CHERI-Muntjac: an efficient, secure, application-class core

Yuecheng Wang, Jonathan Woodruff, Peter Rugg, Alexandre Joannou, Samuel W. Stark and Simon W. Moore

Department of Computer Science and Technology, University of Cambridge

Abstract

The research interest in CHERI has been increasing over the past decade. Various research projects have been conducted on commercial Morello and CHERI-Toooba [1] over the years, a high parameterisable core written in BlueSpec SystemVerilog. There is an ongoing research need of more diverse CHERI systems to evaluate and utilize CHERI more extensively. Moreover, there is a need for more commercial grade cores written in SystemVerilog to aim commercial adoption. While CHERIoT from Microsoft and SCI Semi demonstrates CHERI for microcontrollers, we demonstrate CHERI for a commercial application-class scalar core by extending Muntjac from lowRISC.

Introduction

Muntjac is an open-source collection of components which can be used to build a multi-core, Linux-capable system-on-chip [2]. The main components it includes are a 64-bit RISC-V core, a MESI capable cache system, and TileLink interconnect peripherals.
Muntjac has a few features that make it an interesting candidate for a CHERI extension. It has a simple 5-stage in-order pipeline which is easy to understand, nevertheless it implements the required extensions to boot Linux; it also has an interesting data cache implementation to support faster write-back and refill; it is written in SystemVerilog which is favored by industry, compared to Bluespec SystemVerilog, the language of most current open-sourced CHERI processors. In addition, Muntjac has a small area footprint on FPGA which allows evaluation of CHERI systems with higher concurrency than previously possible.

CHERI extension

Our CHERI-extended Muntjac implements the CHERI V9 specification [3]. It is a work in progress, and it has been tested using TestRIG [4] throughout development. CHERI-Muntjac is currently in the final stages of booting CheriFreeRTOS in simulation, with a delivery target of booting CheriBSD on FPGA by the time this work is presented.
This section will briefly describe the changes we made to the Muntjac SoC to support CHERI.

Extending the pipeline

The majority of the changes to support CHERI were in the pipeline of the core. When adding CHERI, it is important to decide how the capability should flow through the pipeline, as the capability needs to be converted between several formats across the pipeline units. The Bluespec CHERI-RISC-V implementations used a two-stage decompression: one for the register file and one for the execute stage.
To avoid complicating Muntjac's elegant pipeline, we perform a single capability decompression into the register file with all fields necessary for execution. If area usage becomes a concern, we can revisit this decision in the future.
We use micro-architectural primitives from the formally verified cheri-cap-lib library [5] to support secure capability compression and manipulation. We plan to re-design and re-implement the cheri-cap-lib library [5] in SystemVerilog once we have a working CHERI-Muntjac.

Preserving the cache structure

Muntjac's cache system has some unique features which required special attention when modifying it to support CHERI. It has a 64-bit interface to the L1 cache but uses the interleaving scheme in Figure 1 to provide a wider 128-bit interface at the LLC [2]. The CHERI extension requires reading and writing 128-bit

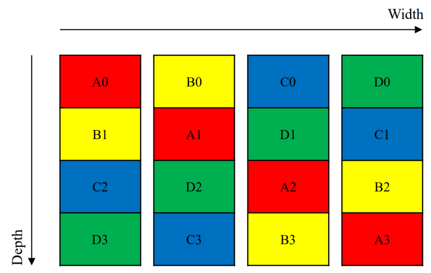

Figure 1: *Muntjac data cache interleaving [2]*

capability words atomically. Other CHERI RISC-V cores choose to double the granule size of their RAM to support this, which changes the cache structure. Naively doubling the granule size on Muntjac would require doubling the LLC interface width to 256-bit, or disabling cache interleaving, which are both non-optimal.
To preserve Muntjac's cache interleaving logic while keeping the width of the bus and cache constant, we re-purposed this logic to support the wider, 128-bit pipeline accesses. As shown in Figure 1 each block represents a 64-bit-wide data bank, and a single access can span multiple banks, which are also used in multiple ways. For instance, to write 128 bits of data, the data will be split into A0 and A1 and written separately; on read, we combine the output of A0 and A1 into one 128-bit word.
We are currently experimenting with this feature; it is functioning correctly, and we are working on improving the interleaving algorithm to increase its performance for 128-bit data access.

SystemVerilog Tag Controller

CHERI requires a single tag bit for every capability word in memory, and a tag controller is required to manage these tags [6] . Instead of using the BlueSpec Tag controller [6], we chose to implement a CHERI tag controller in SystemVerilog as a component connected to Muntjac's TileLink interconnect. Previous AXI CHERI tag controllers have used the AXI .user field to communicate tags, but the TileLink standard lacks a user-defined field. For a general solution, we extended the TileLink protocol with a user-defined field to pass the CHERI tag bit.
A simple version of this tag controller is functioning correctly now, and we plan to implement the hierarchical tag table optimization after CHERI-Muntjac boots on CheriBSD. We also plan to use this design to evaluate several novel tag optimization strategies.

Verifying CHERI extension

Our delivery targets include the following phases:

1. **Testing against TestRIG**. We have used TestRIG extensively throughout the development process. We now pass all CHERI module tests, and we are running increasingly deep sequences of the CapRandom test-suite to identify subtle divergences from the model.
2. **Booting CheriFreeRTOS in simulation**. We are at the stage of trying to boot CheriFreeRTOS (CHERI-extended FreeRTOS) on Muntjac in simulation. As debugging an OS boot on FPGA is time consuming, running CheriFreeRTOS in simulation can help us identify many problems early before running on FPGA.
3. **Booting CheriBSD on FPGA**. Booting CheriBSD on Muntjac running on FPGA is our delivery target. We plan to re-run some benchmarks to compare it with other CHERI processors we have.

TestRIG-driven development

TestRIG has been the main testing framework for verifying our CHERI-Muntjac extension against the CHERI-Sail model. TestRIG guides the implementation of basic features and also helps us identify rarely occurring bugs. The CHERI specification has some subtle details which can be easily missed or misinterpreted; TestRIG identified rare bugs, and its shrinking feature was particularly useful for identifying the root cause of the misbehavior.

Community Contributions

We plan to release documentation describing our CHERI-Muntjac implementation along with the CHERI-Muntjac source codes, so that it can be used as a guide for extending any Linux/FreeBSD-capable RISC-V system with CHERI.
CHERI-Muntjac fills an important gap in the community of CHERI RISC-V processors; current open-source application class designs are not suitable for industrial adoption, as they are written in Bluespec and not focused on efficiency. CHERI-Muntjac can also enable larger CHERI system research. Because Muntjac is very small, we can fit more cores on our DE10-Pro FPGA, allowing research into highly concurrent CHERI systems.

References

[1] P. Rugg et al. *CHERI-cap-lib*. https://github.com/CTSRD-CHERI/Toooba.

[2] Xuan Guo et al. *Muntjac multicore RV64 processor*. Tech. rep. University of Cambridge, 2022.

[3] Robert N. M. Watson et al. *Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 9)*. Tech. rep. University of Cambridge, 2023.

[4] A. Joannou et al. “Randomized testing of RISC-V CPUs using direct instruction injection”. In: *IEEE Design & Test* (2023).

[5] J. Woodruff et al. *CHERI-cap-lib*. https://github.com/CTSRD-CHERI/cheri-cap-lib.

[6] A. Joannou et al. *TagController*. https://github.com/CTSRD-CHERI/TagController.