

CHERI

Morello Consortium update

Robert N. M. Watson (University of Cambridge), Simon W. Moore (Cambridge), Peter Sewell (Cambridge), Peter G. Neumann (SRI), Brooks Davis (SRI), Joakim Bech (Linaro), Luis Machado (Linaro), Mark Nicholson (Arm), Mark Inskip (Arm)

Digital Security by Design (DSbD) - All Hands Meeting
7 May 2021

This work was sponsored in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694. Approved for public release; distribution is unlimited. Sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 ("CTSRD"), 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, either expressed or implied, of the Department of Defense or the U.S. Government.



The Morello Consortium

- **Morello** is a prototype processor, SoC, and board adapting the ARMv8-A architecture to implement the **CHERI protection model**
- Four organisations collaborating to create the **Morello platform prototype** and **baseline software ecosystem**:
 - ARM UK Ltd
 - Linaro
 - University of Cambridge
 - University of Edinburgh
- Updates from each, with some observations as we go ...

Threads of execution

- Arm Morello architecture
- Arm Morello CPU, SoC, and board
- Morello architecture model and proofs, generated tests
- Morello toolchains (Clang/LLVM/LLD/LLDB, GCC, GDB)
- CheriBSD/Morello OS
- Android/Morello OS
- Open-source application corpus
- Documentation and tutorial material
- Support

Architecture,
hardware, and
proofs

Software
ecosystem - multiple
toolchains, OSes

Supporting
material

Mark Inskip

ARM: MORELLO ARCHITECTURE, HARDWARE, TOOLCHAIN, AND ANDROID



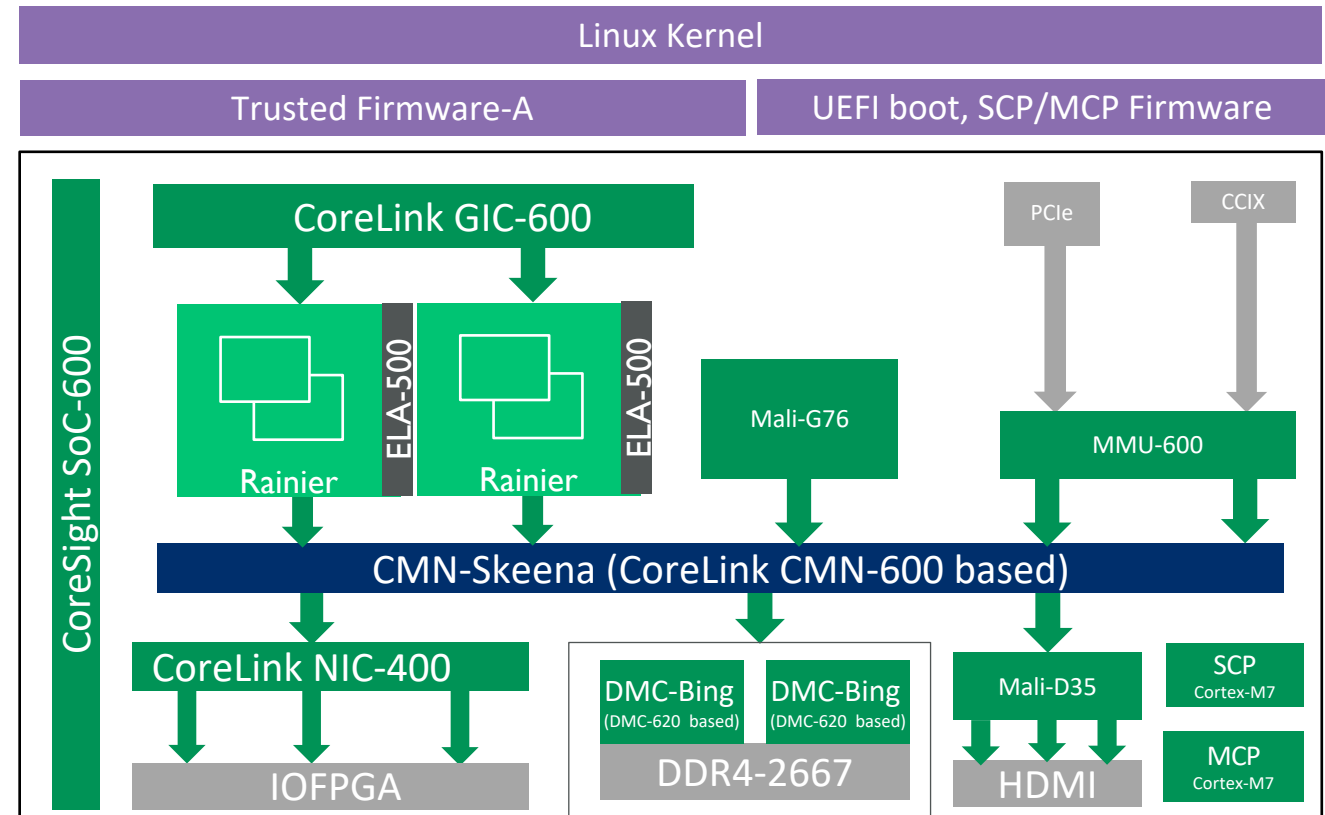
arm

Morello Technology Demonstrator Update May 2021

Mark Inskip, Program Director
Arm Central Engineering

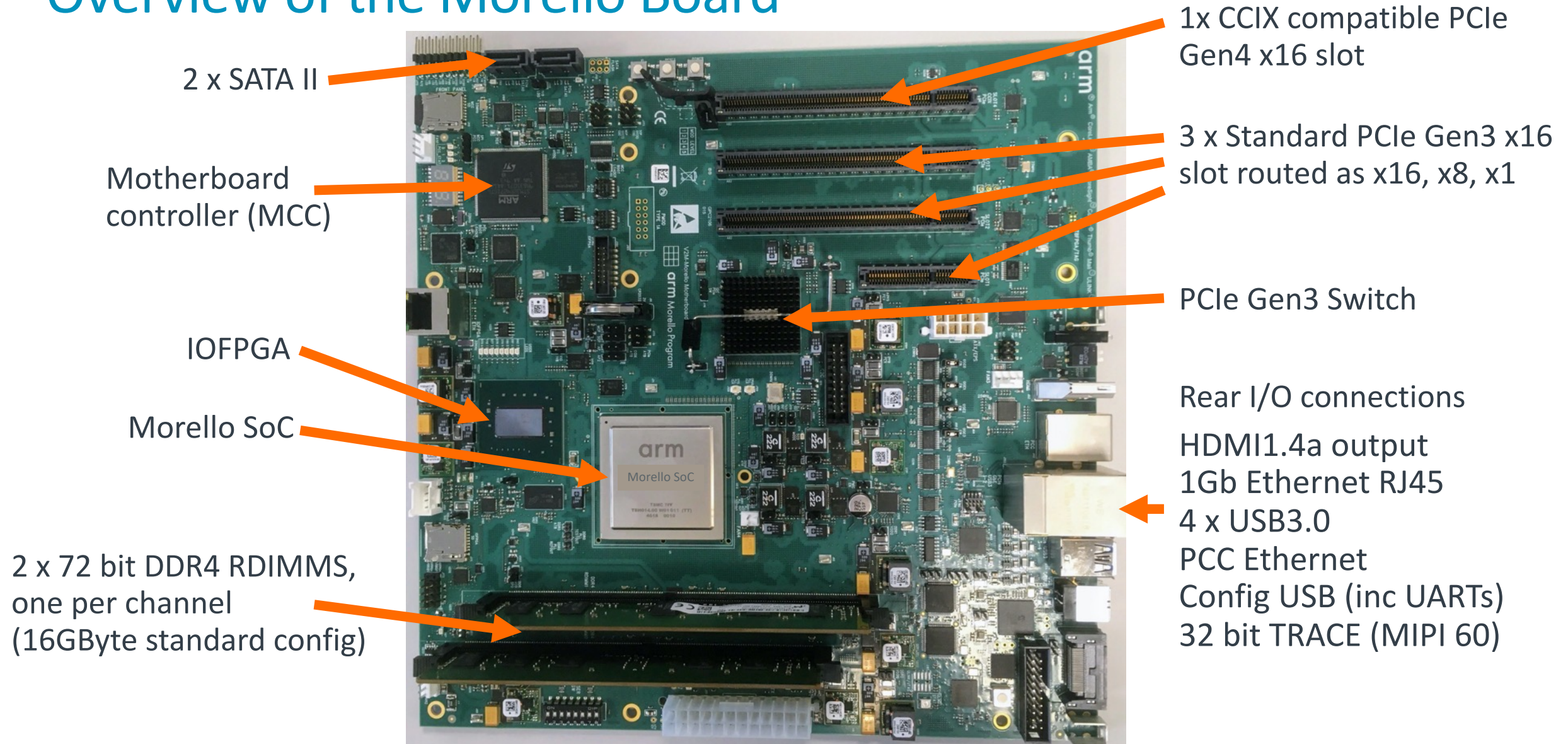
Morello Board: Capability Hardware Prototype Platform

- Silicon implementation of a Capability Hardware CPU Instruction Set Architecture
 - Implements Morello Profile for A-class Prototype Architecture
 - Two clusters each of two Rainier CPUs
 - Interconnect and Memory Controller support for tagged memory
 - Two channel DDR4 DRAM interface
 - PCIe Gen3 and Gen4 x16 interface
 - CCIX (Cache Coherent Interconnect for Accelerators) interface
 - Mid-range GPU, display processor and HDMI output
 - On standard uATX form factor board



- Supporting Arm system IP: GIC-600 (Generic Interrupt Controller), MMU-600 (IO MMU), Dynamic Memory Controller derived from DMC-620, SoC-600 (SoC Debug and Trace), Coherent Mesh Network derived from CMN-600, NIC-400 (Non-coherent interconnect)
- Supporting 3rd party system IP/hardware: PCIe/CCIX Root Complex (PHY and controller), DDR4/3 PHY, DDR4 memory, IO FPGA
- Open-source software stack

Overview of the Morello Board




Existing Arm software for Morello

-----● Oct 2020 -----● Jan 2022 -----

Platform Deliverables

CheriBSD environment available from Cambridge University 

Morello Platform Model (FVP) releases

- Platform Model available for download
- Code [repositories](#) live – hosting:
 -  Headless Android dev environment
 - Initial mods to arm64-v8a AOSP to support Capabilities
 - Open Source firmware (SCP, TF-A, EDK2)

Development board

Arm Internal hardware bring up

Linux Kernel

Morello Kernel

ACK derived baseline for future Morello Kernel development.

C Library

BIONIC Android C lib for purecap applications

Kernel syscall ABI support reliant on library “shim”

Toolchain

CHERI LLVM toolchain

Primary toolchain and utilities *(in association with Cambridge University)*

Linaro

Infrastructure

Initial software access - Infrastructure for code hosting


CI development, contributions support

Current & future Arm Morello enablement plans (provisional)



Morello Development board – aims for initial hardware release

- Parity with FVP release features on hardware platform
 - Android dev environment (Nano config)
 - Initial full Android (Software Rendering + DPU) hardware boot


Android Enablement (ongoing)

- Dynamic Linking support
- Additional pure-cap component ports (shell, libjpeg-turbo, etc)
- Preparatory graphics stack porting
- Investigation: Android Runtime + zygote 


Linux Userspace enablement

1. Simple Linux environment (e.g. busybox): LLVM + musl
2. Simple Linux environment (e.g. busybox): GCC + Glibc 
3. Basic Distro framework 


Morello Kernel enablement

- Incremental progress towards Kernel support for Android & Linux pure-cap environments 
 - Base 64-bit functionality (COMPAT), evolving pure-cap syscall ABI support, enforcement of Capability metadata
 - Aligns with associated C library development



C Libraries

- C libraries initially developed using syscall ABI “shim” - functionality reduced in line with evolving Kernel
 - BIONIC (Android)
 - Musl libc
 - Glibc 

Toolchains


- LLVM / Clang toolchain: additional functionality (C++ Exceptions, Compartments)
- GNU toolchain: Binutils (gas, objdump, LD), GCC, GDB 

Panfrost Graphics

- Community enablement of A64 Panfrost drivers for Bifrost GPU architecture, enabling →
 - A64 driver ports (Mali G76 GPU) & integration into OS 
 - graphics frameworks on Morello hardware, enabling →
 - Purecap (c64) ports of OS graphics components 

Hosting Infrastructure

- Contributions and public CI
- Linaro maintenance

 Longer term (post silicon) development work with deliveries 2022 onwards.

Morello Toolchains

LLVM & GNU Toolchain Plans

LLVM

- **Ongoing:** Regular re-bases to CHERI LLVM.
- **31 March 2021** (Done):
 - C++ exceptions (static linking).
- **30 June 2021:**
 - Performance optimisations.
 - Descriptor ABI (spec, codegen, LLD, LLDB).
- **30 September 2021:**
 - C++ exceptions (dynamic linking).
- **31 March 2022:**
 - More extensive public test.
- **30 June 2022:**
 - Code generation for DDC offsetting mode.

GNU Tools

- **30 September 2021 :**
 - GDB (REL)
- **31 December 2021:**
 - GCC ACLE Intrinsics & C Language Support
- **31 March 2021:**
 - GCC C++ Language Support
 - Glibc

arm

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

ধন্যবাদ

תודה

UNIVERSITY OF CAMBRIDGE: CHERI, ARCHITECTURE, SYSTEMS SOFTWARE, AND APPLICATION CORPUS

CHERI

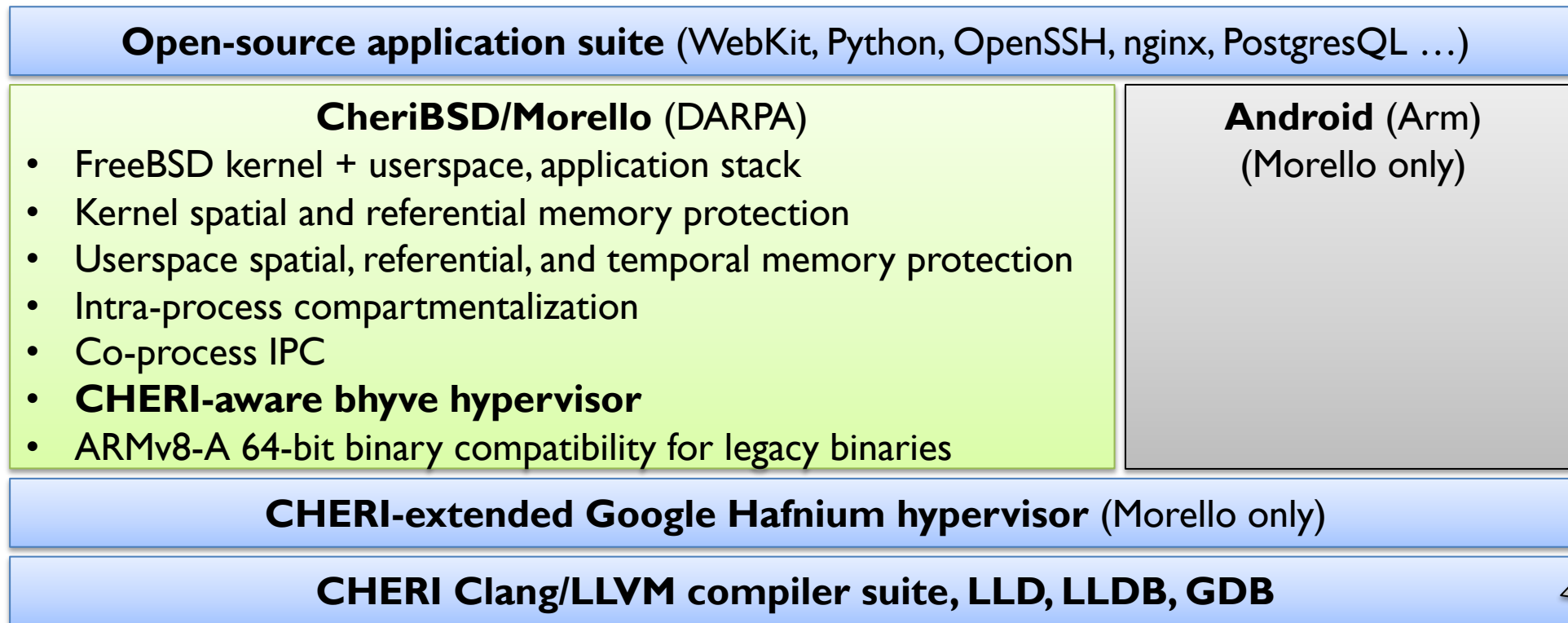
- Hardware-software(-semantics) co-design since 2010; 3x DARPA programmes
 - Atom smash ideas about capability systems with contemporary architecture, microarchitecture, and C/C++-language software – first MIPS, now RISC-V
 - **Introduction to CHERI** is the best starting point, but pre-Morello (UCAM-CL-TR-941)
- **CHERI ISAv8** spec shipped October 2021 (UCAM-CL-TR-951)
 - Compressed capabilities now the “model”; CHERI support for 32-bit RISC-V; many experimental features now non-experimental (e.g., sentries), load-side barrier temporal safety; microarchitecture chapter
 - **Synchronised with final Arm Morello ISA**
- **DSbD CHERI and Morello Capability Essential IP (Version 1)** shipped December 2021 (UCAM-CL-TR-953)
- **CHERI ISAv9** due mid-2021
 - CHERI-RISC-V compressed instructions; CHERI-x86 ISA elaboration; improved CHERI-RISC-V static linking; further updates on microarchitecture

Open-source CHERI-RISC-V microarchitectures

- Demonstrate and evaluate CHERI techniques on FPGA
- Implemented in BlueSpec System Verilog (BSV)
 - CHERI-Piccolo - 32-bit microcontroller; w/o MMU
 - CHERI-Flute - 32/64-bit pipelined processor w/MMU
 - CHERI-Toooba - 64-bit multicore superscalar processor w/MMU
- Used on Xilinx VCU-118 and Amazon AWS F1; Stratix 10 planned
- Run CHERI software stacks (CheriBSD and/or CheriFreeRTOS)
- MIT or Apache-style licenses

CHERI prototype software stack on Morello

- **Complete open-source CHERI-enabled software stack** from bare metal up: compilers, toolchain, debuggers, operating systems, applications – all demonstrating CHERI ideas



- Rich CHERI feature use, but fundamentally incremental/hybridized deployment
- Aim: Mature and highly useful research and development platform for Morello

Open-source CHERI Clang/LLVM/LLD+GDB

- Extensions to the open-source Clang/LLVM/LLD toolchain and GDB debugger
 - Supports CHERI-RISC-V and CHERI-MIPS (deprecated)
 - Upstream vendor for Arm's Morello Clang/LLVM/LLD; input into Morello GDB, LLDB
 - Closely tracks upstream baseline LLVM project
- Two compilation modes:
 - Hybrid CHERI C/C++: Source annotations explicitly mark capability types; goal ABI compatibility } Discouraged
 - Pure-capability CHERI C/C++: All C/C++ pointers, implied or explicit, implemented as capabilities; goal strong protection through new ABI } Encouraged
- **CHERI C/C++ Programming Guide**, June 2020 (UCAM-CL-TR-947)
- Plenty to do:
 - **CHERI Hybrid C/C++ Programming Guide**; delayed as we iterate on what Hybrid C should be
 - Converge CHERI and Morello GDB implementations; develop richer CHERI debugger ideas
 - Explicit specification, as Arm works on a second independent implementation: Morello GCC

CheriBSD operating system

- CHERI adaptation of FreeBSD UNIX (Netflix, Netapp, EMC, Juniper, Sony, Nintendo, ...)
- CheriBSD runs on Morello FVP and on Morello FPGA:
 - AArch64 binary compatibility environment Complete
 - Userspace spatial safety (“CheriABI”) Complete (C and C++)
 - Kernel spatial safety Complete
 - Userspace temporal safety In progress adaptation from RISC-V
 - Co-process compartmentalization Not yet adapted from RISC-V
 - Morello-enabled virtual machines on bhyve Not yet started
- Applications such as OpenSSH, nginx, WebKit, etc., all run well on Morello in CheriABI
- Panfrost framework device-driver adapted; awaiting Collabora Morello GPU driver
- **June 2021 next release** – will ship from CheriBSD main branch rather than Morello dev
- **Key objectives for 2021Q4:** CheriABI applications; co-process compartmentalization

CHERI C compatibility: CheriBSD Code Changes

Area	Files total	Files modified	% files	LoC total	LoC changed	% LoC
Kernel	11,861	896	7.6	6,095k	6,961	0.18
• Core	7,867	705	9.0	3,195k	5,787	0.18
• Drivers	3,994	191	4.8	2,900k	1,174	0.04
Userspace	16,968	649	3.8	5,393k	2,149	0.04
• Runtimes (excl. libc++)	1,493	233	15.6	207k	989	0.48
• libc++	227	17	7.5	114k	133	0.12
• Programs and libraries	15,475	416	2.7	5,186k	1,160	0.02

Notes:

- Numbers from cloc counting modified files and lines for identifiable C, C++, and assembly files
- Kernel includes changes to be a hybrid program and most changes to be a pure-capability program
 - Also includes most of support for CHERI-MIPS, CHERI-RISC-V, Morello
 - Count includes partial support for 32 and 64-bit FreeBSD and Linux binaries.
 - 67 files and 25k LoC added to core in addition to modifications
 - Most generated code excluded, some existing code could likely be generated

C/C++ compatibility: WebKit - JSC Code Changes

Area	Files total	Files modified	% Files	LoC total	LoC changed	% LoC
JSC-C	3368	148	4.4	550k	2217	0.40
JSC-JIT	3368	339	10.1	550k	7581	1.38

Notes:

- JSC-C is a port of the C-language JavaScriptCore interpreter backend
- JSC-JIT includes support for a meta-assembly language interpreter and JIT compiler
- Runs SunSpider JavaScript benchmarks to completion
- Language runtimes represent worst-case in compatibility for CHERI
 - Porting assembly interpreter and JIT compiler requires targeting new encodings
- Changes reported here did not target diff minimization
 - Prioritized debugging and multiple configurations (including integer offsets into bounded JS heap) for performance and security evaluation
 - Some changes may not be required with modern CHERI compiler

QEMU

- QEMU-CHERI
 - Mature, fast, open-source ISA-level emulator for CHERI-RISC-V (+CHERI-MIPS)
 - Support for multithreaded QEMU added, improving multicore emulation speed
- QEMU-Morello
 - Adaptation of our QEMU-CHERI to add Morello ISA support
 - Now boots CheriBSD with CheriABI userspace
 - Ready for experimental third-party use in June 2021?
- QEMU-Userlevel CheriABI
 - Support for userspace-only execution of CheriABI binaries
 - Primarily used in cross-build environments
 - Dynamically linked binaries now in progress
 - Will extend to Morello once mature for CHERI-RISC-V

How to obtain and install CHERI software stack

```

README.md

cheribuild.py - A script to build CHERI-related
software (requires Python 3.5.2+)

This script automates all the steps required to build various CHERI-related software. For example cheribuild.py
[options] sdk will create a SDK that can be used to compile software for the CHERI CPU and cheribuild.py
[options] run-riscv64-purecap will start an instance of CheriBSD built for RISC-V in QEMU.

cheribuild.py also allows building software for Arm's adaption of CHERI, the Morello platform, however not all
targets are supported yet.

Supported operating systems

cheribuild.py has been tested and should work on FreeBSD 11 and 12. On Linux, Ubuntu 16.04, Ubuntu 18.04
and OpenSUSE Tumbleweed are supported. Ubuntu 14.04 may also work but is no longer tested. macOS 10.14
and newer is also supported.

Pre-Build Setup

macOS
When building on macOS the following packages are required:

brew install cmake ninja libarchive git glib automake autoconf coreutils llvm make wget pixman
# Install samba for shared mounts between host and CheriBSD on QEMU
brew install arichardson/cheri/samba
# If you intend to run the morello FVP model you will also need the following:
brew install homebrew/cask/docker homebrew/cask/xquartz socat dtc

Ubuntu
If you are building CHERI on a Debian/Ubuntu-based machine, please install the following packages:

apt-get install libtool pkg-config clang bison cmake ninja-build samba flex texinfo libglib2.0-

Older versions of Ubuntu may report errors when trying to install libarchive-tools. In this case try using apt-
get install bsdtar instead.

RHEL/Fedora
If you are building CHERI on a RHEL/Fedora-based machine, please install the following packages:

dnf install libtool clang-devel bison cmake ninja-build samba flex texinfo glib2-devel pixman-d

Basic usage
If you want to start up a QEMU VM running CheriBSD run cheribuild.py run-riscv64-purecap -d (-d means

```

- One build tool to rule them all: cheribuild
<https://github.com/CTSRD-CHERI/cheribuild>
- Builds, installs, and/or runs:
 - Morello FVP or QEMU CHERI-RISC-V
 - CheriBSD/Morello disk images
 - Small suite of adapted third-party applications
- Up and running with one command (CHERI-RISC-V):
./cheribuild.py --include-dependencies run-riscv64-purecap
- We will integrate QEMU-Morello support with cheribuild in the coming month; only the FVP for now

In-progress research (1/2)

- CHERI RISC-V PPA and dynamic performance (DARPA)
- CHERI-RISC-V ISA and ABI optimization (DARPA)
- Formal modeling and proof for CHERI-RISC-V (DARPA)
- **CHERI and speculative execution** (GCHQ, DARPA)
- Software compartmentalization demonstrator: Apache + TLS (DARPA)
- **CheriFreeRTOS compartmentalized embedded OS** (DARPA)
- **CHERI temporal memory safety in HW/SW (w/Microsoft)** (DARPA)
- CHERI Clang/LLVM/LLD optimizer analysis and improvement (EPSRC)
- **WebKit safety, compartmentalization, JIT (w/Arm)** (Gates)

In-progress research (2/2)

- CHERI-BGAS PGAS-style hardware and software model (LPS)
 - Morello ISA and hardware validation (Innovate)
 - **Spatial, temporal memory safety evaluation on Morell** (Innovate)
 - Formal modeling and proof for Morello (Innovate)
 - Linux CheriABI development and validation (w/Arm) (Innovate)
 - **Open-source desktop ecosystem assessment (w/CapLtd)** (Innovate)
 - Software operational models for CHERI compartmentalization (EPSRC)
 - CHERI support in GPUs and accelerators (EPSRC)
- } See our posters!

Peter Sewell

CAMBRIDGE AND EDINBURGH: FORMAL MODELING AND VALIDATION

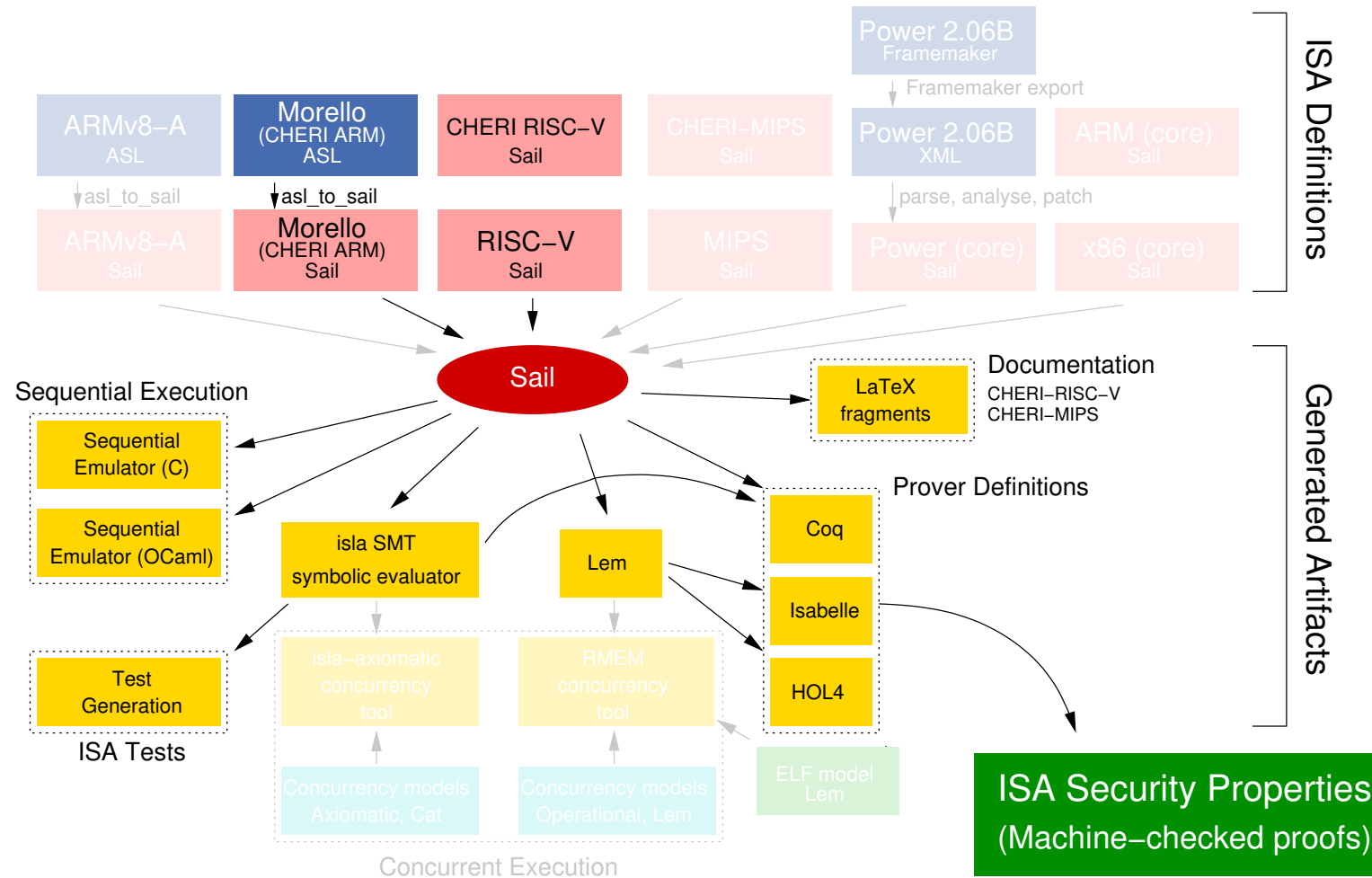
CHERI semantics

- Hardware-software-**semantics** co-design since 2014
 - CHERI-RISC-V, CHERI-MIPS defined in Sail (our ISA definition language)
 - Underlying RISC-V model adopted by RISC-V International
 - Morello ISA (and Armv8-A) auto-translated from Arm ASL to Sail
- [Thomas Bauereiss, Brian Campbell, Thomas Sewell, Kyndylan Nienhuis, Alasdair Armstrong, Prashanth Mundkur, Robert Norton-Wright, Alexandre Joannou, and Alastair Reid, in collaboration with the rest of the CHERI team, Arm, and RISC-V International]

From Sail to ..

- From Sail we generate multiple artefacts:
 - **documentation:** the detailed CHERI-RISC-V and CHERI-MIPS ISA documentation **is** the typeset Sail code
 - **emulation:** we use a Sail-generated C emulator (~400KIPS) as a reference for testing hardware and QEMU against, for initial software bring-up, and (for Arm) to validate the Sail version of the model against Arm tools
 - **symbolic evaluation:** our **Isla** SMT-based symbolic evaluator for Sail produces simplified views of the semantics under given assumptions
 - **test generation:** using Isla we generate interesting ISA tests with good spec coverage
 - **theorem-prover definitions:** we generate Isabelle, Coq, and HOL4 versions of the full ISA definitions to use for proof (also in progress: integration of Isla traces into Coq and Iris for program-logic reasoning)
- This automatic generation enables easy experimentation and helps keeps things in sync.
- All this is for the full sequential ISA definitions, including systems aspects and able to boot OSs, not just idealised fragments.

Sail flow: From ISA models to spec, tests, proof



Provable ISA security properties

- CHERI ISA designs are supposed to provide strong guarantees that system software can rely on to give better security
- But how do we know they do? A single small error in the (~100kLoC) Morello spec could break those guarantees.
- Answer: we state those guarantees as properties of the CHERI ISA specs, and do machine-checked mathematical proof (in the Isabelle proof assistant) that they hold. E.g.:
 - Theorem [**Capabilities cannot be forged (Capability Monotonicity)**] For any intra-domain trace, the reachable capabilities from the final state are no greater than those of the initial state.
 - Theorem [**Compartmentalisation**] Any trace within a properly set-up compartment cannot affect other memory, and can exit the compartment only in controlled ways.
- These are properties of **arbitrary code** above the ISA.
 - Initially for CHERI-MIPS (above the earlier L3 model) [Nienhuis et al., Security & Privacy 2019]
 - Now monotonicity proof essentially complete for Morello, under various assumptions (e.g. about address translation). [Bauereiss, T. Sewell, Campbell, Armstrong]
 - Also proofs and SMT checking of compression scheme.

Formal artifacts

- Sail models and generated prover (esp. Isabelle) definitions:
 - for CHERI-RISC-V, RISC-V, CHERI-MIPS, Armv8-A: all available
 - for Morello: available soon (mid-2021)
- Isabelle proofs for Morello: available soon

- Future plans:
 - software verification above these models
 - semantics and verification for CHERI C, CHERI LLVM (TBD)

Joakim Bech, Luis Machado

LINARO: ECOSYSTEM ENABLEMENT

Linaro's Morello and CHERI contributions

Joakim Bech, Luis Machado
2021-05-04



Toolchain Contributions

Updates

- LLDB - functional in maintenance mode
 - Improved unwinding support (backtraces)
 - Descriptor ABI support

- GDB - functional alpha stage
 - Support for reading/writing capabilities from/to memory
 - Support for calling functions from within GDB
 - Core file support
 - C register dumps
 - Capability tag dumps WIP
 - AUXV / Linkmap support WIP

Toolchain Contributions

Updates

- Binutils / GAS / LD - functional
 - Maintenance and bug fixes

- GLIBC - under development
 - C64 string/memory routines
 - Morello setjmp/longjmp support
 - Syscall interface
 - Sysdeps enablement
 - Static binary support WIP
 - Dynamically-linked binaries WIP

Toolchain Contributions

Planned work

- LLDB
 - Maintenance and bug fixes

- GDB
 - Core file support
 - AUXV / Linkmap
 - Maintenance and bug fixes

- Binutils / GAS / LD
 - Maintenance and bug fixes

- GLIBC
 - Static binary support
 - More sysdeps enablement
 - Dynamically-linked binaries

Software enablement

- Usability from a newcomers perspective
 - Helping out verifying and providing feedback on existing documentation and guidelines.
 - Contributed with guidelines on how to build Morello Nano (AOSP based Morello build)
- Pure-capability enablement
 - Done changes to a few software projects within AOSP
 - Once you've understood how to build and port a project, it's pretty easy to port additional ones

Software project	Diff (*)	Total LOCs (**)	% of code changed	Comment
bzip2	3 files changed, +22, -1	8186	0.00281	MR being reviewed
dnsmasq	1 file changed, +7	14389	0.00049	MR being reviewed
sqlite(3)	2 files changed, +33, -2	469220	0.000007	Libshim issues, MR being reviewed
tcpdump	1 file changed, +7	105408	0.000006	MR being reviewed
toybox	6 files changed, +36, -17	82936	0.00064	MR being reviewed
.../arm/morello-examples	n/a	n/a		MR being reviewed

(*) Patch statistics to change the application to run as a pure capability binary.

(**) `$ find . -type f -name '*.ch' -o -name '*.bp' | xargs wc -l`

CONCLUSION

An evolving CHERI/Morello software ecosystem

- Current activity: Laying the foundations before the hardware ships
 - Architectural formal models and proofs
 - Emulators – FVP and QEMU, and soon QEMU-userlevel
 - ABI, compiler, and toolchain functionality, performance
 - CHERI-enabled OSes including FreeBSD+Android, drivers etc.
 - Basic open-source application stack – WebKit, nginx, OpenSSH, ...
- But there are lots of gaps (e.g., high-level language runtimes)...
 - ... And lots of questions (community, repositories, multi-OS, ...)
- **Invitation: Engage in creating a Morello software community!**

Ways to engage

- Chat with folk from Arm, Cambridge, Edinburgh, and Linaro around this workshop – and see our research posters
- Join us on cheri-cpu.slack.com
 - Email cheri-slack@cl.cam.ac.uk to request an invitation
- cl-cheri-discuss mailing list – low traffic but probably will grow

Q&A