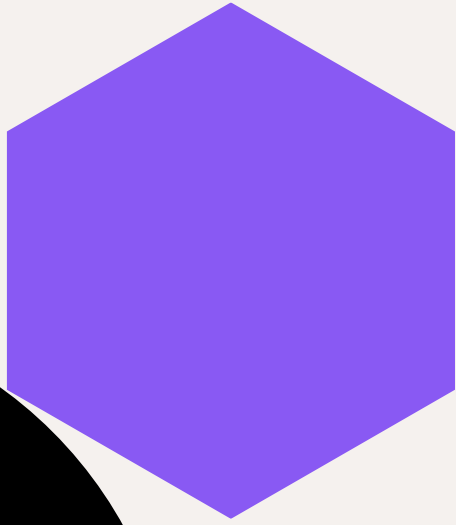
Let’s make a standard for CHERI-RISC-V

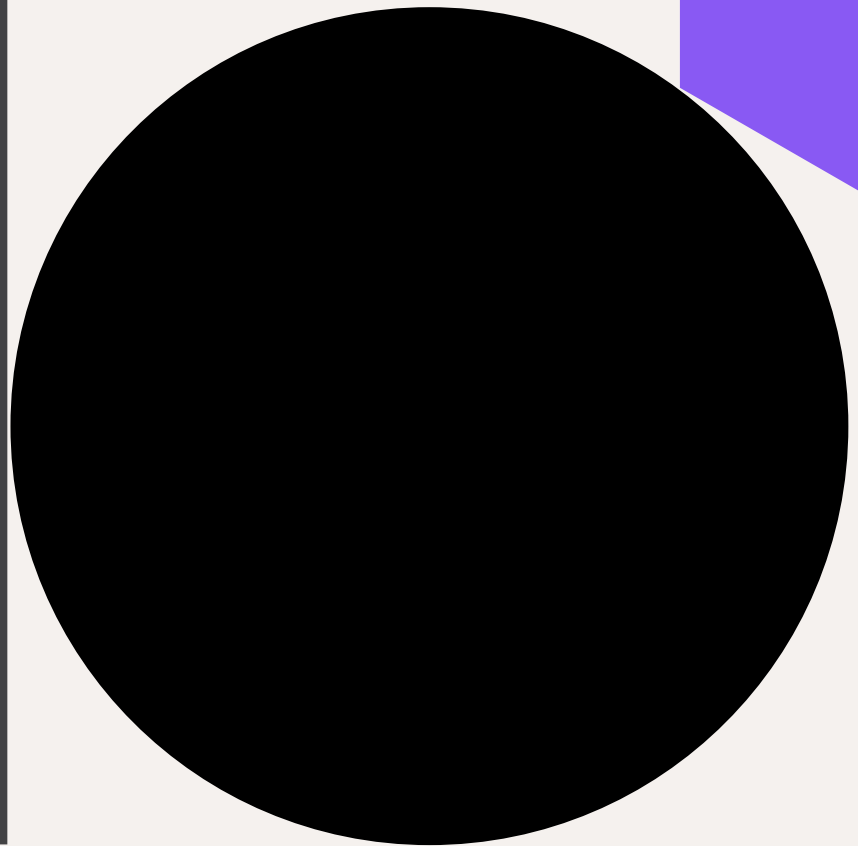
To make memory safety available for everyone, from small cores to high performance

Tariq Kurd, Chief Architect at Codasip

RISC-V Summit Munich, June 2024

Codasip

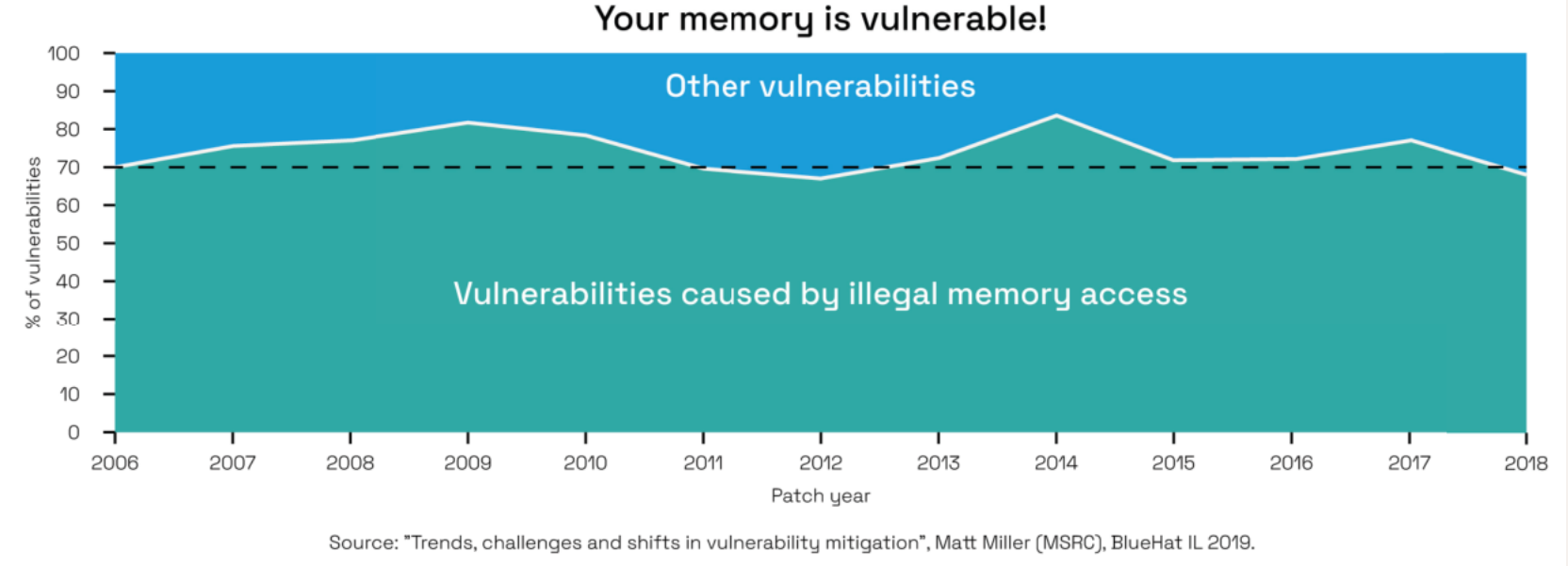




→ CHERI provides **deterministic** memory safety

Codasip

- Memory safety vulnerabilities are costly. For example, losses due to the well-known OpenSSL heartbleed bug are estimated to exceed $500 million. So there is increasing interest, even from the White House and UK government, to mitigate these vulnerabilities.
- CHERI-RISC-V is *the* comprehensive solution to mitigating 70% of vulnerabilities
- See the numerous posters in the expo hall on the subject



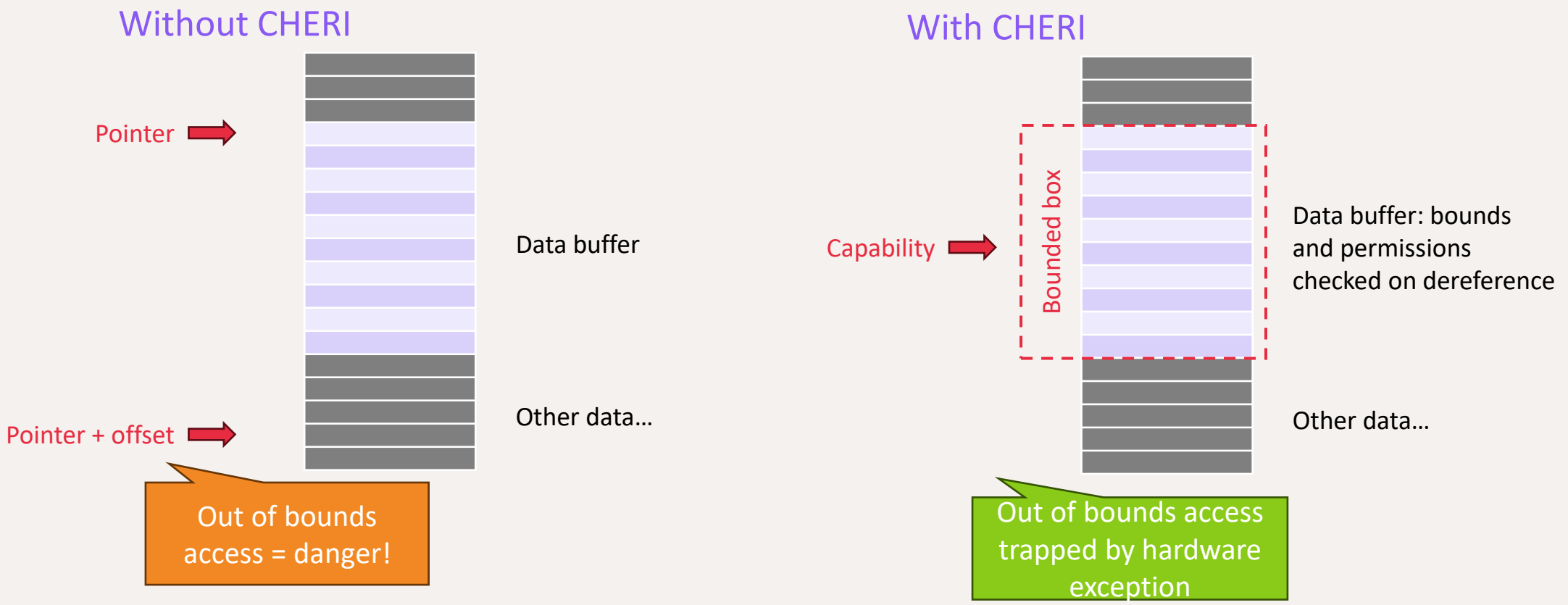
Source: "Trends, challenges and shifts in vulnerability mitigation", Matt Miller (MSRC), BlueHat IL 2019.

→ CHERI concept: deterministic bounds and permissions

Spatial memory safety example, temporal safety also supported

Codasip

Replacing pointers by capabilities – with precise checks and hardware exceptions



→ CHERI exists on multiple architectures and across multiple implementations

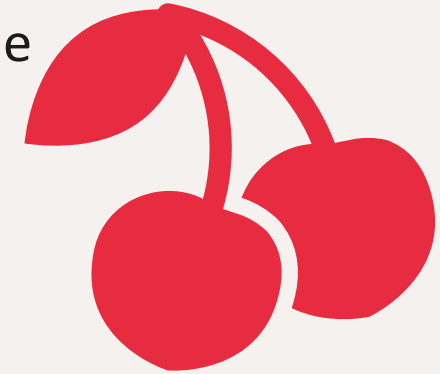
Codasip

- It was originally based on MIPS, now deprecated
- RISC-V is now the base architecture, for the CHERI v9 architecture from Cambridge University (known as CHERI-RISC-V)
- CHERI is available on an ARMv8 (Morello) development board.
  - This can be demonstrated as a workstation running CHERI BSD.
  - Server class NEOVERSE N1
- An x86 prototype sketch is also available
- CHERIoT exists as a branch of CHERI
  - Tiny implentations for IoT devices
- CHERI-RISC-V aims to standardise CHERI for the whole range from IoT to Application Cores, to Server Cores

→ Getting to the RISC-V GitHub repo

Codasip

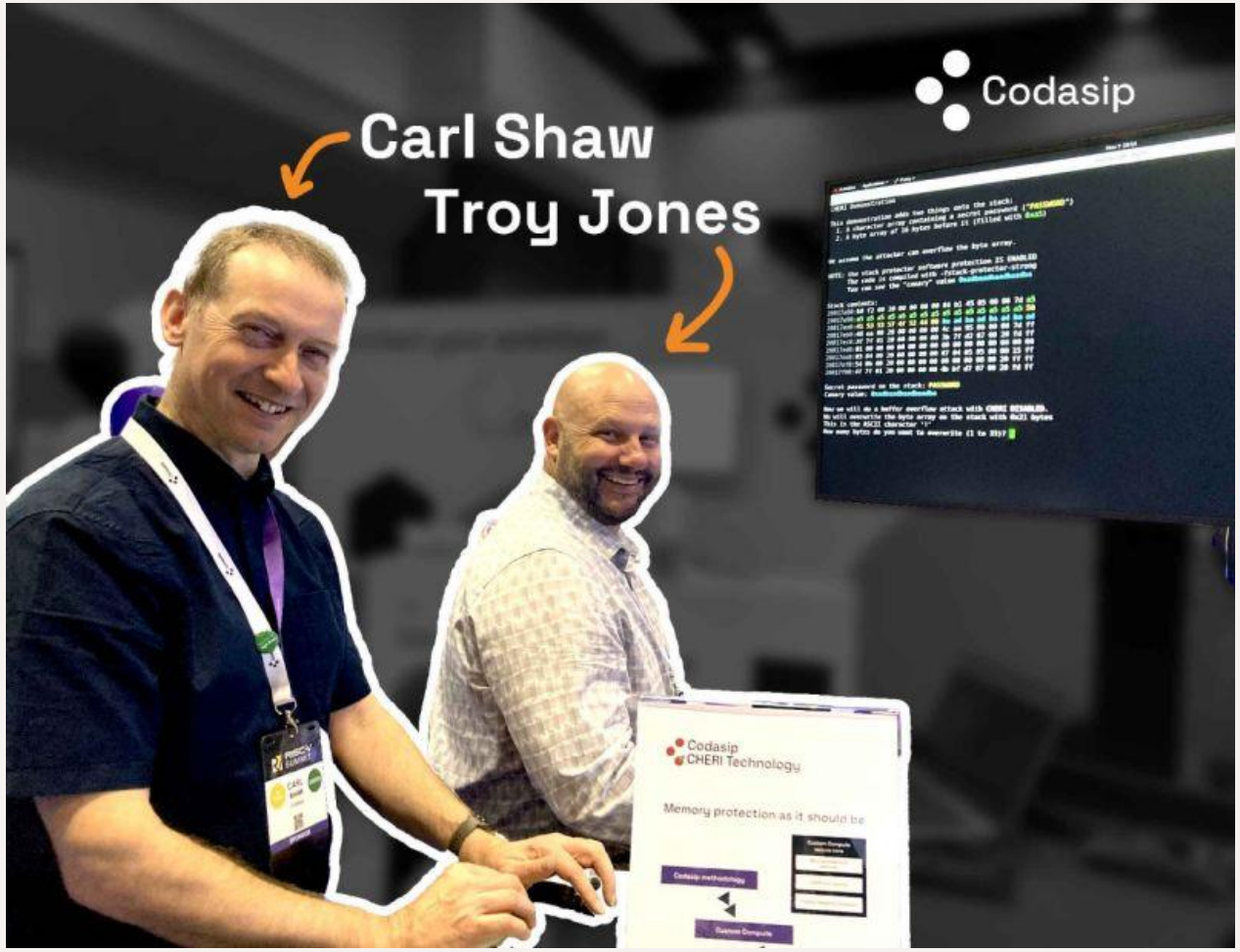
- Codasip started working with Cambridge University on the CHERI-RISC-V specification after a discussion at the RISC-V Summit in Barcelona in June 2023
- Codasip were already working in the background on a different version of the CHERI specification document
  - Extracting well defined features from CHERI v9
    - Postponing experimental and less well-defined features
    - Defining a stable base architecture
  - Filling in gaps, such as there being no debug specification
  - Written as an implementation spec
  - Covers all the necessary questions asked by the implementation and verification teams to allow the product to be built
- Codasip tested this spec on their A730-CHERI core development



→ The Demo at the RISC-V Summit Santa Clara Nov ‘23

Codasip

Carl Shaw and Troy Jones showed the A730-CHERI Application Core prototype detecting a buffer overrun on the stack before the stack canary spotted it, running on an FPGA



→ CHERI-RISC-V v0.7.0

Codasip

January 2024

After review with Cambridge Uni, the Codasip CHERI spec document became v0.7.0 on GitHub:
https://github.com/riscv/riscv-cheri/releases/

The Task Group was formed

Then the real specification work started refining the architecture

→ RV32: Optimised Format since CHERIv9

Codasip

The improved encoding has

- 0 to 2-bit software defined permissions
- 13 to 5-bit architectural permissions with encoding room for expansion
- O to 4-bit reserved field (for new features such as local/global capability access)
- 4 otype bits to 1-bit sealed
- The mantissa has increased from 8 to 10-bits for better precision

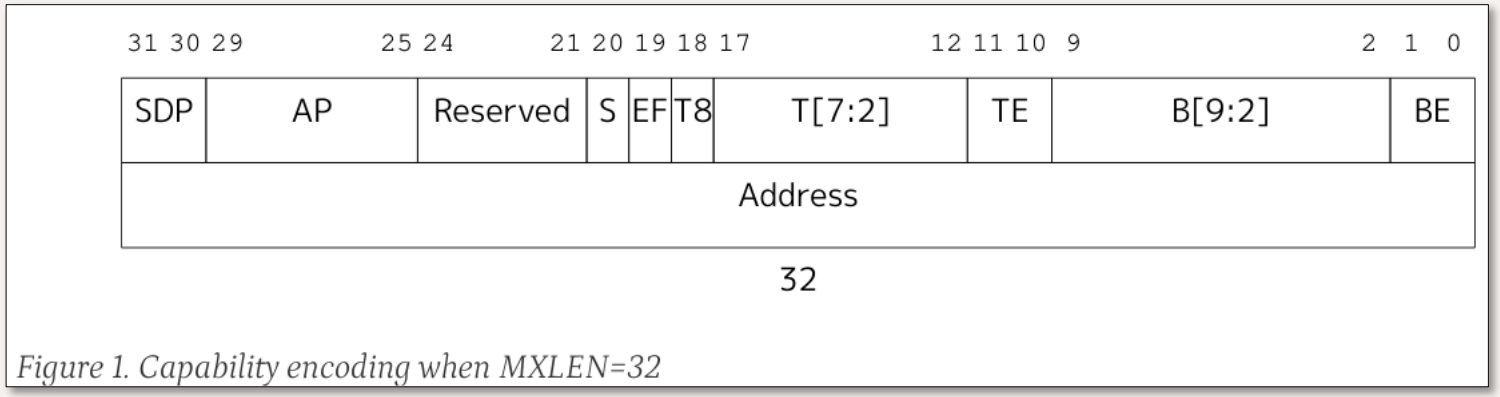
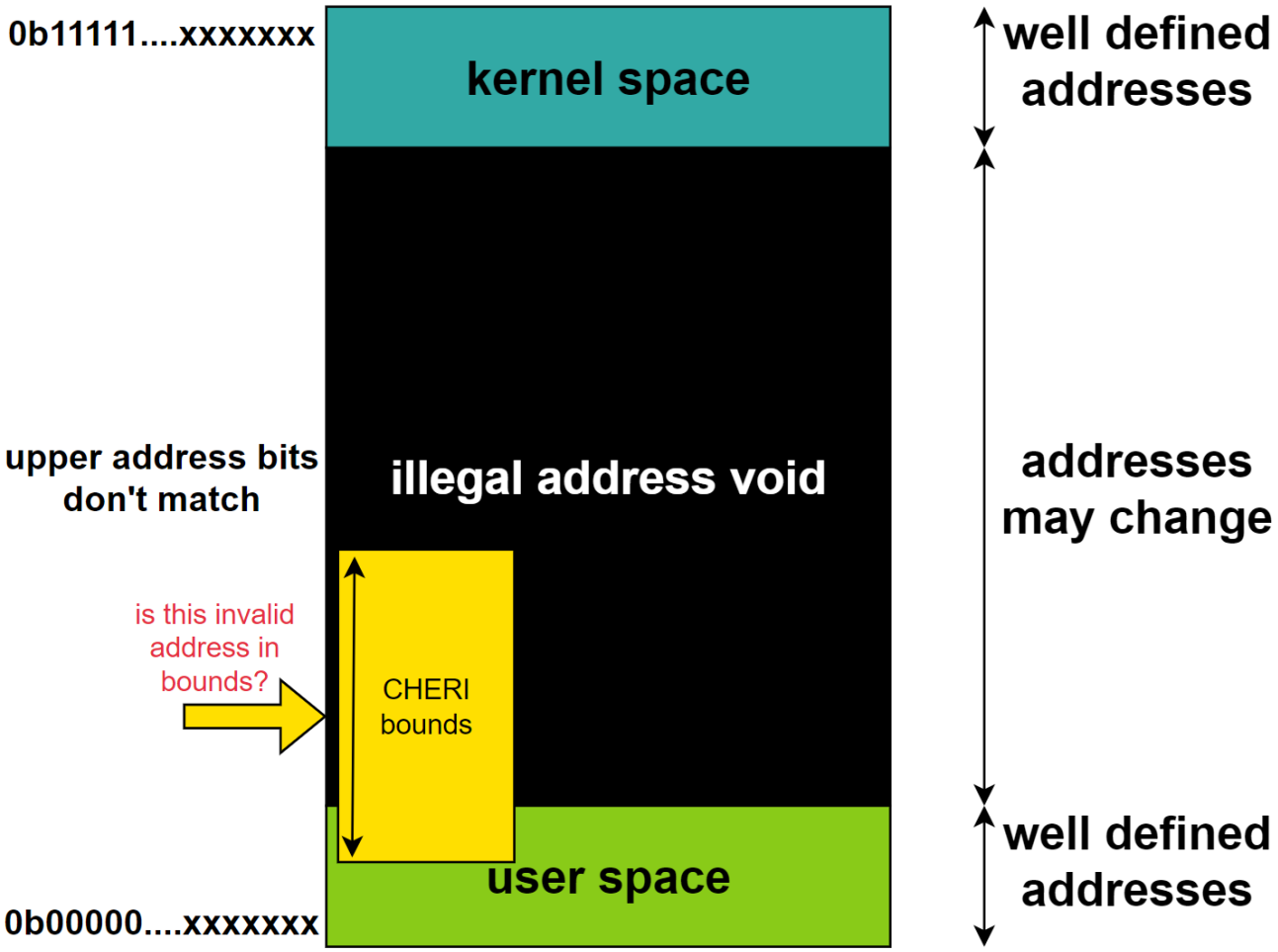


Figure 1. Capability encoding when MXLEN=32

→ CHERIv9 Invalid address handling: the problem



Invalid addresses may change when written to registers such as MEPC for Sv39 or Sv48

Is the new address in bounds or not? Do we need to check after changing the address? It's not cheap to do so.

→ Illegal address handling: the solution

Codasip

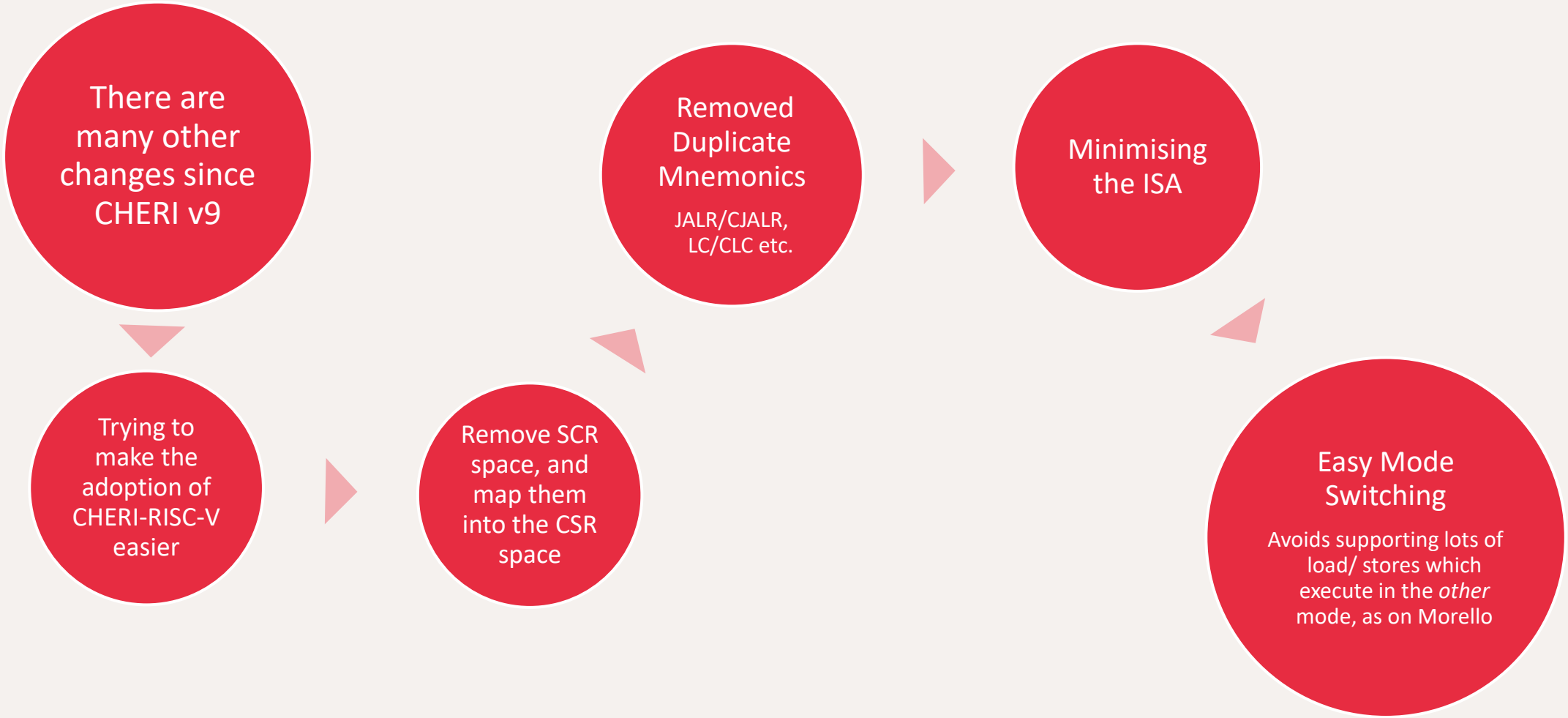
A new CHERI exception type for illegal addresses instead of a bounds check

- For running CHERI software only
  - Take an invalid address CHERI exception, so we don’t care if the address is representable or in bounds or not
- Existing RISC-V code on a CHERI core still takes an access fault

Reduces the size of the bounds comparators

- Previously CHERI required full 64-bit address comparators
  - Now we need 39-bit for Sv39, 48-bit for Sv48
  - This gives a nice power and area saving, and is simpler
- This also compatible with pointer masking as we don’t need to compare the masked range of the address

→ Other changes?



→ What extensions do we have?

Codasip

Extension	Status	Description
ZcheriPurecap	Stable – bug fixes only	Base architecture for a CHERI purecap machine
ZcheriHybrid	Stable – bug fixes only	Implies ZcheriPureCap. Adds legacy RISC-V support
Zabhlrsc	Stable – bug fixes only	Byte/half LR/SC support (independent of CHERI)
Zstid	Software prototyping	Secure thread ID for **Compartmentalisation**
ZcheriHypervisor	Prototype, need PR	CHERI and **Hypervisor** support
ZcheriVector	Prototype, need PR	CHERI and **Vector** optimised support to allow Vector capability **memcpy**
ZcheriPTE	Prototype, PR needs update	Optimised **Revocation** support by supporting capability **accessed** and **dirty** in page tables
ZcheriTransitive	Prototype, PR needs update	Support for **reducing capability permissions** on loading
ZcheriMultiLevel	Research, need PR	Support for locally/globally accessible capabilities with multiple levels
ZcheriTraceTag	Research, Need PR	Support for data capability trace with tags
ZcheriSanitary	Research, Need PR	Support for cleaning capabilities on compartment switching
ZcheriSystem	Research, Need PR	Support for exposing compartment IDs to the system (a better WorldGuard)

→ Getting to RVA23 Compatibility

Codasip

We're currently have RVA22+CHERI fully working for *mandatory* extensions. We want to get to RVA23+CHERI, which has some gaps to fill:

- **Vector+CHERI: Fairly simple – check every unmasked byte of every load/store against the bounds**
  - But the devil is in the details for complex sequenced masked load/stores
  - Indexed loads where the base is zero and the entire address is in the element are a problem: the capability will need to have the address field set to zero, and so the bounds must start at zero
  - Consider adding VLC/VSC to load/store full vector registers including caps, and an associated VCMV to move a whole vector register to support Vector capability memcpy
- **Vector+Hypervisor: more on the next talk**
- **Pointer masking: seems easy – needs confirmation**

→ Code Size Reduction

Codasip

→Already in the CHERI-RISC-V spec

Zcmt – table jump

- JVT becomes a capability

Zcmp – push/pop

- The data-width doubles, and only RV32 is of interest, so effectively use the RV64 stack layout for RV32-CHERI-RISC-V

→ Conclusions

Codasip

CHERI is 100% compatible with RVA22 mandatory extensions

Will soon be with RVA23 too

CHERI runs existing RISC-V code with 100% compatibility

This makes adoption of CHERI much easier

We’re working on CHERIoT compatibility

We want one ratified specification to cover all variants

→ The future….

Codasip

All new Codasip cores will have CHERI

The CHERI Alliance will grow – first members announced

CHERI world domination – all cores memory safe..?

Codasip

↘

That’s all folks

- Collaborate with us: https://github.com/riscv/riscv-cheri
- Join the TG and the bi-weekly meeting
- Tell us what’s missing and help fill in the gaps
- Help drive CHERI to world domination