

The Silicon Anchor for Caliptra Root of Trust – YouTube

(A video by PUFsecurity)

Agenda

WiFi PSWD: **MSFTEXEC2642**

Time	Session	Presenters	Presentation + Q&A
8:30–8:40 AM	Introduction	Bryan Kelly	10 min, no Q&A
8:40–9:05 AM	ASPEED: World's First Silicon with Caliptra Subsystem	Dylan Huang	20 + 5 min
9:05–9:30 AM	AMD: How Did AMD Build Value for AMD Venice Leveraging Caliptra?	Sudhir Mathane	20 + 5 min
9:30–10:05 AM	Caliptra SS Stack, Tools & Verification, and Hands-on Development Pointers	Vishal Soni, Zach Halvorsen	30 + 5 min
10:05–10:30 AM	PUFsecurity: Caliptra Specification to Silicon Reality & a Turnkey Solution for Ecosystem Integrations	Andy Chen, Kent Chuang	20 + 5 min
10:30–10:45 AM	**Coffee Break**		**15 min**
10:45–11:00 AM	RFC and Contributions Process	Zach Halvorsen	10 + 5 min
11:00–11:25 AM	Marvell: Integrating a PUF into Caliptra, Hardening the Root of Trust	Craig Barner	20 + 5 min
11:25–11:50 AM	Network Boot/Recovery and Application to System BMCs	Dylan Huang	20 + 5 min
11:50 AM–12:15 PM	Caliptra Subsystem Fuse Controller & Debug: First Step Toward Securing the Supply Chain	Emre Karabulut	20 + 5 min
12:15–12:35 PM	**Lunch**		**20 min**
12:35–12:55 PM	SSD Ecosystem: An Update on the Caliptra 2.1 Subsystem & SSDs	Ulrich Hansen	15 + 5 min
12:55–1:20 PM	Confidential Compute & Caliptra in the World of AI	Roksana Golizadeh Mojarad	20 + 5 min
1:20–1:35 PM	Future Roadmap	Andreas, Bryan	10 + 5 min
1:35–2:00 PM	NXP: Embracing & Leading Hardware Contributions to Caliptra Ecosystem	Bart Vertenten	20 + 5 min



Introduction

RECOGNITION

Micron

I3C Validation & Collaboration

Caliptra

RECOGNITION

ASPEED

World’s First Silicon with Caliptra 2.2 Subsystem

Caliptra

RECOGNITION

NXP

Open-source USB to the open ecosystem

Caliptra

RECOGNITION

Flavien Salt

High-Quality Independent Security Research on Caliptra Subsystem

Caliptra

ASPEED AST1040 & Caliptra 2.1 Subsystem Integration

World’s First Silicon with Caliptra Subsystem

End-to-end I3C streaming boot, proven on real silicon | AST1040 target + AST2700 BMC host | Caliptra Workshop · Oct 12, 2026

© ASPEED Technology. Property and Confidential

SESSION SCOPE

ASPEED integration, proven on real silicon

01
Real silicon
Caliptra 2.1 SS runs on the actual AST1040 chip — not a model or FPGA.
Proven live today

02
End-to-end boot
AST2700 BMC streams the full boot path to AST1040 over I3C — host to Zephyr shell.
One continuous demo

03
ASPEED integration
I3C recovery target, runtime PLDM transport, and SoC boot handoff around Caliptra.
Focus of this talk

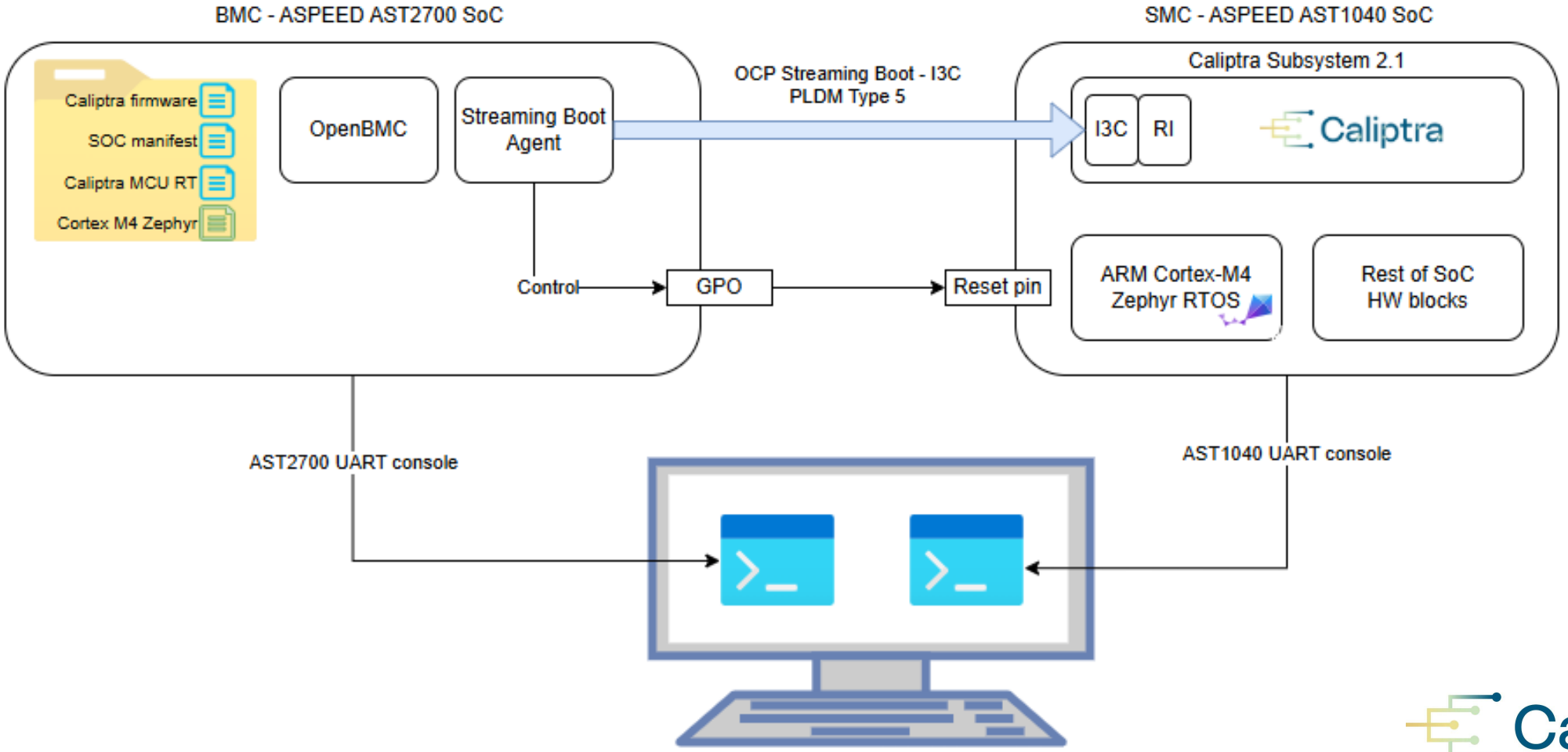
Generic Caliptra SS boot & RT FW stack were covered in the earlier session — here we focus on the ASPEED integration and its live proof.

Top-level view: what ASPEED built, and the on-hardware evidence that it works end to end.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

Demo setup connects AST2700 host to AST1040 target



© ASPEED Technology. Property and Confidential

Streaming Boot Agent runs in OpenBMC user space on the AST2700.

ASPEED INTEGRATION

AST1040 proves four boot-critical integration points

	Integration point	Path exercised in the demo	Observable evidence
1	**Native I3C recovery endpoint**	AST2700 streams Caliptra FW, SoC manifest, and MCU RT over OCP streaming boot / I3C.	**Host log: 3 FIFO transfers (182 / 34 / 248 KB)**
2	**ROM to runtime chain**	MCU ROM initializes I3C, reaches mailbox readiness, downloads firmware, then jumps to MCU RT.	**Target log: ROM -> MCU RT**
3	**Mailbox and authorization**	Caliptra accepts mailbox commands and authorizes the Cortex-M4 image before boot.	**Target log: authorization complete**
4	**Runtime SoC handoff**	PLDM Type 5 over MCTP / I3C delivers the Cortex-M4 Zephyr image and starts the CM4 boot path.	**UART: zephyr~$**

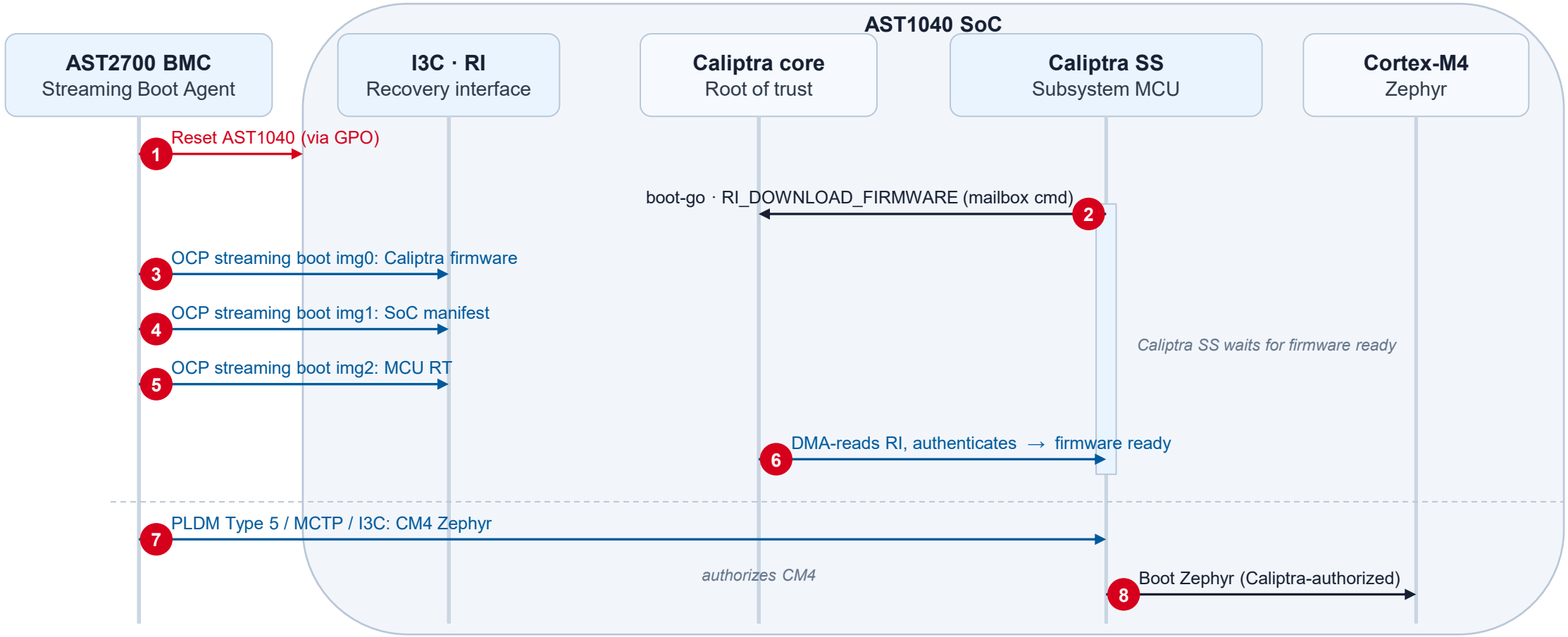
Topology, boot sequence, and live evidence for each checkpoint follow.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

BOOT SEQUENCE

Streaming boot flows through the I3C recovery interface



Phase 1 · OCP streaming boot (steps 1–6) Phase 2 · runtime PLDM (steps 7–8)

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

STREAMING BOOT

Two-phase delivery: OCP streaming boot, then runtime PLDM

PHASE 1 — OCP STREAMING BOOT · three boot images streamed over I3C

0x0
Caliptra firmware
Initial Caliptra FW payload streamed by AST2700.
182 KB · firmware ready

0x1
SoC manifest
Image metadata and trust inputs for authorization.
34 KB · manifest accepted

0x2
Caliptra MCU RT
Runtime that starts the post-boot service path.
248 KB · MCU RT running

PHASE 2 — RUNTIME PLDM

Phase 2 — PLDM Type 5 / MCTP over I3C: the BMC delivers the Cortex-M4 Zephyr image (~880 KB); Caliptra authorizes it before CM4 boot.

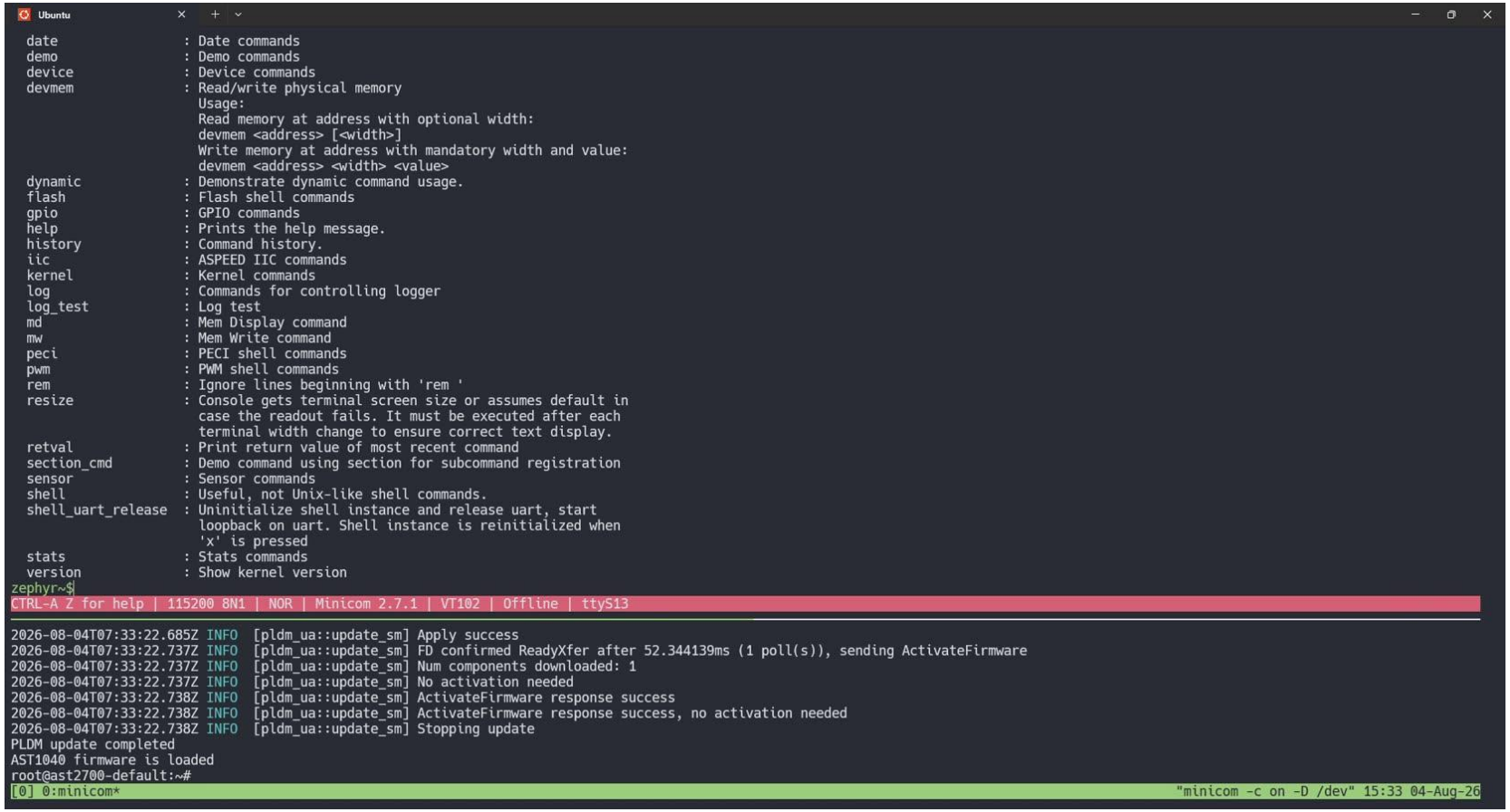
Streaming boot stages the trust base; PLDM then delivers and authorizes the application image — over the same I3C link.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

LIVE DEMO · RECORDING

See the streaming boot run end to end



AST2700 host (bottom) drives AST1040 (top): reset → OCP streaming boot → PLDM → Zephyr

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

LIVE DEMO · BMC HOST

AST2700 host: recovery and PLDM delivery

AST2700-I3C_boot.txt — BMC host console (OpenBMC)

```
$ /bin/run_demo.sh
Reset AST1040
Send AST1040 firmware via I3C recovery interface
[INFO] Device /dev/i3c-mctp-7 is opened
[INFO] Sending via FIFO: caliptra-fw_align_256.bin (186112 bytes)
[INFO] Sending via FIFO: ast1040-...-auth-manifest.bin (34560 bytes)
[INFO] Sending via FIFO: ssmcu-runtime_align_256.bin (253952 bytes)
[INFO] set-eid --dest-eid 0 0x08   ->  Target endpoint is now EID 0x08
Starting PLDM update: src EID=0x09, FD EID=0x08
[pldm_ua::update_sm] Transferred 1024 / 901496 bytes ...
[pldm_ua::update_sm]  ... streaming ~880 KB over I3C ...
[pldm_ua::update_sm] Transfer complete success
[pldm_ua::update_sm] Verify success
[pldm_ua::update_sm] Apply success
PLDM update completed
AST1040 firmware is loaded
```

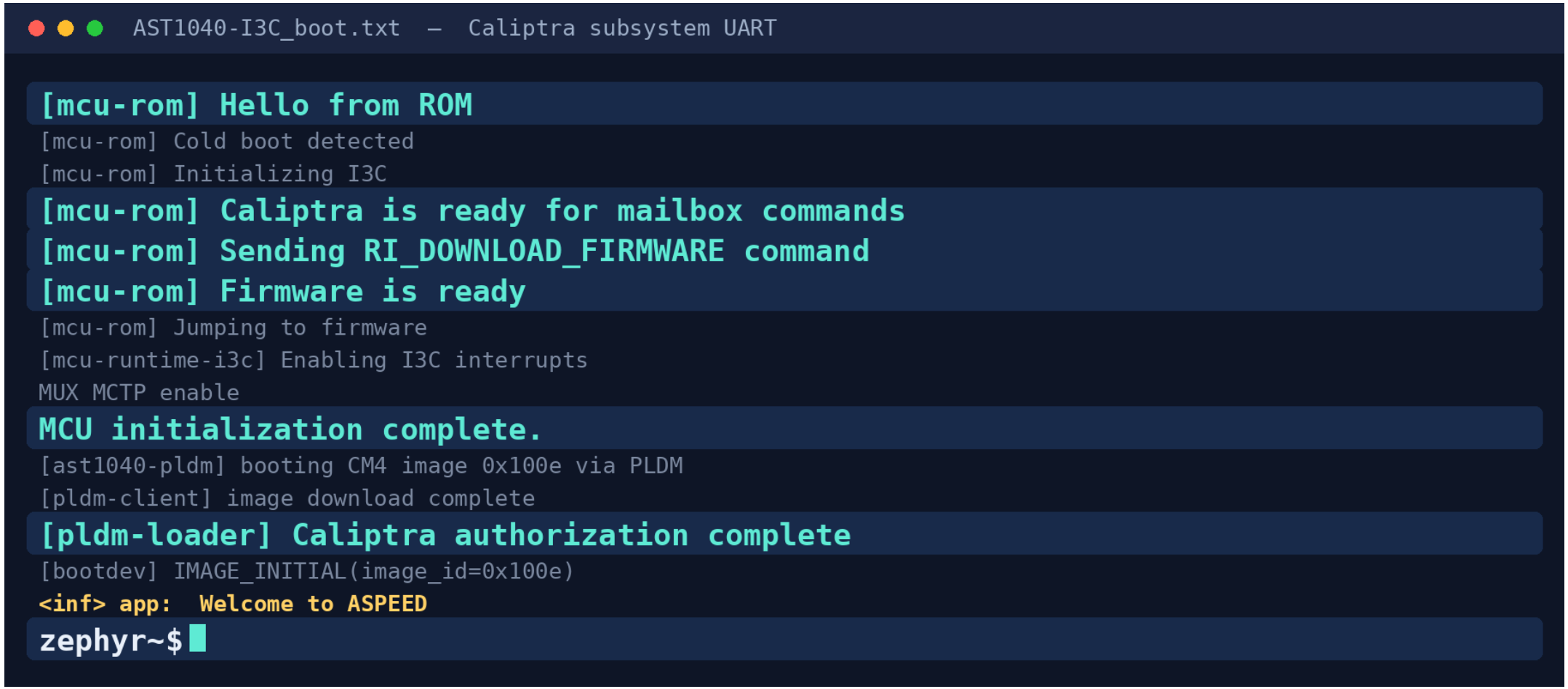
I3C SCL at 8 MHz (max 12.5) on purpose — each phase transition stays visible.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

LIVE DEMO · TARGET

AST1040 target: ROM to Zephyr shell



Target reaches the Zephyr shell — the Caliptra-authorized CM4 image booted.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

NEXT STEP · STREAMING BOOT AGENT

OpenBMC community is building the OCP recovery agent

OPENBMC COMMUNITY · gerrit.openbmc.org, topic "ocp-fw-recovery" · under review

LIBRARY

libocp

C++ library for OCP Secure Firmware Recovery v1.0: PROT_CAP, DEVICE_STATUS, RECOVERY_CTRL, INDIRECT_CTRL / DATA.

Protocol over a pluggable Transport

CLI TOOL

ocp-recovery-tool

Standalone BMC CLI: GetDeviceID, GetDeviceStatus, GetRecoveryStatus, PerformOCPRecovery.

Bring-up and manual recovery

UPDATE SERVICE

Recovery update daemon

phosphor-bmc-code-mgmt service: D-Bus Software.Update from a PLDM package; Entity Manager config.

Not in early integration yet

GAP FOR CALIPTRA 2.x SUBSYSTEM STREAMING BOOT

No I3C transport

Community Transport is I2C / SMBus only (/dev/i2c-N, ioctl I2C_RDWR). The Caliptra SS recovery interface is an I3C target.

No v1.1 INDIRECT_FIFO

libocp implements the v1.0 INDIRECT_CTRL window and rejects FIFO-only targets; Caliptra SS streams its images through the FIFO.

The earlier demo used our in-house agent. This integration adds the I3C and INDIRECT_FIFO support required by Caliptra SS.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

EARLY INTEGRATION · ASPEED I3C SUPPORT

ASPEED early integration adds the Caliptra I3C recovery path

BMC SOFTWARE STACK (AST2700)

ocp-recovery-tool PerformOCPRecovery, status queries — COMMUNITY + ASPEED

libocp Target OCP recovery protocol — COMMUNITY + ASPEED

I3CSysfsTransport Transport over sysfs — ASPEED

user ▲▼ kernel `/sys/bus/i3c/devices/<bus>-<pid>/ocp_recovery/{cmd,cmd_resp}`

ocp-recovery I3C driver binds DCR 0xBD — ASPEED

Linux I3C core + AST2700 I3C controller — LINUX

I3C bus

Caliptra SS I3C — two I3C devices

OCP recovery (DCR 0xBD)	MCTP (i3c-mctp)
Phase 1 · streaming boot	Phase 2 · runtime PLDM

ASPEED OPENBMC SDK · EARLY INTEGRATION

Base
Community libocp + ocp-recovery-tool (upstream review patchset 6); update daemon not yet

+ I3C transport
I3CSysfsTransport and --i3c option; finds the recovery device on a given I3C bus

+ v1.1 INDIRECT_FIFO
FIFO streaming for Caliptra SS: power-of-two chunks, up to 128 B

+ Multi-stage hand-off
Checks Recovery Pending / Awaiting Image between the three images

Kernel (ASPEED Linux)
ocp-recovery driver: binds DCR 0xBD; OCP recovery I3C framing (2-byte length + PEC)

Downstream snapshot is not the latest patchset — our feature work keeps tracking upstream.

© ASPEED Technology. Property and Confidential

EARLY INTEGRATION · BMC LOG

ocp-recovery-tool + ASPEED I3C driver on AST2700

bmc_demo.log — AST2700 BMC console (OpenBMC), excerpt

```
root@ast2700-default:~# echo 1 > /sys/bus/i3c/devices/14c2f000.i3c15/rescan
[   92.525921] ocp-recovery 15-fffe005a10a5: ocp-recovery bound; sysfs at 15-fffe005a10a5/ocp_recovery/
root@ast2700-default:~# cat /sys/bus/i3c/devices/15-fffe005a10a5/dcr
0xbd
root@ast2700-default:~# ocp-recovery-tool --i3c -b 15 PerformOCPRecovery -i caliptra-fw_align_256.bin \
      -i ast1040a0-default-auth-manifest.bin -i ssmcu-runtime_align_256.bin
using indirect FIFO mode
[1/3] writing 'caliptra-fw_align_256.bin' (186112 bytes) to CMS 0
  128/186112 bytes (0%)  …  186112/186112 bytes (100%)
waiting for device to accept next stage...
[2/3] writing 'ast1040a0-default-auth-manifest.bin' (34560 bytes) to CMS 0
  128/34560 bytes (0%)  …  34560/34560 bytes (100%)
waiting for device to accept next stage...
[3/3] writing 'ssmcu-runtime_align_256.bin' (253952 bytes) to CMS 0
  128/253952 bytes (0%)  …  253952/253952 bytes (100%)
recovery successful
```

Early integration run: kernel binds the recovery device by DCR 0xBD; ocp-recovery-tool streams all three images over I3C.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

EARLY INTEGRATION · RECORDING

ocp-recovery-tool: AST1040 ROM to MCU runtime



BMC (top): rescan (ENTDAA) adds the bus-15 recovery device, ocp-recovery-tool streams; AST1040 (bottom) reaches MCU RT.

© ASPEED Technology. Property and Confidential

Caliptra ASPEED

AST1040 is ready for Caliptra 2.1 integration —
the end-to-end I3C streaming boot path is proven on real silicon.

Next: align on customer integration requirements and production boot-flow.
Early integration: community libocp + ocp-recovery-tool, extended by ASPEED for I3C.

© ASPEED Technology. Property and Confidential



Building a Differentiated Trust Architecture with Caliptra

Sudhir Mathane – AMD

Caliptra

AMD

together we advance_

The landscape is shifting

Two parallel shifts are redefining what platform trust needs to do.

Cloud Service Providers

- Fleet-scale trust — automate admission, quarantine and recovery
- Heterogeneous TCB — CPU, GPU, NIC, DPU, BMC and firmware all matter
- Platform authenticity — turn platform evidence into automated policy
- Operational resilience — secure update, rollback protection and recovery
- Actionable attestation — evidence must drive policy, not just produce telemetry

Agentic AI Infrastructure

- Ephemeral identity — agents are short-lived and dynamically instantiated
- Delegated authority — who is the agent acting for, and with what privilege?
- Dynamic trust context — model, tools, data and execution environment keep changing
- Continuous authorization — trust may need reevaluation at every sensitive action
- Provenance & accountability — know which agent, platform and credentials caused an action

Security cannot be a one-time boot event. It becomes an operational control plane.

Caliptra
AMD
together we advance_

The ring model assumed static principals

Agentic AI is a new kind of principal — not static code, not a human. The ring model wasn't designed for it.

Conventional security primitives: Authentication, encryption, integrity, access control & attestation

Ring 3 → 0	**Agentic AI tenant** Short-lived. Delegated authority. **Mutates per action.**
Ring –1	**Hypervisor / virtualization layer** Workload isolation. Changes per deployment.
Ring –2	**System Management Mode (SMM)** Highly privileged. Difficult to observe.
Ring –3	**SoC firmware** Persistent platform control plane.
Ring –4	**Hardware Root of Trust** Stable. Minimal.

↑ *most change*

Traditional auth and attestation was designed for static code at a known privilege level.

Agentic AI changes identity, authority and context with every action.

↓ *most stability*

Caliptra
AMD
together we advance_

A different kind of trust problem

Static attestation is necessary — but no longer sufficient.

The old model

— Authenticate once at boot
— Trust the code statically
— Ring boundaries are fixed
— Evidence = snapshot at start

The new reality

— Principal changes per action
— Delegated authority is ephemeral
— Trust context mutates continuously
— Who is the agent, really?

How does Caliptra help

— Guard assets at Ring −4
— Strong access control behind RoT
— Agent uses assets without seeing them
— Transparent, open firmware adds the missing trust layer

Don’t fight agentic AI evolution at Ring 0-3. Guard what it needs at Ring −4. Let agentic AI stack operate without touching the assets directly.

Caliptra

AMD
together we advance_

Caliptra, one of its kind initiative



Reduce duplicated plumbing with common Root Of Trust
Build the platform-specific trust story via effective integration

Caliptra

AMD
together we advance_

EPYC AMD EPYC™ 9006 Caliptra Integration Philosophy

Common RoT ≠ common product.
The differentiator is the coherent platform trust architecture & boundary, evidence chain and operating model you build around it for desired scale.

Caliptra

UC 1 - Trustworthy execution

What Caliptra authenticates, measures and records across the boot sequence.

Caliptra ROM	*Immutable*
authenticates + measures ↓	
Caliptra FW	*Authenticated + measured.*
extends trust chain ↓	
SoC FW	*Trust chain extended.*
measures ↓	
UEFI / BIOS	*Measured. Reset images recorded.*
propagates ↓	
x86 Components	*Measured boot propagated.*

What this means

— Caliptra's authenticates and executes first mutable code.
— Each layer is either measured by Caliptra or by a component Caliptra already measured.
— BIOS reset images are recorded across reboots.
— The x86 measured boot can extend this chain all the way to the OS and hypervisor boundary.

BIOS reset image recording

Caliptra captures firmware images across reboots. If what runs today differs from what ran yesterday, you know.

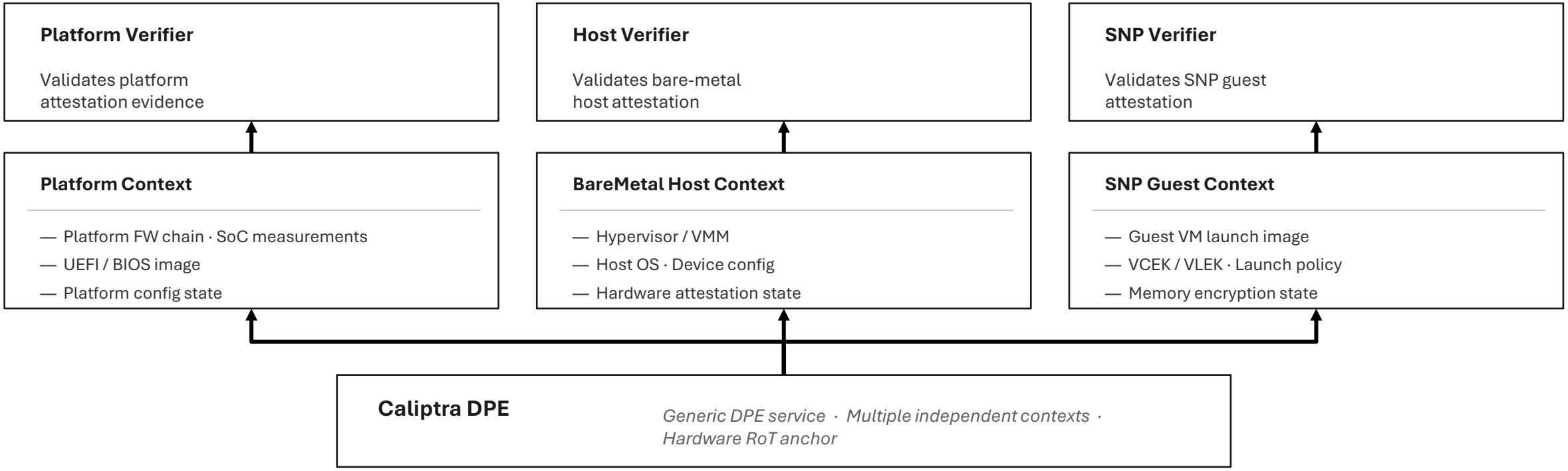
There are no trusted-by-assumption components. Every layer either carries a measurement or is one.

Caliptra

AMD
together we advance_

UC 2 - Trustworthy verifiability

One Caliptra DPE. Three contexts. Each capturing a distinct TCB.



One Caliptra DPE service. Three independently verifiable trust domains. Each verifier sees only its own TCB.

Caliptra

AMD
together we advance_

UC 3 – Owner authorization

The platform owner or the tenant defines what is allowed to run. Caliptra enforces it.

Provisioning

Platform Owner/Tenant

creates and provisions:

— Owner Auth Keys
— SoC FW Manifest

Owner Auth Keys →

SoC FW Manifest (authorized module hashes) →

Caliptra

Enforces:

— Owner Auth Keys
— SoC FW Manifest (hashes)

Boot time — Caliptra authorizes each firmware module against the manifest

Caliptra

SoC FW Manifest	Hash	Status
FW Module 1	3a9f1c8b...	✓
FW Module 2	b17c4a2e...	✓
FW Module N	—	✗

⟺

FW Module 1	*SoC ROM init*	✓ **Authorized**
FW Module 2	*Platform services*	✓ **Authorized**
FW Module N	*Unknown binary*	✗ **Rejected**

Sovereign-ready: Owner-controlled. Caliptra-enforced at every boot. No unauthorized firmware runs — regardless of who built or delivered the platform.

Caliptra
AMD
together we advance_

UC 4 – Delegated attestation authority

Owner diversifies entropy. Owner endorses keys. Attestation stays sovereign.

Caliptra — diversified attestation keys

platform owner fuses entropy portion into Caliptra → key derivation is uniquely diversified per deployment

Platform Owner

GET_CSR
Endorse with owner CA
IDevID → AMD CA → Owner CA

Tenant

GET_CSR
Endorse with tenant CA
IDevID → AMD CA → Tenant CA

SNP Guest

GET_CSR
Endorse with guest CA
IDevID → AMD CA → Guest CA

Platform Owner + Tenant

SPDM Path

- → Load endorsed cert chain into SPDM responder
- → SPDM requester retrieves attestation quote
- → Quote carries delegated owner / tenant CA

SNP Guest

SNP Path

- → Load endorsed cert chain into SNP attestation framework
- → Guest retrieves SNP attestation report
- → Report carries delegated guest CA

Sovereign-ready: Own the SoC before it reaches production — not just operate it. Supply chain security built in. Attestation sovereign by design.

Caliptra
AMD
together we advance_

UC 5 – Trust but verify

Three evidence streams. One verifier. Verification plus auditability.

Venice / Caliptra

Attestation Quote

— Signed by Caliptra (HW-rooted)
— Hash of each FW module at boot
— Freshness + lifecycle state

AMD

Reference Measurements

— Approved hash per FW module
— Published per firmware version
— Signed by AMD

OCP SAFE Auditors

Short Form Reports (SFR)

— Hash of audited FW module
— Security health assessment
— Published per module + version

Verifier

Quote hash : Reference hash : Audited hash

Trusted + Auditable deployment

The SFR hash is the link: same hash in the quote, the reference and the audit report — not just verified but independently audited.

Caliptra
AMD
together we advance_

Integration Best Practices

Integrator reality: the security value lives in the seams

A clean block integration is necessary. A coherent platform integration is what customers actually consume.

Reset • clock • power

Who comes up first? What is trusted before Caliptra is ready?

Boot • update • recovery

What is authenticated, measured, recoverable and rollback-protected?

Lifecycle • fuses • debug

Manufacturing, production, service and failure states must be intentional.



Host interface • mailbox

Ownership, concurrency, timeouts and error semantics become security contracts.

Measurements • attestation

Claims must map cleanly to platform state and customer policy.

Manufacturing • PSIRT

Provisioning and vulnerability intake are part of product security, not “after launch.”

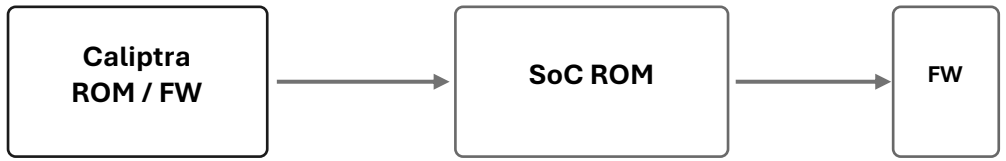
Best practice #0: create a written integration contract for every seam.

Caliptra AMD together we advance_

Best practice #1 — define the trust boundary before writing RTL

Passive MODE

SoC retains more platform boot / update responsibility

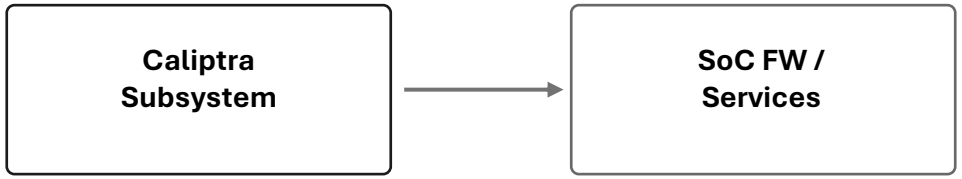


Decide explicitly:

- Which early state must be stashed/measured?
- Who owns update and recovery?
- When is attestation valid or invalid?

SUBSYSTEM MODE

Caliptra becomes a broader RoT service for the SoC



Decide explicitly:

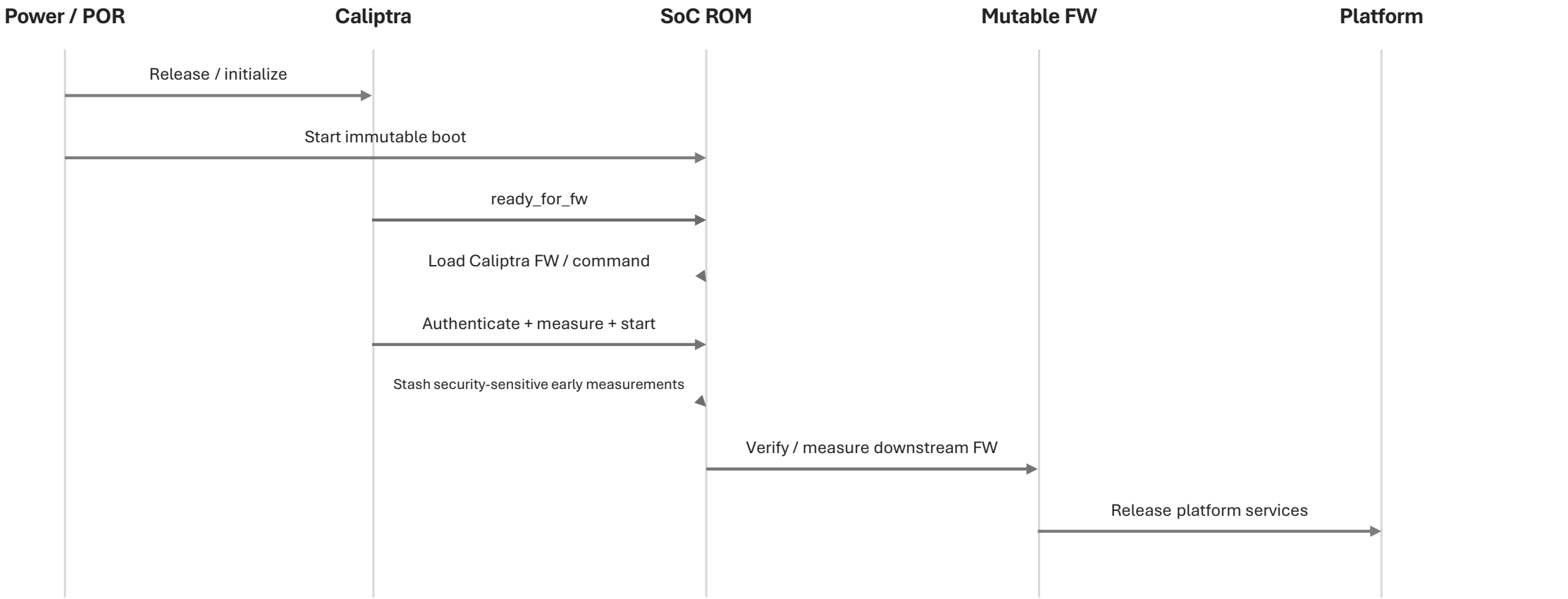
- Service boundaries and API ownership
- Recovery / streaming-boot behavior
- Crypto service and key-release policy

Anti-pattern: "we will decide the TCB later." By then, the boot path and lifecycle architecture have already decided it for you.

Caliptra
AMD
together we advance_

Best practice #2 — treat boot choreography as security architecture

Reset ordering, measurement timing and failure behavior are part of the trust claim.

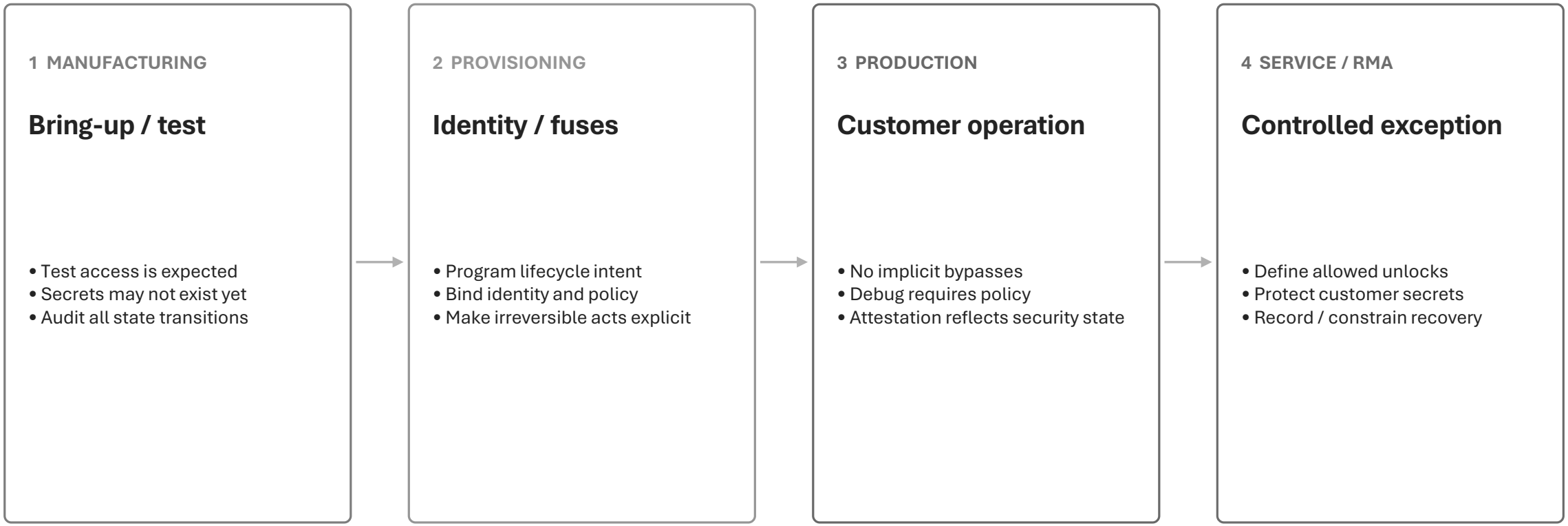


Design the negative path with equal rigor: timeout • bad signature • SVN rollback • measurement overflow • mailbox contention • reset during update

Caliptra
AMD
together we advance_

Best practice #3 — make LCS, DFT & debug part of the same model

A production RoT is only as strong as the least-controlled path around it.



Three invariants to verify across every lifecycle state

Secrets Are roots and derived keys inaccessible through test/debug paths?

Execution Can DFT/DFD redirect the RoT into an unplanned flow?

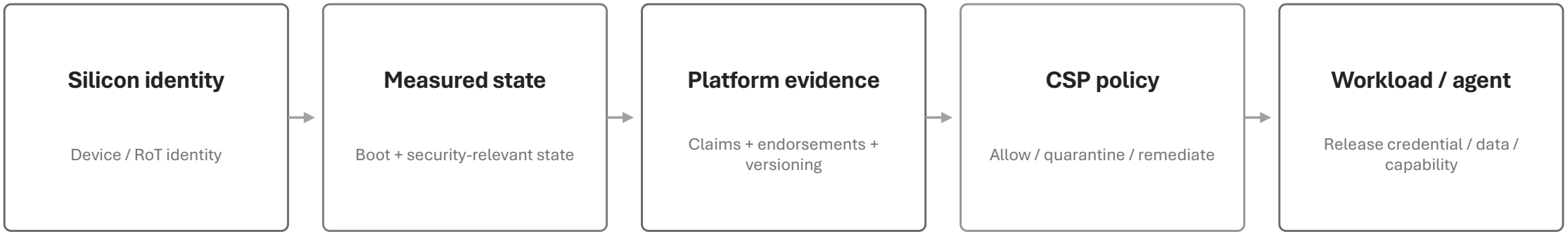
Evidence Does attestation accurately expose the security/debug state?

Caliptra

AMD
together we advance_

Best practice #4 — turn attestation into a product contract

The valuable output is not a quote. It is an evidence chain a customer can automate against.



Integrator checklist

Stable claim semantics	Versioned collateral	Failure semantics	Policy action
What does each measurement actually prove?	Can customers map evidence to an approved reference?	What happens on missing, stale or partial evidence?	Which automated decisions are safe to make?

Attestation without a defined policy action is telemetry. Attestation with stable semantics becomes infrastructure.

Caliptra

AMD
together we advance_

Best practice #5 — make upstream security part of product lifecycle

Open source reduces opacity. It does not remove the integrator's ownership.



Best integrators treat Caliptra like a security dependency with a product owner, a reproducible bill of materials, and a vulnerability-response path.

Caliptra AMD together we advance_



Caliptra Workshop - October 2026

Firmware: Core & MCU

Zach Halvorsen & Vishal Soni

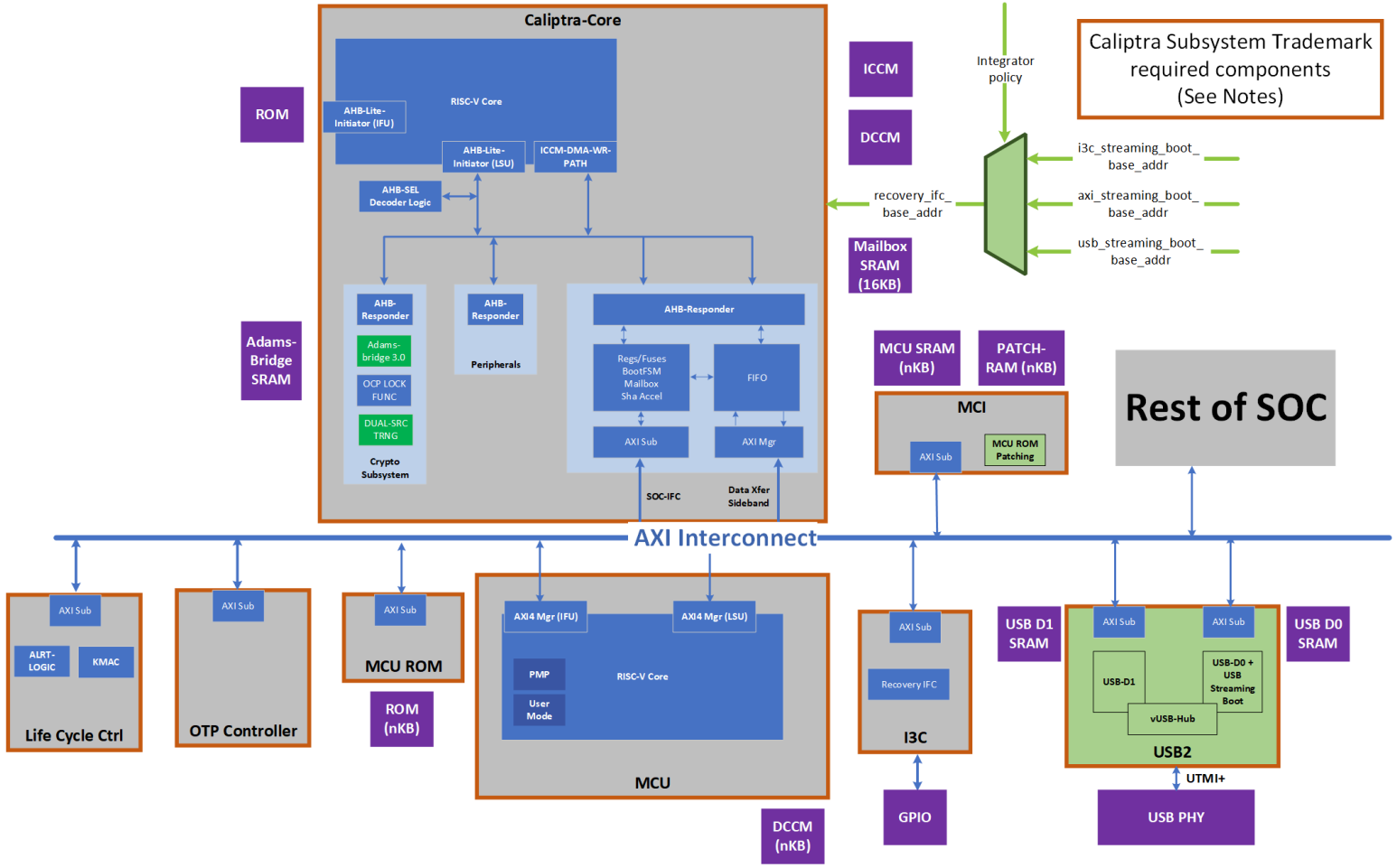
October 12th, 2026

Workshop agenda

Architecture and Firmware

- Subsystem features and design principles
- Caliptra Core and MCU responsibilities
- SDK services and integrator extensibility
- SPI flash, I3C and USB streaming boot
- Two-phase boot and runtime handoff
- Environment setup, emulators and tests

Caliptra Subsystem



Caliptra Subsystem Features

Caliptra 2.x security: **ML-DSA & ML-KEM** with Adams Bridge, DICE extensions, AES, PQC key-vault extensions and PQC PCR signing.

Subsystem support: **AXI DMA** assist, manufacturing and product debug unlock, UDS programming and streaming boot.

Lifecycle and **fuse controller** integration.

OCP streaming **boot over I3C**.

Design Principles

Open and extensible

Open development on GitHub. An SDK for integrator-specific RoT applications.

Secure and safe

Security and isolation practices. Memory-safe development in Rust.

Consistent

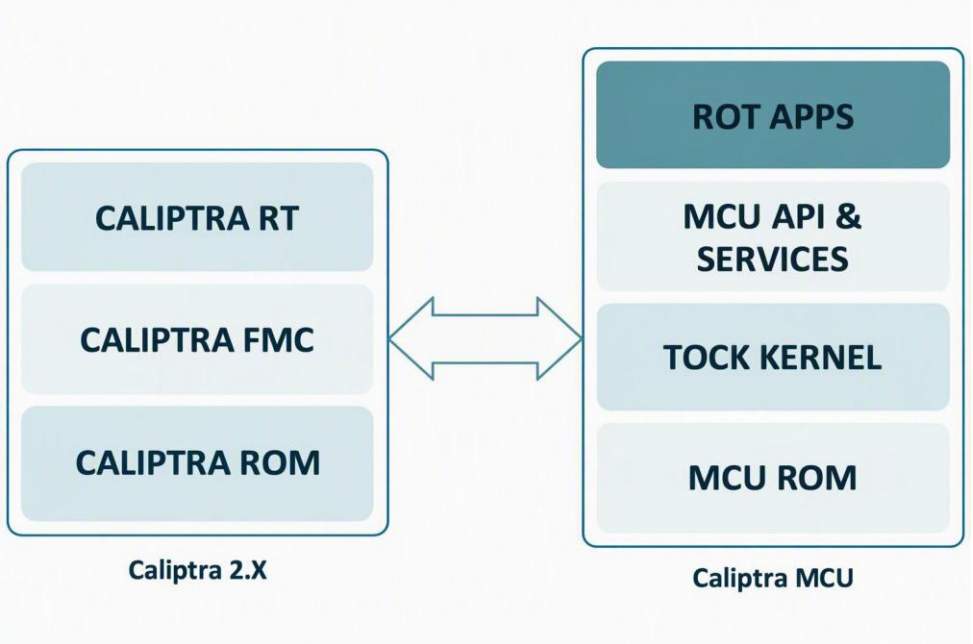
Common secure boot, measured boot, attestation, recovery and streaming boot. Standards include TCG, DMTF, OCP and PCIe.

Compliance objectives

OCP SAFE, FIPS 140-3 readiness and the Caliptra trademark.

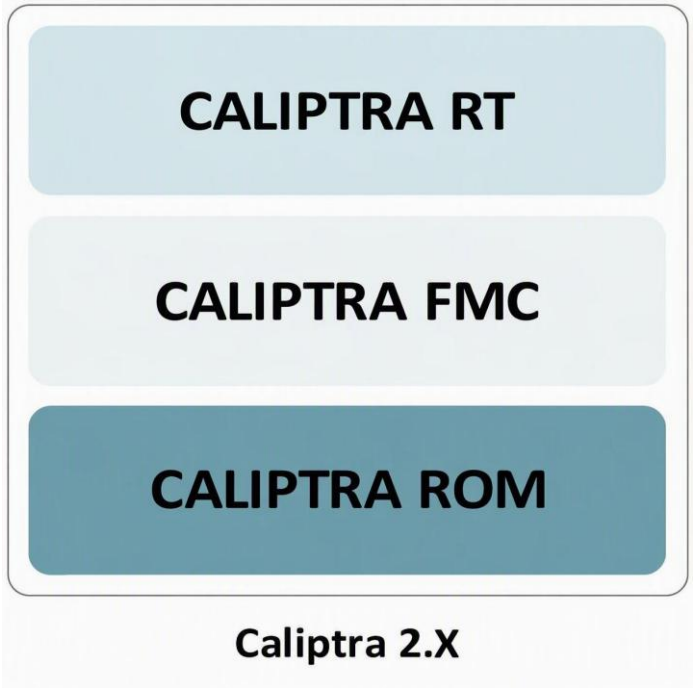
Subsystem Stack

- Combination of Caliptra 2.0 and MCU
- Caliptra 2.0 provides the security services
- Caliptra MCU provides the RoT services for the SoC and Platform

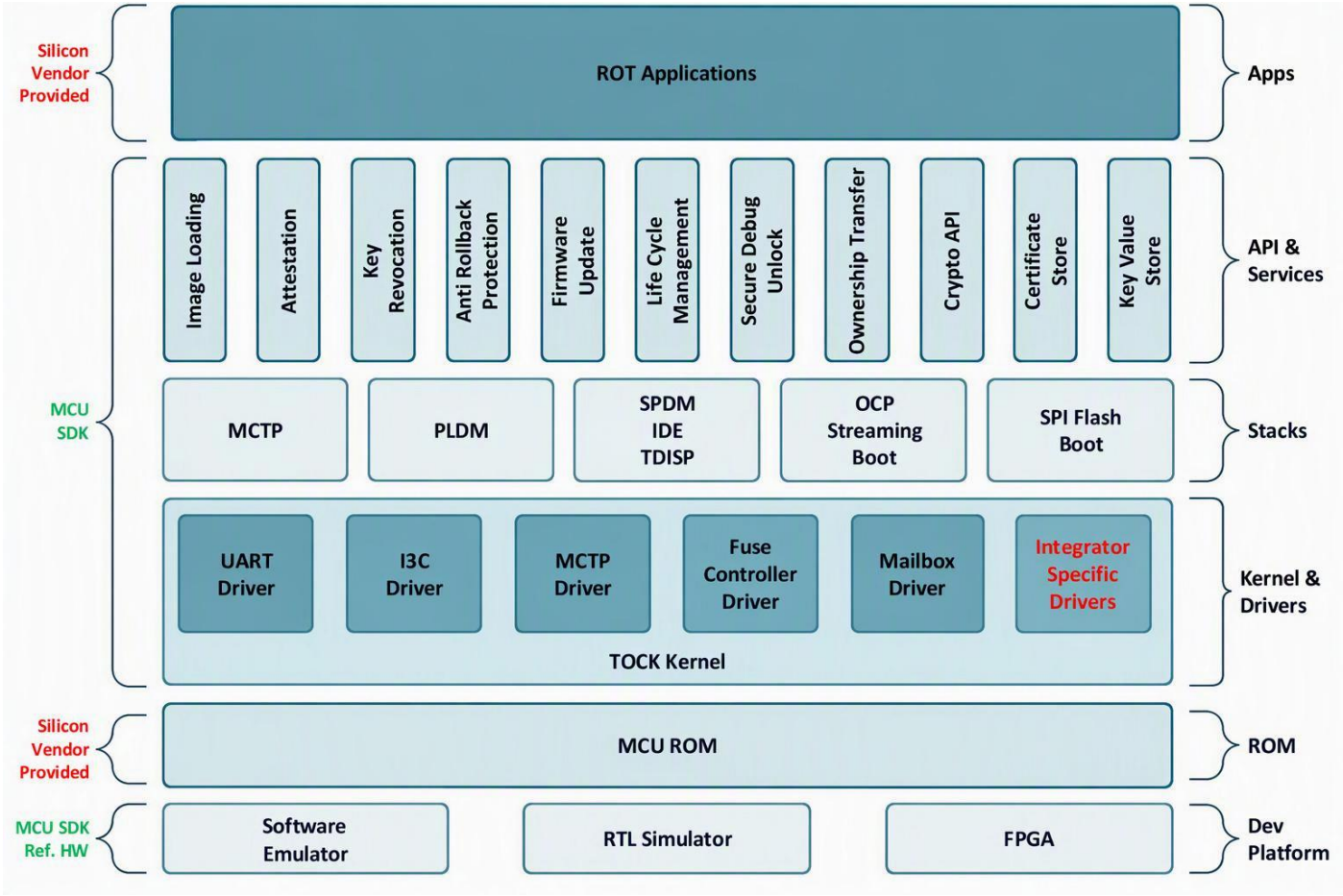


Caliptra 2.x

- Caliptra 2.x comprises of consortium governed ROM, FMC and Runtime
- Caliptra 2.x provides foundational security features for MCU
  - Quantum Resilient DICE
  - OCP Recovery Boot
  - Classical and Post Quantum Cryptography API support
  - Debug Unlock support
  - Quantum Resilient DPE



MCU Stack


Silicon Vendor Provided
ROT Applications
Apps
Image Loading
Attestation
Key Revocation
Anti Rollback Protection
Firmware Update
Life Cycle Management
Secure Debug Unlock
Ownership Transfer
Crypto API
Certificate Store
Key Value Store
API & Services
MCU SDK
MCTP
PLDM
SPDM IDE TDISP
OCP Streaming Boot
SPI Flash Boot
Stacks
UART Driver
I3C Driver
MCTP Driver
Fuse Controller Driver
Mailbox Driver
Integrator Specific Drivers
TOCK Kernel
Kernel & Drivers
Silicon Vendor Provided
MCU ROM
ROM
MCU SDK Ref. HW
Software Emulator
RTL Simulator
FPGA
Dev Platform

MCU SDK

- Development Tools
  - Emulators, RTL Simulators & FPGA
- Reference MCU ROM
  - Reference MCU ROM implemented in Rust
  - Illustrates integration b/w Caliptra 2.x and MCU
- Tock Kernel
  - Provides user/kernel mode isolation via RISCV MPU
  - Driver Model for extensibility

- Stacks & API
  - MCTP & PLDM
  - PLDM Firmware Update
  - OCP Streaming Boot
  - SPDM 1.4 based Attestation and Secure Sessions
  - PCIe IDE & TDISP support
  - Measurement API to support SoC measured boot
  - SPI Flash Boot
  - Image Loading
  - Firmware Update infrastructure
  - Ownership Transfer

Integrator Extensibility

- ROM Development
  - MCU ROM is integrator specific
  - Ability to extend the reference Rust ROM to build MCU-specific ROM
- Drivers
  - Flexible & async driver model provided by Tock kernel
  - SoC specific drivers (UART, SPI, IDE/TDISP, etc.)
- SOC API & Stacks
  - Ability for SOC to provide their own stack and user mode interfaces
- RoT Applications
  - Allows integrators to synthesize ROT applications specific to their use cases
  - Allows extensibility points for integrator specific message handlers for external communication
  - Enables running of legacy C code
  - Allows multiple application isolation via Tock kernel & MMU

Upcoming Enhancements

01 In-Field Provisioning

Enhancing authorized command support to enable secure, flexible, and robust provisioning directly in field deployments.

Next steps to include better vendor and owner flexibility with plans to intercept the next version of the SPDM authorization specification (DSP0289).

02 OCP EAT Attestation

Expanding OCP Entity Attestation Token (EAT) support to strengthen platform identity and remote attestation trust chains with a uniform solution.

03 Caliptra 2.2 Integration

Upgrading ROM and firmware support to include Caliptra 2.2 hardware to leverage next-generation Silicon Root of Trust security architectures.

04 Modularity & Extensibility

Streamlining component modularity to reduce dependencies and boost system stability, while providing clean APIs to easily empower third-party vendor extensions.

Compliance and trademark plans

Planned compliance targets:

Caliptra Trademark

OCP SAFE

FIPS 140-3

Planned support for running the FIPS ACVP test suite.



SPI Flash Boot

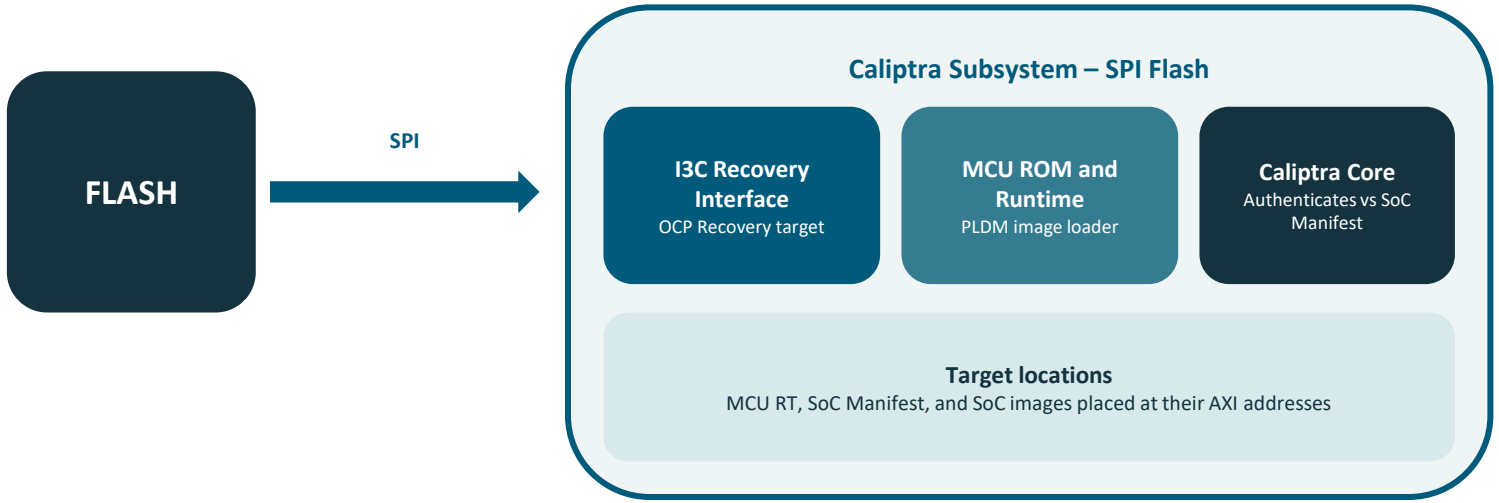
- Supported Topologies
  - ➢ Single flash: one device split into primary (active image) and recovery halves
  - ➢ Dual flash: two devices on separate chip selects - flash 0 primary, flash 1 recovery

Scaling and Partitions

➢Multi flash: multiple controllers and devices for scale and redundancy

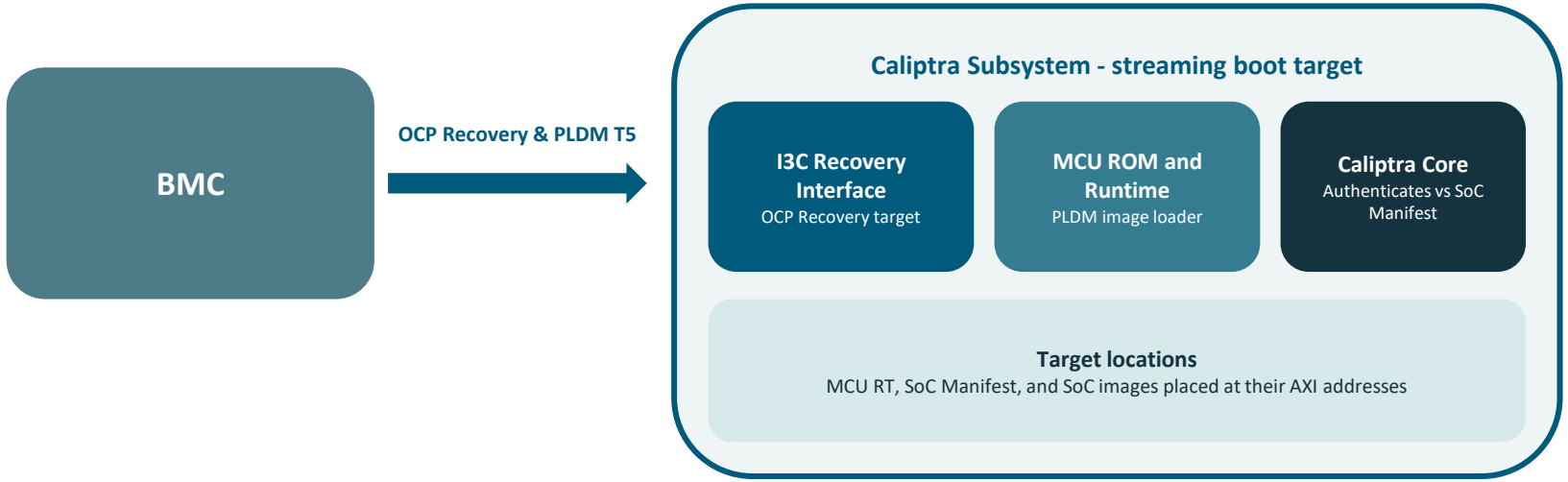
➢Primary flash may also hold staging, certificate, and log partitions

➢Recovery flash always carries a known-good image



I3C Streaming Boot

- Recovery Interface
  - I3C interface acts as a target device implementing the OCP Recovery specification
  - MCU ROM initializes I3C and the recovery interface before releasing Caliptra from reset
- Streaming the Images
  - A recovery agent, typically a BMC, streams the PLDM T5 package over I3C
  - AXI streaming bypass lets the MCU write images directly, skipping I3C protocol logic



USB streaming boot: roles and setup

USB host and recovery device

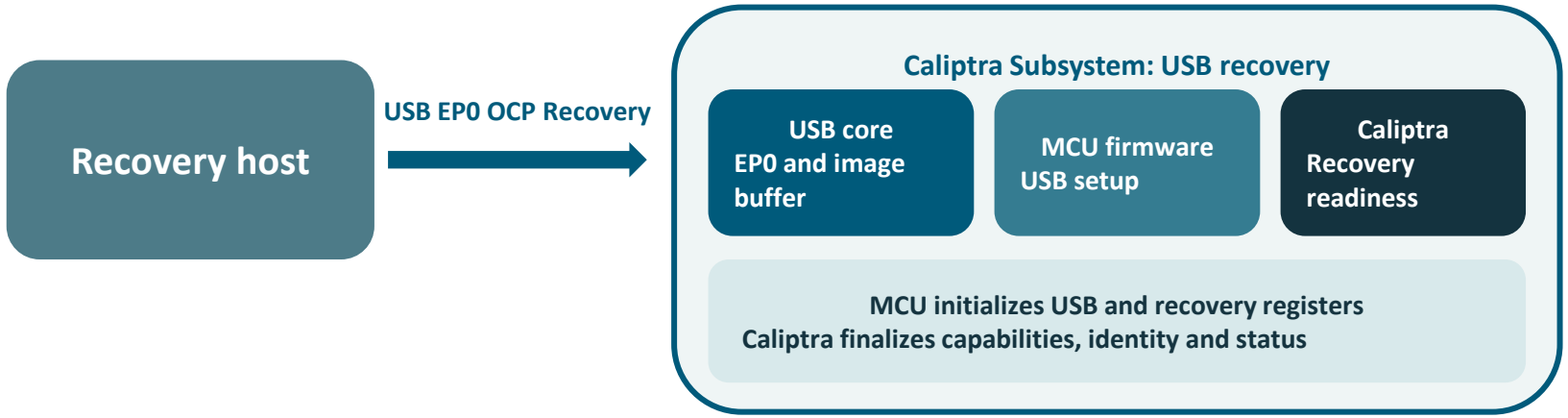
The host enumerates the USB device, discovers recovery capabilities and pushes recovery images.

The USB core exposes OCP Recovery commands on EP0, buffers image data and reports status.

MCU and Caliptra firmware

MCU firmware initializes USB, installs descriptors and completes standard enumeration with placeholder recovery registers.

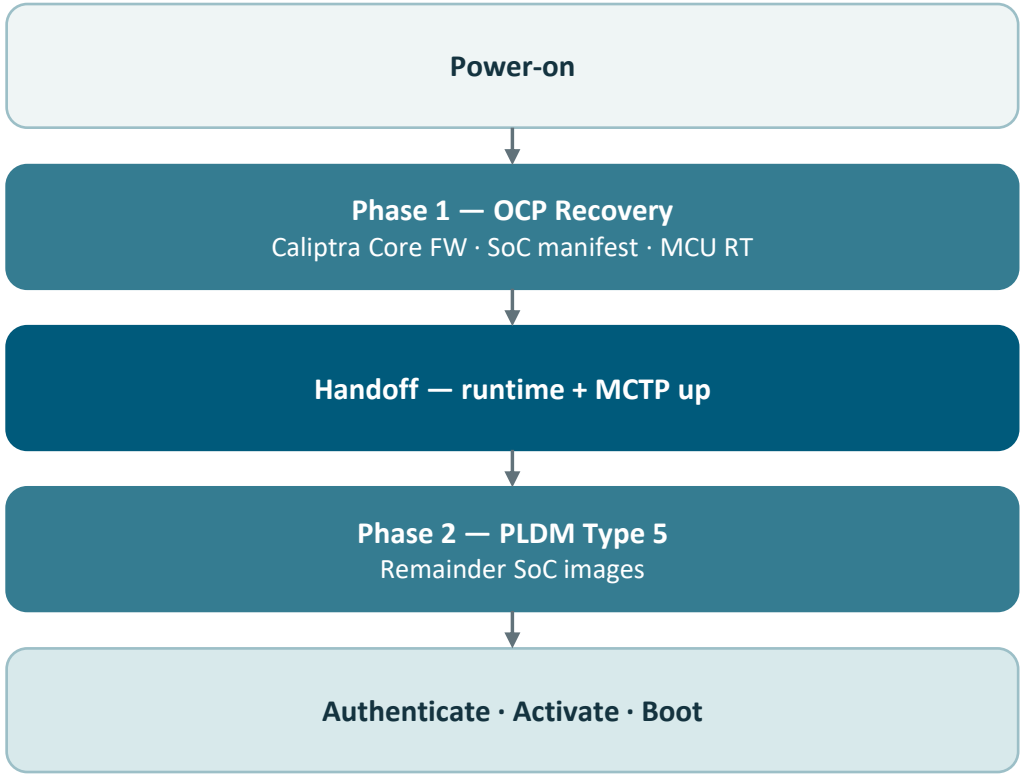
Caliptra finalizes capability, identity and status registers when online, enabling host recovery discovery.



Two-phase streaming boot

The BMC establishes the trusted runtime, then the live MCU pulls the rest.

- Phase 1 — OCP Recovery: BMC pushes an ordered image set (Caliptra FW, SoC manifest, MCU runtime) over the register interface. No MCTP/PLDM/runtime yet.
- Handoff: MCU runtime starts, exposes an MCTP endpoint; bus ownership changes.
- Phase 2 — DMTF PLDM Type 5: MCU pulls remaining SoC images via RequestFirmwareData; BMC acts as Update Agent.
- Caliptra authenticates and gates every stage — streaming boot is the normal boot path.



Getting started with firmware

01 Install Linux or WSL host tools

02 Clone the repository and submodules

03 Check the development environment

04 Run the Core and MCU emulators

05 Run emulator integration tests

Step 1: Install the host tools

What to do

- Commands target Linux or WSL.
- Install git, curl, and a C build toolchain with your OS package manager.
- Install rustup, which manages the Rust compiler and Cargo; choose the default option.
- Open a new terminal or load Cargo into the current shell, then confirm the version.

Command / example output

```
$ sudo apt update
$ sudo apt install -y build-essential curl git
$ curl --proto '=https' --tlsv1.2 -sSf \
    https://sh.rustup.rs | sh
$ source "$HOME/.cargo/env"
$ cargo --version
```

Example output varies by repository revision and environment.

Step 2: Clone the repository

What to do

- Clone with --recursive so the Caliptra hardware submodule comes down too.
- Confirm submodules with git submodule status --recursive; each initialized line begins with a space.

Command / example output

```
$ repo=https://github.com/chipsalliance
$ git clone --recursive \
    "$repo/caliptra-mcu-sw.git"
$ cd caliptra-mcu-sw
$ git submodule status --recursive
```

Caliptra

Step 3: Check the environment

What to do

- From the repository root, confirm the pinned compiler with rustc --version.
- Optionally run the full pre-submit check: cargo xtask precheckin.
- It builds firmware and runs format, lint, unit, and integration checks, so the first run takes several minutes.

Command / example output

```
$ rustc --version
$ cargo xtask precheckin
$ echo $?
0
```

Example output varies by repository revision and environment.

Step 4: Run the emulators



Step 5: Run the emulator tests

What to do

- Run cargo xtask test to build the required firmware and run the integration tests on the MCU emulator through cargo nextest.
- The run succeeds when it exits with status 0 and the summary reports no failed tests.
- Output below is from one emulator-test shard of a successful upstream run.

Command / example output

```
$ cargo xtask test

Example shard result:
146 tests run: 146 passed
1144 skipped

Example second shard:
2 passed, 217 skipped
```

Example output varies by repository revision and environment.

References

https://caliptra.io/

Specs

- Main Caliptra specification 2.0
- Caliptra Subsystem Hardware Specification
- ROM 2.x Specification
- FMC 2.x Specification
- Runtime 2.x Specification
- MCU Firmware and SDK specification



Questions

Please join us in co-developing an industry compliant RoT SDK

Zach Halvorsen & Vishal Soni

caliptra.io

From Caliptra Specification to Silicon Reality

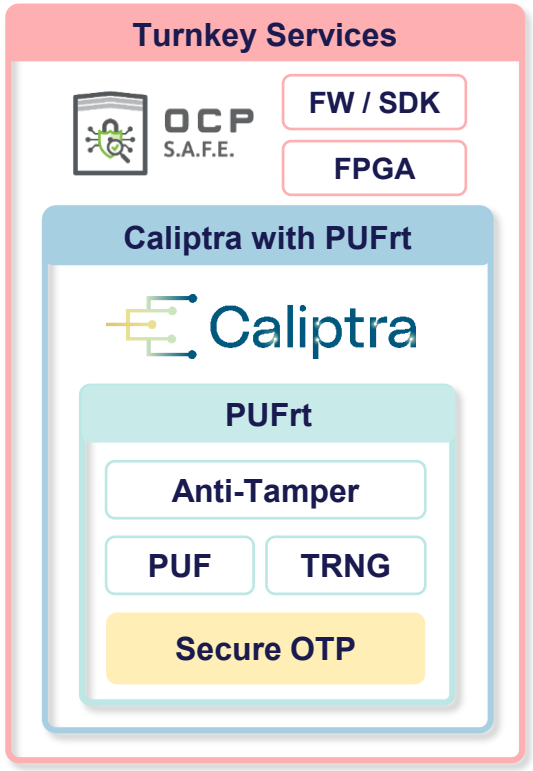
Lessons Learned with Ecosystem Partners

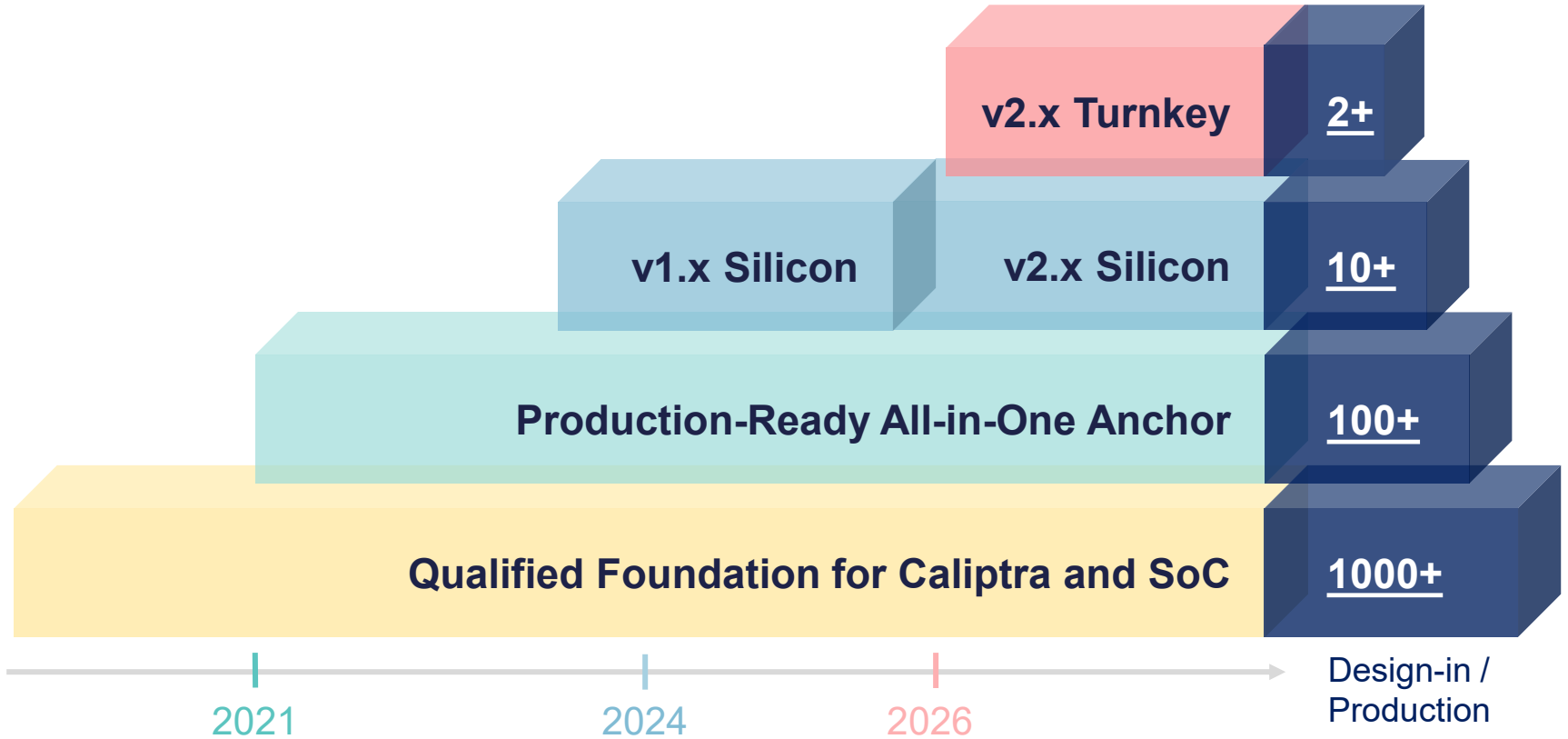


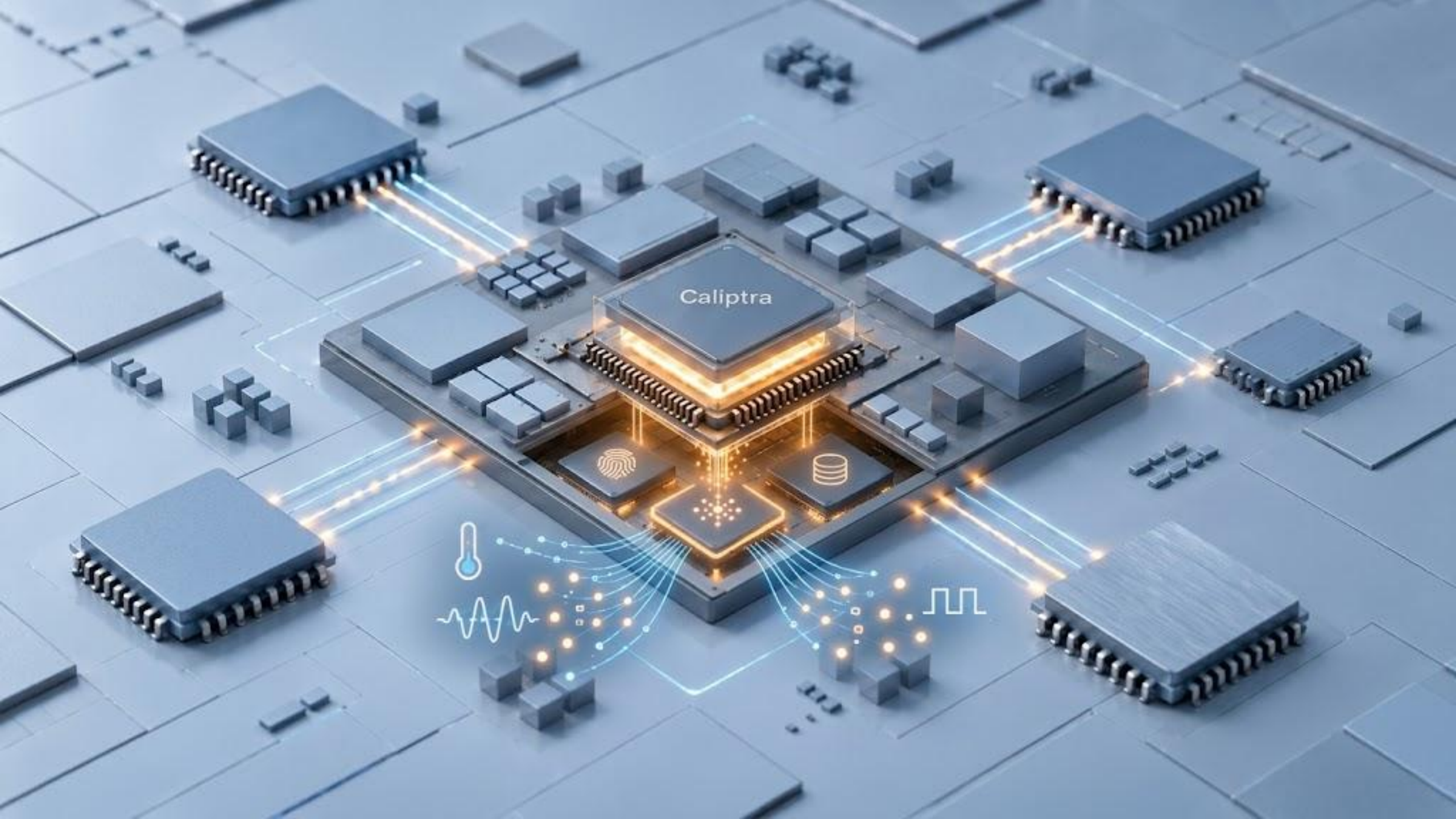


From IP to Services: Evolution with Caliptra

- PUFsecurity has a strong track record providing security solutions to the industry
- From foundational security IPs up to turnkey services to accelerate Caliptra adoption






Caliptra

Agenda

01 Caliptra Implementation: Lessons Learned

02 What we offer: IP and Turnkey

03 Benefits: Compliance and Time-to-Market

04 Industry Adoptions and Takeaways

Agenda

01 Caliptra Implementation: Lessons Learned

02 What we offer: IP and Turnkey

03 Benefits: Compliance and Time-to-Market

04 Industry Adoptions and Takeaways

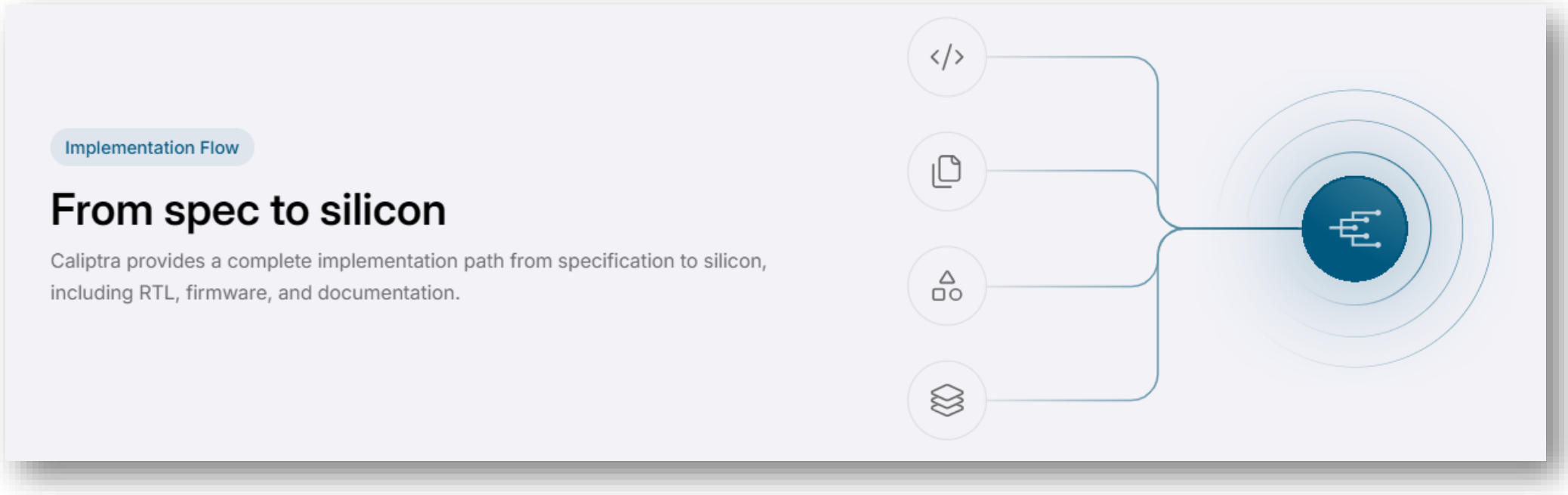
From Open Specification to **Real Silicon**

Open-source offering

- Caliptra RTL and software
- Standard verification flow
- Reusable OCP S.A.F.E. certification

More are needed in real silicon

- **Entropy** and **immutable storage**
- Vendor-specific verification flow
- Certification & **Caliptra Trademark**



Ref. https://chipsalliance.github.io/caliptra-web/#

Considerations for **Physical Implementation**

Unique Device Secret

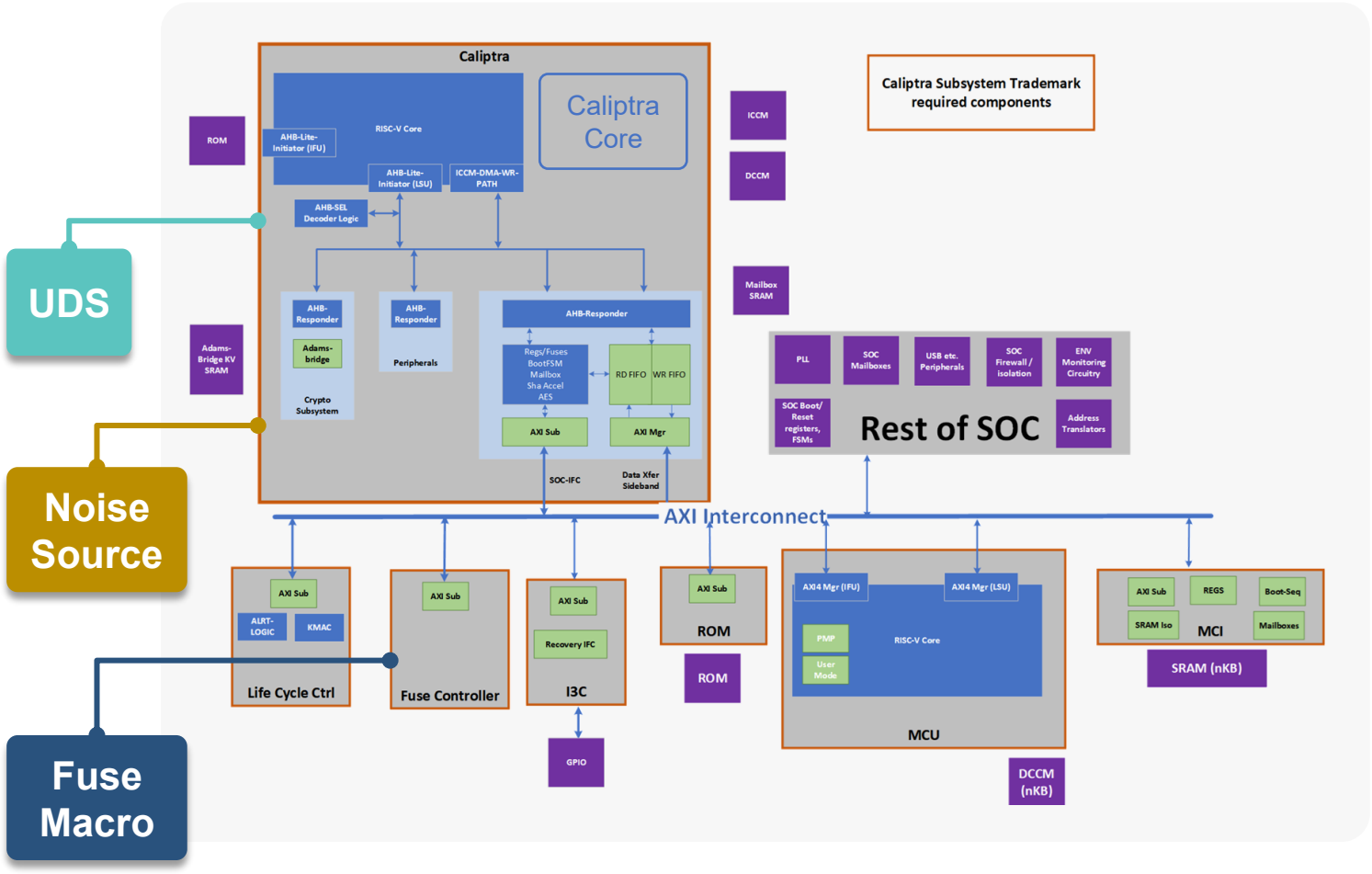
- Device secret
- Provisioned/generated
- Source of obf. Key

Fuse Macro

- Security Storage
- Interface with Fuse CTRL
- Yield & Reliability

Noise Source

- For Caliptra iTRNG
- For UDS generation
- FIPS certification (ESV)



Fuse Map and Provisioning Requirements

OTP/Fuse Density

- Caliptra fuses
- IDevID-cert signatures
- MCU-ROM fuses
- SoC fuses

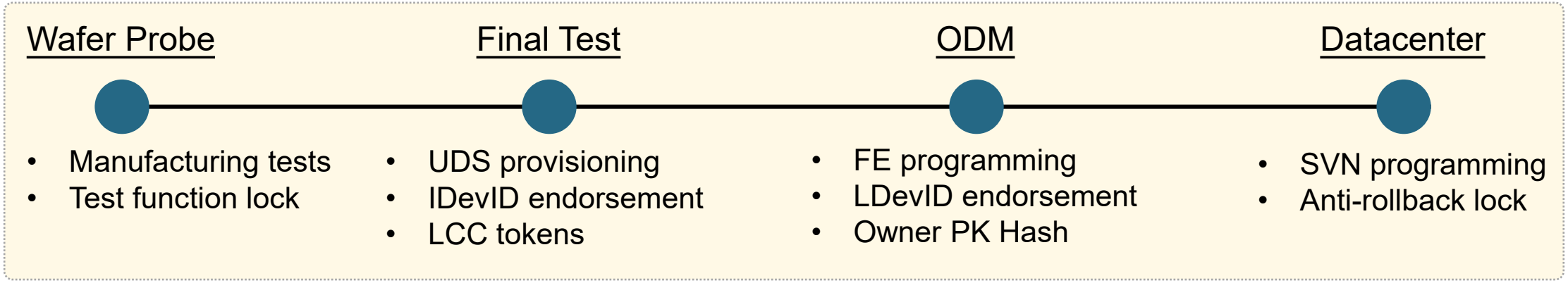
Yield & Reliability

- Yield in adv. nodes
- Retention & endurance
- Self-repair & ECC
- Built-in testing

Field Programmability

- Lifecycle transition
- Ownership transfer
- Security version number
- Anti-rollback protection

Lifecycle Provisioning Flow



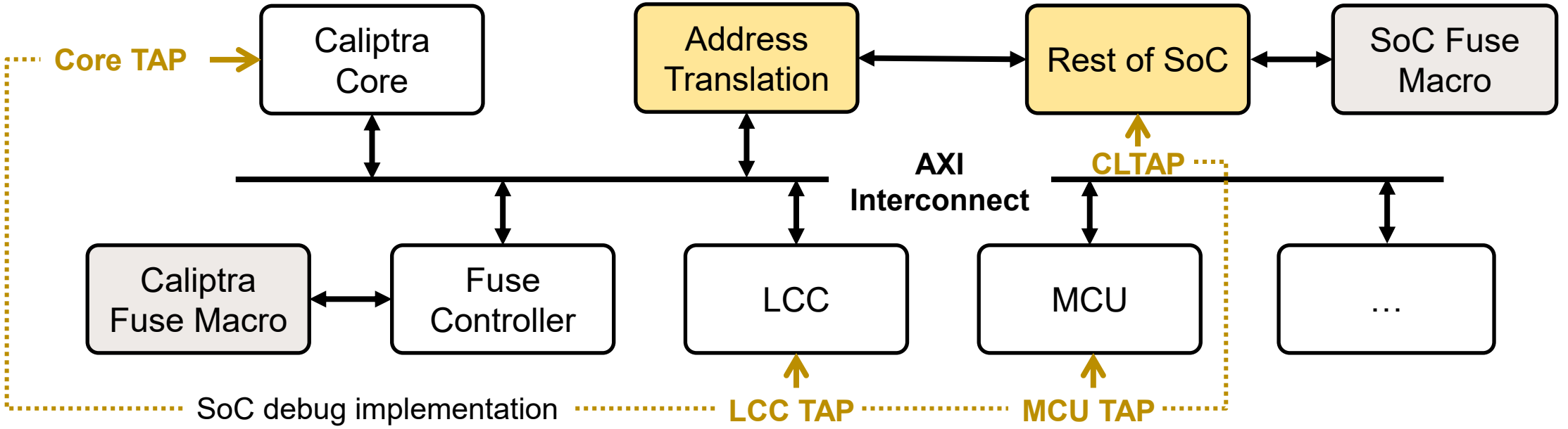
SoC-Level **Integration** & **Debug** Considerations

SoC Integration Considerations

- AXI fabric integration & verification
- Memory macro instances
- Compatibility with legacy design

Caliptra-compliant Debug Design

- JTAG TAP multiplexing
- Chip-level TAP integration
- Debug strap signal implementation



Agenda

01 Caliptra Implementation: Lessons Learned

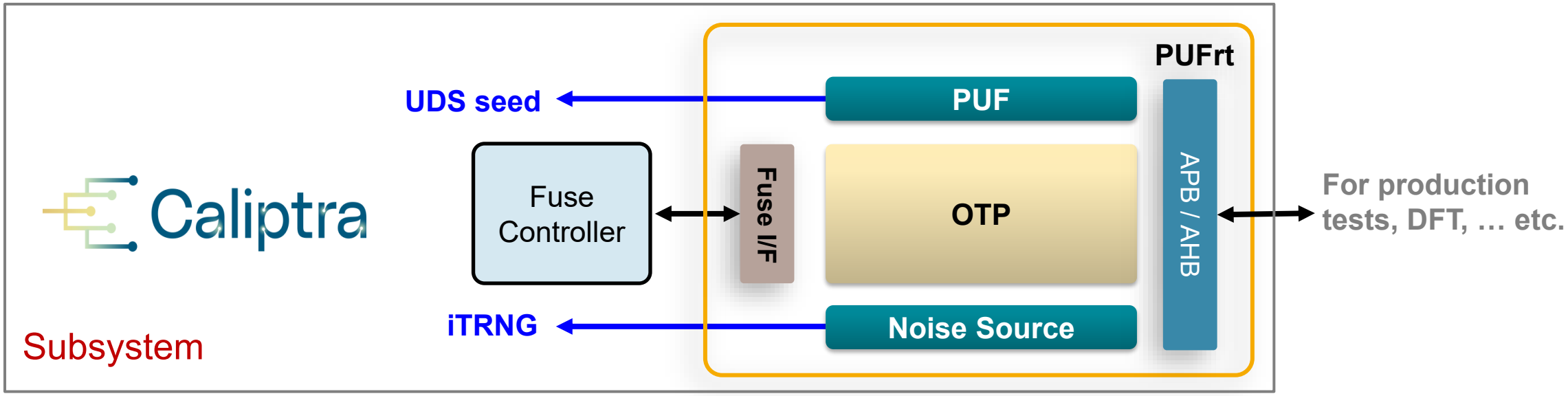
02 What we offer: IP and Turnkey

03 Benefits: Compliance and Time-to-Market

04 Industry Adoptions and Takeaways

PUFrt: **All-in-one Physical Anchor** for Caliptra

- **Pre-hardened PUF** →assured randomness for UDS seed
- **Qualified anti-fuse OTP** in most advanced processes → reliable storage for key and LCS
- **ESV certified Noise source** → met SP800-90B requirement for Caliptra iTRNG
- **Integrated design** with access control & anti-tampering → physical security
- **Advanced Technology** support down to 3nm silicon (2nm and below are coming)



Turnkey Services for Caliptra

Pre-integrated IP Package

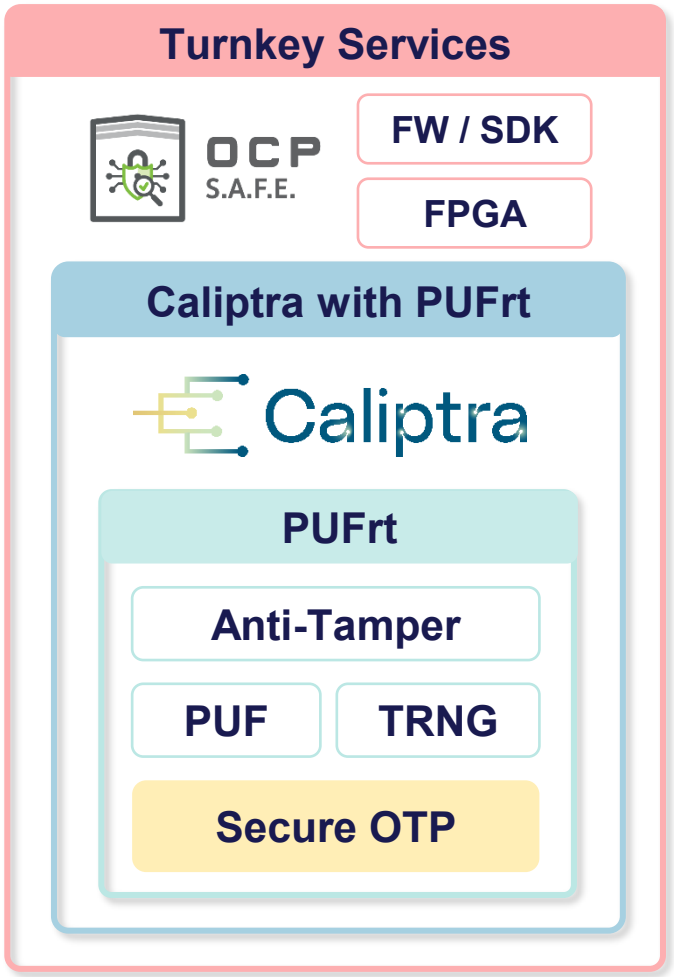
- Design Reference Kit (Physical & Digital IPs)
- Integration and Testing Guide
- MCU SDK with PUFrt drivers

Pre-Verified/Certified Design

- FPGA Verification Platform
- OCP S.A.F.E. Evaluated IP
- NIST ESV Certified Entropy Source

Technical Support Services

- Pre- and Post-Silicon Support
- Certification support (e.g., OCP S.A.F.E.)



Agenda

01 Caliptra Implementation: Lessons Learned

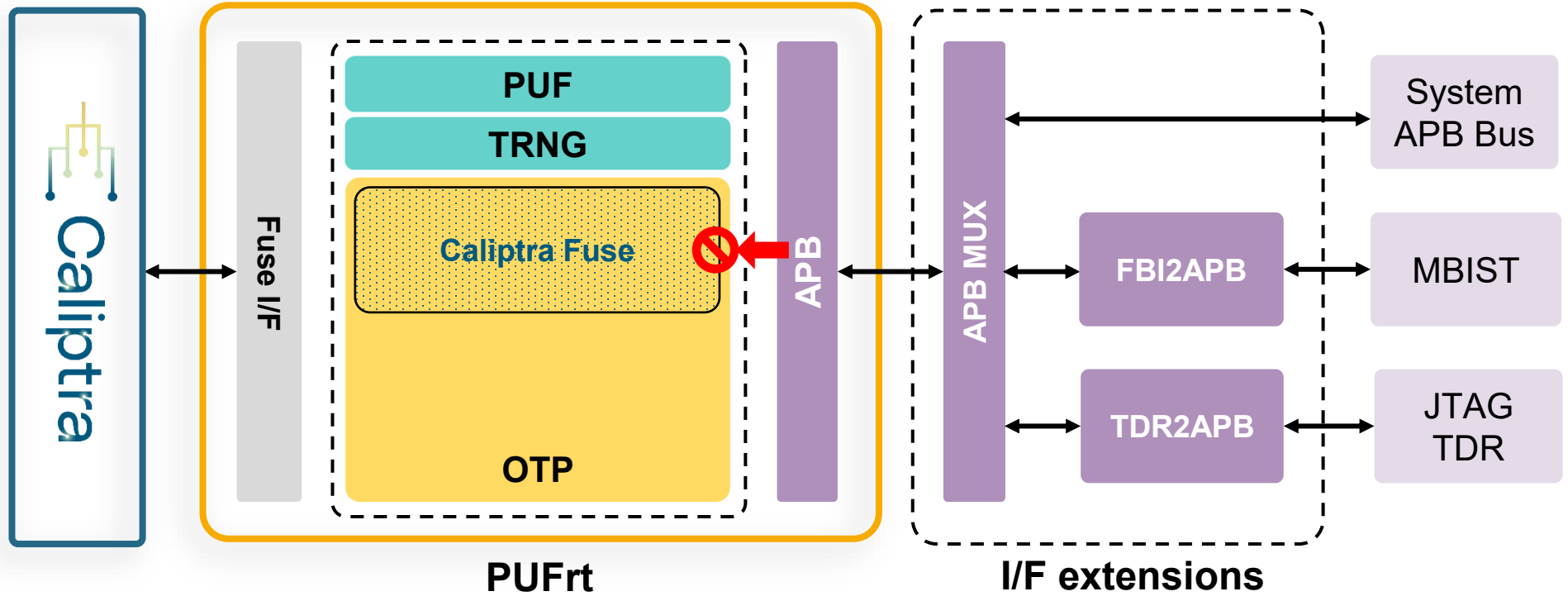
02 What we offer: IP and Turnkey

03 Benefits: Compliance and Time-to-Market

04 Industry Adoptions and Takeaways

Dual Interface Design for Caliptra & SoC

- Caliptra and SoC can share the same OTP macro to save area – e.g., boot code or trimming bits
- OTP partition for Caliptra is hardware-isolated from SoC access, preventing illegal access
- DFT functions are enabled with specific APB bridges, e.g., MBIST and JTAG
- Debug/DFT access to SoC OTP is also controlled by Caliptra's lifecycle state



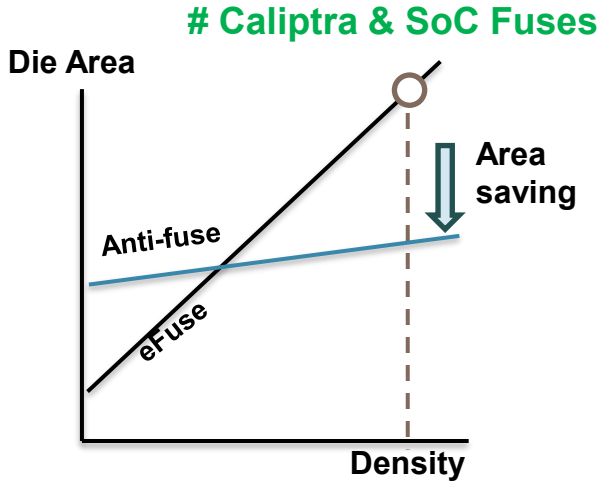
OTP is Foundational for Caliptra and SoC

Anti-fuse OTP is a better choice for Caliptra

- Better area efficiency in advanced node & large density
- More reliable especially supporting many read cycles
- Anti-fuse is field programmable

Emerging use cases for the entire SoC

- Support boot code patch (e.g., MCU-ROM patch)
- Support DFT and SRAM repair
- Trimming for PVT sensors



Emerging Use Cases

Boot Code
Enhanced flexibility for developers

DFT / SRAM Repair
More SRAM used in HPC / adv. node

Trimming for PVT sensors
Distributed sensors for performance tracking

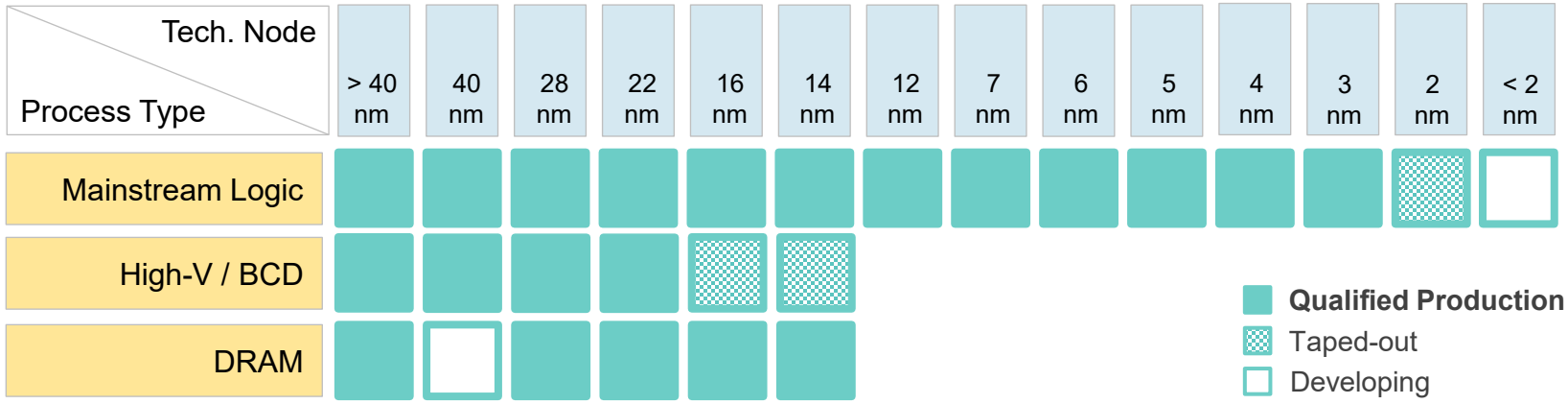
Highly Robust OTP with **Silicon-Proven** Record

Silicon proven yield and reliability

- Built-in OTP self-repair, ECC and voting scheme for yield and reliability
- Qualified retention and endurance with HTOL and HTST

Widely available across foundry, process type, and technology nodes

- Mainstream logic process for HPC, Networking, SSD controller, BMC, …
- HV/BCD process for PMU & DRAM process for LPDDR, HBM, …



tsmc intel foundry services SAMSUNG GlobalFoundries UMC SK hynix more…

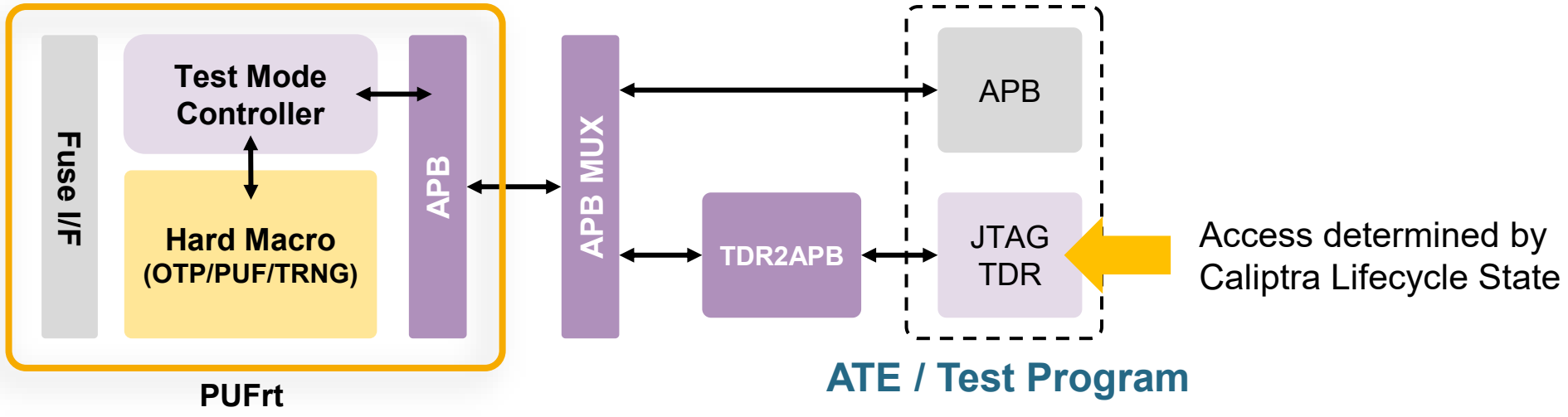
Simple and Secure Manufacturing **Test Flow**

Built-in test routine for ATE without additional testing FW

- One-touch PUF enrollment – enables UDS without provisioning
- Auto-repair and margin-read function – ensures production yield and reliability

Caliptra-compliant testing design with zero-backdoor

- No additional analog pad – eliminates the risk of backdoor access
- Test function locking mechanism – adds extra protection for test access control



OCP S.A.F.E. and Caliptra Trademark

- PUFrt has gone through OCP S.A.F.E. audit done by Tetrel (Scope3)
  - Repo: github.com/opencomputeproject/OCP-Security-SAFE/tree/main/Reports/PUFsecurity

- Caliptra Trademark assessment is also conducted, covering:
  - UDS Seed and Field Entropy
  - Obfuscation Key
  - Fuses Management
  - Random Number Generator (RNG) Implementation



Pre-Integrated Caliptra-SS for **MCU Bring Up**

Customer's Design

Develop your own MCU software on FPGA platform.

Code Caliptra PUFrt

Pre-integrated and fully validated Caliptra-SS + PUFrt for immediate development and seamless migration.

Versal™ AI Core Series **VCK190** Ev-Kit AMD



Use the official Caliptra reference FPGA platform —**no FPGA or platform migration required.**

Pre-Integrated Caliptra-SS for **MCU Bring Up**

Custome your own MCU software on FPGA platform.

grated and fully validated Caliptra-SS + PUFrt
diate development and seamless migration.

Versal™ AI Core
Series **VCK190** Ev-Kit
AMD

official Caliptra reference FPGA platform
—**no FPGA or platform migration required.**

Available on GitHub!

Bring Up Your Own Caliptra

Agenda

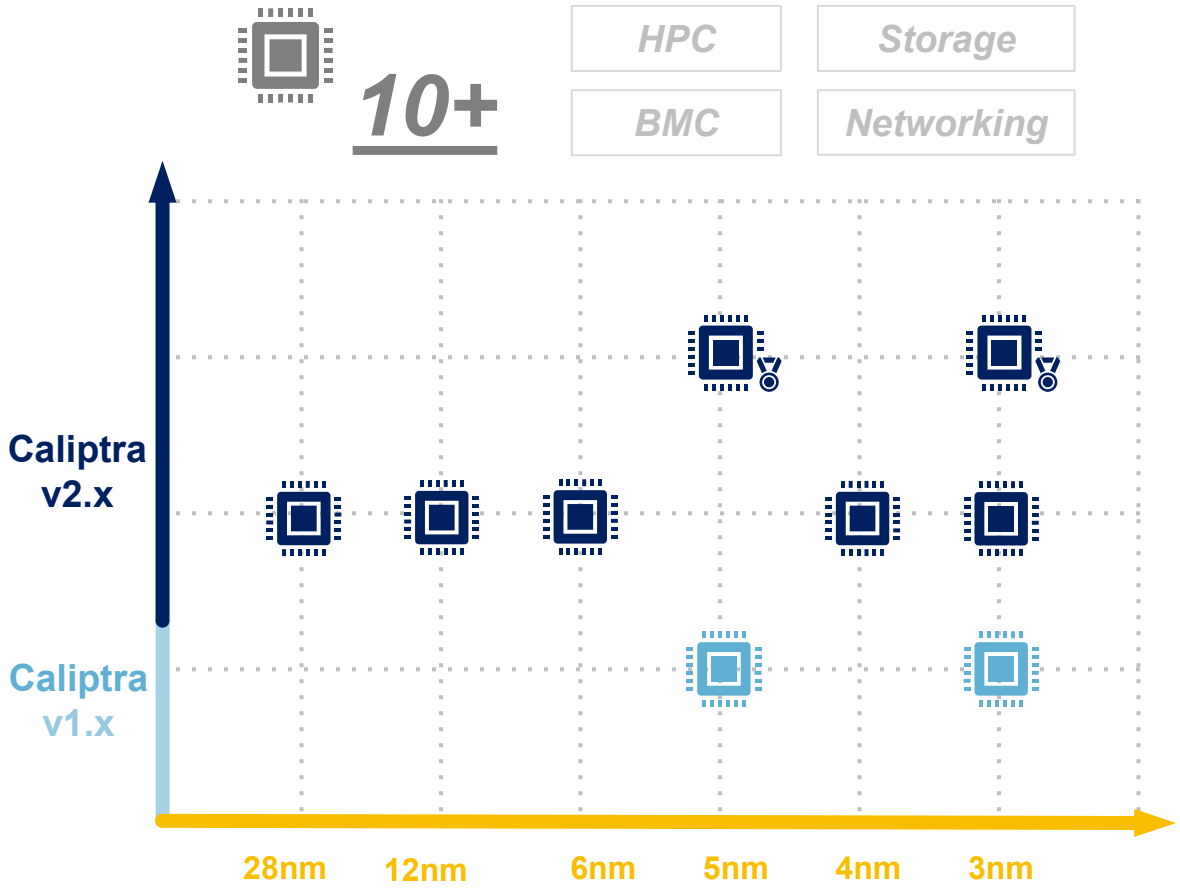
01 Caliptra Implementation: Lessons Learned

02 What we offer: IP and Turnkey

03 Benefits: Compliance and Time-to-Market

04 Industry Adoptions and Takeaways

Successful Partner Adoptions



Turnkey Services
- Two pioneer partners adopted our solution
- Pre-integrated FPGA platform publicly available

PUFrt Physical Anchor
- Full-compliance with Caliptra integration guidelines
- First Caliptra v2.1 in silicon adopts PUFrt

Technology Foundation
- OTP and PUF widely available across technology
- Silicon-proven yield, reliability, and testability

Takeaways

From open spec to real silicon: the SoC integration gaps

Bringing Caliptra into production requires solving critical challenges: physical primitives, legacy compatibility, debug isolation, and manufacturing test.

PUFrt physical anchor: dual interface & zero-backdoor OTP

An all-in-one physical anchor that harmonizes Caliptra with host SoC, delivering high-yield anti-fuse without physical test backdoors.

Turnkey services: accelerating ecosystem adoption:

Pre-integrated IP and end-to-end engineering support—empowering partners to de-risk adoption and achieve first-pass silicon success.

Caliptra PUFsecurity

Caliptra

Thank You .



PUFsecurity

OPEN Compute Project®

Caliptra



SECURITY

Caliptra RFC Process

Bharat Pillilli, Microsoft
Zach Halvorsen, Software Engineer, Google

Caliptra



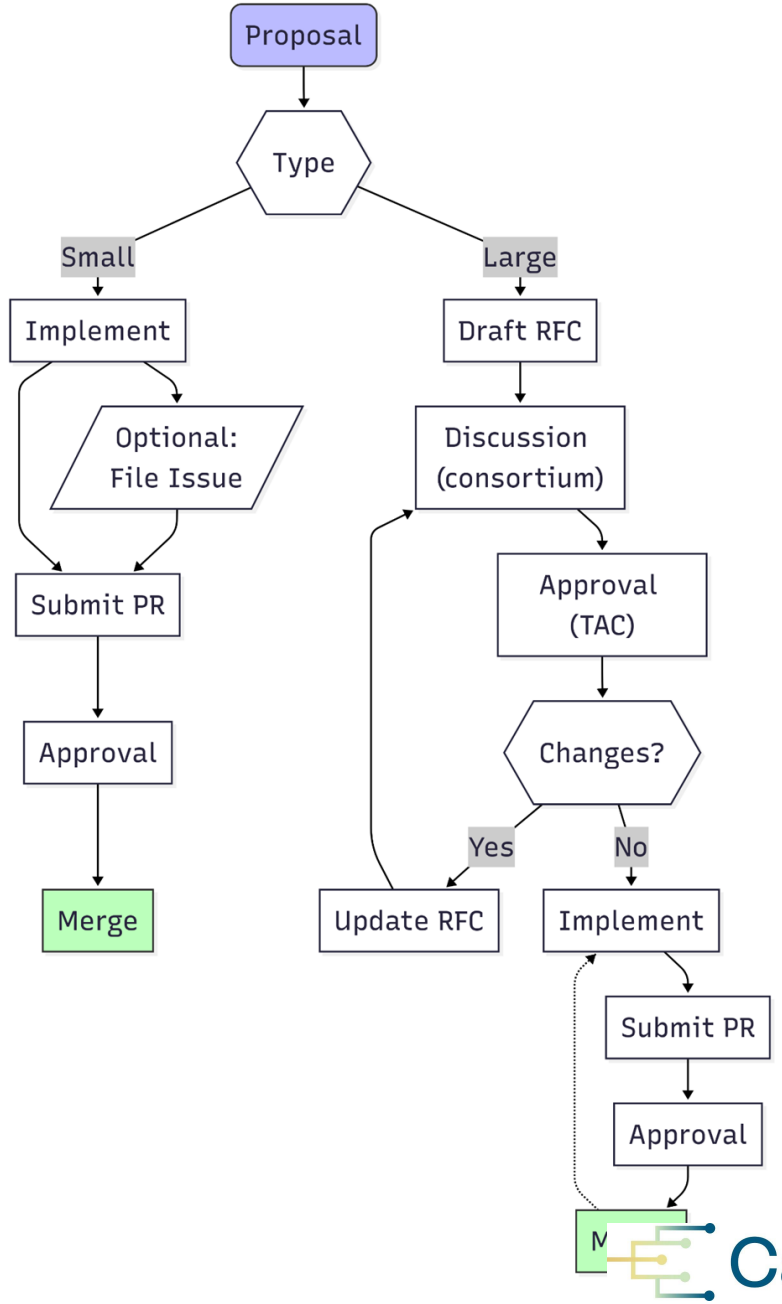
Overview

Caliptra

Process

Two type of changes, both via GitHub:

1. **Small**
   a. [Optional]: File an issue track
   b. implement the change (via a PR)
2. **Large**
   a. draft an RFC as an issue
   b. seek discussion from consortium
   c. seek approval from TAC
   d. apply any changes requested
   e. implement the change (via several PRs)



What is a “Large” change?

- Architecture
  - New HW component
  - Large changes e.g. upgrade Veer from 32-bit to 64-bit
- Boot process
- API
  - Command set
  - Register layout
  - Input/Output
- Security posture
- Maintenance
  - Tools
  - Tool Versions

RFC GitHub Issue Template

HEADER & ABSTRACT

- Title format: [RFC] {RFC title here}
- Brief abstract summarizing proposed changes

SCOPE & IMPACTED REPOS

- Affected repos: project, hardware, software
- Impacts: Specs/docs, FIPS 140-3, APIs, resources, perf

RATIONALE

- Motivation and justification for proposed change
- Target Caliptra release version justification

IMPLEMENTATION TRADEOFFS

- Design alternatives considered & why internal to Caliptra
- [Optional] Links to dev work & test results

TIMELINE & TEST PLAN

- Target release & milestone schedule
- Test plan & suite: env, TB/FPGA, randoms, pass criteria

MAINTENANCE

- Responsible point(s) of contact
- Ongoing maintenance and support plan

RFC Approval

- Caliptra TAC designates delegates
- Delegates review proposals weekly
- Implementation permitted to begin (e.g. as RFC collateral)
    - Contributions not accepted or merged until RFC approval
- Delegates will correspond with proposer and provide response on the RFC

OPEN Compute Project®

A few changes since last presentation

RFC BOARD & STATUS

Public board of RFCs and their status.

https://github.com/orgs/chipsalliance/projects/11/views/4

RFC SUBMISSION REPOSITORY

All RFCs should be submitted to the Caliptra top-level repository

https://github.com/chipsalliance/Caliptra

Caliptra

OPEN Compute Project®

HW RFC Timeline

RFC Submission Details

Caliptra

Hardware RFC Submission Timeline

Standard process and milestone cadence for annual hardware feature releases

1. Open Submissions

4 Months Prior

RFC window opens. Community submits new hardware feature proposals for review.

2. Review & Selection

1 Month Prior

New submissions close. Selection committee decides final release feature set.

3. Announcement

OCP Global (Oct)

Official public announcement of next year's planned hardware features.

4. Implementation

1-Year Duration

Engineering teams implement decided features for full release next year.

Example Execution Timeline (Target Release: October 2028)

June – August 2027

Accept RFC Submissions

3-month open call for proposal submissions.

September 2027

Make Decisions

Submissions close; finalize selected 2028 features.

October 2027

Announce Features

Announce 2028 feature set at OCP Global; begin 1-yr build.

OPEN Compute Project®

2026-2027 Hardware RFC Execution Timeline

Accelerated cycle for current year feature releases

1. Open Submissions

Now Active

RFC window is currently open. Accepting hardware feature proposals for the upcoming cycle.

2. Submissions Close

End of November

Call for proposals officially closes at the end of November.

3. Review & Selection

December

Selection committee reviews proposals and decides final feature set.

4. Announcement

January

Official public announcement of selected features and rollout roadmap.

Current Execution Schedule

Now – November 2026

Accepting RFCs

Open call for proposal submissions currently underway.

December 2026

Make Selections

Submissions closed; committee evaluates and finalizes feature choices.

January 2027

Announce Features

Announce final hardware feature set; initiate implementation phase.

Caliptra

RFC Progress & Status

CHIPS Alliance Technical Steering Committee Board Overview

59

Total RFC Submissions

Active proposals tracked across the CHIPS Alliance organization workspace.

17

Finished / Merged

Proposals fully reviewed, approved, and integrated into official specifications or repository standards.

9

Under Review

Active community discussion and Working Group evaluation currently underway.

3

In Implementation

Approved RFCs actively being implemented into target codebases and hardware designs.

A Word on Trademark

- Engage with SRP early, >6 mo from tapeout
- Get PSIRT access, confirm checklist version for product with WG
- Ask WG clarifying questions early on. May inform architecture, e.g.
  - PUF vs Fuse
  - RTL Keys vs ECO
  - Scan chain design

Trademark versioning is coming
(A checklist specific to a RTL/FW release)

Industry Contributions

- Marvell & MCU ROM patching
- Seimens & USB VIP
- NXP & USB2
  - We will deep dive on this ‘large’ RFC example



Thank you

Caliptra

HARDWARE ROOT OF TRUST · CALIPTRA

Integrating a PUF into Caliptra: Hardening the Root of Trust



What we did, why we did it, and what we learned

Craig Barner

Senior Distinguished Engineer, Custom Silicon Engineering, Marvell

A practical, experience-based look at adding a Physically Unclonable Function to a Caliptra-based root of trust

Caliptra

AGENDA

Five stops on this story



1 **The problem**

Why a fused secret alone is a liability



2 **The idea**

Strengthening the Unique Device Secret with a PUF



3 **The safety net**

A disable path that never becomes a backdoor



4 **The tension**

Certifying entropy quality without exposing the secret



5 **The lesson**

Locking debug access to the right lifecycle state

Caliptra

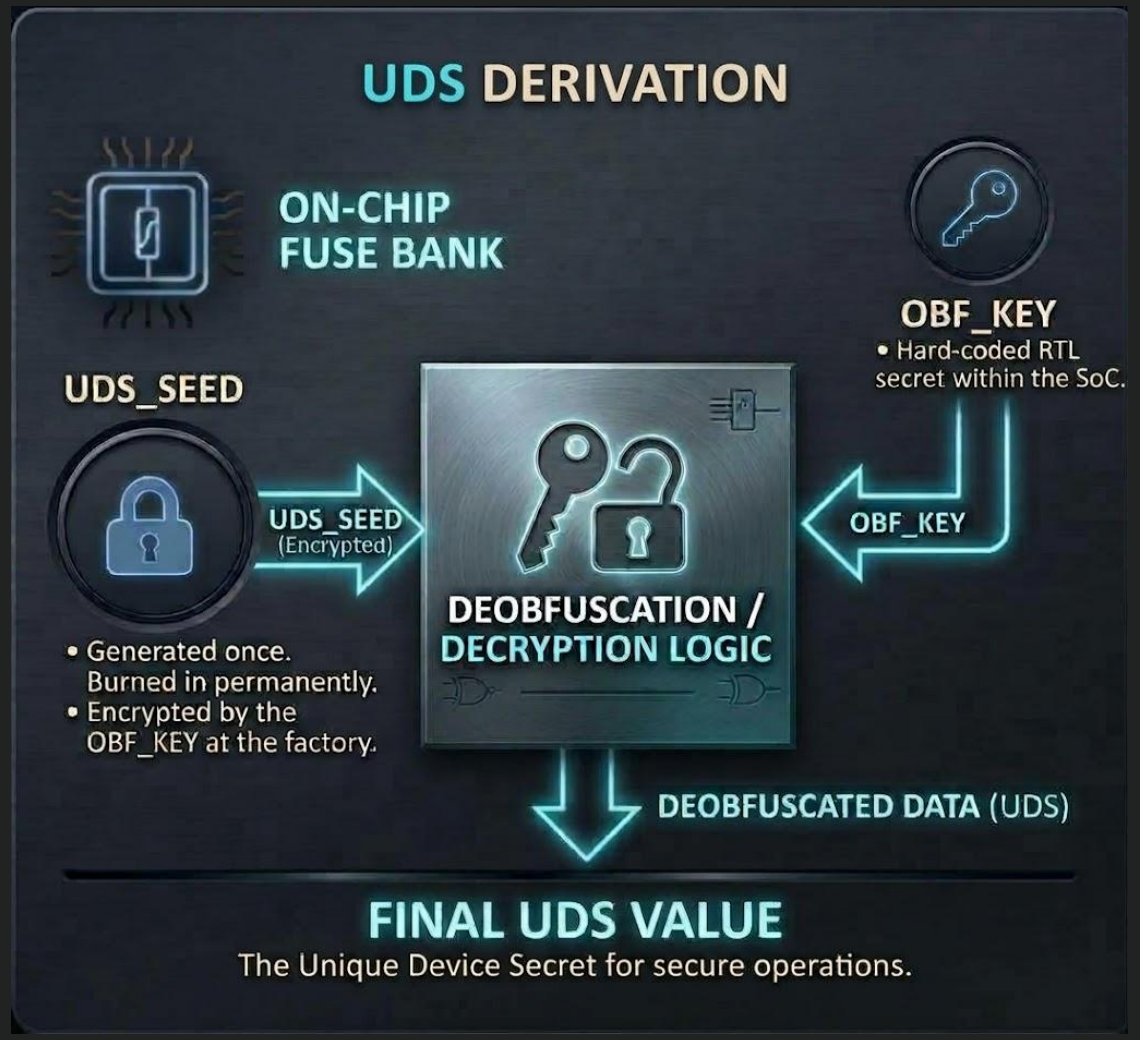
THE PROBLEM

Root Value Is a Static and Fixed Target

- Caliptra's identity chain starts with an Unique Die Seed, which anchors every key derived afterward
- Traditionally, generated at random once, and permanently burned into on-chip fuses (deobfuscated at boot)
- A fused seed is powerful, but it is also static: once a value exists anywhere, it is a fixed target for the lifetime of that value

The question we asked:

Could the root secret be derived from something that is never stored, never shared, and unique to every chip?



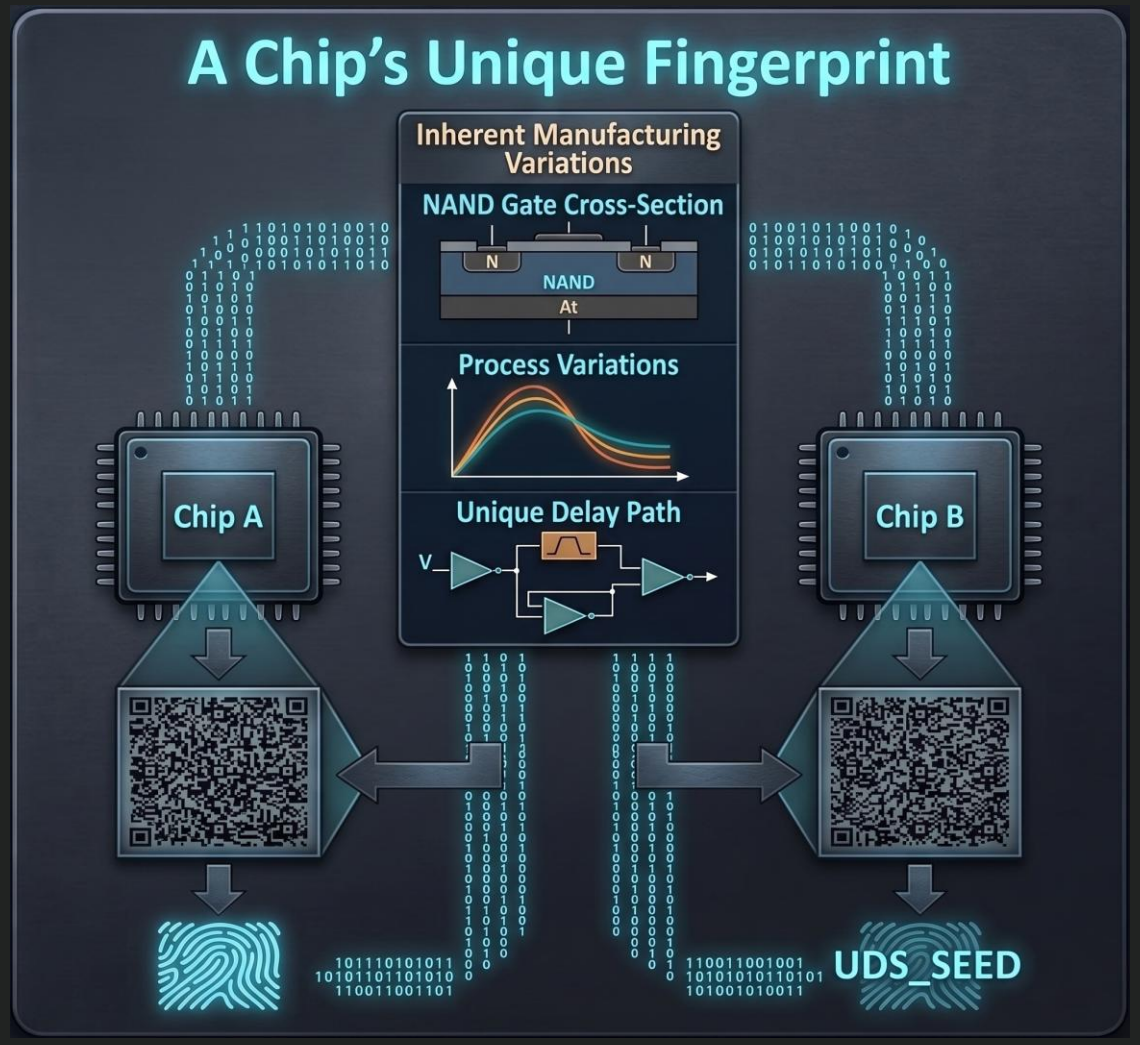
Caliptra

THE IDEA

A Physically Unclonable Function

- What it is: A hardware IP block that leverages random, intrinsic manufacturing variations to generate a reproducible unique physical fingerprint for a chip
- What it does: Produces a device-unique secret key for security uses like authentication and encryption, without ever storing that key in memory
- How it works: It regenerates the same key each time from its physical entropy source, verifying it against OTP stored (non-secret) helper data instead of storing the key itself

A PUF turns "a secret we must protect forever" into "a secret that only exists for a moment, on that chip, when it's needed."



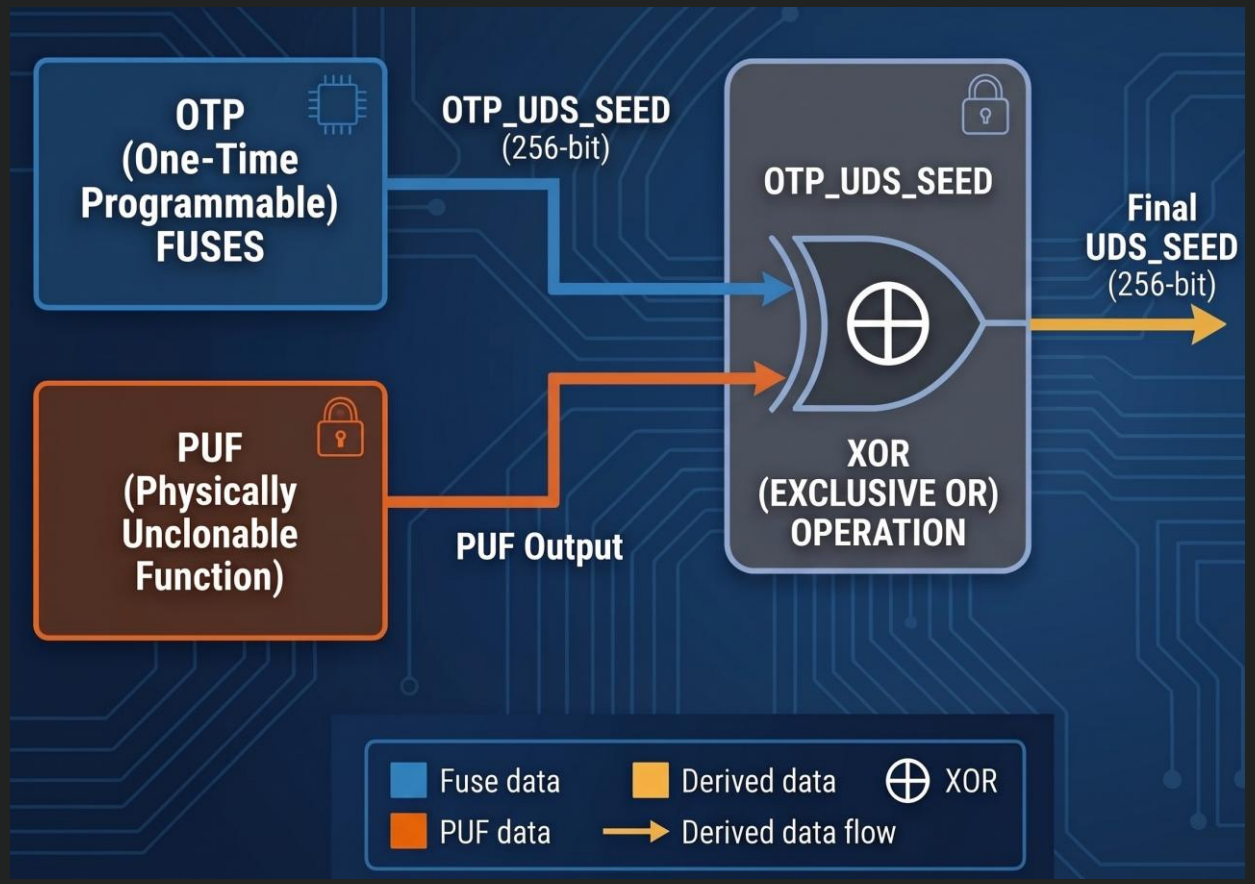
Caliptra

THE DESIGN

Defense in depth, not replacement

- Rather than replacing the fuse-based secret, we combine the PUF's output with it using a simple XOR operation
- The combined result depends on two independent things: a value stored in silicon fuses, and a value that only exists physically in the PUF hardware.
- This is additive by design, it strengthens the existing secret rather than creating a new single point of failure.

An attacker now has to compromise both the fuse contents and the unclonable PUF response — not just one or the other.



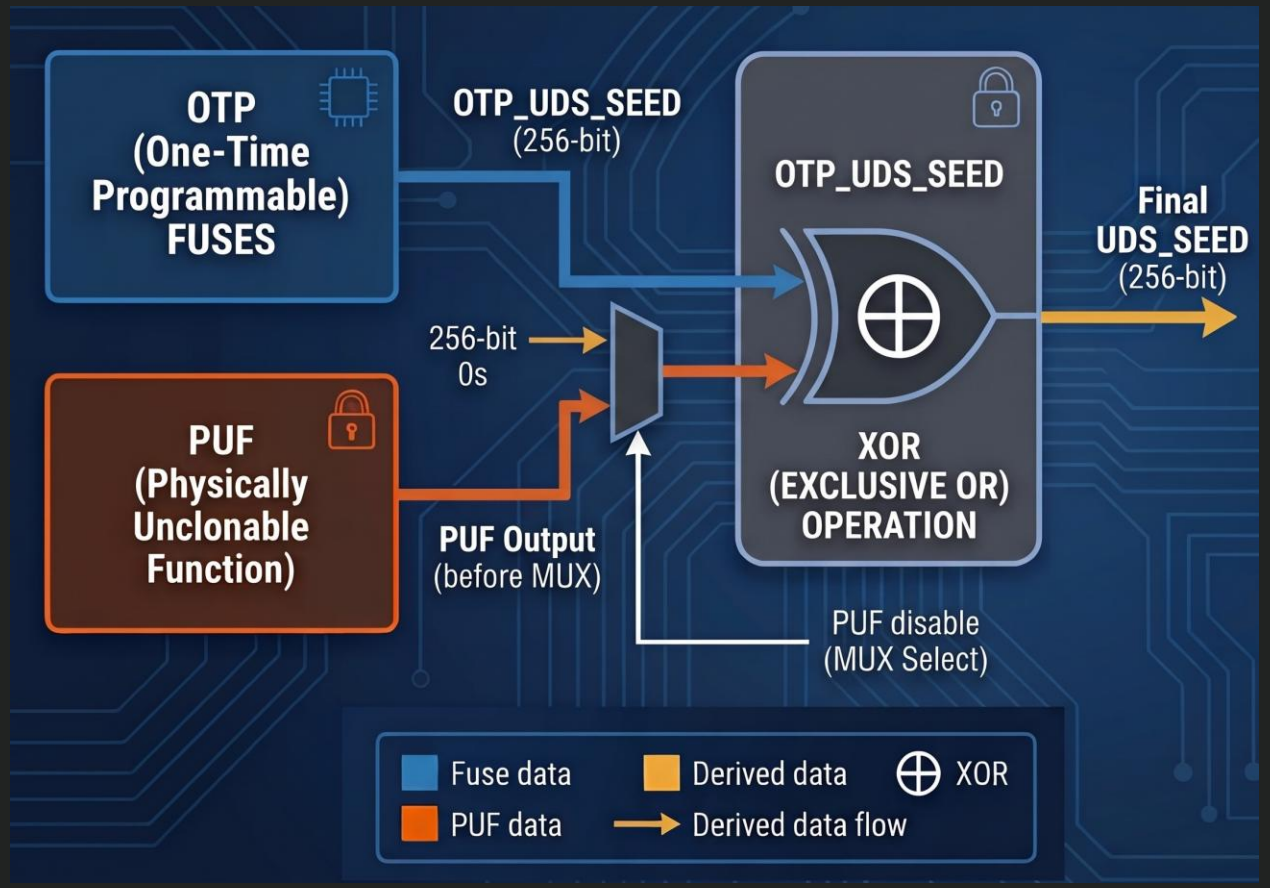
Caliptra

THE SAFETY NET

A disable path that degrades, not breaks

- We built in a seamless fallback path that allows the PUF contribution to be disabled.
- When disabled, the resulting secret is mathematically indistinguishable from the original fuse-only implementation.
- This safety net gave engineering the confidence to deploy the PUF, knowing a validated fallback existed if issues ever emerged at production scale.

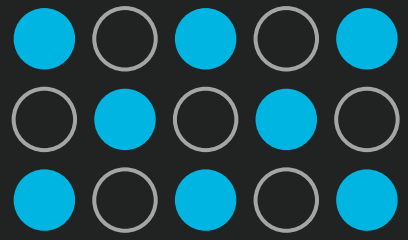
Proven in verification: the disabled path is bit-for-bit identical to the original fuse-only value



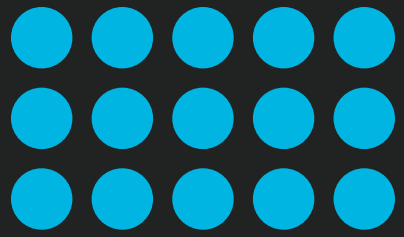
Caliptra

THE DISABLE SWITCH

A fleet-wide failsafe

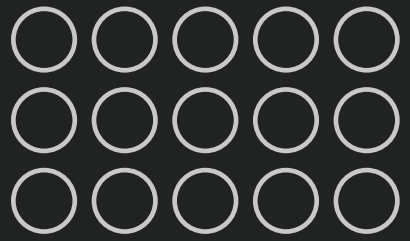


PUF SPLIT FLEET



PUF ACTIVE FLEET

Fleet-Wide Policy Decision



PUF DISABLED FLEET

NEVER SPLIT THE FLEET

Any disable decision is deliberate and fleet-wide; never a partial split between PUF-active and PUF-disabled devices.

AUTOMATIC RECOVERY

MCU ROM detects an aging-check failure and falls back to the fuse-only secret automatically; never a bricked part.

NOT A REPAIR TOOL

Not used to rescue individual parts during manufacturing test for yield; this is a fleet-safety mechanism only.

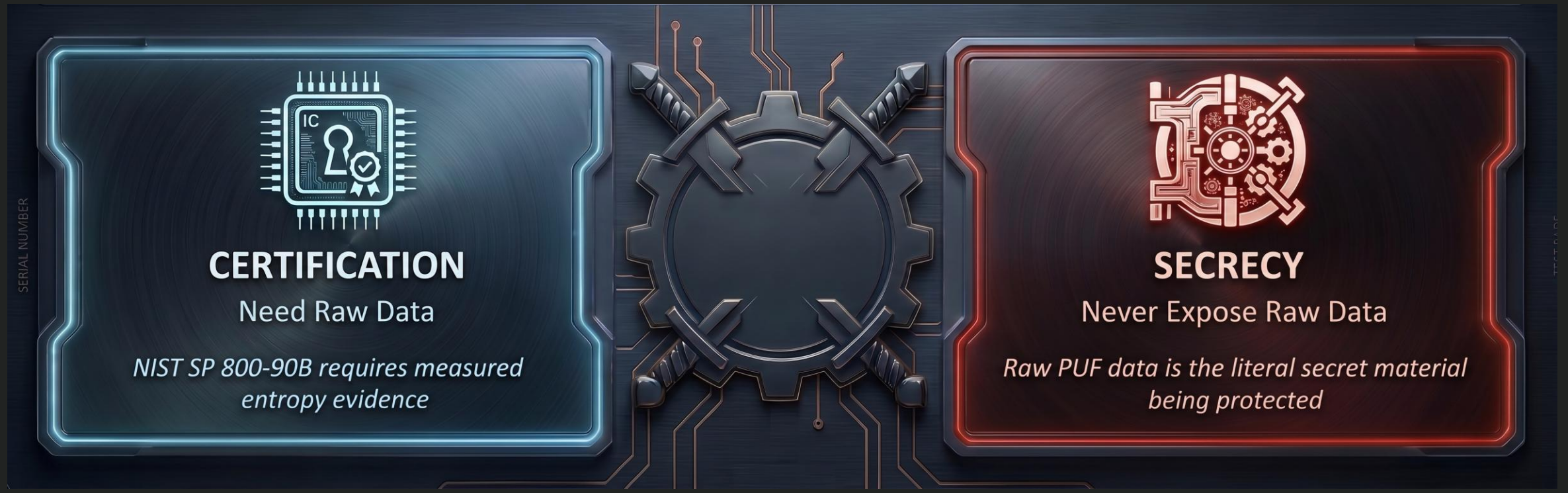
Governance matters as much as cryptography.

Caliptra

THE TENSION

Proving entropy quality without exposing the secret

- How do you prove entropy quality without exposing the secret?



Every debug or certification hook becomes part of the security boundary.

Caliptra

THE RESOLUTION

Hardware Protects Root entropy : readable only in RMA, and never after activation

1 INSTANTIATE

Mailbox cmd: PUF_INSTANTIATE_CMD

- Health test
- Generates the stable-bits (entropy)
- Programs helper data into PUF OTP (one time)
- **No key generated**

2 ACTIVATE

Mailbox cmd: PUF_ACTIVATION_CMD

- Generates key
- Burns Aging Reagent + REAGENT_DONE flag
- Disables stable-bit read permanently
- **One-time, irreversible**

3 EVERY BOOT AFTER

Automatic on each boot

- PUF regenerates key + runs aging test
- Key goes to Caliptra only (XOR with UDS seed)
- **MCU sees only a 64-bit digest**

Who can read the raw PUF entropy?

Before activation — MCU can read it only in RMA

🔒 After activation — Hardware rejects every MCU command to the PUF, in every lifecycle state

ROM blocks Manufacturing/Production until activation is complete (or PUF is disabled).

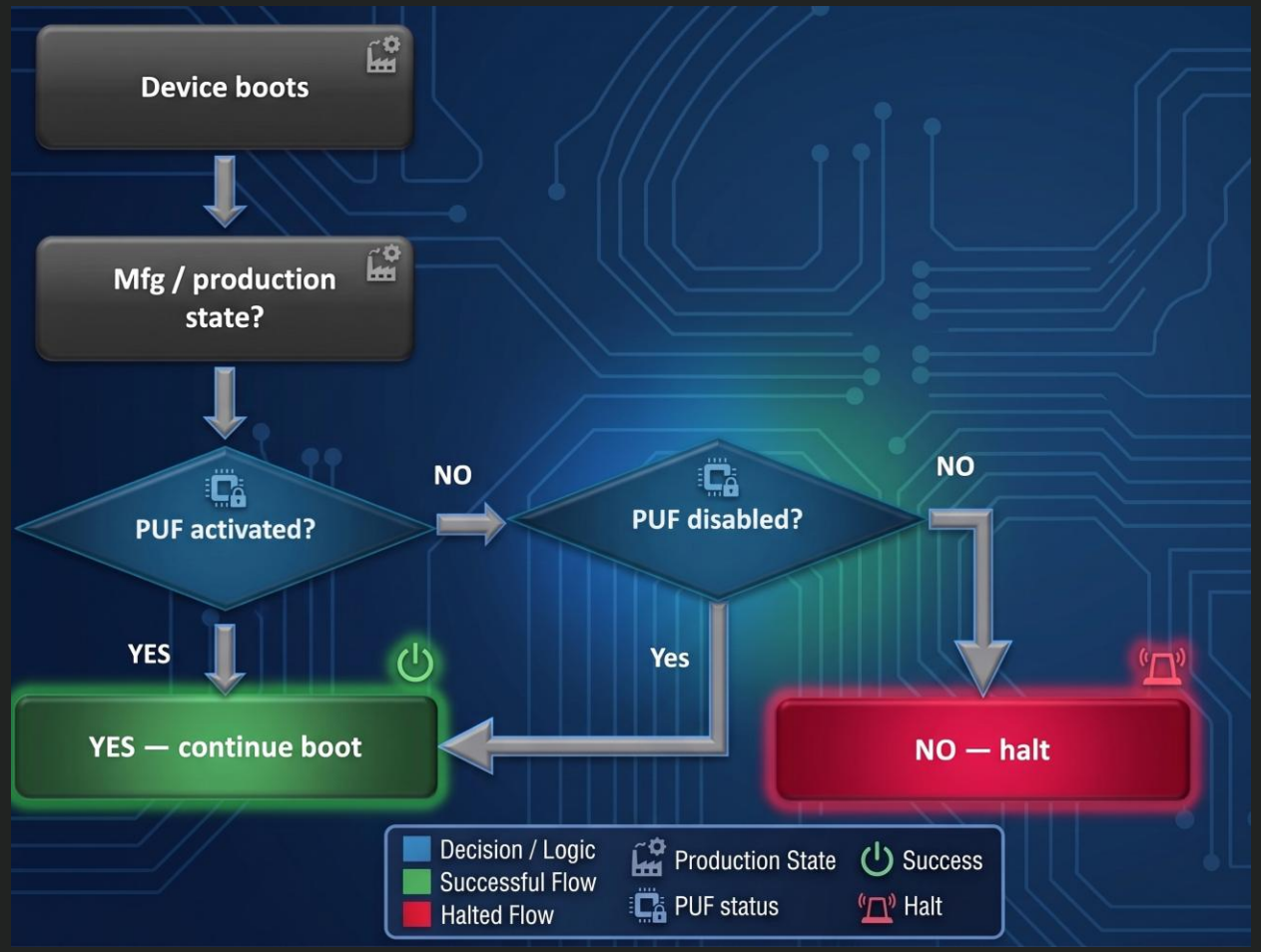
Caliptra

CLOSING THE GAP BEFORE PRODUCTION

Lifecycle enforcement

- A device must never reach Manufacturing or Production with a path to PUF key extraction still available.
- MCU ROM enforces this policy by validating that debug access has been irreversibly disabled and stopping boot if the check fails.
- Before exiting test and validation, the device must either activate the PUF and lock debug permanently or disable PUF participation altogether.

Only designated sampling parts retain PUF-read capability; all others permanently disable PUF output before production.



Caliptra

TAKEAWAYS

What we'd tell another team doing this



Start with the threat model, not the feature

Only add complexity where it closes a real, articulable risk



Build safety valves with governance attached

Know in advance who can flip the switch, why, and scope it fleet-wide



Treat every debug and certification hook as a security control

Scope it to the narrowest lifecycle window possible



Verify the “off” state as rigorously as the “on” state

Graceful degradation matters as much as the new feature working



Look for existing behavior that already gives you the safety property

Before adding new logic, check what the system already enforces

Caliptra



Closing Thoughts

A PUF let us turn a permanently stored secret into one that exists only for a moment, on one physical chip. Every added capability — combination, disable, debug, entropy measurement — was scoped by a matching safeguard.

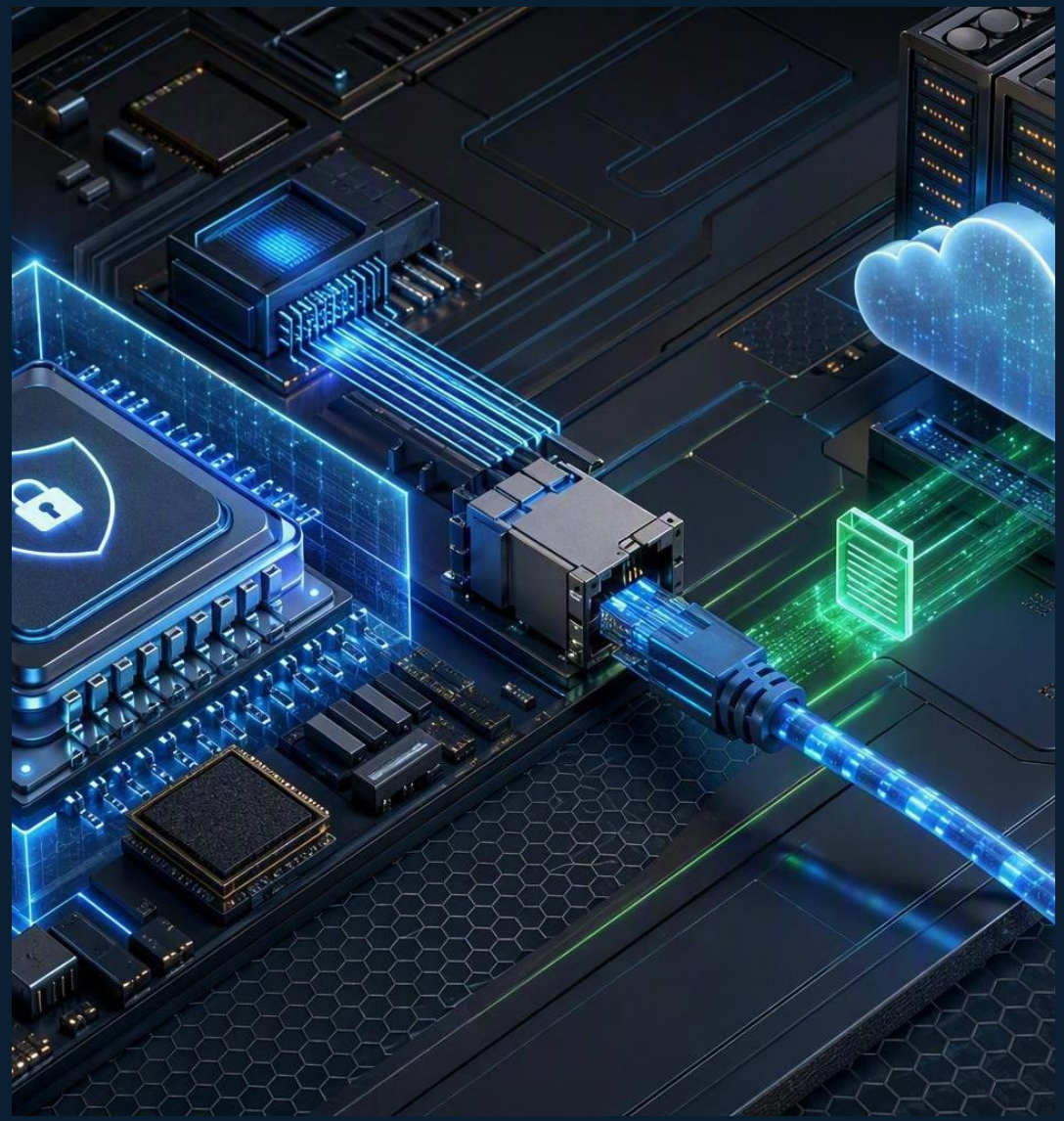
The result: a root of trust that is stronger, still falls back safely, and never leaves a production device with an open door.

Thank you — happy to take questions

Caliptra

NETWORK BOOT & RECOVERY

A Collaboration of ASPEED & Microsoft



Use Case Application:
A System BMC Recovery

Caliptra

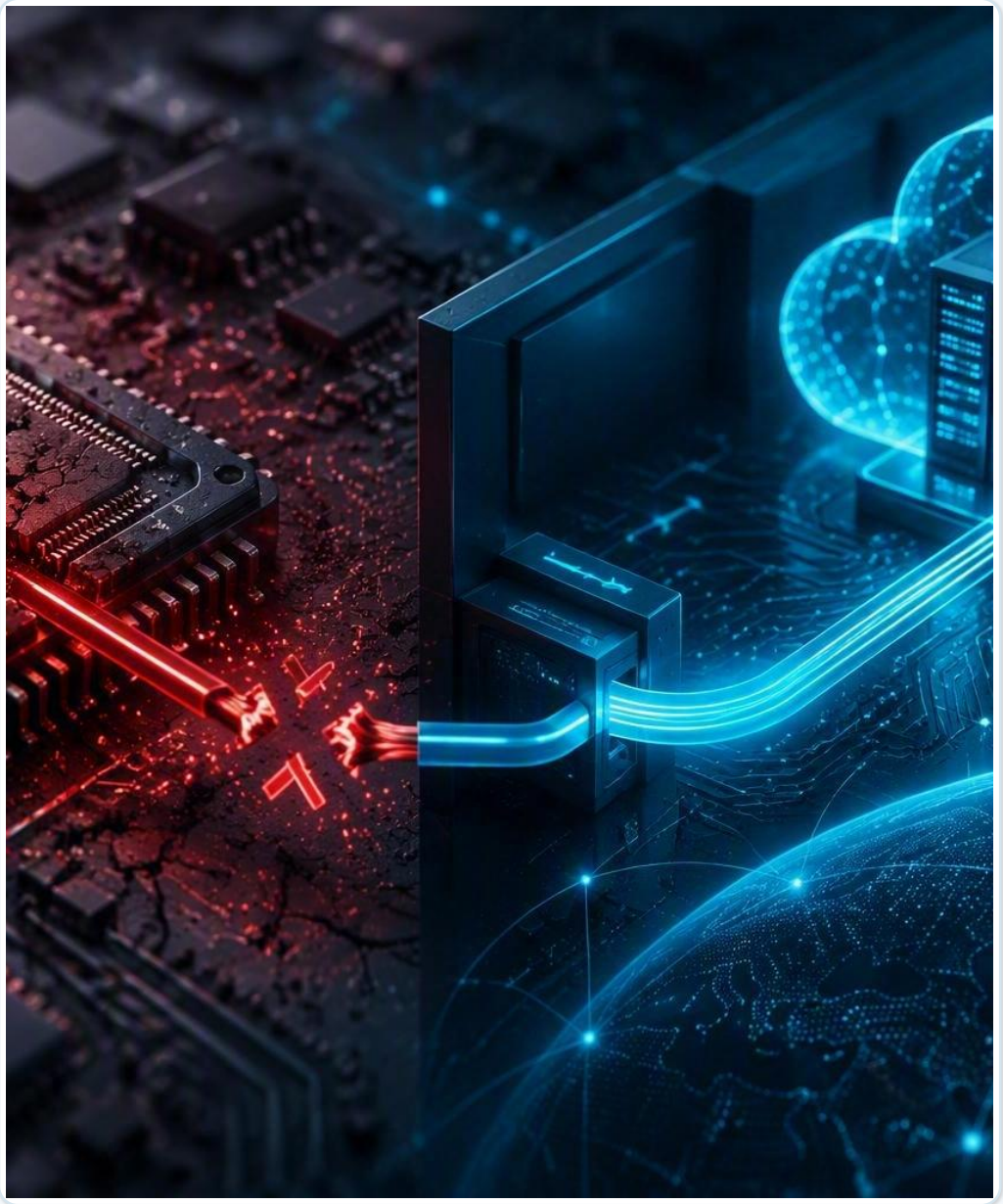
A USE CASE

An Unrecoverable Platform BMC

- Recovery/streaming boot available for devices downstream of BMC

What if BMC cannot boot?

- **RISKS:**
  - Systems are permanently disabled
  - Manual intervention costly in hyperscale environments
- **Constraints**
  - Needs a scalable path at DC level to recover
  - Secure image retrieval
  - Resilient fallback path



THE APPROACH

Add the network as another boot source

A Network Boot Coprocessor (NBP) fetches images; the existing Caliptra boot path verifies them.

Design goals

1. **Minimal MCU ROM footprint**
   Network stack lives in the NBP, not in MCU ROM
2. **Consistent with OCP streaming boot**
   Early firmware uses the same Recovery I/F
3. **Trust anchored in Caliptra**
   Images are verified; the transport is not trusted
4. **Resilient fallback**
   Used when flash boot fails; optional write-back to flash

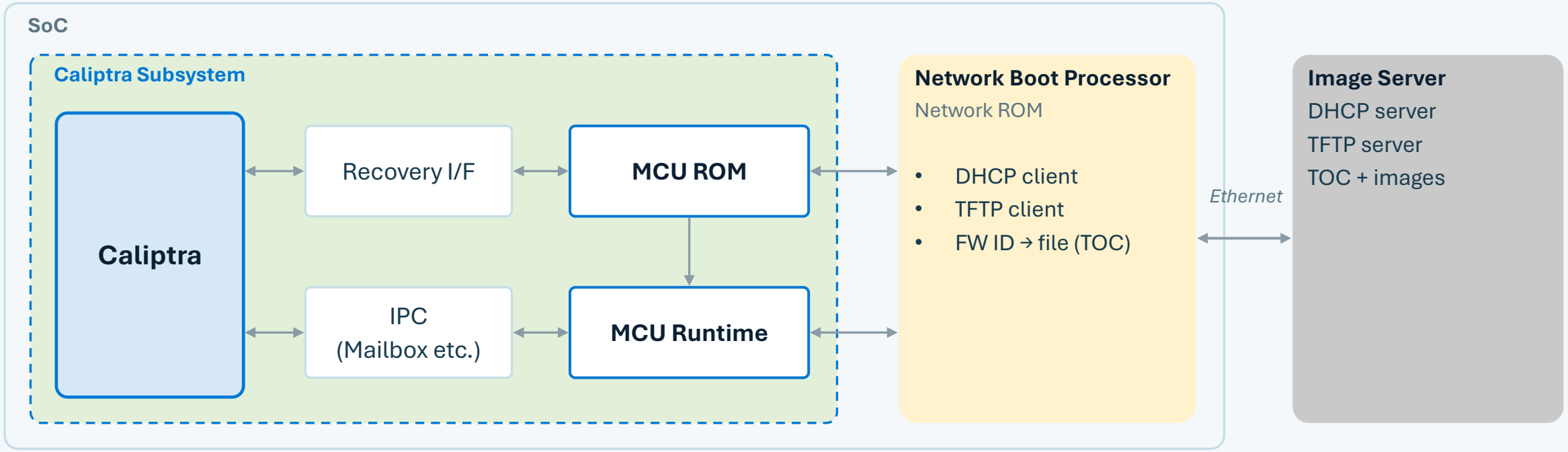
Solution

- Dedicated coprocessor with a ROM network stack (DHCP + TFTP)
- Automatic network setup; IPv6 preferred, IPv4 fallback
- A TOC maps firmware IDs to image files
- Images streamed to the MCU in chunks; the MCU feeds Caliptra
- Same path loads the remaining SoC images at runtime

First practical use case: BMC self-recovery

ARCHITECTURE

Three roles, one trust boundary



MCU orchestrates

Which image next and where it goes; writes the bytes

NBP transports

Network setup, TOC lookup, chunked download; no trust decisions

Caliptra decides

Verifies early firmware; authorizes every SoC image

Caliptra

MCU ⇔ NBP INTERFACE

Boot Source Provider: one interface, any source

MCU firmware requests images by logical firmware ID; the provider hides where the bytes come from.

Operation	Purpose
`initiate_boot`	Network setup, fetch TOC
`get_image_metadata`	Size, checksum, version
`download_image`	Chunked stream, ACK per chunk
`finalize`	Release resources

Network is one provider; the same interface can front other boot sources.

Firmware ID	Image
`0`	Caliptra FMC + RT
`1`	SoC Manifest
`2`	MCU Runtime
`0x1000_0000+`	SoC images

Why DHCP + TFTP

- Fits a ROM environment
- Trust does not rely on the transport

Caliptra

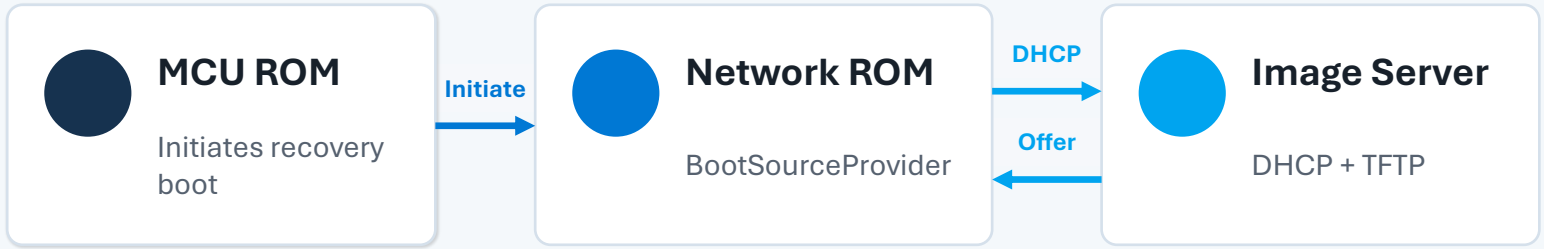
TWO-STAGE BOOT

Two stages, two Caliptra trust gates

The NBP stays available after Stage 1 so MCU Runtime can fetch the remaining SoC images.

	Stage 1 · MCU ROM	Stage 2 · MCU Runtime
Images	Caliptra FMC + RT, SoC Manifest, MCU Runtime	Remaining SoC firmware
Data path	NBP → MCU ROM → Caliptra Recovery I/F	NBP → MCU RT → load address from SoC manifest
Boot model	Same as OCP streaming boot (I3C / USB)	MCU RT image loader
Trust gate	**Caliptra verifies, as in any streaming boot**	**Caliptra RT authorize() per image**

Optional flash write-back: recovered images are committed to flash after authorization (default) or after boot success.



SECURITY MODEL

The network delivers bytes. Caliptra decides trust.

UNTRUSTED

Network + NBP

- DHCP / TFTP unauthenticated, unencrypted
- NBP makes no trust decisions
- Worst case: denial of service

CONSTRAINED

MCU

- Writes only to the Recovery I/F or manifest-defined load addresses
- Acts on Caliptra's verdict

TRUSTED

Caliptra

- Same verification as flash or streaming boot
- SoC manifest binds addresses and digests
- authorize() gates every SoC image

Hardening the untrusted side

- Minimal DHCP / TFTP feature set
- Bounded chunk sizes, timeouts, retries
- NBP isolation: own ROM / SRAM, MCU-controlled reset, AXI filtering
- Image server on an isolated management network

Out of scope: image confidentiality in transit; availability against on-path attackers.

THE TAKEAWAY

From “stranded” to “recoverable”

NBP breaks the dependency on a single local firmware source.

1. **Local boot** — Try normal firmware
2. **Decision** — Usable image?
3. **NBP** — Find + stream firmware
4. **Caliptra** — Authenticate + authorize
5. **Continue** — Boot the platform

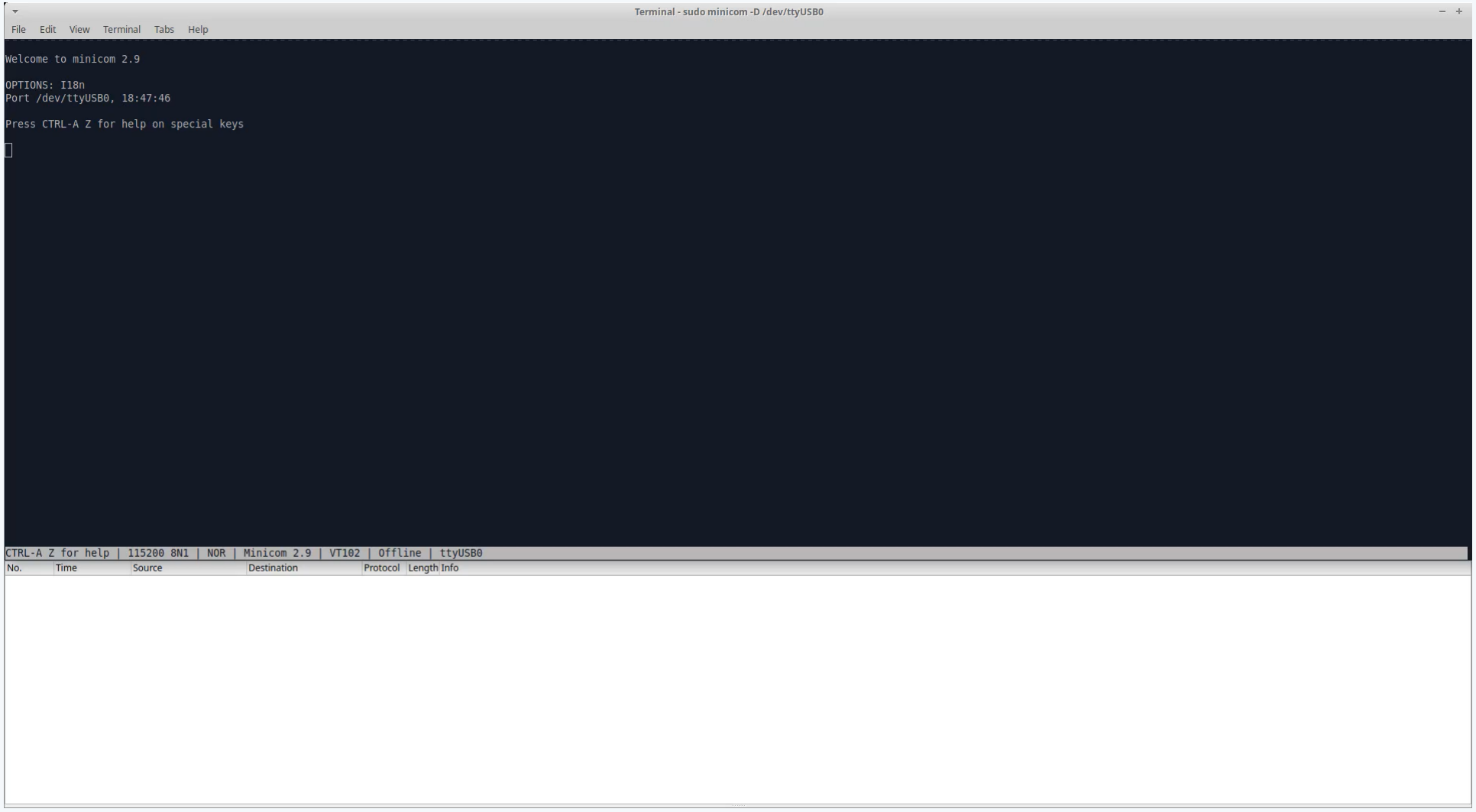
Optional: stage recovered images and write them back to flash for the next boot.



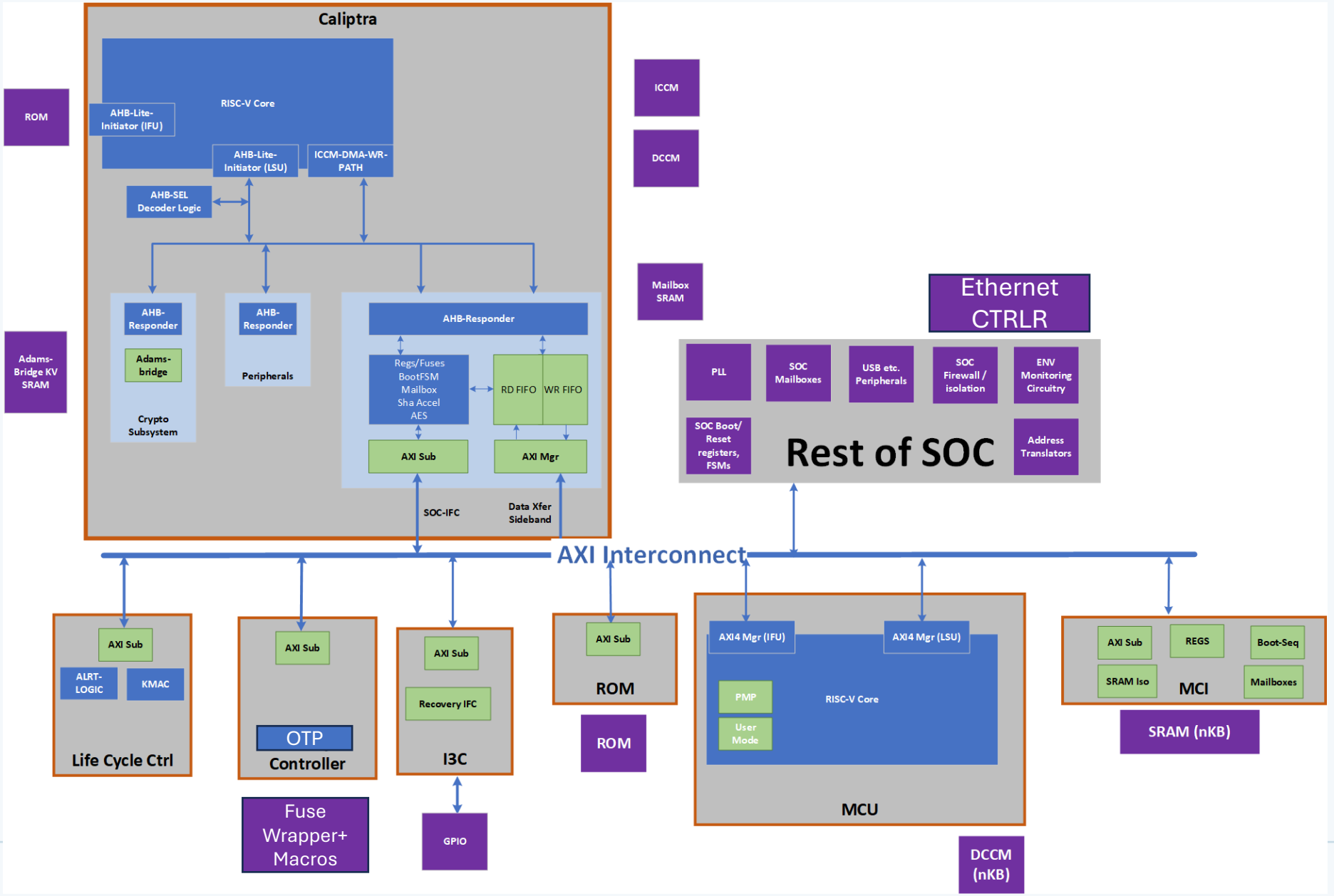
ASPEED DEMO / VIDEO

Network boot demo video

From early firmware loading to Linux kernel startup

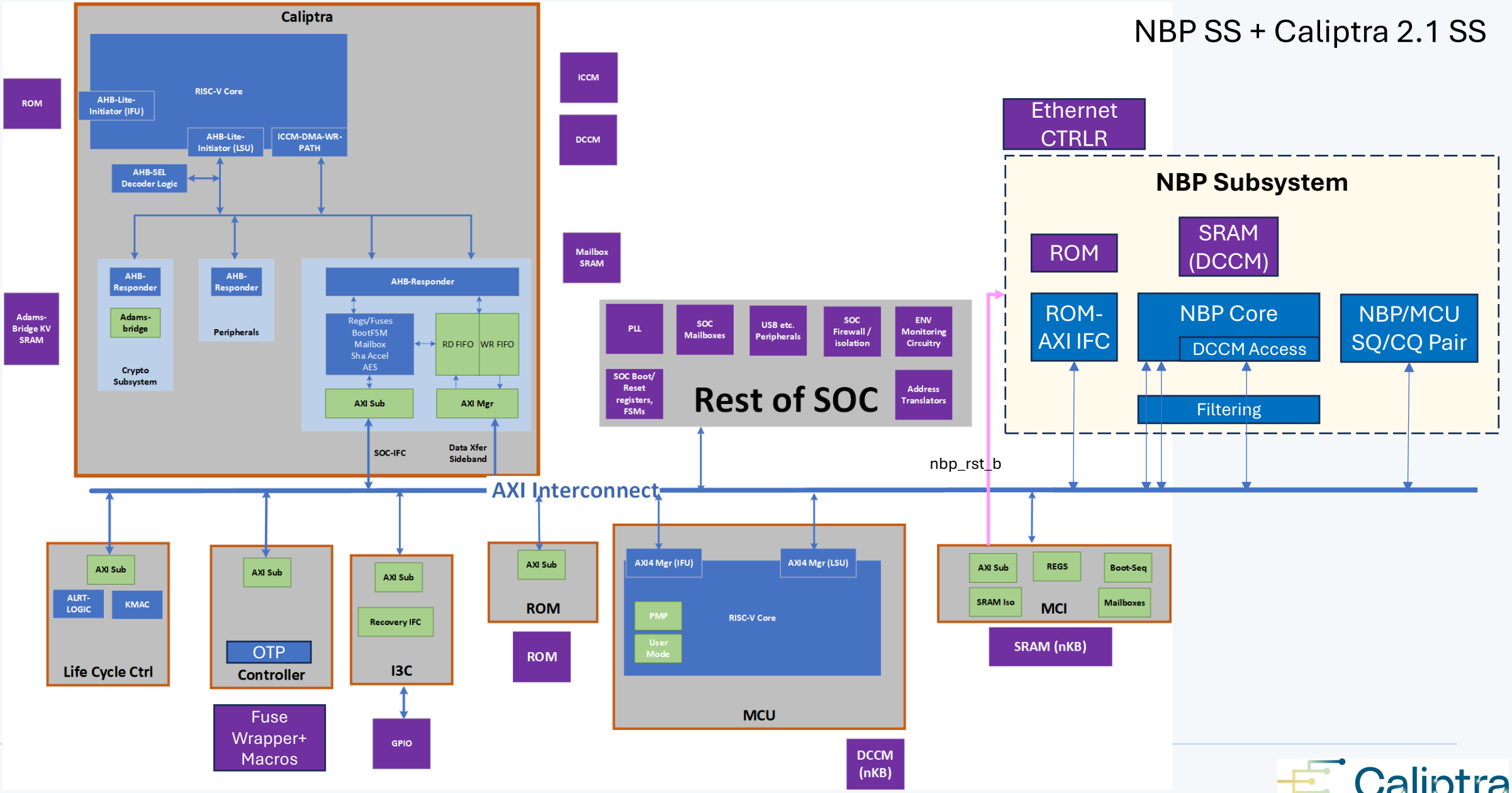


Building with Caliptra



Building with Caliptra

NBP SS + Caliptra 2.1 SS

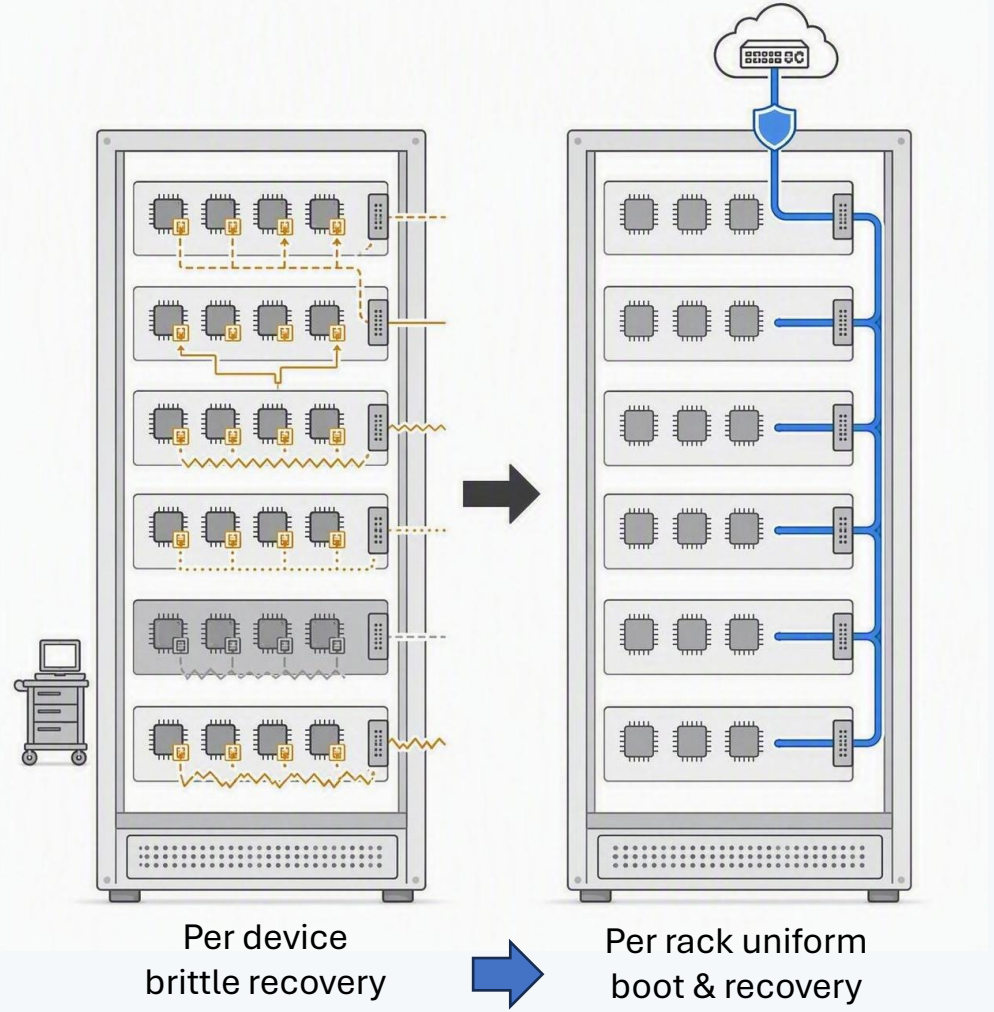


Caliptra

Next: What the NBP architecture enables

The architecture is less about one coprocessor and more about making recovery pervasive, uniform, and rack-scale ready.

- Eliminate the BMC's last mutable firmware flash dependency.
- Use one mainstream firmware-load path for both boot and recovery.
- Reduce/Eliminate custom-made GPIO, mux, timing, and physical-service recovery flows.
- Support faster, infrastructure-controlled update and recovery at rack scale.
- Preserve Caliptra as the authenticator while NBP remains the staging engine.
- Move upcoming BMCs toward native network-boot capability.



BOTTOM LINE: THE FIRMWARE LOAD INTERFACE BECOMES THE RECOVERY INTERFACE

More Information…

Network Boot Spec

MCU ROM Source Code

NBP Source Code

Network Boot Demo @ OCP – Visit MSFT Booth

Caliptra Onboarding for ODM Partners
Silicon Lifecycle & Debug

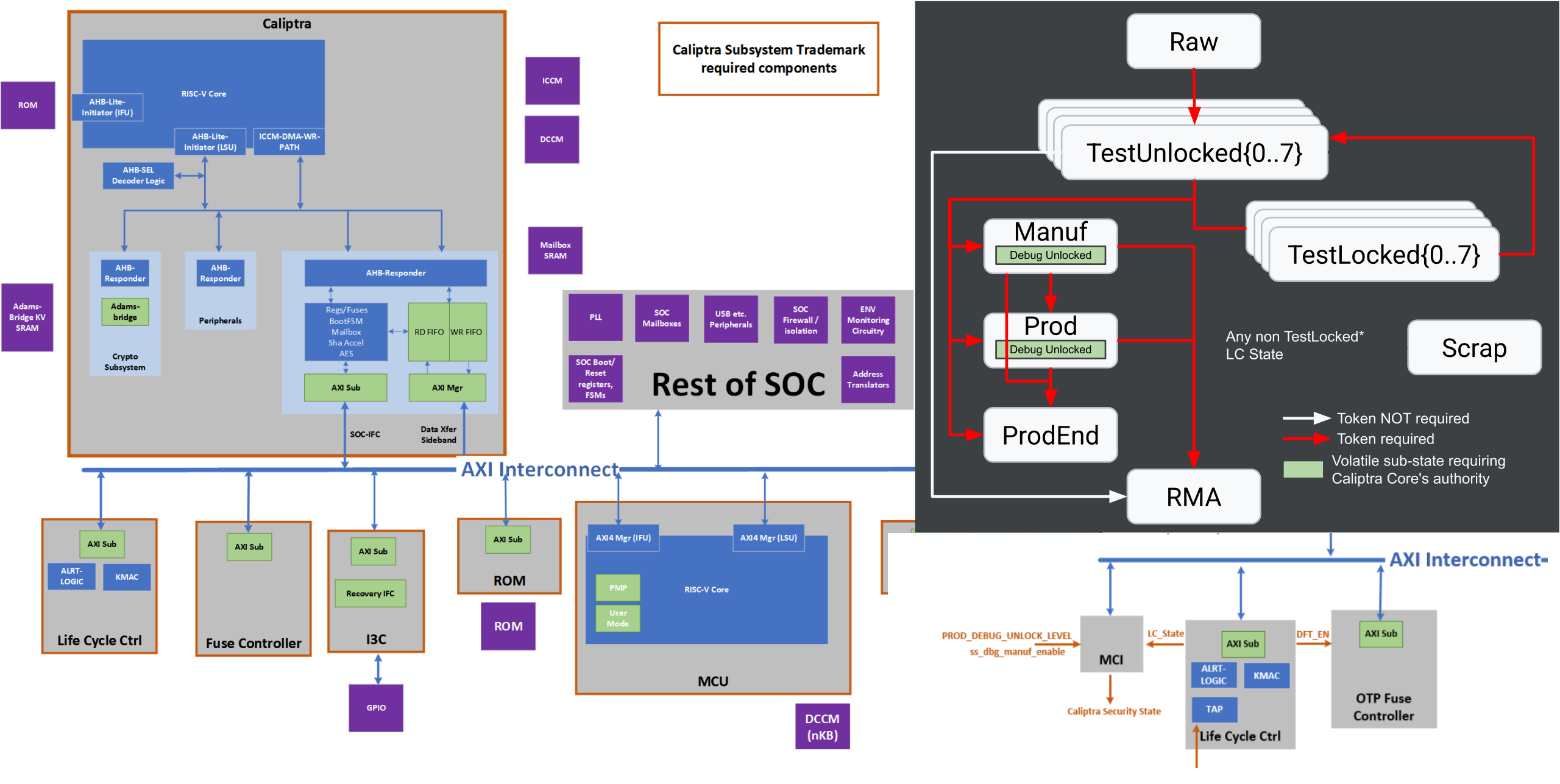
Device provisioning stages, fuses, and debug unlock

ATE / Factory

ODM Board Build

System Integration

Datacenter / RMA


Caliptra
RISC-V Core
AHB-Lite-Initiator (IFU)
AHB-Lite-Initiator (LSU)
ICCM-DMA-WR-PATH
AHB-SEL Decoder Logic
AHB-Responder
Adams-bridge
Crypto Subsystem
AHB-Responder
Peripherals
AHB-Responder
Regs/Fuses BootFSM Mailbox Sha Accel AES
RD FIFO
WR FIFO
AXI Sub
AXI Mgr
SOC-IFC
Data Xfer Sideband
ROM
ICCM
DCCM
Mailbox SRAM
Adams-Bridge KV SRAM
Caliptra Subsystem Trademark required components
PLL
SOC Mailboxes
USB etc. Peripherals
SOC Firewall / isolation
ENV Monitoring Circuitry
SOC Boot/ Reset registers, FSMs
Rest of SOC
Address Translators
AXI Interconnect
AXI Sub
ALRT-LOGIC
KMAC
Life Cycle Ctrl
AXI Sub
Fuse Controller
AXI Sub
Recovery IFC
I3C
GPIO
AXI Sub
ROM
ROM
AXI4 Mgr (IFU)
AXI4 Mgr (LSU)
PMP
User Mode
RISC-V Core
MCU
DCCM (nKB)
Raw
TestUnlocked{0..7}
Manuf
Debug Unlocked
Prod
Debug Unlocked
ProdEnd
RMA
TestLocked{0..7}
Any non TestLocked* LC State
Scrap
Token NOT required
Token required
Volatile sub-state requiring Caliptra Core's authority
AXI Interconnect
PROD_DEBUG_UNLOCK_LEVEL
ss_dbg_manuf_enable
MCI
Caliptra Security State
LC_State
AXI Sub
ALRT-LOGIC
KMAC
TAP
Life Cycle Ctrl
DFT_EN
AXI Sub
OTP Fuse Controller

Caliptra

Life-Cycle States Across the Provisioning Stages

Which LCC states each stage lives in, and where debug unlock, RMA and SCRAP become possible

01

ATE / Factory

Silicon vendor

- RAW
- TEST_UNLOCKED
- MANUF
- MANUF debug

02

ODM Board Build

ODM board + system build

- MANUF
- PROD
- MANUF debug
- PROD debug (level 0+)

03

System Integration

ODM system integration

- PROD
- PROD debug (level 0+)

04

Datacenter / RMA

CSP operations

- PROD
- PROD debug (level 0+)
- RMA
- SCRAP

From ODM board build onward: both MANUF and PROD parts can enter debug mode
End of life: RMA opens hardware debug; SCRAP disables the device.

Caliptra

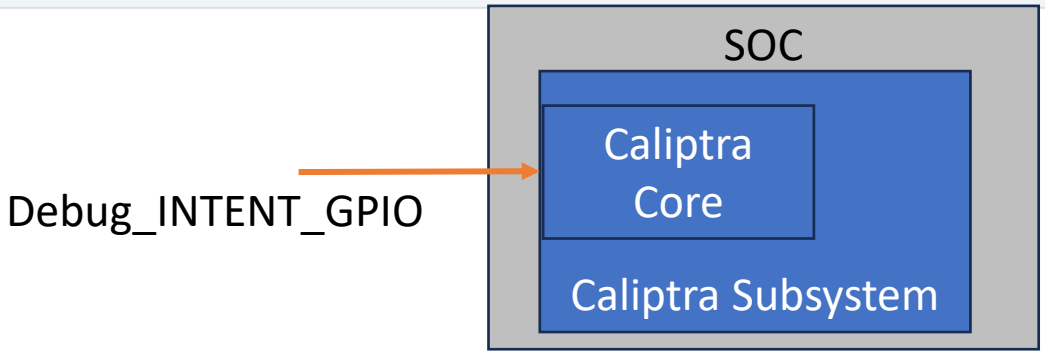
SoC Debug Capabilities and Expressing Debug Intent

What the Caliptra Subsystem offers the SoC at t=0, and the construction-specific paths for delivering the intent

What the subsystem provides

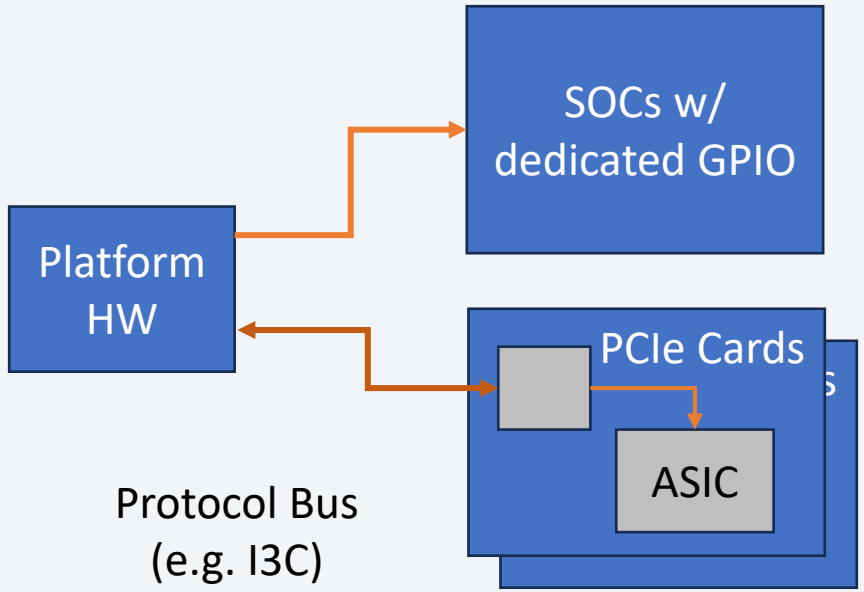
1. **Time-zero HW and ROM debug** — the debug intent is sampled only at t=0; once authorized, the Caliptra and MCU TAP interfaces open.
2. **Secrets first** — UDS and Field Entropy are wiped or never latched, so no identity keys can be derived in debug mode.
3. **Delivery is SoC construction specific** — how DEBUG_INTENT reaches the subsystem is the integrator's choice; authorization is unchanged.

Where the intent lands: the Caliptra Subsystem inside the SoC



Delivering the intent: Platform integration view

A dedicated GPIO, or a protocol bus such as I3C — the Caliptra-side contract is identical



Caliptra

SOC Hardware Debug over JTAG (t=0) Unlock Flow

What the debug manager, the SoC and the Caliptra Subsystem each do, from t=0 through runtime

1
Assert `SoC DEBUG_INTENT GPIO`
Debug Manager expresses the intent to debug before power good — it is only sampled at t=0.

2
Set `Caliptra BootFSMBrk`
Boot FSM is halted if we want to debug Subsystem with ROM.

3
SoC powers up
Power good reaches the subsystem;.

4
Caliptra samples the debug intent
At t=0 only. UDS and Field Entropy are wiped or never latched, so no keys can be derived.

5
ROM or RT FW evaluates the credential
Note: Don’t forget to assert BootGO if you set BootFSMBrk

6
Caliptra debug ports open

7
SoC debug ports open
MCI mask registers OR in DFT_EN, SOC_DFT_EN and SOC_HW_DEBUG_EN for the granted level.

8
ROM or RT FW runs with `debug ports open`

Caliptra

SOC Hardware Debug over JTAG Unlock Flow at RT

Platform wants to unlock SOC at a future time

Assert `SoC DEBUG_INTENT GPIO`
Debug Manager expresses the intent to debug before power good — it is only sampled at t=0.

2
Set `Caliptra BootFSMBrk`
Boot FSM is halted if we want to debug Subsystem with ROM.

SoC powers up
Power good reaches the subsystem;.

Caliptra samples the debug intent
At t=0 only. UDS and Field Entropy are wiped or never latched, so no keys can be derived.

SOC FW Loads (Caliptra, MCU etc. FW)

Debug Unlock Command Flow through Caliptra Core Mailbox (whenever unlock is required)

SoC (JTAG) debug ports open

ROM or RT FW runs with `debug ports open`

Caliptra

SOC Firmware Debug

No need to unlock JTAG
SOC Firmware = “Any” FW except Caliptra core
Pre-step: Set Debug bit in SOC manifest

1
Assert `SoC DEBUG_INTENT GPIO`
Debug Manager expresses the intent to debug before power good — it is only sampled at t=0.

2
SoC powers up
Power good reaches the subsystem;.

3
Caliptra samples the debug intent
At t=0 only. UDS and Field Entropy are wiped or never latched, so no keys can be derived.

4
Caliptra samples the debug intent
At t=0 only. UDS and Field Entropy are wiped or never latched, so no keys can be derived.

Load/Stream `Caliptra Core FW`

5
`Caliptra Core FW Runs`

6
SOC RT FW Manifest with debug bit = 1 loaded/streamed

7
Caliptra core checks if debug intent is set

8
SOC `debug FW images can run (if authentications pass)`

Caliptra

Caliptra Firmware Debug

No need to unlock JTAG
Pre-step: Set Debug bit in SOC manifest

1
Assert `SoC DEBUG_INTENT GPIO`
Debug Manager expresses the intent to debug before power good — it is only sampled at t=0.

2
SoC powers up
Power good reaches the subsystem;.

3
Caliptra samples the debug intent
At t=0 only. UDS and Field Entropy are wiped or never latched, so no keys can be derived.

4
Caliptra samples the debug intent
At t=0 only. UDS and Field Entropy are wiped or never latched, so no keys can be derived.

5
`Caliptra Core FW Loaded/Streamed`

6
Caliptra core ROM checks if debug intent is set

7
Caliptra Debug Image can run if passes authentication

Caliptra

Fuses, Lifecycle, and Debug Unlock Fuses

Fuse map v2.1 — what gets burned, and how a locked production part is opened for debug

Key fuse partitions

`SECRET_MANUF_PARTITION`

CPTRA_CORE_UDS_SEED — root of the identity chain

`SECRET_PROD_PARTITION_0..3`

CPTRA_CORE_FIELD_ENTROPY_0..3 — four one-shot slots

`SW_MANUF_PARTITION`

IDEVID cert attributes, HSM identifier, SoC stepping ID

`VENDOR_HASHES_MANUF/PROD`

Vendor PK hashes, owner PK hash and valid flag

`SW_TEST_UNLOCK_PARTITION`

CPTRA_SS_MANUF_DEBUG_UNLOCK_TOKEN

`SECRET_LC_TRANSITION_PARTITION`

Test unlock, TEST-to-MANUF, MANUF-to-PROD, RMA tokens

Debug unlock

Manufacturing debug unlock

Used while the part is still in MANUF lifecycle state. Needs to be provisioned in TEST_UNLOCKED state (unprovisioned security state).
The requester presents the token and Caliptra first applies sha512 and then compares it against fuses before releasing enabling debug.

Production debug unlock

For parts already in PROD. No shared secret is burned in.
Authorized by signature: eight CPTRA_SS_PROD_DEBUG_UNLOCK_PKS entries hold the public key hashes permitted to authenticate an unlock request, so debug authority can be delegated per level and revoked.
Signing authority is challenged with a nonce

Caliptra

Caliptra Onboarding for ODM Partners

Device provisioning & Certificate Extraction

Caliptra

What Caliptra Gives the Device

A fuse-rooted identity chain, built once in silicon and endorsed through manufacturing

IDEVID

UDS seed in fuses. CSR extracted at ATE

LDEVID

UDS CDI is combined with Field-entropy and then generates this certificate

ALIASFMC

FMC bound attestation

Roots that must exist before any of this works

UDS seed and vendor public key hashes are burned at ATE, so the part reaches your line already in PROD lifecycle state, secure-boot enforced, running production-signed firmware only.
Field entropy is programmed last — after it, LDEVID and everything below it change. Order matters.

Caliptra

Provisioning Stages and Who Does What

Each stage hands off to the next one a device plus a data set — certificates, CSRs, and serial numbers

01 ATE / Factory

Silicon vendor

- Vendor SoC specific fuses (e.g. PK hash, LCC tokens, UEID etc.)
- Manage lifecycle Transitions
- Provision root seeds (e.g. UDS-seed) and CSR extraction

02 ODM Board Build

ODM board + system build

- Device has a fully functional iRoT
- CSP specific flows happen here (e.g. FE programming, LDEVID flow, Owner PK hash)
- I3C available to BMC OOB communication
- Prod Debug Unlock available for HW debug
- Debug FWs can be loaded securely

03 System Integration

ODM system integration

- Some ODM board build functions may run here (CSP specific flows)
- All ODM board build capabilities continue to be available (e.g. Endorse ALIASFMC certificate)
- DOT Flow for system (typically happens here)

04 Datacenter / RMA

Microsoft operations

- Extract manufacturing, infra-owner, tenant cert chains
- Attestation using signed firmware measurements

Caliptra

Caliptra

SSDs & Caliptra Subsystem

State of the SSD Ecosystem

Ulrich Hansen, October 2026

Caliptra for SSDs

Background

- A step-function improvement for SSD Security, Recoverability and circularity
- Convergence & adoption conversations with SSD ecosystem for ~ 2 years



Objectives for Caliptra Integration for SSDs

- Consistent Caliptra SSD integrations, capabilities and behaviors
- Align SSD ecosystem on integration choices & timelines
- Maintain Backward compatibility
- Leverage industry standards

Caliptra

SSD-Specific Caliptra Integration Considerations

A Caliptra Integration Guidance for SSDs

- Backward Compatibility
  - SMBUS needs
  - Boot FW from NV-Storage
  - NVMe-MI and legacy flows
- Streaming Boot
  - Support for streaming boot mode through SNIA SFF
- Industry Standards
  - TCG Storage feature alignment with OCP LOCK
  - Clarifications & Conflict resolutions with specs. Eg: SSD boot timing

Caliptra

SSD Ecosystem State and Outlook

The Caliptra journey for SSDs has Begun

- Several PCIe Gen6 SSDs with Caliptra Subsystem 2.1 in CY27
  - Some vendor-to-vendor variations exist (e.g. OCP DOT)

- Opportunity for more consistency in the next gen Caliptra integrations
  - Consistent support for OCP DOT
  - SSD standard solution for physical presence detection for sensitive flows (e.g. RMA)
  - USB out-of-band support with Caliptra 2.2 adoption
  - Work with ecosystem to close gaps against Caliptra SS trademark

Thank you for making Caliptra SS for SSDs a reality!

Caliptra

Confidential Computing and Caliptra in the Era of AI

Building verifiable trust from silicon to models, data, and inference

PROTECT | MEASURE | ATTEST



Roksana Golizadeh-Mojarad, Microsoft

Why AI changes the trust equation

AI systems combine valuable data, valuable models on a heterogeneous hardware/software stack.

High-value Assets

Protect model weights, data, and outputs.

Silicon heterogeneity

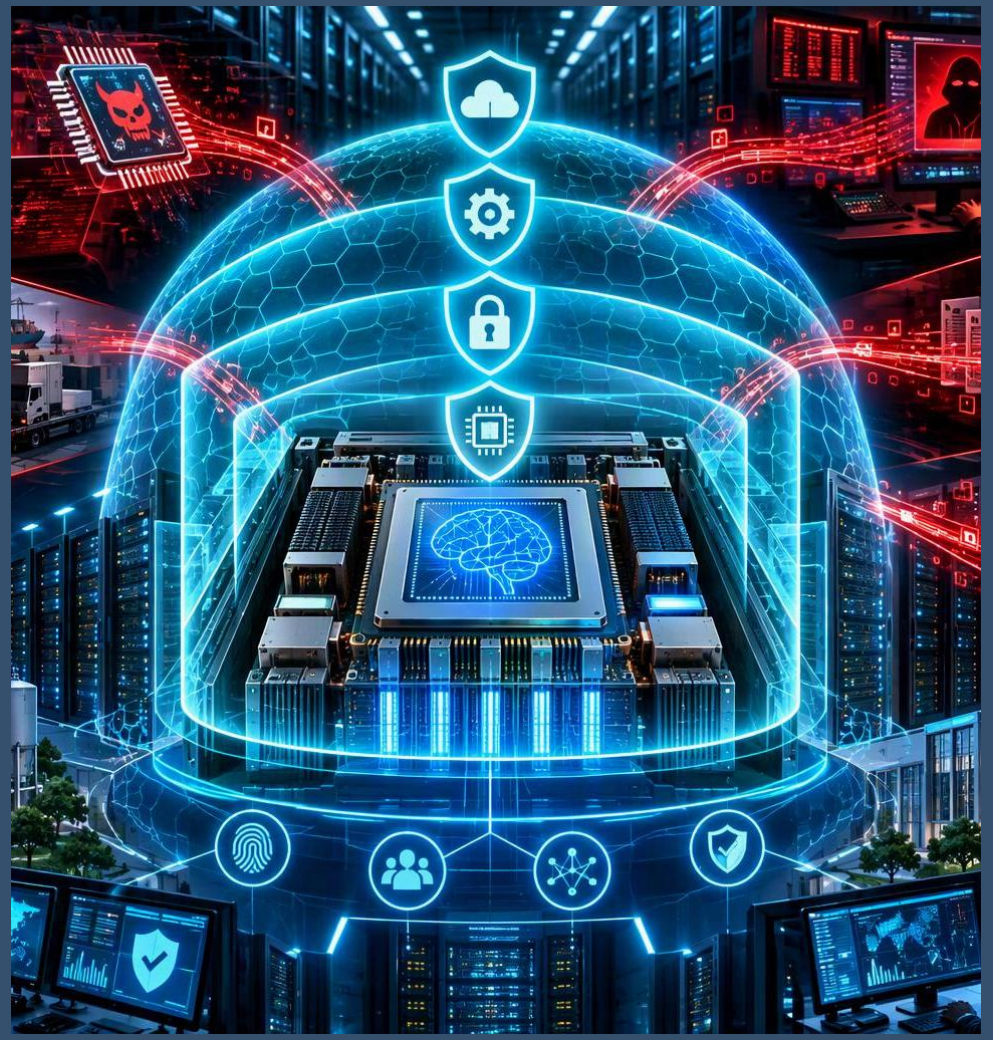
CPUs, GPUs, NICs/DPUs, storages and accelerators expose different trust models and evidence.

Expanded attack surface

More hardware, firmware, software, and pipelines expand risk.

Trust must be established

Trust needs transparent, signed evidence.
Fragmented proprietary RoT for CPUs and GPUs, … creates visibility gaps.



Core shift

Shift to verifying where AI model runs before sharing model and data.

Confidential computing: protect data while it is in use

A hardware-based trusted execution environment (TEE) isolates code and data from higher-privileged software.

1

Encrypt + isolate

Protect workload memory and execution inside a hardware-enforced boundary.

2

Measure

Create cryptographic measurements of the launched firmware and hardware configuration.

3

Attest

Sign evidence so a remote verifier can assess identity and security posture of system.

4

Release by policy

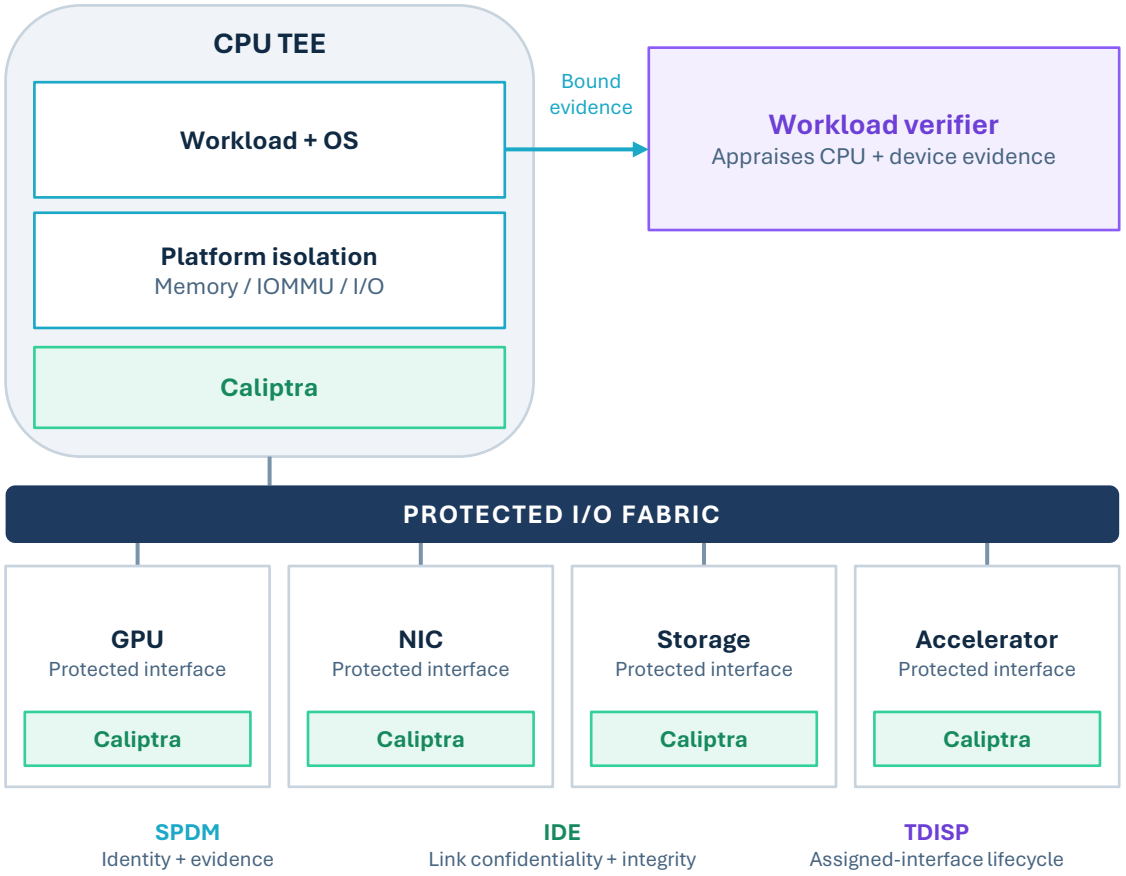
Release models, or data only when evidence satisfies policy.

Outcome: confidential AI enables sensitive workloads to execute with stronger protection and verifiable launch conditions.

Caliptra: a silicon root of trust for cloud devices

Caliptra defines a minimal, composable set of root-of-trust services implemented in an SoC or ASIC.

A confidential platform rooted in Caliptra



Device identity

Hardware-anchored identities support authentication and lifecycle trust.

Measured boot

Integrity measurements extend from immutable code into mutable firmware.

DICE / DPE chain

Layered identities and measurements bind keys to software state.

Sign evidence

Attestation exposes verifiable evidence to platform and workload policy.

Design intent

Consistency across diverse cloud hardware, without forcing every device into the same architecture or vendor-specific root of trust.

Complementary roles: Caliptra + confidential computing

Caliptra establishes platform trust; confidential computing enforces workload isolation.

TRUST FLOWS UPWARD · Measurements and identities chain from immutable silicon into firmware and workload state

1 LAYER 1

Caliptra silicon root of trust

Device identity, cryptographic roots, DICE/DPE chain, sign evidence

2 LAYER 2

Device firmware + configuration

Secure/measured boot and authorized updates

3 LAYER 3

CPU / GPU / accelerator TEE

Hardware isolation, memory protection, TEE evidence

4 LAYER 4

AI runtime / confidential VM or container

Protected execution and workload measurement

5 LAYER 5

AI model + data

Policy-gated secrets, prompts, weights, training data

POLICY FLOWS DOWNWARD · Verifiers decide whether to admit a node, schedule a job, or release AI assets and data

End-to-end attestation for an AI job

The trust decision should happen before sensitive models or data material crosses the boundary.

1

Measure

Caliptra and attesting FW record and log TCB measurements, configurations, and policy.

2

Generate Attestation

Attesting FW generates attestation report and Caliptra signs evidences with its DICE identity Chain keys.

3

Verify + evaluate

An attestation service validates evidence and evaluates expected-state policy.

4

Authorize

The control plane admits the node and releases models and data only on success.

5

Monitor

Re-attest after firmware update, migration, or policy updates.

Core shift: AI infrastructure moves from *"trusting corporate promises"* to **"verifying cryptographic reality"**

What this enables for AI

Security becomes an admission-control and data-release property, not only a perimeter property.

Confidential training 01

Protect proprietary or regulated datasets and model weights during computation.

Confidential inference 02

Keep prompts, retrieved context, model weights, and outputs inside an attested environment.

Multi-party AI 03

Allow organizations to collaborate without broadly exposing raw data to operators or peers.

Trusted accelerator pools 04

Schedule only onto devices whose identity, firmware, and confidential-compute state satisfy policy.

Verifiable AI services 05

Provide evidence about the platform state behind a service before clients share sensitive inputs.

Supply-chain assurance 06

Connect device provenance and lifecycle checks to production admission and remediation.

Architecture decisions and open challenges

The technology is powerful, but production trust depends on clear boundaries, evidence semantics, and lifecycle operations.

Attestation composition architecture

How are CPU, GPU, DPU, BMC, accelerator, firmware, and workload claims combined without ambiguity?

Reference values & Endorsements

How are endorsements, reference values defined consistency across the ecosystem?

Policy ownership

Who defines acceptable firmware versions or security version, debug state, migration policy, monitoring policy and revocation behavior?

Lifecycle + updates

How do rollback protection, key rotation, hitless update, and re-attestation interact?

Design principle: minimize implicit trust. Make identities, measurements, endorsements, policies, and decisions explicit and auditable.



Caliptra Roadmap

USB on its way
Crypto Agility
Power, Performance & Area optimizations

October 12th, 2026



NXP Contributions to Caliptra Open Ecosystem

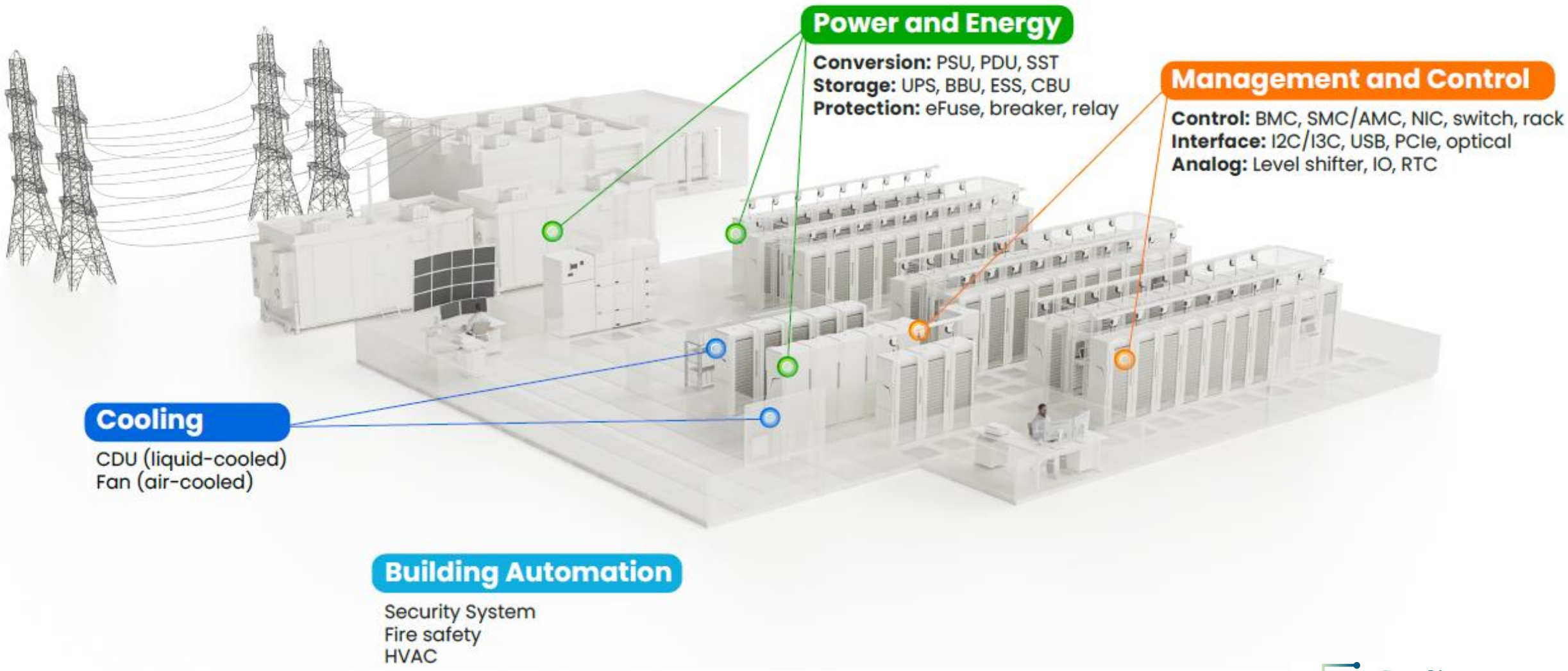
Bart Vertenten

October 12, 2026

| **Public** | NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2026 NXP B.V. Version 3.0.

NXP Data Center Segmentation and Focus Areas

nxp.com/DataCenter



Caliptra

Data center - Trends

Data Center Management & Security

- Board Management Controllers (BMCs) provide platform management
- Satellite Management Controllers (SMCs)
  - Enhance security as RoT
  - Extending management capabilities through OBMF
  - Enable scalable & disaggregated architectures

Industry transition toward USB for Boot and Management

- USB is everywhere in consumer space! Now USB everywhere in DC is the new theme!
- Why? – Go to USB Everywhere panel talk at the OCP

Security is at the Forefront through Consistency & Transparency

- Post-quantum security is a must-have capability.
- Open-source Caliptra Subsystem emerging as a widely adopted Root of Trust for securing infrastructure.

Caliptra

NXP as a Trend Setter
With Leadership, Contribution and Collaboration

- **Open HW ecosystem is a new frontier**
  - New to the world of hardware but not new concept
  - NXP enabling through Contribution & Consumption

- **Collaboration & Communication are Key**
  - Clearly defined required features & capabilities
  - Encouraged and Enabled collaborative work between NXP & Caliptra

- **Results Speak!**
  - Started with NXP's contribution of a well proven silicon USB IP
  - USB hub feature added by NXP
  - USB OCP streaming boot added by Microsoft
  - 21 USB PRs & 12 Caliptra SS PRs into GIT repos spanning design, validation, integration, documentation (more coming)
  - Intense Collaboration to create Caliptra SS FPGA with USB2
  - **USB streaming boot for OCP Demo**
  - All this work in ~ 3 months!

Caliptra

A History of NXP USB 2.0 RTL Module

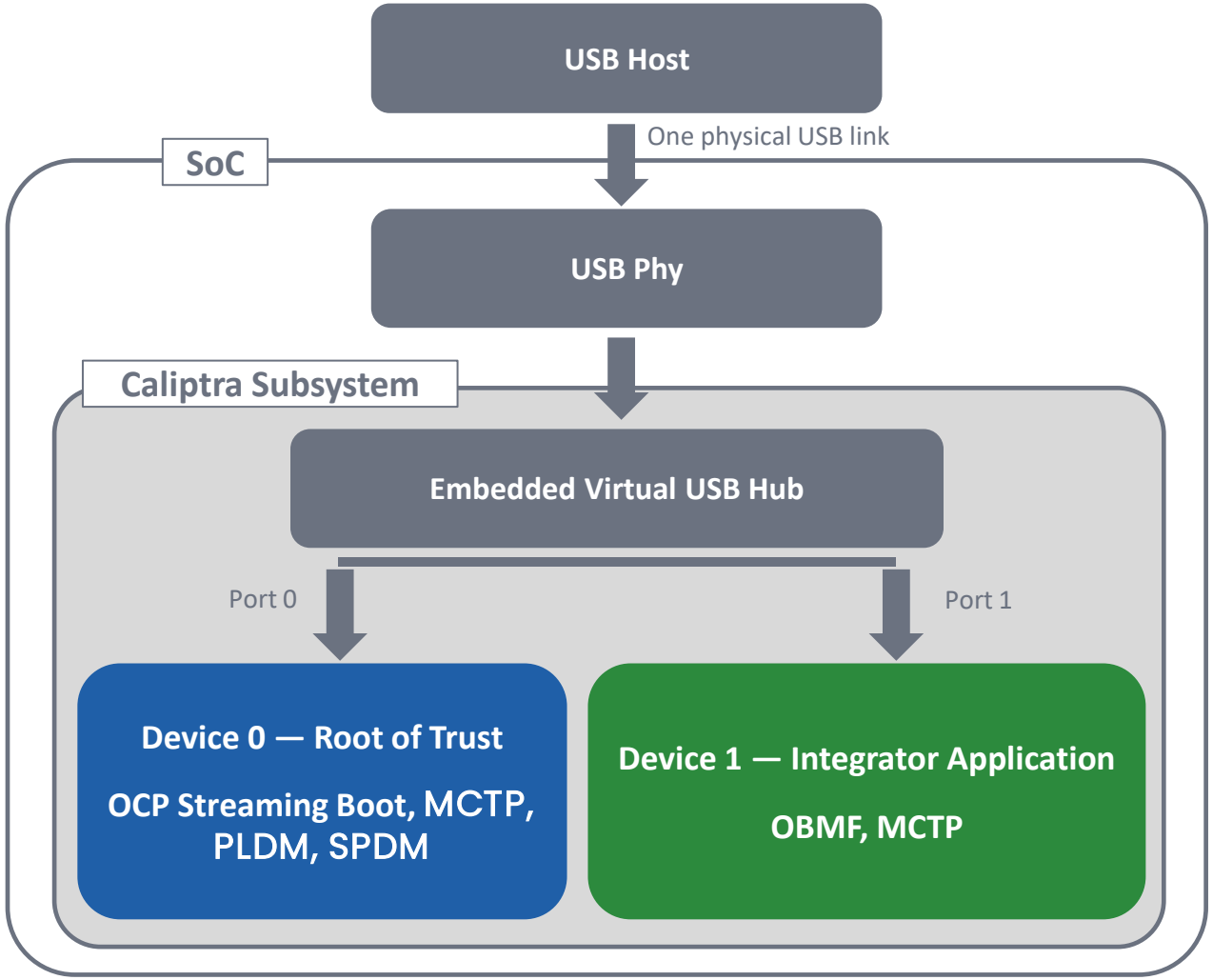
- Philips/NXP has a long history in USB2.0 standards and IP development
  - Major contributions to USB2.0, eUSB2 v1.2 and UTMI+/ULPI specification
  - Internal USB2 IP development for USB2 host, device and hub controllers for more than 25 years.
  - NXP USB2 IP controllers have been used in multiple Philips/NXP products and licensed to external companies.

- USB2 controller IP is a commodity IP

- Upgrading Caliptra with USB2 comes with major benefits compared to I3C
  - Higher bandwidth for the same number of pins and voltage levels.
  - Simplified USB hub architecture
    - Single USB port to support 2 independent on-chip microcontrollers

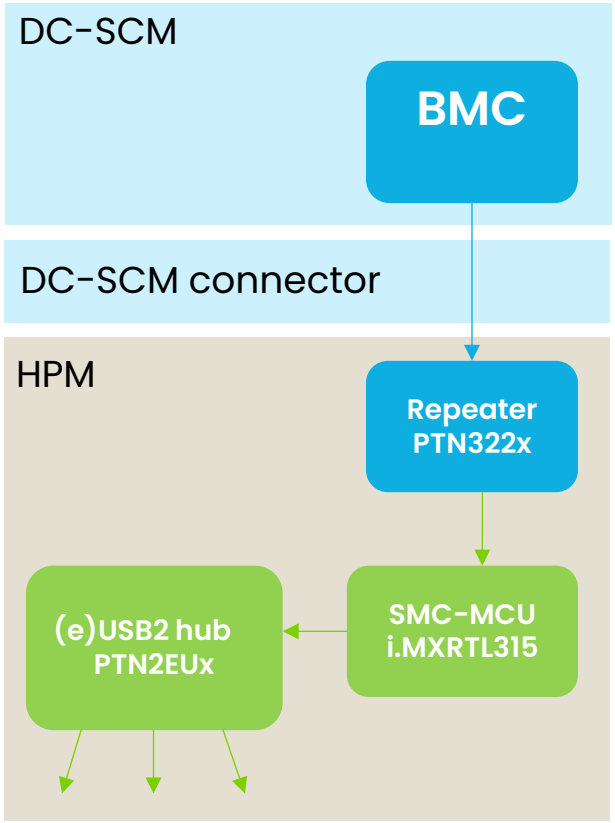
Porting legacy “IP” to public domain

- Synchronizing internal IP database with Caliptra git repository
  - Each internal release to be submitted as pull request on Caliptra
- USB2 controller IP updates only needed to support new requirements
  - Hub functionality
  - OCP streaming boot
- Code review by CoPilot agent in Github provides valuable inputs to improve code.
- Verification testbench to be upgraded with USB2 VIP
- FPGA implementation of controller IP to run USB2.0 compliance tests



Caliptra

A NXP's Application Example for Next Gen Platforms



- NXP delivers multiple components to data center applications and supports the OCP trend
  - MCX N and MCX A product family
  - i.MXRTL315 supports (e)USB2.0 embedded virtual hub and USB2 streaming boot
  - eUSB2 repeater, hub and bridges
- Scalable to Accelerator Modules
- Supports OBMF over USB

Caliptra

Get in touch

Bart Vertenten

bart.vertenten@nxp.com

nxp.com

| **Public** | NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2026 NXP B.V. Version 3.0.

Caliptra

Thank You

Caliptra



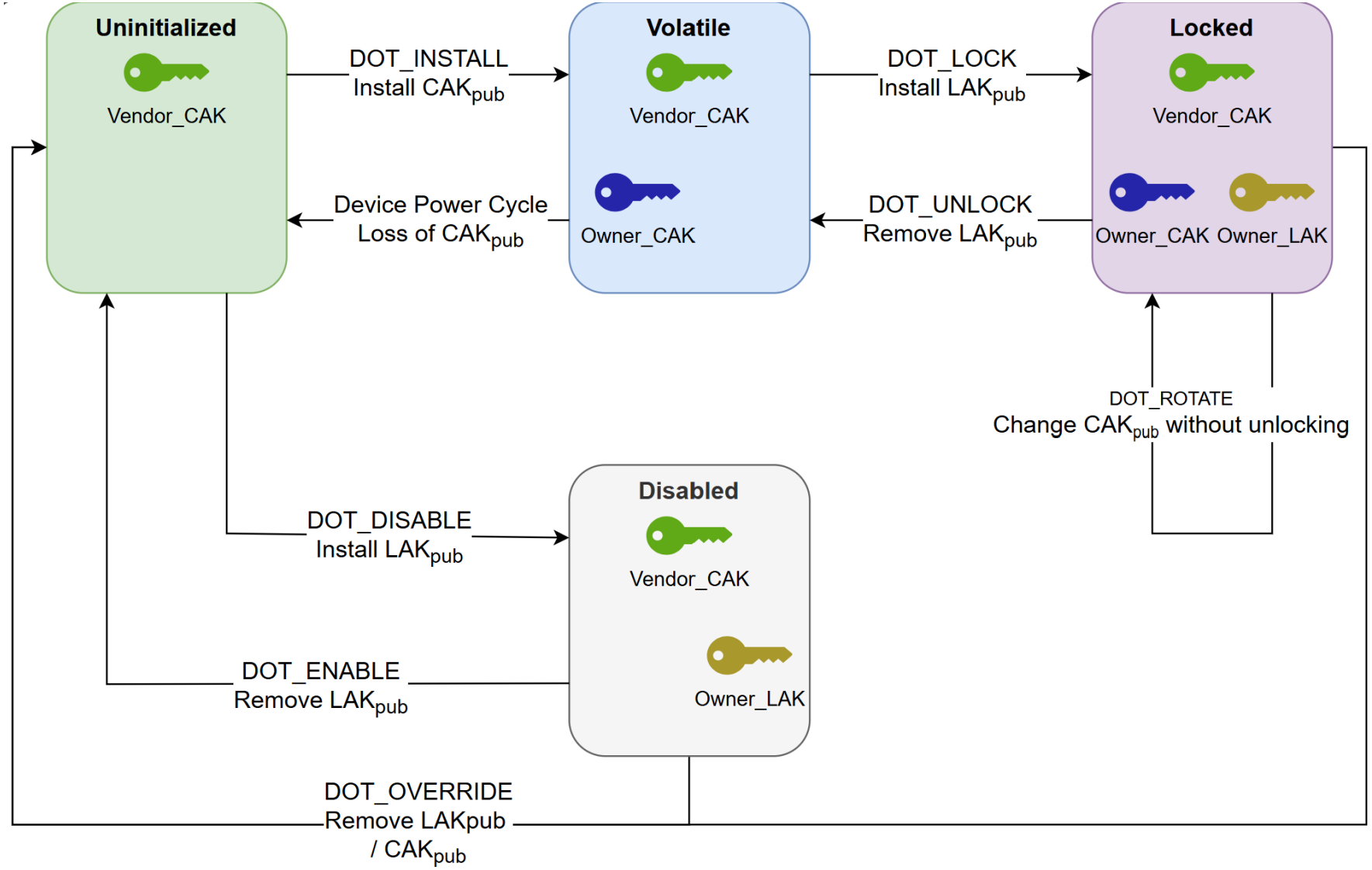
DOT In Caliptra

Vishal Mhatre, Emre Karabulut

DOT Key Terms

Term	Meaning
Code Authentication Key	Owner's public key used to verify owner-signed firmware. The corresponding private signing key stays with the owner.
Locking Authorization Key	Public key used to verify authorization to unlock/release ownership. The holder of the corresponding private key authorizes the operation.
DOT Effective Key	Key derived from the DOT root and fuse-state counter; used to HMAC-authenticate the DOT blob.
DOT Blob	Persistent ownership record containing CAK/LAK information, metadata, and an authentication tag.
DOT Fuse Counter (n)	Number of burned DOT fuse bits. Tracks ownership transitions and binds the blob to the current ownership generation.
Mutable Locking DOT	Ownership persists across power cycles in an authenticated, device-bound DOT blob, and releasing ownership requires a LAK-authorized unlock.
Volatile DOT	The owner's CAK is stored in volatile memory and must be reinstalled after each power cycle.

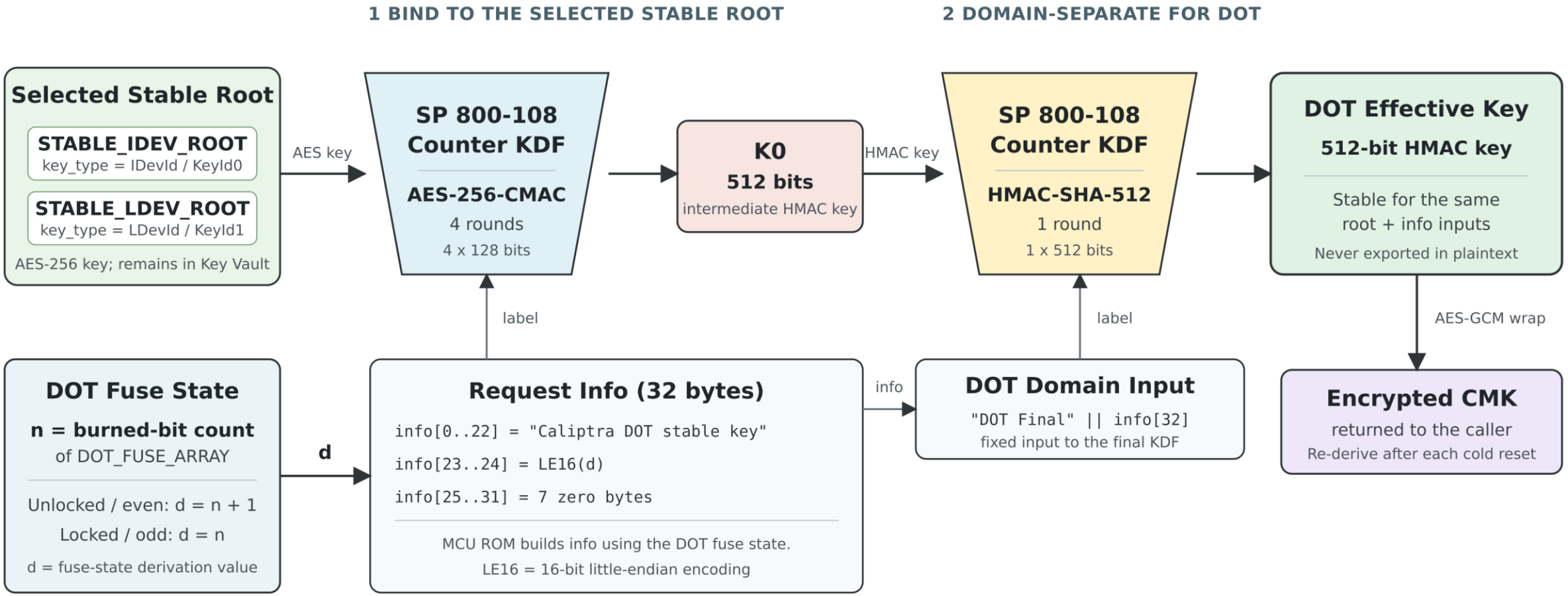
OCP DOT Flow



Caliptra

DOT Effective Key Derivation

CM_DERIVE_STABLE_KEY derives a device-bound HMAC key tied to the DOT fuse state.



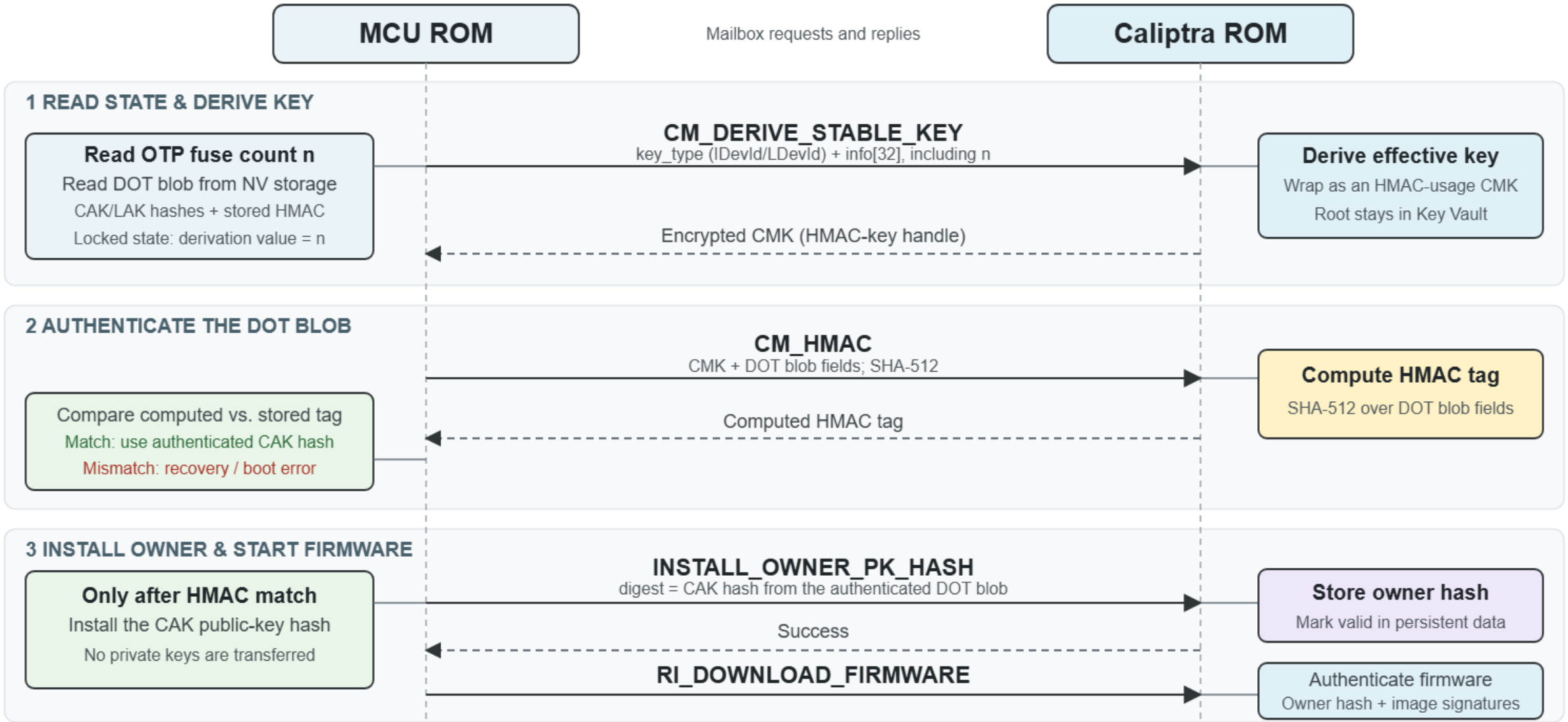
Purpose: use the returned CMK with CM_HMAC to seal or authenticate the DOT blob.

Caliptra

DOT Owner-Key Installation at Cold Boot

MCU ROM drives the DOT policy; Caliptra ROM provides cryptographic and mailbox services.

Scope: initialized, locked DOT (odd n), CAK present; Caliptra ROM mailbox ready.



DOT "key install" means installing the authenticated CAK hash, not importing private keys.

Caliptra

DOT Commands

Caliptra MCU runtime commands for Device Ownership Transfer.

Command	What it does
`DOT_STATUS`	Report DOT enablement, lock state, and burned-fuse count.
`DOT_ENABLE`	Permanently enable DOT without installing an owner.
`DOT_LOCK`	Install CAK/LAK hashes and persistently lock device ownership.
`DOT_DISABLE`	Lock DOT without a CAK; require LAK authorization to unlock.
`DOT_ROTATE`	Replace CAK/LAK hashes and advance the DOT fuse count by two.
`DOT_UNLOCK_CHALLENGE`	Generate a fresh challenge for a LAK-authorized ownership unlock.
`DOT_UNLOCK`	Verify LAK signatures and release locked ownership.
`GET_DOT_BACKUP_BLOB`	Export the current DOT blob for backup and recovery.
`DOT_RECOVERY`	Authenticate a backup DOT blob and restore the active record.
`DOT_OVERRIDE_CHALLENGE`	Validate vendor recovery keys and issue a fresh challenge.
`DOT_OVERRIDE`	Verify vendor recovery signatures and reset locked ownership.

Caliptra