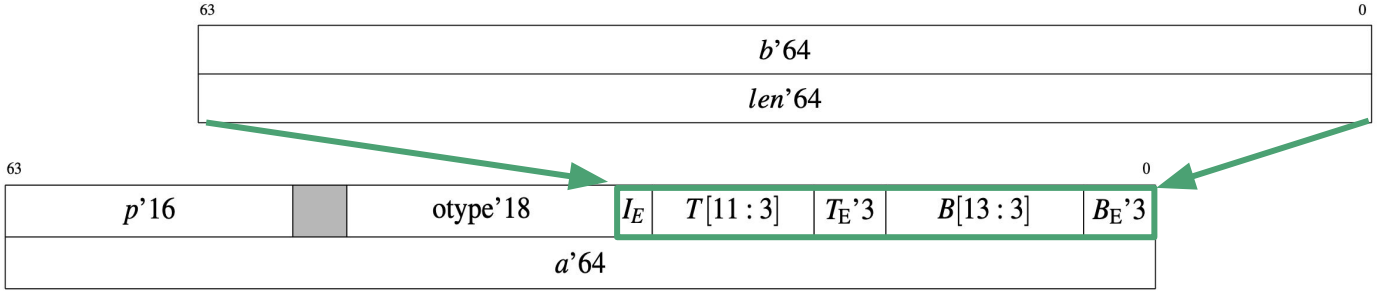
CHERI Concentrate in ACL2

Maxine Xin (with Carl Kwan & Bill Young)
Nov 2023

Introduction

- Capabilities
  - unforgeable tokens of authority that grant and describe access to a region of memory
- CHERI:Capabilities Hardware Enhanced RISC Instructions (Watson et al., 2020)
  - RISC ISA that supports capabilities
- CHERI Concentrate (Woodruff et al., 2019)
  - 256-bit raw CHERI capabilities (128-bit bounds)
  - Compress **128-bit** bounds to **27-bit**
  - Reduces L2 cache misses by 50% - 70%



Contribution

- Formalized & verified **CHERI concentrate properties** by utilizing GL
- Proved **conversion** between architectural capabilities (**uncompressed**) and memory capabilities (**compressed** with CHERI concentrate) correct
- Built the **mechanism to integrate compressed capabilities** into y86 ISA

Motivation

- Previous formalization
  - with HOL for **32-bit** system
- ISA v8 report on **64-bit** system:
  - General description of the concentrate algorithms
  - CHERI concentrate properties
- How to ensure CHERI concentrate implementation satisfies the properties?

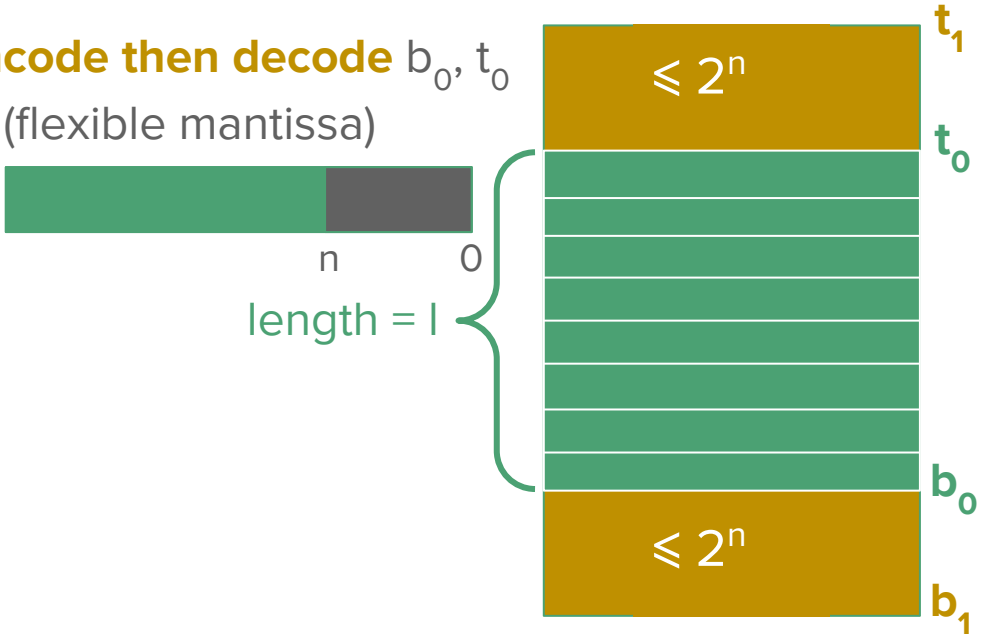
ACL2!!

Research Process

- **Model** CHERI concentrate encoding and decoding functions
- **Prove** properties about encoding-decoding conversion

CHERI Concentrate Properties

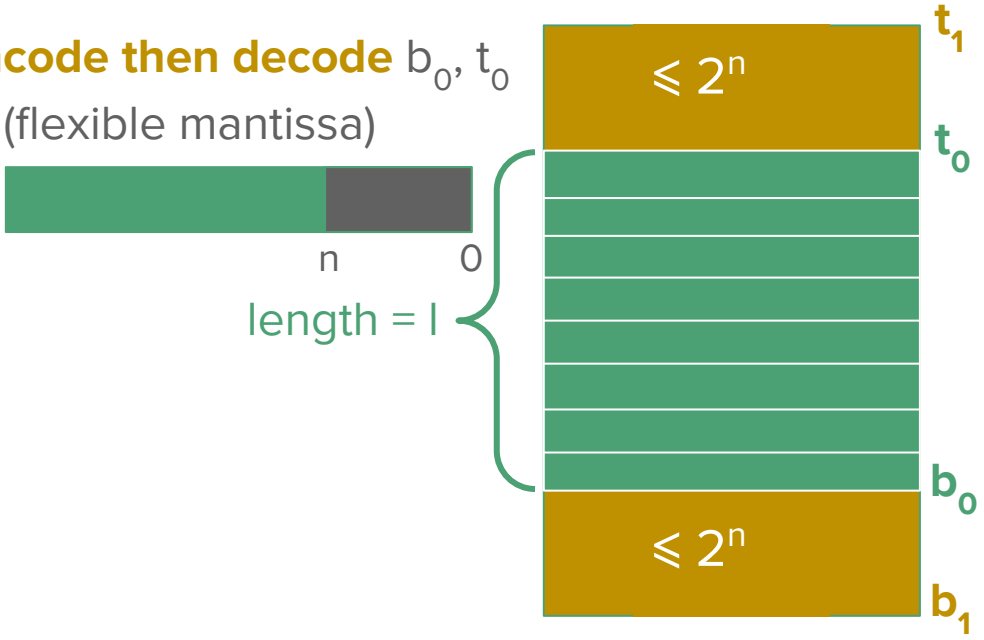
- ***b_0, t_0***: **raw** base and top; ***b_1, t_1***: **encode then decode** b_0, t_0
- *n*: number of bits in lost precision (flexible mantissa)
- *l*: length of accessible region



CHERI Concentrate Properties

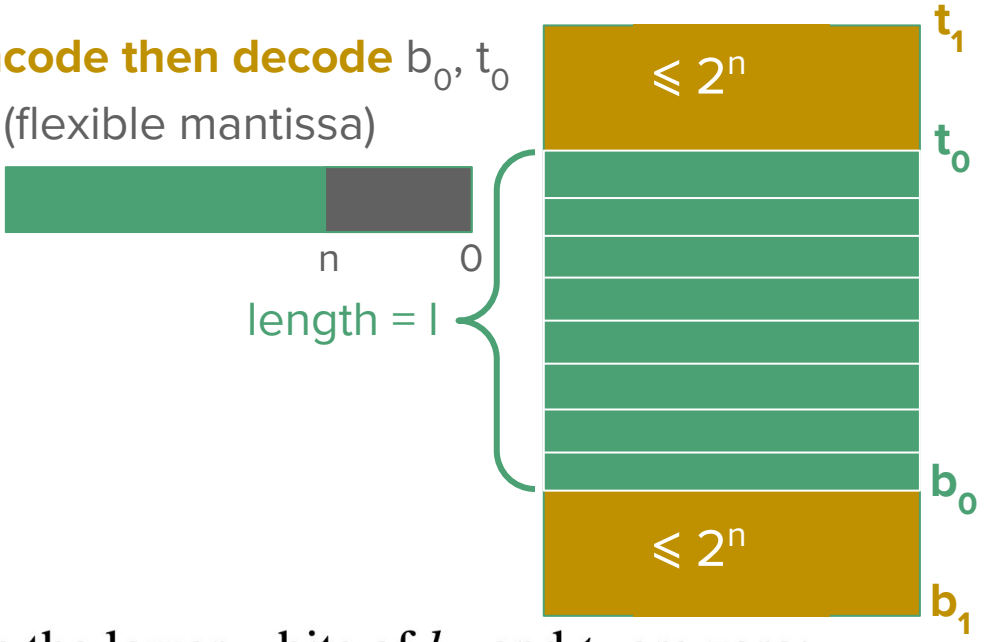
- ***b_0, t_0***: **raw** base and top; ***b_1, t_1***: **encode then decode** b_0, t_0
- *n*: number of bits in lost precision (flexible mantissa)
- *l*: length of accessible region

1. $b_0 \geq b_1$ for any b_0, t_0;
2. $b_0 - b_1 \leq 2^n$ for any b_0, t_0;
3. $t_0 \leq t_1$ for any b_0, t_0;
4. $t_1 - t_0 \leq 2^n$ for any b_0, t_0;



CHERI Concentrate Properties

- ***b_0, t_0***: **raw** base and top; ***b_1, t_1***: **encode then decode** b_0, t_0
- *n*: number of bits in lost precision (flexible mantissa)
- *l*: length of accessible region



1. $b_0 \geq b_1$ for any b_0, t_0;
2. $b_0 - b_1 \leq 2^n$ for any b_0, t_0;
3. $t_0 \leq t_1$ for any b_0, t_0;
4. $t_1 - t_0 \leq 2^n$ for any b_0, t_0;
5. $b_0 = b_1$ when $\ell < 2^{12}$ or when the lower n bits of b_0 and t_0 are zero;
6. $t_0 = t_1$ when $\ell < 2^{12}$ or when the lower n bits of b_0 and t_0 are zero.

Technical challenges

- **Lack of implementation details** in CHERI report
  - hard to implement
- Long sequence of **bit manipulation** computations
  - hard to prove properties

GL (Swords & Davis, 2011), framework for proving **finitely**-bounded ACL2 theorems by **bit-blasting**

Technical challenges - Solution

- **Lack of implementation details** in CHERI report

- Long sequence of **bit manipulation** computations

GL (Swords & Davis, 2011), framework for proving **finitely**-bounded ACL2 theorems by **bit-blasting**

Example of GL

- $b_0 = b_1$ when $\ell < 2^{12}$
- $t_0 = t_1$ when $\ell < 2^{12}$

```
(def-gl-thm decode-encode-equal-small-seg
    :hyp   (and (valid-addr-p addr base len)
                (valid-b-l-p base len)
                (< len (expt 2 12)))
    :concl (equal (decode-compression (encode-compression len base) addr)
                  (bounds (+ len base) base))
    :g-bindings `((base ,(gl::g-int 0 3 65))
                  (len ,(gl::g-int 1 3 66))
                  (addr ,(gl::g-int 2 3 65))))
```

Example of GL - case-split

Not enough resources → 1 min!

```
(def-gl-param-thm decode-encode-b-bound-len>2^12
    :hyp   (and (valid-addr-p addr base len)
                (valid-b-l-p base len)
                (<= (expt 2 *tw*) len))
    :concl (<= (bounds->base (decode-compression (encode-compression len base) addr))
               base)
    :param-bindings
    `((((low 12) (high 16)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 16) (high 20)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 20) (high 24)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 24) (high 28)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 28) (high 32)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 32) (high 36)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 36) (high 40)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 40) (high 44)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 44) (high 48)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 48) (high 52)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 52) (high 56)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 56) (high 60)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 60) (high 64)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
      (((low 64) (high 65)) ,(gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65)))))
    :param-hyp (and (<= (expt 2 low) len) (< len (expt 2 high)))
    :cov-bindings (gl::auto-bindings (:mix (:nat base 65) (:nat len 65) (:nat addr 65))))
```

Ongoing / Future Work

- Develop CHERI ISA with y86
- Prove CHERI ISA executions correct in y86

Thanks for listening!
Questions?