

CVA6-CHERI

An Open-source RV64Y Implementation
for Commercialization



Zero-Day
Labs

Bruno Sá, **Jonathan Woodruff**, Peter Rugg,
Alexandre Joannou, Sandro Pinto, Simon W. Moore

CHERITech'25
November 14, 2025



CAPABILITIES
LIMITED



CAPABILITIES
LIMITED

CVA6-CHERI Project: A tale of two organizations

Zero-Day Labs



Goal: Initial, viable implementation to test interaction of hypervisor and CHERI for mixed-criticality, embedded systems

(Preliminary presentation at RISC-V Summit Europe, 2024; released early 2025)

Capabilities Limited



Goal: Zcheri reference design of sufficient quality and performance

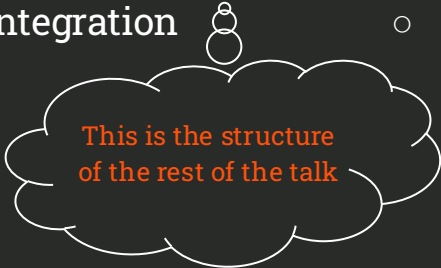
showcase the potential of CHERI-RISC-V implementations



Capabilities Limited RV64Y CVA6-CHERI Project

From research ISA prototype toward ratified extension production candidate

- Implementation
 - RV64Y instruction decoding
 - “Mode switch” instruction
 - New, integrated capability CSRs
- Verification
 - Formal verification of Capability operations
 - More complete TestRIG integration
- Area Optimization
 - Eliminated PCC redundancy
 - Consolidated Execute functions
 - Removed research architecture features
- Timing Optimization
 - Performed capability decompression in parallel to byte select
 - Removed selectors from the register file for research architecture’s fast register clearing



This is the structure
of the rest of the talk

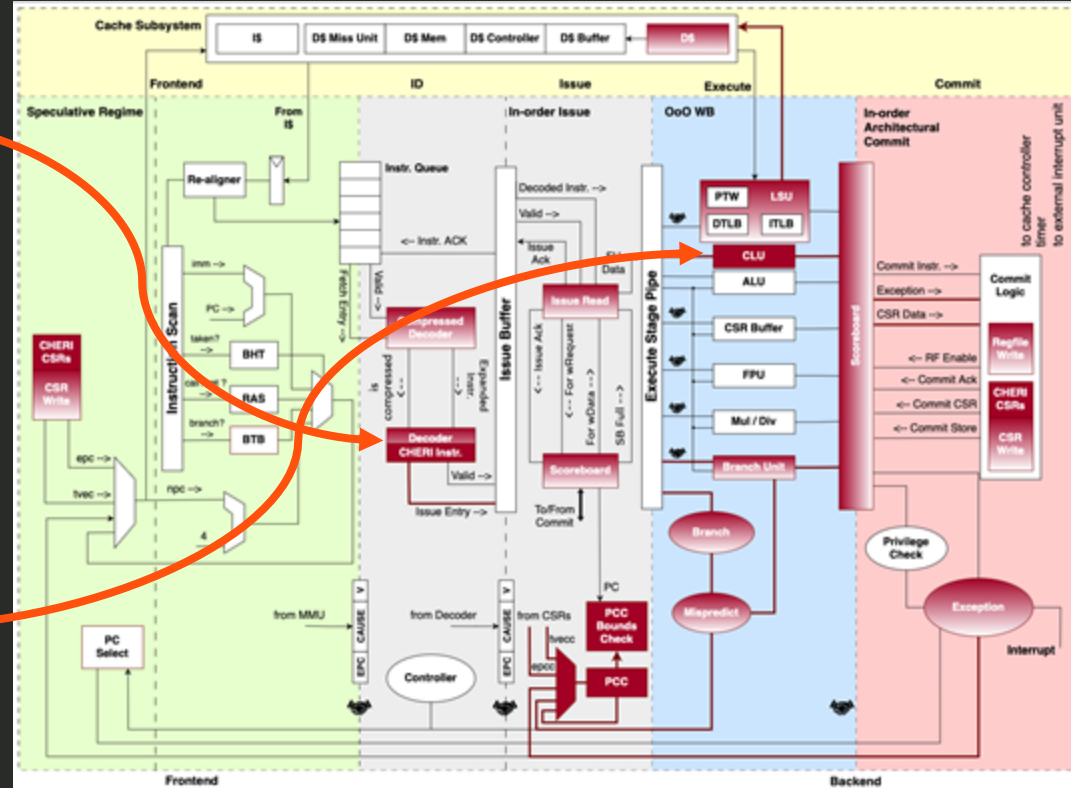


Implementation



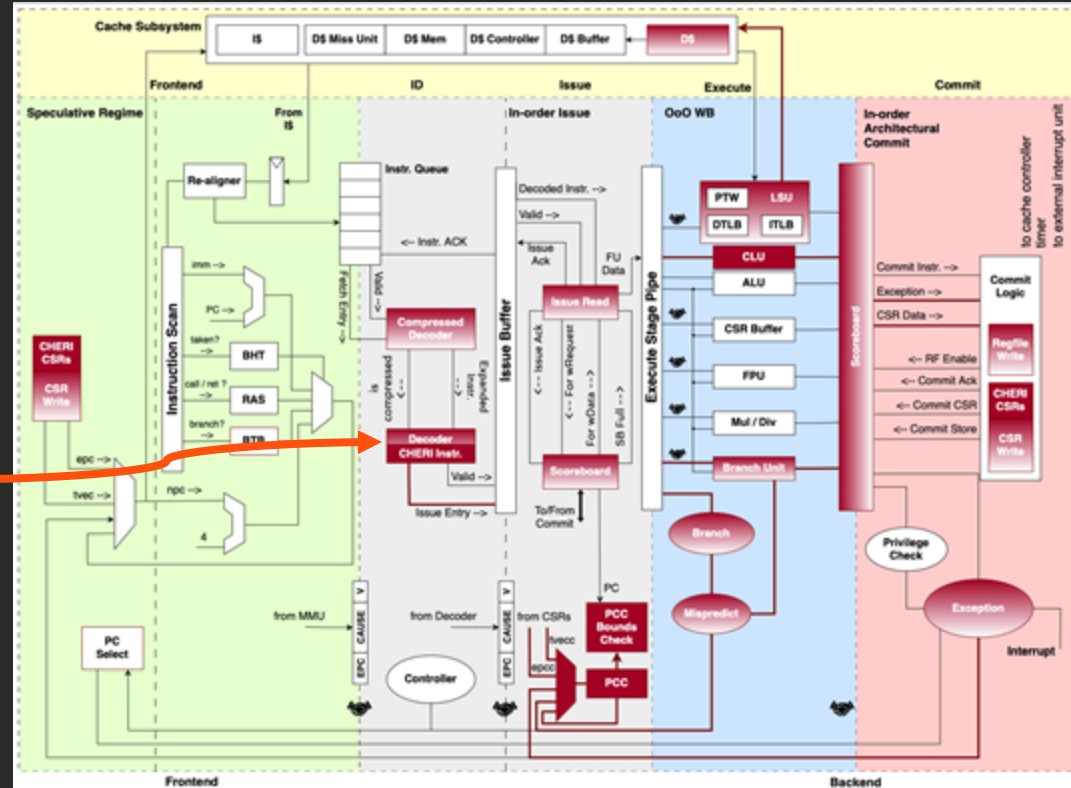
Y Extension Pipeline Updates

- Completed Decode of new instructions
 - Removed many Cambridge research architecture instructions
 - Moved remaining instructions under existing major opcodes, matching the candidate RV64Y architecture
- Execute nearly unchanged (beyond updated CHERI Library)



Implemented mode switch instructions in Decode

- CHERI mode asserts all loads and stores against address register bounds (rather than default bounds)
- YMODESWY/I instructions set the decoding mode to CHERI or Non-CHERI
- “Mode bit” of PCC owned by Decode
 - Mode is assigned in Decode
 - Modifications of the mode in Decode are reflected immediately
 - Jumps that change mode (mode coming from register value) cause a mispredict. This is rare.



Verification



SystemVerilog Capability Library

Reusable Y extension microarchitectural functions
in SystemVerilog and Bluespec

```
function Bool capInBounds(CapFat cap, TempFields tf, Bool inclusive);
// Check that the pointer of a capability is currently within the bounds
// of the capability
Bool ptrVStop = inclusive ? cap.addrBits <= cap.bounds.topBits
                          : cap.addrBits < cap.bounds.topBits;
// Top is ok if the pointer and top are in the same alignment region
// and the pointer is less than the top. If they are not in the same
// alignment region, it's ok if the top is in Hi and the bottom in Low.
Bool topOk = (tf.topHi == tf.addrHi) ? ptrVStop : tf.topHi;
Bool baseOk = (tf.baseHi == tf.addrHi) ? cap.addrBits >= cap.bounds.baseBits
                                       : tf.addrHi;
return topOk && baseOk;
endfunction
```

Bluespec

```
/**
 * @brief Function to check if capability is within bounds.
 * @param cap capability in register format.
 * @param cap_meta_data capability decoupled bounds meta data.
 * @param inclusive 0 - includes top in the in bounds check.
 *                1 - excludes top in the inbounds check.
 * @returns 0 if not in bounds and 1 if in bounds.
 */
function automatic bool_t is_cap_reg_inbounds(cap_reg_t cap, cap_meta_data_t meta_data, bool_t inclusive)
    mw_t addr_mid = cap.addr_mid;
    bool_t check_addr = inclusive ? addr_mid <= cap.bounds.top_bits
                                : addr_mid < cap.bounds.top_bits;
    bool_t check_top = (meta_data.top_hi_r == meta_data.addr_hi_r) ? check_addr : meta_data.top_hi_r;
    bool_t check_base = (meta_data.base_hi_r == meta_data.addr_hi_r) ? addr_mid >= cap.bounds.base_bits
                                                                    : meta_data.addr_hi_r;
    return check_top && check_base;
endfunction
```

SystemVerilog

Tidy in both cases. Thanks to Bruno for developing a clean SystemVerilog style!



Capability Library Verification

- Use SymbiYosys-based equivalence proofs between Bluespec and SystemVerilog functions
- >21 functions proved equivalent
- Discovered ~5 bugs (one serious) in the initial library
- Guides ports to new versions...

```
*def_cap_mem_equiv_0arg(is_valid_reg, cap_mem_wrap_is_cap_reg_valid, module_wrap128_isValidCap, 1)
*def_cap_mem_equiv_larg(set_valid_reg, cap_mem_wrap_set_cap_reg_valid, module_wrap128_setValidCap, 1, 129)
*def_cap_mem_equiv_0arg(is_valid, is_cap_mem_valid, module_wrap128_isValidCap, 1)
*def_cap_mem_equiv_larg(set_valid, set_cap_mem_valid, module_wrap128_setValidCap, 1, 129)
*def_cap_mem_equiv_0arg(get_flags, get_cap_mem_flags, module_wrap128_getFlags, 1)
*def_cap_mem_equiv_larg(set_flags, set_cap_mem_flags, module_wrap128_setFlags, 1, 129)
*def_cap_mem_equiv_0arg(get_hard_perms, get_cap_mem_reg_hperms, module_wrap128_getHardPerms, 12)
*def_cap_mem_equiv_0arg(get_addr, get_cap_mem_addr, module_wrap128_getAddr, 64)
*def_cap_mem_equiv_larg(set_addr_unsafe, set_cap_mem_addr_unsafe, module_wrap128_setAddrUnsafe, 64, 129)
*def_cap_mem_equiv_larg(set_addr_unsafe_reg, cap_mem_wrap_set_cap_reg_addr, module_wrap128_setAddrUnsafe, 64, 129)
*def_cap_mem_equiv_larg(set_addr_inc_unsafe, set_cap_mem_addr_inc, module_wrap128_addAddrUnsafe, 11, 129)
*def_cap_mem_equiv_0arg(get_base, cap_mem_wrap_get_cap_reg_base, module_wrap128_getBase, 64)
*def_cap_mem_equiv_0arg(get_top, cap_mem_wrap_get_cap_reg_top, module_wrap128_getTop, 65)
*def_cap_mem_equiv_0arg(get_length, cap_mem_wrap_get_cap_reg_length, module_wrap128_getLength, 65)
*def_cap_mem_equiv_0arg(get_offset, cap_mem_wrap_get_cap_reg_offset, module_wrap128_getOffset, 64)
*def_cap_mem_equiv_larg(set_addr, cap_mem_wrap_set_cap_reg_address, module_wrap128_setAddr, 64, 129)
*def_cap_mem_equiv_larg(is_in_bounds, cap_mem_wrap_is_cap_reg_inbounds, module_wrap128_isInBounds, 1, 1)
*def_cap_mem_equiv_larg(set_bounds, cap_mem_wrap_set_cap_reg_bounds, module_wrap128_setBounds, 64, 128)
*def_cap_mem_equiv_larg_bsv_argconv(set_otype, cap_mem_wrap_set_cap_reg_otype, module_wrap128_setKind, otype_to_bsv_kind, 18, 129)
*def_cap_mem_equiv_larg_bsv_argconv(seal, cap_mem_wrap_seal, module_wrap128_setKind, otype_to_bsv_kind, 18, 129)
*def_cap_mem_equiv_0arg(unseal, cap_mem_wrap_unseal, wrap_bsv_unseal, 129)
```

Macro which asserts
equivalence between
two functions

Name the two
functions to be
compared

Specify the size
of return type

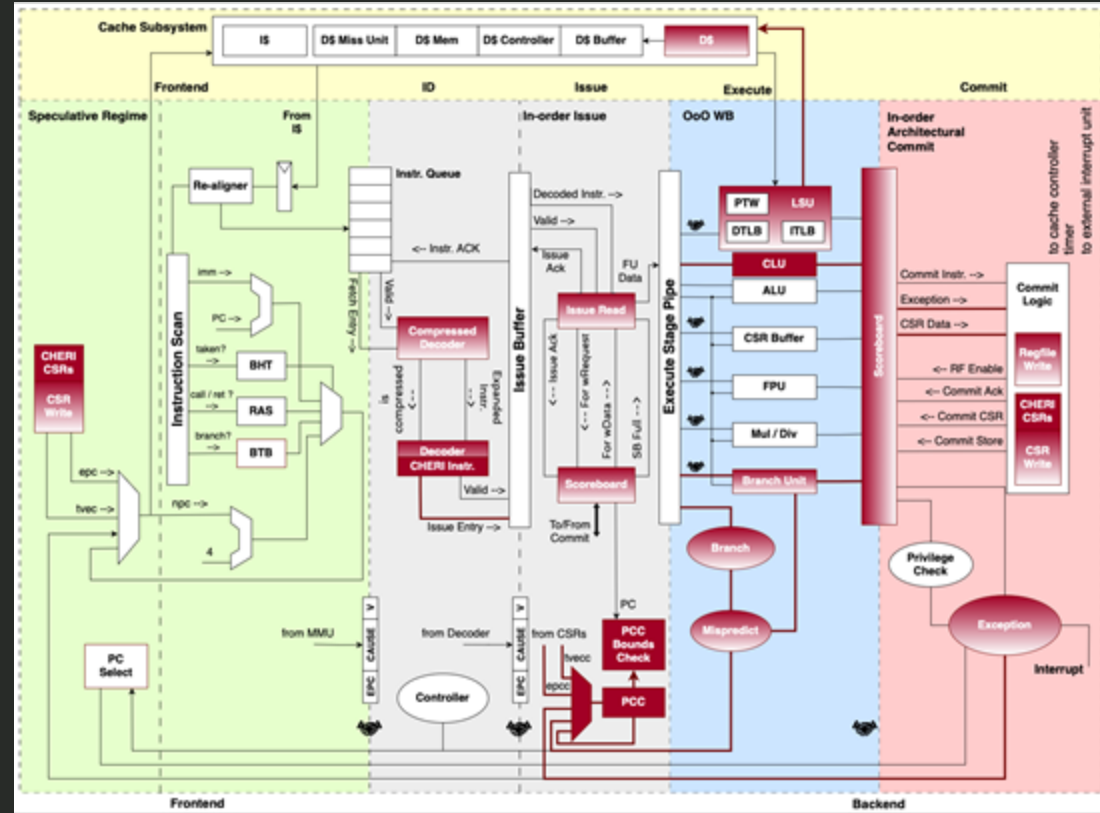


Area Optimization



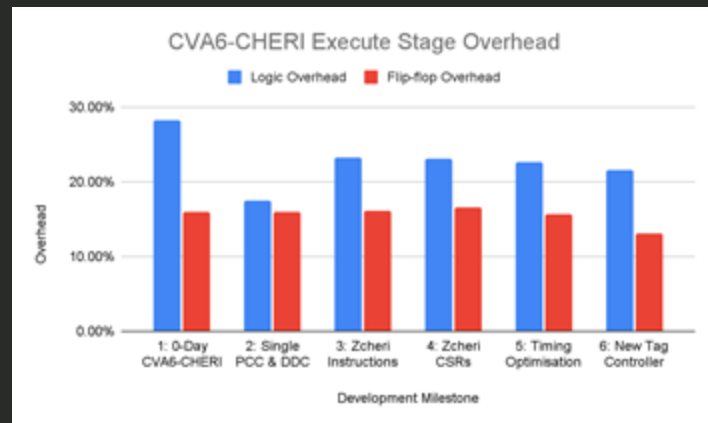
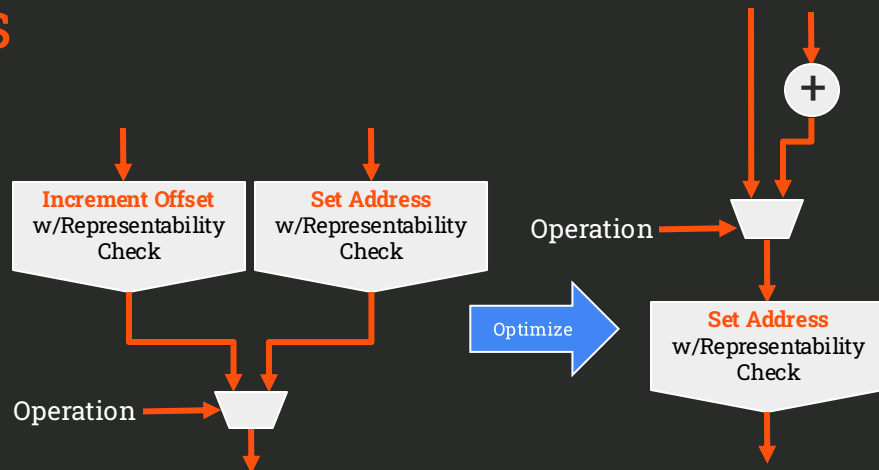
CAPABILITIES
LIMITED

- Logic overhead is tracked less precisely in Vivado, but recent builds also have no logic overhead for the frontend.



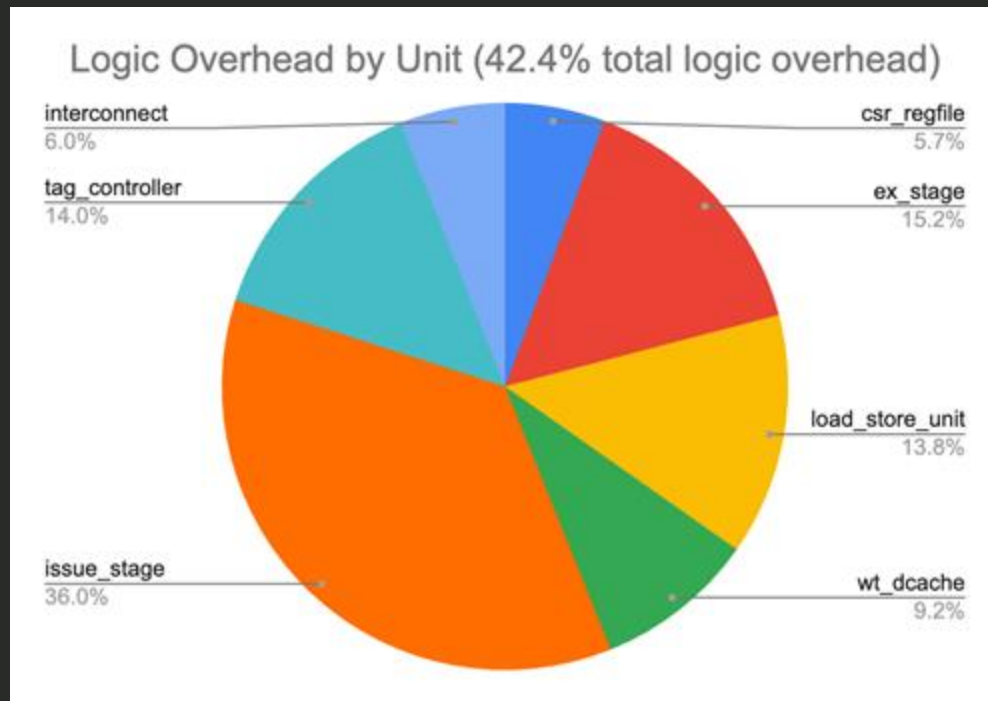
Consolidated Execute Functions

- Eliminated “Offset” operations
 - CFromPtr and CGet/SetOffset were relative to the **base** of the capability
 - This required decoding the base with special logic
 - These operations are very rare in pure capability programs, and they are not in the candidate RV64Y extension
- Consolidated YADD and YADDRW
 - YADDRW (pointer add) had a dedicated, optimized Execute circuit from cheri-cap-lib
 - ADDY (SetAddress) also had an optimized circuit
 - Moved to simply adding 64-bits, and then using a shared SetAddress function
- Also consolidate YLT (CheckSubset) and YBLD (BuildCap) checks due to tag clearing semantics in RV64Y
- 17% of Execute logic overhead eliminated



CVA6-CHERI Area Summary

- Currently 42.4% LUT (logic) overhead on Kintex 7 (Genesys 2 board).
 - Doesn't account for DSP units used for floating point
- 50.7% flip-flop overhead.
 - Doesn't account for block-ram memory, which constitutes caches
- Tested with Floating Point, Hypervisor, dual commit ports (not superscalar)
- Vector support and superscalar would reduce overhead for CHERI

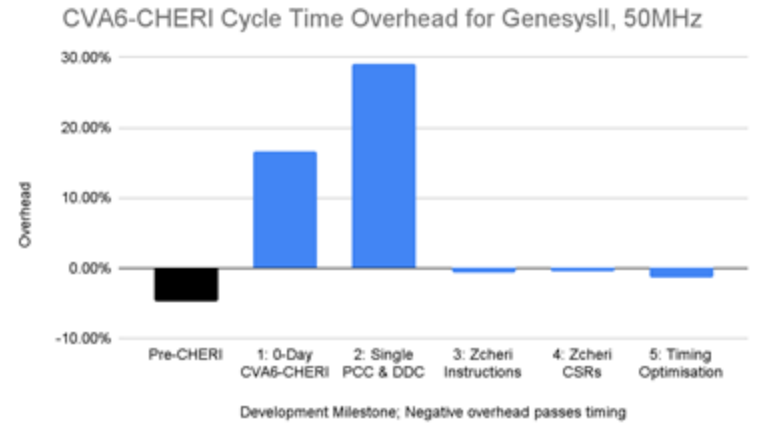
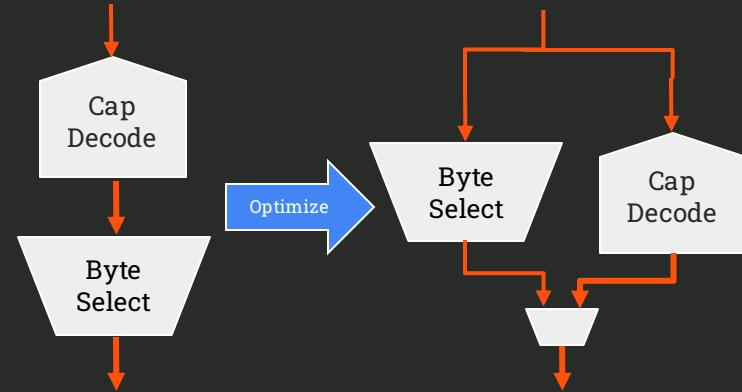


Timing Optimization



Capability Decode in Parallel to Data Selection

- The critical path for CVA6-CHERI became decoding capabilities into the register file
(cap_reg_t conversion function from cheri-cap-lib)
- The decode was followed by byte selection logic to move the loaded byte to the bottom of the destination register.
- As capability loads do not need byte selection
- We were able to decode capabilities into the register format concurrently with byte selection, muxing the desired result at the end.
- This enabled CVA6-CHERI to pass timing at 50MHz on the GenesysII board with Kintex 7 FPGA.
(Previous design achieved only 41MHz)



Software Support



Software Bringup on CVA6-CHERI

- We have booted **CheriBSD** on CVA6 Genesys 2 FPGA

Added code to setup HPM counters for
“libstatcounters” performance diagnosis

- Runs full CheriBSD test suite

Successfully enter userspace

- Validated spatial safety, temporal safety, and control-flow safety

- Also booted **CHERI-seL4 Microkit**

OpenSBI v1.5

Purecap OpenSBI

[illegible]

CPU identified with "Zcheri" support

```

...
Boot HART Base ISA      : rv64imafdc
Boot HART ISA Extensions : zihpm,zcheripurecap,zcherihybrid

```

Setting up HPM counters...

Control successfully handed over to CheriBSD

```

...
---<<BOOT>>---

```

```
...
CHERI pure-capability kernel with revocation support
```

CheriBSD fully capability aware

```
Physical memory chunk(s):
  0x80000000 - 0xbf7fffff, 1016 MB ( 260096 pages)
```

Tiny slice of DRAM (0xbf800000-0xbffffff)
reserved for tags

```
Logging in as root...
```

```
2025-08-22T11:10:01.110837+00:00 - login 41 - - login on console as root
# cheribsdtest-mt-c18n -a && cheribsdtest-purecap -a
TEST: bounds_global_static_uint8: Check bounds on global static uint8_t
PASS: bounds_global_static_uint8
TEST: bounds_global_uint8: Check bounds on global uint8_t
PASS: bounds_global_uint8
```

Run commands from UART terminal

```
...
SUMMARY: skipped 2, passed 300, expectedly failed 8
```

Purecap test success:
no unexpected failures

Skipped:

```
cheri_revoke_loadside: Xcheri revocation test not supported on Zcheri
Expected failures:
```

```
vm_shm_open_anon_unix_surprise: tag transfer succeeded
vm_cap_share_fd_kqueue: tag transfer
vm_cap_share_sigaction: tag transfer
cheri_revoke_shm_anon_hoard_unmapped: 'check_revoked(*map)' is FALSE!
cheri_revoke_shm_anon_hoard_closed: 'check_revoked(*map)' is FALSE!
SUMMARY: skipped 1, passed 295, expectedly failed 5
```

Successful tests for c18n support



Thank You!

Questions?

<https://github.com/Capabilities-Limited/cheri-cva6>

jonathan.woodruff@cl.cam.ac.uk
alexandre.joannou@cl.cam.ac.uk



CAPABILITIES
LIMITED