The BOOM Project

Christopher Celio, **Jerry Zhao, Abraham Gonzalez, Ben Korpan,**
Krste Asanovic, David Patterson

Outline

- **Current state of the project**
- How to get started
- How to contribute
- Things to work on
- Research ideas
- Conclusion and discussion

Project Maintainers

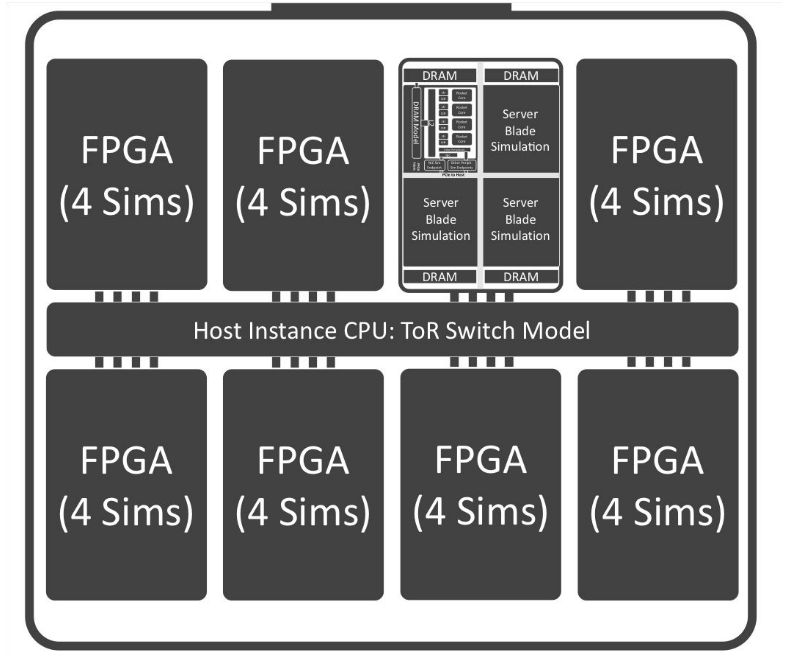
- Chris Celio is now at Esperanto Technologies
  - Works on the ET-Maxion out-of-order core
- Work at Berkeley
  - Jerry Zhao - Senior undergraduate
    - Maintenance + Keystone support + vectors
  - Abe Gonzalez, Ben Korpan - First-year graduate students
    - Ring Scalar design of BOOM
    - Keystone and security additions
    - General maintenance + Documentation updates

ADEPT

Berkeley Architecture Research

Recent/Ongoing Work

- Taped out as BROOM
- Integrated with FireSim - a open-source FPGA-accelerated simulation tool that runs on Amazon EC2 F1
  - Boot Linux on a multi-core BOOM with 16 GB DDR3, UART, Ethernet NIC in the cloud for 50 cents/hour at ~100 MHz
- Work towards BOOMv3 release
  - Memory model
  - RVC
  - Performance debugging
  - And more!



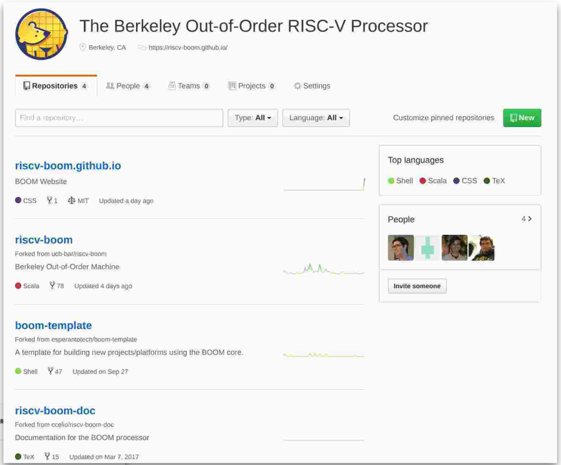
FireSim 32 Node Rack (128 Cores)

Outline

- Current state of the project
- **How to get started**
- How to contribute
- Things to work on
- Research ideas
- Conclusion and discussion

Getting Started

- Central location of information is the BOOM website and GitHub organization
  - Website: riscv-boom.github.io
  - GitHub: github.com/riscv-boom
- Website contents
  - Link to GitHub organization
  - BOOM v2 Specification
    - 70 pages of documentation on microarchitecture, future work, etc
  - Publications/Talks/Reports
  - Current team members



The Berkeley Out-of-Order RISC-V Processor

Berkeley, CA https://riscv-boom.github.io/

riscv-boom.github.io
BOOM Website
● CSS 1 MIT Updated a day ago

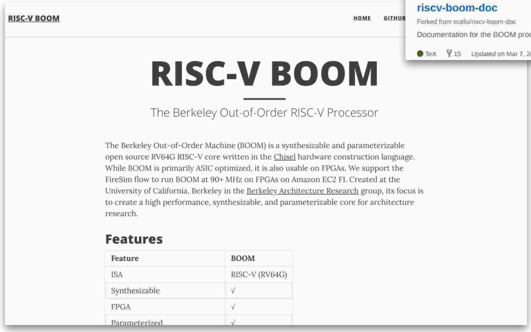
riscv-boom
Forked from ucb-bar/riscv-boom
Berkeley Out-of-Order Machine
● Scala 78 Updated 4 days ago

boom-template
Forked from esperantotech/boom-template
A template for building new projects/platforms using the BOOM core.
● Shell 47 Updated on Sep 27

riscv-boom-doc
Forked from ccelio/riscv-boom-doc
Documentation for the BOOM processor
● TeX 15 Updated on Mar 7, 2017

Top languages
● Shell ● Scala ● CSS ● TeX

People 4



RISC-V BOOM

The Berkeley Out-of-Order RISC-V Processor

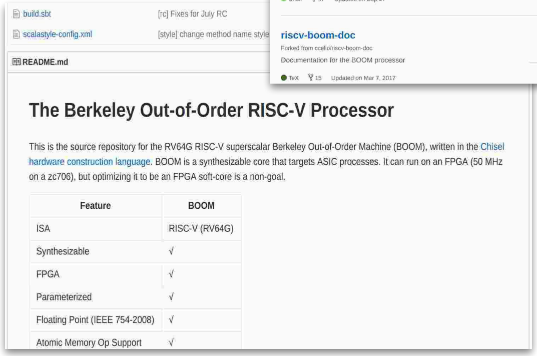
The Berkeley Out-of-Order Machine (BOOM) is a synthesizable and parameterizable open source RV64G RISC-V core written in the Chisel hardware construction language. While BOOM is primarily ASIC optimized, it is also usable on FPGAs. We support the FireSim flow to run BOOM at 90+ MHz on FPGAs on Amazon EC2 F1. Created at the University of California, Berkeley in the Berkeley Architecture Research group, its focus is to create a high performance, synthesizable, and parameterizable core for architecture research.

Features

Feature	BOOM
ISA	RISC-V (RV64G)
Synthesizable	√
FPGA	√
Parameterized	√

BOOM GitHub Organization

- riscv-boom
  - The core source code
  - Note: That elements of RocketChip are reused
- boom-template
  - Template for building new projects with the BOOM core
  - Includes all necessary materials to build/run the core
- riscv-boom.github.io
  - Project website
- riscv-boom-doc
  - Documentation for the core, project tools, etc



build.sbt [rc] Fixes for July RC

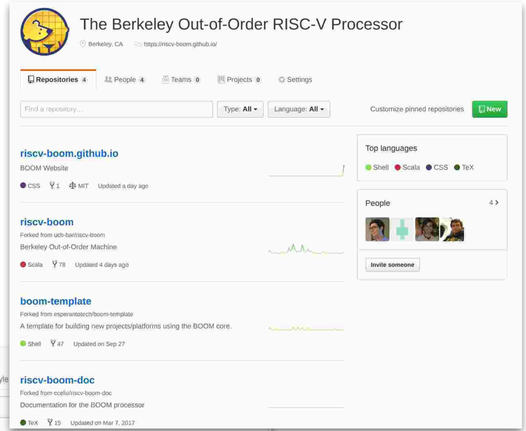
scalastyle-config.xml [style] change method name style

README.md

The Berkeley Out-of-Order RISC-V Processor

This is the source repository for the RV64G RISC-V superscalar Berkeley Out-of-Order Machine (BOOM), written in the Chisel hardware construction language. BOOM is a synthesizable core that targets ASIC processes. It can run on an FPGA (50 MHz on a zc706), but optimizing it to be an FPGA soft-core is a non-goal.

Feature	BOOM
ISA	RISC-V (RV64G)
Synthesizable	√
FPGA	√
Parameterized	√
Floating Point (IEEE 754-2008)	√
Atomic Memory Op Support	√



The Berkeley Out-of-Order RISC-V Processor

Berkeley, CA https://riscv-boom.github.io/

riscv-boom.github.io
BOOM Website
● CSS 1 MIT Updated a day ago

riscv-boom
Forked from ucb-bar/riscv-boom
Berkeley Out-of-Order Machine
● Scala 78 Updated 4 days ago

boom-template
Forked from esperantotech/boom-template
A template for building new projects/platforms using the BOOM core.
● Shell 47 Updated on Sep 27

riscv-boom-doc
Forked from ccelio/riscv-boom-doc
Documentation for the BOOM processor
● TeX 15 Updated on Mar 7, 2017

Top languages
● Shell ● Scala ● CSS ● TeX

BOOM v2 Specification

- Documentation on
  - Core units/stages
    - Most detailed implementation resource outside of code
  - Repository layout and reused resources from RocketChip
  - Verification
  - Tools and tests
    - Using gem5 pipeline viewer
    - Torture tests and RISC-V tests
  - Physical realization information
  - Future work
    - Includes preliminary design ideas for each piece
- Will look to change to ReadTheDocs format

The Berkeley Out-of-Order Machine (BOOM) Design Specification

Christopher Celio, David Patterson, and Krste Asanović
University of California, Berkeley, California 94720–1770
celio@eecs.berkeley.edu

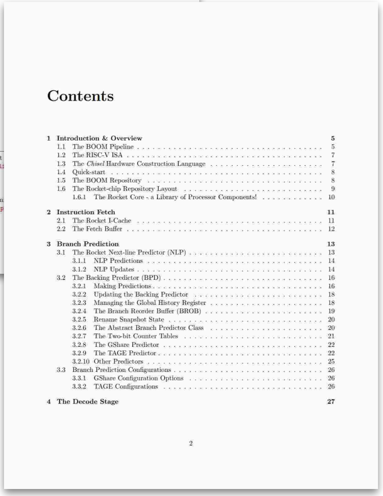
December 4, 2016

This draft is a work-in-progress.

The information in this publication is subject
This document is available at: https://cceli

Copyright © 2016 Christopher Celio

This work is licensed under the Creative Com
4.0). To view a copy of this license, visit http



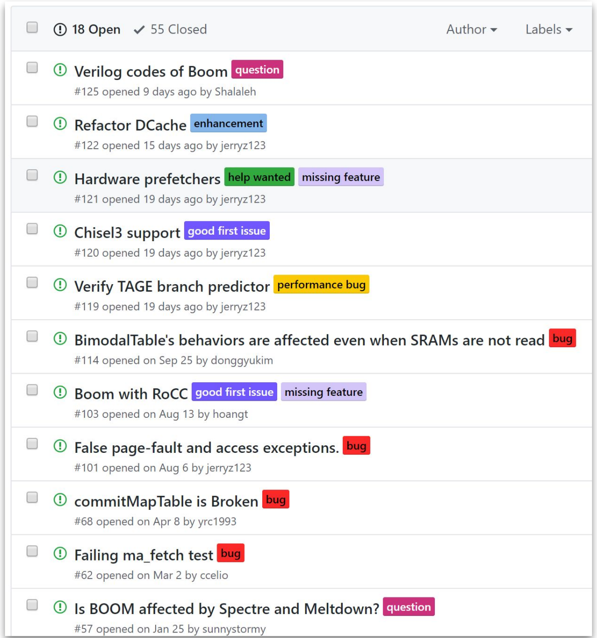
Contents

2

Outline

Contribute to the project

- Bug reports
  - Verify against Spike ISA simulator
  - Add issue with reproducible code snippets
- Feature Additions
  - Reach out to the Berkeley team to discuss major changes
    - Keep BOOM consistent
    - Avoid conflicting work
    - Bounce around ideas, get help, etc
  - Submit a pull request (PR)
    - Detailed description of changes
    - Don't forget to update documentation alongside code



Verilog codes of Boom question
#125 opened 9 days ago by Shalaleh

Refactor DCache enhancement
#122 opened 15 days ago by jerryz123

Hardware prefetchers help wanted missing feature
#121 opened 19 days ago by jerryz123

Chisel3 support good first issue
#120 opened 19 days ago by jerryz123

Verify TAGE branch predictor performance bug
#119 opened 19 days ago by jerryz123

BimodalTable's behaviors are affected even when SRAMs are not read bug
#114 opened on Sep 25 by donggyukim

Boom with RoCC good first issue missing feature
#103 opened on Aug 13 by hoangt

False page-fault and access exceptions. bug
#101 opened on Aug 6 by jerryz123

commitMapTable is Broken bug
#68 opened on Apr 8 by yrc1993

Failing ma_fetch test bug
#62 opened on Mar 2 by ccelio

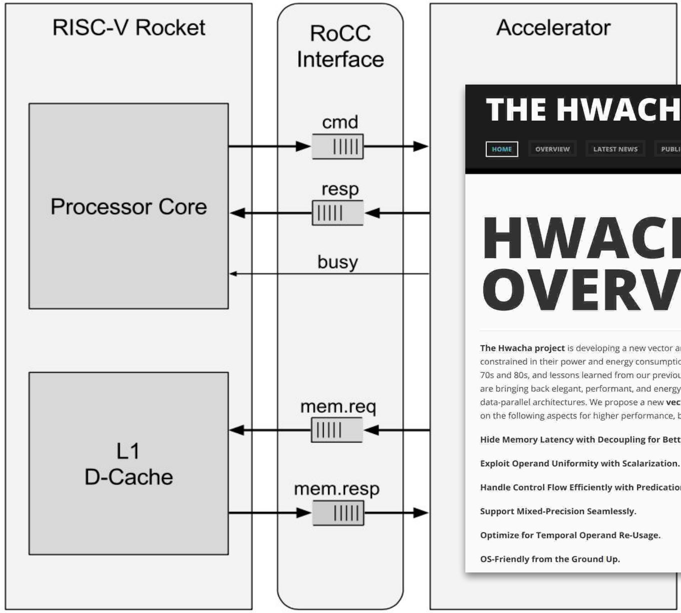
Is BOOM affected by Spectre and Meltdown? question
#57 opened on Jan 25 by sunnystormy

Outline

- Current state of the project
- How to get started
- How to contribute
- **Things to work on**
- Research ideas
- Conclusion and discussion

RoCC Support

- Enable support for co-processors and accelerators
  - Hwacha vector accelerator
  - Crypto units, DMA engines
  - + Your custom block
- Not trivial to implement
  - Adapt RoCC to an OoO machine
  - Where do instruction bits live?
  - One-at-a-time, stall until RoCC instruction is ready to commit
  - BoCC?



HWACHA OVERVIEW.

The Hwacha project is developing a new vector architecture for future computer systems that are constrained in their power and energy consumption. Inspired by traditional vector machines from the 70s and 80s, and lessons learned from our previous vector-thread architectures Scale and Maven, we are bringing back elegant, performant, and energy-efficient aspects of vector processing to modern data-parallel architectures. We propose a new **vector-fetch architectural paradigm**, which focuses on the following aspects for higher performance, better energy efficiency, and lower complexity.

Hide Memory Latency with Decoupling for Better Efficiency.

Exploit Operand Uniformity with Scalarization.

Handle Control Flow Efficiently with Predication.

Support Mixed-Precision Seamlessly.

Optimize for Temporal Operand Re-Usage.

OS-Friendly from the Ground Up.

Memory System Changes

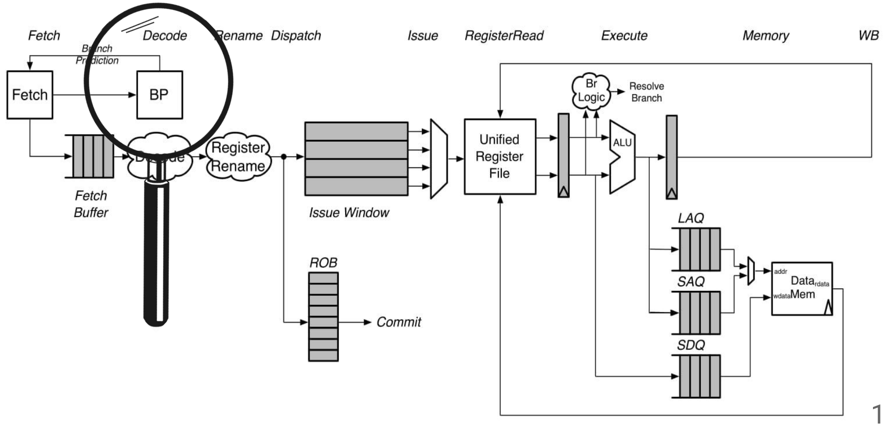
- Total Store Ordering and Weak Memory Ordering
  - Currently implements the relaxed consistency model
  - Update to support RVWMO, order loads to the same address
- Prefetchers
- Memory Disambiguation Predictor
  - Predict dependencies between different memory accesses
- Multi-ported data-cache
  - Commit multiple loads per cycle

RISC-V Compressed Instructions

- Enables smaller 16b encodings of common instructions
  - All 16b instructions map to a 32b longer instruction
  - 32b instructions can now be aligned to the half-word boundary
- Challenges
  - A few 16b instructions exhibit different behaviour than 32b instructions
    - Ex. JAL may return PC+2 or PC+4 depending on which version of the instruction it was
    - Must track if it was a 16b/32b instruction through the pipeline
  - Require code changes in several structures
    - Reorder Buffer (ROB)
    - Branch Predictor
    - More!

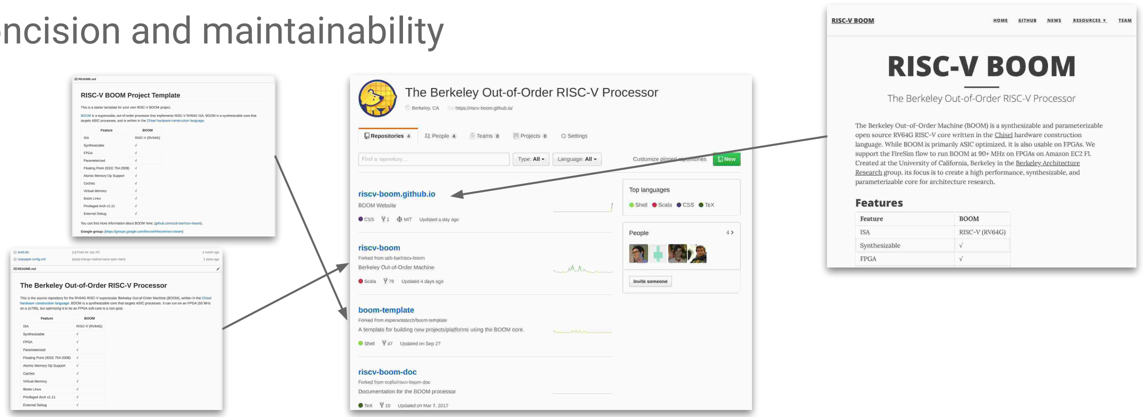
Verification

- Tests
  - Unit tests for particular sub-components/stages
  - More coverage on pipeline
- Add co-simulation of commit trace against an ISA simulator
- Initial things to verify
  - Specification compliance
  - Branch Prediction
    - TAGE
    - GShare
  - Much more!
- Work on improving simulation visibility



General maintenance

- Documentation additions
  - Update documentation for new additions
  - Transition to ReadTheDocs
- Centralized GitHub organization updates
  - Move all relevant BOOM repo’s here for easy access and visibility
- Refactoring
  - Update code for clarity, concision and maintainability



README.md

RISC-V BOOM Project Template

This is a starter template for your own RISC-V BOOM project.

BOOM is a superscalar, out-of-order processor that implements RISC-V RV64G ISA. BOOM is a synthesizable core that targets ASIC processes, and is written in the Chisel hardware construction language.

Feature	BOOM
ISA	RISC-V (RV64G)
Synthesizable	√
FPGA	√
Parameterized	√
Floating Point (IEEE 754-2008)	√
Atomic Memory Op Support	√
Caches	√
Virtual Memory	√
Boots Linux	√
Privileged Arch v1.11	√
External Debug	√

You can find more information about BOOM here: (github.com/ucb-bar/riscv-boom).

Google group: (https://groups.google.com/forum/#!forum/riscv-boom)

build.sbt	[rc] Fixes for July RC	a month ago
scalastyle-config.xml	[style] change method name style check	2 years ago

README.md

The Berkeley Out-of-Order RISC-V Processor

This is the source repository for the RV64G RISC-V superscalar Berkeley Out-of-Order Machine (BOOM), written in the Chisel hardware construction language. BOOM is a synthesizable core that targets ASIC processes. It can run on an FPGA (50 MHz on a zc706), but optimizing it to be an FPGA soft-core is a non-goal.

Feature	BOOM
ISA	RISC-V (RV64G)
Synthesizable	√
FPGA	√
Parameterized	√
Floating Point (IEEE 754-2008)	√
Atomic Memory Op Support	√
Caches	√
Virtual Memory	√
Boots Linux	√
Privileged Arch v1.11	√
External Debug	√

The Berkeley Out-of-Order RISC-V Processor

Berkeley, CA https://riscv-boom.github.io/

riscv-boom.github.io
BOOM Website
● CSS 1 MIT Updated a day ago

riscv-boom
Forked from ucb-bar/riscv-boom
Berkeley Out-of-Order Machine
● Scala 78 Updated 4 days ago

boom-template
Forked from esperantotech/boom-template
A template for building new projects/platforms using the BOOM core.
● Shell 47 Updated on Sep 27

riscv-boom-doc
Forked from ccelio/riscv-boom-doc
Documentation for the BOOM processor
● TeX 15 Updated on Mar 7, 2017

Top languages: ● Shell ● Scala ● CSS ● TeX

People 4

RISC-V BOOM

RISC-V BOOM

The Berkeley Out-of-Order RISC-V Processor

The Berkeley Out-of-Order Machine (BOOM) is a synthesizable and parameterizable open source RV64G RISC-V core written in the Chisel hardware construction language. While BOOM is primarily ASIC optimized, it is also usable on FPGAs. We support the FireSim flow to run BOOM at 90+ MHz on FPGAs on Amazon EC2 F1. Created at the University of California, Berkeley in the Berkeley Architecture Research group, its focus is to create a high performance, synthesizable, and parameterizable core for architecture research.

Features

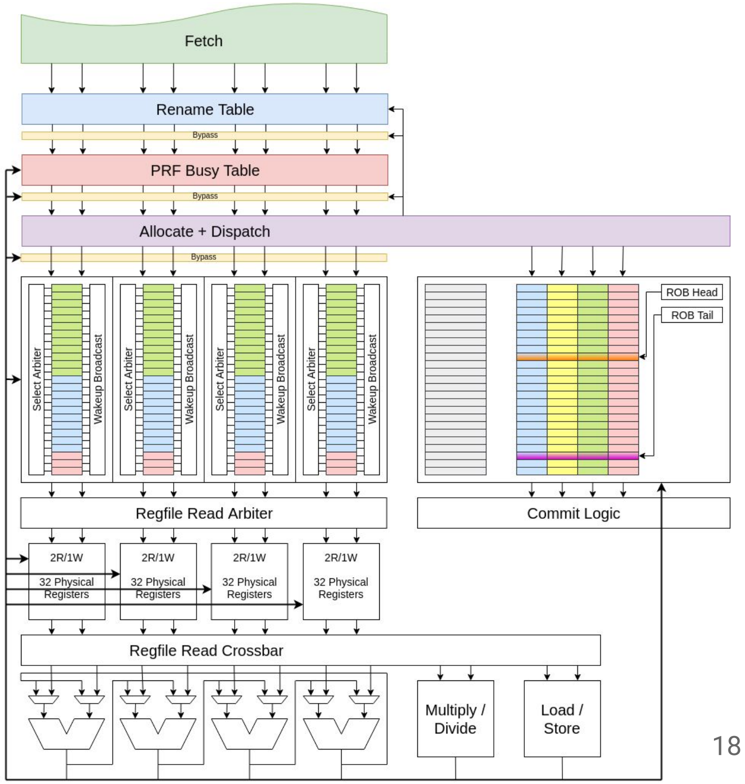
Feature	BOOM
ISA	RISC-V (RV64G)
Synthesizable	√
FPGA	√

Outline

- Current state of the project
- How to get started
- How to contribute
- Things to work on
- **Research ideas**
- Conclusion and discussion

Microarchitectural Exploration

- Chisel + RISC-V ecosystem enables agile architecture research
  - Don’t settle on modelling; build and evaluate fully functional systems!
- BOOM is an important piece of groundwork
  - Can be leveraged to research high performance microarchitecture
  - 15,000+ lines of concise, partitioned code: feasible for one person to implement features
- Current ideas:
  - “RingScalar” integer execution core
  - Support for RISC-V vector extension



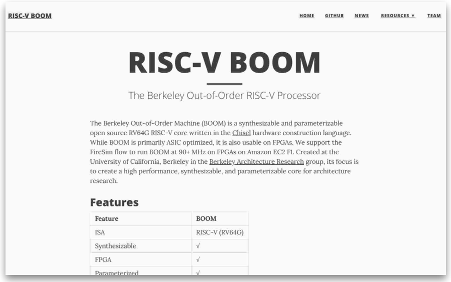
Hardware Security

- Good news? Not vulnerable to Meltdown or Foreshadow
- Bad news? Vulnerable to general Spectre style attacks
  - Actually good news
- BOOM - baseline for security research
- Leverage Chisel + RocketChip productivity to explore mitigations
  - Secure caches
  - Cache partitioning
  - Timing obfuscation
- A secure cache proof-of-concept is in the works
  - 1 man-week to block transient cache refills
- Enclaves are cool too

Outline

Conclusion

- Berkeley Out-of-Order Machine
  - Concisely expressed in ~15k LOC Chisel
  - RV64G + Priv v1.11
  - Synthesizable
  - Extensible
  - OPEN SOURCE!
- A lot of work to be done
  - Verification
  - Useful features
- Great opportunity to do make an impact
  - Architecture research
  - Work on an open source project



RISC-V BOOM

The Berkeley Out-of-Order RISC-V Processor

The Berkeley Out-of-Order Machine (BOOM) is a synthesizable and parameterizable open source RV64G RISC-V core written in the Chisel hardware construction language. While BOOM is primarily ASIC optimized, it is also usable on FPGAs. We support the FireSim flow to run BOOM at 90+ MHz on FPGAs on Amazon EC2 F1. Created at the University of California, Berkeley in the Berkeley Architecture Research group, its focus is to create a high performance, synthesizable, and parameterizable core for architecture research.

Features

Feature	BOOM
ISA	RISC-V (RV64G)
Synthesizable	√
FPGA	√
Parameterized	√

What do you want to do?

What would you like to see in BOOM and how can we help?

Questions and thanks!

- Get started today
  - Website: riscv-boom.github.io
  - GitHub: riscv-boom/riscv-boom
- Contact us
  - Google group: https://groups.google.com/forum/#!forum/riscv-boom

- Thanks to all who have worked on BOOM
  - Pi-Feng Chiu, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Henry Cook, Palmer Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, Jack Koenig, Jim Lawson, Yunsup Lee, Richard Lin, Eric Love, Martin Maas, Chick Markley, Albert Magyar, Howard Mao, Miquel Moreto, Quan Nguyen, Albert Ou, Brian Richards, Colin Schmidt, Wenyu Tang, Stephen Twigg, Huy Vo, Andrew Waterman, Angie Wang, and more

Acknowledgements

- Research partially funded by DARPA Award Number HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and ASPIRE and ADEPT Lab industrial sponsors and affiliates Intel, Google, Siemens, SK Hynix, Seagate, Huawei, Nokia, NVIDIA, Oracle, and Samsung. Approved for public release; distribution is unlimited. The content of this presentation does not necessarily reflect the position or the policy of the US government and no official endorsement should be inferred. Any opinions, findings, conclusions, or recommendations in this paper are solely those of the authors and does not necessarily reflect the position or the policy of the sponsors.

- Projects mentioned
  - FireSim: Website: https://fires.im/ Paper: sagark.org/assets/pubs/firesim-isca2018.pdf
  - Hwatcha: Website: http://hwacha.org/
  - RingScalar: Paper: https://dspace.mit.edu/handle/1721.1/34012