CTSRD

CRASH-worthy Trustworthy Systems Research and Development

Robert N. M. Watson
Peter G. Neumann

DARPA CRASH PI Meeting
Arlington, VA, USA
8 November 2011

Approved for public release. 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/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

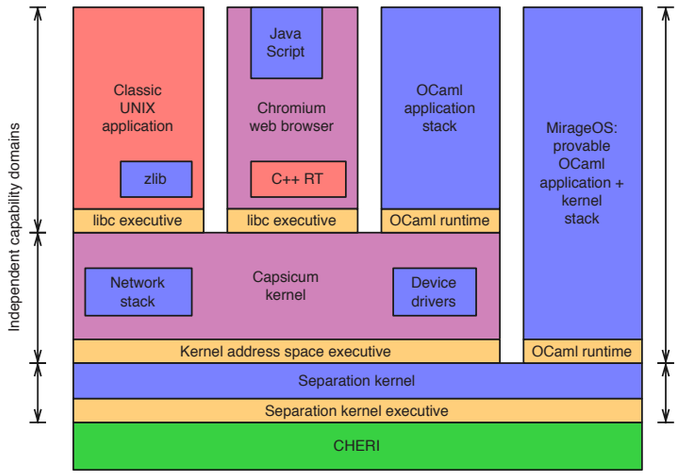
SRI International

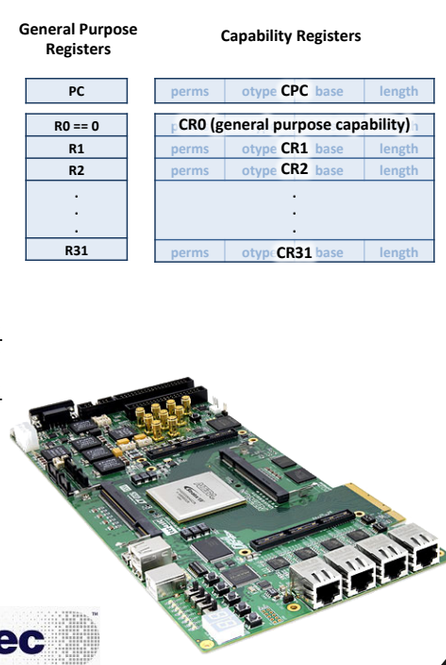
University of Cambridge

May 2011 CTSRD review meeting in Cambridge, UK:
SRI, Cambridge, DARPA, external oversight group

CTSRD elements

CHERI processor and software



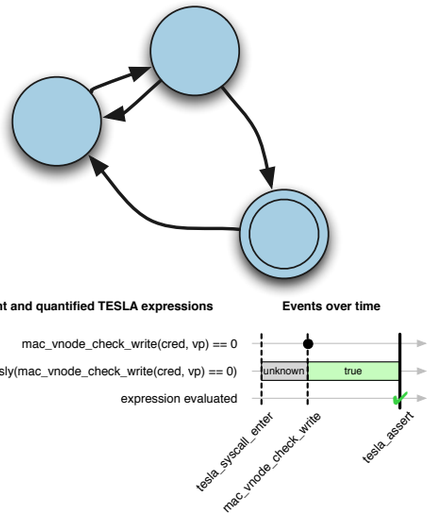


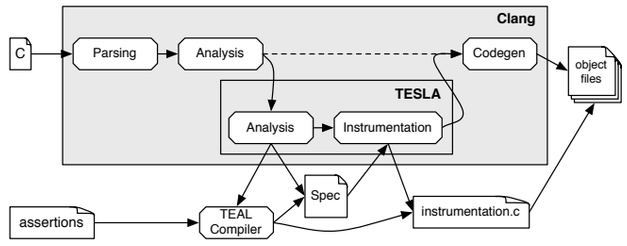
- Legacy application code compiled for general-purpose registers
- Hybrid code blending general-purpose registers and capabilities
- High-assurance capability-only code; stand-alone or in "pools of capabilities"
- Per-address space memory management and capability executive

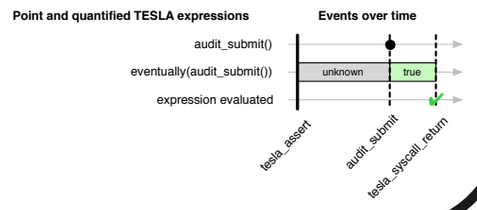
bluespec

TESLA assertion system

```
int
vn_rdwr(enum uio_rw, struct vnode *vp, ...,
  struct ucred *active_cred, struct ucred
  *file_cred, ...)
{
  if (rw == UIO_WRITE) {
    tassert_syscall(previously(mac_vnode_check_write(
      active_cred, file_cred, vp) == 0));
    tassert_syscall(eventually(audit_submit());
  ...
}
```







CTSRD continuously **validates security design principles** and **enforces application security structure** using a **formally grounded hybrid capability architecture**.

SRI International

UNIVERSITY OF CAMBRIDGE

Observations from Capsicum

- Software designs that employ the principle of least privilege are neither easily nor efficiently represented in current hardware.
- Kernels and programming language runtimes (TCBs) building directly on hardware in C are enormous and unsound.
- Software TCB implementations embody artifacts of security policies rather than design principles.

CTSRD elements

CHERI: capability hardware enhanced RISC instructions

Produce formally analysed Bluespec RISC soft core supporting granular protection via a hybrid capability model.

Develop separation kernel, hybrid Capsicum OS, and a future capability-enabled operating system.

TESLA: temporally enforced security logic assertions

Translate security principles from design into software implementation by blending temporal logic expressions and software / hardware-assisted runtime validation.

CTSRD research threads

- TESLA design and prototype
- BERI platform
- CHERI ISA and prototype
- PI meeting demonstration
- CHERI ISA formal methods
- Bluespec formal methods

The security assertion problem

- Good programmers enforce invariants through extensive use of software assertions
  - Tens of thousands of assertions in the FreeBSD kernel, as well as complex assertion infrastructures (WITNESS, …)
- In C/C++, assertions test **instantaneous** properties
- Interesting security properties are generally **temporal** (relating multiple points in time) rather than instantaneous!
  - Check before use, eventual audit, security meta-data cycles
  - Memory safety, protocol state machine conformance
- Programmers resort to manual instrumentation: **verbose**, **time-consuming**, and **error-prone**

Temporally Enhanced Security Logic Assertions (TESLA)

- Embed design-time security principles into executable code
- Borrow liberally from model checking, but as a dynamic technique:
  - Represent assertions as temporal logic or automata (TEAL)
  - Program events are symbols consumed by automata
  - Instantiate automata instances on demand, check for violations
- Assertions are tested on experienced paths rather than all paths
  - Continuous validation of principles by the program runtime
  - On failure: panic(), stack trace, DTrace events allowing scripting
- Potential for CHERI hardware assist via tightly coupled threads
- Combined with CHERI, checks can become "mandatory"

Since the PI meeting

- Ilias Marinos summer intern (continuing with TESLA during masters)
  - Conversion of external TESLA toolchain into clang plugins
  - Extensions to automata syntax to support more assertion types
  - Progress towards inline assertions
- Ben Laurie (Google) exploring TESLA support for OpenSSL: check that security APIs are being used correctly by consumers
- Bjoern Zeeb (FreeBSD) investigating TESLA support for the FreeBSD network stack: check TCP, ND6, IPSEC state machines
- Poster at the LLVM workshop in London; lots of industrial interest

TESLA next steps

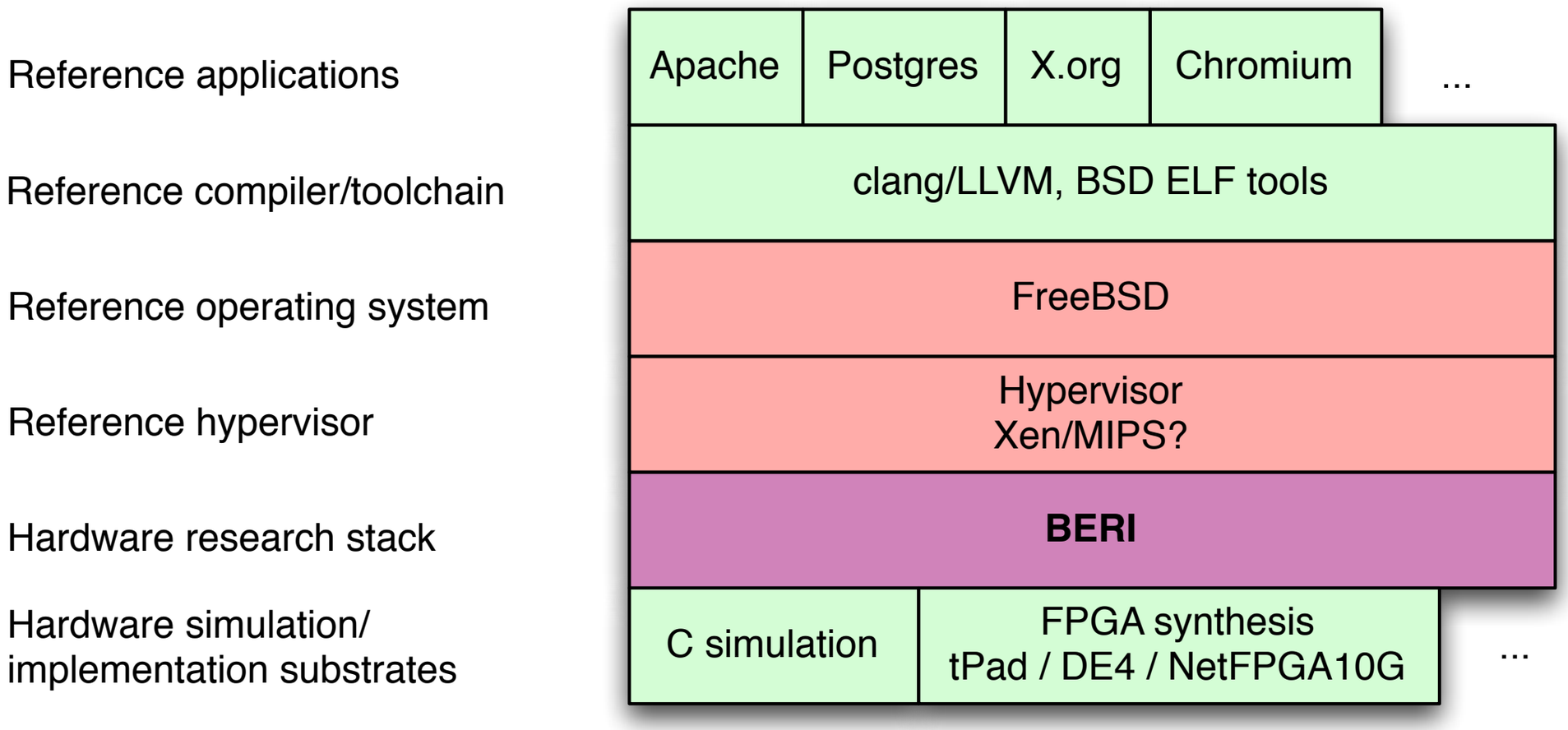
- Complete conversion to pure clang without external tools
- Revisit decision to use clang AST transform
  - Motivated by greater type awareness than afforded by LLVM IR
  - We may move instrumentation into LLVM and do analysis in clang
- Bring industry/open source visitors back to the Computer Laboratory for followups now that we support the syntax and features they require
  - Deploy in FreeBSD, Xen, OpenSSL communities
  - Site visits to various bay area companies in Spring 2012
- Add support for real time, distribution assertions

Bluespec extensible RISC implementation (BERI) and the hardware-software research problem

- Hardware, software, and network protocol researchers work in largely independent silos
  - Treat each others' corpuses as constants for experiments
- But we want to answer **multi-variable** research questions:
  - What happens as we vary both TLB size and OS strategy?
  - Was conflation of CPU memory virtualisation and protection a mistake?
  - What are the interactions of energy efficiency optimisation across both hardware and software?
  - How should "portable" OS message passing interfaces span a variety of hardware semantics?
- Create a reusable open source foundation from CHERI
- Recently presented idea at Barrelfish workshop in Cambridge
  - Immediate opportunities due to inadequate hardware and research platforms

BERI

Bluespec Extensible RISC Implementation



Complete hardware-software research platform

Apache/BSD-licensed from top to bottom

SRI International

UNIVERSITY OF CAMBRIDGE

BERI status

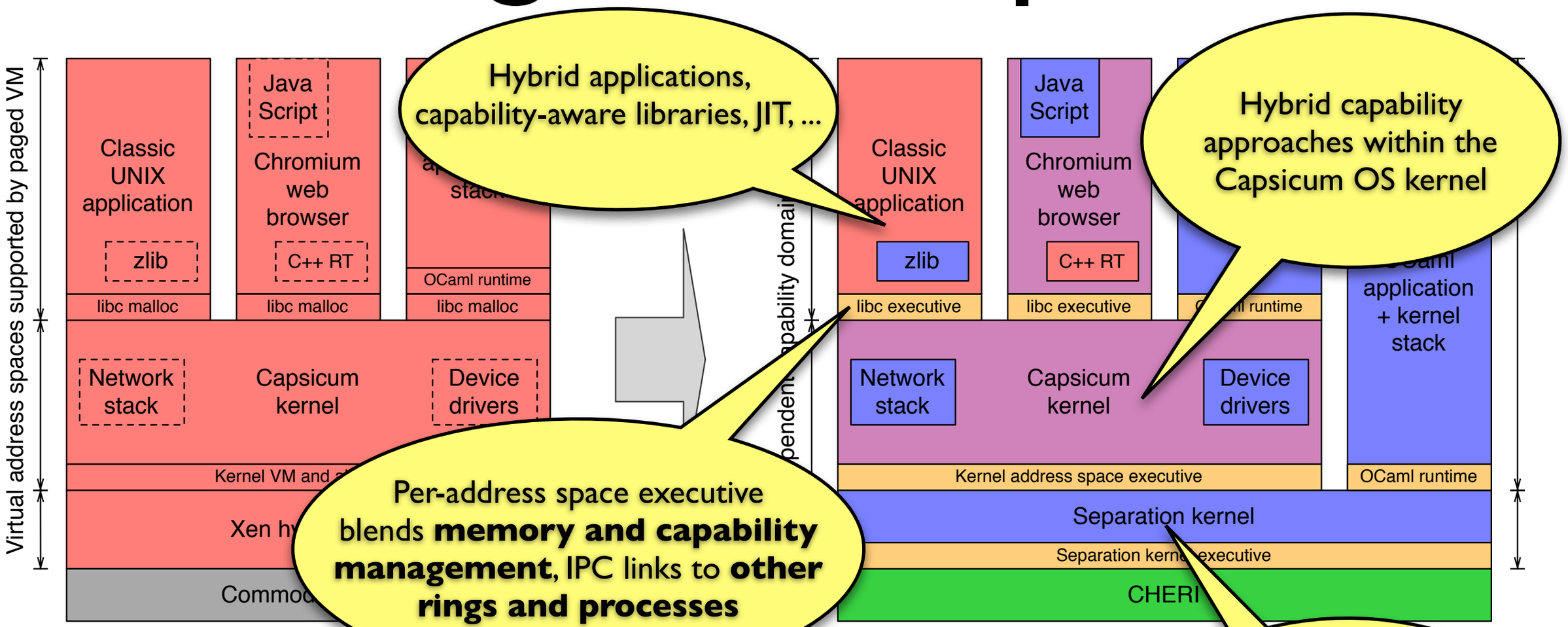
- Over the last year
  - Soft single-core 64-bit MIPS processor
  - Terasic DE-4, tPad - Altera FPGA + some peripherals
  - Commodity Uboot boot loader, research Deimos microkernel
- In progress
  - FreeBSD adaptation -- creeping up on single-user mode
  - 64-bit MIPS LLVM backend
  - First research project: CPU capability protection model
- Just starting on...
  - Multithreading, multicore; rack-scale memory interconnects
  - Port to NetFPGA 10G platform
  - FreeBSD device drivers for common Altera and Xilinx peripherals
- First open source release in Spring 2012 including OS/toolchain stack

Unusual OS port perspective: **fix hardware** rather than work around in software!

Capability Hardware Enhanced RISC Instructions (CHERI)

- Goal: **orders of magnitude** more protection domains than current CPUs to better support **fine-grained software compartmentalisation**
- Hardware enforcement of program protection structure
  - Memory capabilities (segments) and object capabilities
  - New capability registers, tagged memory
  - VM context switches → hardware message passing within an address space
  - RISC philosophy: minimal, compiler friendly hardware support to provide **efficient** and **debuggable** protection
- Capsicum's **hybrid capability model** principles
  - Run legacy code, capability code, and blends within a single process!
- Compiler changes: LLVM IR extensions to support protection

Evolving CHERI picture



CHERI activities

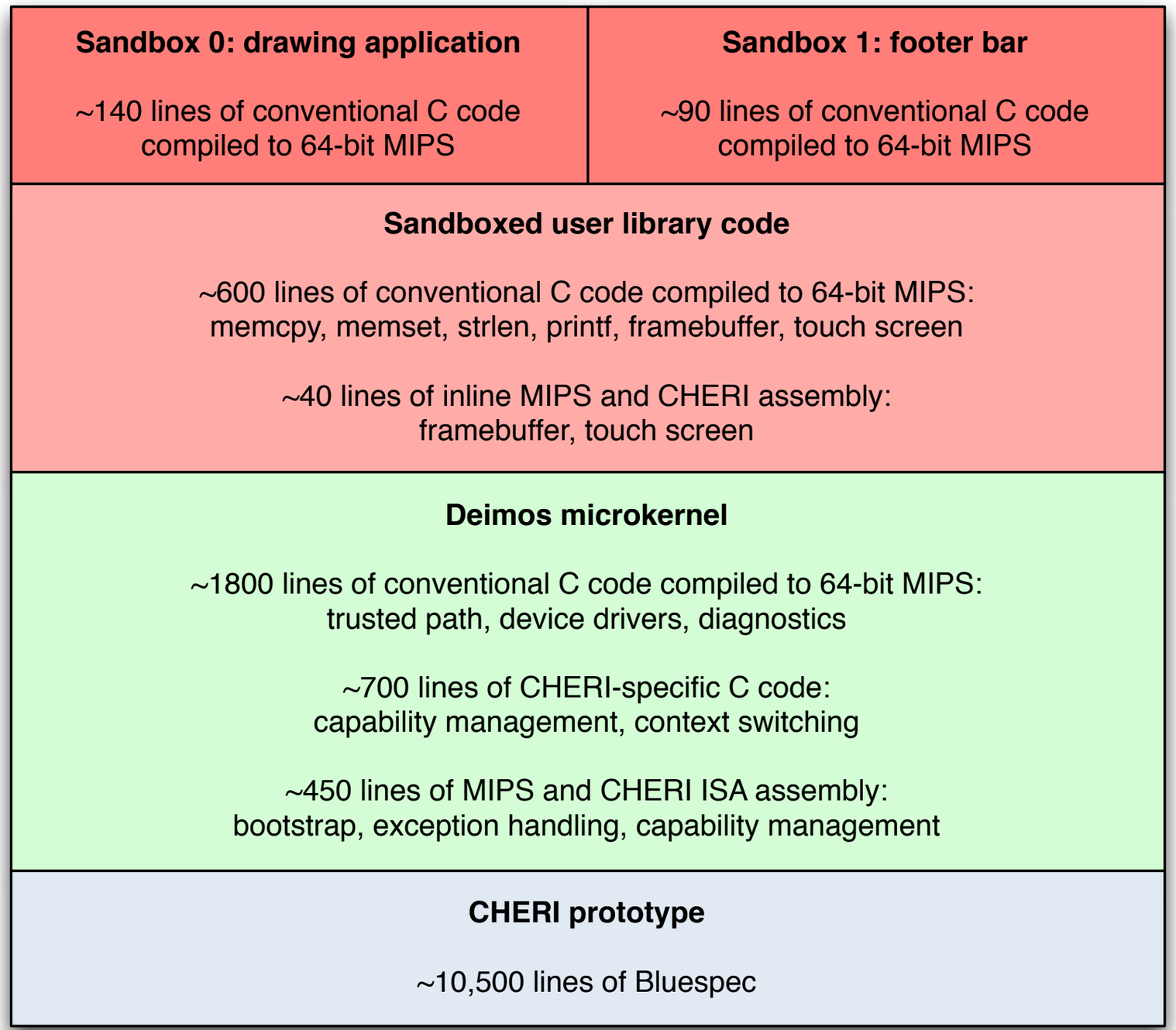
1. Develop hybrid capability ISA and hardware architecture supporting efficient
2. Prototype CHERI on BERI foundation
3. Investigate decomposition and minimisation of low-level TCBs: hypervisor, OS kernel, language runtimes, key applications
4. Map languages into new primitives
5. Pragmatically apply formal methods to HW and SW
6. Apply Capsicum's hybrid capability philosophy: short-term security benefits, long-term capability system vision

CHERI prototype

26 October 2011 - 2:00am BST

- **Deimos** capability microkernel in simulation and FPGA
  - Majority of Deimos is commodity MIPS ISA
  - Protection implemented using the capability coprocessor
- Capability-based separation between applications
  - Applications compiled from C into unaugmented MIPS ISA
  - Simple GUI-based applications: VGA display, touch screen
  - Hardware trusted path using capability model
- Demonstration
  - Capability-based delegation of hardware frame buffer regions
  - Early hybridised applications
  - Application code compiled from C into MIPS ISA
  - Portions of graphics library code, device drivers in CHERI ISA
- FreeBSD port remains "in progress" -- post-PI meeting focus

CHERI demo



- Single-core, pipelined CHERI synthesised in Altera FPGA
- Microkernel and apps
  - Deimos microkernel
  - Touchscreen drawing app
  - Footer bar application
- Capability-enforced sandboxing
- Trusted path for user I/O
- Hybrid capability model
  - 84% portable C
  - 10% CHERI assembly
  - 6% 64-bit MIPS assembly
  - Extended GNU assembler
  - Unmodified gcc

BERI platform goals

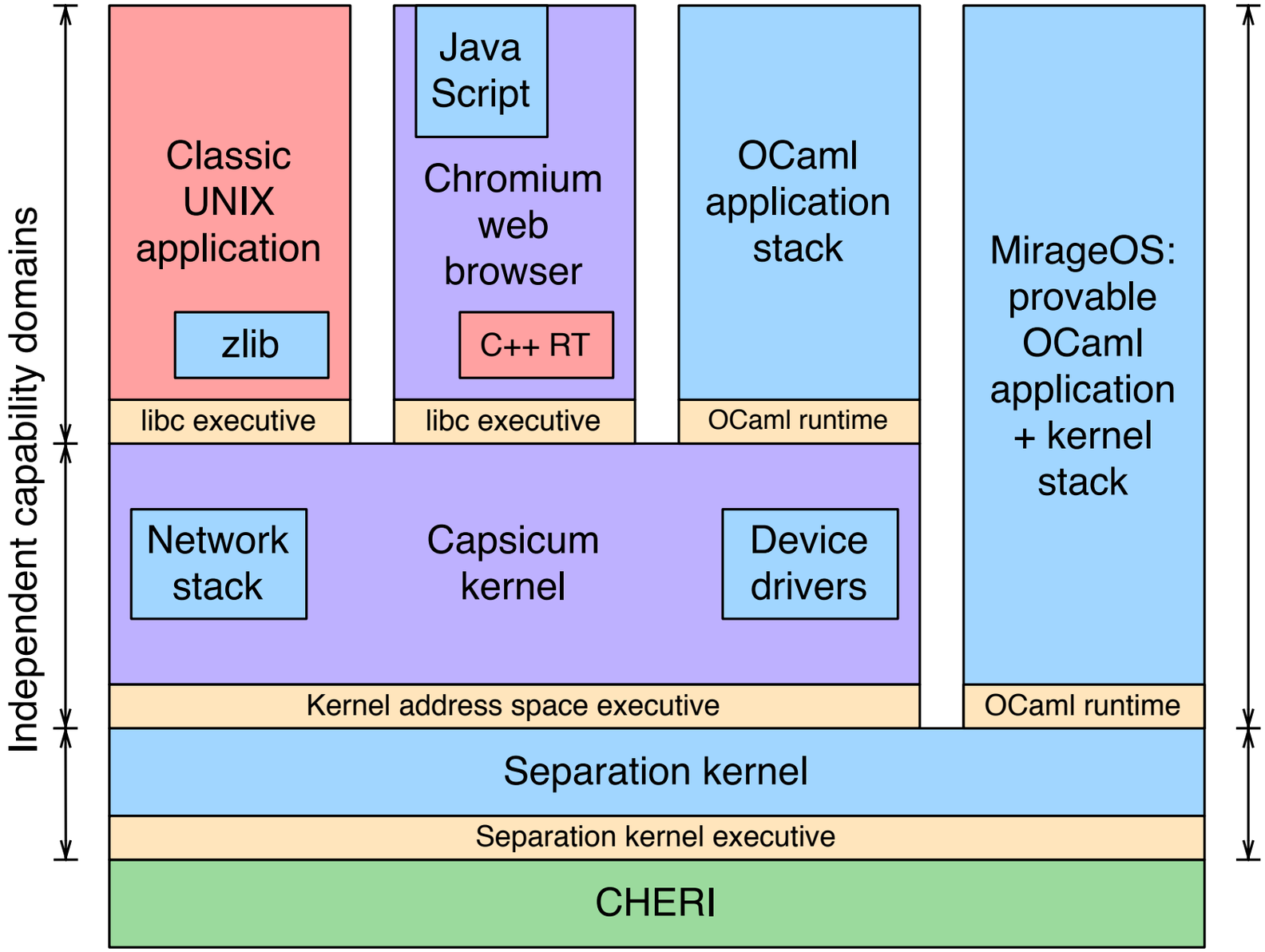
- Re-multithreading of prototype
- Basic multicore support
- Finish support for attaching to processor using GDB
- Finish 64-bit MIPS LLVM back end
- Netboot via Uboot
- FreeBSD booting is immediate post-PI meeting goal
- FreeBSD device drivers for additional peripherals
- Bring up the X server, Apache, and other applications

CHERI goals

- Finish FreeBSD multi-user mode support for CHERI
  - Boot-time configuration (FDT)
  - CHERI context switch code -- allow user applications to start to use capabilities!
    - Address space executive = run-time linker + memory allocation + gateway to system call interface
- Begin work on capability extensions to LLVM IR
- Experimentation with object capabilities in C
- Initial TLB vs. capability context switch performance analysis

A layered approach to formal methods with CHERI

- Prove higher-level software properties such as isolation
  - e.g., process isolation, object capability calling conventions
- Prove security properties of the capability mechanism in CHERI
  - e.g., nonforgeability, nonbypassability
- Establish the correctness of the capability ISA implementation with respect to the capability specification
  - e.g., nonalterability, atomic operation
- Establish the correctness of the general-purpose ISA implementation with respect to the MIPS specification
  - e.g., arithmetic, delay branches



Legacy application code compiled for general-purpose registers

Hybrid code blending general-purpose registers and capabilities

High-assurance capability-only code; stand-alone or "pools of capabilities"

Per-address space memory management and capability executive

ISA-layer verification

- Developed initial capability ISA specification hand-written Z
- Have recoded the specification in the PVS theorem prover to
  - prove security properties of instruction sequences and
  - execute the ISA specification
- Are now exploring a mechanical conversion from Bluespec into a model checker
  - We would like to show refinement (i.e., equivalence between unpipelined and pipelined implementations)
  - We would like to prove that the implementation implements the ISA

Bluespec verification

- Prove correspondence between the implementation and the various models
- Expose high-level internals of Bluespec compiler
- Extend the executable PVS model to work off Bluespec compiler intermediate output
  - Allow CHERI implementation to be checked directly
- Exercise tests generated automatically from the executable PVS model against the implemented hardware in the FPGA

Formally designing CHERI; design for verification

- We are not just building a design and then proving its properties
  - ISA documentation generated from formal specification
    - Goal of running our test suite on an executable form of the specification
    - Redesign in Bluespec underway to restructure the CPU to make proofs of refinement and correctness easier
- Improvements to the Bluespec compiler to support direct feeding into formal methods tools, including PVS and SAL

Collaboration

- Increasing industry, open source interest in TESLA and CHERI
- Direct engagement with Google Research
- Ongoing dialog within the CRASH programme
  - BAE/Harvard, UTexas Austin
- $(MRC)^2$: MRC contract to extend notions from CHERI into data centre switching

TESLA and BERI conclusions

- TESLA - temporal assertion system now a prototype
  - Engaging directly with open source and industry over last 6 months, and going forward
  - Larger-scale deployment over next 12 months
- BERI - Bluespec extensible RISC implementation
  - Invested a bit more time now in building CHERI to be generalisable
  - First open source release aiming for mid-2012

CHERI conclusions

- CHERI hybrid capability architecture: MIPS + capabilities
  - Bluespec-based prototype in simulation and synthesised to Altera-based boards
    - Port to NetFPGA 10G (Xilinx) in progress
    - Capability-aware separation kernel/executive now running (small) MIPS applications
- Exciting demonstration to show off separation
  - Demonstrates hybrid capability model: commodity MIPS code side-by-side with CHERI code

Q&A