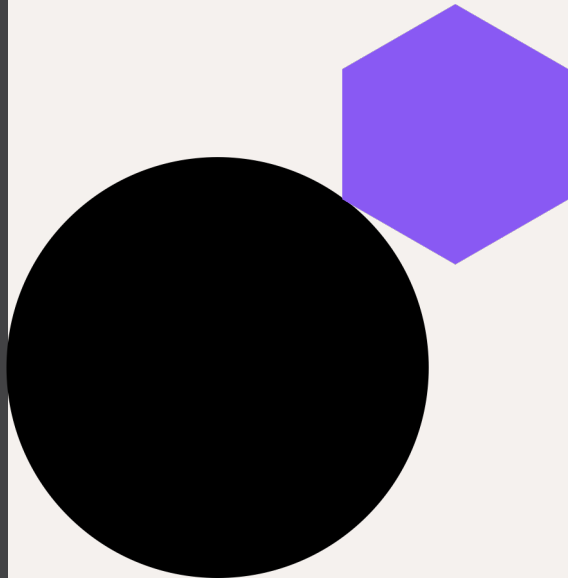
Let’s make a standard for CHERI-RISC-V

Codasip



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

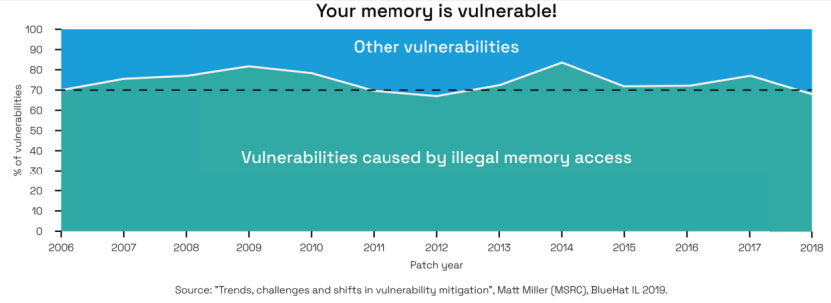
Tariq Kurd, Chief Architect at Codasip, Based in Bristol UK

RISC-V Day Tokyo, February 2025

→ 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. Heartbleed included a buffer overrun which would have been prevented by CHERI.
- A recent White House report "*Back to Building Blocks: A Path Toward Secure and Measurable Software*" requires memory safety & recommends CHERI
  - https://www.whitehouse.gov/wp-content/uploads/2024/02/Final-ONCD-Technical-Report.pdf
- CHERI is part of the UK Semiconductor strategy:
  - https://www.gov.uk/government/publications/national-semiconductor-strategy
- CHERI-RISC-V is *the* comprehensive solution to mitigating 70% of vulnerabilities



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

→ CHERI-Linux is real on RISC-V

- The Codasip X730 RVA22 RISC-V application core runs CHERI-Linux on an FPGA
- Dual-issue in-order 4-ALU (early/late) core with full support for CHERI, and 100% compatibility for RISC-V (legacy) binaries.
- It implements the latest specification, as explained in this presentation
- The world's first commercially available CHERI-RISC-V application core
- It can be used to demonstrate successful detection of the SSL Heartbleed exploit and many many others
- CHERI is also very good at detecting programming errors, not just for detecting exploits.

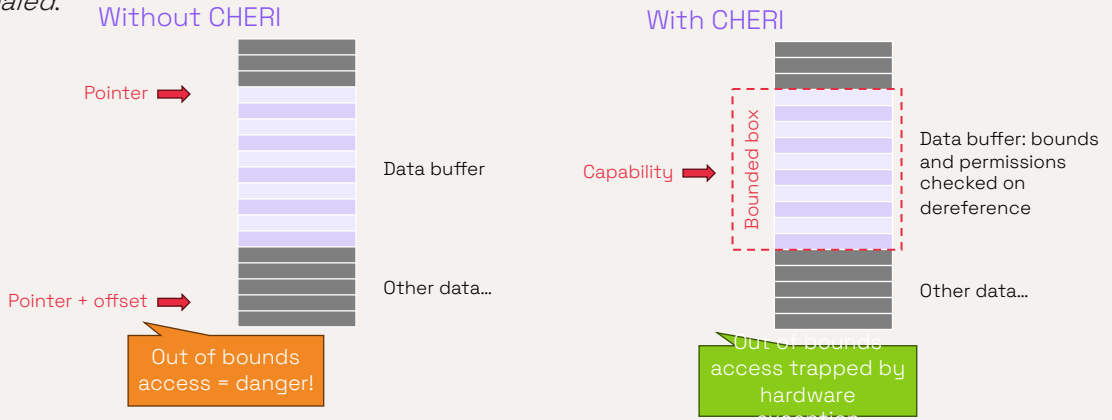
→ CHERI concept: deterministic bounds and permissions

Spatial memory safety example, temporal safety also supported

Codasip

Replacing pointers by capabilities – with precise checks and hardware exceptions.

Every address becomes a *capability* with *bounds* and *permissions*, and which can be *sealed*.



→ 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 CheriBSD.
  - Server class NEOVERSE N1
- An x86 prototype sketch is also available
- CHERIoT is a variant of CHERI based on the 0.9.5 RISC-V proposed standard; it is for RV32E implementations for IoT devices. Silicon for OpenTitan is expected this year.
- CHERI-RISC-V aims to standardise CHERI for the whole range from IoT to Application Cores, to Server Cores

→ CHERI-RISC-V Spec 0.9.5

https://github.com/riscv/riscv-cheri/releases/tag/v0.9.5

- Latest spec release from October 2024
  - Silicon is expected in 2025 for CHERIoT, CHERI-RISC-V RV32 and CHERI-RISC-V RV64
  - *CHERIoT is built on top of this spec release and has additional features*
- New features since CHERI v9
  - RVA23 compatibility
    - Vector, Hypervisor, Pointer Masking
  - Load Mutable
  - Capability levels (local/global)
  - Allow alternate capability formats
  - New instructions: GCMODE, GCTYPE, MODESW.*
  - Improved integer pointer mode CSR handling
  - Compatibility with Zilsd/Zcld, newly ratified
  - Add Xtval2 for extra exception reporting
  - Legacy debugger support

→ Page Table Entries

Codasip

CHERI-RISC-V adds two new bits to the page table entries to allow software to sweep memory to remove stale capabilities, to prevent use-after-free violations.

PTE.CW - mandatory

- Never allow a capability with the tag set to be stored if CW=0

PTE.UCRG – optional (Svucrg) Capability Read Generation

- Adds whether the current page needs to be swept before it can be loaded from (currently for Userspace pages only)

Thread identification for compartmentalisation

- Necessity to identify current thread
- Needs protection not guaranteed by mechanisms already present
- Allows nestable compartmentalisation

	Index	Writeable by	Readable by
`mtidc`	`0x780`	{M} + ASR	{M}
`vstidc`	`0xA80`	{M, H} + ASR	{M, H}
`stidc`	`0x580`	{M, H, S} + ASR	{M, H, S}
`utidc`	`0x480`	{M, H, S, U} + ASR	{M, H, S, U}

Current software evaluation in library compartmentalisation on CHERI-RISC-V by the University of Cambridge

→ RVA23 Compatibility – new mandatory features

Codasip

- Hypervisor+CHERI
  - Add CRE (CHERI Register Enable) to guest OSs to allow them to have CHERI enabled or not
    - The hypervisor itself needs to be CHERI aware
  - Extend HS-mode/VS-mode registers to support CHERI
    - Add CRE, and the CHERI trap value reporting fields
  - Currently no support for CHERI in the stage-2 page table entries (no CW or xCRG bits)
- Pointer Masking+CHERI
  - Bounds decoding altered to support pointer masking
  - Changing the number of masked bits affects the meaning of the bounds

→ RVA23 Compatibility – Vector

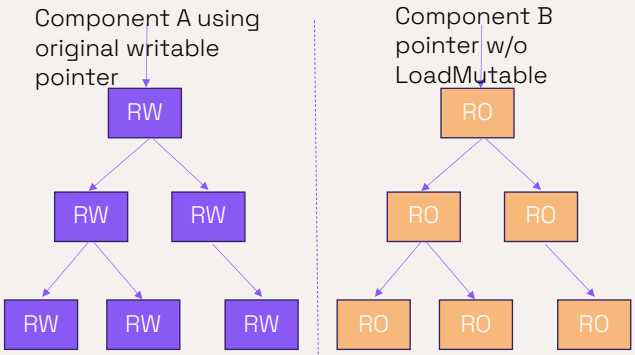
Codasip

- Vector+CHERI – *do not extend Vector registers to hold capabilities (yet – future work)*
- Applying CHERI to vector loads/stores/atomics requires that every *active element* of access is authorized, do not check *inactive elements*.
  - If there are no active elements, then don't take a CHERI exception, even if the authority capability has no validity tag.
  - For fault-only-first loads, only take a CHERI exception if it affects element 0, so if element 0 is masked and the authority capability has no validity tag then *don't take a fault*, only reduce vl to 1

→ CHERI-RISC-V v0.9.5 feature – Load Mutable

Codasip

- Allows **efficient sharing of read-only views** of data
- Assume component A wants to share a tree with component B
- Stripping LoadMutable removes write (and LoadMutable) permission from any loaded capabilities
- So removing it from the root pointer makes the entire data structure recursively read-only
- Without this feature sharing a tree-like structure
  would require making a deep copy
- Not a completely new feature, it has also been implemented in Morello and CHERIoT



→ CHERI-RISC-V v0.9.5 feature – Levels (local/global)

Codasip

- When sharing capabilities between compartments it is important to restrict the flow of certain pointers
  - E.g. prevent storing the caller's stack arguments to a heap data structure
- As usual this can be done via deep copies or indirection, but we want to be more efficient
- Capabilities are assigned to a level
- Add new permission bits to prevent storing of lower levels (StoreLevel) a restrict loading to the authorizing capability's level (ElevateLevel)
- Prior implementations (CHERI v9,Morello,CHERIoT) support one level bit (local vs global)
  - This is the baseline for this extension, but we designed it to be extensible to multiple bits of levels

→ What extensions do we have? Several!

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
Zish4add	Stable – bug fixes only	SH4ADD as the datatype is 128-bit (RV64 only)
ZabhlrsC	Stable – bug fixes only	Byte/half LR/SC support (independent of CHERI)
Svucrg	Stable – in software prototyping	Optimised **Revocation** support by supporting capability **accessed** and **dirty** in page tables
Zstid	Stable – in software prototyping	Secure thread ID for **Compartmentalisation**
ZcheriLevels	Stable – in software prototyping	Support for locally/globally accessible capabilities with multiple levels
ZcheriVector	Research, Need PR	CHERI and **Vector** optimised support to allow Vector capability **memcpy**
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)

Green and blue extensions are in 0.9.5, red are future work

→ Code Size Reduction for RV32

Codasip

→Already in the CHERI-RISC-V 0.9.5 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 RVA23 mandatory extensions

CHERI runs existing RISC-V code with 100% compatibility

This makes adoption of CHERI much easier

The specification can be used as a base for CHERIoT

→ The future....

Codasip

All new Codasip cores will have CHERI

The CHERI Alliance is here, see https://cheri-alliance.org/

CHERI world domination – all cores memory safe..?

Codasip

↘

That’s all folks

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