State of the Union

Krste Asanovic
UC Berkeley, RISC-V Foundation, & SiFive Inc.
krste@berkeley.edu

1st RISC-V Summit
Santa Clara Convention Center, Santa Clara, CA
December 4, 2018

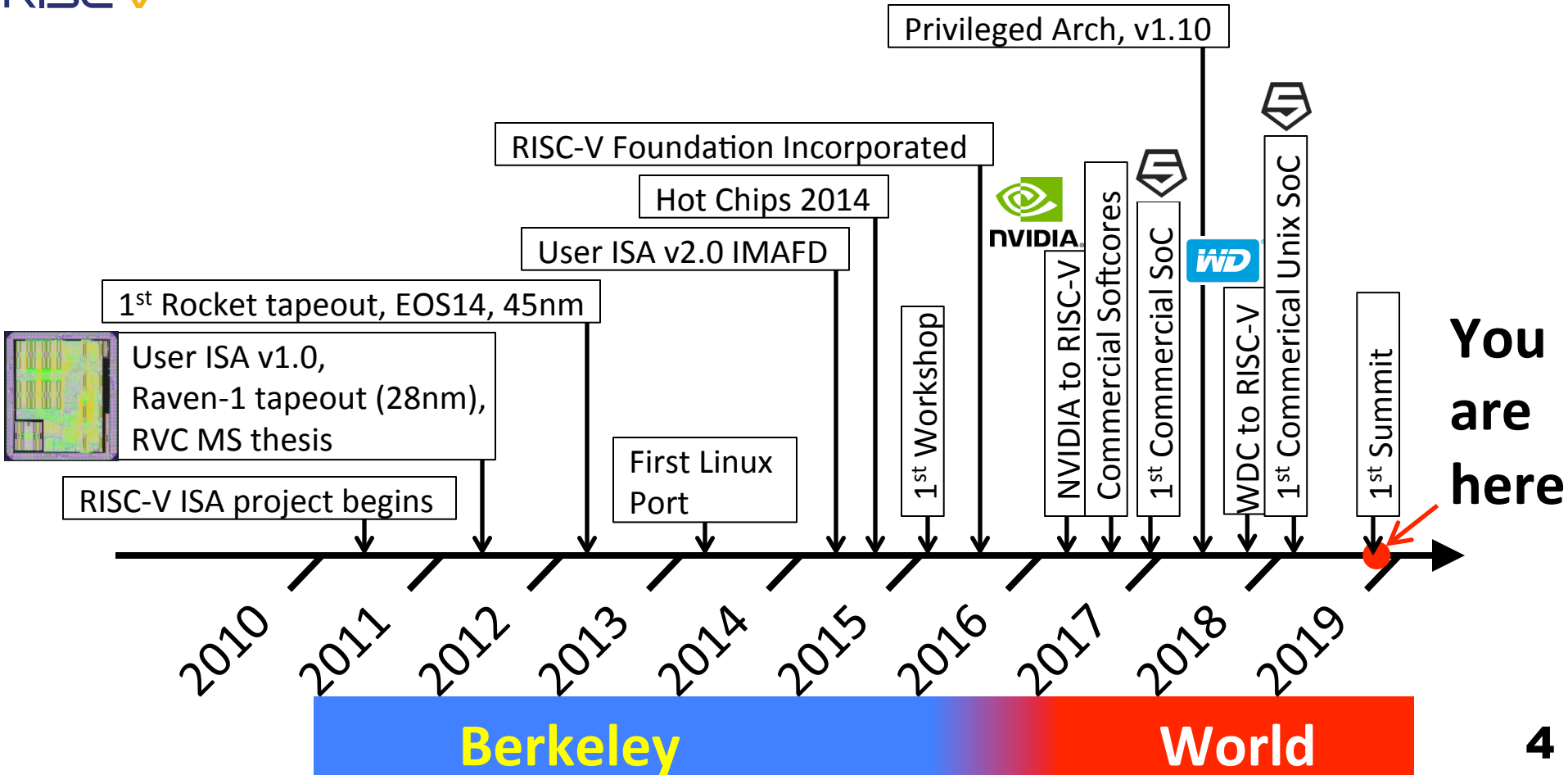
What is RISC-V?

- A high-quality, license-free, royalty-free RISC ISA specification originally from UC Berkeley
- Standard maintained by non-profit RISC-V Foundation
- Suitable for all types of computing system, microcontrollers to supercomputers
- Numerous proprietary and open-source cores
- Experiencing rapid uptake in industry and academia
- Supported by growing shared software ecosystem
- A work in progress…

What's Different about RISC-V?

- ***Simple***
  - Far smaller than other commercial ISAs
- ***Clean-slate design***
  - Clear separation between user and privileged ISA
  - Avoids µarchitecture or technology-dependent features
- A ***modular*** ISA designed for ***extensibility/specialization***
  - Small standard base ISA, with multiple standard extensions
  - Sparse and variable-length instruction encoding for vast opcode space
- ***Stable***
  - Base and standard extensions are frozen
  - Additions via optional extensions, not new versions
- ***Community designed***
  - With leading industry/academic experts and software developers

RISC-V Timeline



Why is RISC-V so popular?

- Engineers sometimes don't see forest for the trees
- The movement is not happening because some benchmark ran 10% faster, or some implementation was 30% lower power
- The movement is happening because new business model changes everything
  - Pick ISA first, then pick vendor or build own core
  - Add your own extension without getting permission
- Implementation features/PPA will follow
  - Whatever is broken/missing in RISC-V will get fixed

Modest RISC-V Project Goal

Become the industry-standard ISA for all computing devices

So, how's it going?

RISC-V Ecosystem

Software

Open-source software:
Gcc, binutils, glibc, Linux, BSD, LLVM, QEMU, FreeRTOS, ZephyrOS, LiteOS, SylixOS, …

Commercial software:
Lauterbach, Segger, Micrium, ExpressLogic, …

RISC-V Foundation

ISA specification | Golden Model | Compliance

Hardware

Open-source cores:
Rocket, BOOM, RI5CY, Ariane, PicoRV32, Piccolo, SCR1, Hummingbird, …

Commercial core providers:
Andes, Bluespec, Cloudbear, Codasip, Cortus, C-Sky, Nuclei, SiFive, Syntacore, …

Inhouse cores:
Nvidia, +others

ISA Specification Ratification

- Updated version of original Berkeley unprivileged ISA standards (RV32I/RV64I+MAFDQC) now under public review for ratification
  - Public review closes Friday Dec 21st, 2018
- Memory model (RVWMO) ratified!
- Debug spec 0.13 closed public review, awaiting BoD approval
- Privileged arch v1.11 begins public review today!
  - Public review closes Jan 25th, 2019
- Core of ISA ratified as Foundation standard for 2019

Unprivileged ISA changes vs. 2014 spec

- Memory model(s): RVWMO, RVTSO extension
- FP NaN representation, NaN boxing, FMIN/FMAX
- CSR instructions moved out of I (now Zicsr)
- FENCE.I moved out of I (not Zifencei)

- Much improved commentary
- Platform mandates removed

Golden Model

- Original Spike C model replaced by formal model
- Formal spec group complete several formal models of sequential execution in December
- Developing formal “Feature Model” as way of managing dependencies among ISA options
- Follow-on formal model group to address concurrency and interaction with RVWMO

Compliance Suites

- Improving framework for test suite
- More companies involved in delivering components
- Official RFP in process to complete engineering work

- Gary Crumley (WD) will be program manager for this work
- Collaboration with Linux Foundation on project

Foundation Board of Directors' Award 2018

- Daniel Lustig, NVIDIA
- "For his leadership and technical contributions in developing the RISC-V memory consistency model"

Fragmentation versus Diversity

- Fragmentation, doing the same thing in multiple different ways

- Diversity, supporting many use cases
  - RISC-V designed to be modular, support specialization
  - Especially important in embedded applications

Ongoing ISA extensions work

- Fast interrupts (CLIC) – stable, ratifiable in 2019?
- V extension (Vectors) – getting stable, ratifiable in 2019?
- Hypervisor extension – stable, ratifiable in 2019?
- Crypto (builds on vectors) – builds on vectors
- TEE (trusted execution environments) – refocused
- B extension (Bit Manipulation) – recently restarted
- J extension (dynamic translation support) – ongoing discussion
- P extension (DSP extensions) – recently started
- Sv128 (Large secure address space) – recently started

Standards Expectation Management

- Existing base + standard extensions, 8.5 years in development, only now being ratified
- Large ISA modules take time and expert input, experts disagree, community seeks perfection
  - Debug took nearly 3 years, more to do on top of base
  - Vectors 3+ years in process
- Small updates/fixes can happen faster
- We will need to update our processes to cope with real standards life cycle

ISA Standards Timeline

1. Define need
2. Gather inputs, hold meetings, whiteboard-level design arguments, finish stable draft
3. Build implementation according to draft
4. Build software for implementation
5. Evaluate, iterate from step 2 (maybe many times)
6. Community agrees to push for ratification
7. Ratification
8. Deployment / Compliance testing
9. Replacement

Prioritizing ISA Extension Work

- Will implementers build it?
- Will this increase RISC-V design wins?
- Does this fill a hole in RISC-V capability?
- Does this standardize something being done incompatibly today?
- Does more than one member care about it?
- Are there capable volunteers to work on it?

ISA Standards Priorities for 2019

- Develop implementations and software stacks for CLIC, vector, hypervisor
- Unix platform standard
  - Server platform, SBI, UEFI/ACPI?, security features, …
- Embedded ABI
  - Reduce library code size, reduce interrupt latency
- Zfinx
  - Embedded systems with floating-point out of X registers
- Improve code compression
  - Compilers, libraries, additional compressed instructions

Embedded ABI

- Original ABI developed for Unix platforms
- Embedded ABI goals
  - reduce interrupt latency by reducing argument registers, caller-saved registers
  - reduce code size by dropping 128b FP support
  - Provide compatible interface are RV32E and RV32I
- Proposal in prep, to start in 2019

Zfinx “Float in X registers”

- Intended for embedded platforms
- Removes f registers from ISA
- Floating-point operations take values from x registers
- Supported for both E and I base ISAs

- Reduces system size, interrupt latency, simplifies ABI
- Frees up floating-point load/store opcode space in compressed extension

Improving Embedded Code Size

- Better documentation/ white papers on existing techniques
- Improved library code for code size reduction, including new embedded ABI
- Make compilers aware of compressed instructions
- Further compiler optimizations
- Repurpose FP load/store encodings for Zfinx
- Other new instructions

Summary RISC-V into 2019

- Core standards almost all ratified
- Software ecosystem filling out rapidly
- Time to standardize platform profiles
- Lots of work ahead