OPEN
Compute Project

Caliptra: A Datacenter System on a Chip (SOC) Root of Trust (RoT)

Revision 1.0

Version 0.5

July 2022

CONTRIBUTORS:

Bryan Kelly (Microsoft)

Andrés Lagar-Cavilla (Google)

Jeff Andersen (Google)

Prabhu Jayana (AMD)

Piotr Kwidzinski (AMD)

Rob Strong (AMD)

John Traver (AMD)

Louis Ferraro (AMD)

Ishwar Agarwal (Microsoft)

Anjana Parthasarathy (Microsoft)

Bharat Pillilli (Microsoft)

Vishal Soni (Microsoft)

Marius Schilder (Google)

Sudhir Mathane (AMD)

Nathan Nadarajah (AMD)

Kor Nielsen (Google)

Revision Table

Date	Revision #	Author	Description
February 2022	0.2	Prabhu Jayana (AMD) Bryan Kelly (Microsoft) Piotr Kwidzinski (AMD) Andrés Lagar-Cavilla (Google) Jeff Andersen (Google)	Initial proposal draft
March 2022	0.4	Rob Strong (AMD) Piotr Kwindzinski (AMD) Prabhu Jayana (AMD)	- Migrated to OCP template - edits to clarify language use and added sections related LifeCycle support, fuse, crypto requirements, Kat support, etc.
April 12, 2022	0.5	Rob Strong (AMD)	Various edits and formatting modifications to get to v.5
June 2022	0.51	Rob Strong (AMD)	Updated the FW Signing/Verification Algorithms section - added references to OCP Secure Boot specification.

			Updated Physical Attack Countermeasures - updated the section to reference NiST paper ([14]) that discusses SCA as well as their countermeasures.
June 2022	0.52	Nathan Nadarajah (AMD) Sudhir Mathane (AMD)	Added Threat Model section and initial content (work-in-progress)
July 2022	0.53	Louis Ferraro (AMD)	Added Device Resilience chapter and updated related text.
August 2022	0.54	Piotr Kwidzinski (AMD)	Updated with OCP template and feedback. Added License and Appendix sections.

Table of Contents

License

Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in Open Web Foundation Modified Contributor License Agreement (“OWF CLA 1.0”) (“Contribution License”) by:

AMD
Google
Microsoft

Usage of this Specification is governed by the terms and conditions set forth in **Open Web Foundation Modified Final Specification Agreement (“OWFa 1.0”) (“Specification License”).**

You can review the applicable OWFa1.0 Specification License(s) referenced above by the contributors to this Specification on the OCP website at http://www.opencompute.org/participate/legal-documents/. For actual executed copies of either agreement, please contact OCP directly.

Notes:

1) The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING

NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Acknowledgements

The Contributors of this Specification would like to acknowledge the following companies for their feedback:

Compliance with OCP Tenets

Please describe how this Specification complies to the following OCP tenets. Compliance is required for at least three of the four tenets. The ideals behind open sourcing stipulate that everyone benefits when we share and work together. Any open source project is designed to promote sharing of design elements with peers and to help them understand and adopt those contributions. There is no purpose in sharing if all parties aren't aligned with that philosophy. The IC will look beyond the contribution for evidence that the contributor is aligned with this philosophy. The contributor actions, past and present, are evidence of alignment and conviction to all the tenets.

Openness

The Caliptra source for RTL and firmware will be licensed using the Apache 2.0 license. The specific mechanics and hosting of the code are work in progress due to CHIPS alliance timelines. Future versions of this spec will point to the relevant resources.

Efficiency

Caliptra is used during the boot sequence and upgrade cycles, so it cannot yield a measurable impact on system efficiency.

Impact

Caliptra brings consistency and transparency to a foundational area of security and confidential compute. Open source in-package RoTs have not been attempted before in the industry. It is challenging to align partners, and it is challenging to align a common core of functionality everyone agrees upon in this space. Caliptra breaches multiple traditional blockers and creates new ground for the industry and for open ecosystems.

Scale

Caliptra is a committed intercept for Google and Microsoft first party Cloud silicon. It is also a committed intercept for AMD server silicon products. This scale covers both a significant portion of the Cloud market as well as one of the two key CPU vendors in hyperscale and enterprise.

Scope

This document defines technical specifications for a Caliptra RTM[1] used in Open Compute Project. This document, along with the [baseline specification] shall comprise product's technical specification.

Overview

This document provides definitions and requirements for a Caliptra RTM. The document then relates these definitions to existing technologies, enabling third device and platform vendors to better understand those technologies in trusted computing terms.

Acronyms and Abbreviations

For the purposes of this document, the following abbreviations apply:

Abbreviation	**Description**

[1] *Caliptra. Spanish for "root cap" and describes the deepest part of the root*

BMC	Baseboard Management Controller
CA	Certificate Authority
CDI	Composite Device Identifier
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CSP	Critical Security Parameter
DICE	Device Identifier Composition Engine
DME	Device Manufacturer Endorsement
DRBG	Deterministic Random Bit Generator
ECDSA	Elliptic Curve Digital Signature Algorithm
FMC	FW First Mutable Code
GPU	Graphics Processing Unit
IDevId	Initial Device Identifier
iRoT	Internal RoT
KAT	Known Answer Test
LDevId	Locally Significant Device Identifier
MCTP	Management Component Transport Protocol
NIC	Network Interface Card
NIST	National Institute of Standards and technology
OCP	Open Compute Project
OTP	One-time programmable
PKI	Public Key infrastructure
PUF	Physically unclonable function

RoT	Root of Trust
RTI	RoT for Identity
RTM	RoT for Measurement
RTR	RoT for Reporting
SoC	System on Chip
SPDM	Security Protocol and Data Model
SSD	Solid State Drive
TCB	Trusted Computing Base
TCI	TCB Component Identifier
TCG	Trusted Computing Group
TEE	Trusted Execution Environment
TRNG	True Random Number Generator
UECC	Uncorrectable Error Correction Code

Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

References

[1] NIST Special Publication 800-193 Platform Firmware Resiliency Guidelines

[2] IEEE Standard for Local and Metropolitan Area Networks Secure Device Identity

[3] DMTF Security Protocol and Data Model (SPDM) Specification (DSP0274)

[4] OCP Security WG: Ownership Transfer

[5] Global Platform Technology Root of Trust Definitions and Requirements Version 1.1 Public Release June 2018

[6] TCG DICE Layering Architecture Version 1.0 Revision 0.19 July 23, 2020

[7] NIST Special Publication 800-56A Revision 3 Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography

[8] NIST Special Publication 800-108 Recommendation for Key Derivation Using Pseudorandom Functions

[9] NIST Special Publication 800-208 Recommendation for Stateful Hash-Based Signature Schemes

[10] Ownership and Control of Firmware in Open Compute Project Devices (Open Compute Project, Security WG)

[11] Open Compute Project Secure Boot Specification

[12] TCG DICE Attestation Architecture Version 1.00 Revision 0.23 March 1, 2021

[13] TCG Hardware Requirements for a Device Identifier Composition Engine Family “2.0” Level 00 Revision 78 March 22, 2018

[14] Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing

[15] Attestation V1.0 White Paper

Theory of Operation

Establishing a core root of trust along with a chain of trust that attests to the integrity of configuration and mutable code is fundamental to the overall security posture of silicon devices.

Traditional RoT architectures have offered a multitude of intrinsic security services and hosted security applications on a trusted execution environment (TEE) that consist of (but not limited to) hardware capabilities (cryptographic and microprocessor), ROM, Firmware & API infrastructure. These solutions have been instantiated in discrete or integrated forms in various platform & component architectures.

Some of these solutions are either proprietary or aligned to specific parts of an industry standards/consortium/association specifications (e.g.,National Institutes of Standards and technology (NIST), Open Compute Project (OCP), Trusted Computing Group (TCG), Distributed Management Task Force (DMTF), Institute of Electrical and Electronics Engineers (IEEE), etc.) and may be certified to various conformance standards (e.g., NIST cryptographic algorithm Validation program (CAVP), etc.).

Establishing a consistent root of trust on very different hardware configurations while maintaining configuration and deployment flexibility is challenging. There is no uniform configuration across Cloud Service Providers. Example: A system with host processors, has very different firmware security measures when compared to systems without head-nodes or host processors.

The OCP Security WG specifications are making progress towards establishing the platform and peripheral security architecture recommendations necessary to attain the desired consistency in platform security orchestration.

The objective of this specification is to define core RoT capabilities that must be implemented in the SoC or ASIC of any device in a cloud platform. The collection of these RoT capabilities is referred to as the ***Silicon RoT Services (Silicon RoT).***

Silicon RoT Goals

The scope of a Caliptra Silicon RoT is deliberately minimalistic in nature to drive agility of specification definition, to maximize applicability, and to drive industry alignment, consistency and faster adoption of foundational device security primitives. A well and narrowly defined specification maximizes architectural composability, reusability across CSPs, products and vendors, and feasibility of open sourcing.

Enhancements, advanced use cases & applications are outside the scope of this specification and may be developed in the form of a roadmap for the Silicon RoT and community engagement.

Caliptra defines a design standard for a Silicon internal RoT baseline. The standard satisfies a Root of Trust for Measurement (RTM) role. The open-source implementation of Caliptra drives transparency into the RTM and measurement mechanism that anchors hardware attestation. The Caliptra Silicon RoT must boot the SoC, measure the mutable code it loads, and measure and control mutation of non-volatile configuration bits in the SoC. The Caliptra Silicon RoT reports these measurements with signed attestations rooted in unique per-asset cryptographic entropy. As such, the Caliptra Silicon RoT serves as a Root of Trust for Identity for the SoC.

No other capabilities are part of this specification, to satisfy the criteria for success outlined above, and to decouple platform integrity capabilities that can be enforced and evolve independently via other platform devices or services – such as Update, Protection and Recovery.

Within this scope, the goals for a Caliptra 1.0 specification include:

- Definition and design of the standard silicon internal RoT baseline:
    - Reference functional specification:
        - Scope including RTM and RTI capabilities
        - Control over SoC non-volatile state, including per asset entropy
    - Reference APIs:
        - Attestation APIs
        - Internal SoC services
    - Reference implementation
    - Open Source Reference (including RTL and firmware reference code):
        - For implementation consistency, leverage open source dynamics to avoid pitfalls and common mistakes
        - For accelerated adoption (e.g., so that future products can leverage existing designs and avoid having to start the design process from scratch)
        - For greater transparency, to avoid fragmentation in the implementation space
    - Firmware and RTL logical design are open, managed by consortium.
- Consistency - across the industry in the internal RoT (iRoT) architecture and

implementation

  - DICE Identity, Measurement & Recovery
- The silicon iRoT scope includes all datacenter-focused server class SoC / ASIC (datacenter focused) devices (SSD - DC, NIC, CPU, GPU - DC):
  - Critical priority are devices with the ability to handle user plain text data
    - Top priority are CPU SoCs
    - Other examples include SmartNIC and accelerators
  - Over time scope includes further data center devices
    - SSD, HDD, BMC, DIMM

Explicitly out of scope is how silicon integration into backend work is performed such as:

- Foundry IP integration
- Physical design countermeasures
- Analog IPs
- Post manufacture test and initialization (OSAT)
- Certification

Use Cases

The Silicon RoT use cases can be supported through the adoption of specific industry standards and association/consortium specifications. Refer to *Industry Standards and Association Consortium Specifications*.

In this version, Caliptra Silicon RoT desired capabilities address the basics of supply chain security use cases.

Supply Chain Security

- **Mutable Code Integrity:** The objective here is to prove the device is running genuine firmware that the device manufacturer can vouch for its authenticity & integrity, and the device owner can ensure only authorized updates are applied to the device. This flow is aligned with [Reference **4**] and can be achieved with dual signature verification of equal imposition.
- **Configuration & Lifecycle Management**: allow the platform owner to securely configure the RoT capabilities, and enable/authorize lifecycle state transitions of the SoC.

DICE-as-a-Service

A Caliptra RTM exposes a "DICE-as-a-Service" API, allowing Caliptra to derive and wield a DICE identity on behalf of other elements within the SoC. Use-cases for this API includes serving as a signing oracle for an SPDM responder executing in the SoC Application Processor.

Industry Standards and Association / Consortium Specifications

This specification follows the industry standards and specifications listed in References.

NIST SP800-193 Platform Firmware Resiliency

Per [Reference **1**], RoT subsystems are required to fulfill three principles: *Protection, Detection* and *Recovery*. The associated RoT services are referred to as:

- **The Root of Trust for Update (RTU)** is responsible for authenticating firmware updates and critical data changes to support platform protection capabilities.
- **The Root of Trust for Detection (RTD)** is responsible for firmware and critical data corruption detection capabilities.
- **The Root of Trust for Recovery (RTRec)** is responsible for recovery of firmware and critical data when corruption is detected, or when instructed by an administrator.

These RoT services can be hosted by a complex RoT as a whole or it can be spread across one or more components within a platform. This determination has a basis in physical risk.

Physical adversaries with reasonable skill can bypass a discrete RoT's detection capabilities, for example, with SPI interposers.

However, an RoT embedded within an SoC or ASIC represents a much higher detection bar for a physical adversary to defeat. For this reason, a Caliptra Silicon RoT shall deliver the **Detection** (or Measurement) capability.

With the objectives of minimalistic scope for Silicon RoT and maximizing applicability and adoption of this specification, **Update** and **Recovery** are decoupled from Caliptra and are expected to be provided by an external RoT subsystem such as a discrete RoT board element on a datacenter platform. Because a physical adversary can trivially nullify any Recovery or Update capabilities, no matter where implemented, decoupling represents no regression in a security posture, while enabling simplicity and applicability for the internal SoC silicon RoT.

Detection of corrupted critical code & data (configuration) requires strong end to end cryptographic integrity verification. To meet the RTD requirements, Silicon RoT shall:

- Cryptographically measure its code & configuration
- Sign these measurements with a unique attestation key
- Report measurements to a host and/or external entity, which can further verify the authenticity & integrity of the device (a.k.a Attestation)

Measurements include **Code** and **Configuration**. Configuration includes invasive capabilities that impact the user SLA on Confidentiality -- for example, the enablement of debug capabilities that grant an operator access to raw, unencrypted registers for any tenant context. In order to measure and attest Configuration, the Silicon RoT must be in control of the Configuration.

As an extension to controlling Configuration, the Silicon RoT must control the security states (refer to *Caliptra Security States*). Certain security states by design grant full invasive capabilities to an external operator, for debug or field analysis.

Measurements must be uniquely bound to the device & its manufacturer to a minimum. This establishes the need for **Identity** services in the Silicon RoT, that serves as the basis for key derivation and attestation authenticity.

For further details about how Caliptra addresses NIST SP800-193, refer to Device Resilience.

Trusted Computing Group (TCG) DICE Attestation

In accordance with OCP Attestation specification, devices must have a Cryptographic Identity for the endorsement of attestation quotes. The RTM implementation follows TCG DICE (refer to [Reference 6], [Reference 12] and [Reference 13]). One of the benefits of TCG DICE device identities is having renewable security. This renewability complements ownership transfer and circular economy. The new owner is not burdened with the identity of the previous owner, nor is the new owner burdened with trusting an irrevocable hardware identity certificate. This benefits the transferee, as their identities can be revoked through standard PKI mechanisms. DICE based certificates are fully compatible with Public Key Infrastructure (PKI), including full life cycle management and PKI Certificate Revocation List (CRL).

Operational security during the manufacture process is critical, to ensure the DICE entropy is securely initialized, certified, and registered, avoiding any pilfering of this asset by eavesdroppers. Operational security is outside the scope of this specification.

Threat Model

This section describes the Caliptra RoT threat model in terms of profile of the attacks and of the attackers that the Caliptra RoT is expected to defend against.

Threat model as described here takes into account attacker profile, assets and attack surfaces or paths to these assets based on attacker profiles. Following sections delve into each of these topics.

Threat scenarios as comprehended by assets & possible attack paths are as complete as possible but focus on the worst case scenarios. Thus not every attack path to asset is captured in this threat model.

Attacker Profiles

Attacker profile is the outcome of following factors like tools accessible to the attacker, level of access to the target of evaluation & expertise of the attacker to use these methods. Next level of details of these capabilities scoped for this discussion are as follows.

Tools Accessible to Attacker

Attack Tools	Type of Attack	Purpose/Usage
● Debuggers ● Fuzzing devices ● Image reverse engineering tools ● Software payloads	Logical Fault Injection	● Find logical and functional vulnerabilities & exploit those to achieve unauthorized operations
● Clock fault injectors ● Voltage fault injectors ● Electromagnetic fault injectors ● Optical fault injectors ● Micro probing	Environmental Fault Injection	● Alter execution flow of the critical decision points, especially in the early execution
● Power analyzers ● Timing analyzers (Scopes etc) ● Low speed bus analyzers ● Optical emission analyzers	Side Channel Analysis	● Infer security sensitive information by analyzing various operational conditions

● Microscopic imaging ● Reverse engineering ● Scanning electron microscope imaging Focussed ion beam (FIB)	Chip Invasive Attacks	● Decapsulation, Depackaging, rebonding to probe internals of the chip

Level of Access to TOE

Type of Access	Levels of Access	Attack Paths Available
Physical Access	Unrestricted access for physical and logical attacks	● Chip invasive ● Chip non invasive
Remote Access	Limited access for attacks with both privileged & unprivileged access rights	● Chip non invasive attacks ● Network attacks

Definition of Expertise* (JIL)

Proficiency level	Definition	Detailed definition
Expert	Can use chip invasive, fault injections, side channel & logical tools Understands HW & SW in depth Familiar with implemented ● Algorithms ● Protocols ● HW structures ● Principle and security concepts	● Familiar with developers knowledge namely algorithms, protocols, hardware structure, principles ● Techniques and tools for attacks
Proficient	Can use fault injections, side channel & logical tools Understands HW & SW in reasonably Familiar with security behavior	Familiar with • security behavior, classical attacks

Laymen	No particular expertise	No particular expertise

Types of Attacks

Physical Attacks

A physical attacker has full access to the electrical and physical components and interfaces/connectors/ports of the SoC/ASIC in which the Caliptra RoT is integrated without restriction.

Invasive attacks involving depackaging/delayering of the SoC/ASIC is out-of-scope.

Non-Invasive attacks are in-scope.

- Fault Injection attacks
    - *Counter measurements - as strong recommendation*
- Power and Electromagnetic analysis attacks
    - *Counter measurements - as strong recommendation*

Attack	Description	Threat Model Scope
Electromagnetic – Passive	Attacker observes the electromagnetic power spectrum and signals radiated from the product.	Includes all attacks at all frequency ranges, including radio frequencies, infrared, optical, and ultraviolet. Excludes attacks requiring removing the package lid.

Electromagnetic – Active	Attacker directs electromagnetic radiation at the product or portions of the product.	Includes all attacks at all frequency ranges, including radio frequencies, infrared, optical, and ultraviolet. Excludes attacks requiring removing the package lid.
Electric – Passive	Attacker probes the external pins of the package and observes electrical signals and characteristics including capacitance, current, and voltage signal.	Includes both analog attacks and digital signal attacks. Excludes attacks requiring removing the package lid.
Electric – Active	Attacker alters the electrical signal or characteristics of external pins.	Includes both analog attacks and digital signal attacks. Excludes attacks requiring removing the package lid.
Temperature – Passive	Attacker observes the temperature of the product or portions of the product.	Excludes attacks requiring removing the package lid.

Temperature – Active	Attacker applies external heat sources or sinks to alter the temperature of the product, possibly in a rapid fashion.	Includes all temperature ranges (e.g. pouring liquid nitrogen over the package or heating the package to above 100C) Excludes attacks requiring removing the package lid.
Sound - Passive	Attacker observes the sounds emitted by the product.	Includes all frequencies. Excludes attacks requiring removing the package lid.

Table 3: Attacks and Threats

Logical Attacks

Attack	Description	Included / Excluded
Debug/Register Interfaces	Manipulation of externally accessible registers of the Caliptra RoT	Includes all buses accessible to components external to Caliptra RoT including JTAG and SMN.

Software Interfaces	Attacker invokes software interfaces exposed by the Caliptra RoT to external components.	Includes all externally exposed software interfaces from both non-RoT firmware as well as interfaces accessed by external IP blocks. Includes exploiting both design and implementation flaws. For High Value Assets only (see next subsection), the attacker is assumed to fully control all mutable code of the SoC, including privileged Caliptra RoT mutable code.
Side channel - Timing	Attacker observes the elapsed time of different sensitive operations.	Includes attacks where the attacker actively stimulates the sensitive operations while timing.

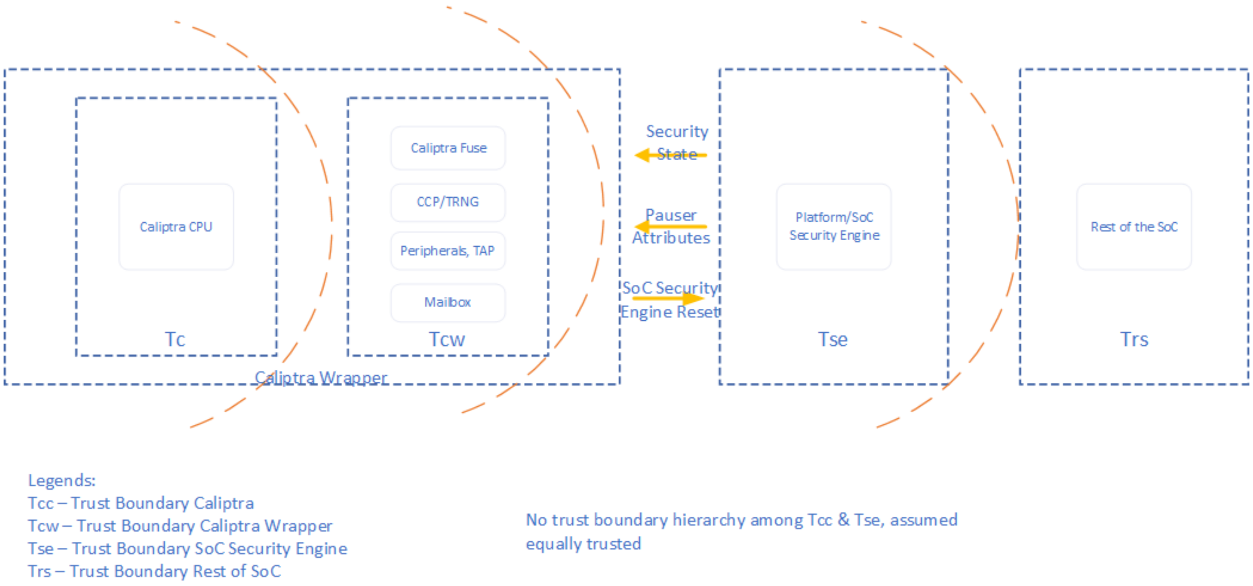
Cryptographic Analysis	Attacker observes plaintext, ciphertext, related data, or immediate values in cryptography to overcome cryptographic controls	Includes all practical cryptanalysis attacks. Assumes NIST-unapproved algorithms provide no security. (e.g. SHA-1, Single DES, ChaCha20) Assumes any cryptographic algorithm that provides less than 128 bits of security (as determined by NIST SP 800-57) provides no security. Excludes quantum computer attacks. This exclusion will be removed soon.

Table 4: Logical Attacks

Trust Boundaries

Following diagram establishes trust boundaries for the discussion of threat modeling. Caliptra based SoCs are expected to have Caliptra as silicon RoT, platform or SoC security engine to orchestrate SoC security needs & rest of the SoC.

Trust levels of Caliptra and the SoC security engine are not hierarchical. These two entities are responsible for different security aspects of the chip.


Caliptra CPU
Tc
Caliptra Fuse
CCP/TRNG
Peripherals, TAP
Mailbox
Tcw
Caliptra Wrapper
Security State
Pauser Attributes
SoC Security Engine Reset
Platform/SoC Security Engine
Tse
Rest of the SoC
Trs
Legends:
Tcc – Trust Boundary Caliptra
Tcw – Trust Boundary Caliptra Wrapper
Tse – Trust Boundary SoC Security Engine
Trs – Trust Boundary Rest of SoC
No trust boundary hierarchy among Tcc & Tse, assumed equally trusted

Caliptra Assets & Threats

Assets are defined to be secrets or abilities that must be protected by an owner or user of the asset. Ownership means that the owner has the responsibility to protect these assets and must only make them available based on a defined mechanism while protecting all other assets. An example of when an owner must protect assets would be moving from secure mode to unsecure. Another example would be moving from one owner to another. Before moving through these transitions, the owner will need to ensure all assets are removed, disabled or protected based on use-case definition.

Asset Category	Asset	Security Property	Attacker Profile	Attack Path	Mitigation
Fuse/OTP high value secrets	UDS Seed	Confidentiality and Integrity	Expert	Malicious manufacturing spoofing on UDS seeds	UDS obfuscation/encryption with class RTL key
				Invasive attack (fuse analysis)	Shield fuse IP
				Boot path tampering while retreving UDS values	UDS obfuscation/encryption with class RTL key
			Expert	Attempting to derive die specific keys by knowing UDS, KDF	Confine unobfuscated UDS & subsequent derivations to key valut
	Field Entropy	Confidentiality and Integrity	Expert	Malicious manufacturing spoofing on field entropy	Field entropy obfuscation/encryption with class RTL key
				Invasive attack (fuse analysis)	Shield fuse IP
				Boot path tampering while retreving field entropy values	Field entropy obfuscation/encryption with class RTL key
			Expert	Attempting to derive die specific keys by knowing field entropy, KDF	Confine field entropy & subsequent derivations to key valut
	FW authentication keys	Integrity	Proficient	Glitching	1. Redundant decision making on critical code execution

					2. Error check before consuming values from fuse 3. Environmental monitoring & protection
	Versioning information from fuses	Integrity	Proficient	Glitching	Environmental monitoring & protection
	IDEVID CERT chain	Integrity	Proficient	Glitching	1. Environmental monitoring & protection 2. Error check before consuming values from fuse various ways
Die unique assets	UDS (802.1AR Unique Device Secret)	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key valult, not readable by SW 2. SCA protections
	DICE UDS (DICE Unique Device Secret)	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key valult, not readable by SW 2. SCA protections
	CDI0 (DICE component device identifier for Layer 0)	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key valult, not readable by SW 2. SCA protections
	CDI1-n ((DICE component device	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel	1. Secrets locked in key valult, not readable by SW 2. SCA protections

	identifier for Layer x)			attack to infer secret	
	IDevID_Priv	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key vault, not readable by SW 2. SCA protections
	LDevID_Priv	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key vault, not readable by SW 2. SCA protections
	Obfuscation Key	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key vault, not readable by SW 2. SCA protections
	Alias_Key_Priv	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key vault, not readable by SW 2. SCA protections
	Alias_Key_Priv	Confidentiality and Integrity	Proficient	1. Software reading actual secrets 2. Side channel attack to infer secret	1. Secrets locked in key vault, not readable by SW 2. SCA protections
Root of trust execution	ROM FW	Integrity	Proficient	Glitching	1. Redundant decision making on critical code execution 2. Environmental monitoring &

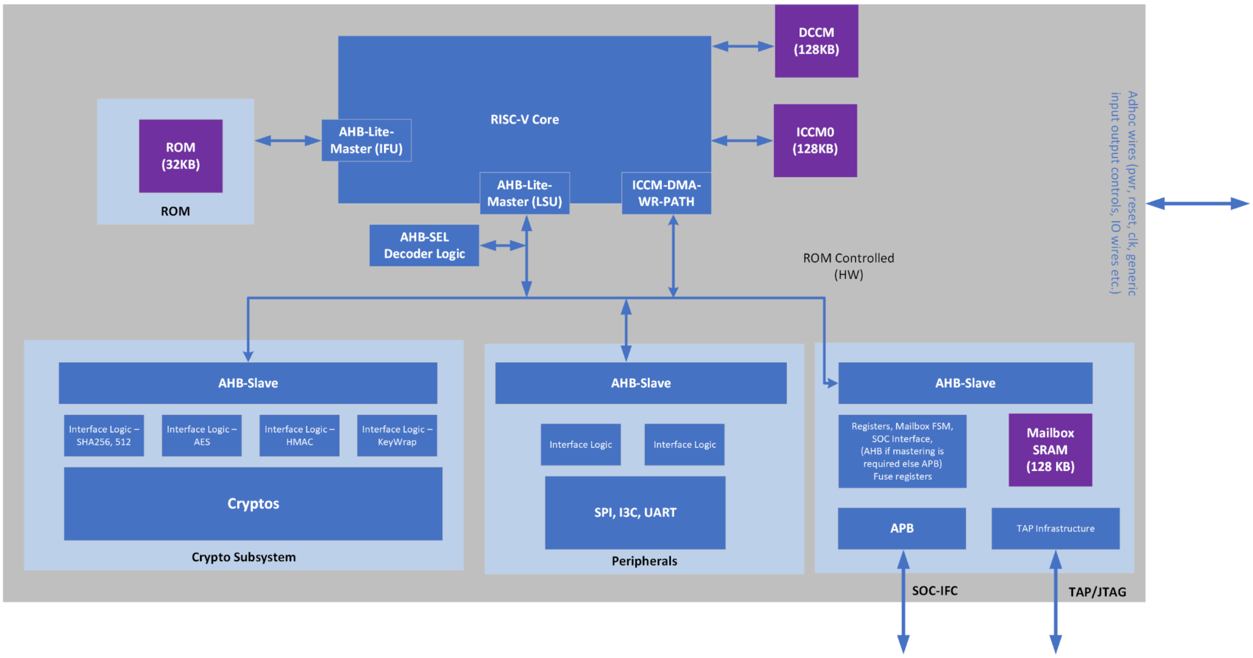
					protection
	Execution of unauthorized runtime FW	Authenticity & Integrity	Proficient	Modify boot media	Authenticity & integrity check using PKI DSA upon power on
	Execution of unauthorized runtime FW	Authenticity & Integrity	Proficient	Arbitrary payload pushed into execution	Authenticity & integrity check using PKI DSA during software updates & power on
	Rollback Attack	Versioning	Proficient	1. Modify boot media to host older versions 2. Bypass version check during boot	1. Authenticity & integrity check using PKI DSA upon power on 2. Failproof, audited boot code implementation responsible to load images
	Control flow	Integrity & Confidentiality if applicable	Proficient	1. Return & jump addresses manipulation 2. Return values, errors tampering 3. Stack overflow 4. Buffer overflows 5. Privilege escalations & highjacking 6. etc/tbd	Various control flow integrity measures Secure coding practices and auditing implementation
Boot measurements protected by Caliptra	Boot Measurements that Caliptra as RTM gathers, stores and reports	Integrity	Expert	1. Manipulate measurements AiTM while in transit to Caliptra 2. SoC sends manipulated measurements to Caliptra	

Caliptra inputs	Security state	Integrity	Proficient	Glitching	Environmental monitoring & protection
	Mode selection (active, passive selections)	Integrity	Proficient	Glitching	Environmental monitoring & protection
	Pauser Attribute	Integrity	Proficient	Glitching	Environmental monitoring & protection
	JTAG debug	Integrity	Proficient	1. Attempt to manipulate RoT execution via JTAG to non POR flows 2. Attempt to retrieve device secrets via JTAG when product is field-deployed 3. Attempt to retrieve device secrets via JTAG when product is under development/debug	Implement security mode management within Caliptra

Table 2: Assets

High Level Architecture

A Caliptra RTM subsystem has the following basic, high-level blocks:



See details in HW Section.

Caliptra Profiles

Caliptra supports two modes of integration with different security postures and FW loading flows. The first is Active Profile (AP) where Caliptra loads its FW directly from persistent storage (eg. SPI) while in Passive Profile (PP) Caliptra FW is being pushed into mailbox SRAM buffer by SoC immutable code (HW or ROM) controlling persistent storage.

Active Profile

When Caliptra is integrated into an SOC in AP mode, Caliptra RTM is the first uncore microcontroller taken out of reset with direct access to persistent storage. The flow for boot is as follows:

1. Hardware executes SOC power-on reset logic.
2. Caliptra ROM executes first and performs cryptographic identity generation, reads in Caliptra firmware from flash.
3. Caliptra ROM measures and verifies its firmware before loading/executing it. Refer to Error Reporting and Handling for details regarding FMC verification failures.

4. After loading its own firmware, Caliptra copies the SOC First Mutable Code (FMC) into an SOC internal SRAM mailbox buffer and measures that firmware.
5. At this point, Caliptra may signal to SOC ROM and SOC uncore to continue power-on reset as shown in Figure 1.

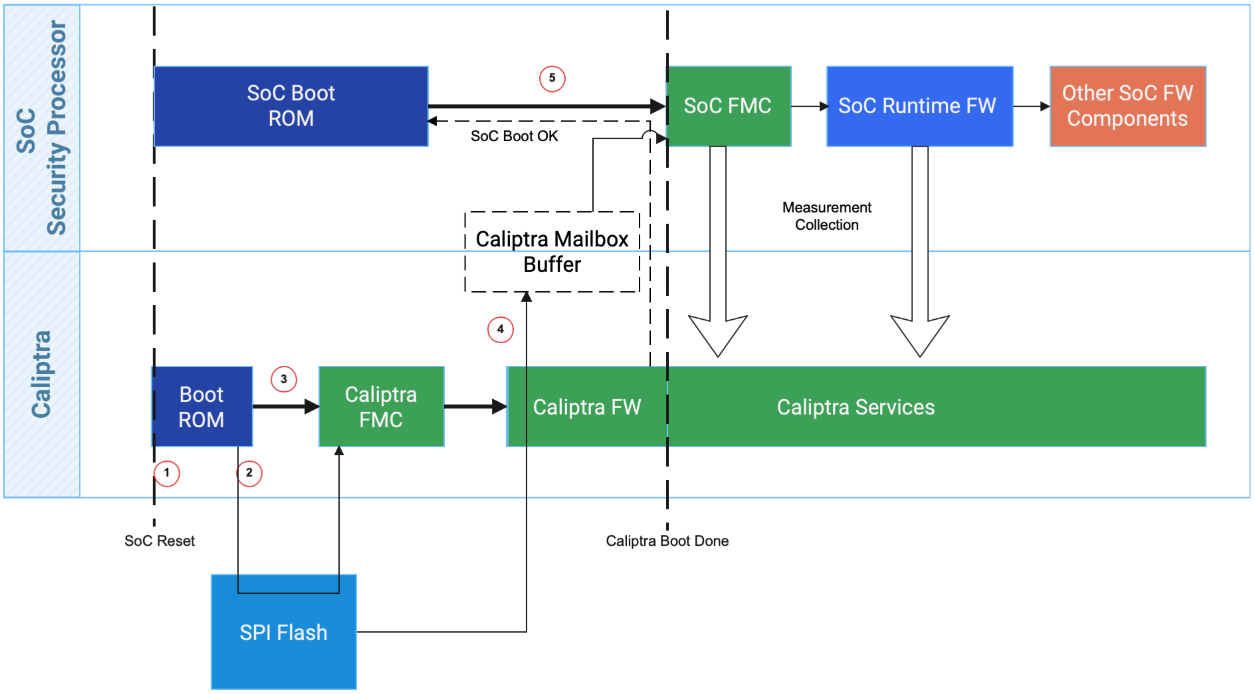


Figure 1: AP Boot Flow

In the AP profile, the Caliptra trusted computing base (TCB) for integrity of Core Root of Trust measurement is the Caliptra security controller and ROM. The verification of measurement includes:

1. The SOC design ingests firmware through Caliptra
2. Caliptra IP, Caliptra ROM, and Caliptra Firmware

Passive Profile

To facilitate ease of integration and reliable measurement, the Caliptra RTM is the first uncore controller to be taken out of reset by the SoC ROM. Once loaded, it provides callback signals for the remaining SoC subsystem to resume normal reset flow.

All firmware that is loaded from an outside entity (and subsequently executed on the microprocessor) shall be considered untrusted; this firmware's measurements shall be reported to the Caliptra RTM before it is allowed to run within the SoC.

In the PP profile, the Caliptra trusted computing base (TCB) for integrity of Core Root of Trust measurement is the Caliptra security controller and SoC ROM. The verification of measurement mechanism includes:

1. Hardware executes SOC power-on reset logic.
2. SOC ROM executes, reads Caliptra firmware into Mailbox SRAM buffer.
3. SOC ROM signals Caliptra for its ROM to execute
4. SOC ROM pauses and waits for a signal (resume) from Caliptra
5. Caliptra ROM cryptographically authenticates its FW, measures and (if valid) executes its FW and then derives cryptographic identities
6. Caliptra signals back to SOC to resume reset.
7. SOC reads in its firmware, cryptographically authenticates its FW and provides measurements to Caliptra before executing.

Refer to Error Reporting and Handling for details regarding Caliptra and SoC firmware load and verification error handling.

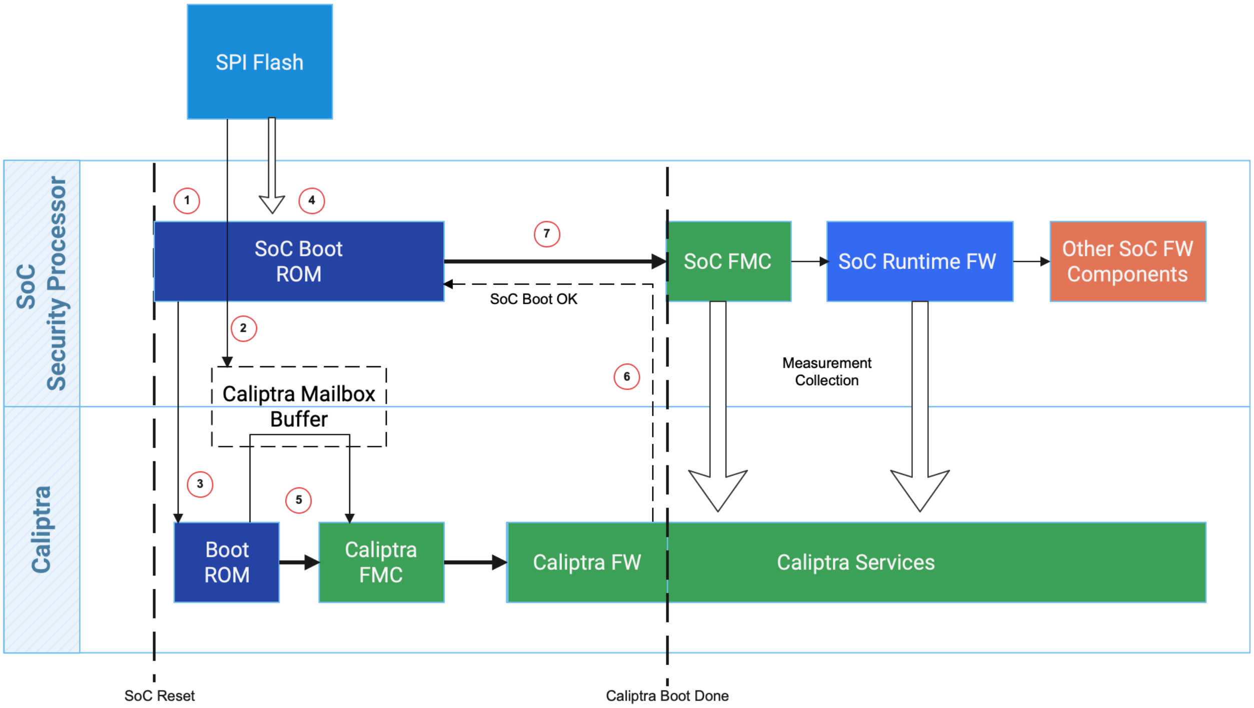


Figure 2: PP Boot Flow

The PP profile is less intrusive to integrations, but extends the TCB for Caliptra to include SOC ROM. The verification of measurement mechanism integration includes:

1. The SOC design that executes SOC power-on reset logic.
2. SOC ROM, SOC boot controller
3. Caliptra IP, Caliptra ROM, and Caliptra Firmware.

4. SOC first mutable code.

The trusted computing base for the SOC is larger in PP, but simplifies integration while preserving many of the Caliptra security guarantees.

Identity

A Caliptra RTM must provide its runtime code with a cryptographic identity in accordance with the TCG DICE specification. This identity must be rooted in ROM, and provides an attestation over the security state of the RTM as well as the code that the RTM booted.

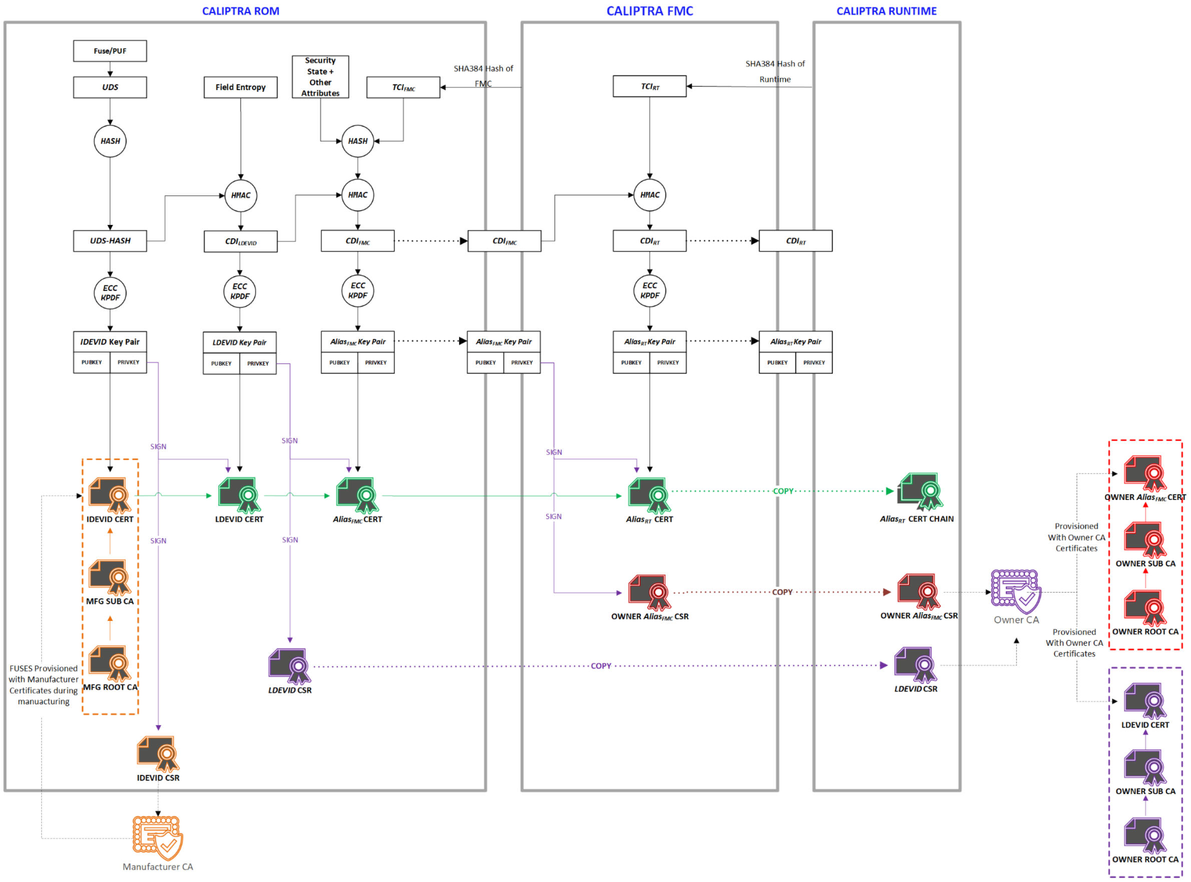


Figure 3: DICE Cert/Key Generation

UDS

A combination of mask ROM and HW macros must implement the DICE key derivation and power-on latch, hiding the UDS and making only the CDI-derived signing key visible to firmware.

The Caliptra UDS is stored in fuses, and is encrypted at rest by an obfuscation secret[2] known only to Caliptra. The UDS, once read by Caliptra ROM at boot, is then used to derive the IDevID identity.

IDevID key

A Caliptra RTM's IDevID key is a hardware identity generated by Caliptra ROM during manufacturing. This key must be solely wielded by Caliptra ROM, and shall never be exposed externally at any phase of the Caliptra life cycle. IDevID is used to endorse LDevID. See below for further details on IDevID provisioning.

[2] This obfuscation secret may be a chip-class secret, or a chip-unique PUF, with the latter preferred.

LDevID key

While it is recommended that implementations of Caliptra add physical attack countermeasures to protect fuses from imagery SDC attacks, SoC fuses generally have varying levels of resistance to physical attackers. While it is important to protect device security assets with physical attack countermeasures, a good design principle is to assume compromise. Renewable security, often referred to as trusted computing base recovery, is a base design principle in the Caliptra RTM. To mitigate the risk of UDS compromise for devices that may have been exposed to sustained physical attack in the supply chain, Caliptra RTMs shall support field-programmable entropy which factors into the device's LDevId identity. The LDevId identity is endorsed by IDevID and in turn endorses the FMC Alias key.

Caliptra's field-programmable entropy shall consist of at least four 32-byte slots. An owner may decide to program as few or as many slots as they wish. Upon programming new entropy, on next reset the device will begin wielding its fresh LDevID. Owners will need to validate the new LDevID by way of IDevID.

Note that LDevID is intended to hedge against the event that a supply-chain attacker has obtained UDS - and by extension, $IDevID_{priv}$. Therefore, IDevID's endorsement of LDevID should not be the sole signal to a user that LDevID is trustworthy. Owners should also work to ensure that their device onboarding flow - wherein field entropy is provisioned and LDevID is registered - is resistant to remote man-in-the-middle attackers that may attempt to use a previously-exfiltrated UDS to register a forged LDevID.

FMC alias key

The LDevID CDI is mixed with a hash of FMC, as well as the security state of the device, via a FIPS-compliant HMAC, to produce CDI_{FMC}. ROM uses CDI_{FMC} to derive the $Alias_{FMC}$ keypair. ROM wields LDevID to issue a certificate for $Alias_{FMC}$. The $Alias_{FMC}$ certificate includes measurements of the security state and FMC. ROM makes CDI_{FMC}, $Alias_{FMC}$, and its certificate, available to FMC.

FMC wields $Alias_{FMC}$ to issue a CSR for $Alias_{FMC}$. FMC then mixes CDI_{FMC} with a hash of runtime firmware to produce CDI_{RT}. FMC uses CDI_{RT} to derive the $Alias_{RT}$ alias keypair. FMC wields $Alias_{FMC}$ to issue a certificate for $Alias_{RT}$. This alias certificate includes measurements of runtime firmware. FMC makes CDI_{RT}, $Alias_{RT}$, its certificate, available to application firmware, while withholding CDI_{FMC} and $Alias_{FMC}$.

Security state

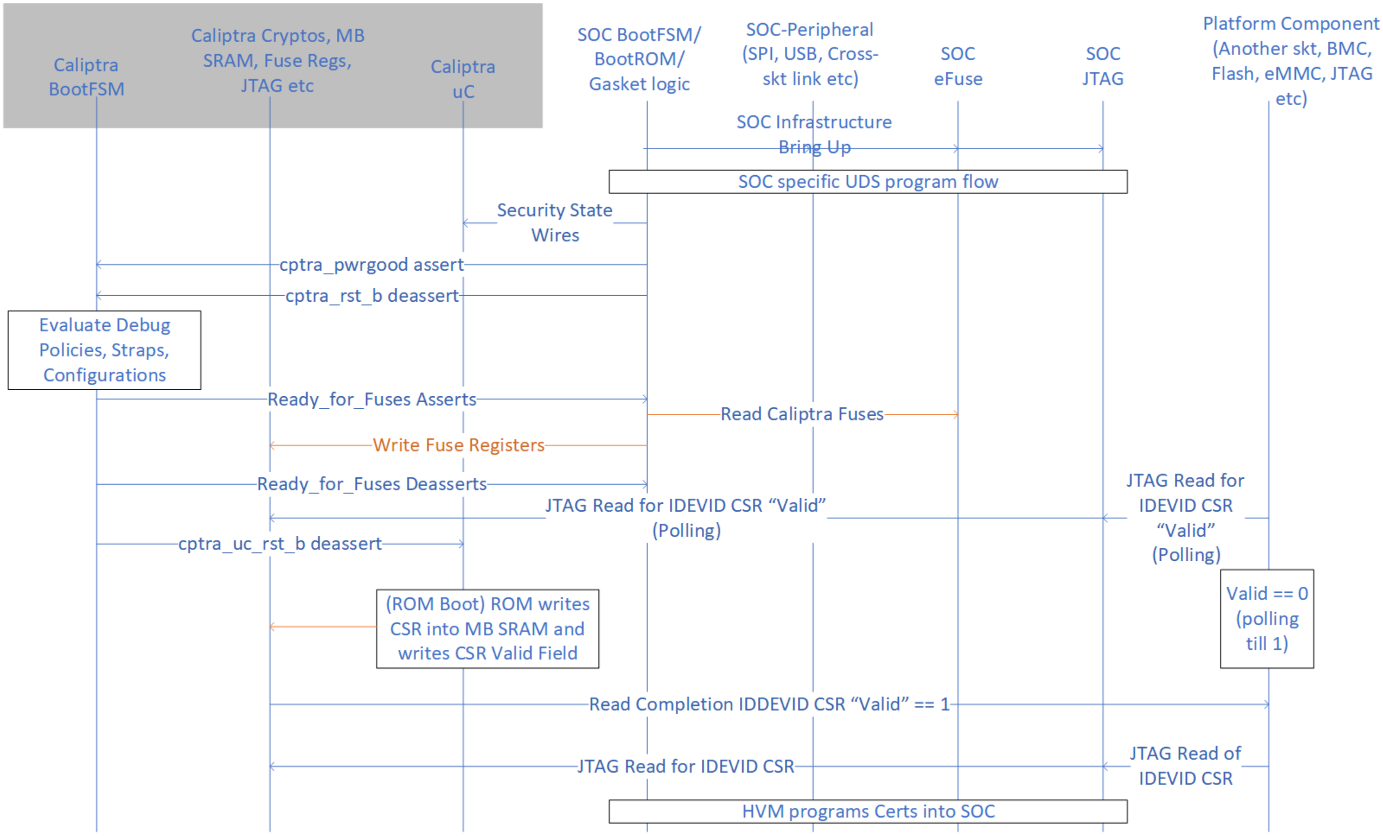
Devices may support features like debug unlock or JTAG. These features, when enabled, significantly alter the security state of the device. The configuration of these features shall be captured in the device's DICE identity. The security state shall be captured as an input to the FMC's CDI, and represented within the FMC's alias certificate.

Owner endorsement

Caliptra RTM firmware shall be signed by the vendor. In addition, this firmware may also be signed by the owner when ownership control is enforced. If a second signature is present for ownership enforcement, Caliptra must extract the owner's public key from the firmware image during cold boot, and latch the owner key into Caliptra's RAM for the remainder of its uptime[3]. Caliptra will then use both the vendor key and owner key to verify hitless firmware updates.

Caliptra shall attest to the value of the owner key, enabling external verifiers to ensure that the correct owner key has been provisioned into the device. Caliptra shall do so by including the owner key as an input to the FMC's CDI (as part of "other attributes" from Figure 3 above), and represent it within the FMC's alias certificate.

Provisioning IDevID during manufacturing



Caliptra BootFSM
Caliptra Cryptos, MB SRAM, Fuse Regs, JTAG etc
Caliptra uC
SOC BootFSM/ BootROM/ Gasket logic
SOC-Peripheral (SPI, USB, Cross-skt link etc)
SOC eFuse
SOC JTAG
Platform Component (Another skt, BMC, Flash, eMMC, JTAG etc)
SOC Infrastructure Bring Up
SOC specific UDS program flow
Security State Wires
cptra_pwrgood assert
cptra_rst_b deassert
Evaluate Debug Policies, Straps, Configurations
Ready_for_Fuses Asserts
Read Caliptra Fuses
Write Fuse Registers
Ready_for_Fuses Deasserts
JTAG Read for IDEVID CSR "Valid" (Polling)
JTAG Read for IDEVID CSR "Valid" (Polling)
cptra_uc_rst_b deassert
(ROM Boot) ROM writes CSR into MB SRAM and writes CSR Valid Field
Valid == 0 (polling till 1)
Read Completion IDDEVID CSR "Valid" == 1
JTAG Read for IDEVID CSR
JTAG Read of IDEVID CSR
HVM programs Certs into SOC

Figure 4: Device Manufacturing Identity Flow

1. High Volume Manufacturing programs NIST compliant UDS into fuses using SOC specific fuse programming flow. Note that this UDS goes through an obfuscation function within Caliptra IP. Hence it is fine for HVM to generate the UDS.

[3] This memory should only be volatile in the face of a power loss event. See details in HW Section.

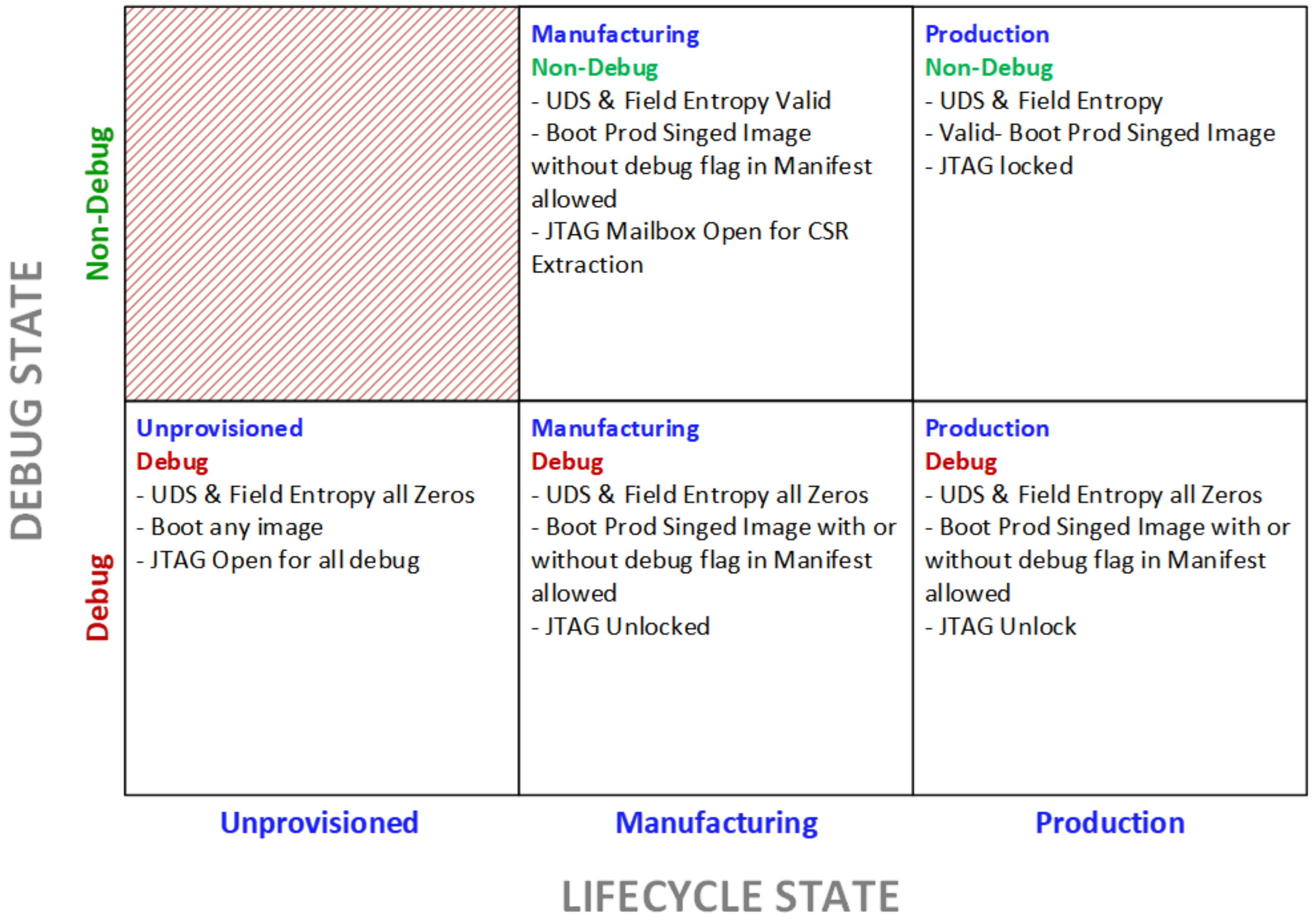
2. SOC will drive the security state indicating that its a manufacturing flow. Refer to “Caliptra Security States” for encodings
3. SOC will follow the boot flow as defined in Caliptra IP HW boot flow to assert cptra_pwrgood & deassert cptra_rst_b, followed by writing to the fuse registers.
4. HVM must now poll for “CSR Valid” bit available as Caliptra IP hardware register over JTAG
5. ROM will look at the manufacturing state encoding & populates the Caliptra internal SRAM [MB SRAM hardware structure is reused] with the CSR and write to Caliptra internal register to indicate CSR is valid (refer to Caliptra ROM spec & Identity section in this document on the ROM steps to generate the CSR)
6. HVM polling reads through JTAG will see that CSR is valid at this point
7. HVM must now read the CSR over JTAG
8. HVM must write a bit over JTAG that it has completed reading CSR
9. Caliptra IP HW will now open up the Caliptra Mailbox for SOC usages such as FW loading (if required in some HVM flows)
    a. Note that until the above write is complete, SOC will not get a grant/lock of the APB-exposed mailbox interface.

Certificate Format

Device Identity Certificates are following X.509 v3 format described in RFC 5280. The values in the X.509 certificate shall follow the DICE TCBInfo fields, as defined in [12]. The owner public key shall be extended into VendorInfo, with the security operational state reflecting the flags of DICE TCBInfo. Additional fields may be extended into VendorInfo.

[**TODO for 0.8 release**: The x509 owner key, JTAG state, public key used to verify firmware should be extended in the Cert.]

Caliptra Security States



DEBUG STATE	Unprovisioned	Manufacturing	Production
Non-Debug	▨	**Manufacturing** **Non-Debug** - UDS & Field Entropy Valid - Boot Prod Singed Image without debug flag in Manifest allowed - JTAG Mailbox Open for CSR Extraction	**Production** **Non-Debug** - UDS & Field Entropy - Valid- Boot Prod Singed Image - JTAG locked
Debug	**Unprovisioned** **Debug** - UDS & Field Entropy all Zeros - Boot any image - JTAG Open for all debug	**Manufacturing** **Debug** - UDS & Field Entropy all Zeros - Boot Prod Singed Image with or without debug flag in Manifest allowed - JTAG Unlocked	**Production** **Debug** - UDS & Field Entropy all Zeros - Boot Prod Singed Image with or without debug flag in Manifest allowed - JTAG Unlock

LIFECYCLE STATE

Figure 5: Caliptra Security States

Definitions

- **DebugLock:** Caliptra JTAG is NOT open for uController & HW debug
- **DebugUnlock:** Caliptra JTAG is open for uController & HW debug
- **Unprovisioned:** Blank/unprogrammed fuse part
- **Manufacturing:** Device is going through manufacturing flow where High-Volume-Manufacturing (HVM) Caliptra fuses are being programmed
- **Production:** All Caliptra's HVM Fuses are programmed.

Notes:

- Caliptra' security state is determined by the SOC's security state and SOC device lifecycle state.
- Caliptra's state is considered a mode of operation

    - Caliptra security state is defined by the upper most bit of the encoding below; 1=DebugLocked & 0=DebugUnlocked
    - Lower 2 bits are mapped to device lifecycle (Unprovisioned, Manufacturing, Production)
- SOC's security state may also be influenced by its own device life cycle.
- Caliptra's security state determines Caliptra's debug state and the state of its security assets.
- In general, if Caliptra is in unsecure state, all keys, assets are 'zeroized'. Zeroized may mean switching to all 0s or 1s or debug keys based on the key. Refer to Caliptra Assets for a description of Caliptra assets.

{Security State, Device Life Cycle State[1:0]}	State	Definition	State Transition Requirement
000b	DebugUnlocked & Unprovisioned	This is Caliptra's default state; it is used for development and early Caliptra bring up. This state is not used to provision the Caliptra assets. In this state: ● UDS and all other identity critical assets shall not be not programmed in fuses. Unprogrammed Fuse bits shall be read as 0s (zero). ○ The debug UDS shall be obfuscated and de-obfuscated using the debug obfuscation key. ● Obfuscation key: The debug obfuscation key shall be used ● Caliptra JTAG is unlocked and allows microcontroller debug ● Caliptra JTAG can access IP internal registers through FW or directly	Unprovisioned to any other state required cold boot cycle of Caliptra & SOC
100b	DebugLocked & Manufacturing	Caliptra is commanded to enter this state during the secure High-Volume-Manufacturing (HVM) process. In this state: ● UDS and other identity critical assets shall be programmed into Fuses. They	Manufacturing -> Unsecure State transition possible without power cycle and Caliptra will clear all the security critical

		are written into Caliptra fuse registers, similar to the ‘Secure’ state. ● All security assets shall be in production mode (production UDS and obfuscation shall be used) ● Caliptra JTAG shall be locked – microcontroller debug shall be disabled ● Caliptra microcontroller can be interrupted through JTAG mailbox	assets/registers before JTAG is opened Manufacturing -> Secured state possible ONLY with a power cycle Refer to Provisioning During Manufacturing for details on manufacturing and provisioning details.
101b	DebugLocked & Production	All security assets are in production mode. In this state: ● Production UDS and obfuscation key shall be used. ● CPU execution shall be enabled ● All ‘backdoor’ functionality shall be disabled (e.g., developer functions/functionality that could reveal sensitive information or result in escalation of privileges, etc.) ● Debug functions shall be disabled ○ Caliptra JTAG is locked – microcontroller debug shall be disabled ○ Caliptra microcontroller shall not be interruptible through JTAG mailbox ● DFT functions shall be disabled	DebugLocked -> Debug Unlocked possible without power cycle and Caliptra will clear all the security critical assets/registers before JTAG is opened
011b	DebugUnlocked & Production	This state is used when debugging of Caliptra RTM is required. When in this state: ● UDS and other identity critical assets are programmed into Fuses. They may not have been written into Caliptra fuse registers if the unsecure state entered before Caliptra is out of reset. If unsecure state transition happened after fuses are written to Caliptra, they are cleared on seeing the security state	Debug Unlocked -> Debug Locked possible ONLY with a power cycle.

		transition from secure/manufacturing -> unsecure ● **Caliptra state:** All security assets are in debug mode (UDS & Obfuscation key are in production state) ○ UDS: Reverts to a ‘well-known’ debug value ○ Obfuscation key: Switched to debug key ○ Key Vault is also cleared ○ Caliptra JTAG is unlocked and allows microcontroller debug ○ Caliptra JTAG can access IP internal registers through FW or directly	

Table 1: Security States

Note: End of life state is owned by SOC. In end-of-life device life cycle state, Caliptra shall not not be brought out of reset.
Note: Other encodings are reserved and always assumed to be in a secure state.

Each of these security states may be mapped to different SOC level debug & security states. SOC’s requirement is that if the SOC enters a debug state, then Caliptra must also be in Unsecured state where all assets are cleared.

Service Surface

The service surface of a Caliptra RTM has multiple vectors. All use cases are control plane services, useful to power on a system or start a task. Supporting line rate high performance IO cryptography or any other data path capability is not required.

- **Logic IOs:** required to indicate status of the IP, availability of a message through APB, and to enable/disable certain debug capabilities (like JTAG enable/disable)
- **Command mailbox**: Caliptra shall offer services to other parts of the SoC:
  - **Loading firmware**: Caliptra firmware is loaded via the mailbox at cold-boot. In addition, Caliptra firmware can be loaded at runtime to support hitless/impactless updates.
  - **DICE-as-a-Service**: A Caliptra RTM shall expose a "DICE-as-a-Service" API, allowing Caliptra to derive and wield a DICE identity on behalf of other elements within the SoC.
    - A potential use case includes serving as a signing oracle for an SPDM responder executing in the SoC Application Processor.

Device Resilience

As noted earlier in this document, Caliptra has a role to play in maintaining the resilience posture of the SoC as defined by NIST SP800-193 Platform Firmware Resiliency Guidelines [1]. As the Silicon RTM, Caliptra is either responsible for, or participates in, various Protection and Detection requirements described in the NIST publication.

The following list describes the NIST SP800-193 requirements that Caliptra shall meet, either on its own or in conjunction with other components within the SoC or Platform. Requirements not listed should be assumed not covered and out-of-scope for Caliptra. In particular, most requirements related to firmware update and recovery are out-of-scope and must be handled by other components of the system.

NIST SP800-193 Chapter	Requirement	Caliptra Responsibility
4.1.1	All security mechanisms and functions shall be founded to Roots of Trust.	Caliptra forms the basis for all trust in the SoC starting from execution of its immutable ROM. See chapter on Secure Boot Flow.
4.1.1	If Chains of Trust (CoT) are used, RoT shall serve as the anchor for the CoT.	All other firmware shall be authenticated and executed as part of a Chain of Trust extended from the Caliptra ROM. See chapter on Secure Boot Flow.
4.1.1	All RoTs and CoTs shall either be immutable or protected using mechanisms which ensure all RoTs and CoTs remain in a state of integrity.	All other firmware is authenticated and executed as part of a Chain of Trust extended from the Caliptra ROM. See chapter on Secure Boot Flow.
4.1.1	All elements of the Chains of Trust for Update, Detection and Recovery in non-volatile storage shall be implemented in platform firmware.	Caliptra forms the basis for Root of Trust for Measurement (or Detection). All other silicon RoT capabilities are extended by additional firmware loaded in the SoC and anchored by the Caliptra RTM.
4.1.1	The functions of the RoTs or CoTs shall be resistant to any tampering attempted by software running under, or as part of, the operating system on the host processor.	Caliptra shall run on a dedicated microcontroller, isolated physically from access by other components in the system.
4.1.1	Information transferred from the	Caliptra shall verify the authenticity of its

NIST SP800-193 Chapter	Requirement	Caliptra Responsibility
	software on the host processor to the platform firmware shall be treated as untrusted.	firmware using an approved digital signature verification mechanism.
4.1.1	CoTs may be extended to include elements that are not from non-volatile storage. Before use, those elements shall be cryptographically verified by an earlier element of the CoT.	Caliptra shall verify the authenticity of its firmware using an approved digital signature verification mechanism. Caliptra shall also measure the SoC Security Processor FMC code before it is verified and executed by the SoC.
4.1.2	If the key store is updateable, then the key store shall be updated using an authenticated update mechanism, absent unambiguous physical presence through a secure local update.	Hashes for the keys used to authenticate Caliptra FW are programmed into fuses during manufacturing. If a key is deemed to be compromised, that key may be revoked and the next key used instead. See chapter on Fuse/OTP Requirements.
4.1.3	Each platform device which implements a detection capability shall rely on either a Root of Trust for Detection (RTD), or a Chain of Trust for Detection (CTD) which is anchored by an RTD, for its detection.	Caliptra forms the basis for all trust in the SoC starting from execution of its immutable ROM. All other firmware shall be authenticated and executed as part of a Chain of Trust extended from the Caliptra ROM. See chapter on Secure Boot Flow.
4.1.3	The RTD or CTD shall include or have access to information necessary to detect corruption of firmware code and critical data.	Caliptra relies on hashes of authorized keys stored in fuses. Those hashes are then checked against public keys found in firmware headers to authenticate Caliptra's runtime firmware. Caliptra relies on redundancy in the fuses to protect the key and configuration data. See chapter on Fuse/OTP Requirements
4.2.3	If Critical Platform Firmware code in non-volatile memory is copied into RAM to be executed (for performance, or for other reasons) then the firmware program in RAM shall be protected from modification by software or shall complete its function before software starts.	Caliptra shall run on a dedicated microcontroller, isolated physically from access by other components in the system.
4.2.3	If Critical Platform Firmware uses	Caliptra shall run on a dedicated

NIST SP800-193 Chapter	Requirement	Caliptra Responsibility
	RAM for temporary data storage, then this memory shall be protected from software running on the Platform until the data's use is complete.	microcontroller, isolated physically from access by other components in the system.
4.2.3	Software shall not be able to interfere with the intended function of Critical Platform Firmware. For example, by denying execution, modifying the processor mode, or polluting caches.	Caliptra shall run on a dedicated microcontroller, isolated physically from access by other components in the system. In addition, the Caliptra subsystem begins execution before other firmware is allowed to run.
4.2.4	Critical data shall be modifiable only through the device itself or defined interfaces provided by device firmware. Examples of defined interfaces include proprietary or public application programming interfaces (APIs) used by the device's firmware, or standards-based interfaces. Symbiont devices may rely on their host devices to meet this requirement.	Caliptra receives firmware and configuration input only via defined interfaces within the SoC. See chapter on Mailboxes.
4.2.1.3	The authenticated update mechanism shall be capable of preventing unauthorized updates of the device firmware to an earlier authentic version that has a security weakness or would enable updates to a version with a known security weakness.	Caliptra supports a mechanism for detecting and preventing execution of a prior firmware image that is no longer authorized. See chapter on Anti-rollback Support.
4.3.1	A successful attack which corrupts the active critical data or the firmware image, or subverts their protection mechanisms, shall not in and of itself result in a successful attack on the RTD or the information necessary to detect corruption of the firmware image.	Caliptra shall verify the signature of any firmware it loads during each boot. If the signature verification fails, Caliptra shall notify the SoC that firmware recovery must be performed. Refer to Error Reporting and Handling.
4.3.1	Verify integrity, using an approved digital signature algorithm or	Caliptra shall perform digital signature verification of its FMC code, as well as that

NIST SP800-193 Chapter	Requirement	Caliptra Responsibility
	cryptographic hash, of device firmware code prior to execution of code outside the RTD.	of the SoC Security Processor FMC before they are allowed to execute.
4.3.1	If firmware corruption is detected, the RTD or CTD should be capable of starting a recovery process to restore the device firmware code back to an authentic version.	Caliptra shall notify the SoC via the Mailbox interface to initiate the recovery process.
4.3.1	The detection mechanism should be capable of creating notifications of firmware corruption.	Caliptra shall notify the SoC via the Mailbox interface to initiate the recovery process.
4.3.1	The detection mechanism should be capable of logging events when firmware corruption is detected.	It is the responsibility of the SoC to log any corruption events upon notification by Caliptra.
4.3.2	The RTD or CTD shall perform integrity checks on the critical data prior to use. Integrity checks may take the form, for example, of validating the data against known valid values or verifying the hash of the data storage.	Caliptra relies on redundant fuses to store its configuration data, which is owned and passed to Caliptra through the Mailbox.
4.3.2	The RTD or CTD should be capable of creating notifications of data corruption.	Refer to Error Reporting and Handling.
4.3.2	The detection mechanism should be capable of logging events when data corruption is detected.	It is the responsibility of the SoC to log any corruption events upon notification by Caliptra.

Secure Boot Flow

A Caliptra RTM shall follow/implement the secure boot guidelines as described in [Reference 11].

Detailed flow described in HW Section.

Caliptra RTM hitless update

Caliptra shall preserve its runtime firmware's DICE identity across hitless updates.

Informative comment: this is done because it is unsafe to unlock UDS again, and infeasible to extend the DICE hierarchy with the new firmware's measurements.

Caliptra shall provide a means of indicating in a firmware image's signed header whether runtime update of Caliptra firmware should be enabled. This bit shall be made evident in the firmware's alias key certificate and used in the firmware's CDI derivation.

Anti-rollback Support

A Caliptra RTM shall provide Fuse banks (refer to *Table 9: Caliptra Secret Fuse Descriptor Table*) that are used for storing monotonic counters to provide anti-rollback enforcement for Caliptra mutable firmware. Each distinctly-signed boot stage shall be associated with its own anti-rollback Fuse field. Together with the vendor, a Caliptra RTM allows owners to enforce strong anti-rollback requirements, in addition to supporting rollback to a previous firmware version – this is a critical capability for hyper scalar owners.

Every mutable Caliptra RTM boot layer shall include a SVN value in the signed header. If a layer's signed SVN value is less than the current counter value for that layer's fuse bank, the Caliptra RTM shall refuse to boot that layer, regardless of whether the signature is valid.

Each signed boot layer shall also include a MIN_SVN value in the signed header. Upon successful validation of a signed boot layer, if the layer's signed MIN_SVN value is greater than the current counter value for that layer's fuse bank, Caliptra shall increment that fuse bank counter until it equals MIN_SVN.

Vendors shall issue security-critical fixes requiring anti-rollback protection in sets of two signed firmware:

- Version number (X+1) which carries the security fix, with an incremented SVN and an unchanged MIN_SVN, as compared to version (X).
- Version number (X+2), identical to (X+1) except its MIN_SVN value has been incremented.

Owners may upgrade their fleet in two stages: first from (X) to (X+1), and then from (X+1) to (X+2).

If an owner does not require the ability to roll back during qualification, they can choose to perform a single upgrade of their fleet, from (X) to (X+2), skipping over the intermediate (X+1).

Each of Caliptra's internal anti-rollback fuse banks shall support a minimum counter value of 64. This feature is expected to be used relatively sparingly.

If a given firmware image's SVN is less than its MIN_SVN value, that image shall be considered invalid.

Alternatively, platform vendors may prefer to manage firmware storage and rollback protection in a different manner, such as through a dedicated Platform RoT. In such cases, the vendor may wish to disable Anti-rollback support from Caliptra entirely. This disable is available via an OTP/fuse setting.

Informative comment: Example

The following is a worked example of how the anti-roll back mechanism may be used to revoke a signed image while supporting rollback to the prior image.

- Assuming Caliptra is in the following state:
- Currently running firmware version: 4.2
- Caliptra firmware's signed SVN value: 1
- Caliptra firmware's signed MIN_SVN value: 1
- Caliptra's firmware anti-rollback fuse bank counter value: 1

A vulnerability is discovered in firmware version 4.2. The vendor issues a fix in firmware version 4.3. However, owners may still wish to roll back to firmware version 4.2, while version 4.3 is being qualified in their fleet. Updating to firmware version 4.3 will place the Calipra RTM in the following state:

- Currently running firmware version: 4.3
- Caliptra firmware's signed SVN value: 2
- Caliptra firmware's signed MIN_SVN value: 1
- Caliptra's firmware anti-rollback fuse bank counter value: 1

In this state, since the anti-rollback fuse bank counter has not yet been incremented, the Caliptra RTM will still allow the firmware to roll back to version 4.2.

Once firmware version 4.3 is fully qualified, the owner will wish to revoke version 4.2. The vendor will issue a follow-up firmware version 4.4, which will place the Calipra RTM in the following state:

- Currently running firmware version: 4.4
- Caliptra firmware's signed SVN value: 2

- Caliptra firmware's signed MIN_SVN value: 2
- Caliptra's firmware anti-rollback fuse bank counter value: 2

Upon validating firmware version 4.4, the Caliptra RTM will note that that firmware's signed MIN_SVN value is 2, and will increment its internal fuse bank counter to match. Once a chip has received firmware version 4.4, it will no longer be able to roll back to version 4.2. However, it *will* continue to be able to roll back to version 4.3.

Physical Attack Countermeasures

A Caliptra RTM shall implement counter measures designed to deter both glitching (also referred to fault-injection (FI) and side-channel attacks (simple power analysis (SPA) and differential power analysis (DPA)).

The Caliptra threat model guides the priority of which physical countermeasures are based on a specific physical implementation.

From the top, an adversary in the supply chain has essentially unlimited time to glitch the chip and make it reveal any private key material or symmetric secrets. One Glitch To Rule Them All is one example with recency bias. The most critical counter-measures must prevent non-destructive extraction of those secrets. Otherwise, an adversary succeeding may be able to silently impersonate production serving assets at a later time.

General protection of the embedded microprocessor while running arbitrary firmware is required to protect the UDS or other private entropy from logical and physical attacks, including firmware running within the Caliptra itself. Control flow integrity, analog reference voltage and clock sources, as well as defensive programming are encouraged. Likewise, pointer authentication, encryption, separate code vs data stacks, and memory tagging are all encouraged.

Randomly generated per part entropy is subject to physical inspection attacks in the supply chain, as well. The Fuses storing the UDS entropy shall be protected to a degree that forces an attacker to perform a destructive operation to read their values. Decapping and fibbing attacks should at least penetrate enough layers and metal shielding to render the part useless, if not being outright impossible to carry out. Entropy tied to a damaged asset typically requires injection of counterfeit devices in the supply chain, which is a very powerful adversary model.

Another way to obtain access to secret entropy with “unlimited supply chain time” is to observe side channels while the SoC is executing. Because a Caliptra RTM is expected to be a <1 mm^2 fraction of a large SoC, side-channel mitigation is required only against extremely resourceful attackers that can wade and discern through a large number of confounding signals and power profiles. With that priority in mind, DPA and DMA attacks should be mitigated via decoy value generation.

Any private key material or symmetric key material embedded in the RTL (and therefore “global”) must be treated as having low value, reaching zero value in a number of quarters. A supply chain attacker can destructively obtain the key material, and loss of one part is not going to trigger any alarms.

Mitigation against SCA is not necessarily trivial and may be implemented in a variety of ways. [14] provides a comprehensive overview of methods and techniques used in various SCA as well as recommendations for countermeasures against such attacks (including feasibility and applicability). Additionally, there are many academic papers available from NIST and other resources that discuss SCA and their countermeasures.

Compliance and Certification Requirements

Due to the Identity service surface offered to other SoC subsystems, a Caliptra RTM may fall under the ToE (Target of Evaluation) of an application that wishes to attain a specific compliance level for business reasons.

It is important to highlight it's not necessary for the RTM itself to unilaterally attain e.g. FIPS 140-3 L3. It is only relevant insofar the RTM is included in the "bag" that wants to obtain a compliance certification. For example, if a cloud provider wants to FIPS-certify PCIe link encryption in transit rooted to an ASIC identity emanating from a Caliptra RTM.

Refer to [15] for requirements related to Keys, Entropy, and Random Bits and cryptographic modules and algorithms.

Known Answer Test (KAT) Support

In order to certify a cryptographic module, pre-operational self-tests must be performed when the system is booted. Implementation of KATs are required for FIPS certification. However, regardless of FIPS certification, it is considered a security best practice to ensure that the supported cryptographic algorithms are functioning properly so as to guarantee correct security posture.

KAT execution are described as two types

- Pre-operational Self-Test (POST)
- Conditional Algorithm Self-Test (CAST)

A detailed description of the POST and CAST KATs can be found at csrc.nist.gov.

KAT Type	If fails
POST	Failure of a POST KAT (e.g., ECDSA) shall result in Caliptra RTM ***boot failure***. A reset may or may not result in successful POST completion.
CAST	Failure of a CAST KAT shall cause Caliptra RTM to fail any operation that has a dependency on the associated cryptographic algorithm.

Table 6: KAT Failure Mitigations

Crypto Algorithm	Caliptra Boot ROM	Caliptra FMC	Caliptra Runtime FW
ECDSA[4]	Yes	Yes	Yes
SHA[5]	Yes	Yes	Yes
DRBG	No	No	No
HMAC	Yes (CDI generation)	No	No
KDF	Yes	Yes	No

Table 7: POST/CAST Usage

As shown in *Table 7: POST/CAST Usage*, since the cryptographic algorithms required by the Caliptra RTM Boot ROM are considered POSTs and those same algorithms are used by Caliptra FMC and FW (green boxes), there is no requirement that FMC and Runtime FW implement CASTs for those algorithms.

[4] ECDSA is used for FW verification and SPDM (signing)

[5] SHA – is used with ECDSA, HMAC and for generating measurements

FW Signing/Verification Algorithms [updated]

Caliptra firmware is composed of multiple layers: an FMC and an application firmware image. Each layer is signed individually by ECDSA P384 keys.

Each layer is signed by a vendor-controlled key. In addition, each layer may also be signed by an owner-controlled key. The image header contains both the owner public key as well as the signature using that key.

During boot, Caliptra ROM shall verify the vendor signature over FMC before allowing that FMC to run.

See the Owner endorsement section for how the owner key is used.

Caliptra RTM FW signature generation and verification shall follow the requirements described in [Reference 11]

Post-Quantum Cryptography (PQC) Requirements

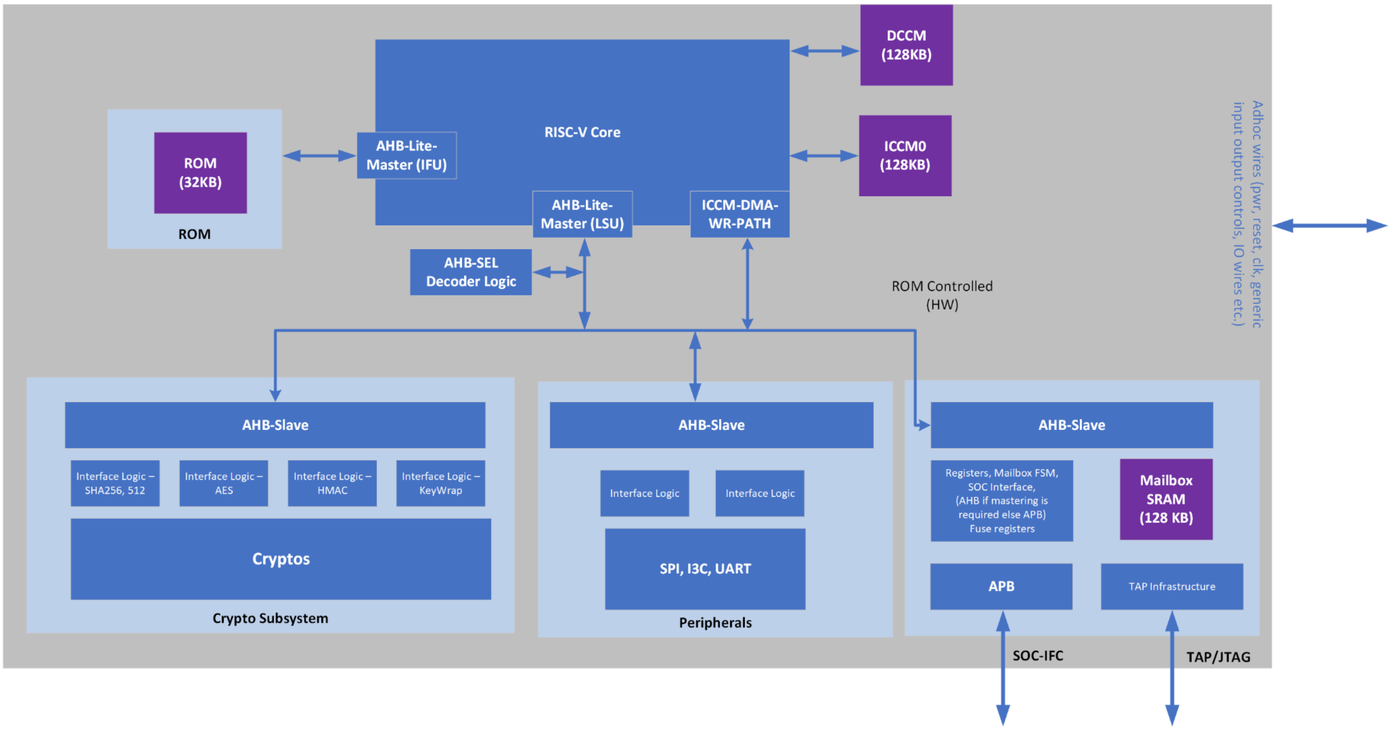
As NIST publishes new standards with PQC resilience, algorithms applicable to Caliptra will be described in this document.

Key Rotation

Firmware signing key rotation shall follow the requirements described in [Reference 11].

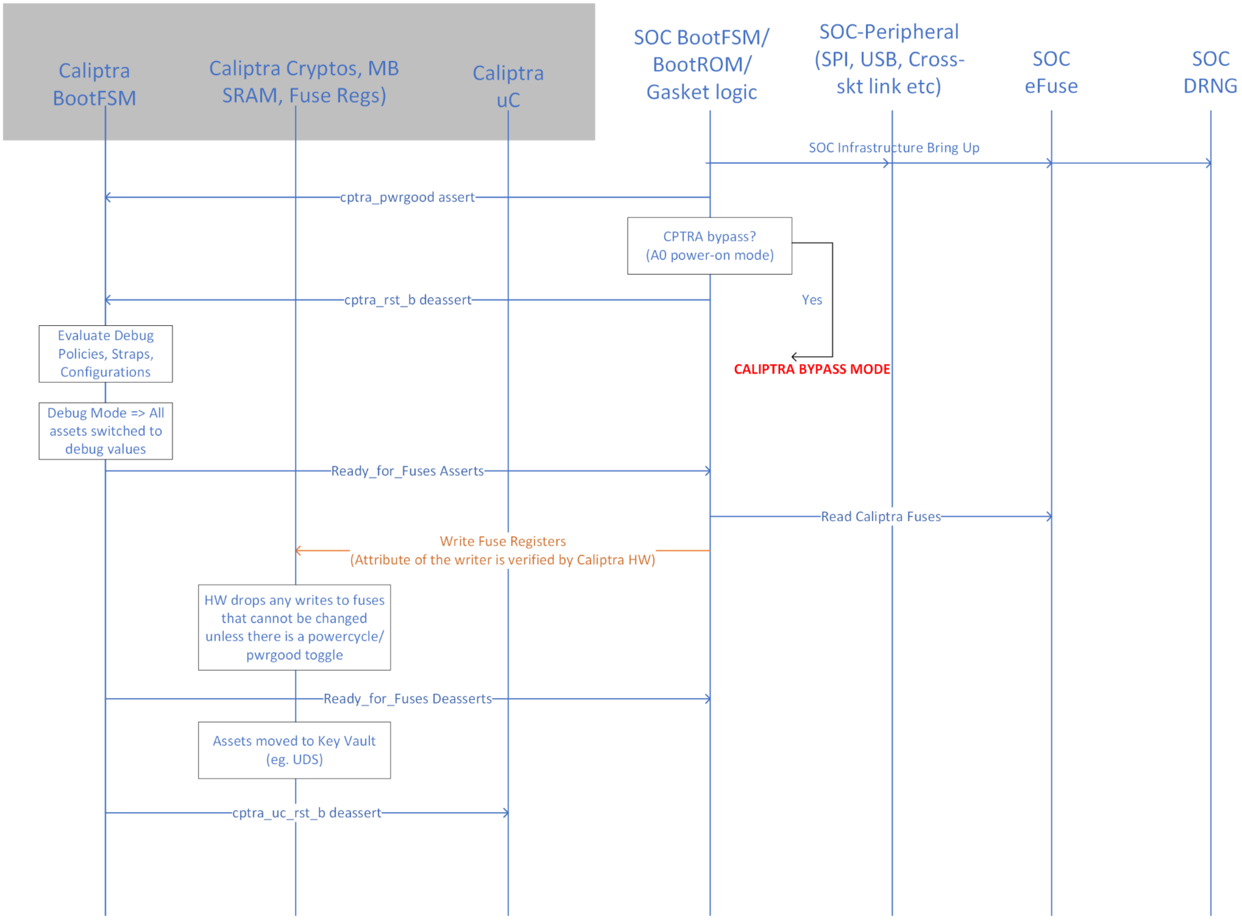
HW Section

HW Block Diagram



- SRAM requirements:
    - 128KB of ICCM0 and ICCM1 (used for rollback of impactless update)
        - **Open:** Without ICCM1, SOC owns the flow to stage the FW if the FW updated through impactless flow fails to run appropriately. Potential SRAM savings to be discussed.
    - 128KB for Mailbox as a staged SRAM (for FW staging of impactless updates to do authentication checks on the FW before moving to ICCM)
    - 128KB for DCCM and 32KB for ROM
- Crypto requirements
    - SHA256 SHA384, SHA512
    - ECC Secp384r1 w/ HMAC-DRBG - Key Generation, Sign & Verification
    - HMAC SHA384
    - AES256-ECB, CBC, GCM
- SWeRV EL2 from chips alliance is used for RISC-V
- APB is the choice of SOC facing interface
- JTAG is exported at the IP interface

Caliptra IP HW Boot Flow

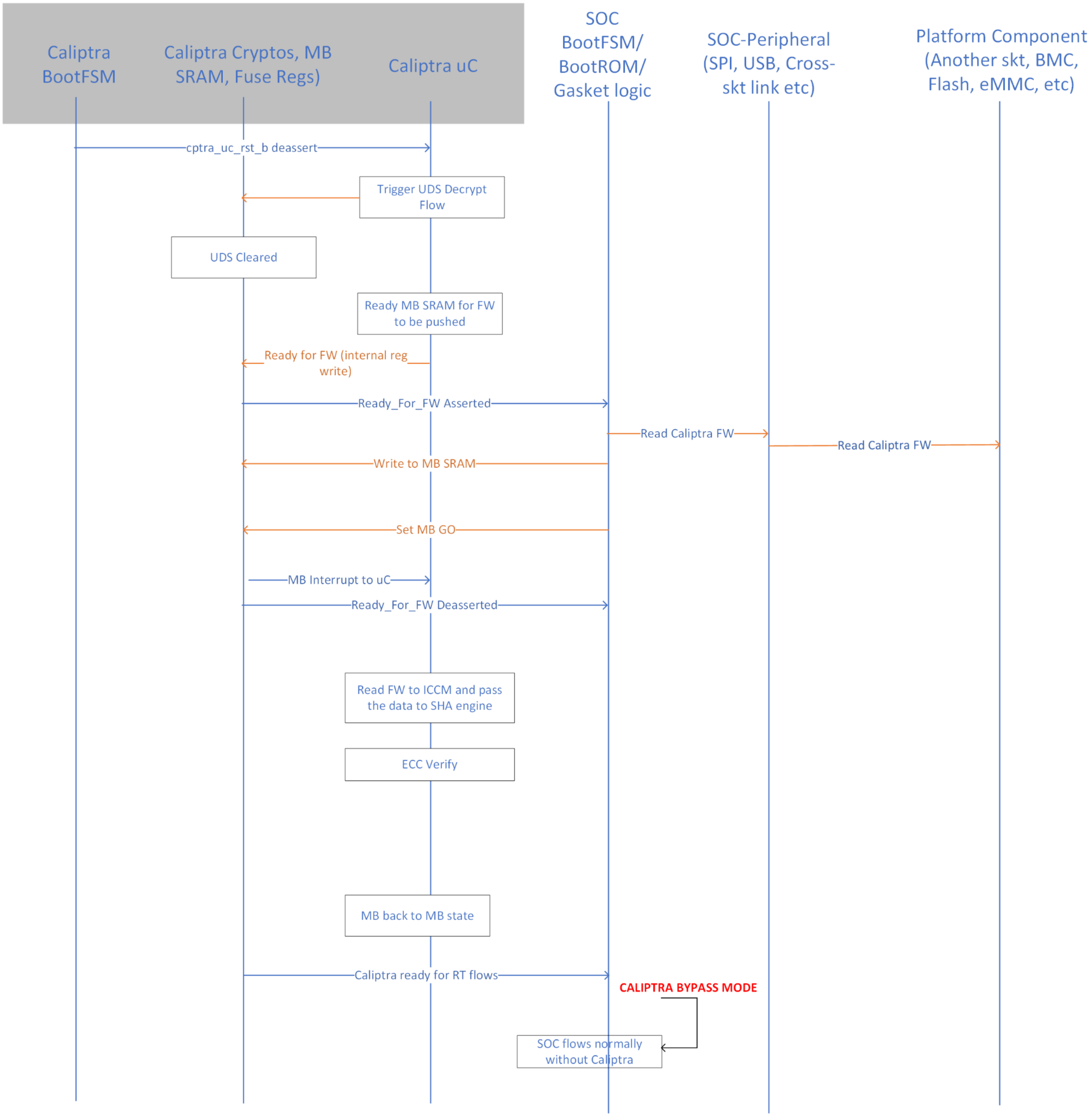

Caliptra BootFSM
Caliptra Cryptos, MB SRAM, Fuse Regs)
Caliptra uC
SOC BootFSM/ BootROM/ Gasket logic
SOC-Peripheral (SPI, USB, Cross-skt link etc)
SOC eFuse
SOC DRNG
SOC Infrastructure Bring Up
cptra_pwrgood assert
CPTRA bypass? (A0 power-on mode)
Yes
CALIPTRA BYPASS MODE
cptra_rst_b deassert
Evaluate Debug Policies, Straps, Configurations
Debug Mode => All assets switched to debug values
Ready_for_Fuses Asserts
Read Caliptra Fuses
Write Fuse Registers (Attribute of the writer is verified by Caliptra HW)
HW drops any writes to fuses that cannot be changed unless there is a powercycle/ pwrgood toggle
Ready_for_Fuses Deasserts
Assets moved to Key Vault (eg. UDS)
cptra_uc_rst_b deassert

1. As a part of SOC boot flow, SOC may have a bunch of infrastructure and other entities that boot. That part of the flow is outside the scope of this document. If SOC chooses to bypass Caliptra, then it should have a capability to bypass the Caliptra entirely through its proprietary flow. This may be needed for A0 power on and other early validation.
2. Cptra_pwrgood is asserted to the Caliptra IP block.
3. Cptra_rst_b is deasserted to the Caliptra IP block. Refer to the integration specification for guidelines on the minimum number of cycles b/w these two signals
4. Caliptra IP will now evaluate the strap settings driven through various interface wires (eg. passive vs active mode, security/debug state of the SOC etc)
5. If SOC is in a debug mode, then security assets are cleared/switched to debug mode
6. Caliptra IP will assert Ready_For_Fuse wire to the SOC
7. SOC will populate the fuse registers and set a fuse write done bit in the same fuse register block. Note that Caliptra HW drops writes to any registers that cannot be changed unless there is a power cycle (eg. UDS). So SOC is free to write all the registers.

    a. **Open:** To reduce the overall complexity, there is a proposal to write a SOC generated random number as a part of fuse population. From there, FW will implement DRBG using DRNG as a starting point.

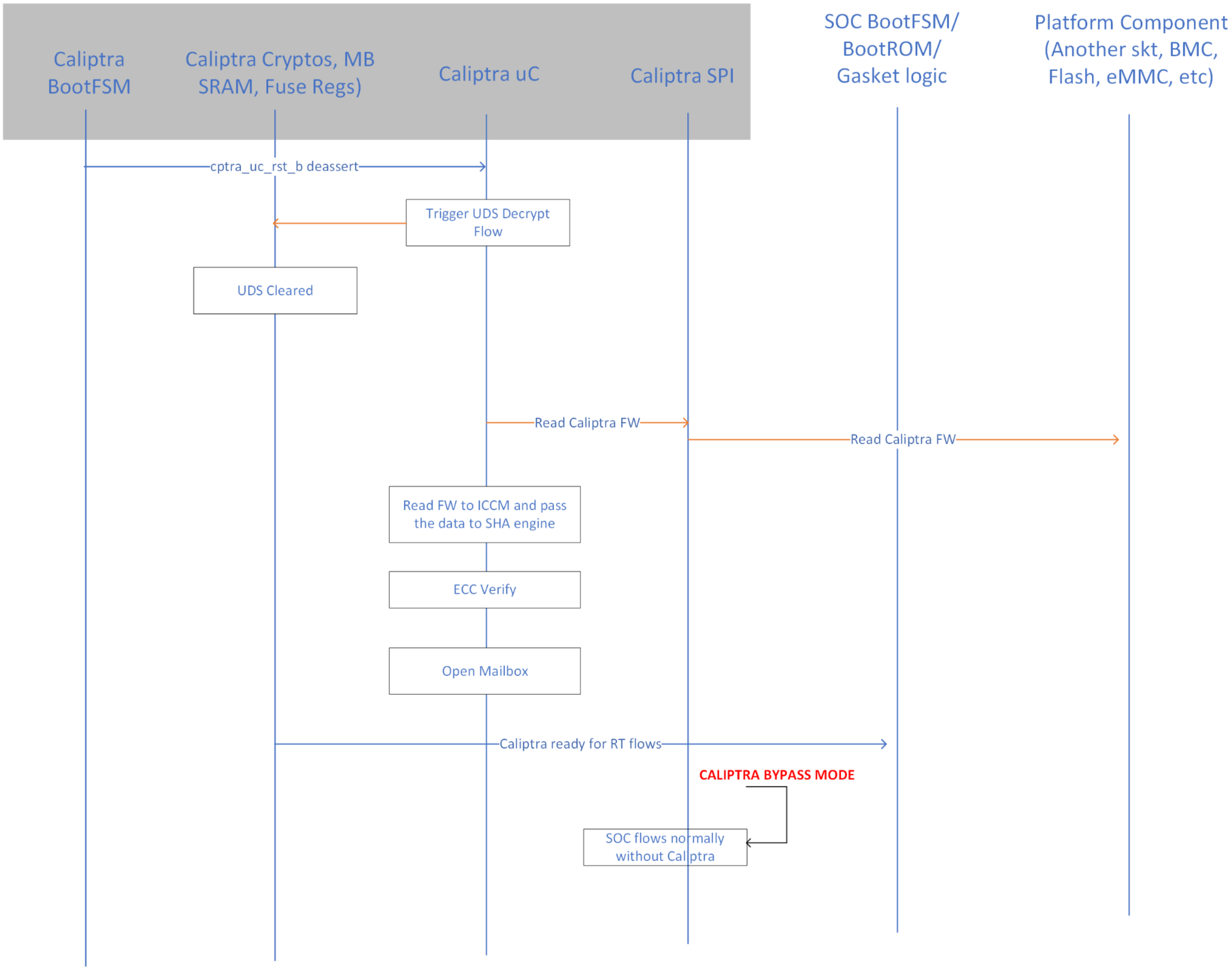
8. Caliptra IP will deassert Ready_for_Fuse wire as soon as the fuse write done register is written.
9. Caliptra IP moves security critical assets in fuse registers (eg. UDS) to Key Vault.

Caliptra FW Push Flow (Passive mode)


Caliptra BootFSM
Caliptra Cryptos, MB SRAM, Fuse Regs)
Caliptra uC
SOC BootFSM/ BootROM/ Gasket logic
SOC-Peripheral (SPI, USB, Cross-skt link etc)
Platform Component (Another skt, BMC, Flash, eMMC, etc)
cptra_uc_rst_b deassert
Trigger UDS Decrypt Flow
UDS Cleared
Ready MB SRAM for FW to be pushed
Ready for FW (internal reg write)
Ready_For_FW Asserted
Read Caliptra FW
Read Caliptra FW
Write to MB SRAM
Set MB GO
MB Interrupt to uC
Ready_For_FW Deasserted
Read FW to ICCM and pass the data to SHA engine
ECC Verify
MB back to MB state
Caliptra ready for RT flows
CALIPTRA BYPASS MODE
SOC flows normally without Caliptra

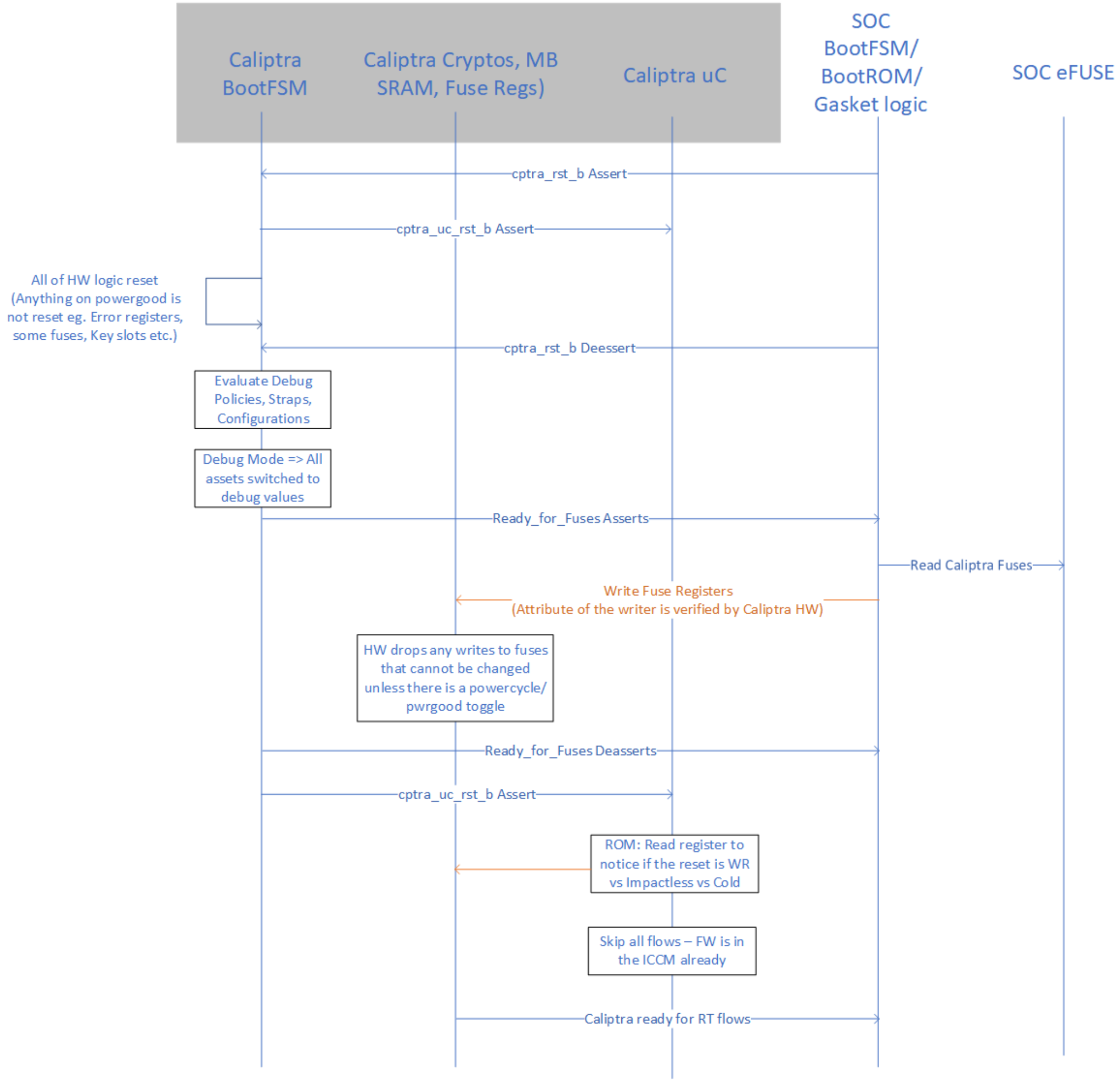
1. Once Caliptra uController is out of reset, ROM starts executing and triggers the crypto block to run the UDS decrypt flow.
2. Caliptra ROM will now enable the Mailbox. (until this point any accidental/non-accidental writes that target the mailbox are dropped by the hardware)
3. Caliptra ROM will assert READY_FOR_FW wire. This is done by writing an internal register. This register is also visible to read on the APB interface. SOC can choose to poll on this bit instead of using the wire (it is SOC integration choice).
4. SOC will follow the mailbox protocol and push Caliptra FW into the mailbox
5. Caliptra's mailbox HW will assert an interrupt to the uController once the GO is written per mailbox protocol. See Mailbox for specifics.
6. Once Caliptra's FW is authenticated and loaded into ICCM, uController will assert READY_FOR_RTFLOWS wire. Refer to ROM & FW spec on next level specifics of various security flows that happen within this step (eg. DICE).

Caliptra IP FW Load Flow (Active Mode)


Caliptra BootFSM
Caliptra Cryptos, MB SRAM, Fuse Regs)
Caliptra uC
Caliptra SPI
SOC BootFSM/ BootROM/ Gasket logic
Platform Component (Another skt, BMC, Flash, eMMC, etc)
cptra_uc_rst_b deassert
Trigger UDS Decrypt Flow
UDS Cleared
Read Caliptra FW
Read Caliptra FW
Read FW to ICCM and pass the data to SHA engine
ECC Verify
Open Mailbox
Caliptra ready for RT flows
CALIPTRA BYPASS MODE
SOC flows normally without Caliptra

1. Once Caliptra uController is out of reset, ROM starts executing and triggers the crypto block to run the UDS decrypt flow.
2. Caliptra ROM will use the internal SPI peripheral to read the platform flash and load the FW. Note that SPI will be operating in basic functional single-IO mode and at 20MHz frequency.
3. Once Caliptra's FW is authenticated and loaded into ICCM, uController will assert READY_FOR_RTFLOWS wire. Refer to ROM & FW spec on next level specifics of various security flows that happen within this step (eg. DICE).

CPU Warm Reset or PCIe Hot Reset Flow → Caliptra IP reset


Caliptra BootFSM
Caliptra Cryptos, MB SRAM, Fuse Regs)
Caliptra uC
SOC BootFSM/ BootROM/ Gasket logic
SOC eFUSE
cptra_rst_b Assert
cptra_uc_rst_b Assert
All of HW logic reset (Anything on powergood is not reset eg. Error registers, some fuses, Key slots etc.)
cptra_rst_b Deessert
Evaluate Debug Policies, Straps, Configurations
Debug Mode => All assets switched to debug values
Ready_for_Fuses Asserts
Read Caliptra Fuses
Write Fuse Registers (Attribute of the writer is verified by Caliptra HW)
HW drops any writes to fuses that cannot be changed unless there is a powercycle/ pwrgood toggle
Ready_for_Fuses Deasserts
cptra_uc_rst_b Assert
ROM: Read register to notice if the reset is WR vs Impactless vs Cold
Skip all flows – FW is in the ICCM already
Caliptra ready for RT flows

Note: Since Caliptra IP may be placed in a “S5” domain of the device, there may be devices where Caliptra IP may not go through reset on a device hot reset or CPU warm reset. But the flow shows what happens when such a reset happens.

1. Caliptra IP’s reset is asserted by the SOC
2. Caliptra’s internal BootFSM will reset the uController and then resets all the logic (including the SOC facing APB interface). Only registers or flops that are sitting on powergood are left to have the same value. Note that SRAMs do not have a reset.
3. Caliptra IP’s reset is de-asserted by the SOC
4. At this point the HW boot flow will be the same cold boot flow
5. Caliptra’s ROM reads an internal register to differentiate b/w warm vs cold vs impactless flow. If it's a warm reset flow, then it skips DICE key gen, FW load flows (because keys were already derived and FW is already present in ICCM). This is an important reset time optimization for devices that need to meet the hot reset spec time.

Note: Cold reset flow is not explicitly mentioned but it would look like cold boot flow as Caliptra IP has no state through a cold reset.

Mailbox

The Caliptra Mailbox is a 128KB buffer used for exchanging data between the SoC and the Caliptra microcontroller (uC).

SoC side will communicate with the mailbox over an APB interface. This allows the SoC to identify the device using the interface to ensure that the mailbox, control registers, and fuses are read or written only by the appropriate device.

When a mailbox is populated by SoC, an interrupt to the FW to indicate that a command is available in the mailbox. The uC will be responsible for reading from and responding to the command.

When a mailbox is populated by the uC, we will send a wire indication to the SoC that a command is available in the mailbox as well as updating the MAILBOX STATUS register. The SoC will be responsible for reading from and responding to the command.

Mailboxes are generic data passing structures, we will only enforce the protocol for writing to and reading from the mailbox. How the command and data is interpreted by the FW and SoC are not enforced in this document.

Sender Protocol

Sending data to the mailbox:

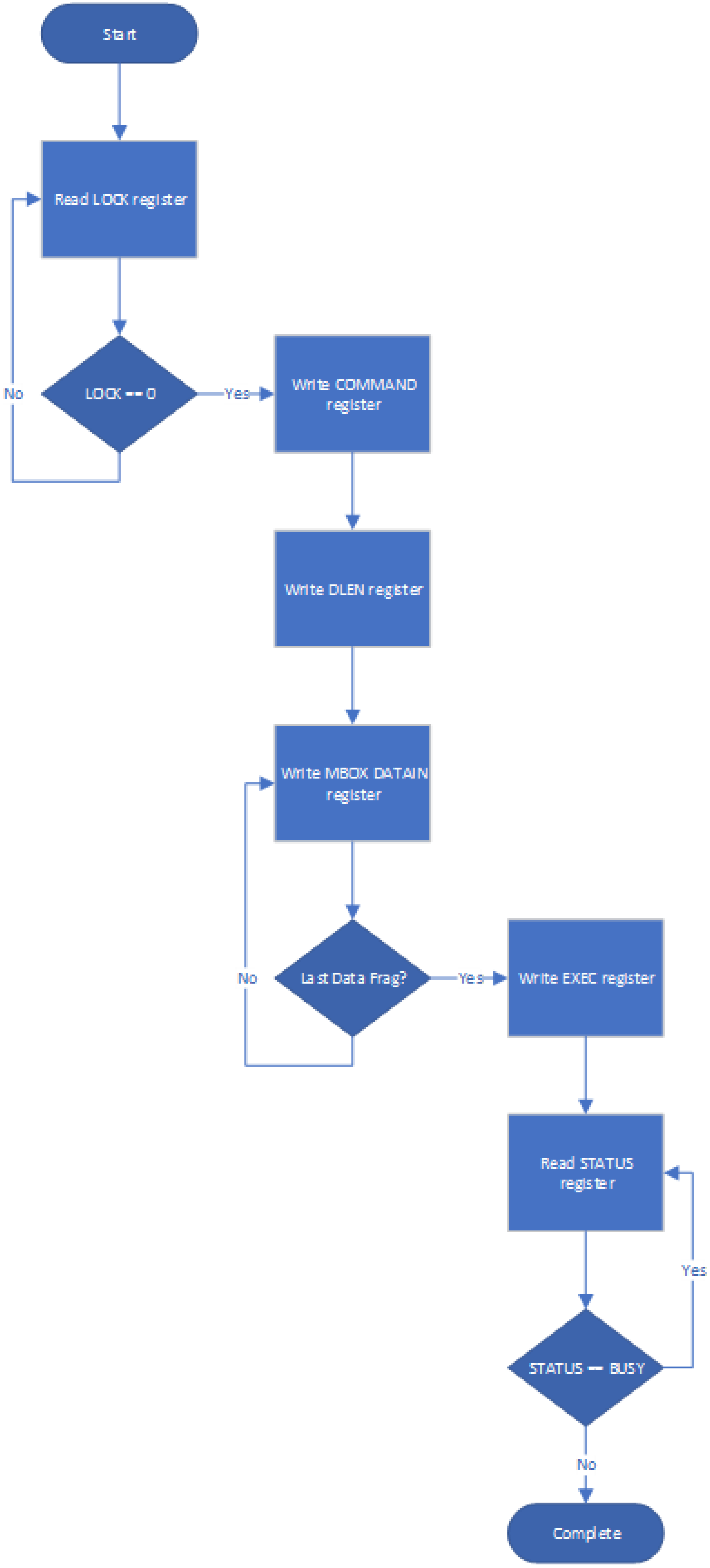
1. Requester queries the mailbox by reading the LOCK control register.
    a. If LOCK returns 0, LOCK is granted and will be set to 1.

    b. If LOCK returns 1, MBOX is locked for another device.

2. Requester writes the command to the COMMAND register.
3. Requester writes the data length in bytes to the DLEN register.
4. Requester writes data packets to the MBOX DATAIN register.
5. Requester writes to the EXECUTE register.
6. Requester reads the STATUS register.
    a. Status can return:
    b. DATA_READY – Indicates the return data is in the mailbox for requested command
    c. CMD_COMPLETE – Indicates the successful completion of the requested command
    d. CMD_FAILURE – Indicates the requested command failed
    e. CMD_BUSY – Indicates the requested command is still in progress

Notes on behavior:
Once LOCK is granted, the mailbox is locked until that device has concluded its operation. We should have a mechanism to terminate a lock early or release the lock if the device does not proceed to use it.

Mailbox is responsible for only accepting writes from the device that requested and locked the mailbox.

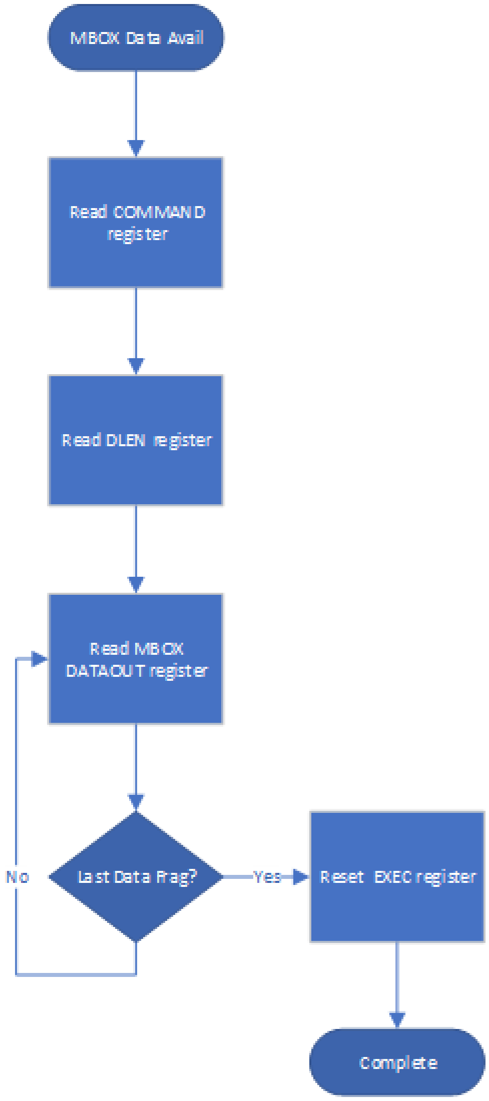

Start
Read LOCK register
LOCK == 0
No
Yes
Write COMMAND register
Write DLEN register
Write MBOX DATAIN register
Last Data Frag?
No
Yes
Write EXEC register
Read STATUS register
Yes
STATUS == BUSY
No
Complete

Receiver Protocol

Upon receiving indication that the mailbox has been populated, the appropriate device can read the mailbox. This is indicated by a dedicated wire that is asserted when Caliptra populates the mailbox for SoC consumption, also by the STATUS register returning DATA_READY

Receiving data from the mailbox:

1. Receiver reads the COMMAND register.
2. Receiver reads the DLEN register.
3. Receiver reads the MBOX DATAOUT register.
   3.1. Continue reading MBOX DATAOUT register until DLEN bytes are read.
4. Receiver resets the EXECUTE register.
   4.1. This releases the LOCK on the mailbox.



User Attributes

The PAUSER field of the APB interface will be used to encode device attributes for the requester utilizing the SoC interface. These values can be used for:

- Ensuring the device that was granted the LOCK is the one that accesses the mailbox, dlen, command, and status registers.
- Could be used to prioritize who is next granted the LOCK.

Architectural Registers

These registers are accessible over APB to be read according to the register permissions.
TODO: Additional registers are WIP,
TODO: Bit level definitions here or in integration spec?

MBOX FUNC	Address	SoC Permissions	Description
LOCK		RO	Refer to Mailbox (read/write protocol) for additional details.
COMMAND		RW	
DLEN		RW	
MBOX_DATAIN		WO	
MBOX_DATAOUT		RO	
EXECUTE		RW	
STATUS		RO	
HW_ERROR_FATAL		RO	
HW_ERROR_NON_FATAL		RO	
FW_ERROR_FATAL		RO	
FW_ERROR_NONFATAL		RO	
HW_ERROR_INFO_ENCODING		RO	
FW_ERROR_INFO_ENCODING		RO	
BOOT_STATUS_REGISTER		RO	
FLOW_STATUS_REGISTER		RO	

Table 8: SoC Registers

Fuse Requirements

- Fuse registers are programmable whenever IP goes through reset (after cptra_rst_b asserts & de-asserts) and before the fuse registers are locked from writes
- Some Fuse registers also carry additional attributes such as write-once. To rewrite such a fuse register, IP needs to go through power cycling (pwrgood assert & deassert)

To ensure that the security claims of a Caliptra RTM are achieved, specific Fuse capabilities must be supported:

- Fuses that hold Caliptra RTM secrets shall not be readable by any mutable code in the SOC
- If JTAG is enabled pre or post SoC reset, a Calpitra RTM's dedicated Fuses shall not be accessible (shall not be readable, shall not be writable) by a Caliptra RTM or other SoC IP. This restriction shall be applicable to a Caliptra RTM's Fuse shadow registers as well (refer to Physical Attack Countermeasures).
- SoC should ensure that the integrity of each fuse is maintained through the life of the part. The integrity of the fuses can be maintained by fuse redundancy, ECC or other means determined sufficient by the SoC.

Note: In addition to the Fuse bits described below, Fuse bits necessary to support ownership principles as described in [Reference 10] shall be supported.

The following table describes a Calpitra RTM's Fuse map:

Name	**Size (bits)**	**ACL**	**Description**
UDS Seed (Obfuscated)	384	ROM	DICE Unique Device Secret Seed. This seed is unique per device. The seed is scrambled using an obfuscation function.
Field Entropy (Obfuscated)	1024	ROM	Array of 4 32-byte seeds, field-programmable by the owner, used to hedge against UDS disclosure in the supply chain.
KEY MANIFEST PK HASH 0	384	ROM FMC RUNTIME	SHA-384 hash of Key Manifest Signing ECC P-384 Public Key

KEY MANIFEST PK HASH 1	384	ROM FMC RUNTIME	SHA-384 hash of Key Manifest Signing ECC P-384 Public Key
KEY MANIFEST PK HASH 2	384	ROM FMC RUNTIME	SHA-384 hash of Key Manifest Signing ECC P-384 Public Key
KEY MANIFEST PK HASH 3	384	ROM FMC RUNTIME	SHA-384 hash of Key Manifest Signing ECC P-384 Public Key
KEY MANIFEST PK HASH MASK	4	ROM FMC RUNTIME	One-hot encoded list of revoked Key Manifest PK Hash
KEY MANIFEST SVN	32	ROM FMC RUNTIME	Key Manifest security version number.
BOOT LOADER SVN	32	ROM FMC RUNTIME	Boot Loader security version number.
RUNTIME SVN	128	ROM FMC RUNTIME	Runtime Firmware security version number.
ANTI-ROLLBACK DISABLE	1	ROM FMC RUNTIME	Disables Anti-rollback support from Caliptra.
IDEVID CERT CHAIN	32768	ROM FMC RUNTIME	4 KB of fuse storage for Manufacturer IEEE IDEVID Certificate chain

Table 9: Caliptra RTM Fuse Map

Fuse Programming

All Fuse based cryptographic keying material and seeds (e.g. UDS Seed) shall be generated (on-chip or off-chip) per requirements described in [Reference 11].

Fuse Zeroing

When a cryptographic key (or their hashes) are retired/revoked, then its associated Fuse storage shall be zeroed. Since by default, unprogrammed Fuse bits are read as ‘0b’, zeroed in this context requires that all zero Fuse bits in a field be programmed to ‘1b’; Fuse bits already programmed to ‘1b’ must never be attempted to be programmed to ‘1b’. Zeroing a Fuse field is summarized as:

1. Update the associated revoked bit implemented in Fuse
2. Read the current Fuse field that requires zeroization
3. XOR the read field with a value of all 1s that is equivalent in length to the read field
4. Program the Fuse field with the XORed value

Error Reporting and Handling [0.8 release]

This section describes Caliptra error reporting and handling.

Condition	When occurs	Remediation
SoC FMC verification failure due to invalid digital signature invalid, invalid anti-rollback value)	During boot as described in Active Profile, Passive Profile, Secure Boot Flow	
Caliptra FMC invalid	During boot as described in Active Profile, Passive Profile, Secure Boot Flow	
Caliptra runtime FW invalid	During boot as described in Active Profile, Passive Profile, Secure Boot Flow	
Fuse programming errors		

Table 10: Errors and remedies

Appendix A - Checklist for IC approval of this Specification (to be completed by contributor(s) of this Spec)

Complete all the checklist items in the table with links to the section where it is described in this spec or an external document .

Item	Status or Details	Link to detailed explanation
Is this contribution entered into the OCP Contribution Portal?	Yes – contribution in the portal	N/A
Was it approved in the OCP Contribution Portal?	Yes	N/A
Is there a Supplier(s) that is building a product based on this Spec? (Supplier must be an OCP Solution Provider)	Yes	AMD
Will Supplier(s) have the product available for GENERAL AVAILABILITY within 120 days?	No	Silicon design, tapeout, and integration timelines exceed 120 days Please have each Supplier fill out Appendix B.

Appendix B - AMD - OCP Supplier Information and Hardware Product Recognition Checklist

(to be provided by each supplier seeking OCP recognition for a Hardware Product based on this specification)

Company: AMD
Contact Info: piotr.kwidzinski@amd.com

The product intercepts is beyond 180 days in the future and details of the products are still kept confidential by AMD until closer to release.

At this point the AMD products are not seeking any certification or recognition. It’s too early.

Appendix C - Contribution Process FAQs (unchanged from template)

As a contributor to a hardware specification, here are some questions that often come up.

Q1. What type of hardware specification am I contributing to OCP? Is it any of the below?

a. base specification for a de-facto standard (new standard with no hardware product on the horizon)
b. base specification for an intended physical <hardware product type> (product may be coming but within the next 1-2 years)
c. modification of an existing specification (state which existing spec is being modified)
   i. either a complete revision update or
   ii. a minor version update
d. design spec (based on an existing base specification) with more refined design details (product coming in 12-15months)
e. a detailed specification for a <hardware product type> for a very specific product being available in 3-6months of approval of this Spec
f. If none of the above, please contact OCP Staff for better direction.

Q2. How do I know if what I am contributing will be accepted by OCP?

a. Before contributing any specifications, please contact either OCP Staff (Archna Haylock or Michael Schill) or the Project Lead for the Project that best represents your contribution. For example, if you are contributing a Server Specification, please contact one of the Server Project Leads. You can see all the Projects here.
b. They will help you with your contribution and help you navigate the process.

Q3. What is the contribution process for my hardware spec?

a. Follow the flow for your spec type here.
b. This flow is subject to change so please check with the OCP Staff for more information or any questions.

Q4.What if my spec is not developed yet and I want to collaborate with other companies?

a. Please contact either OCP Staff (Archna Haylock or Michael Schill) or the Project Lead for the Project that best represents your contribution.
b. They will help you find other collaborators and help you with the contribution process for a multi-party contribution.

Q5.I have a question on the Contribution License Agreement.

a. Please contact OCP Staff and we can help you with questions.

Q6.Do I need to have a product in order to contribute a spec?

a. Please see Q1. Some types of specs do not require an immediate product. Some do. Please work with the OCP Staff on better direction on your specification type.