ETH zürich

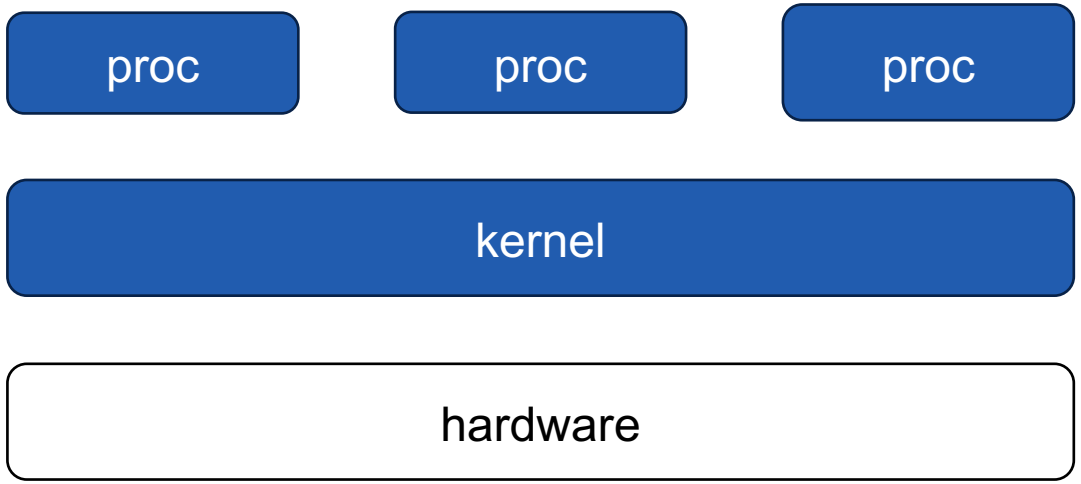
D INFK

Specifying the *de facto* OS of a production SoC

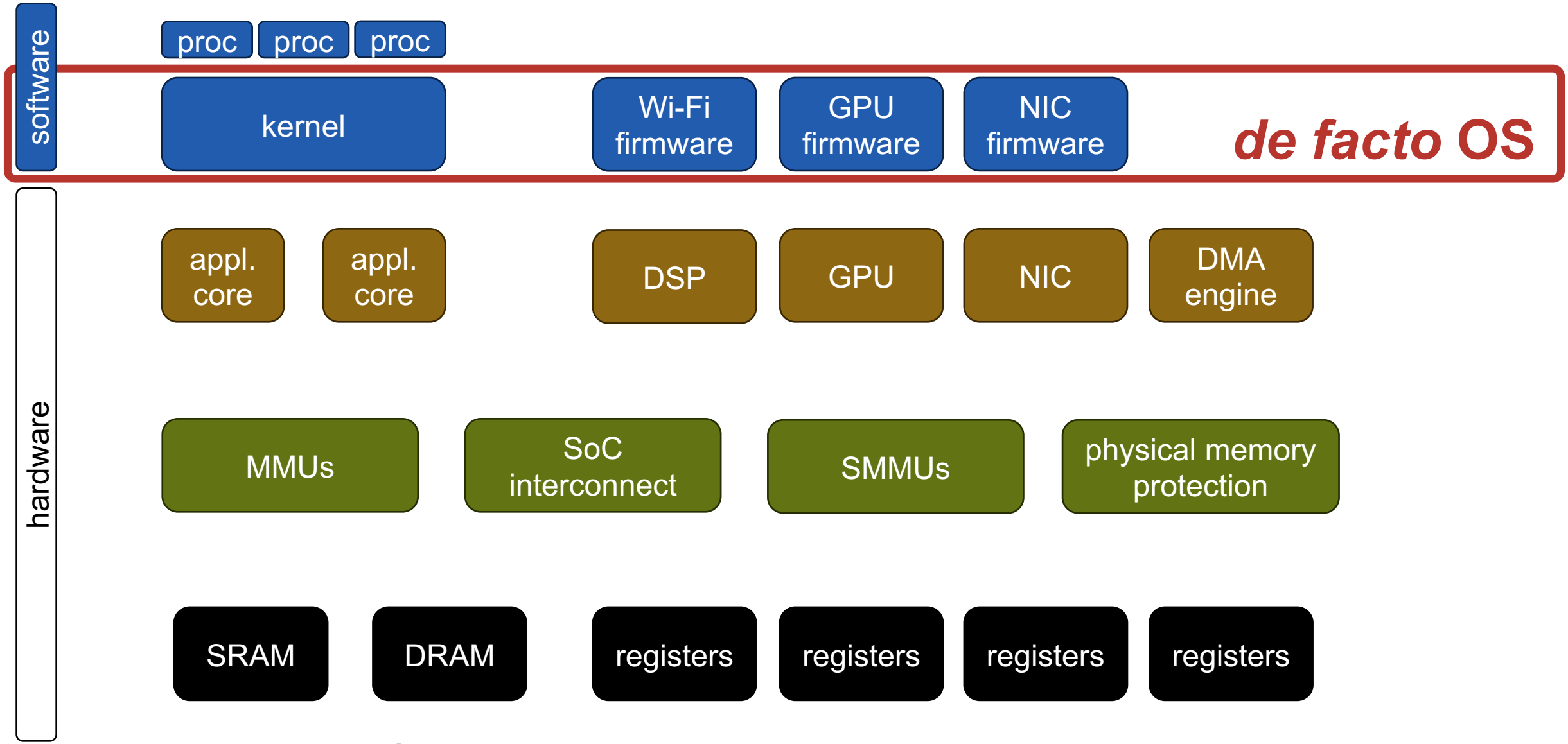
Ben Fiedler, Roman Meier, Jasmin Schult,
Daniel Schwyn, Timothy Roscoe

supported by Google arm

OS isolation on modern SoCs

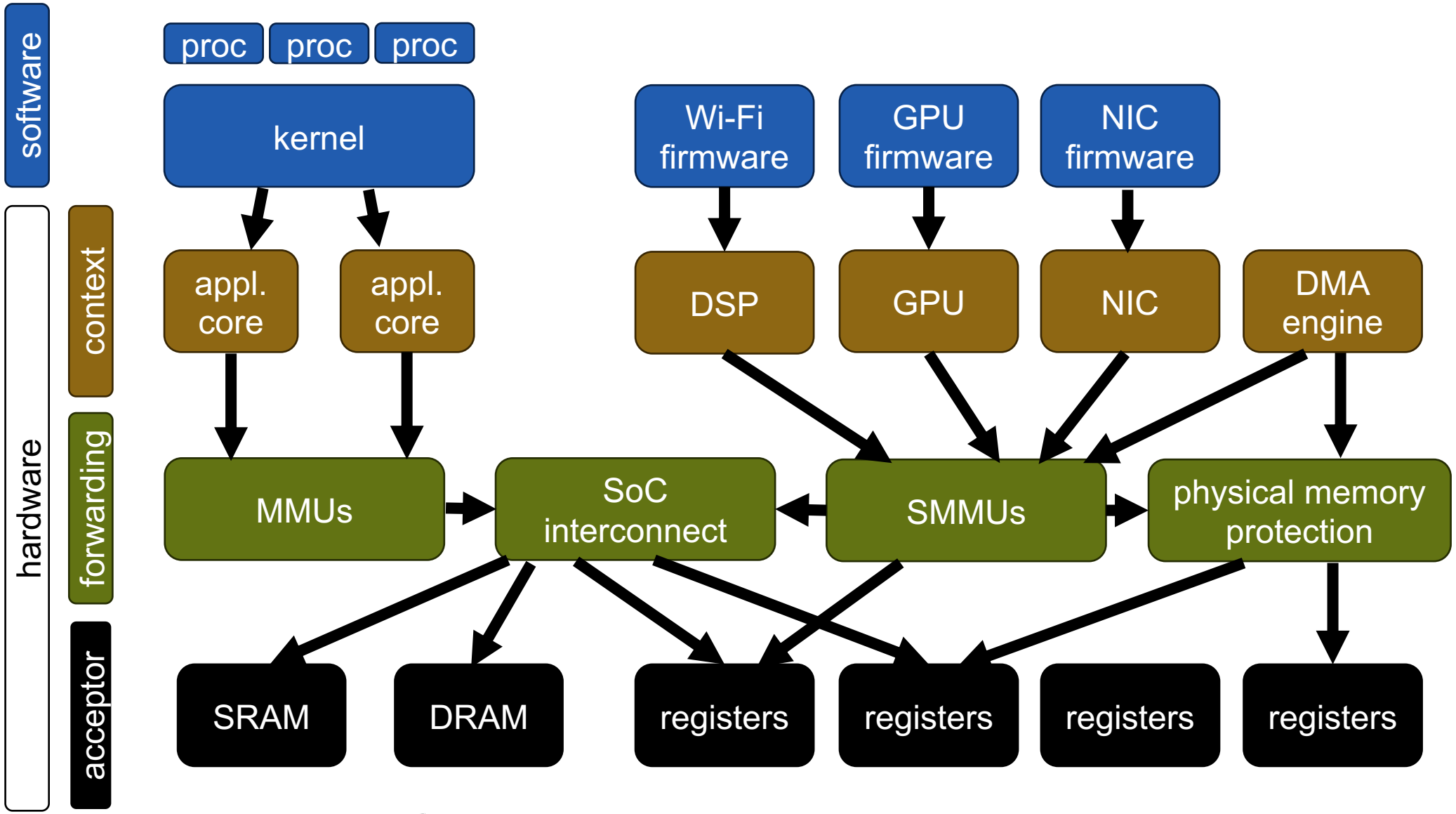


OS isolation on modern SoCs

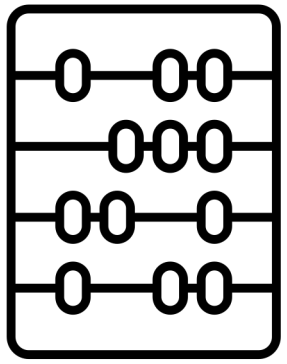


How do we make sense of this?

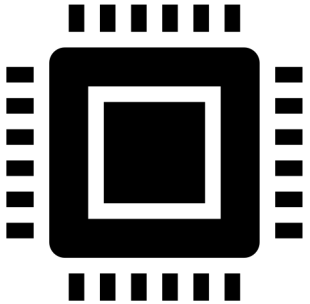
Memory operations are software interactions



Let's try a principled approach



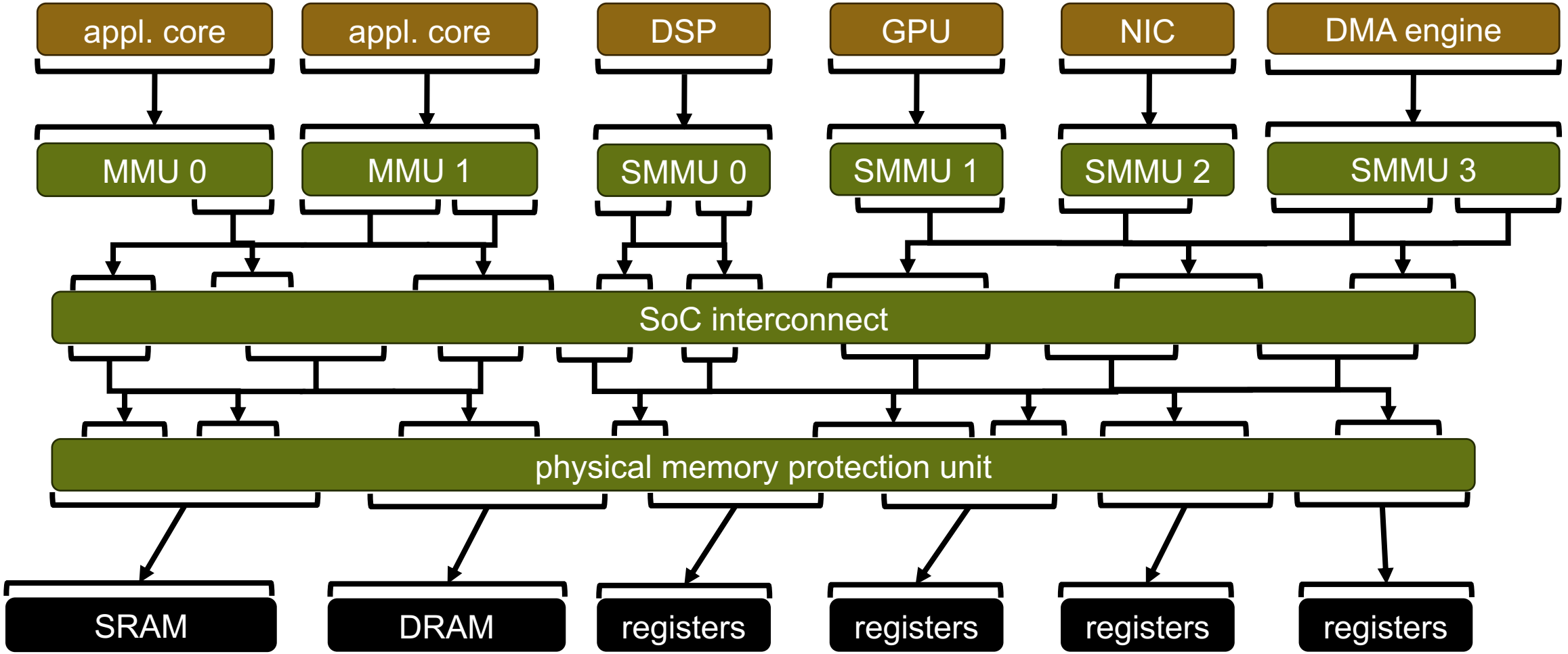
Formal model for hardware memory addressing



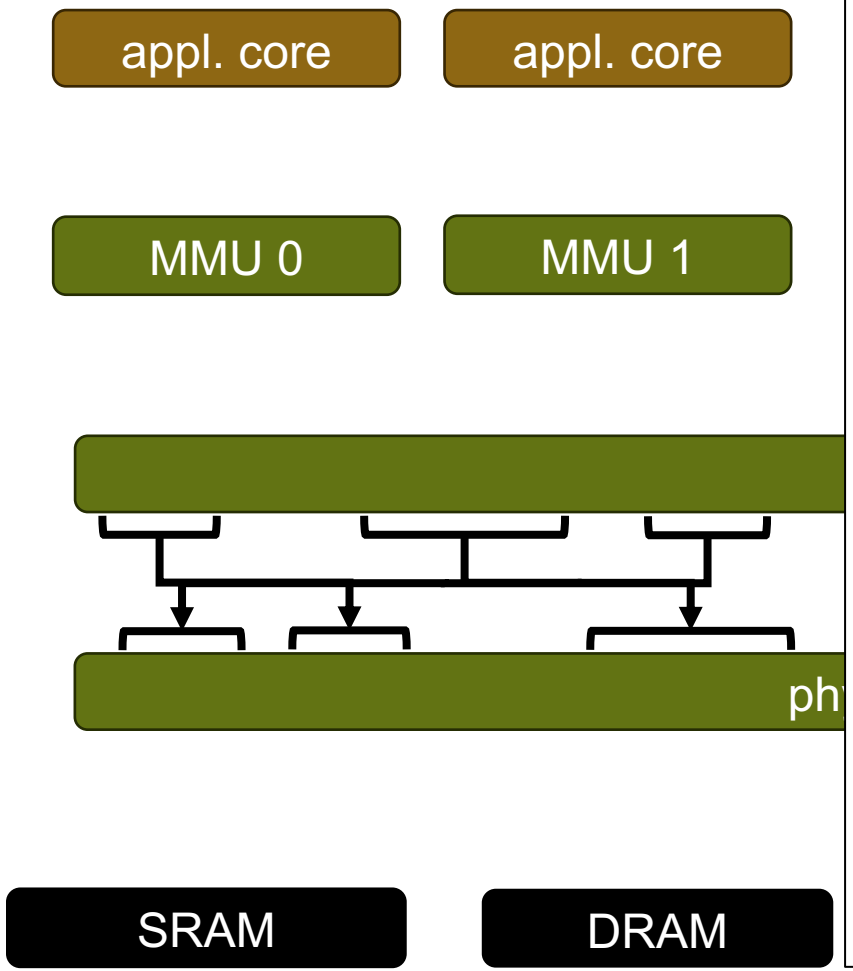
Write down an **existing System-on-Chip**

Compute and **refine** the trust graph of a de facto OS
Learn new principles for hardware and OS design

Decoding nets model the underlying topology

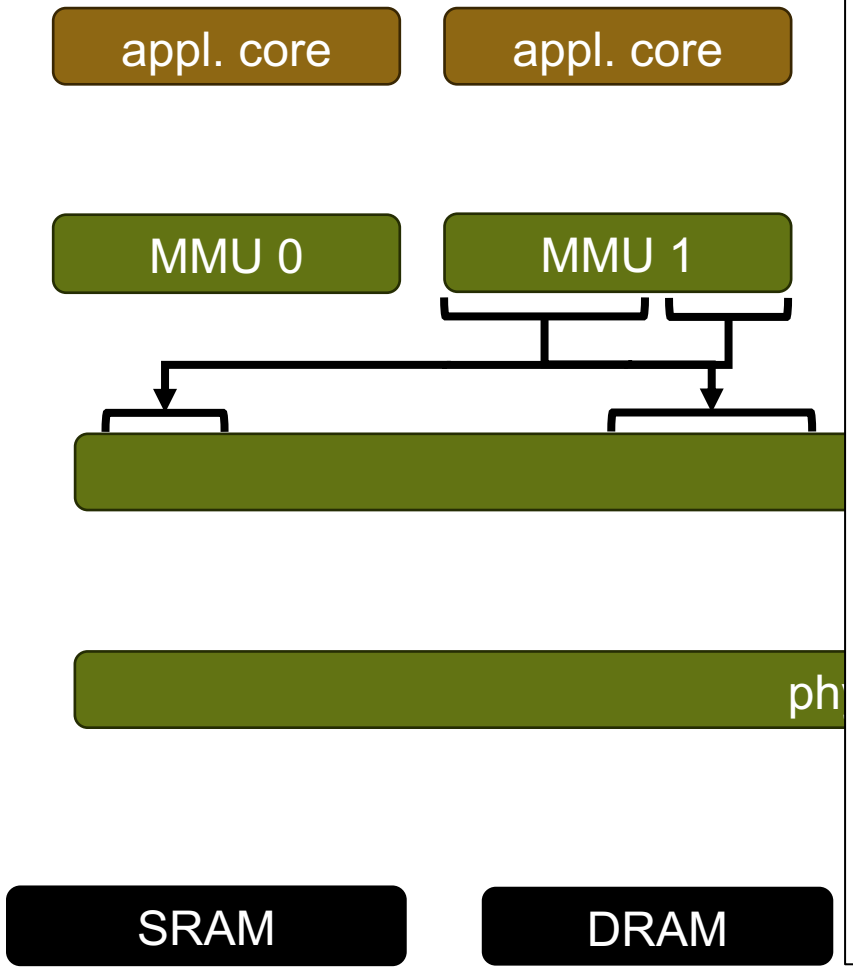


Writing down decoding nets in Sockeye3



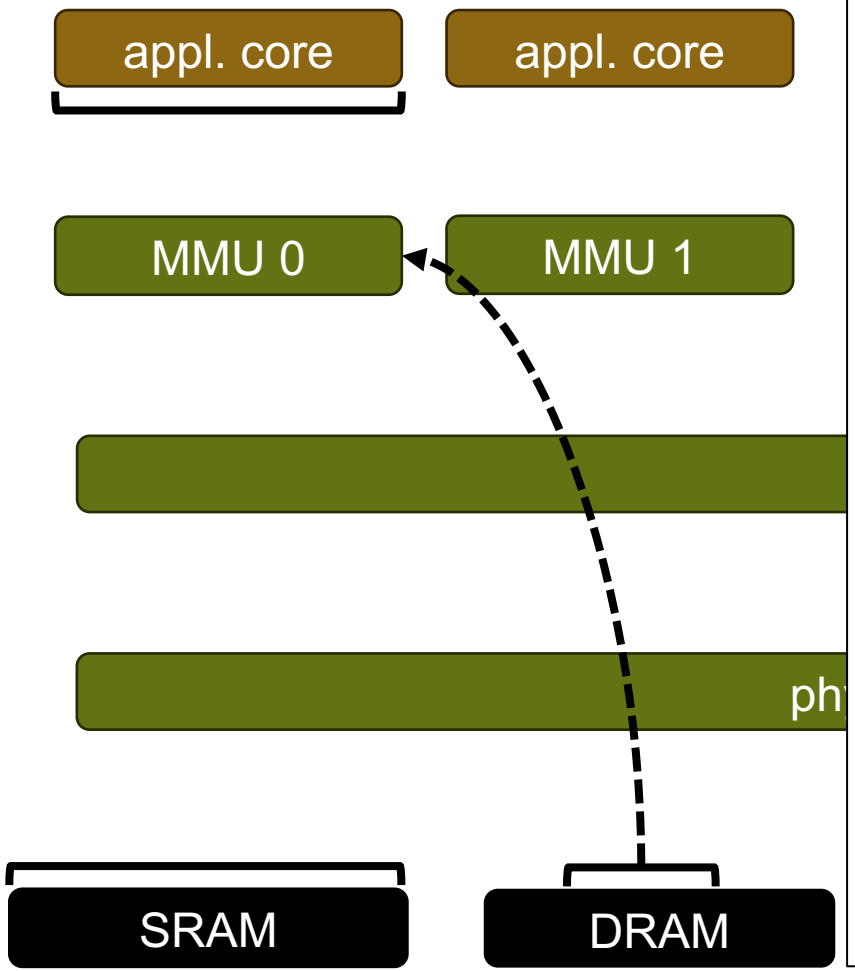

```
// Sockeye3: LISA+/Gem5/device trees-like language for
// specifying SoCs

let ic = add_address_space(0, 2^48);
let pmp = add_address_space(0, 16 * SIZE_GB);

add_mapping(ic, 0x8000_0000, pmp, 16 * SIZE_GB);
add_mapping(ic, 0x7430_0000, pmp, 2 * SIZE_MB);
add_mapping(...);
```

Writing down decoding nets in Sockeye3



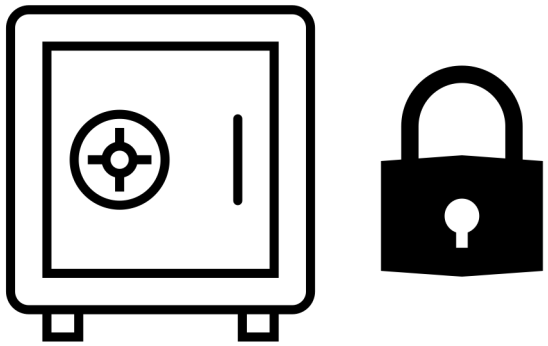

```
// Components are specified once an can be reused

let core0 = Arm::CortexA53::new();
let pt_proc0 = PageTable::new(/* ... */);
let mmu = Arm::v8::MMU::new(ic);
let proc0 = mmu.create_virtual_as(core0);
```

Writing down decoding nets in Sockeye3



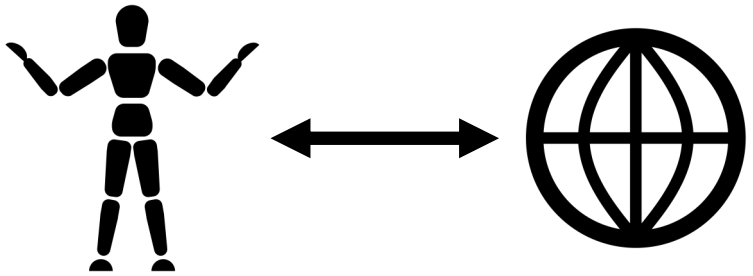

```
// Semantic annotations for underlying analysis

set_accept(dram); // terminates address resolution
set_context(core0); // is a context that issues
                    // memory operations


// within the MMU module: encode reconfiguration
// possibilities
let page_tables = new_region(dram, ptbase, 2 * SIZE_MB);
set_translate(mmu0, proc0, page_tables);
```

Challenges when defining SoC semantics



Be as precise as possible, however some components **cannot** be analyzed (sealed firmware, no documentation, ...)

→ **overapproximate** safely

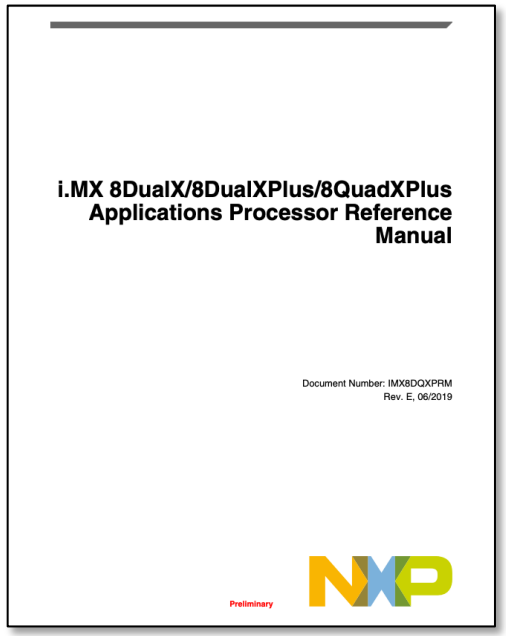


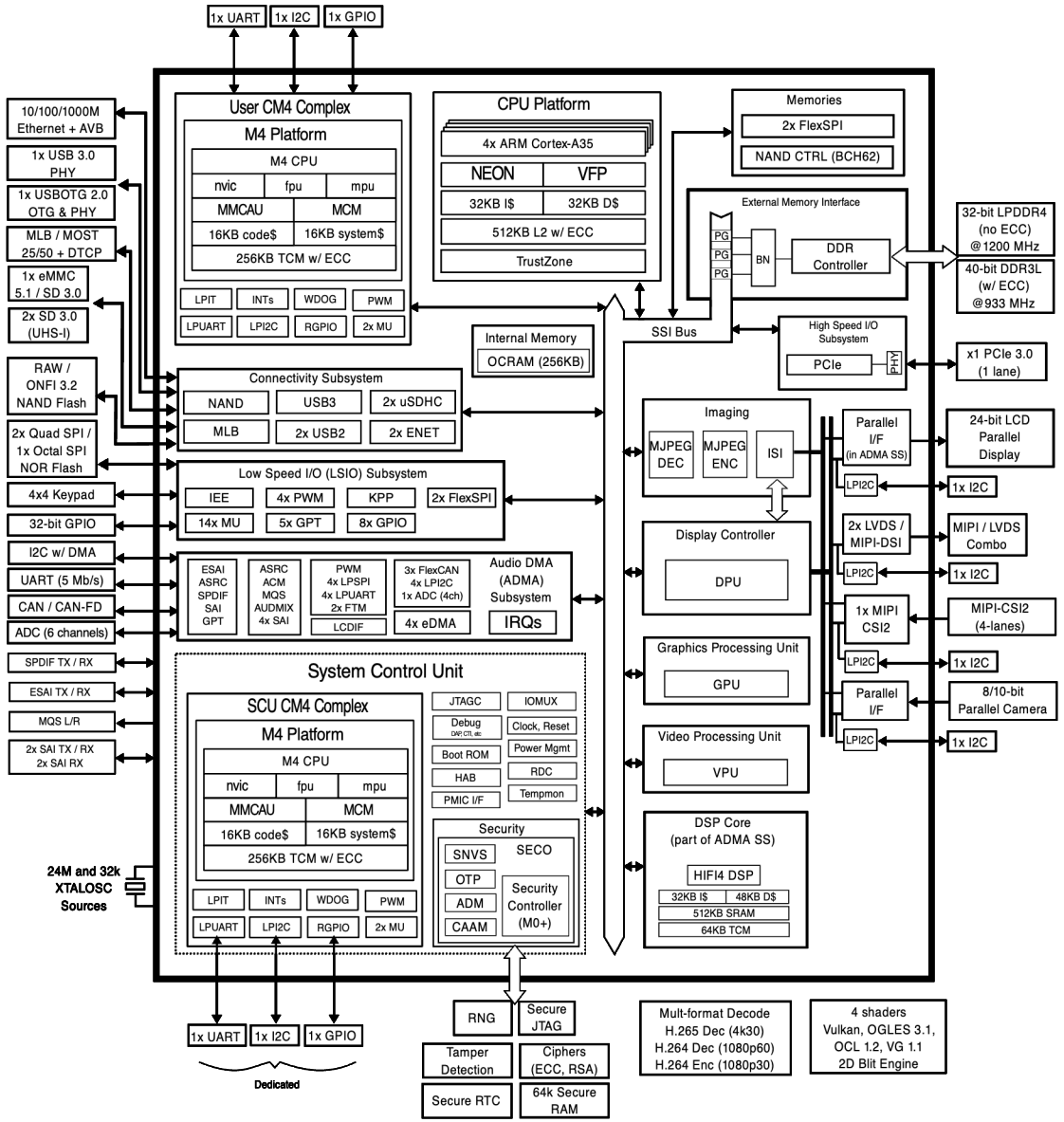
How to ensure that model is **accurate** with respect to the **real world**?

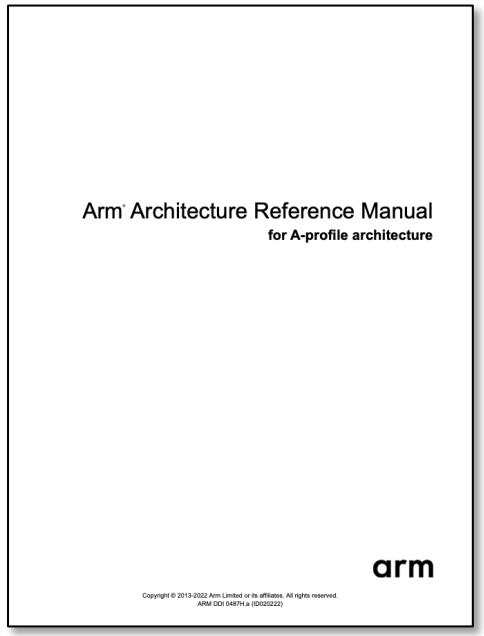
→ generate **litmus tests**?

What’s in a modern SoC?

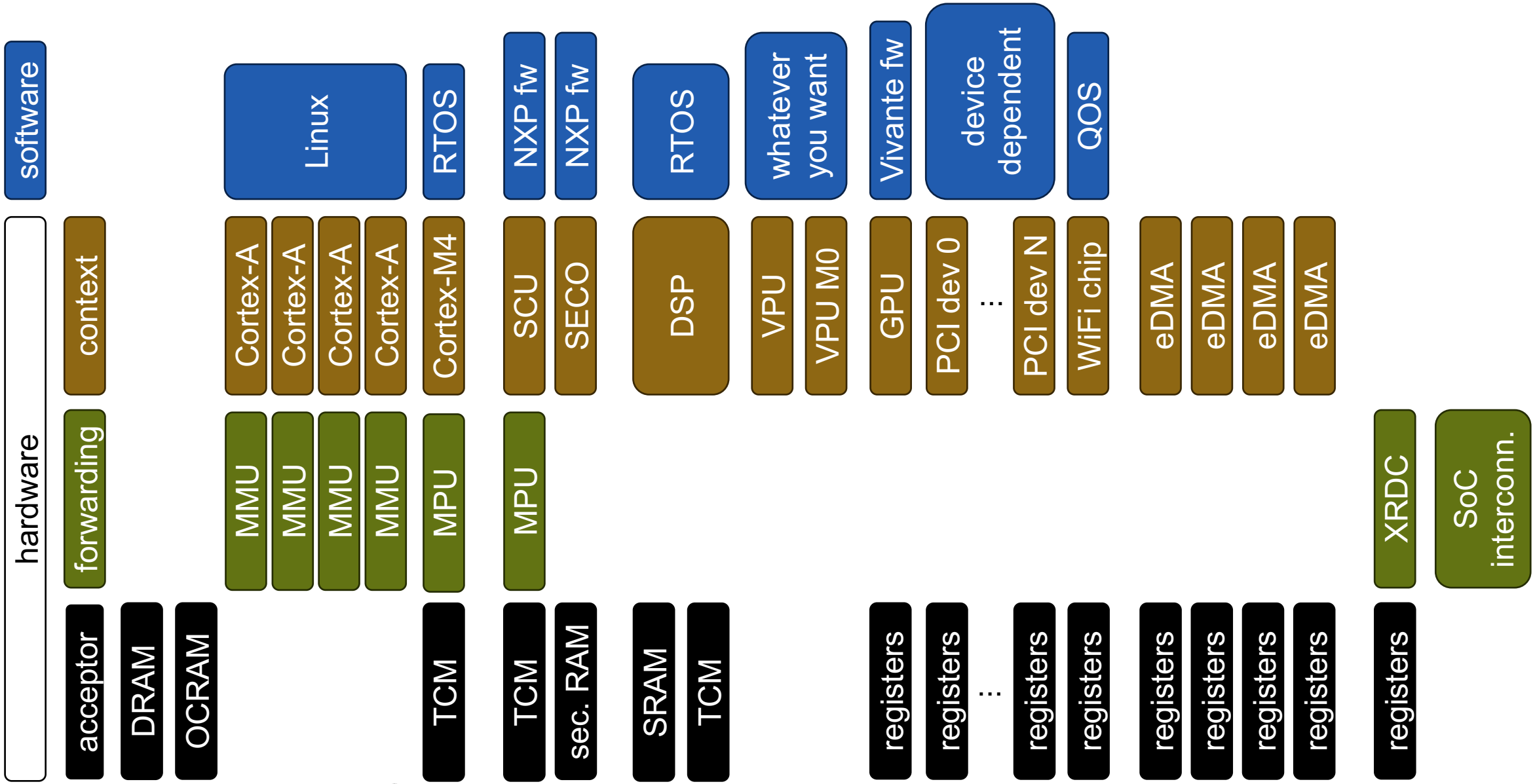
The decade-old i.MX8 SoC





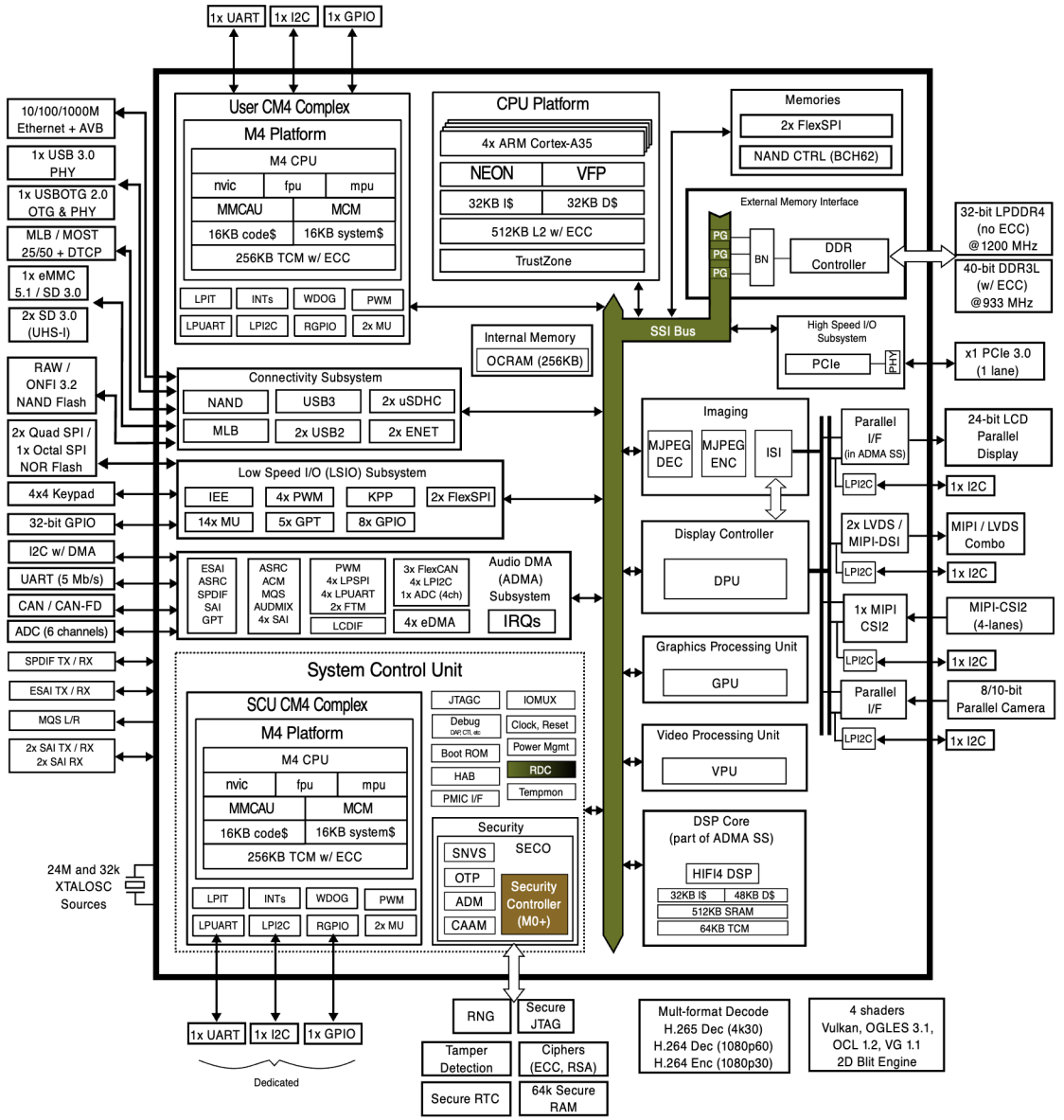


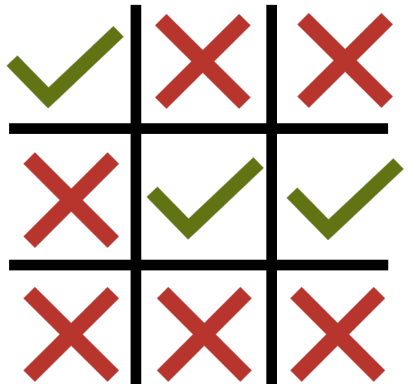
A brief and incomplete overview of the i.MX8



The Extended Resource Domain Controller (XRDC)

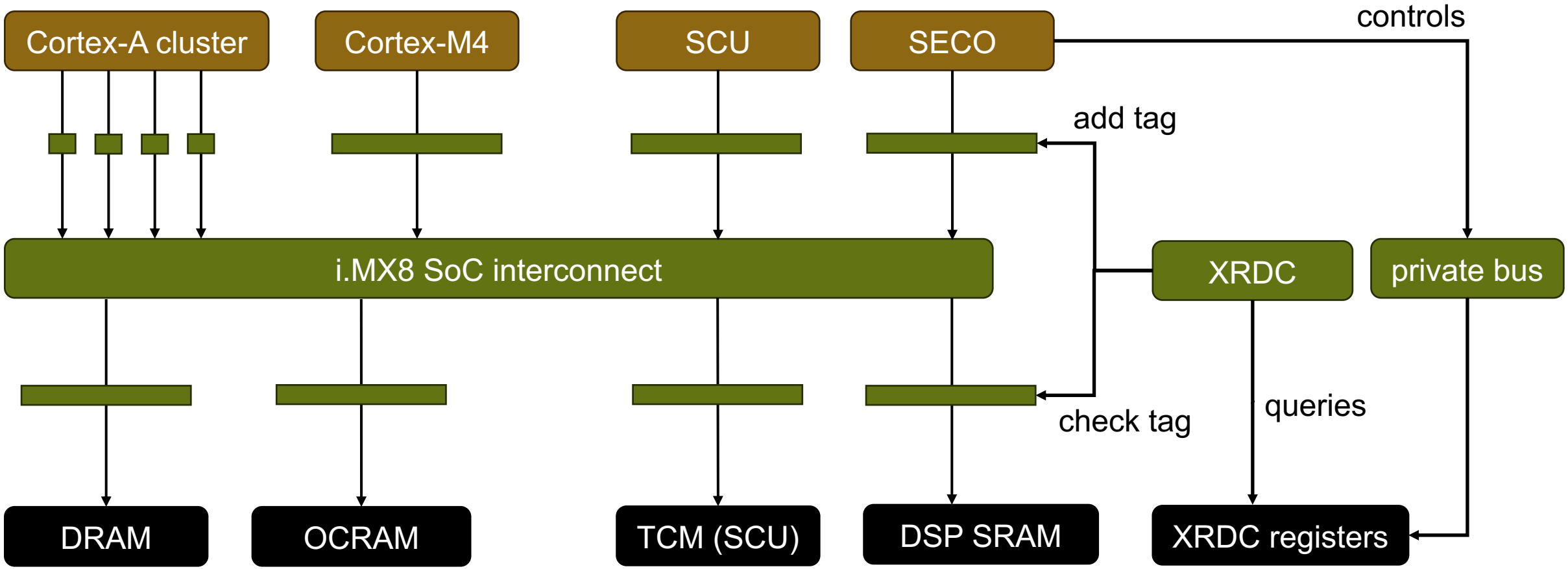
i.MX8-specific hardware protection mechanism, programmed by SECO



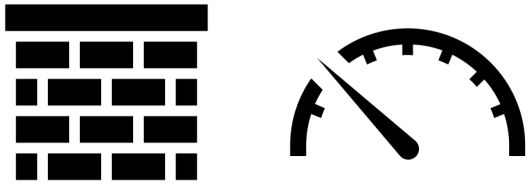


Partition-based isolation scheme

The Extended Resource Domain Controller (XRDC)



The Extended Resource Domain Controller (XRDC)

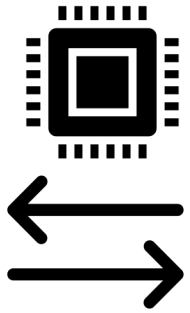


Strong, hardware-enforced isolation but (seemingly) **arbitrary limitations**

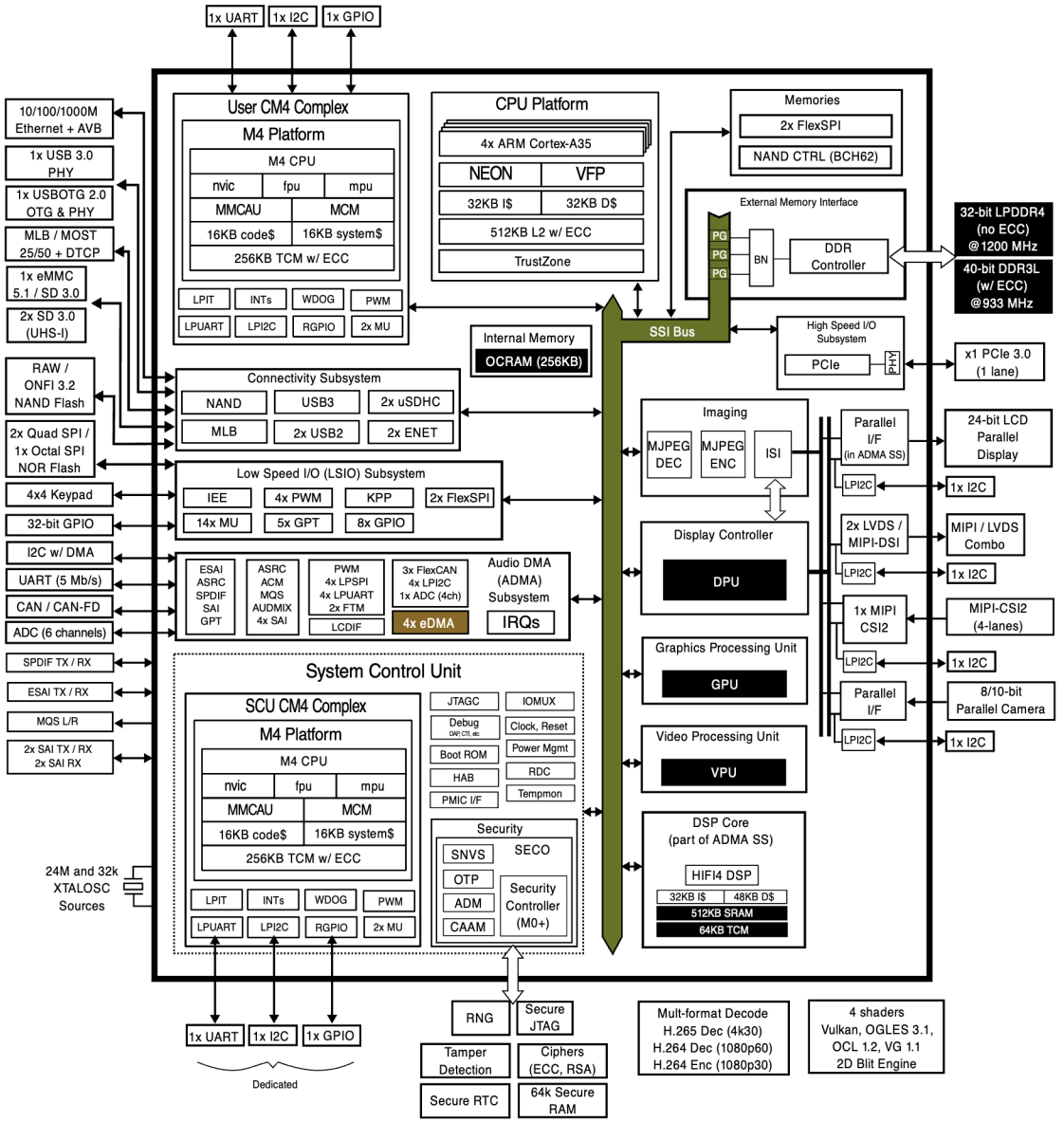
→ what are the actual limits?

Hardware designers are aware of the number of things on the main memory bus

Enhanced Direct Memory Access (eDMA)

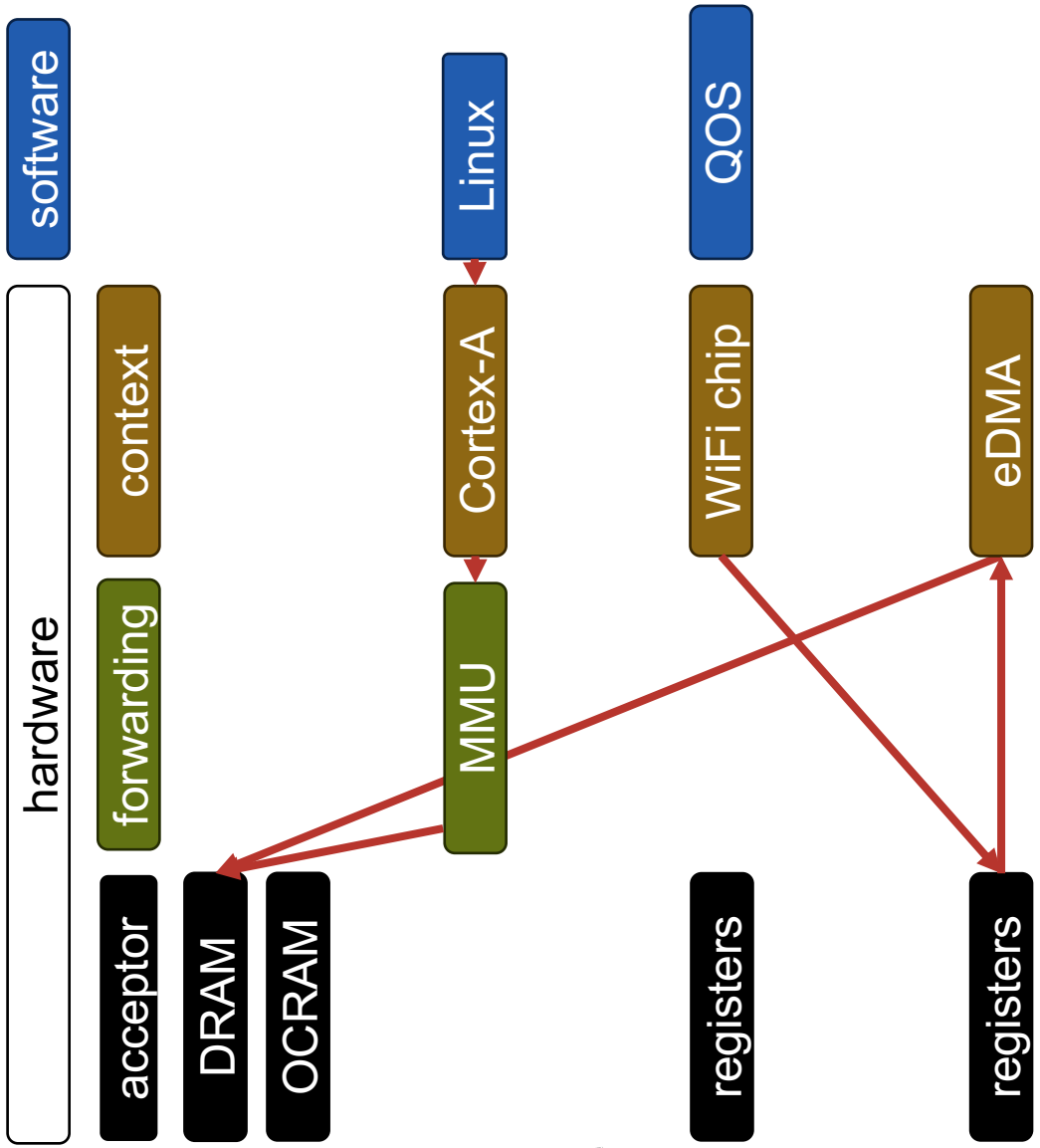


programmable, bidirectional
(e)DMA engine



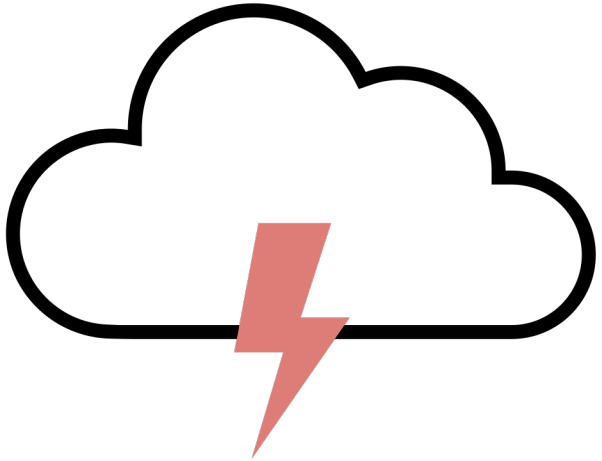
full interconnect access

Enhanced Direct Memory Access (eDMA)

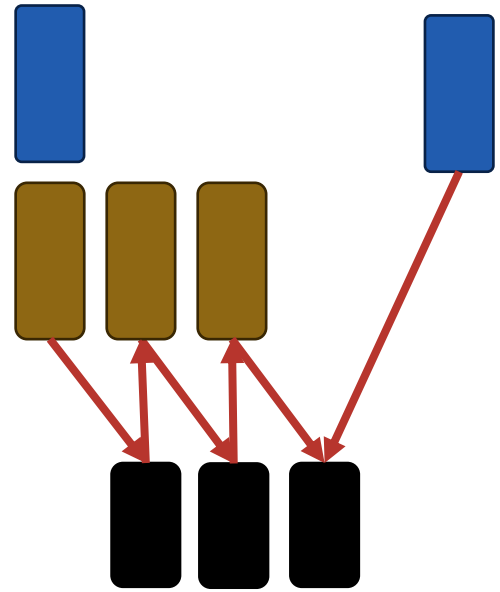


1. The eDMA engine could access Linux' DRAM memory **directly**
2. The eDMA engine could be programmed to access Linux' DRAM **on behalf of another** context

Enhanced Direct Memory Access (eDMA)



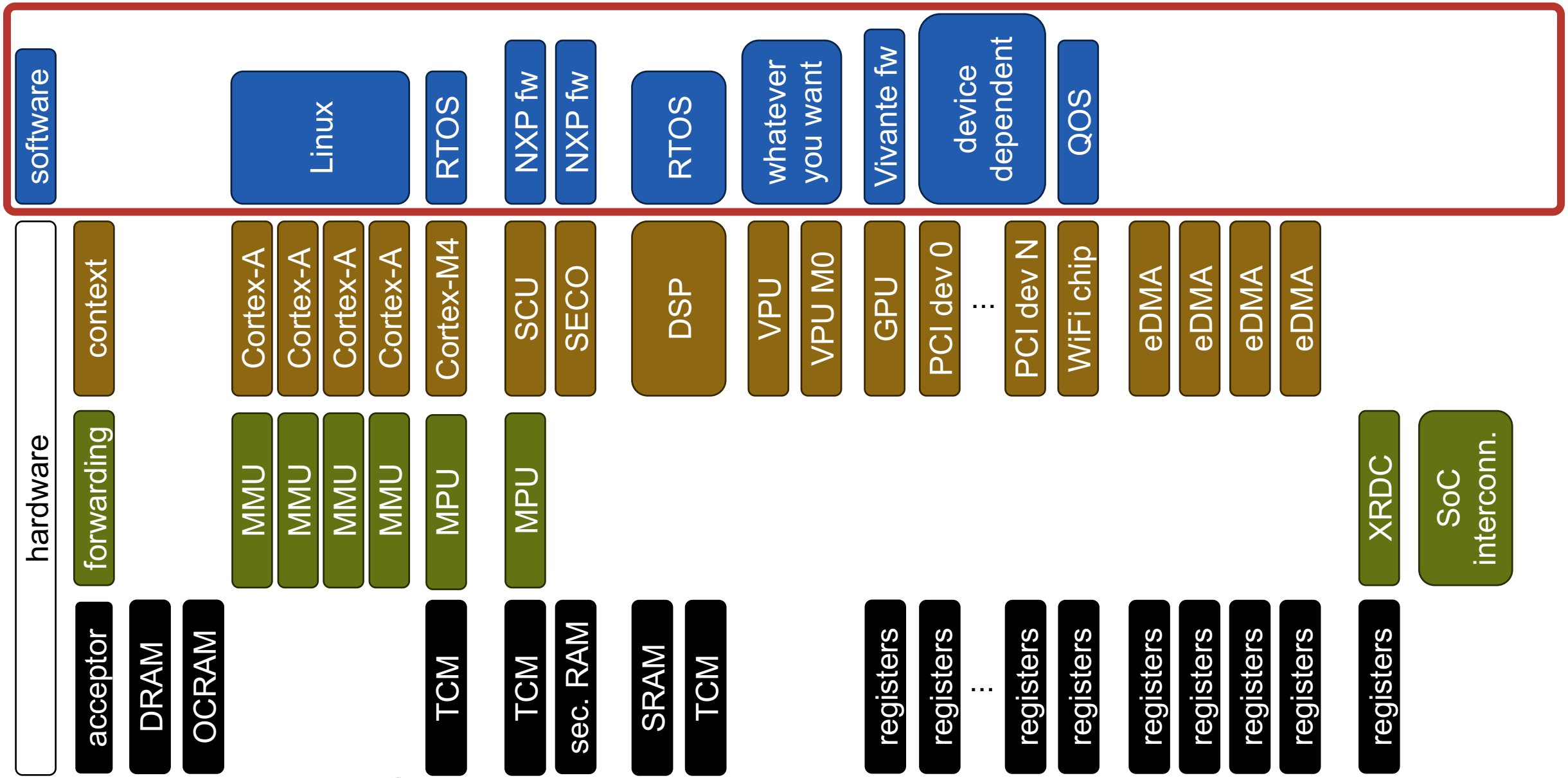
Individual channels can be assigned to **different XRDC partitions**, can lead to authority mismatch



eDMA engine can increase memory access capabilites via **transitivity**

→ not unique to eDMA, but eDMA has no firmware on top

The *de facto* OS of the i.MX8



The *de facto* OS of the i.MX8

Everything is **trusted**, nothing is **trustworthy**

Solid foundation allows understanding how different **trust assumptions** affect overall quality of guarantees

- Which components should be verified (first)?
- What guarantees **measurably** reduce the amount of trust in the *de facto* OS?