [Pro22])

Due to these factors, DI-Sign-HEP[4] aims to demonstrate methods for implementing the Caliptra specification in a comprehensive manner, utilizing open-source methodologies at all levels. Closed approaches (e.g. random number generators) are only pursued to accelerate DI-Sign HEP. Clearly defined interfaces make these easily interchangeable and can thus be replaced step by step by powerful open implementations. Specifications and documents are made freely available, and all developed hardware descriptions are published as source code. The required build environment is provided with the source code too. The indispensable development tools for the creation of a layout (Graphic Design System (GDS)-file) exclusively open-source tools, meaning that the fundamental functionality can be replicated by third parties at any time. An exception to this principle are closed-source components which we will use in our first tape-out. We are conducting research on how to replace them finally. Again, closed technology is used to speedup the development process only. The final results themselves are then based solely on open methods.

It is also important to ensure that the manufacturing process is as transparent as possible. For this purpose, we use the open IHP PDK, which provides a 130 nm chip technology. This means that the layouts are also freely available as GDSs files and do not prevent third parties from checking them. On the contrary, the consortium would like to see error descriptions and information on how the security of the implementation can be further improved. In addition to the obvious advantages for security checks, research will also reap benefits, as the latest research methodologies can be evaluated in practice.

To achieve the objectives of the project, special attention should be paid to the following points.

Open until tapeout: All used components (components which are traditionally called Intellectual Property (IP), tools, PDK) are open source. Exceptions are components that are implemented using proprietary technology for accelerating the development process. In the future, these will be replaced by open variants, which is why particular attention is paid to suitable interfaces.

[4] https://hep-alliance.org/

Vulnerabilities in RNG: Random Number Generators (RNGs) play a crucial role in cryptographic operations, simulations, and secure communications. However, they can be susceptible to various types of attacks, especially if their entropy sources are weak or predictable. The following aspects should be considered to ensure the security and reliability of RNGs:

1. Entropy Source Strength
   - Ensure that the entropy sources used in the RNG are unpredictable and robust against external influence.
   - Use multiple independent entropy sources to mitigate the risk of a single point of failure.
   - Monitor entropy levels in real time to detect any degradation in randomness quality.
2. Algorithm Robustness
   - Utilize a suitable cryptographically secure pseudo-random number generators (CSPRNGs) compliant with recognized, backdoor-free standards.
   - Avoid reliance on deterministic RNGs unless properly seeded with high-entropy input.
   - Regularly update RNG implementations to mitigate known vulnerabilities and weaknesses.
3. Resistance to Predictability Attacks
   - Protect against state compromise attacks by periodically reseeding the RNG.
   - Implement forward secrecy mechanisms to prevent attackers from inferring past or future values.
   - Secure entropy collection and storage to prevent unauthorized access or manipulation.
4. Regular Entropy and Statistical Testing
   - Conduct continuous testing of RNG output using statistical tests to detect patterns or biases.
   - Implement health checks and self-tests to ensure RNG integrity before use.
   - Log entropy tests and anomalies for audit and compliance purposes.

Vulnerabilities in NVM: Non-Volatile Memory (NVM) is used for persistent data storage in various applications, including firmware, cryptographic keys, and system configurations. However, NVM can pose significant security risks, particularly in the following areas:

1. Unauthorized Access and Data Extraction
   - NVM retains data even when power is lost, making it a target for attackers attempting to retrieve sensitive information.
   - Strong encryption (e.g., AES-256) should be used to protect stored data, ensuring that even if physical access is gained, the data remains unreadable.
   - Secure key storage mechanisms have to be used, such as Hardware Security Modules (HSMs) or Trusted Platform Modules (TPMs), to protect cryptographic material.

2. Tampering and Integrity Attacks
    - Attackers may attempt to modify firmware, bootloaders, or configurations stored in NVM to introduce backdoors or malware.
    - Cryptographic integrity checks should be implemented using hash functions (e.g., SHA-256) and digital signatures to verify the authenticity of stored data.
    - Usage of secure boot mechanisms should be required to ensure that only trusted firmware and software are loaded.
3. Side-Channel and Physical Attacks
    - Physical probing, voltage glitching, or electromagnetic analysis can be used to extract sensitive data from NVM.
    - Countermeasures should be employed such as active tamper detection, randomized memory access patterns, and shielding against electromagnetic analysis.
    - Hardware-enforced access control mechanisms could be utilized to prevent unauthorized read/write operations.
4. Access Control and Authentication
    - Access to NVM can be restricted by enforcing authentication mechanisms, such as hardware-based access control lists (ACLs) or secure enclaves.
    - Role-based access control (RBAC) should be used to ensure that only authorized system components or users can modify stored data.
    - Secure erasure mechanisms (e.g., cryptographic erase) should be implemented to prevent unauthorized data recovery when decommissioning or repurposing devices.
5. Secure Firmware Updates and Patching
    - Firmware updates have to be delivered securely using digitally signed packages to prevent unauthorized modifications.
    - Rollback protection should be implemented to prevent attackers from downgrading firmware versions to exploit older vulnerabilities.
    - Attestation mechanisms have to be used to verify firmware integrity before execution.
6. Data Retention and Secure Deletion
    - Some types of NVM retain data for extended periods, even after deletion attempts. Secure wipe techniques should be implemented such as overwriting multiple times or leveraging built-in secure erase commands.
    - For particularly sensitive applications, self-encrypting storage has to be considered where encryption keys are destroyed to render data irrecoverable.

Interdisciplinary Approaches: The results of the project should be modifiable for different application areas like memory controllers, automotive systems, medical technology, aerospace technology, and critical infrastructure and IoT devices in general. For this reason, SIGN-HEP aims to provide an industry-agnostic solution.

Adaptability: Proprietary semiconductor nodes may require specific IP adaptation and optimizations, and SIGN-HEP is open to such adaptations to improve industrial usability and portability.

EDA Workflows : The whole source code is deployable with an open-source EDA toolchain and PDK. The functional results in the TapeOut GDS-II files are independent from the used EDA workflow.

This document provides the specification for SIGN-HEP, a prototypical, Caliptra-compatible HSM. The project SIGN-HEP (Secure Industrially applicable Generally Normalized HSM based on open EDA tools and Processors) builds upon the open Caliptra RoT specification[5]. SIGN-HEP is prepared to update specifications if Caliptra specifications are updated (as of writing, to version 2.0). SIGN-HEP is designed to integrate security directly into chips for use in various environments.

1.1 Motivation

In recent years, Open Source Chip Design, such as Yosys, OpenRoad, and the open SkyWater PDK, and more broadly, Open Source Hardware, such as RISC-V and OpenTitan designs, have gained significant visibility and use. This new paradigm, in Germany pushed by the VE-HEP project, has transformed skepticism into curiosity and led to first products. This evolution highlights the growing potential of open-source methodologies in addressing real-world challenges.

The DI-SIGN-HEP project takes on the critical task of addressing these challenges through a detailed gap analysis of existing open-source design approaches. It aims to identify and overcome the barriers preventing open-source methodologies from achieving product-ready designs. A key area of focus is HSMs, which have gained prominence in response to the increasing demand for secure and transparent systems. The integration of open-source principles into the design of RoT elements is particularly noteworthy. Traditionally, RoT elements represent some of the most tightly guarded intellectual property in hardware design, making open development in this domain not only challenging but also groundbreaking.

DI-SIGN-HEP seeks to integrate the Caliptra framework, HEP-HSM, and the open-source IHP-Open130-G2 PDK to establish an industrially viable, product-ready environment. By prioritizing accessibility and transparency (wherever possible), the project aims to demonstrate how high-quality RoT elements can be seamlessly integrated into product designs while meeting stringent industrial standards. This initiative aims to push the boundaries of what is achievable with open-source hardware and sets a new benchmark for secure, reliable, and transparent electronic design.

Through this endeavor, the project aspires to demonstrate the transformative potential of open-source approaches in the most challenging and traditionally restricted domains of hardware security.

1.2 Background

Every IT security strategy relies on hardware security modules as solid foundations. Building step by step on this foundation, higher layers of a system or user software can provide the security assurances needed.

[5] https://github.com/chipsalliance/Caliptra/blob/main/doc/Caliptra.md#industry-standards-and-specifications

Kerckhoffs' principle is a fundamental building block of IT security. The design principles and development of a security solution should be disclosed; security is based exclusively on the (secret) key. This principle has been proven to be fundamental to the design of cryptographic algorithms and software based on them, and it is not questioned by the research community, while proprietary security modules may rely on using secret production methods. For this reason, many security solutions are based on open-source software libraries that can be more easily verified by experts. Openness alone is not a guarantee of security, as such software may also contain errors and even backdoors. However, verification is relatively straightforward and can be performed at every stage of development. This also offers the further advantage that the security of such systems is not based on proprietary knowledge, which means that all necessary improvements can be made through the cooperation of a diverse community.

Hardware development stands in stark contrast to this unusually successful, open and cooperative development model. Partly because of historical reasons, hardware development is based on a development model based on closeness and secrecy. It is evident that this development approach yields significant economic benefits, however, it is incongruous with the prerequisites of IT security. Assessing the quality of security measures, cooperative revision and improvement,and the dissemination of knowledge is made massively more difficult or even impossible, which can lead to massive security gaps in security modules or have already occurred.

Because of this, the HEP project has developed technical methods and tools that enable the design of cryptographically secure hardware in a completely open process (cf. [Hen+24]). All design and development phases can be traced due to the utilization of open EDA tools and their foundation on open hardware descriptions. The outcome is a layout description within an open PDK for a structure size of 130 nm. This means that even the lowest level of a hardware description can be checked and traced. Based on the results of VE-HEP, the DI-Sign HEP project aims to develop methods for the construction of a hardware security module compatible with the Caliptra standard. Therefore, the DI-Sign HEP project aims to develop completely open methods for the construction of a hardware security module. This should show that open methods for the construction of hardware can also be commercially interesting and can deliver products that can be economically used later. Caliptra is particularly interesting because it is being promoted by a broad industrial consortium and is therefore likely to be successful in terms of commercial implementation. Furthermore, it is the only comparatively mature example of a standard for a security module that is publicly available (more recent than OpenTitan). Due to these factors, DI-Sign HEP aims to ensure the strictest possible implementation of the Caliptra standard, using current open methods. This allows the results to also be used in a commercial (proprietary) EDA tool environment.

1.2.1 VE-HEP Project

The VE-HEP project ("Hardening the value chain through open source, trustworthy EDA tools and processors") initiated research into open PDKs outside the U.S., resulting in the availability of the open IHP PDKs, which will be further developed in SIGN-HEP.

1.2.2 Caliptra RoT Specification

- Caliptra is designed to be used with proprietary PDKs, meaning certain key components of the HSM design at the GDS level (tape-out) are not publicly accessible. These components include the True-Random-Number Generator (TRNG) and NVM,

both critical for security. As observed by Shumow and Ferguson (cf. [SF07]), such components may contain Trojans.

- The Caliptra RoT specification is primarily data center component-oriented but can be adapted for use in other applications, such as automotive systems, memory controller chips, edge computing applications and various IoT devices. The reference implementation is provided in a subset of SystemVerilog to facilitate integration and customization.

To this end we plan to design a open source hardware security module, reuse many components of Caliptra, aiming for full functional compatibility with Caliptra. This means that the original Caliptra firmware will be used.

1.3 Formal Verification

Just as in the VE-HEP project, formal verification will be an integral part of ensuring correctness of the provided HSM. The formal verification efforts will primarily focus on the functional correctness of the hardware components that are added to or modified within the Caliptra framework. This primarily includes the RISC-V core and parts of the cryptographic hardware. Since the VexRiscv will also serve as the RISC-V core in this project, the formal proofs from the VE-HEP project, concerning the compliance of the VexRiscv with the RISC-V ISA, can be reused and extended. We will additionally investigate how to verify security properties of the interfaces to the Caliptra environment, in particular the protocol to access the Mailbox component. The proofs are formalized in SpinalHDL, connected to the various hardware modules through automated scripts, and finally executed using Yosys/SymbiYosys. We use SpinalHDL not only because most of the components in question are written in the same language, but also because it allows us to express and reason about properties at a very abstract level. This way, we can both design and automate more complex proofs that are still feasible and comprehensible.

1.4 Collaboration

- Funded by the German Federal Ministry of Research, Technology and Space (formerly Ministry of Education and Research), SIGN-HEP is a collaborative project that brings together both research and industry partners.

- Industrial partners include Swissbit, Hyperstone, and IAV Automotive Engineering. Non-industrial partners are: IHP, TU Berlin, Hochschule RheinMain, Ruhr Universität Bochum, DFKI Bremen, cf. https://www.elektronikforschung.de/projekte/di-sign-hep and https://www.linkedin.com/posts/ihp_opensource-securitymodules-chipsalliance-activity-7226605009210146817-z8wr/

2 Application Scenarios

To validate and demonstrate the capabilities of SIGN-HEP a system demonstrator is being planned. This demonstrator is designed to replicate real-world scenarios across various platforms, ensuring that the core functionalities of SIGN-HEP are tested and verified.

It will show how the developed and manufactured system can be integrated into different application environments. The demonstrator will cover two different application cases, which represent the knowledge domains and target applications of the industrial partners from the research project SIGN-HEP. In the first use case, the HSM will be used by a

connected controller to ensure security-relevant tasks like secure boot, secure update, as well as secure communication. The goal is to demonstrate the technological matureness of the whole system, as well as detect eventual gaps for improvement. The second use case will implement the developed SIGN-HEP HSM in conjunction with a memory controller to test its usability in the scope of memory relevant functions by a secure storage. An overview of the different system components for the SIGN-HEP demonstrator, the evaluation hosts, and the SIGN-HEP system on a chip (SoC) will be given. Also the two application environments are being described in this chapter, as well as the planned use case scenarios for each of them.

2.1 SIGN-HEP Demonstrator and SIGN-HEP SoC

For practicality reasons the SIGN-HEP-SoC will be manufactured as a standalone chip which can then be connected to the application or memory controller via an applicable interface (e.g. SPI). In a productive scenario the SIGN-HEP hardware would be integrated in a SoC for the specified task.

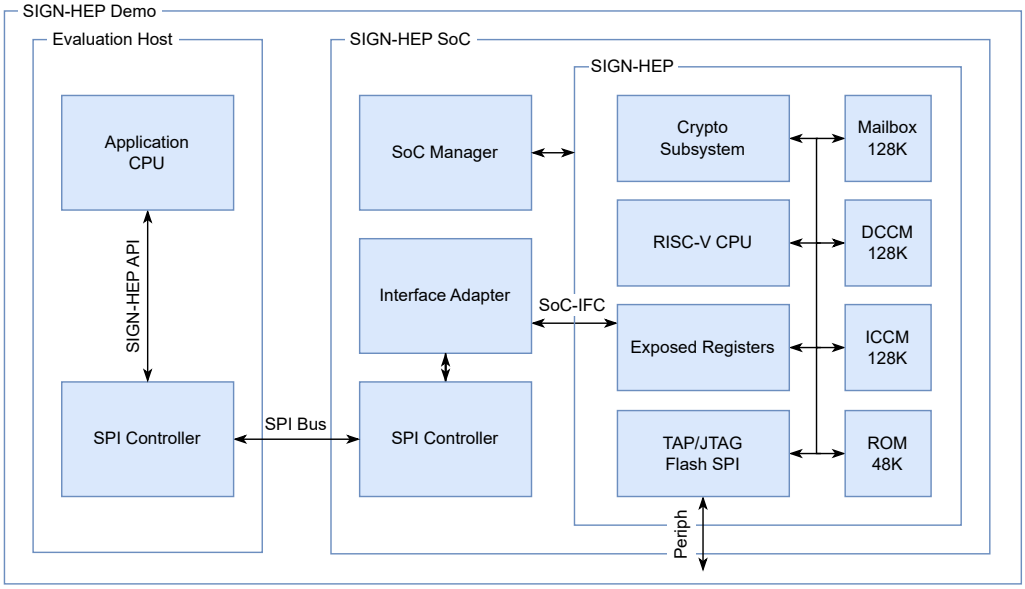

Figure 2: High-Level View of the SIGN-HEP Demonstrator.

In Figure 2 an overview of the SIGN-HEP Demonstrator setup for the components of an Evaluation Host and the SIGN-HEP SoC as well as their connections is illustrated.

2.1.1 SIGN-HEP SoC

What we define as SIGN-HEP will be a hardware-integrateable block that can be integrated into larger systems or standalone chips to implement the functionalities easily and convenient. SIGN-HEP inherits the Caliptra RoT architecture and integrates the specified Caliptra RoT capabilities. To keep the overhead and points of failure low, we only manufacture the SIGN-HEP SoC with a minimal interface as a SoC. This will be called "SIGN-HEP SoC" (cf. Figure 2). The integrated interface on the SIGN-HEP SoC will be a Serial Peripheral Interface (SPI) bus. The SIGN-HEP SoC consists therefore of the following components:

- SIGN-HEP Block

- SPI Controller (Slave)
- Interface Adapter
- SoC Manager

The first to components have been specified above. The Interface Adapter realizes a Conversion logic between the SPI controller and the SIGN-HEP interfaces and the SoC Manager handles power, reset, clock, IO wires, and other management tasks. The SPI will be available in different forms and formats throughout the development. These include a compiled simulator, an Field-programmable gate array (FPGA) implementation, and finally a manufactured Application-specific integrated circuit (ASIC). Each implementation should share the same functional behavior and core Register-transfer level (RTL). The compiled simulator provides a software-based environment for early development, testing, and exploration, enabling the integration and evaluation of components without the need for physical hardware. An FPGA provides a reconfigurable and adaptable platform for the purpose of prototyping and validating the design in a near-hardware environment, thereby enabling rapid iterations. As the last iteration, the ASIC serves as a hardware demonstrator, showing the SoCs capabilities in a final form, optimized for power, area, and performance. Different functionalities that will be accomplished by the SIGN-HEP SoC are:

- Loading SIGN-HEP software: via SoC SPI bus, or SIGN-HEP Flash SPI.
- SoC reset and initialization: Procedures for resetting and initializing the SIGN-HEP block and SoC.
- SoC debugging and diagnostics: Methods for debugging and performing diagnostics on the SIGN-HEP SoC.

2.1.2 Evaluation Host Environments

The host environments are responsible for controlling and evaluating the SIGN-HEP SoC in the different Evaluation scenarios and their respective test scenarios. These Environment Hosts should emulate real-world counterparts of the given tasks, in our case as application Core or Memory Controller. For testing purposes the host environments can also be equipped with extended components and interfaces for debugging, measurement or controlling operations which would otherwise not be part in an production environment.

NAND Flash Controller Environment The NAND Flash Controller Environment will be used to evaluate the SIGN-HEP SoC in combination with NAND Flash memory. For this purpose, communication via SPI to the SIGN-HEP SoC must be realized, as well as communication via the NAND Flash Controller SPI to a supported NAND Flash memory.

- NAND Flash controller
  - system to use SIGN-HEP SoC in combination with NAND Flash memory
  - included SPI Controller (Master) is required for communication with the SIGN-HEP SoC.
  - SIGN-HEP SoC is used in the same way as commercial TPM chips

Application Unit Environment The Application Unit Environment uses the SIGN-HEP SoC for cryptographic system functions. This will include use cases for secure boot and update, as well as encryption and decryption of data. It is planned to connect the Application Unit to the SIGN-HEP Soc via SPI bus. Other components of the Environment Unit or a specific system is not yet declared and depends on the requirements of use cases and test scenarios.

SIGN-HEP API The Application Unit runs test software to send Caliptra commands to the SIGN-HEP and receive the corresponding Caliptra responses.

- A software library that implements the required Caliptra commands.
- Communicates with SIGN-HEP via the SPI interface.

2.2 SIGN-HEP Use-Cases

For demonstrating the functionality of the SIGN-HEP SoC in the discussed and presented host environments practical use-cases will be selected, designed and tested. These use cases should comply with requirements for given cybersecurity standards or Protection Profiles.

Application Unit Environment The Use-Cases in the Application Unit Environment lean on security relevant capabilities in industrial, automotive or other comparable environments for embedded devices. A real Central Processing Unit (CPU) core (e.g., ARM) on a board, or a simulated core (e.g., RISC-V) on a Windows/Linux PC (preferred for simplicity), with an attached SPI controller.

Possible Use-Cases:

- Secure Boot: Ensures that the device boots only trusted software by verifying a digital signature.
- Secure Update: Secure mechanisms for updating firmware, including authentication and integrity checks to prevent unauthorized or malicious updates.
- Access Control: Implementing robust access control mechanisms to restrict access to the device and its functions to authorized users only.
- Encryption (and Decryption): Use of encryption to protect data, ensuring that sensitive information is not accessible to unauthorized parties.
- Authentication: Authentication methods for users and devices to verify identities before granting access.
- Intrusion Detection and Prevention: Capabilities to detect and respond to unauthorized access attempts or other malicious activities.
- Security Patching: Ability to apply security patches and updates promptly to address vulnerabilities.
- Data Integrity: Ensuring the integrity of data through checksums, hashes, or other methods to detect and prevent data corruption or manipulation.

NAND Flash Controller Environment The Nand Flash controllers typically implement an internal RoT for secure boot and firmware update and have an I2C or SPI interface to external secure elements (e.g. smartcards) for specific requirements of different security levels.

In this demonstrator example, a NAND flash controller is used in the sense of an application CPU with a build-in SPI controller to demonstrate the possible use cases:

- Secure storage
- Secure Hash Algorithms (sha) / streaming sha
- Symmetric cryptography, e.g., Advanced Encryption Standard (AES)
- Asymmetric cryptography, e.g., Elliptic-curve cryptography (ECC)
- PQC support

More and more Swissbit storage products are continuously improved and secured by their own security functions such as RoT as well as secure boot and update functions. For different security levels or customer requirements in specific markets, smartcard, TPM or specific secure elements can be connected to the NAND Flash controller via UART, I2C or SPI. Swissbit's stated goal is to use Caliptra with the specific improvements from the Sign-HEP project as an embedded HSM in the SoC/eSoC in future NAND flash controller (SoC) or chiplet (eSoC) ASIC designs. These ASIC designs are too complex in design scope to be implemented in the context of Sign-Hep. As an alternative to embedding Caliptra in the SoC design, Swissbit use the option of replacing an existing secure element with a Caliptra HSM, which could well be a real use case for some applications.

3 Implementation

In this section we document and motivate our design decisions as well as their implications for security. The design will be used to demonstrate a complete Caliptra implementation from a 130 nm process as well as serving as a component in the demonstrator later in the project. Notably, to reduce area usage, we decided to use external memories and an external bus system for the connection of the mailbox with the application SoC. The entire module might be made on 130 nm for upcoming releases, or it could be made in a smaller technology.

3.1 The Caliptra Top Level

We want to give an overview of the Caliptra top-level to motivate the design choices.

The toplevel of Caliptra contains many components we will either directly include or have to find appropriate replacements for. It contains a CPU with two tightly attached memories for data and instructions, and a ROM for the bootloader and some parts of the firmware. Furthermore, there is an AHB-Lite bus that connects to peripherals like accelerators (AES / SHA256 / SHA512 / ECC / HMAC), a pseudo random number generator and entropy sources, a UART, the mailbox, a DMA port to the CPUs memories, as well as vaults for PRCs, keys and data. The mailbox (required for the communication between the application SoC and Caliptra) also comes with its own memory and a APB3 bus that connects to the application SoC. The mailbox can be written and read from both busses and the component in Caliptra manages the arbitration and locking. A JTAG

interface is also exposed for development purposes. Figure 3 contains a diagram of the described Caliptra toplevel.

Memory	Size
ICCM	128 kB
DCCM	128 kB
Mailbox	128 kB
ROM	42 kB

The three 128 kB SRAMs require a total of 384 kB. On a 130 nm process, this memory requires a lot of area in comparison to FPGAs. By no means a prohibitive amount, but for the purposes of this project, we will use external memories to implement ICCM, DCCM, and the SRAM of the mailbox.

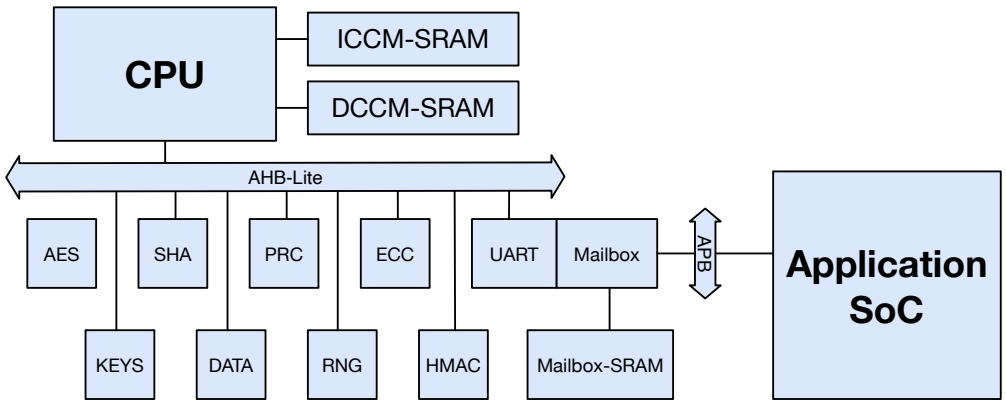

Figure 3: A diagram of the caliptra top-level as described above.

3.2 Caliptra Integration

3.2.1 Replacing the CPU

As part of this project, we plan to demonstrate that a functionally complete Caliptra with an unmodified firmware can be instantiated with a different RISC-V CPU. This approach demonstrates that Veer can be replaced by other RISC-V implementations. This means that no compromises need to be made regarding the implementation language (SystemVerilog vs. SpinalHDL), as the SystemVerilog support of the open development tools may not be sufficient at this time. Furthermore, as a MetaHDL, SpinalHDL offers unusual flexibility in exploring the design space, i.e., numerous variants (e.g., cache, bus systems, or different peripherals) can be quickly implemented, simulated, and tested to find an optimal solution. Therefore, replacing Veer with Vexriscv is of central importance to the project.

3.3 External Memories

Serial bus protocols can be used to attach external memories without introducing a wide bus. QSPI or octal SPI like external pseudo-static RAM devices are commercially available at trivial cost. We plan to realize area savings by employing such devices to implement the ICCM, DCCM, and Mailbox SRAM. The interfaces for these devices are also already available in open source and can be easily integrated into any design.

3.4 External Application SoC

Similarly to the memory, we also intend not to integrate the application SoC into the design and plan to have instead a 32 bit wide APB3 bus. This allows us to realize the application SoC on an FPGA for evaluation and demonstration purposes.

3.4.1 Motivation

Not integrating the application SoC with Caliptra will again allow to not integrate the memory of the application SoC and the application SoC itself. This allows us to evaluate the prototype with different application SoCs we plan to implement on an FPGA. More importantly, it allows us to reduce the area requirements for our tape-outs and hence allow for more tape-outs and designs.

3.5 Root Of Trust Elements (OTP, PUF, Entropy)

Caliptra specifies the requirements for the RoT-elements, including external entropy[6], CSRNG[7], and OTP[8].

Initially, we will rely on third-party IP development to design RoT elements integrated into the open-source PDK "IHP-Open130-G2." This strategy ensures the development of a complete set of elements that adhere to state-of-the-art security standards while enabling characterization and preparation for certification. While these elements will be fully accessible to the open-source community, including RTL, documentation, testbenches, timing data, and design guidelines, the publicly available GDS will be a "phantom GDS." By opening up a domain traditionally shrouded in secrecy, this project marks a milestone for the open-source hardware security community.

The elements in particular are central and critical to the security of a security module. Openness enables the identification of gaps in implementations and allows for improvement by third parties. We consider these advantages for security gains to be greater than those that may result from strict secrecy, which is one of the central arguments of the old development model. By disclosing the implementation details, we are not only strictly following Kerkhoff's principle established in security research. The project also benefits from insights into rapidly developing physical attack techniques because an implementation that is transparent at all levels is also available for improvements by third parties.

Therefore, the SIGN-HEP project aims to go even further by developing fully open elements for the IHP-Open130-G2 process. These elements will leverage CMOS and/or resistive memory (RRAM) technologies, pushing the boundaries of open hardware development. The modularity of the Caliptra specifications ensures that switching between the two approaches is straightforward due to the well-defined interfaces.

In addition to development efforts, a key area of research will focus on evaluating the strengths and weaknesses of the two approaches in terms of security, transparency, and usability. This is not a trivial task, as it requires a comprehensive analysis of how these approaches align with the broader goals of security and openness in hardware development.

3.6 Security Implications of External Busses

The decision to use external busses for the memory has direct implications for security, as attackers with physical access can trivially compromise the firmware if the instruction

[6] https://github.com/chipsalliance/caliptra-rtl/blob/main/docs/CaliptraHardwareSpecification.md#external-trng-req-hw-api

[7] https://github.com/lowRISC/opentitan/tree/master/hw/ip/csrng

[8] https://github.com/chipsalliance/caliptra-ss/blob/main/docs/Caliptra

memory can be controlled by the attacker. Similarly, the communication of the application SoC with the Caliptra system is also susceptible to attack if the the bus is subject to manipulation. In the framework of this porject, however, it is necessary to be somewhat conservative with the area. We believe the memory integration as well as integration of Caliptra with an application SoC pose no fundamental problems except for an increased area and hence cost.

3.7 Potential Mitigations

Here we want to discuss how to potentially securely use external memories and busses in a way that is secure even with physical attackers. If available and implemented, this technology would significantly enhance the utility of 130nm process nodes, and in general allow to build secure systems larger than a chip. The "MAGIC" mode unifies authenticated encryption with associated data and and error correction [Lam+24; Kou+20]. In combination with caching this could be a building block for the secure use of external memories.

Apart from encryption, authentication and error correction it's also necessary to prevent the replay of old states of the memory at future points in time to have external memory that offers similar security properties of integrated memory. This can be achieved using a monotonic counter to count writes to the part of the memory in question in the associated data of the AEAD primitive in question. Further research into securing external system components might yield a satisfactory solution for later in the project.

4 Resulting Requirements for the CPU Core Prototype

4.1 The Veer EL2 Core as CPU

Central processing unit (CPU) of the Caliptra System-on-Chip (SoC) is the Veer EL2 Core. In order to guarantee compatibility and performance, the replacement core selected must satisfy all pertinent specifications.

These requirements may arise from several key factors Integration with SoC hardware interfaces, memory compatibility, and firmware and instruction set architecture (ISA).

The replacement of the Veer EL2 Core with a Vex Core effectively drives the investigation on the flexibility of the Caliptra framework and its ROT elements and security components. Furthermore, to ensure the flexible adaptation and usage of the Caliptra framework for diverse application settings, ranging from constrained embedded devices to trusted elements in servers, the computational power of the Caliptra framework's core may undergo modifications. Consequently, it is imperative to ensure the flexibility for an agile modification of the RISC-V core in Caliptra for various application settings. Compared to the Veer Core, the Vex RISC-V core is implemented in SpinalHDL and is fully parameterized. This enables the core to adapt more easily due to the nature of the hardware description language of SpinalHDL during the design and development phase.

The replacement core must align with the existing hardware components and their associated interfaces within the SoC. Additional details for the Interfacing to the SoC are provided in Sec. 3.1 and Sec. 4.1.1. Sec. 3.1 lists the required memory components as well as their properties. The replacement must be compatible with the Caliptra firmware and support the ISA subset for which the firmware is compiled. Section 4.1.2 provides more details on the firmware requirements. These considerations ensure that the functionality of Caliptra and the firmware is preserved under the changes we intend to enact.

4.1.1 Requirements from the SoC

The core within the SoC primarily communicates with the rest of the system through an AHB Lite bus. Its integration relies on several key memory interfaces and architectural considerations:

- Tightly Coupled Memories (ICCM and DCCM): The core features two tightly coupled memories, ICCM (Instruction Closely Coupled Memory) and DCCM (Data Closely Coupled Memory). These are mapped to the AHB Lite bus via the CPU's DMA slave port. This mapping is essential, particularly for operations such as firmware loading, and must be maintained if these memories are utilized.
- OTP ROM for Boot Loader Storage: An OTP ROM is directly connected to the CPU and is used to store the boot loader. This connection forms a critical part of the SoC's initialization process.
- Mailbox Memory: Another memory interfacing with the core is the mailbox, which facilitates communication within the SoC.
- CPU Core Requirements: The core must be capable of interfacing effectively with these memory components and the AHB Lite bus to ensure seamless operation within the SoC.
- Peripheral Interfaces: No specific changes are required for peripheral interactions. These will continue to operate via the AHB Lite bus as designed in the SoC.

These considerations collectively guide the selection or design of a core that is fully compatible with the SoC's architecture and functionality.

4.1.2 Requirements from the Firmware

The firmware for the SoC is currently compiled with the `rv32imac_zicsr` instruction set architecture (ISA). This imposes several requirements on the CPU to ensure compatibility and efficient operation:

- Compressed Instruction Support: The firmware utilizes compressed instructions, necessitating that the core supports this feature. If the core does not support compressed instructions, larger memory sizes would need to be integrated to accommodate the uncompressed instruction set, leading to increased costs.
- ZICSR Extension: The core must support the ZICSR extension for control and status register (CSR) operations. Alternatively, modifications to the firmware would be required to align it with the CSR capabilities of the core.
- Atomic Instructions (A Extension): If the firmware employs atomic instructions, the core must support the A extension to handle these operations.
- Multiplication and Division (M Extension): The core must include support for the M extension to execute multiplication and division instructions required by the firmware.

These requirements are critical to ensure seamless integration and functionality of the firmware with the core. Addressing them appropriately will help maintain performance and avoid additional development or hardware costs.

5 Requirements for open Root of Trust Elements

In this chapter, we outline the requirements for open Root of Trust (RoT) elements in developing an open hardware security module. These elements include hardware components such as One-Time Programmable (OTP) memory, entropy sources, Physically Unclonable Functions (PUFs) and anti-tamper mechanisms. Additionally, we discuss derived elements like Unique Identifiers (UID), Hardware Unique Keys (HUK) and Cryptographically Secure Random Number Generators (CSRNG). We also address specific considerations for open-source implementations of these RoT elements.

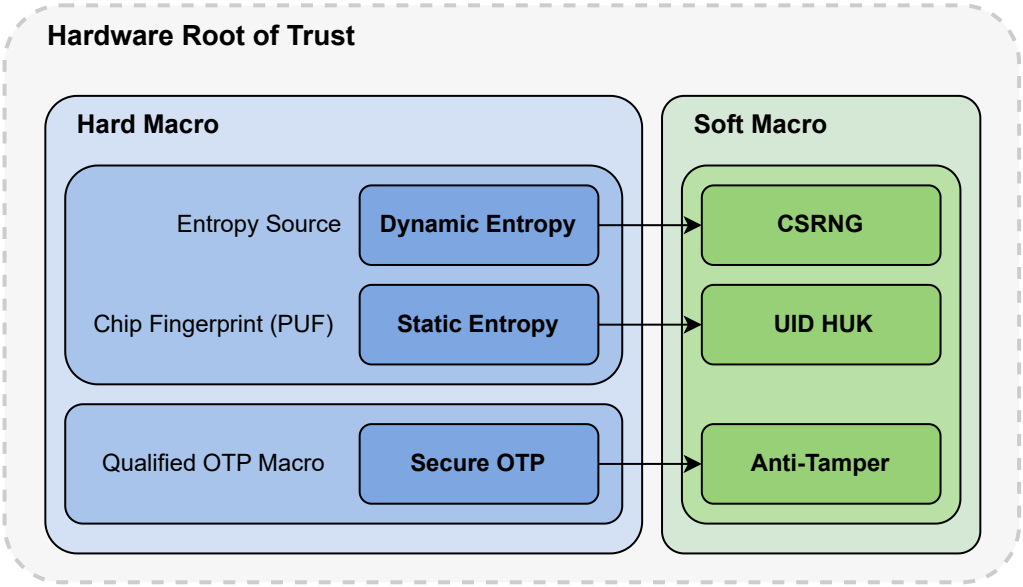

Figure 4: Root of Trust Elements.

5.1 Hardware Elements

The following section provides an overview of commonly used methods and technologies for implementing hardware-based Root of Trust (RoT) elements. It is important to emphasize that this does not imply that all the described approaches will be adopted or integrated into the SIGN-HEP project. Instead, the aim is to establish a reference framework to derive and evaluate requirements for open alternatives. These references help us identify essential features and guide the development of open, transparent, and verifiable RoT components compatible with our project goals.

5.1.1 OTP (One-Time Programmable Memory)

One-Time Programmables (OTPs) are non-volatile memory elements integrated into silicon ICs that can be permanently set (programmed) a single time. This memory stores only a few crucial parameters. Typical examples are device or serial numbers, cryptographic keys, boot configuration flags, or individual calibration values. This data rarely needs to be changed and must be stored permanently and tamper-proof.

OTP rely on physical, irreversible changes – such as blowing a fuse or altering an antifuse structure – during programming. Once these bits are written, they cannot be erased or reprogrammed, making OTPs well-suited for securely storing device configuration, keys, or calibration data that must remain tamper-resistant and unmodifiable throughout the

product's lifetime. OTP memory can be written only once and is permanently fixed, while non-volatile memory allows for multiple write and erase cycles, enabling data to be updated over time. Therefore, the OTP memory must fulfill a series of requirements.

For storing keys, Caliptra does not specify how this is to be done. Implementations of Caliptra will therefore typically use non-volatile memory technologies as available to a fab. Such memory designs might contain vulnerabilities and are not freely available yet. For SIGN-HEP, we aim at an open design of non-volatile memory. As of writing, no such designs exist. Therefore, we are conducting research on this. Furthermore, we intend to license a memory design, for quickly implementing and testing the remainder of our implementation.

- **Obfuscation:** The data in OTP memory must be obfuscated to prevent readout via optical or other analysis techniques. We will investigate whether a secure shield can be implemented, following the design of [Ngo+17].
- **Lifetime Stability:** The OTP must ensure data remains stable throughout the device's lifetime without self-healing properties.
- **Storage Capacity:** The memory should have adequate capacity to store necessary configuration data, keys, and security settings.
- **Read Speed:** Reading from OTP should be fast enough to meet the performance requirements.
- **Level of Integrability:** OTP memory should integrate easily into existing hardware designs.
- **Manipulation Resistance:** The OTP should be resistant to tampering or unauthorized modification.
- **Permanence:** Once written, data in OTP should be permanent and unchangeable.

5.1.2 Entropy Source

The entropy source is a fundamental component of the Root of Trust, responsible for generating high-quality random data critical to cryptographic operations. This section outlines the requirements for ensuring robust randomness, including compliance with AIS31 and NIST standards, and addresses challenges related to stability, environmental robustness, and resistance to external attacks.

The entropy source in the Caliptra Root of Trust must generate high-quality random data with a minimum rate of 50 kHz, exceeding 0.997 bits of entropy per output bit to meet AIS31 PTG.2 and NIST standards. It uses a conservative design, requiring 2,048 raw bits to produce each 384-bit conditioned entropy sample, ensuring robust cryptographic security. Besides this, an entropy source must fulfill some more generic requirements:

- **Validation of Entropy Class:** The entropy source should pass the test runs of the "Die Harder" suite to ensure an appropriate quality of entropy, as this test suite represents a basic test in various security validations for certification. In the case of Common Criteria certification based on AIS31, a complete stochastic model is required to model the physical effects of the random number generator that describes the distribution of the random numbers generated to ensure true randomness.
- **Stability:** It must maintain stable performance across different temperatures and over the device's lifetime.

- **External Attack Resistance:** The entropy source should be stable against various external attacks such as voltage manipulation, laser interference, physical tampering, and electromagnetic attacks.

5.1.3 PUF (Physically Unclonable Function)

Physically Unclonable Functions (PUFs) utilize inherent physical variations in hardware to generate unique and unclonable identifiers. This section details the requirements for consistency, tamper resistance, and environmental robustness, emphasizing their role in secure key generation and authentication processes.

- **Unclonability:** PUFs should generate responses that are unique and cannot be cloned.
- **Consistency:** Responses to specific challenges should be consistent over time and under different conditions.
- **Tamper Resistance:** PUFs must be resistant to physical tampering and attempts at reverse engineering.
- **Challenge-Response Pairs:** PUFs should support a large number of unique challenge-response pairs.
- **Environmental Robustness:** PUF responses should remain stable across varying environmental conditions.
- **Low Latency:** The generation of PUF responses should be fast enough to meet application performance needs.

5.1.4 Anti-Tamper Mechanisms

Anti-tamper mechanisms are vital for detecting and responding to physical attacks or unauthorized attempts to access secure hardware. This section highlights strategies such as physical shielding, intrusion detection sensors, and active measures to protect critical RoT components from tampering.

- **Detection and Response:** Implement mechanisms to detect and respond to physical tampering attempts.
- **Shielding:** Use physical shielding to protect sensitive components from probing and tampering.
- **Active Measures:** Employ active anti-tamper measures such as sensors that detect and respond to intrusion.

5.2 Derived from Hardware Elements

This chapter explores critical elements derived from the foundational hardware components of the Root of Trust. These derived elements play an integral role in ensuring secure device operations and cryptographic functionality.

5.2.1 UID (Unique Identifier)

The Unique Identifier (UID) is a securely derived, immutable identifier unique to each device. This section discusses its importance in ensuring tamper resistance and its role in enabling device-specific authentication and identification.

- **Uniqueness:** Each UID must be globally unique to prevent duplication or impersonation of devices.
- **Immutability:** Once generated and provisioned, should remain unchanged throughout the device's lifetime.
- **Secure Derivation:** The generating process must be secure, preventing prediction, forgery, or reverse-engineering.
- **Tamper Resistance:** Must be stored and protected in a way that resists physical attacks, probing, or extraction attempts.
- **Authentication Support:** Should enable device-specific authentication mechanisms and contribute to secure key derivation.
- **Confidentiality:** Access to the UID should be strictly controlled to prevent unauthorized reading or replication.
- **Environmental Robustness:** Must remain stable and retrievable across all supported operating conditions and device aging effects.
- **Low Latency Access:** Retrieval of the UID should be efficient to meet real-time or performance-critical application needs.

5.2.2 HUK (Hardware Unique Key)

A Hardware Unique Key (HUK) is a cryptographically secure key derived from hardware elements. This section outlines its function in device-specific cryptographic operations, emphasizing its confidentiality and secure generation process.

- **Uniqueness:** Each HUK must be unique to the individual device to ensure cryptographic separation between devices.
- **Secure Derivation:** Should be derived from hardware-based secrets or identifiers in a way that prevents prediction or duplication.
- **Confidentiality:** Must never be exposed in raw form outside secure boundaries and should only be used internally for cryptographic operations.
- **Tamper Resistance:** Storage and usage of the HUK must be protected against physical probing, side-channel attacks, and invasive hardware manipulation.
- **Non-Volatility or Regeneration:** Must be reliably retrievable or regenerable throughout the device lifetime without degradation or loss.
- **Key Usage Control:** Access to the HUK should be strictly controlled, and it should be used only in authorized cryptographic functions such as key derivation, encryption, or authentication.
- **Environmental Robustness:** Must remain stable and consistent across the full operational environmental range and aging effects of the device.

- **Standards Compliance:** The generation and usage processes should comply with relevant cryptographic standards and guidelines.

5.2.3 CSRNG (Cryptographically Secure Random Number Generator)

The Cryptographically Secure Random Number Generator (CSRNG) relies on a validated entropy source to produce high-quality random numbers for cryptographic applications. This section highlights its security standards, integration requirements, and importance in maintaining the robustness of cryptographic systems. Following the work of Peetermans et al. [PRV19], which emphasizes the design of high-throughput and provably secure hardware random number generators, we aim to ensure that the implemented CSRNG meets rigorous statistical quality and resistance against state compromise. We have investigated approaches described in [Che+12] and [Peb+24], both of which present promising architectures for secure entropy harvesting and conditioning. It is, however, not yet clear whether these designs can be fully implemented within the project duration. Therefore, research continues towards realizing an open TRNG that can serve as a robust entropy source for the CSRNG, ensuring compliance with relevant security standards and long-term maintainability in open-source contexts.

- **High-Quality Entropy:** The CSRNG must be seeded with entropy sources meeting statistical randomness requirements.
- **Standard Compliance:** Design should follow established guidelines such as NIST SP 800-90A/B/C.
- **Resilience Against State Compromise:** Incorporate mechanisms to recover security after partial internal state exposure, as discussed in [PRV19].
- **Throughput and Latency:** Ensure sufficient generation speed to meet application performance requirements without compromising randomness quality.
- **Integration with TRNG:** Seamless coupling with a TRNG for continuous entropy refresh and enhanced unpredictability.
- **Tamper Resistance:** Protection against side-channel analysis, fault injection, and manipulation of entropy sources.
- **Long-Term Reliability:** Maintain consistent performance across environmental conditions, device aging, and operational stress.

5.3 Specifics on Open Source RoT Elements

Traditionally, RoT elements contain confidential design elements, and are tamper resistant. Both characteristics make attacks more difficult. An open RoT-element might be easier to attack. Therefore, it will be analyzed if counter-measures are needed in the following fields:

5.3.1 Open Source License

Ensure compliance with open-source licenses that allow for modification, distribution, and usage in both commercial and non-commercial applications. We will provide or use an openly accessible server (e.g. `github.com`) on which the results will be made available.

5.3.2 Design Visibility

It will be investigated whether some parts of the design should be kept confidential, e.g., about the places where keys are stored and about the random number generation. As a potential remedy, parts of the GDS-file could be black-boxed and possibly made available for scrutiny under an NDA.

5.3.3 Open Documentation

Open documentation must be scrutinized to ensure that it does not provide attackers with information that could compromise the security of the RoT elements.

5.3.4 RTL Code Derivation

Assess what information about the design and potential vulnerabilities can be inferred from the RTL (Register Transfer Level) code.

5.3.5 Minimal Set of Views

Determine the minimal set of views and documentation, including RTL code and GDS layout, necessary to make the hardware usable while maintaining security.

5.3.6 Usability with Open-Source Tools

Ensure that the design and documentation are compatible with open-source tools to facilitate wide adoption and community-driven improvements.

By adhering to these specifications, we aim to develop robust and secure open hardware security modules that not only leverage the strengths of open-source collaboration but also prioritize security, transparency, and scalability. Open-source collaboration fosters innovation through community-driven contributions, enabling continuous improvement and rapid identification of vulnerabilities. This approach allows us to harness collective expertise, driving higher standards in both performance and reliability.

At the same time, we remain steadfast in our commitment to ensuring the highest levels of security, which is paramount in today's complex digital landscape. Our hardware modules will undergo rigorous testing, peer reviews, and validation processes to meet and exceed industry standards. By combining the flexibility of open development with stringent security protocols, we are aiming to create a solution that will provide unparalleled trust, integrity, and resilience for various applications, ranging from consumer devices to critical infrastructure.

In doing so, we contribute to a future where security is not compromised for accessibility but instead enhanced through open collaboration, resulting in hardware solutions that are both dependable and adaptable to evolving technological challenges.

6 Requirements and Selection of Development Tools

All tools have to be open-source and version control-friendly. If toolchains overlap, for example during ASIC and FPGA synthesis, the same tools should be chosen such that both toolchains can benefit from each other's experiences and contributions.

6.1 HDL

The integration with Caliptra makes the support of SystemVerilog a requirement. Thus, SystemVerilog should ideally be supported by all tools. If this is not (yet) available, the tools should be structured in a way that allows the addition of new HDL frontends. If neither of these requirements can be fulfilled, SystemVerilog has to be converted to Verilog early in the toolflow as a workaround.

6.2 ASIC Toolchain

The ASIC toolchain should optimally be integrated with the IHP-open-PDK already. If that is not the case, the toolchain must have mature and tested support for custom PDK integrations instead. The toolchain must be able to build full ASICs, including macro prehardening and integration as well as I/O placement and metal filling. Validation tasks such as design rule checking and layout-versus-schematic must be supported. Support for design-for-test, for example the insertion of scan chains, are optional, but would be nice to have.

6.3 Security

A comprehensive approach that integrates secure cryptographic algorithms and a well-defined security architecture is crucial for effectively employing Caliptra in security-sensitive applications. Furthermore, the implementation and potential hardware-targeting attacks must be assessed during the design phase. This comprehensive analysis should encompass both active and passive semi-invasive, as well as invasive attack scenarios. Invasive attacks typically aim to manipulate either a circuit or memory to extract secret keys. In this implementation of the Caliptra framework, we utilise physical unclonable functions to render the key material inaccessible while the core remains offline. Consequently, key extraction becomes more challenging as the key material is not permanently stored in memory; instead, it is generated on-the-fly when the core is powered on.

Non-invasive attacks are within the scope of the Caliptra framework, and non-invasive fault injection attacks will be investigated in the SIGN-HEP project. Fault injection techniques, particularly laser fault injection, are used for the security evaluation of integrated circuits (ICs) due to their ability to induce highly localized temporal and spatial effects. Initially, we will simulate parts of the logic using a fault injection simulator, resulting in a vulnerability map for the simulated logic. Along with assessing the IC's resilience against fault injection, cryptographic circuits will also be hardened.

The Caliptra framework already includes a masked implementation of AES and SHA-512 that can be used to perform side-channel protected encryption or hashing. As both implementations only support first-order masking and cannot be easily changed to higher-order security, the existing implementations in EASIMask will be used as they can be masked at arbitrary orders. This is of course only necessary if higher-order attackers are considered, otherwise, the existing masked implementations are sufficient. Side-channel protection is **not** necessary for ECC routines as they operate on public data that do not need additional protection.

Fault-injection attacks can potentially threaten all cryptographic routines by altering the performed operations, either resulting in an extraction of secret data, or at least in a malfunctioning of the chip. As a consequence, all cryptographic modules benefit from a hardening against fault-injection attacks by employing redundancy and some form of error detection or correction. Notably, for ECC, hardening against fault-injection attacks

is sufficient (as it only works on public data), whereas AES and SHA-512 may need hardening against attacks combining side-channel and fault-injection attacks.

For the automated hardening of cryptographic circuits against fault injection attacks, the tool EASIMask will be used and extended. EASIMask requires the cryptographic circuits to be written in SpinalHDL. If an implementation in SpinalHDL is not available or possible, an additional frontend for EASIMask that parses the circuit into the internal circuit representation would be necessary. To identify potential vulnerable parts of the circuits, a simulator capable of simulating the circuits as well as the injected faults is needed. It is especially important that the simulator is able to simulate larger circuits, as the complexity of hardened circuits significantly increases over unprotected designs. Based on the outputs of the simulator, the vulnerable parts of the design have to be marked for the physical hardening. This might be done directly via annotations in the used SpinalHDL-Code, but also through other data formats (e.g., JSON), that are then read and analyzed by EASIMask.

7 Meeting Security Requirements for Certification

This specification part addresses the security requirements, evaluation methods, and standards that the system must meet to achieve certification. The SIGN-HEP project aims to prepare the system for certification by ensuring security validation of both individual design components and the system as a whole, as well as the tools that SIGN-HEP relies on in its workflow. The following key points form the basis of this process:

- A detailed understanding of the system architecture and design tools.
- Complete and thorough system documentation.
- A robust testing environment.
- Analysis of certification practices, programs, and standards.

7.1 Initial Context

The SIGN-HEP project builds on the Caliptra RoT specification while introducing design changes and utilizing an open-source tooling set to meet project goals.

While the Caliptra specification is designed with certification in mind, our preparation for certification efforts wil need to cover these main deviations from Caliptra:

Design Deviations from Caliptra:

- We pick an upgraded CPU to align with performance, security, and the open-source tools that SIGN-HEP relies on. In our opinion, Veer is not the optimal choice for our needs, as Vexriscv offers an extreme high degree of flexibility in terms of configuration. With the implementation of this CPU in the meta hardware description language SpinalHDL, it is effortless to incorporate caches, buses, or peripheral components with only a few lines of code. With a novel plug-in architecture, Vexriscv facilitates the integration of instruction set extensions with minimal effort. These features are particularly interesting, as new machine instructions for parts of cryptographic algorithms can be easily developed. By translating SpinalHDL into Verilog, hardening measures against side-channel attacks can be introduced automatically. Furthermore, the behavior of the CPU can be simulated with SpinalHDL, i.e. the simulation itself is developed in the same language as the CPU itself. All these

features provide an extremely flexible platform for systematic evaluation and offer optimal possibilities for research and application.

- We plan to harden hardware-accelerated crypto blocks for specific cryptographic operations.
- We will add new cryptographic blocks to support advanced algorithms and protocols.

Tooling Deviations from Caliptra:

- We will utilize open-source tools for simulation, synthesis, and physical implementation.
- Therefore, documentation and verification processes for the open-source tooling used in SIGN-HEP deviate, too.

Target Certification Focus:
Our primary certification target is FIPS 140-3 (e.g., through the Cryptographic Module Validation Program (CMVP)), as it is a industry standard for cryptographic modules recognized in the U.S. but also aligned with German BSI guidelines. At the same time, we aim to provide documents in a format that could be extended as part of a Common Criteria (ISO/IEC 15408) certification effort. Note that while we support a later certification as best as possible, we will not undergo the actual process.

7.2 Framework for Certification

The certification framework is designed to be flexible and modular, allowing the system to support various certification programs, whether they are well-established or more specialized. This flexibility is achieved by covering a range of core security requirements in a standardized way that can be adapted to specific certifications with modest effort. The framework shall provide:

1. **Extendable Modules:** The testing and validation components shall be modular, meaning they can be expanded or reconfigured to meet specific certification needs. For example, the cryptographic validation module shall support testing for selected standards, with the flexibility to add support for additional standards as required, without significant changes.
2. **Certification Program Abstraction Layer:** The framework shall include an abstraction layer that decouples the core security testing from any specific certification program, allowing users to add the requirements of their chosen certification. This layer shall map certification requirements (e.g., from Common Criteria or CMVP) to the corresponding tests.
3. **Automated Evidence Collection and Reporting:** The system shall automate the collection of test logs, validation results, and documentation, which can be structured to meet the specific needs of targeted certification programs. It shall also provide adjustable reporting templates that allow users to customize the format and content of certification reports.
4. **Reusability Across Standards:** The framework shall utilize common testing techniques and validation processes across multiple certification standards, enabling the same core tests to be reused for different certifications, reducing redundancy.

5. **User Customization and Flexibility:** The framework shall allow users to configure security profiles by selecting the security requirements or tests they need. Additionally, the system shall provide an API to create and integrate custom tests or add new certification requirements.

7.3 Path to Certification

- **Delta Documentation:**
  - Detailed documentation of differences between SIGN-HEP and Caliptra.
  - Justifications for design and tooling deviations, ensuring compliance with certification requirements.
  - Impact analysis of deviations on security and performance.
- **Certification Process:**
  - Alignment of SIGN-HEP design and implementation with relevant standards.
  - Comprehensive testing and validation of cryptographic functions.
  - Ensuring end-to-end security from design to deployment.
- **Future-Proofing:**
  - Consider emerging cryptographic standards to ensure long-term compliance and security in design and implementation.
  - The SIGN-HEP block is designed to easily adapt to future technologies and evolving security requirements.
- **Documentation and Evidence:**
  - Compilation of all relevant documentation, including design specifications, test results, and security analyses.
  - Providing evidence of compliance with certification requirements for each cryptographic function.

Glossary

AES Advanced Encryption Standard. 30

API application programming interface. 30

ASIC Application-specific integrated circuit. 13, 30

CPU Central Processing Unit. 14, 30

CSRNG Cryptographically Secure Random Number Generator. 24, 30

ECC Elliptic-curve cryptography. 30

EDA Electronic design automation. 5, 9, 30

FPGA Field-programmable gate array. 13, 30

GDS Graphic Design System. 6, 10, 25, 30

GNU GNU's Not Unix!. 30

HSM Hardware Security Module. 1, 4, 9–12, 30

HUK Hardware Unique Key. 23, 30

IP Intellectual Property. 6, 30

NVM Non-Volatile Memory. 7, 10, 30

OTP One-Time Programmable. 20, 21, 30

PDK Process Design Kit. 1, 6, 9, 10, 30

PUF Physically Unclonable Function. 22, 30

RNG Random Number Generator. 7, 30

RoT Root of Trust. 4, 9, 12, 17, 30

RTL Register-transfer level. 13, 25, 30

sha Secure Hash Algorithms. 30

SoC system on a chip. 12–14, 30

SPI Serial Peripheral Interface. 12–14, 30

TRNG True-Random-Number Generator. 10, 24, 30

UID Unique Identifier. 23, 30

References

[Che+12] Zouha Cherif et al. "An Easy-to-Design PUF Based on a Single Oscillator: The Loop PUF". In: *2012 15th Euromicro Conference on Digital System Design*. 2012, pp. 156–162. DOI: 10.1109/DSD.2012.22.

[Hen+24] Tim Henkes et al. "Evaluating an Open-Source Hardware Approach from HDL to GDS for a Security Chip Design — a Review of the Final Stage of Project HEP". In: *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2024, pp. 1–6. DOI: 10.23919/DATE58400.2024.10546500.

[Kou+20] Michael Kounavis et al. "The MAGIC Mode for Simultaneously Supporting Encryption, Message Authentication and Error Correction". In: *Cryptology ePrint Archive* (2020).

[Lam+24] Lukas Lamster et al. "Voodoo: Memory Tagging, Authenticated Encryption, and Error Correction through {MAGIC}". In: *33rd USENIX Security Symposium (USENIX Security 24)*. 2024, pp. 7159–7176.

[Ngo+17] Xuan Thuy Ngo et al. "Cryptographically Secure Shield for Security IPs Protection". In: *IEEE Transactions on Computers* 66.2 (2017), pp. 354–360. DOI: 10.1109/TC.2016.2584041.

[Peb+24] Florian Pebay-Peyroula et al. "OpenTRNG: an open-source initiative for ring-oscillator based TRNGs". In: *2024 IEEE International Conference on Design, Test and Technology of Integrated Systems (DTTIS)*. 2024, pp. 1–6. DOI: 10.1109/DTTIS62212.2024.10780212.

[Pro22] OPEN Compute Project. *Caliptra: A Datacenter System on a Chip (SOC) Root of Trust (RoT)*. June 2022. URL: https://www.opencompute.org/documents/caliptra-silicon-rot-services-09012022-pdf (visited on 02/23/2025).

[PRV19] Adriaan Peetermans, Vladimir Rozic, and Ingrid Verbauwhede. "A Highly-Portable True Random Number Generator Based on Coherent Sampling". In: *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. 2019, pp. 218–224. DOI: 10.1109/FPL.2019.00041.

[SF07] Dan Shumow and Niels Ferguson. *On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng*. 2007. URL: https://rump2007.cr.yp.to/15-shumow.pdf.