

# **Rigorous Hardware-Software Codesign for Secure Compartmentalization on Embedded Devices**

Applying Formal Methods to CHERIoT

Murali Vijayaraghavan (muralivi@google.com)

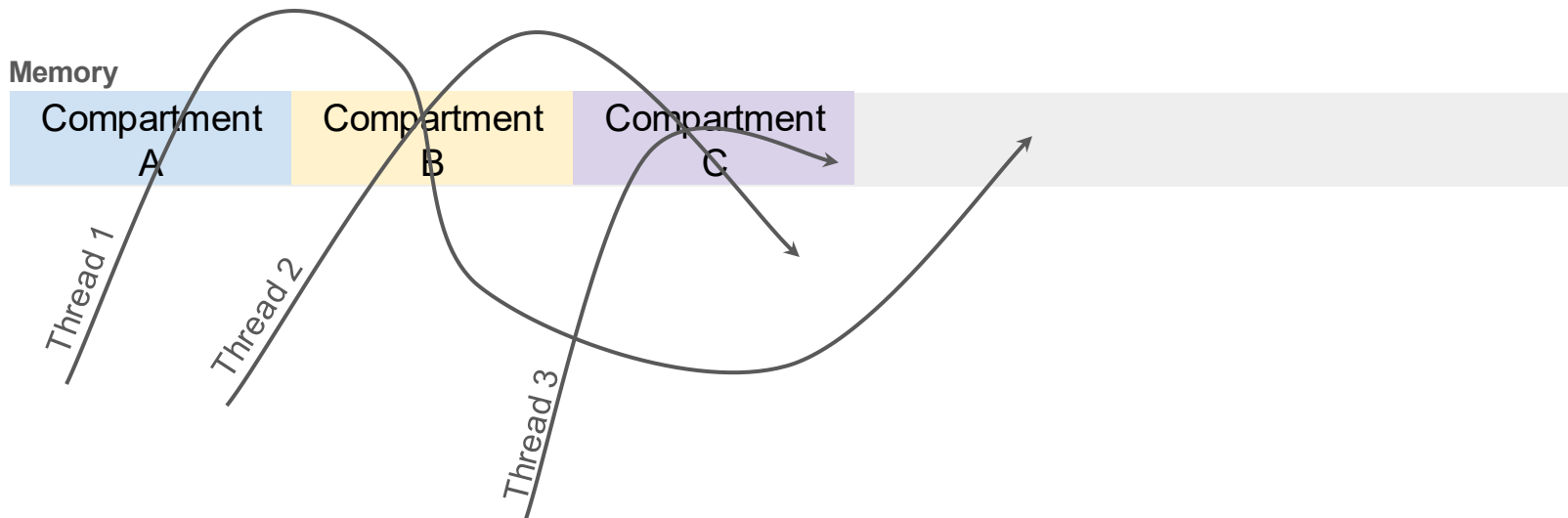
# Talk outline

- Formalization of CHERIoT's multithreaded user execution model in Rocq
- Formalization of CHERIoT ISA in Guru, a DSL in Rocq
- Future: Using formal methods and CHERI to rethink TEE's

# CHERIoT's Multithreaded user execution model

# CHERIoT abstract machine

- Compartments: provide spatial memory isolation/security domains for code and data
- Threads: execute concurrently
  - Each thread “threads” through multiple compartments
  - Context switch between threads is semantically a no-op



# CHERIoT threads

- A thread owns a private stack and a private register context
- Threading through compartments is via cross-compartment function calls
  - Arguments and returns are passed through the stack
    - They can be capabilities passed between compartments
  - Semantics: Atomically
    - switch to callee security domain
    - restrict stack access for callee
    - zero callee stack region on call and return
- Exceptions can occur on thread execution
  - Atomically invoke the code's (i.e. compartment's) exception handler if it exists
  - Otherwise return to caller compartment with error code

# CHERIoT RTOS

A microkernel that provides the semantic abstraction for CHERIoT

- Atomic switching between threads on a single hardware core
- Atomic cross-compartment calls and returns
- Atomic exception handler invocation

The previous constraints on these semantically atomic operations formally define the specification of the RTOS

See [github.com/Cherified/cheriot-abstract-spec](https://github.com/Cherified/cheriot-abstract-spec)

# Issues in current CHERIoT RTOS

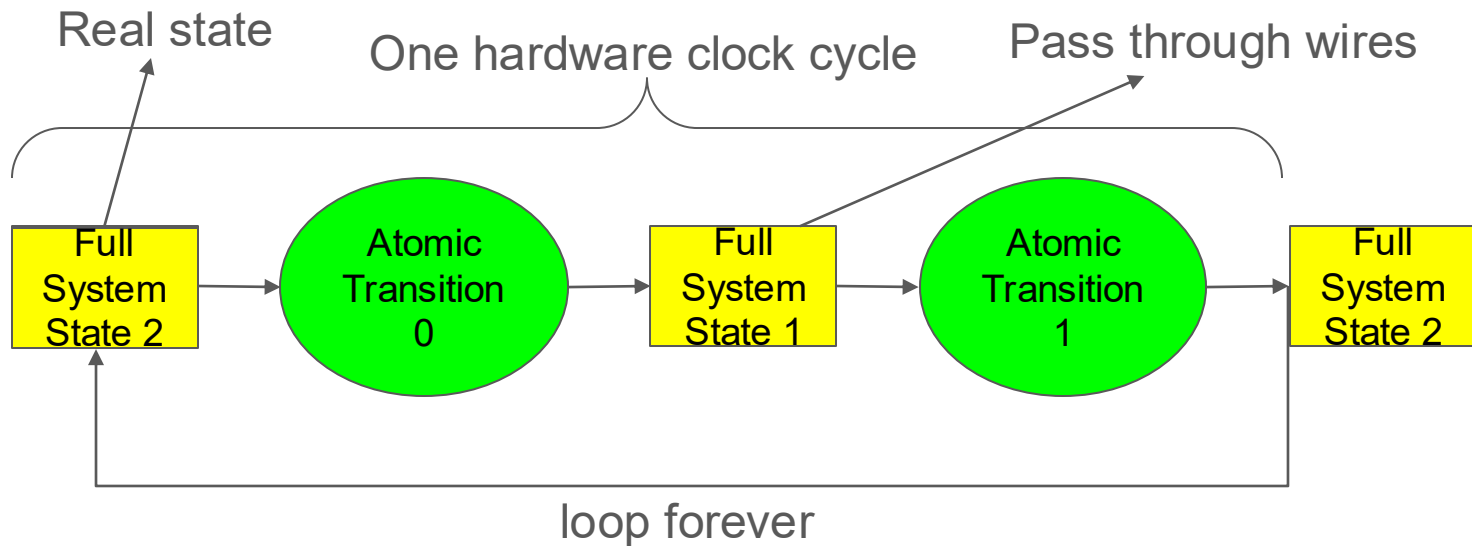
- Tries to turn off (timer) interrupts appropriately to enforce atomicity in single core
  - Known issues where it's conservative at times and unsound at other times  
(<https://github.com/CHERIoT-Platform/cheriot-rtos/issues/47>, <https://github.com/CHERIoT-Platform/cheriot-rtos/issues/372>)
- Does not support multicore execution
  - Proposal to fix using the formal model: <https://github.com/CHERIoT-Platform/cheriot-rtos/issues/589>

## WIP

- Fix the CHERIoT RTOS
- Formally verify the correctness of RTOS w.r.t. formal execution model

# CHERIoT ISA and processor implementation

# Guru: DSL in Rocq for HW specification, proof and synthesis

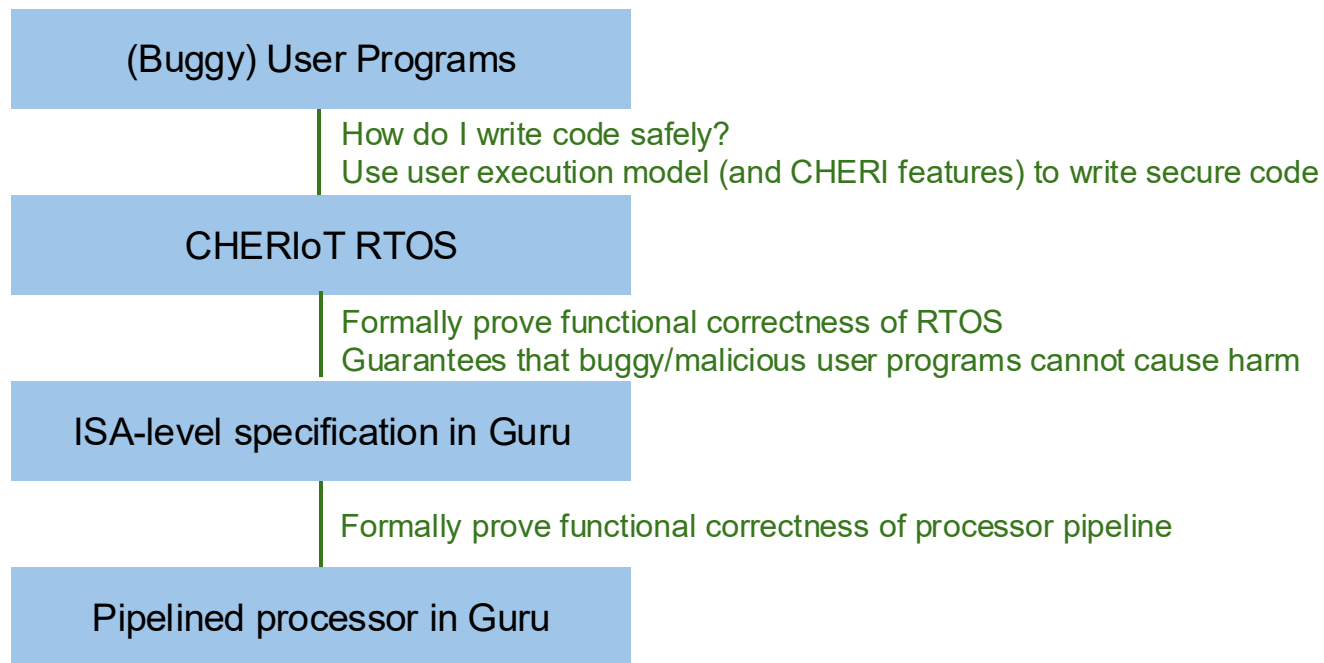


- Atomic transition allows local reasoning without knowing what else happens in the system
  - E.g.: a register write by the execute stage and a register read by decode stage can be reasoned independently while generating bypass logic automatically!
- Guru provides tactics support for proof of equivalence using a simulation relation between two systems

# CHERIoT in Guru







- ISA Specification <https://github.com/Cherified/Cheriot>
- WIP Implementation
  - 3+ stage Pipeline similar to Ibex
    - Stall on load-use
    - Load filters for revoked capabilities (extra cycle for capability loads)

# Towards a CHERIoT hardware-software verified stack



# **WIP: Rethinking TEE's with formal methods and CHERI**

# Logical specification of TEEs

Properties	With formally verified CHERIoT HW/SW stack
Different processes cannot access each other's data (or code)	
Platform (OS, hardware) cannot access user data (or code)	Data  Code 
Processes cannot access platform's data (or code)	
Support intentional communication between processes (and platform)	
Attestation that the correct process runs on the correct platform	

## Preventing Platform from accessing user data and code

- The loader currently loads an unencrypted application binary
  - Instead, it should be modified to load an application binary encrypted by user/application-specific key that is inaccessible by the platform
  - Any updates to the binary or key should be authenticated

## Attest that the correct process runs on the correct platform

- Instead of attesting currently running process+platform pair, use authenticated updates for applications and OS/platform

**These update services (and loader) must be formally verified!**

# What about encryption?

- Any data that goes out of the chip must be encrypted
  - For instance, pin snooping can exfiltrate data going from processor to DRAM
- But within a chip, encryption is
  - **insufficient:** application data can be overwritten by a malicious platform, and
  - **unnecessary:** if confidentiality is formally proven, no need to encrypt data

# Conclusion: Towards a low-cost TEE using CHERI and formal methods

