ETH zürich

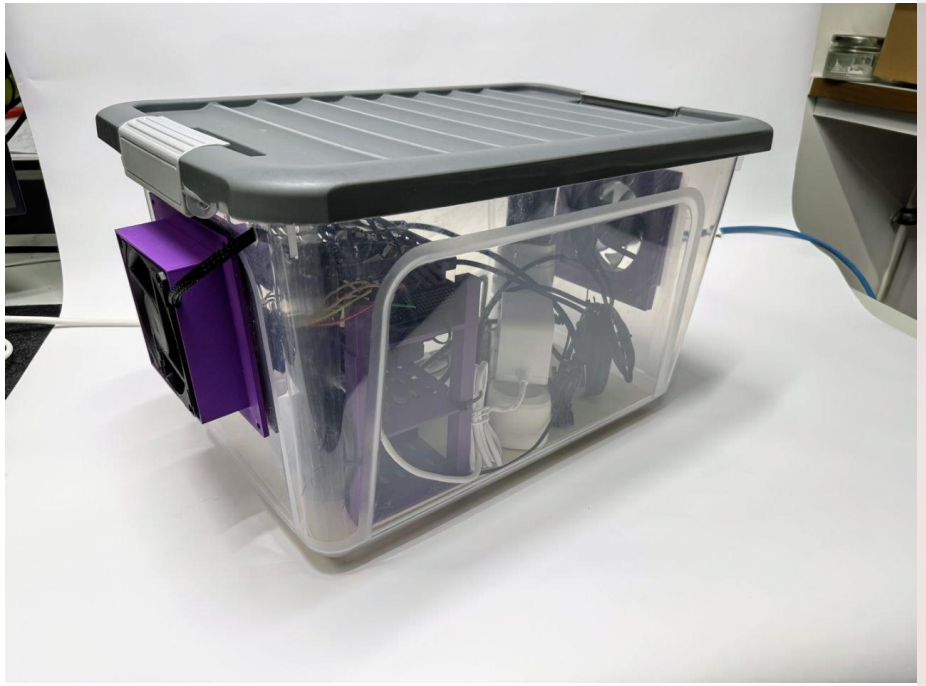
OpenCCA: An Open Framework to Enable Research on Arm CCA

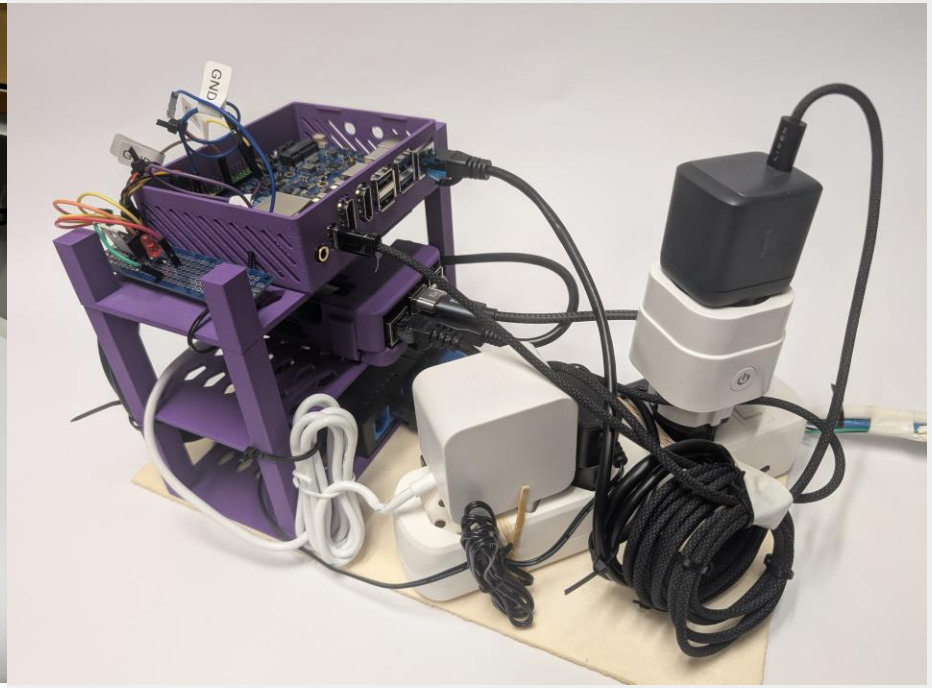
https://opencca.github.io

FOSDEM2026

Andrin Bertschi
PhD Student
Secure & Trustworthy Systems Group

Spoiler: I brought the Box



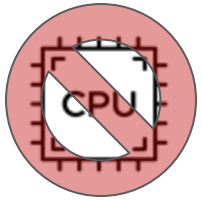


Later: **Live Demo**

Arm CCA Hardware, when?

Main Challenge on Arm CCA:

- No public Arm CCA hardware yet



Initial Rollout for Data Centers:

Locked-Down?
Flashable custom firmware?

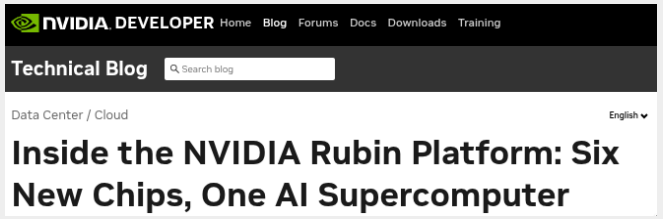
Documentation?
Technical reference manuals?

Affordable?
Enterprise only?

FUJITSU: Monaka (2027)

Microsoft: Cobalt 200 (2026)

Likely: Nvidia: Vera (2026)

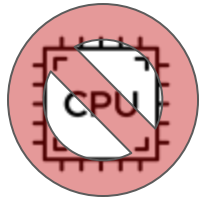


- https://global.fujitsu/en-global/technology/research/fujitsu-monaka
- https://techcommunity.microsoft.com/blog/azureinfrastructureblog/announcing-cobalt-200-azure%E2%80%99s-next-cloud-native-cpu/4469807
- https://developer.nvidia.com/blog/inside-the-nvidia-rubin-platform-six-new-chips-one-ai-supercomputer/

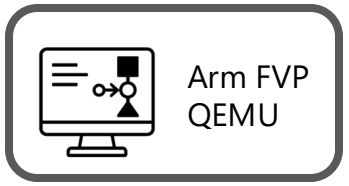
Arm CCA, today?!

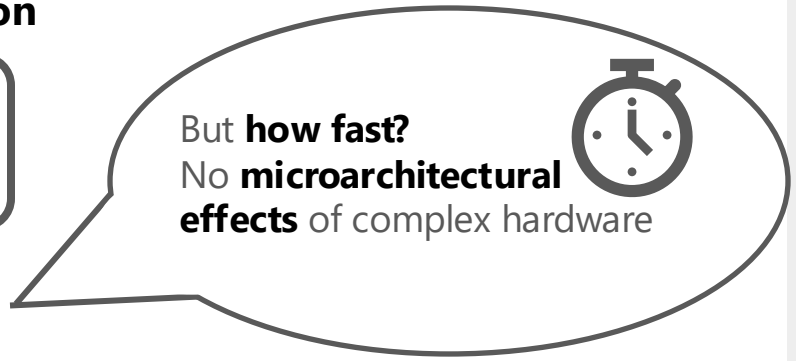
Main Challenge on Arm CCA:

- No public Arm CCA hardware yet



1. Under Simulation



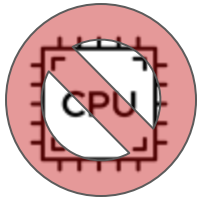


✓ Functionality
✓ Compatibility

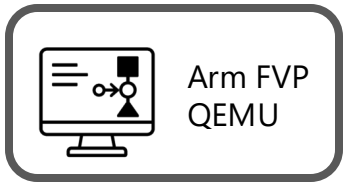
Arm CCA, today?!

Main Challenge on Arm CCA:

- No public Arm CCA hardware yet



1. Under Simulation



- ✓ Functionality
- ✓ Compatibility

2. Under Custom Prototype

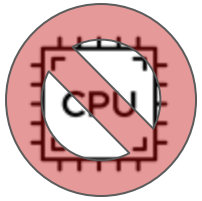
- ⛒ Closed Source
- ⛒ Difficult to compare
- ⛒ Difficult to reproduce
- ⛒ Repeated engineering

For Research:
Approximate research design
on real Armv8 Hardware

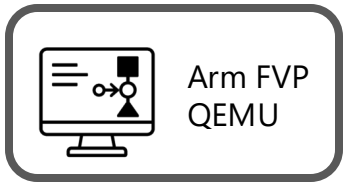
Arm CCA, today?!

Main Challenge on Arm CCA:

- No public Arm CCA hardware yet



1. Under Simulation



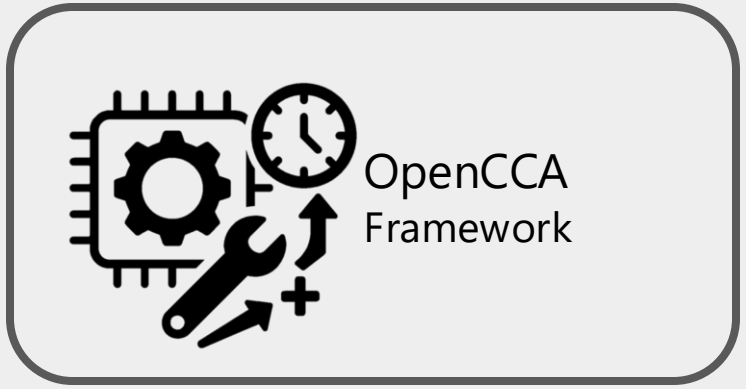
- ✓ Functionality
- ✓ Compatibility

2. Under Custom Prototype

- ⛒ Closed Source
- ⛒ Difficult to compare
- ⛒ Difficult to reproduce
- ⛒ Repeated engineering

The Need for Open Framework for Performance Evaluation

3. Under OpenCCA



OpenCCA Design Goals

Minimal changes to CCA reference stack → Preserve functionality

No security guarantees
Only for benchmarking & accelerator support

Target: **Affordable** & **Open** Armv8 Boards

Focus on ***reusable*** **Framework**
Not specific to a board
Performance estimation

Step 1: Simulation

Arm FVP
QEMU

Validate design

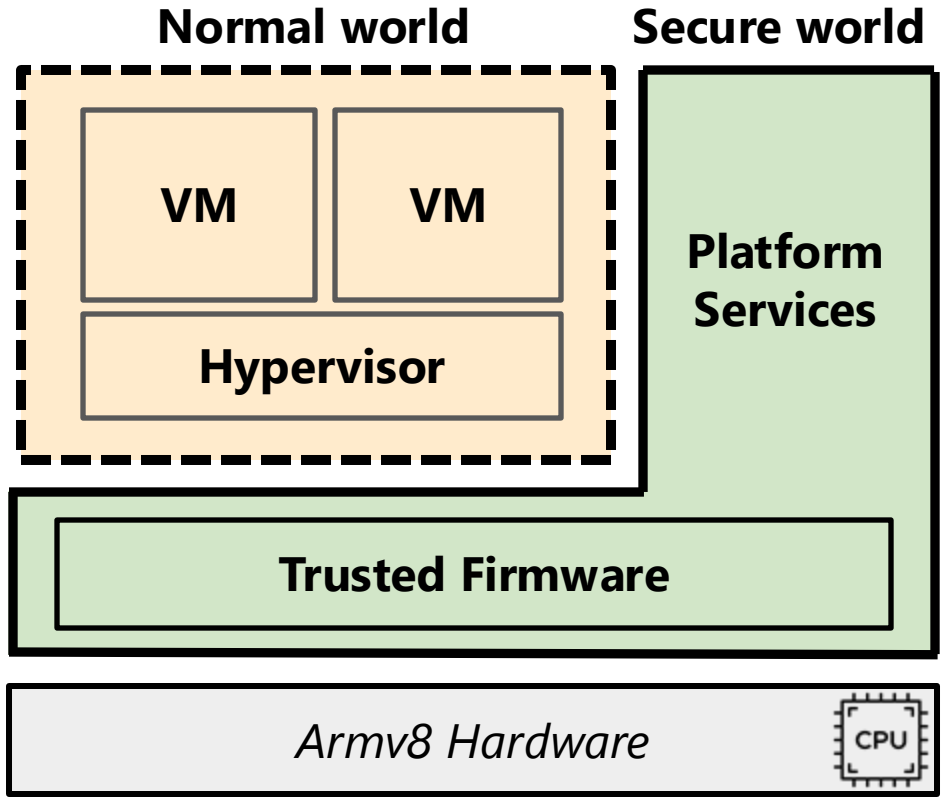
Step 2: OpenCCA

OpenCCA
Framework

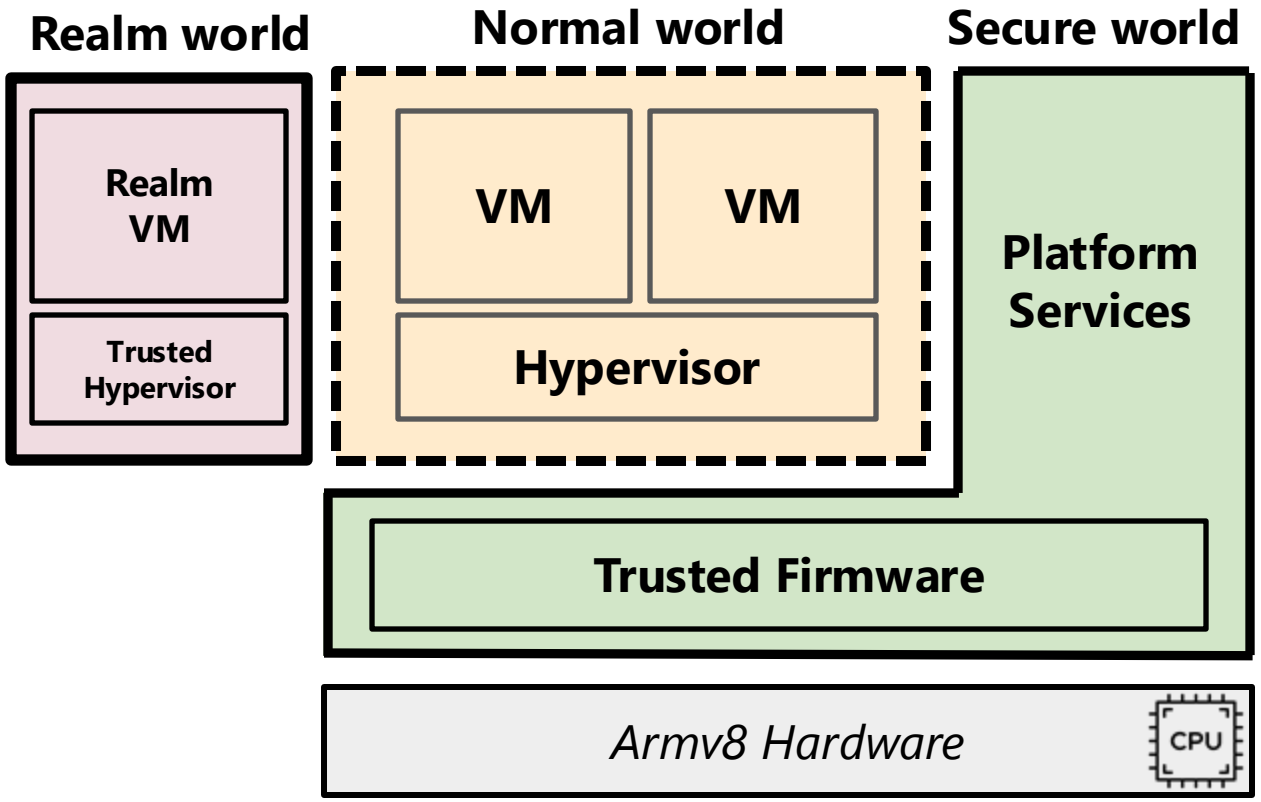
Approximate Performance & Interact with real HW

Background on Arm CCA

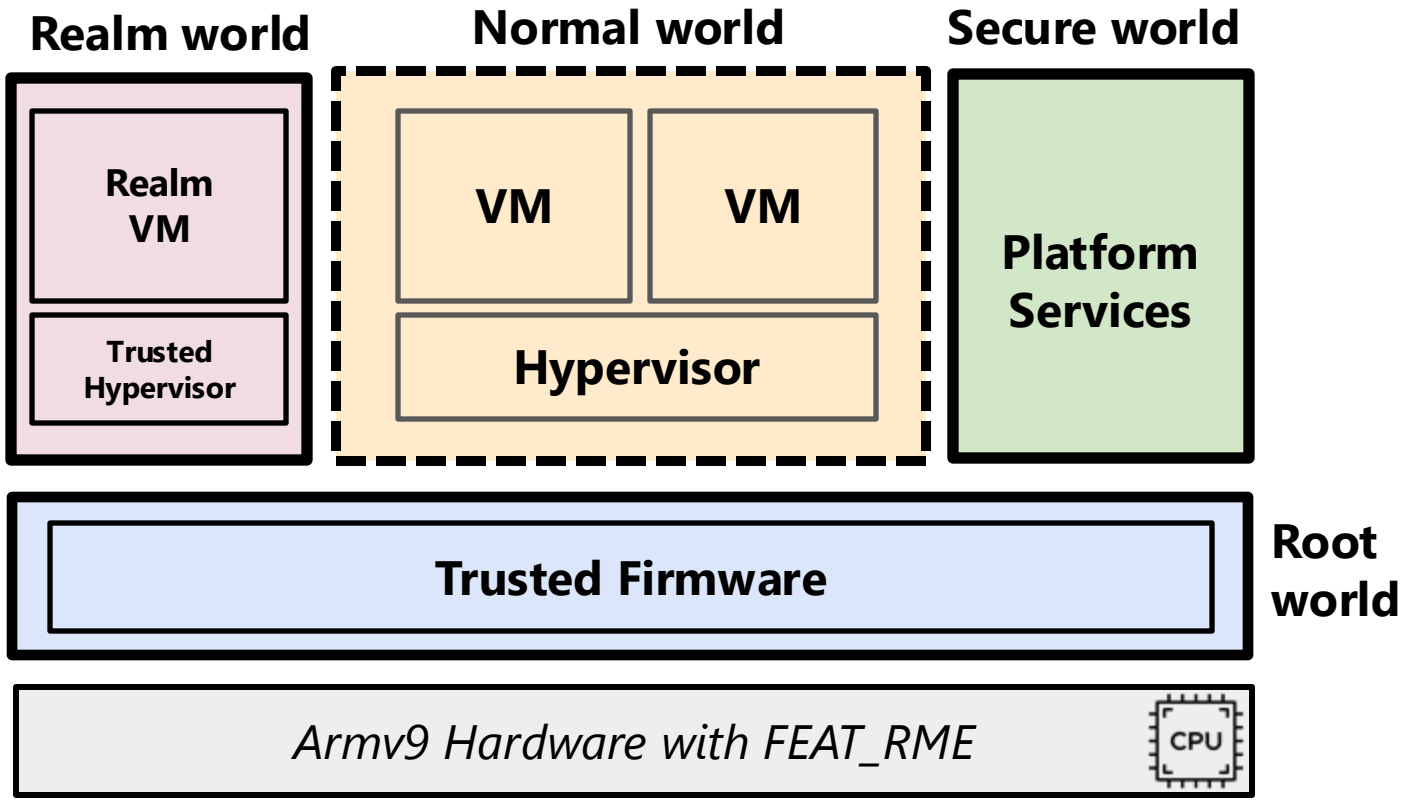
- Before Armv9: TrustZone



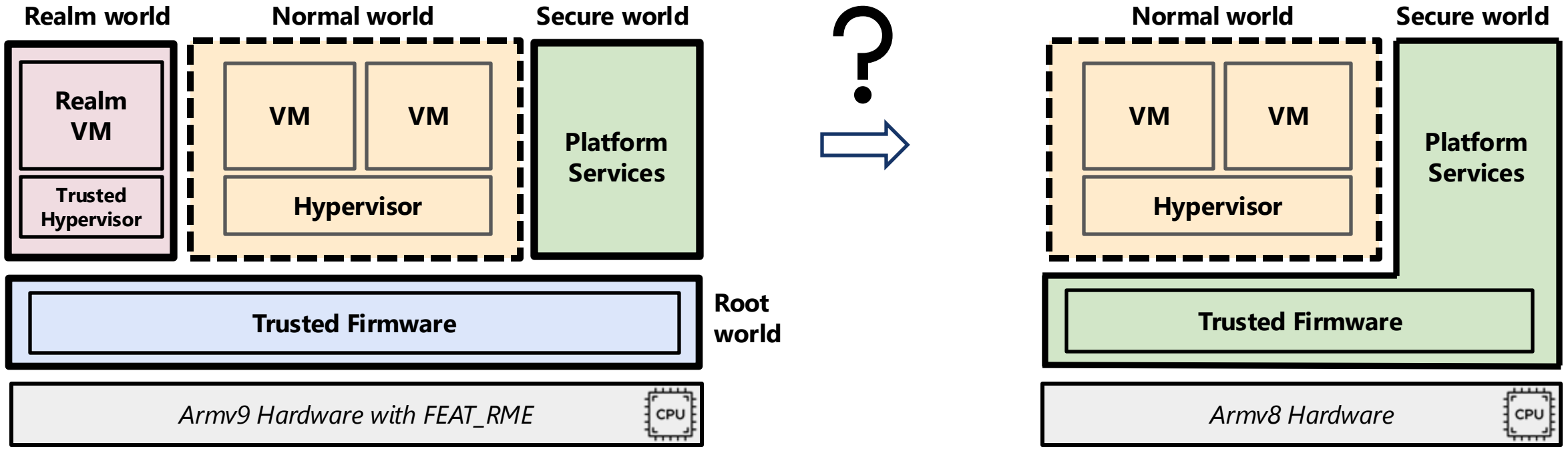
Background on Arm CCA



Background on Arm CCA



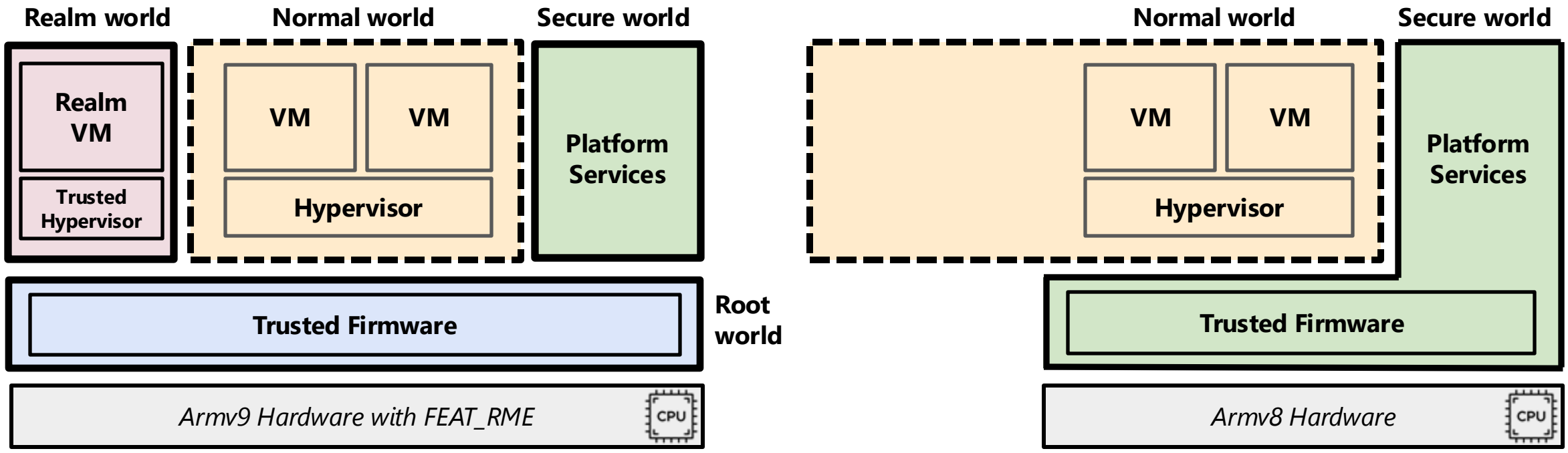
How do you even run Arm CCA on older Hardware?



Armv9

Armv8

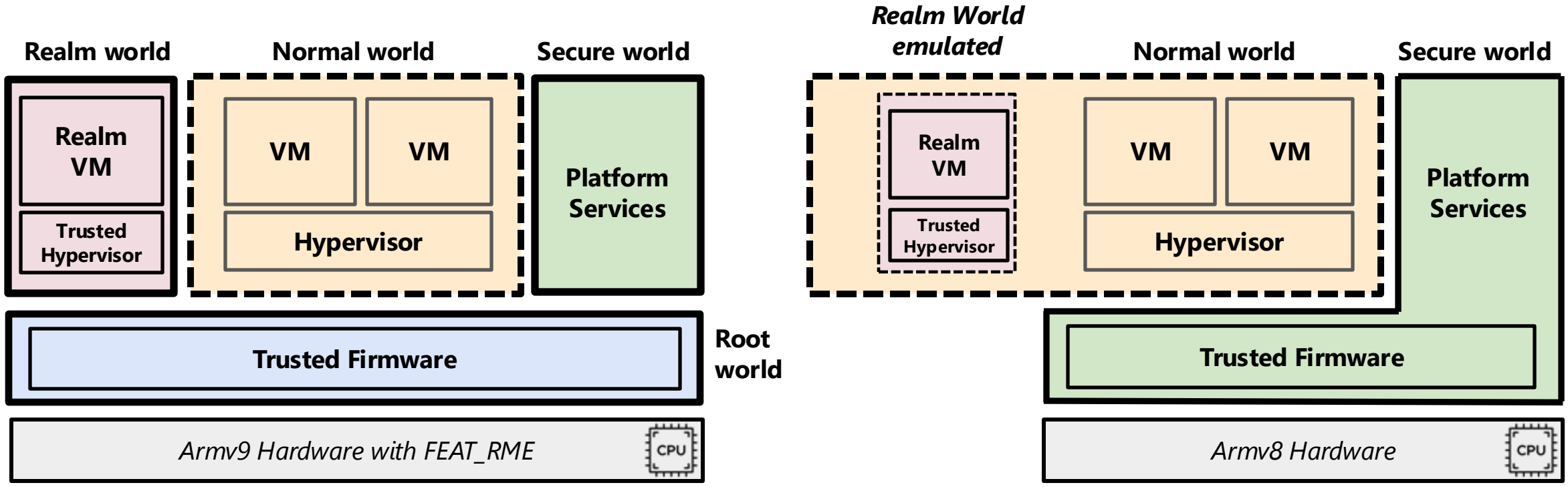
How do you even run Arm CCA on older Hardware?



Armv9

Armv8

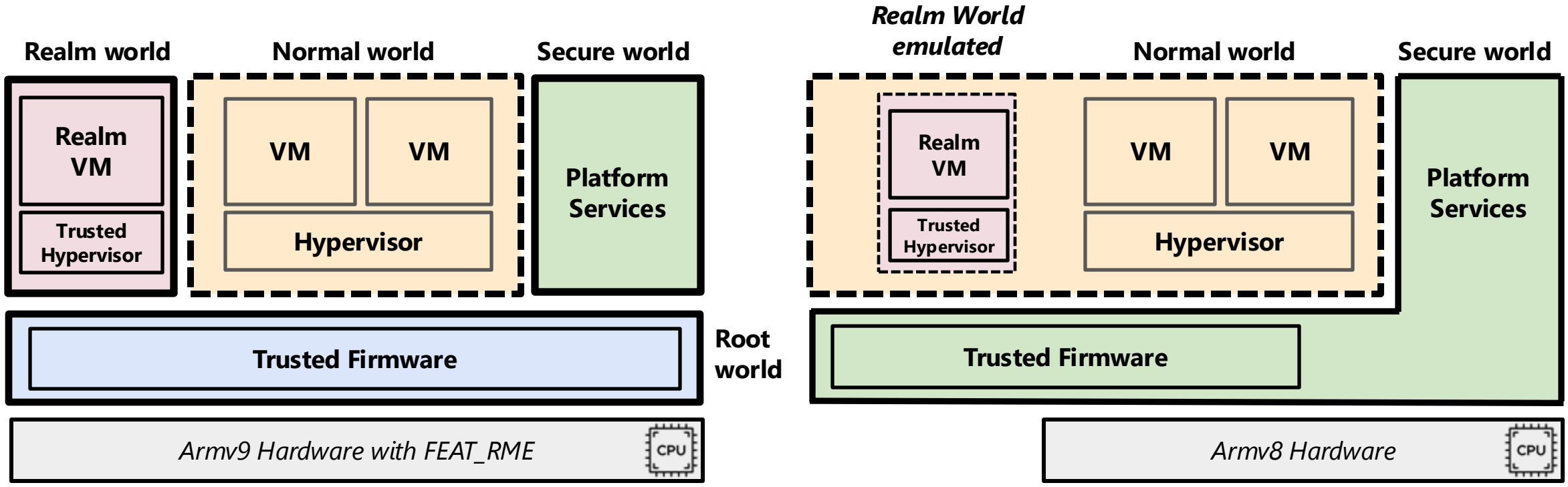
How do you even run Arm CCA on older Hardware?



Armv9

Armv8

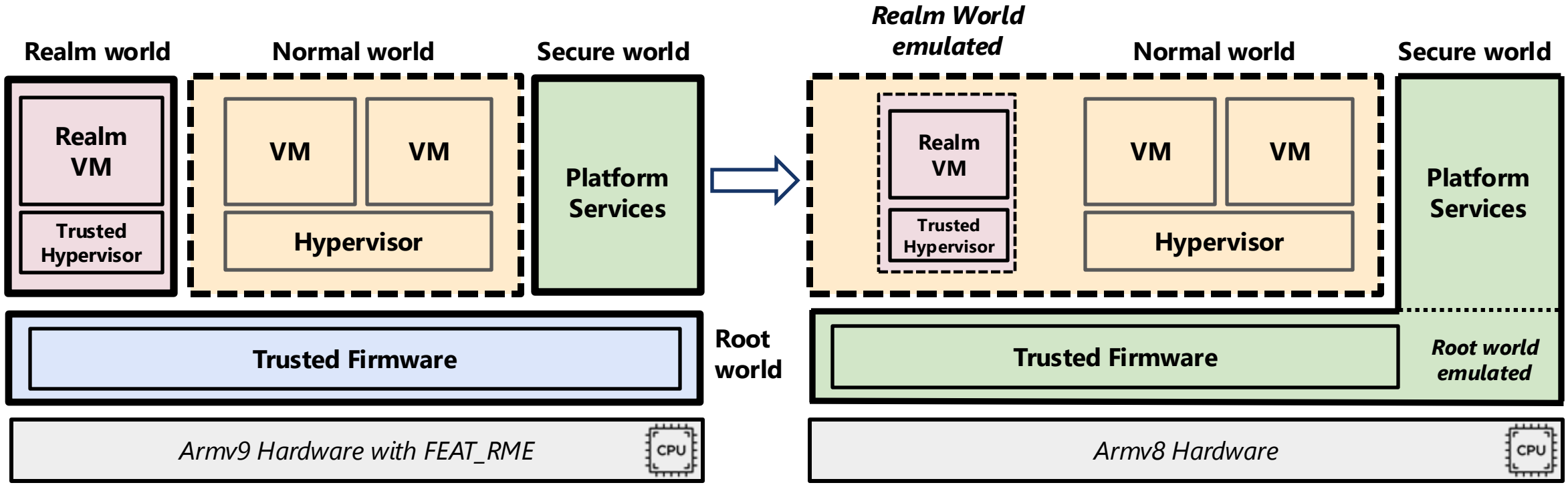
How do you even run Arm CCA on older Hardware?



Armv9

Armv8

How do you even run Arm CCA on older Hardware?

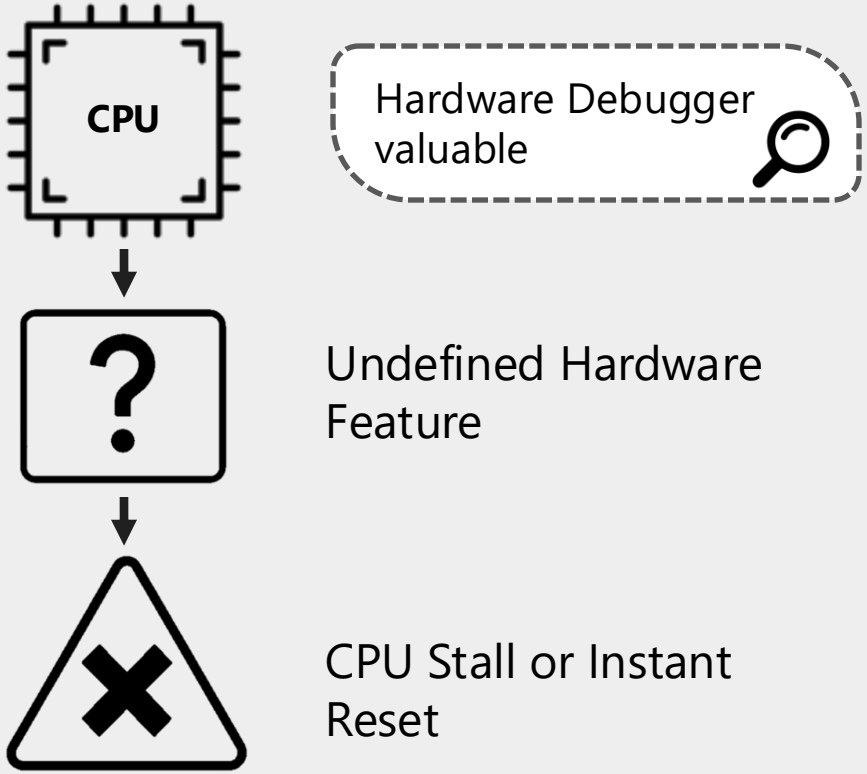


Emulate CCA in software within the constraints of Armv8 Hardware

Arm CCA on Armv8, what breaks?

- **Tradeoff** between **Compatibility** & **Overhead**
- **Fake missing parts in software** while keeping changes small
- Return **predefined values** instead of querying the hardware

Preserve functionality without security



Arm CCA on Armv8, what breaks?

```
#ifdef ENABLE_OPENCCA
   /* re-implement or disable
      for Arm v8.2 */
#else
   /* upstream impl. */
#endif
```

Preserve functionality without security

Disable or re-implement in software:

- FEAT_RME
- FEAT_RNG
- FEAT_S2FWB
- FEAT_TTST
- FEAT_ECV
- FEAT_CSV2
- FEAT_S2PIE
- FEAT_GCS
- FEAT_DIT
- FEAT_PAuth
- FEAT_ExS
- FEAT_AMUv1
- FEAT_SME
- FEAT_SVE
- FEAT_LPA2
- FEAT_HPMN0
- FEAT_MTE2

See code & paper

Arm CCA on Armv8, what breaks?

```
#ifdef ENABLE_OPENCCA
   /* re-implement or disable
      for Arm v8.2 */
#else
   /* upstream impl. */
#endif
```

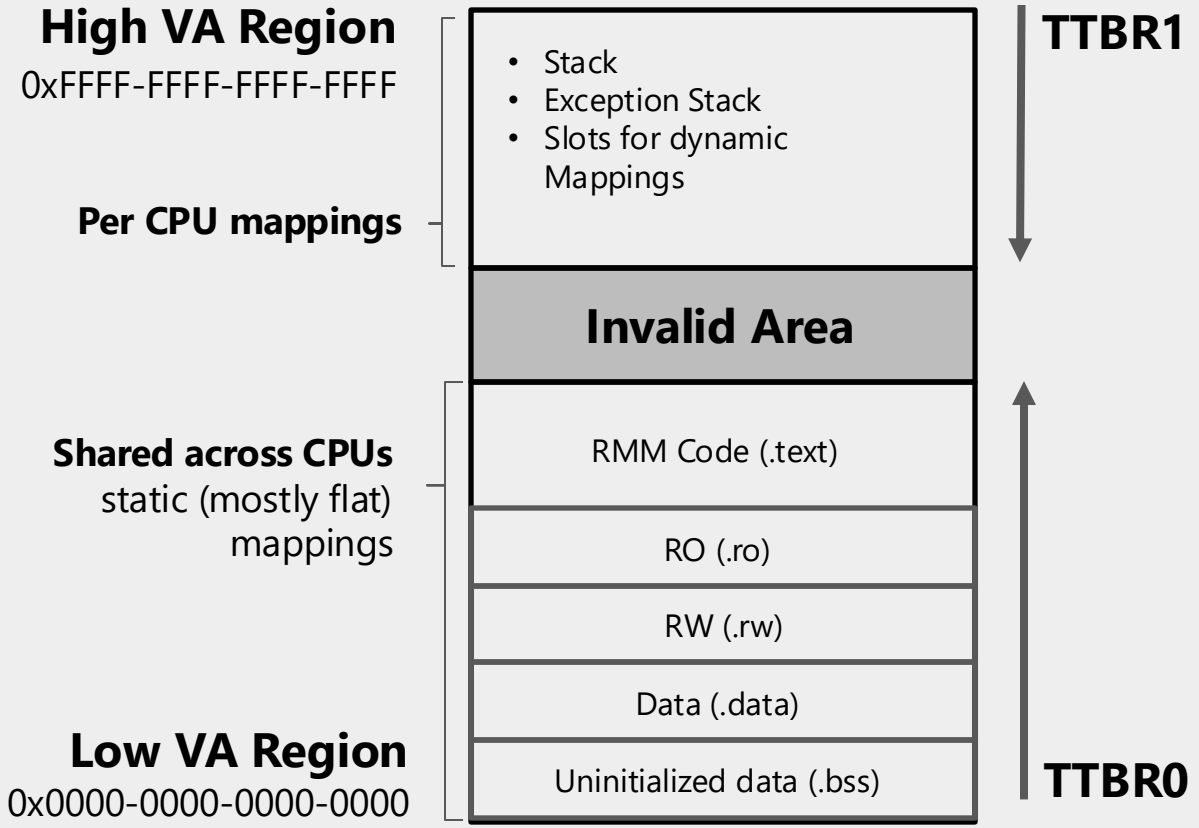
Preserve functionality without security

Disable or re-implement in software:

- FEAT_RME
- FEAT_RNG
- FEAT_S2FWB
- **FEAT_TTST**
- FEAT_ECV
- FEAT_CSV2
- FEAT_S2PIE
- FEAT_GCS
- FEAT_DIT
- FEAT_PAuth
- FEAT_ExS
- FEAT_AMUv1
- FEAT_SME
- FEAT_SVE
- FEAT_LPA2
- FEAT_HPMN0
- FEAT_MTE2

See code & paper

Example: Short Translation Tables (TTST)



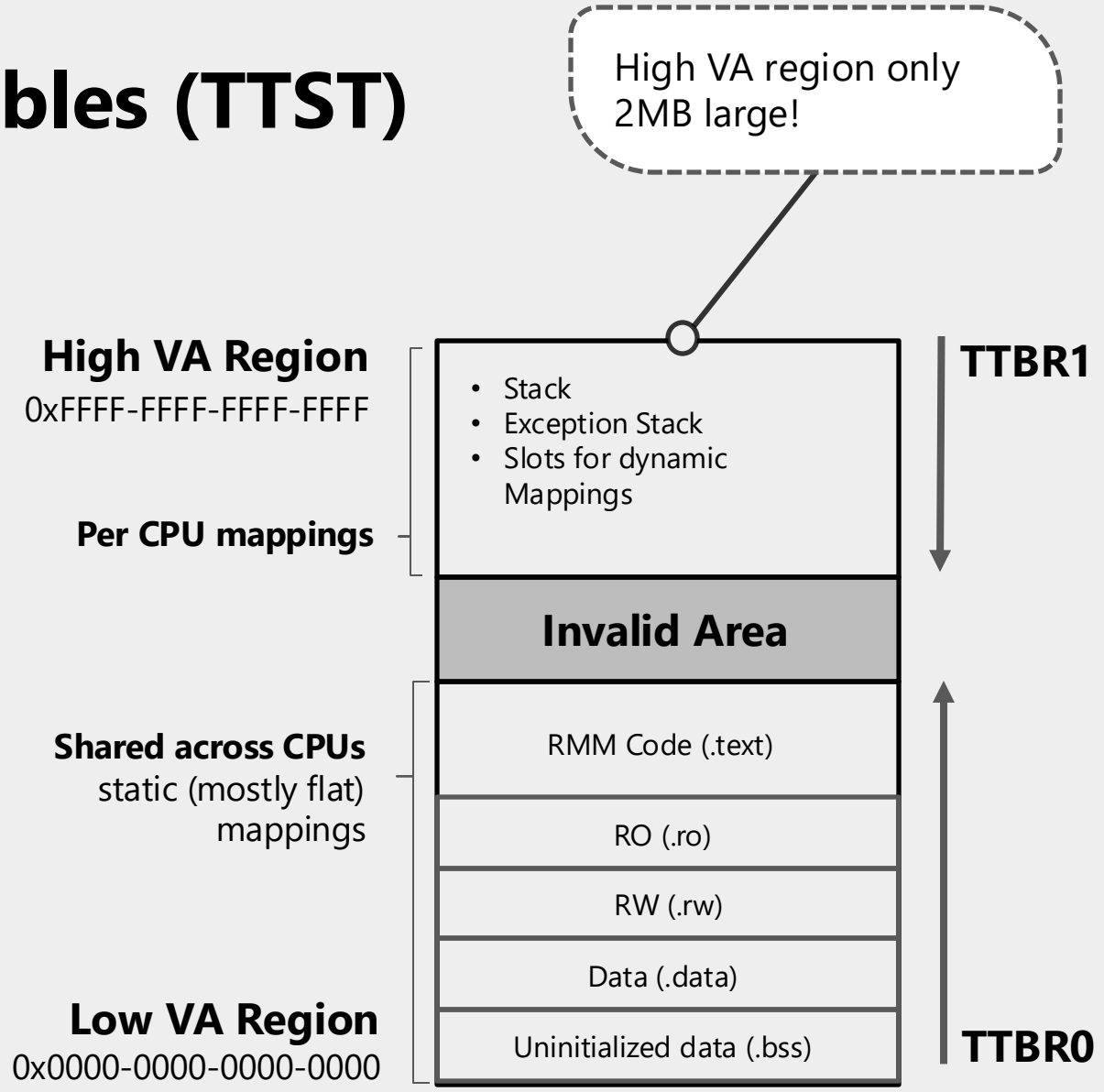
Example: Short Translation Tables (TTST)

Memory Layout in Trusted Hypervisor

- Paging implicitly uses FEAT_TTST (**Armv8.4**)
- Decrease limit on size of VA region
  - Fewer page walks needed (L3 -> Page)
- Not available on our hardware!

Workaround:

- Increase High VA region to 1GB
- Longer page table walk (L2 -> L3 -> Page)



Hardware:

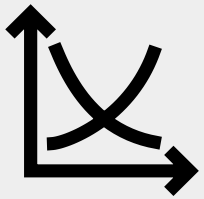
Key Specs: RK3588 SoC

- **Armv8.2** Architecture
- CPU: 4x **Cortex-A76** + 4x **Cortex A55**
- GPU: Arm Mali G610
- Up to 32 GB RAM
- I/O: PCIe 3.0, USB, HDMI

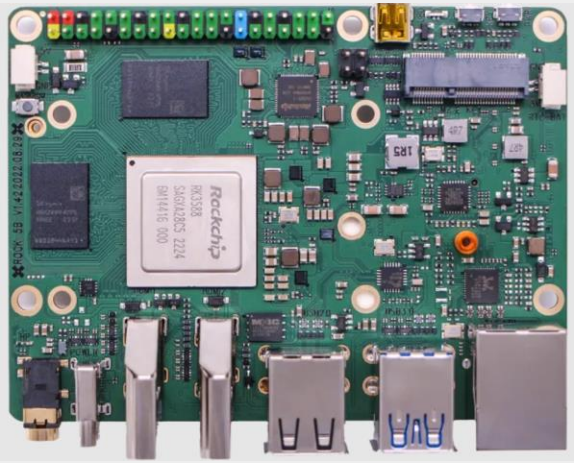
We looked into
~ 40 boards in 2025

No Vendor lock
Unlocked EL3

Documentation
Technical Reference Manual



Affordable +
Available + modern



Radxa Rock5b RK3588 ~ 250 USD

https://radxa.com/products/rock5/5b/

Current Status & Next Steps

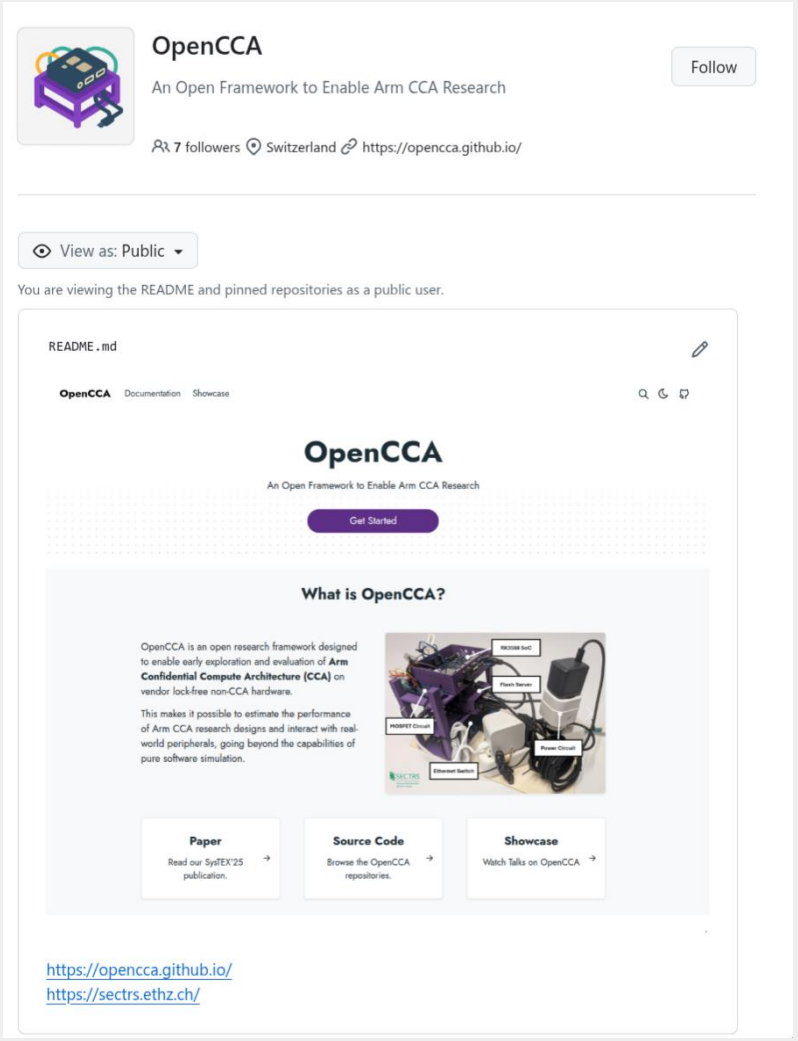
Current Status

- Run confidential VMs with Arm reference stack
  - **TF-A: v2.11**
  - **RMM: v0.5.0**
  - Linux 6.12 (cca/full-v5+v7)
  - Kvmtool (cca/v3)

OpenCCA: + ~2.5k LoC

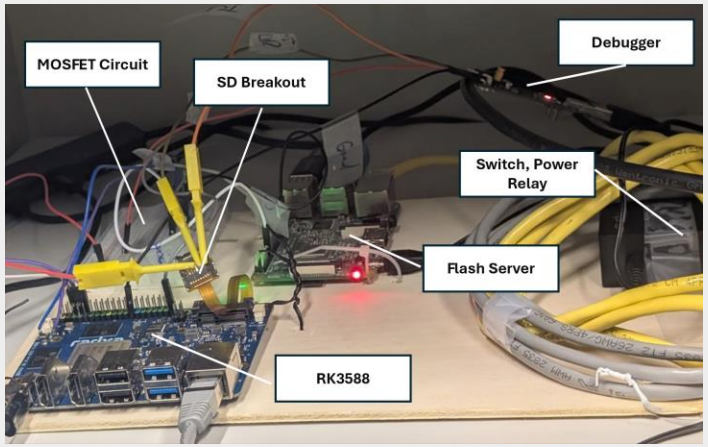
Next Steps:

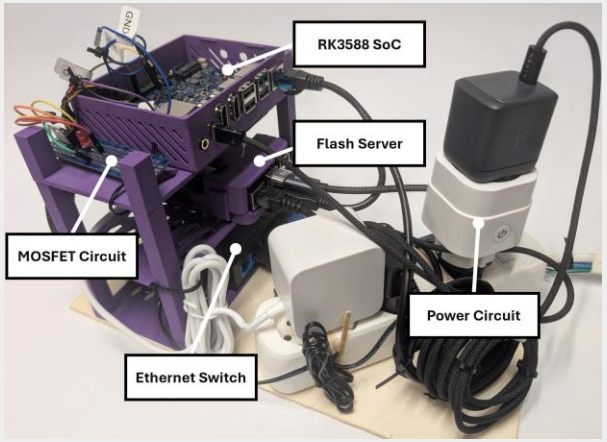
- Update to latest version of reference stack



https://opencca.github.io

OpenCCA on RK3588





- OpenCCA “Box” with support for **firmware flashing and power management**

What this demo shows:

1. Boot a **CVM on OpenCCA**
2. Attach **Mali GPU to CVM**
3. **X over VNC + OpenGL** on Mali

Demo Repo:

What this demo prototypes:

- **IRQ routing (VFIO inspired)**
- **Stage-2 MMIO** mappings for GPU Registers
- Run **GPU driver inside CVM.**

GPU MMIO remains hypervisor shared.

Demo shows systems prototyping workflow

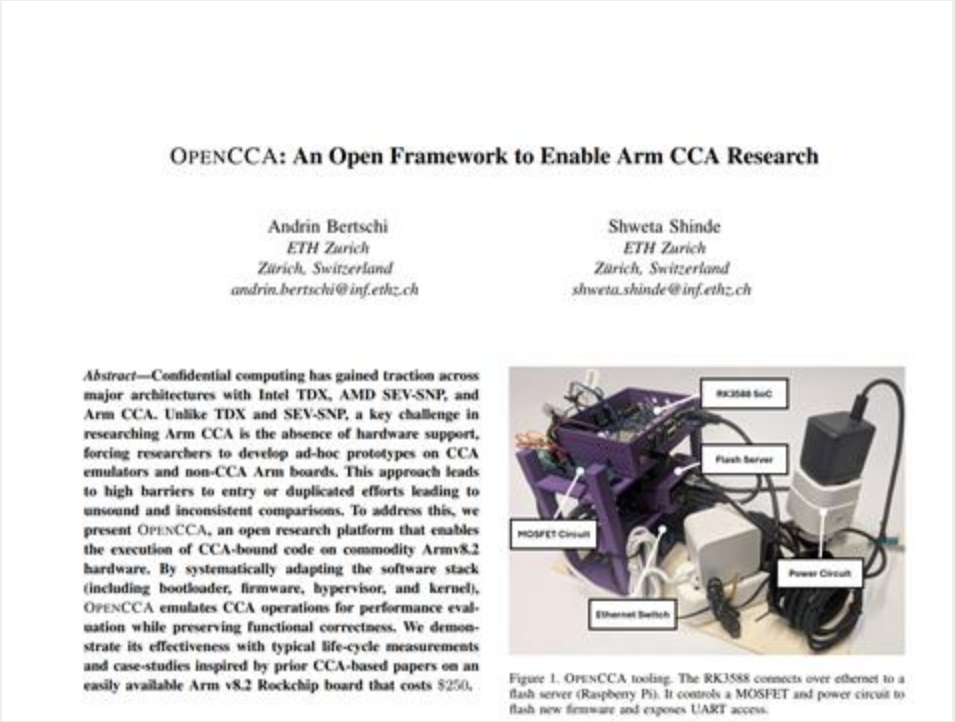
DEMO: Switch to UART now

Thank You

- Paper and source code is online
- Get in touch!

OpenCCA:

- Open *Framework* for Performance Estimations
- Enable CCA on commodity Armv8 hardware for performance and accelerators support



OPENCCA: An Open Framework to Enable Arm CCA Research

Andrin Bertschi
ETH Zurich
Zürich, Switzerland
andrin.bertschi@inf.ethz.ch

Shweta Shinde
ETH Zurich
Zürich, Switzerland
shweta.shinde@inf.ethz.ch

***Abstract*—Confidential computing has gained traction across major architectures with Intel TDX, AMD SEV-SNP, and Arm CCA. Unlike TDX and SEV-SNP, a key challenge in researching Arm CCA is the absence of hardware support, forcing researchers to develop ad-hoc prototypes on CCA emulators and non-CCA Arm boards. This approach leads to high barriers to entry or duplicated efforts leading to unsound and inconsistent comparisons. To address this, we present OPENCCA, an open research platform that enables the execution of CCA-bound code on commodity Armv8.2 hardware. By systematically adapting the software stack (including bootloader, firmware, hypervisor, and kernel), OPENCCA emulates CCA operations for performance evaluation while preserving functional correctness. We demonstrate its effectiveness with typical life-cycle measurements and case-studies inspired by prior CCA-based papers on an easily available Arm v8.2 Rockchip board that costs $250.**

Figure 1. OPENCCA tooling. The RK3588 connects over ethernet to a flash server (Raspberry Pi). It controls a MOSFET and power circuit to flash new firmware and exposes UART access.

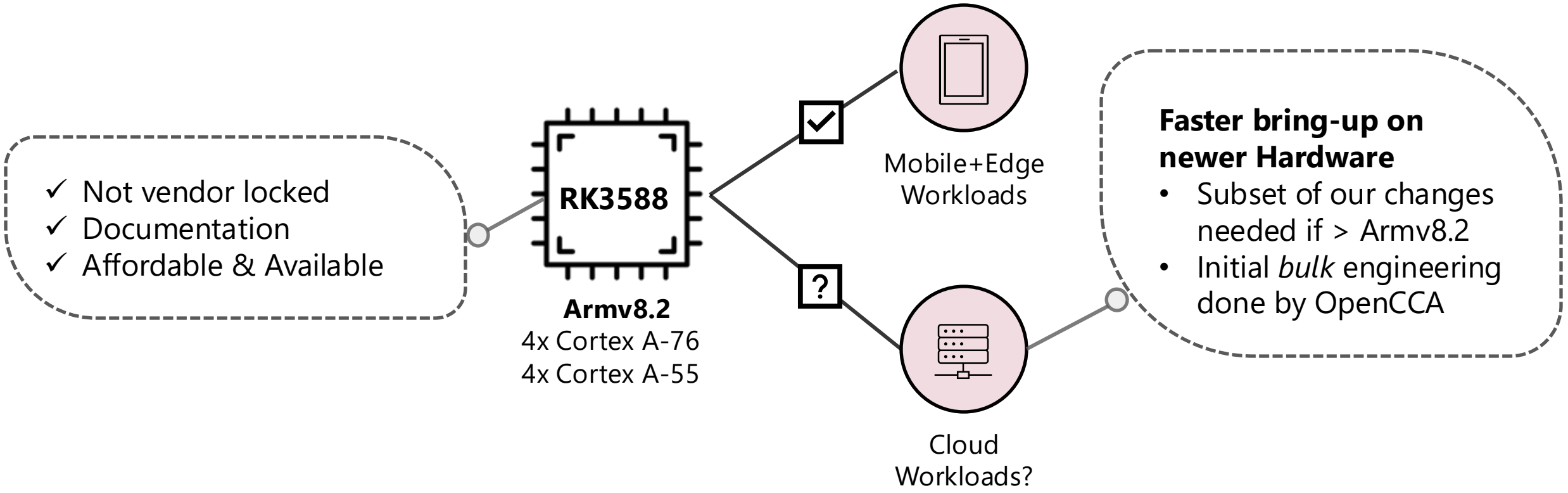
X: andrinbertschi

email: andrin.bertschi@inf.ethz.ch

web: https://opencca.github.io

Backup

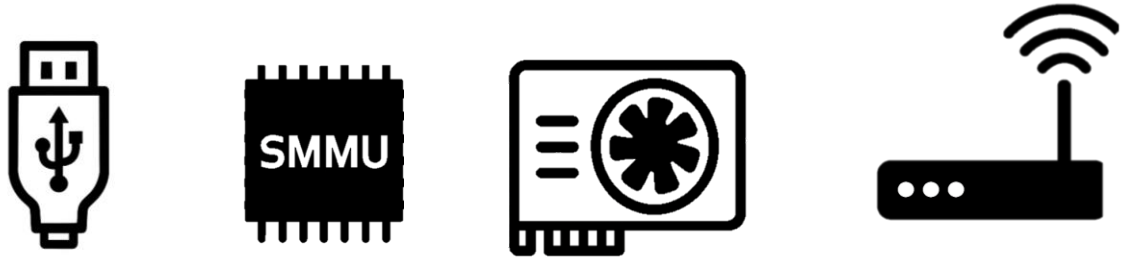
Suitable Workloads on RK3588 and Porting Work to other boards

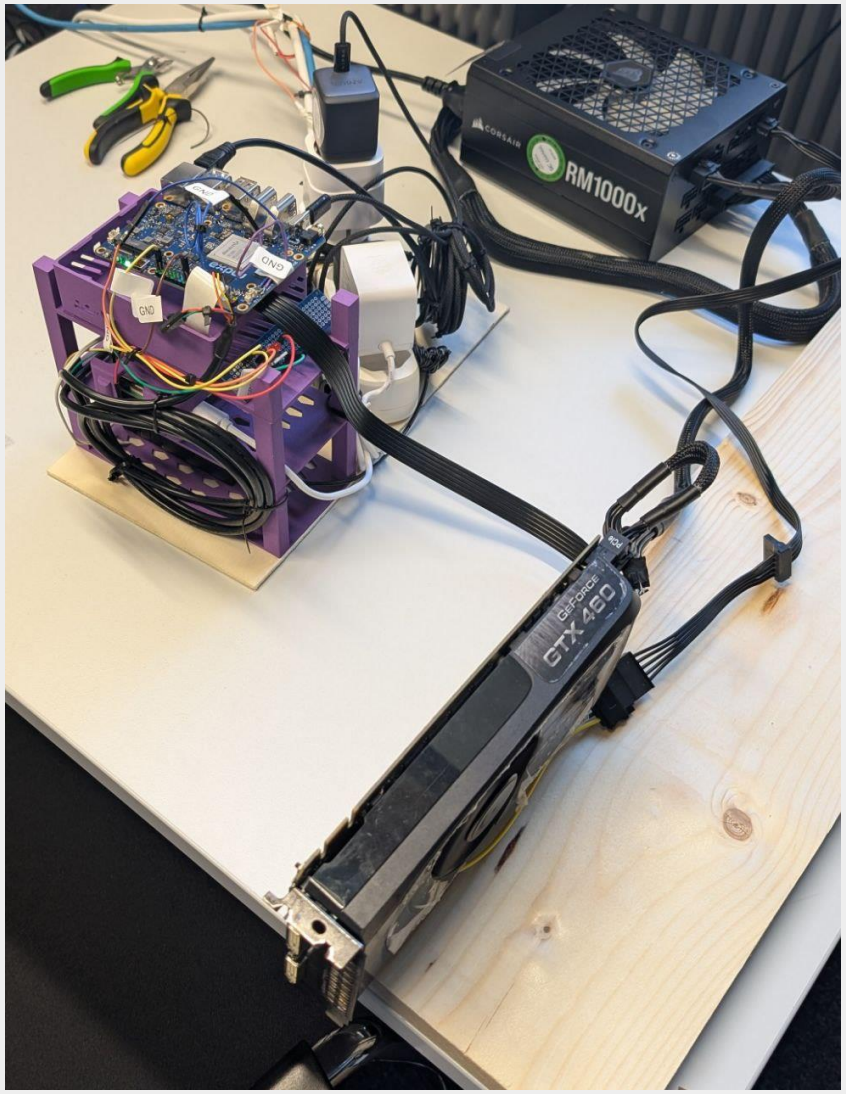


Real Hardware = Real Accelerators

OpenCCA runs on real hardware and can interact with real devices.

- RK3588 exposes PCIe lanes over NVMe slot
- Example: Research on Arm CCA with PCIe devices

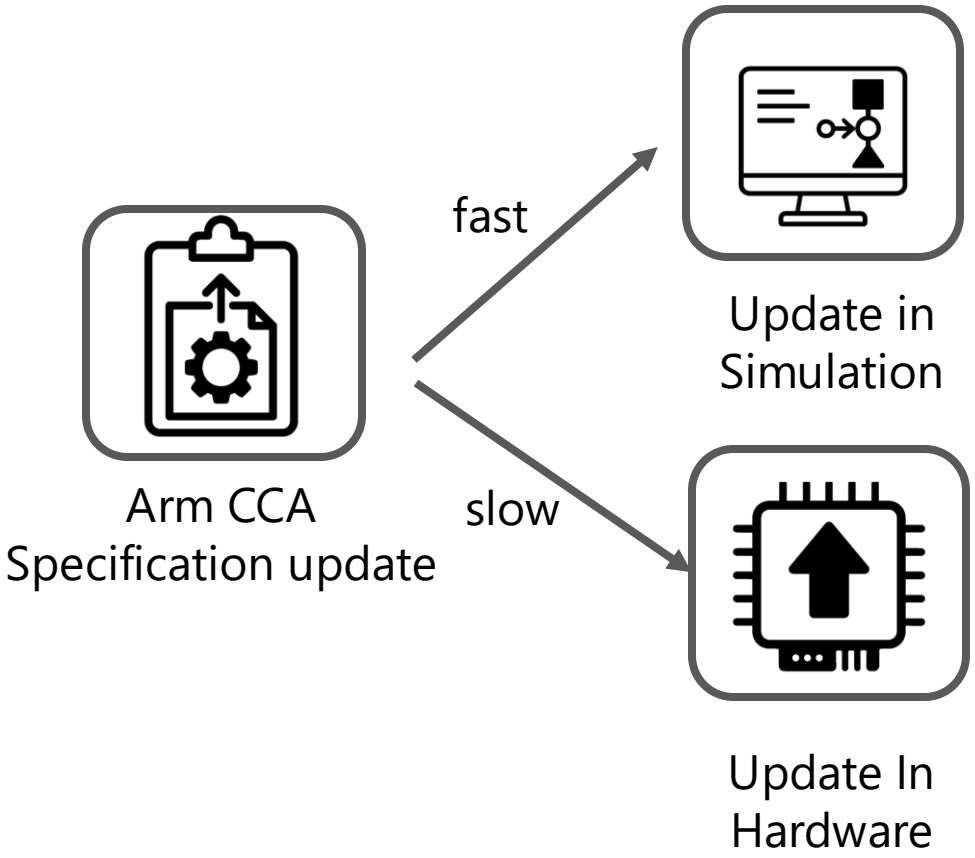




Connect Discrete GPU to OpenCCA

What about OpenCCA once we have CCA Hardware?

OpenCCA bridges Gap between Specification and Hardware Catch-Up for Performance Estimation



New FEAT_ features take time until implemented in HW

Estimate in Software with OpenCCA to experiment with firmware stack & estimate overheads

How we evaluate Designs with OpenCCA

Step 1:
Simulation

Arm FVP
QEMU

Validate design

Step 2:
Benchmark Baseline

For
Comparison

Unmodified OpenCCA stack

Step 3:
Benchmark design

OpenCCA
Framework

New research design

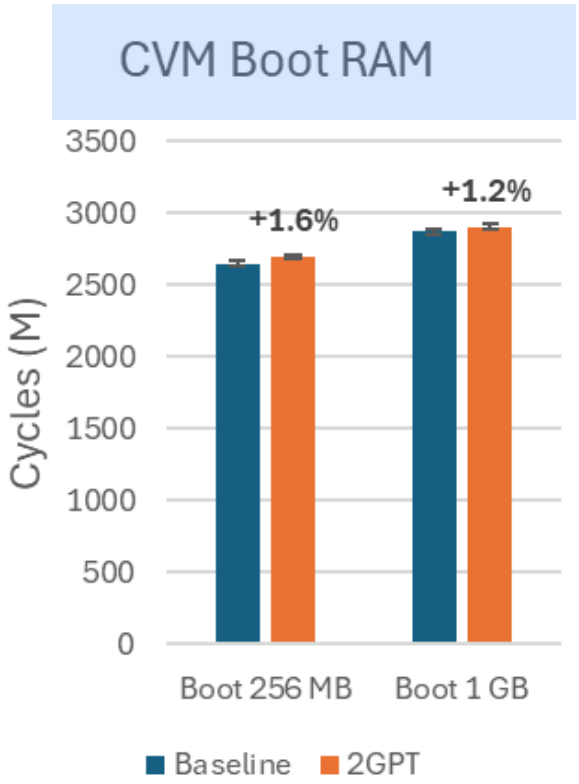
Step 4:
Analyze Results

Example Benchmark Campaign Case Study

See paper

TABLE 6. EVALUATION, RT: ROUND TRIP, DELEGATE: 4KB

Benchmark	Mean		Scale
	Instr	Cycles	
	OPENCCA		
CVM Boot 256 MB	1900	2647	1M
CVM Boot 1 GB	2015	2869	1M
RMI Delegate	2865	7988	1
RMI Version	994	3583	1
RMI RT	932	3370	1
SMC RT	182	421	1
	Two-GPT Case Study		
CVM Boot 256 MB	1928	2690	1M
CVM Boot 1 GB	2039	2902	1M
RMI Delegate	3488	8654	1



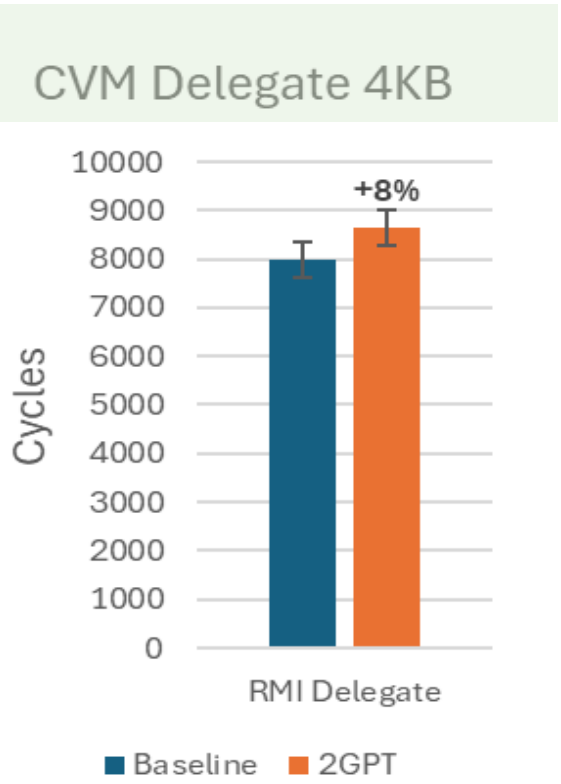


TABLE 6. EVALUATION, RT: ROUND TRIP, DELEGATE: 4KB

See paper

Benchmark	Mean		Stdev		Scale
	Instr	Cycles	Instr	Cycles	
OPENCCA					
CVM Boot 256 MB	1900	2647	6	15	1M
CVM Boot 1 GB	2015	2869	8	18	1M
RMI Delegate	2865	7988	187	365	1
RMI Version	994	3583	120	222	1
RMI RT	932	3370	115	209	1
SMC RT	182	421	44	68	1
Two-GPT Case Study					
CVM Boot 256 MB	1928	2690	9	10	1M
CVM Boot 1 GB	2039	2902	7	18	1M
RMI Delegate	3488	8654	182	372	1