Review of CHERIoT: Complete Memory Safety for Embedded Devices

Konstantinos Koupepas (kk1219), Naman Sharma (nns20), Szymon Kubica (sk4520)

November 23, 2023

Due to the recent growth in IoT market and connectivity, large software stacks consisting of components provided by multiple vendors and parties are used on embedded devices. Because of mutual distrust, memory safety and isolation guarantees are required. This needs to be achieved in the context of real-time requirements often imposed on embedded systems. The authors claim that existing solutions for desktop operating systems violate at least one of the three key requirements: real-time, code size, power and compatibility. In the reviewed paper [1], a solution is proposed by adapting the CHERI capability architecture to the needs of memory-safe embedded systems.

CHERI and Memory Safety in the Context of RTOS

CHERI ISA aims to ensure memory safety by augmenting pointers with additional metadata (creating fat pointers). Pointers together with the metadata are called *capabilties*. They control the bounds of access and permissions (read, write, execute). This mechanism supports the principle of least privilege up to the level of individual fields of an object. CHERIoT was designed with the constraints imposed by real time operating system (RTOS) in mind. In this context a system is real-time if the latency of all operations is bounded and can be reasoned about. This meant that the solution proposed in the paper could not include hardware structures with unpredictable latency.

Paper Contributions

The main contribution is an embedded-systems architecture designed in conjunction with an RTOS offering complete and deterministic memory safety. This is achieved by ensuring architecture-level separation between mutually distrusting compartments. The paper proposes architectural extensions and hardware acceleration for temporal memory safety of cross compartment references and the shared heap allocator of the RTOS. The performance of the solution is considered with different design trade-offs to support a variety of use-cases. The paper evaluates the area and power of hardware assists for a production-quality core.

Modifications to CHERI Introduced in CHERIoT

The design of CHERIoT was intended for embedded devices with memory constraints, hence modifications to the CHERI ISA were proposed. These changes removed some of the expressiveness of CHERI which was redundant in an embedded device and minimised the memory footprint of the capability encoding.

Firstly, to minimize the number of bits used by CHERI capabilities, unused capabilities were removed. Further, combinations of capabilities that were mutually exclusive in the proposed system design (e.g. store and execute) were removed from the ISA. Another innovative measure to reduce bit usage was permission compression of capa-

bilities. In the proposed design only certain combinations of permissions are meaningful, hence the authors use the interdependence between the valid combinations to encode all possibilities using a total of 6 bits.

Then, new capabilities were added for limiting off-stack storage and allowing read-only sharing. A sealed capability in CHERI is immutable (doesn't allow any loads or stores), and a sealed entry is an executable capability that can be sealed and only becomes unsealed by jumping to it. To prevent functions from arbitrarily enabling and disabling interrupts, CHERIoT implements three new types of sealed entry capabilities, allowing for changing the whether the code running with a given capability runs with interrupts disabled.

Despite CHERI ISA not providing any mechanism for temporal safety, its deterministic spatial safety allows for temporal safety schemes that are also deterministic. Existing solutions utilise mechanisms unavailable for embedded systems and therefore, a novel scheme had to be designed. Firstly, heap revocation bits were introduced for marking memory as inaccessible, which introduced a 1.56% memory overhead for heap allocations. Freed memory is quarantined until there are no more outstanding references and then it is cleared and made available for reuse.

To keep the invariant that no capabilities pointing to freed memory can be loaded into registers,a hardware load filter was implemented, that checks for the revocation bit discussed earlier. They claim that this does not introduce any pipeline stalls thanks to the one-cycle load-to-use delay. Using purely software, sweeping revocation introduces significant latency and uses up CPU cycles. So, they also introduced a background pipelined revoker that works when the main pipeline is doing register-to-register operations, which improves the latency of revoking memory. Also, since it is its own circuit, it does not use up CPU resources. To deal with a potential race condition, they made the main pipeline visible to the background revoker to make sure it does not revoke words in the middle of writing.

Authors claim that thanks to the new capabilities along with the temporal memory safety discussed earlier they have achieved the first embedded system that enforces deterministic spatial and temporal memory safety across compartment boundaries. They contrast their solution with the existing temporal safety work on CHERI.

Evidence of the Contributions

The proposed solution is evaluated against three key requirements: area of the processing unit, power consumption, and performance. An ideal implementation for a memory safe embedded system should have similar space, power and performance characteristics otherwise the end user will have to trade-off one of these conventionally-desired characteristics for memory safety – this will only reduce the ubiquity of CHERIoT cores.

To compare space consumption, the authors count gates or gate equivalents for production versions of the CHERIoT-Ibex cores (with varying optimizations) against the current industry-standard RISC-V Physical Memory Protection unit. This is a representative comparison since these cores are ultimately the ones available to the end user, hence will reflect the user's experience rather than research-only variants.

CHERIoT has between 4.5% to 10% area overhead relative to the baseline RISC-V Physical Memory Protection unit, depending on the configuration of CHERIoT being used, this is a relatively minor space difference unlikely to be critical in many applications. In those applications where space is of utmost importance, more so than performance, then one can enable software features to mimic hardware components (i.e. a software-based background revoker) to reduce area consumption of CHERIoT to a maximum of 4.5% above the baseline RISC-V PMP unit.

The same CHERIoT-Ibex and RISC-V PMP cores used for space evaluation are used for power evaluation ensuring consistency, however a caveat for power evaluation metrics is that they are based on power-estimation modelling software rather than silicon-based, hardware-derived data. The authors do acknowledge this limitation and understand that the power model factors gate count heavily into its predictions, which will naturally impact CHERIoT's power characteristics given its higher gate count. The paper states that, in reality, PMP's comparators are engaged on every load/store instruction whereas the CHERI versions do not use comparators for this purpose rather a hardware background revoker which is idle with minimal power consumption in non-allocation-heavy phases of computation. The authors argue that this difference leads to similar power footprints. Ultimately, an ideal evaluation would have profiled physical cores to reduce any uncertainty, however research is often limited by financial and time constraints hence we accept this minor uncertainty, given the power consumption will be in similar orders of magnitude.

To compare the relative performance of the varying processors, the CoreMark and Allocation benchmarks are used, in addition to an example IoT application. These offer representative performance metrics, however compiling the benchmark programs to CHERIoT instruction sets results in some performance loss, due to two compiler bugs acknowledged by the authors. The paper handles this by reasoning about CHERIoT in terms of worst case performance on these benchmarks, equally the authors acknowledge that even in a supposed best case, adding capabilities would have some overhead. Whichever physical design of CHERIoT is used, the performance is within a small constant factor slower rather than an order of magnitude. CHERIoT's performance scales linearly with physical memory size however, this is permissible given upper limits due to size constraints of embedded systems.

Conclusion

The paper is not difficult to follow since the authors introduce CHERI and RTOS at the beginning which contextualises previous, relevant work and motivates the problem. The following sections build on each other logically and present convincing arguments for the proposed solution, which are backed by appropriate evaluation. There are limitations with the evaluation, for example using software models or performance implications of compiler bugs, however these are clearly mentioned by the authors and factored into the overall discussion. Ultimately most researchers are limited by money and time. Given the constraints, we think this paper offers novel solutions to pertinent problems and addresses their feasibility in widespread adoption, whilst leaving scope for futher research should more money or time become available.

Work Distribution

After reading the paper we had a discussion to comment on the things that we have learned. After that we have divided the work up into sections. The primary authors of the sections of the report are listed below.

Introduction - Szymon
CHERI and Memory Safety in the Context of RTOS - Szymon
Paper Contributions - Szymon
Modifications to CHERI Introduced in CHERIoT - Konstantinos
Evidence of the Contributions - Naman
Conclusion - Naman

After the first drafts of the sections were written we had another meeting to refactor the text and ensure that it is within the 3 page limit.

Use of AI tools

We experimented with using AI tools however the quality of the output was not great - very verbose, minimal insight. Hence, we did not use any AI content.

REFERENCES

[1] Saar Amar, David Chisnall, Tony Chen, Nathaniel Wesley Filardo, Ben Laurie, Kunyan Liu, Robert Norton, Simon W. Moore, Yucong Tao, Robert N. M. Watson, and Hongyan Xia. 2023. CHERIoT: Complete Memory Safety for Embedded Devices. In 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23), October 28–November 01, 2023, Toronto, ON, Canada. ACM, New York, NY, USA, 13 pages. https://cheriot.org/papers/2023-micro-cheriot-uarch.pdf