arm Morello Program

# Arm Morello Evaluation Platform - Validating CHERI-based Security in a High-performance System

Richard Grisenthwaite

SVP Chief Architect and Fellow, Arm

Richard.Grisenthwaite@arm.com

UK Research and Innovation

# Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237 (“CTSRD”), with additional support from FA8750-11-C-0249 (“MRC2”), HR0011-18-C-0016 (“ECATS”), and FA8650-18-C-7809 (“CIFV”) as part of the DARPA CRASH, MRC, and SSITH research programs. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694.

We also acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), the ERC ELVER Advanced Grant (789108), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, Microsoft Research Cambridge, Arm Limited, Google, Google DeepMind, HP Enterprise, and the Gates Cambridge Trust.

University of Cambridge, SRI International, etc  Contributors on CHERI: Robert N. M. Watson, Simon W. Moore, Peter Sewell, Peter G. Neumann, Hesham Almatary, Jonathan Anderson, Alasdair Armstrong, Peter Blandford-Baker, Rosie Baish, John Baldwin, Hadrien Barrel, Thomas Bauereiss, Ruslan Bukin, Brian Campbell, David Chisnall, Jessica Clarke, Nirav Dave, Brooks Davis, Lawrence Esswood, Nathaniel W. Filardo, Franz Fuchs, Dapeng Gao, Khilan Gudka, Brett Gutstein, Alexandre Joannou, Mark Johnston, Robert Kovacsics, Ben Laurie, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi, Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, George Neville-Neil, Kyndylan Nienhuis, Robert Norton-Wright, Philip Paeps, Lucian Paul-Trifu, Allison Randal, Ivan Ribeiro, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg, Hassen Saidi, Thomas Sewell, Stacey Son, Ian Stark, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Konrad Witaszczyk, Jonathan Woodruff, Hongyan Xia, and Bjoern A. Zeeb

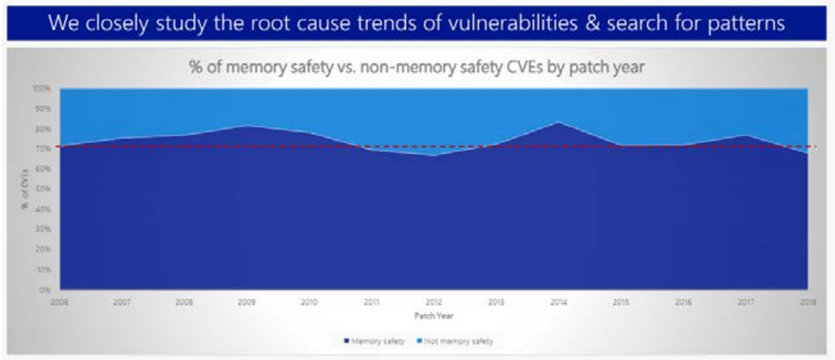
© Copyright 2022 Arm Limited

## Security is the greatest challenge computing needs to address to meet its full potential

© Copyright 2022 Arm Limited

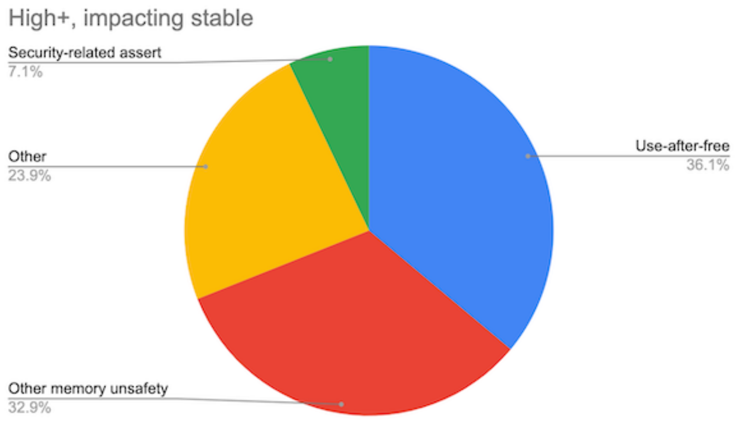
# Memory (Un)safety issues remain major source of CVEs

- Matt Miller (BlueHat 2019) : Microsoft around 70% of CVEs are memory unsafety issues
  - #1 Heap out-of-bounds
  - #2 use-after-free
  - #3 type confusion
  - #4 uninitialized use



- Chromium reports similar issues:
  - [Memory safety - The Chromium Projects]()

  "70% of our serious security bugs are memory safety problems."



- Been around for a very long time
  - Morris Worm 1988 usually credited as the first buffer overflow attack on the internet
  - C/C++ is not going away any time soon in the world's software

© Copyright 2022 Arm Limited

## CHERI architecture in one slide

- CPU architecture adds 128-bit “capabilities” in the register file plus a tag bit
  - Capability contains the address, bounds information, permission information etc
  - The tag bit is metadata that distinguishes a capability from normal data
    - The tag bit prevents “forging” of a capability
    - This functionality gives strong provenance of capabilities
  - Architecture has the ability to “seal” capabilities as well as part of compartmentalisation
- Loads/stores using capabilities as addresses are checked to be legal
  - Within address range and matching the supplied permissions
- Data processing on capabilities has rules to limit operations
  - Bounds cannot be arbitrarily increased, permissions cannot be relaxed etc
- PC is converted to a capability called the PCC to place bounds on the PC
  - Direct Branches will be within the PCC;
  - indirect branches (including returns) can change PCC
- Capability is used in place of a normal pointer in some or all situations
  - Exactly how when this happens is part of the software usage case

© Copyright 2022 Arm Limited

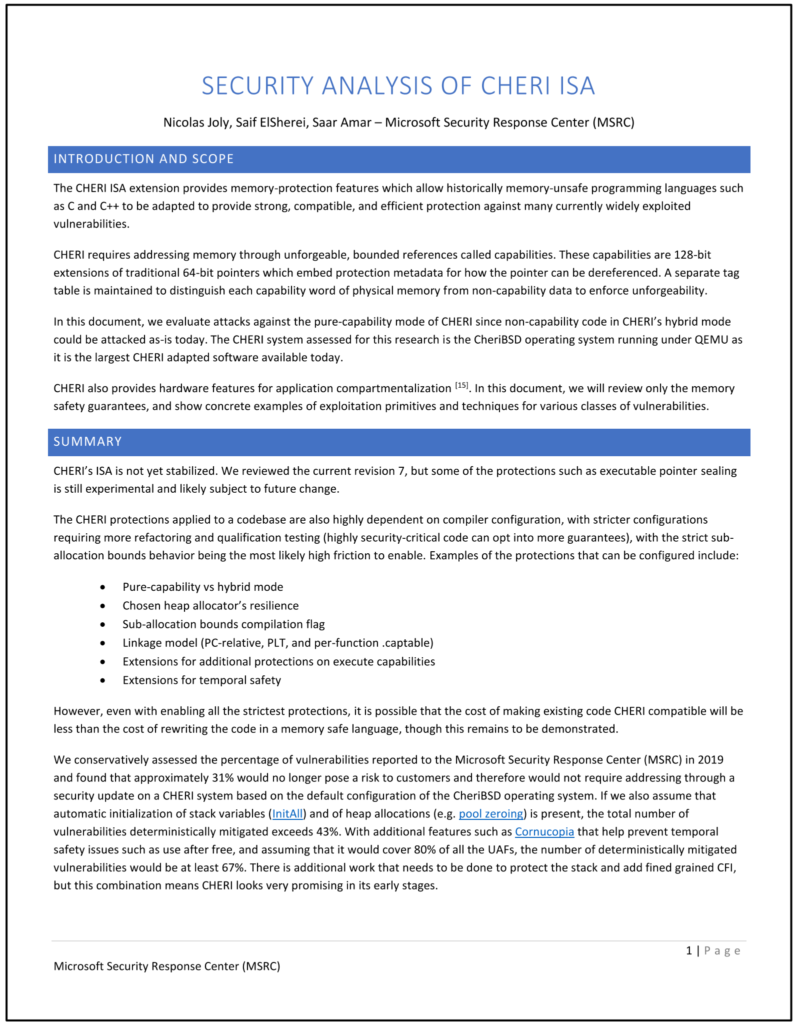
## Two key applications of the CHERI primitives

1. **Efficient, fine-grained memory protection for C/C++**
   - Strong source-level compatibility, but requires recompilation and minor source-code changes
   - Deterministic and secret-free referential, spatial, and temporal memory safety
   - Retrospective studies estimate ⅔ of memory-safety vulnerabilities mitigated
   - Generally modest overhead (0%-5%, some pointer-dense workloads higher)

2. **Scalable software compartmentalization**
   - Multiple software operational models from objects to processes
   - Increases exploit chain length: Attackers must find and exploit more vulnerabilities
   - Orders-of-magnitude performance improvement over MMU-based techniques (<90% reduction in IPC overhead in early FPGA-based benchmarks)

© Copyright 2022 Arm Limited

# Microsoft security analysis of CHERI C/C++



SECURITY ANALYSIS OF CHERI ISA

Nicolas Joly, Saif ElSherei, Saar Amar – Microsoft Security Response Center (MSRC)

INTRODUCTION AND SCOPE

The CHERI ISA extension provides memory-protection features which allow historically memory-unsafe programming languages such as C and C++ to be adapted to provide strong, compatible, and efficient protection against many currently widely exploited vulnerabilities.

CHERI requires addressing memory through unforgeable, bounded references called capabilities. These capabilities are 128-bit extensions of traditional 64-bit pointers which embed protection metadata for how the pointer can be dereferenced. A separate tag table is maintained to distinguish each capability word of physical memory from non-capability data to enforce unforgeability.

In this document, we evaluate attacks against the pure-capability mode of CHERI since non-capability code in CHERI's hybrid mode could be attacked as-is today. The CHERI system assessed for this research is the CheriBSD operating system running under QEMU as it is the largest CHERI adapted software available today.

CHERI also provides hardware features for application compartmentalization [15]. In this document, we will review only the memory safety guarantees, and show concrete examples of exploitation primitives and techniques for various classes of vulnerabilities.

SUMMARY

CHERI's ISA is not yet stabilized. We reviewed the current revision 7, but some of the protections such as executable pointer sealing is still experimental and likely subject to future change.

The CHERI protections applied to a codebase are also highly dependent on compiler configuration, with stricter configurations requiring more refactoring and qualification testing (highly security-critical code can opt into more guarantees), with the strict sub-allocation bounds behavior being the most likely high friction to enable. Examples of the protections that can be configured include:

- Pure-capability vs hybrid mode
- Chosen heap allocator's resilience
- Sub-allocation bounds compilation flag
- Linkage model (PC-relative, PLT, and per-function .captable)
- Extensions for additional protections on execute capabilities
- Extensions for temporal safety

However, even with enabling all the strictest protections, it is possible that the cost of making existing code CHERI compatible will be less than the cost of rewriting the code in a memory safe language, though this remains to be demonstrated.

We conservatively assessed the percentage of vulnerabilities reported to the Microsoft Security Response Center (MSRC) in 2019 and found that approximately 31% would no longer pose a risk to customers and therefore would not require addressing through a security update on a CHERI system based on the default configuration of the CheriBSD operating system. If we also assume that automatic initialization of stack variables (InitAll) and of heap allocations (e.g. pool zeroing) is present, the total number of vulnerabilities deterministically mitigated exceeds 43%. With additional features such as Cornucopia that help prevent temporal safety issues such as use after free, and assuming that it would cover 80% of all the UAFs, the number of deterministically mitigated vulnerabilities would be at least 67%. There is additional work that needs to be done to protect the stack and add fined grained CFI, but this combination means CHERI looks very promising in its early stages.

1 | Page

Microsoft Security Response Center (MSRC)

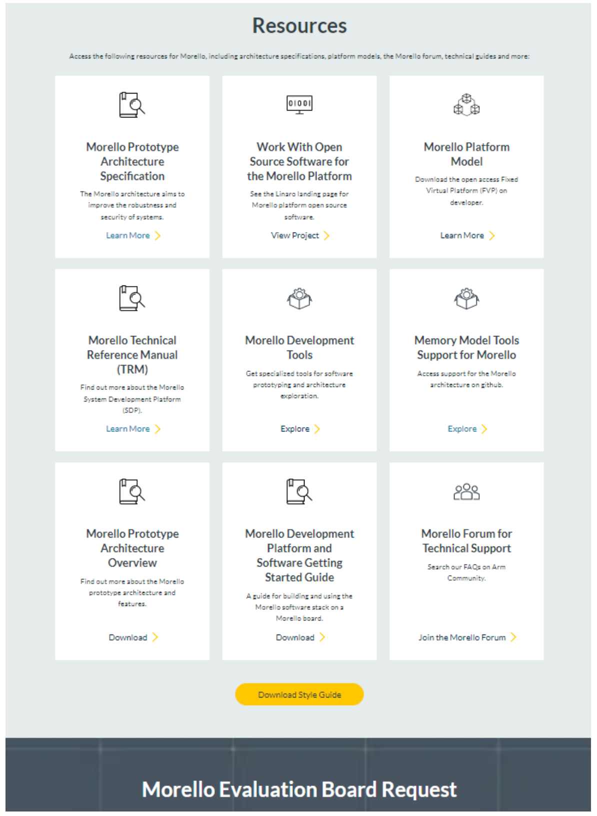
- Microsoft Security Research Center (MSRC) study analyzed all 2019 Microsoft critical memory-safety security vulnerabilities
  - Metric: "Poses a risk to customers → requires a software update"
  - Vulnerability mitigated if **no security update required**
- Blog post and 42-page report
  - Concrete vulnerability analysis for spatial safety
  - Abstract analysis of the impact of temporal safety
  - Red teaming of specific artifacts to gain experience
- CHERI, "in its current state, and combined with other mitigations, it would have **deterministically mitigated at least two thirds of all those issues**"

https://msrc-blog.microsoft.com/2020/10/14/security-analysis-of-cheri-isa/

© Copyright 2022 Arm Limited

# Morello Prototype system: What Has Arm Produced?

https://www.arm.com/morello



Resources

Access the following resources for Morello, including architecture specifications, platform models, the Morello forum, technical guides and more:

**Morello Prototype Architecture Specification**
The Morello architecture aims to improve the robustness and security of systems.
Learn More >

**Work With Open Source Software for the Morello Platform**
See the Linaro landing page for Morello platform open source software.
View Project >

**Morello Platform Model**
Download the open access Fixed Virtual Platform (FVP) on developer.
Learn More >

**Morello Technical Reference Manual (TRM)**
Find out more about the Morello System Development Platform (SDP).
Learn More >

**Morello Development Tools**
Get specialized tools for software prototyping and architecture exploration.
Explore >

**Memory Model Tools Support for Morello**
Access support for the Morello architecture on github.
Explore >

**Morello Prototype Architecture Overview**
Find out more about the Morello prototype architecture and features.
Download >

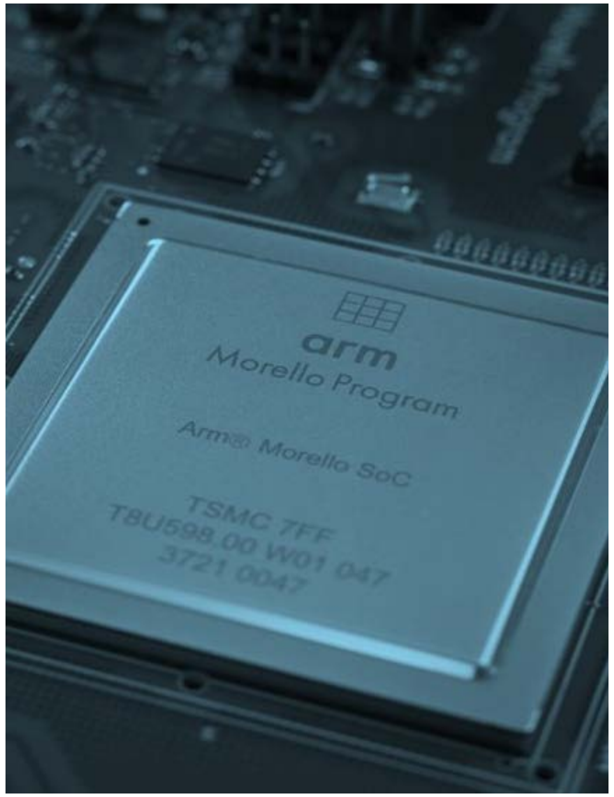
**Morello Development Platform and Software Getting Started Guide**
A guide for building and using the Morello software stack on a Morello board.
Download >

**Morello Forum for Technical Support**
Search our FAQs on Arm Community.
Join the Morello Forum >

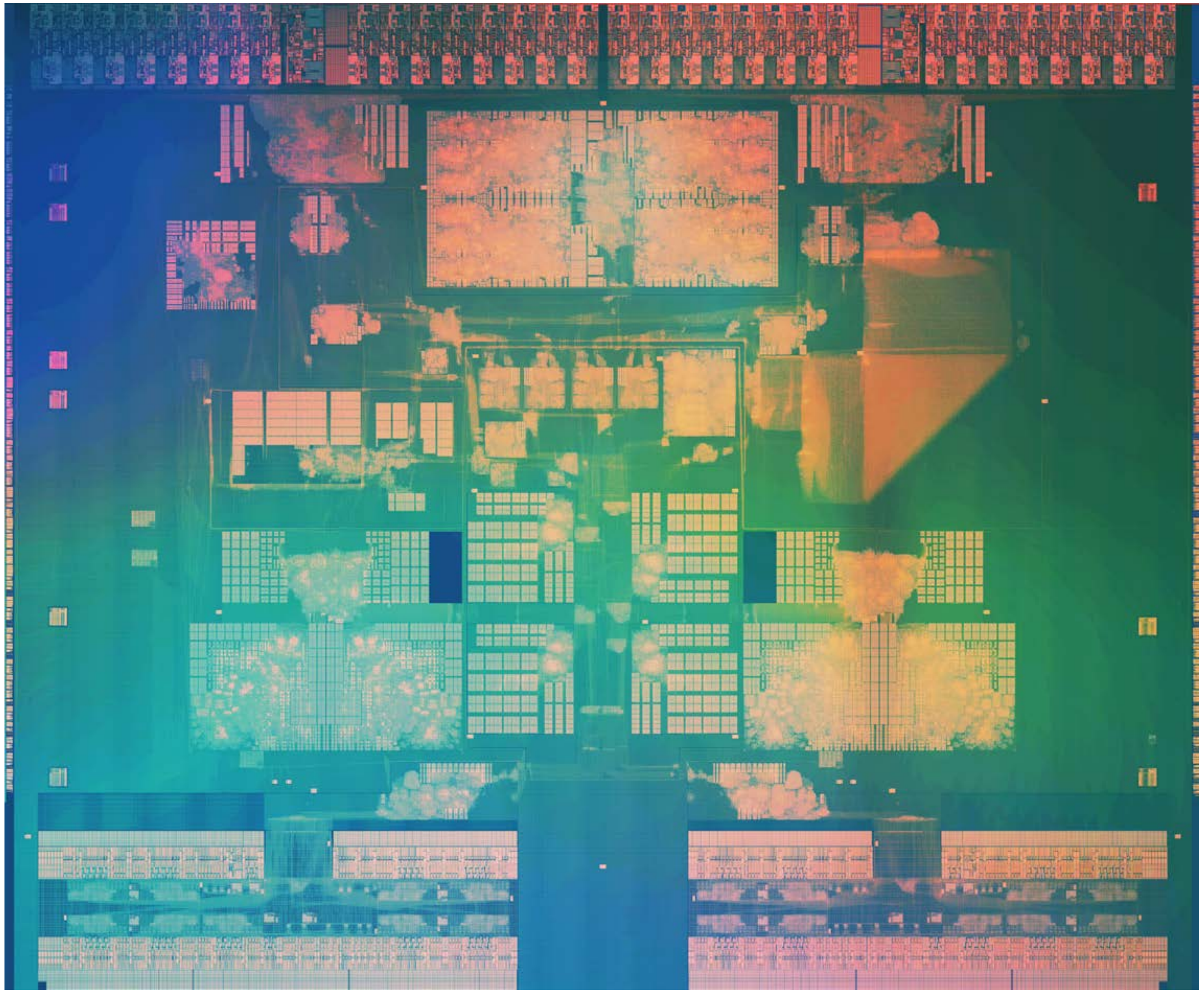
Download Style Guide

Morello Evaluation Board Request

- Morello prototype architecture
- Morello Platform Model (FVP) – a software model of the Morello platform
- Linaro hosted OSS
- Memory Model Tools
- Toolchain
- Partner Forum including FAQ
- Technical reference manual
- Morello test chip and board
- Morello Overview Guide
- Morello Development Platform and Software Stack User Guide
- Future – how-to video



© Copyright 2022 Arm Limited



TSMC N7 Process

2.5GHZ CPU

109.9mm$^2$

© Copyright 2022 Arm Limited

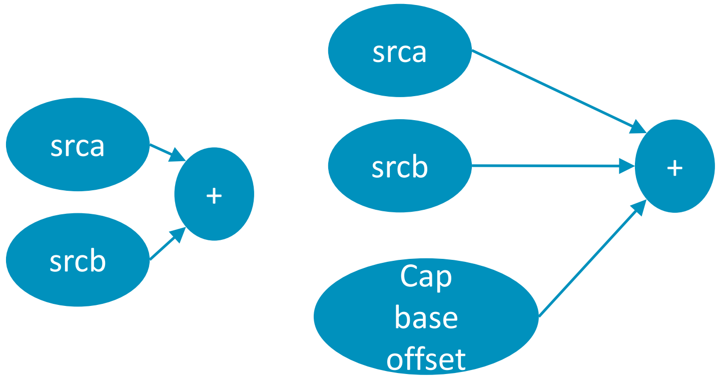
## Extending Structures and Memory to support capabilities

- Increase register file to support 129-bits
  - Area, power, and other register file optimizations need to be considered
  - Could be implemented as separate register file or unified register file
- Requires additional storage at all levels of memory hierarchy (1-bit per 16B of data)
  - Includes caches, buffers, and other microarchitecture structures
  - May widen existing structure or store in separate structures
- System buses need to transport tag information
  - Use existing signals where possible to decrease protocol changes
- Forwarding networks and internal data buses may need to increase
- Decode complexity and area (new instructions, modes, system registers)
  - Strains on decode space availability may require extra execution units or other changes

© Copyright 2022 Arm Limited

# Memory checks and Load Store extensions

- Address generation usually a critical path in load store designs
  - Compartmentalizing legacy code may add an offset to address generation
  - Capabilities require new bounds checks on those addresses

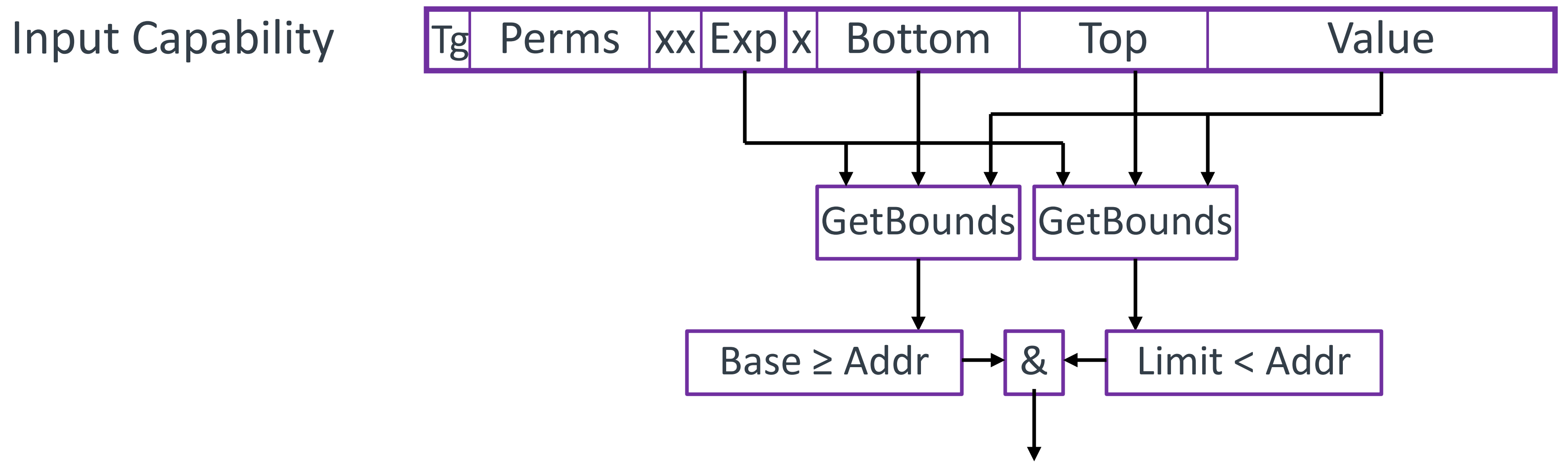


- New faults need to be detected and reported to control and track capabilities in a system
  - For protection (compartmentalization) or performance (revocation)
  - Adds dependency between stored capability and the location to which being stored that did not exist before (which may cause delays if implementation stalls address until data available)
  - MMU Access faults and PTE updates (capability write permission/dirty bit) dependent on store tag
- Fault Address Register (FAR) captures full address for all faults (including late detected, precise, data dependent faults)
  - May require additional storage to propagate full address throughout pipeline
- Capability instruction implementation must maintain atomicity
  - Makes cracking instructions more difficult, especially for atomic instructions such as CAS, ST{L}XP

© Copyright 2022 Arm Limited

# Extending data processing - Bounds Checking

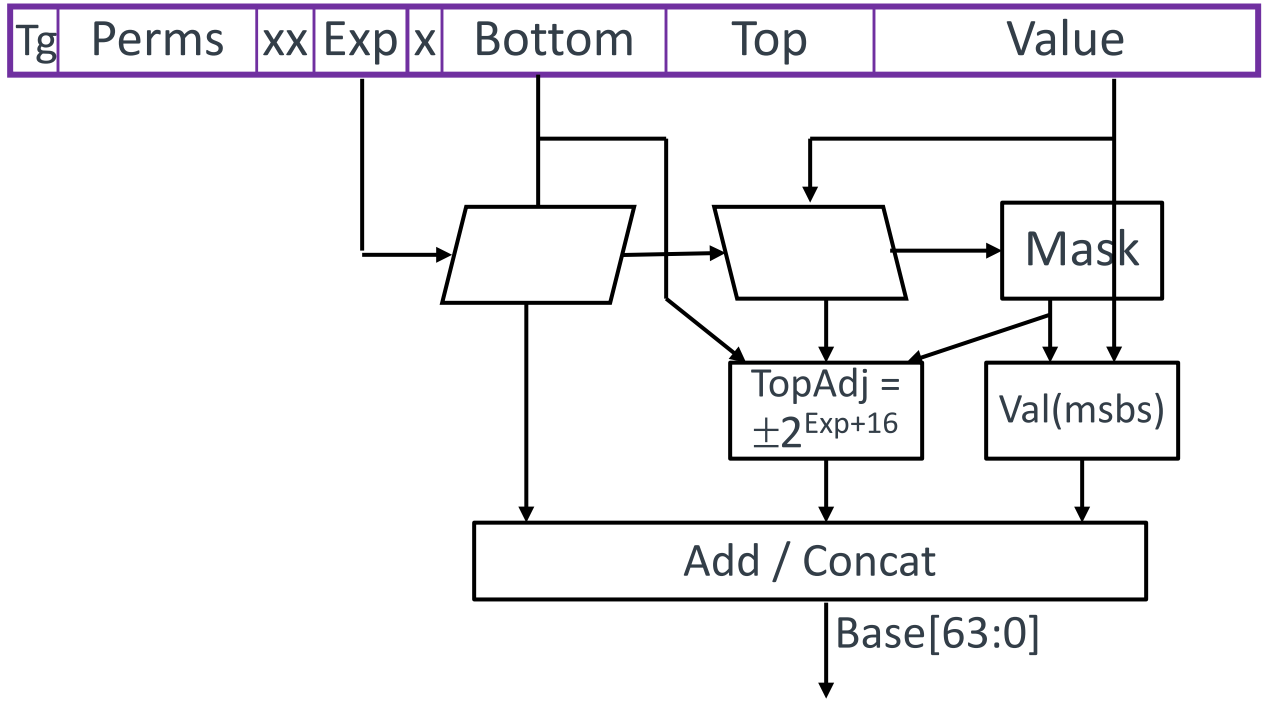
- Upper and Lower Bounds information is compressed into 64 bits
- When bounds checks are needed, this has to be decompressed
  - Needed for all loads/store operations and branches – done in parallel with address generation
  - Also needed to cover advanced capability operations in the integer unit



© Copyright 2022 Arm Limited

# Extending data processing - GetBounds logic

- Decoding of the compressed CHERI format for bounds
  - One bounds check requires two shifters, one adder, two short comparators (for TopAdj) and one wide comparator (to compare decompressed Bound against address)

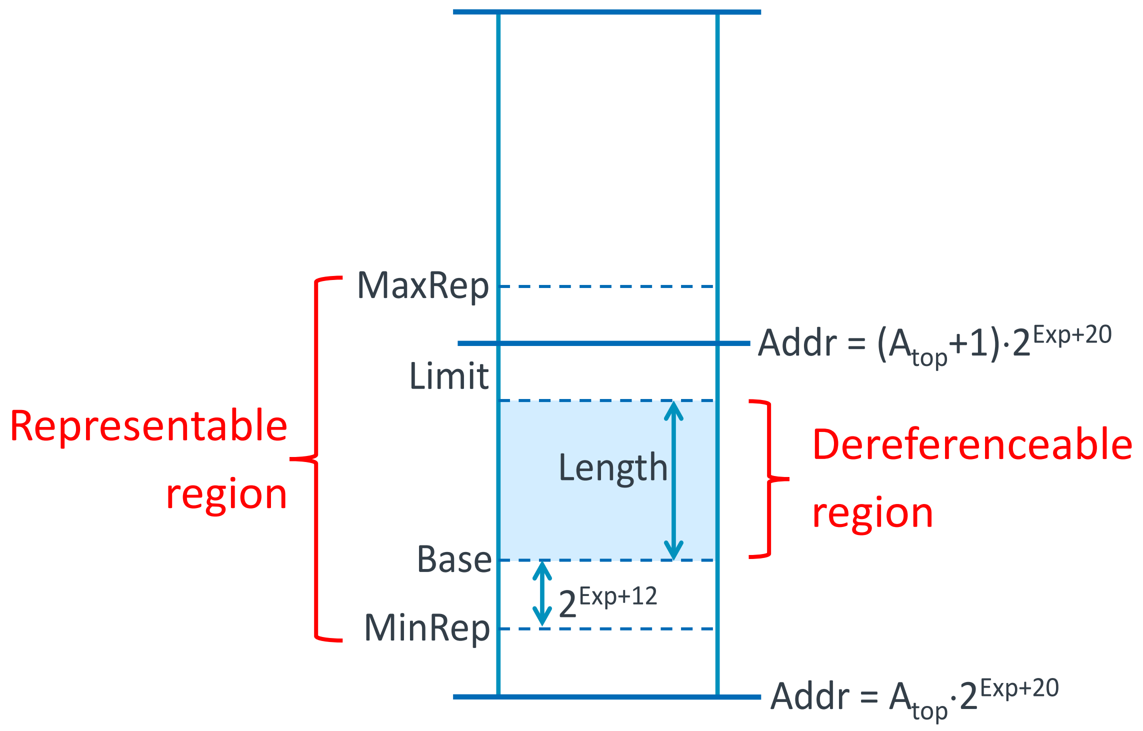


© Copyright 2022 Arm Limited

# Extending the data processing - Representability

Representability checks needed to make sure capability modifications are valid

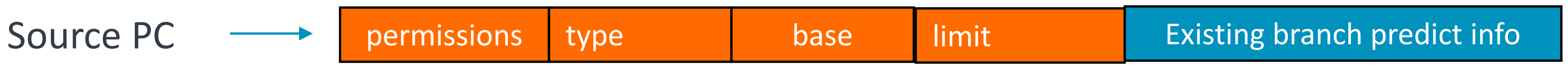
- Representability is an artifact of compressing two 64-bit bounds and a 64-bit pointer into 128 bits
- This affects arithmetic operations since taking the pointer too far out of bounds isn't representable
- Representability is designed to be fast for the common case (add/subtract) but other cases require full decompresson (absolute value)



© Copyright 2022 Arm Limited

# Instruction fetch and control flow prediction for capabilities

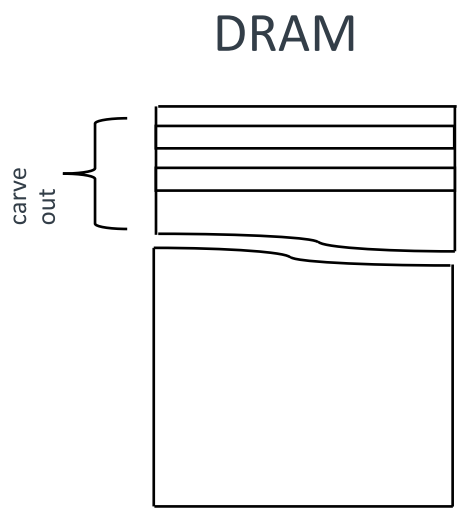
- PC is extended to 129-bits (Program Counter Capability - PCC)
- Branches need representable checks, bounds checks, and new fault handling
  - Includes new and legacy branches
  - Indirect legacy branches need expensive representable check that may add cycles
- How to handle indirect branches to a capability (might change PCC):
  - Extend branch predictor to hold PCC (simple 1-bit direction prediction or more complex and larger predictions)
  - Statically predict PCC does not change and take branch mispredict penalty if wrong
  - Stall PCC dependencies until PCC known
- Stalling PCC option reduces concerns in modifying/growing existing branch unit but introduces performance costs in delaying instructions dependent on new PCC

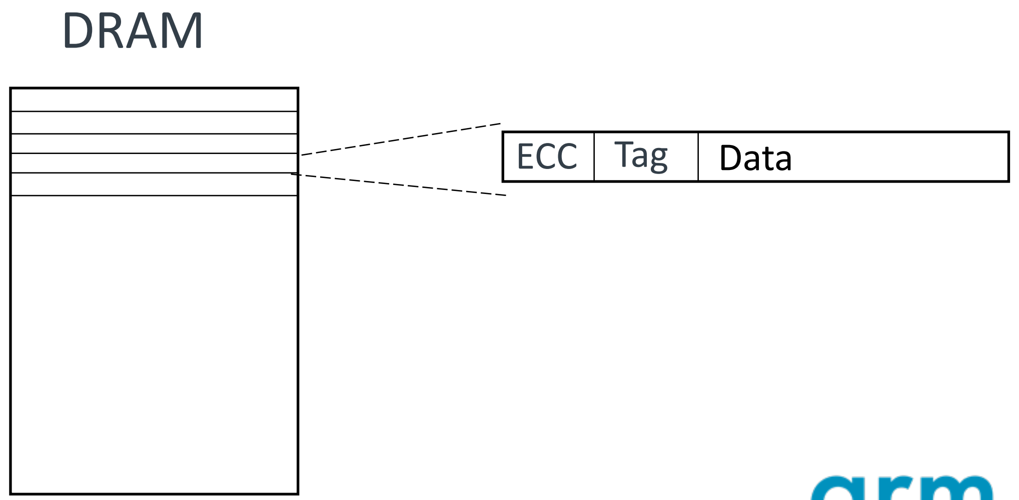


© Copyright 2022 Arm Limited

## Holding the CHERI capability validity tag in the memory

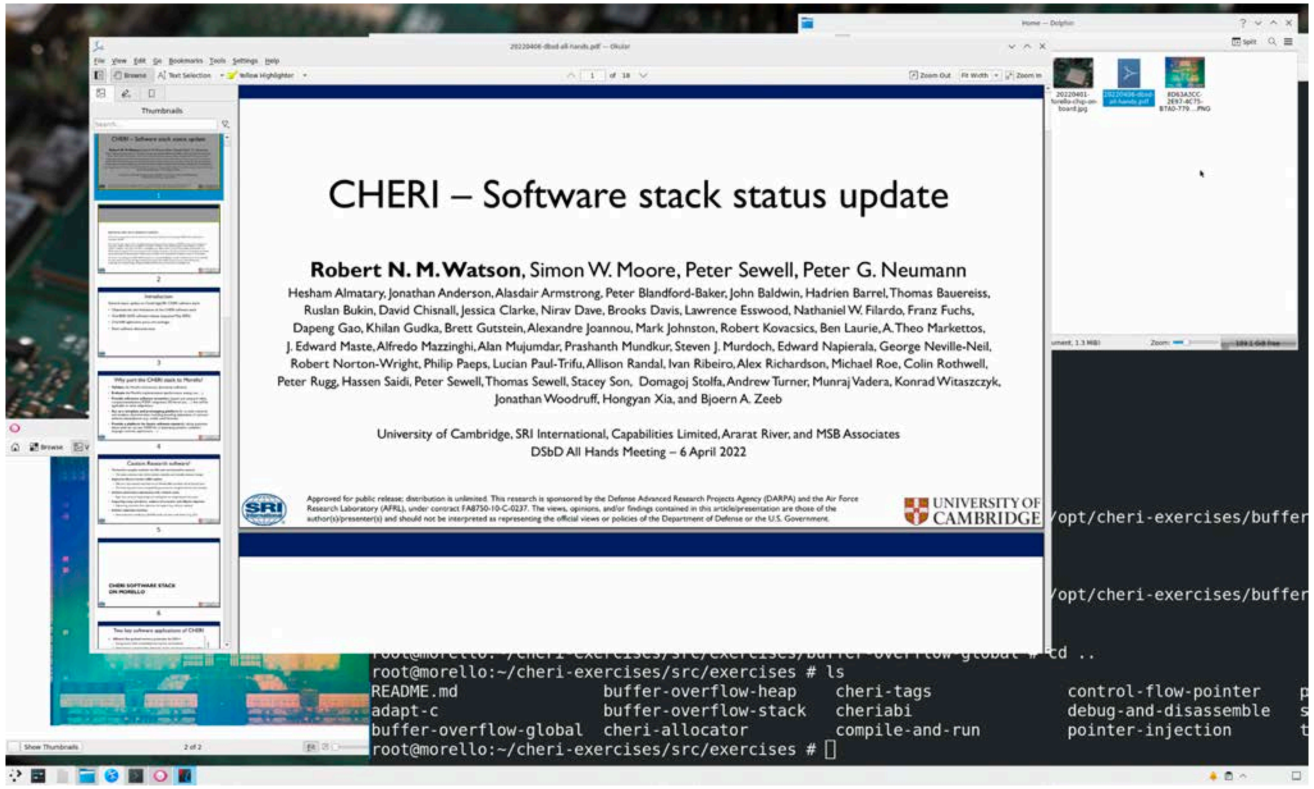
- Considerations should be made for tags in DRAM in terms of area and performance (similar to MTE)
- Add tags as separate carve out space in DRAM or as part of data within (such as ECC)
- Carve out space for tags reduces effective size of DRAM
- Increased latency to read both tags and data (two accesses)
  - Can be mitigated with other structures (tag caches, compression caches, etc.)
- If using existing ECC bits for tag may decrease resolution of single bit errors





© Copyright 2022 Arm Limited

# Demonstration Morello open-source desktop software stack



CHERI – Software stack status update

**Robert N. M. Watson**, Simon W. Moore, Peter Sewell, Peter G. Neumann
Hesham Almatary, Jonathan Anderson, Alasdair Armstrong, Peter Blandford-Baker, John Baldwin, Hadrien Barrel, Thomas Bauereiss, Ruslan Bukin, David Chisnall, Jessica Clarke, Nirav Dave, Brooks Davis, Lawrence Esswood, Nathaniel W. Filardo, Franz Fuchs, Dapeng Gao, Khilan Gudka, Brett Gutstein, Alexandre Joannou, Mark Johnston, Robert Kovacsics, Ben Laurie, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi, Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, George Neville-Neil, Robert Norton-Wright, Philip Paeps, Lucian Paul-Trifu, Allison Randal, Ivan Ribeiro, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg, Hassen Saidi, Peter Sewell, Thomas Sewell, Stacey Son, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Konrad Witaszczyk, Jonathan Woodruff, Hongyan Xia, and Bjoern A. Zeeb

University of Cambridge, SRI International, Capabilities Limited, Ararat River, and MSB Associates
DSbD All Hands Meeting – 6 April 2022

Approved for public release: distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

UNIVERSITY OF CAMBRIDGE

```
/opt/cheri-exercises/buffer

/opt/cheri-exercises/buffer

cd ..
root@morello:~/cheri-exercises/src/exercises # ls
README.md              buffer-overflow-heap   cheri-tags         control-flow-pointer    p
adapt-c                buffer-overflow-stack  cheriabi           debug-and-disassemble   s
buffer-overflow-global cheri-allocator        compile-and-run    pointer-injection       t
root@morello:~/cheri-exercises/src/exercises # 
```

- CHERI / Morello LLVM toolchains
  - Implements sub-language and C/C++ language pointers as capabilities
- CheriBSD adapted version of FreeBSD
  - Complete open-source operating system
  - Memory-safe kernel and userspace
  - Supports legacy aarch64 binaries
  - Multiple compartmentalization models, including fast IPC and sandboxed libraries
- X11, KDE-based desktop environment
  - 6MLoC adapted by 1 FTE in 3 months
  - 0.026%LoC changes for memory safety
  - 73.8% assessed vulnerability mitigation rate

© Copyright 2022 Arm Limited

## Formal proving of security properties of Morello

- Fundamental security property of CHERI architectures is Reachable Capability Monotonicity:
  - normal code execution, of arbitrary code, cannot increase the set of available capabilities
- Conventional testing cannot provide high assurance - but formal methods can prove it holds in general
  - Applied at the architecture specification language level
  - Arm ASL (62K lines of code) -> SAIL -> Isabelle Proof Assistant
  - 3 security issues found before tapeout
- SAIL ISA specification also used for ISA test generation
  - Stressing capability corner cases based on strong ISA understanding
- **Machine-checked mathematical proofs of whole-ISA security properties of a full-scale industry ISA**

Verified Security for the Morello Capability-enhanced Prototype Arm Architecture. Bauereiss (1), Campbell (2), Sewell (1), Armstrong (1), Esswood (1), Stark (2), Barnes (3), Watson (1), Sewell (1). In ESOP 2022. (1) University of Cambridge, (2) University of Edinburgh, (3) Arm Ltd. http://www.cl.cam.ac.uk/~pes20/morello-proofs-esop2022.pdf

© Copyright 2022 Arm Limited

arm Morello Program

## What Feedback Do We Want To Get from Morello?

- Answers to the performance questions for a wide range of different usage models
- Compelling examples of Capabilities offering security/performance improvements
  - Backed up by “Red-teams” having attacked the system and demonstrated security of the system
  - Compelling in comparison with existing deployed state of the art approaches
- Better Understanding of
  - Different languages and run-times can use capabilities, not only C and C++, but also Javascript, Java
  - Fine-grained compartmentalisation can be used
  - Answers to the performance questions for a wide range of different usage models
- A showcase to encourage other architectures to adopt capabilities
- Experience of what the right SoC hardware is for building capabilities
- An architectural approach with formally proven security properties

→ **What to put into the future Arm architecture for an industrial deployment**

© Copyright 2022 Arm Limited

## Questions?

© 2022 Arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה

© Copyright 2022 Arm Limited

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks

© Copyright 2022 Arm Limited