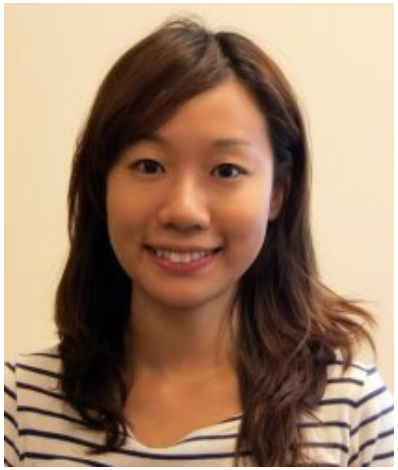
# BOOM v2

## an open-source out-of-order RISC-V core

Christopher Celio, Pi-Feng Chiu, Borivoje Nikolic, David Patterson, Krste Asanović

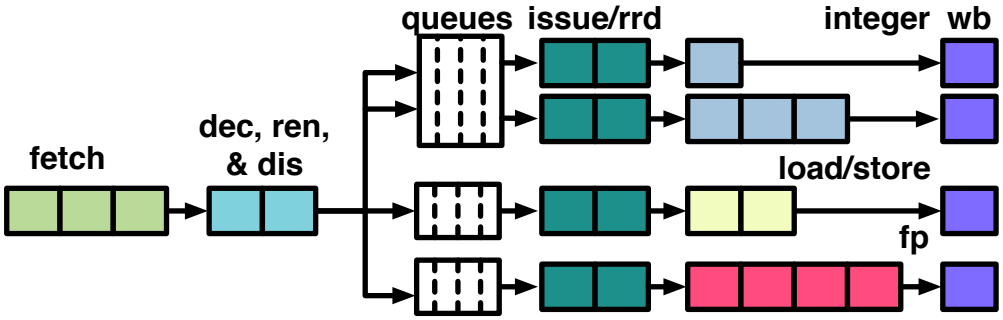
**2017 RISC-V Workshop #7**




Berkeley
Architecture
Research

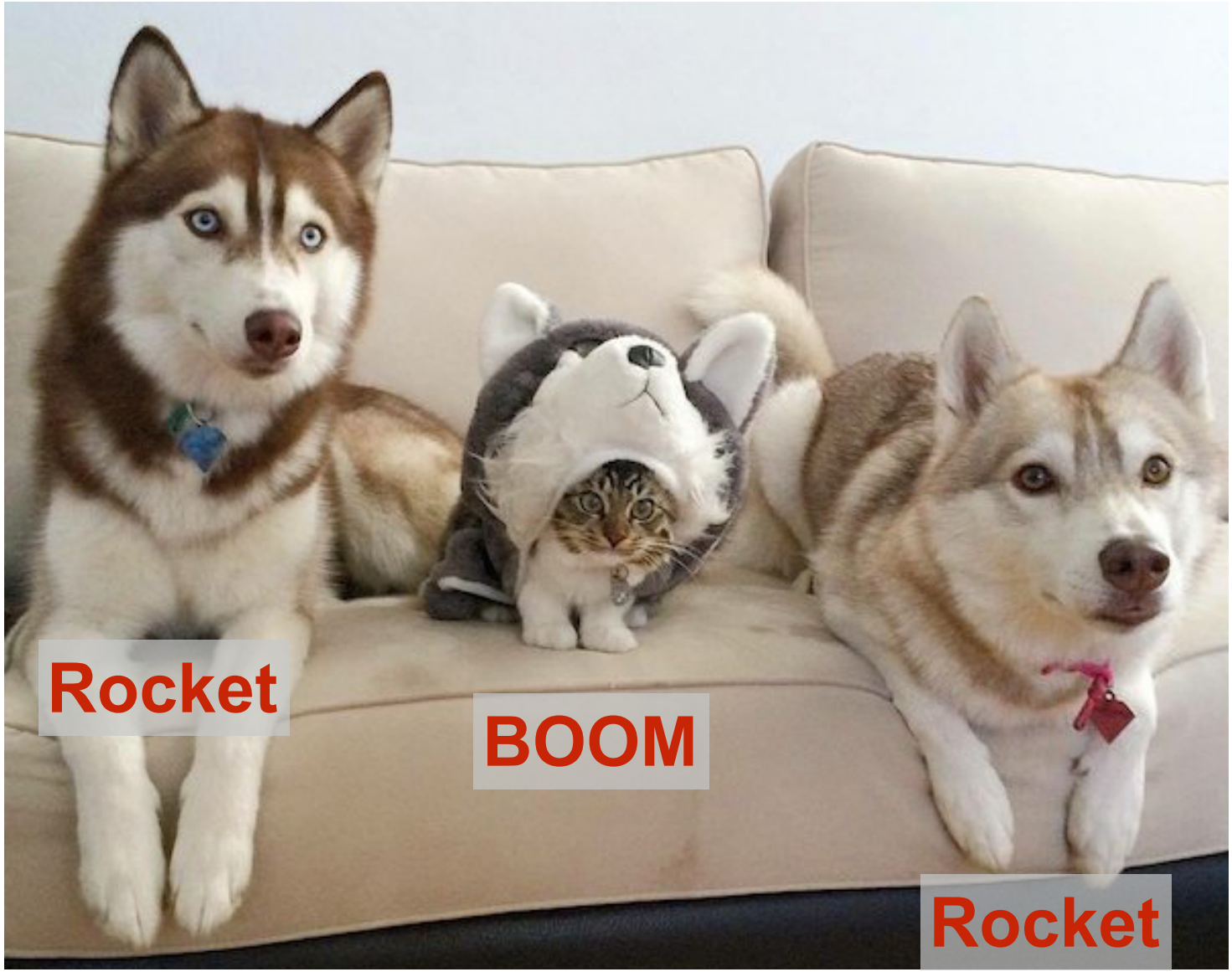
# What is BOOM?




- "Berkeley Out-of-Order Machine"
- out-of-order
- superscalar
- implements **RV64G**, boots Linux
- It is synthesizable
- it is open-source
- written in **Chisel** (16k loc)
- It is parameterizable generator
- built on top of Rocket-chip SoC Ecosystem

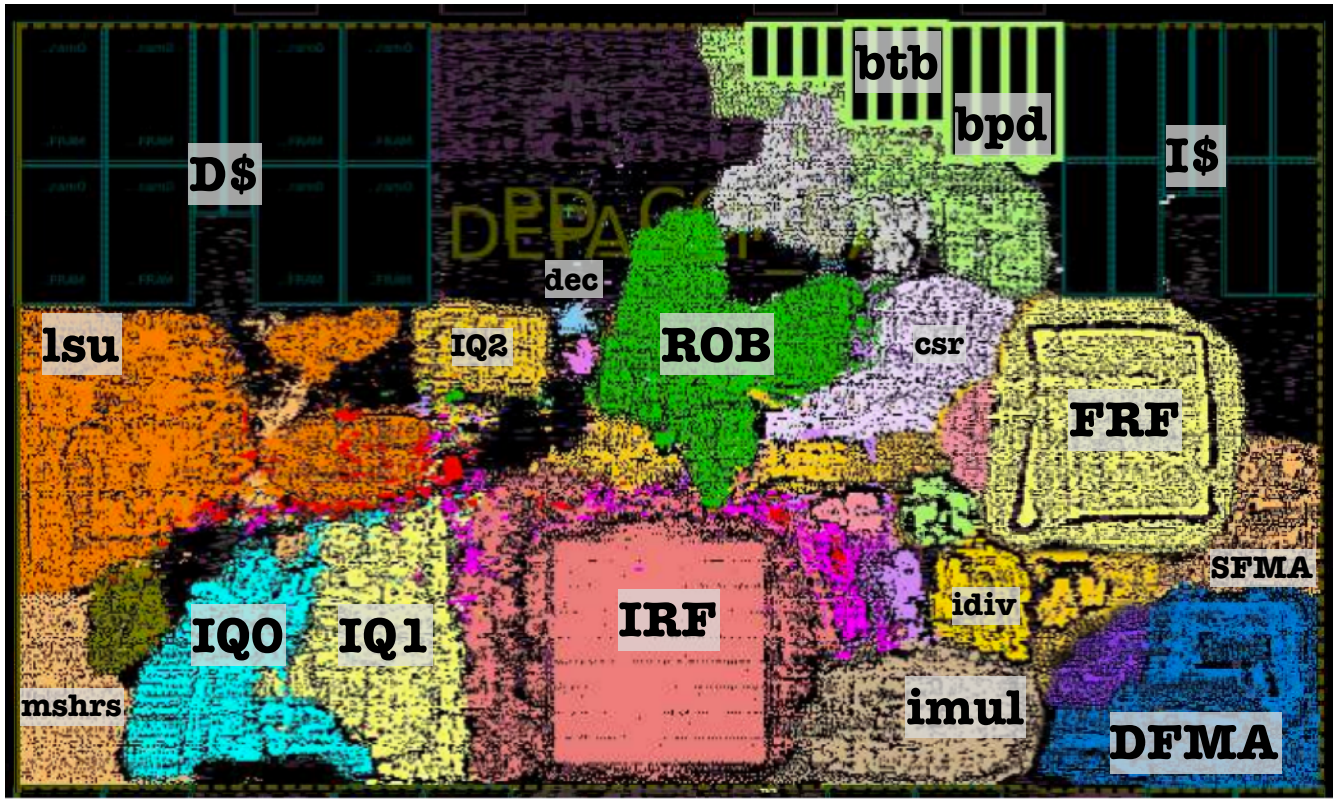
http://ucb-bar.github.io/riscv-boom

# BOOM fits into Rocket-chip SoC



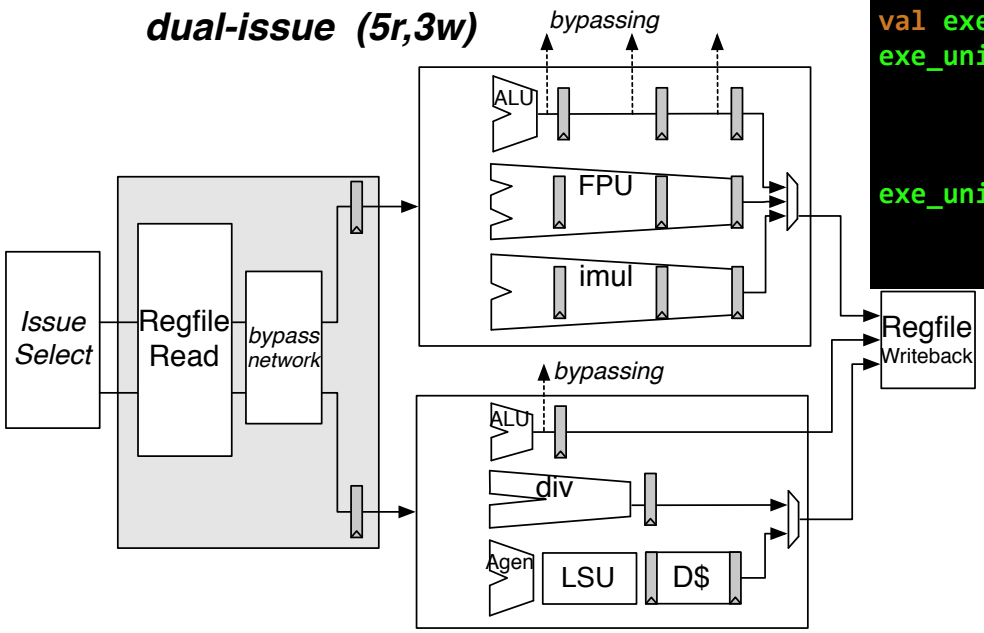
# Lots of neat features

- advanced branch prediction
  - BTB, RAS (1-cycle)
  - gshare and TAGE-based conditional predictor implementations (3-cycle)
  - fits in single-port SRAM
- loads can issue out-of-order wrt to other loads, stores
- hardware performance counters (~40 events)



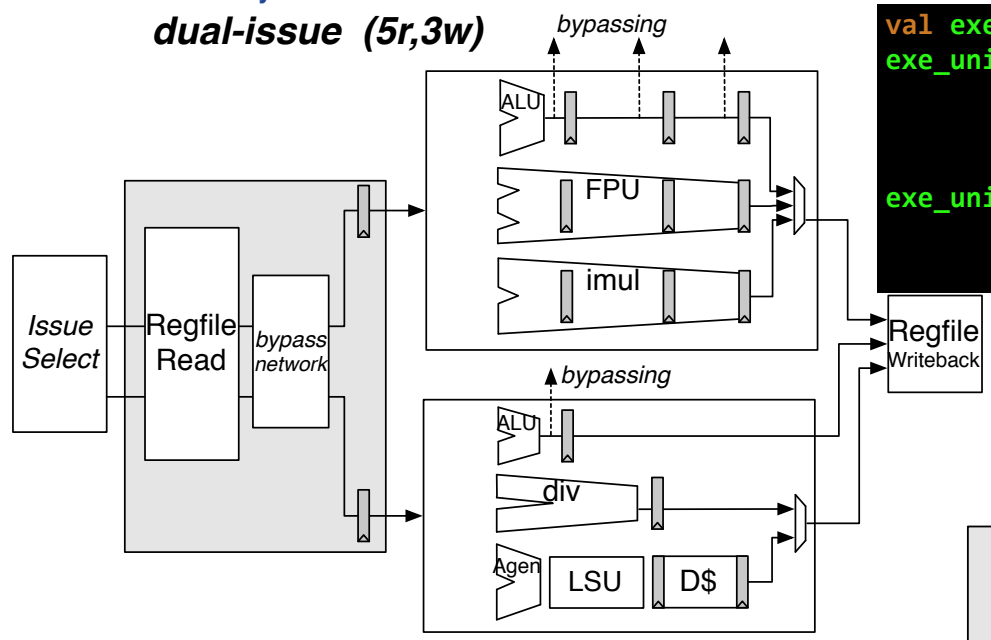
# Parameterized Superscalar

***dual-issue (5r,3w)***



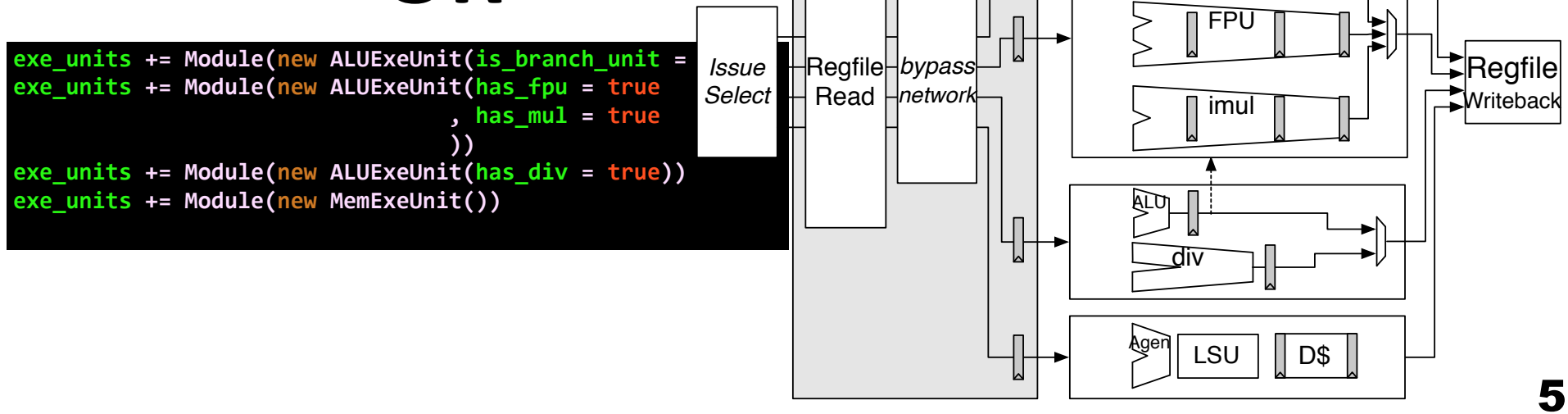

```
val exe_units = ArrayBuffer[ExecutionUnit]()
exe_units += Module(new ALUExeUnit(is_branch_unit    = true
                                  , has_fpu          = true
                                  , has_mul          = true
                                  ))
exe_units += Module(new ALUMemExeUnit(fp_mem_support = true
                                  , has_div          = true
                                  ))
```

# Parameterized Superscalar




```
val exe_units = ArrayBuffer[ExecutionUnit]()
exe_units += Module(new ALUExeUnit(is_branch_unit    = true
                                  , has_fpu          = true
                                  , has_mul          = true
                                  ))
exe_units += Module(new ALUMemExeUnit(fp_mem_support = true
                                  , has_div          = true
                                  ))
```

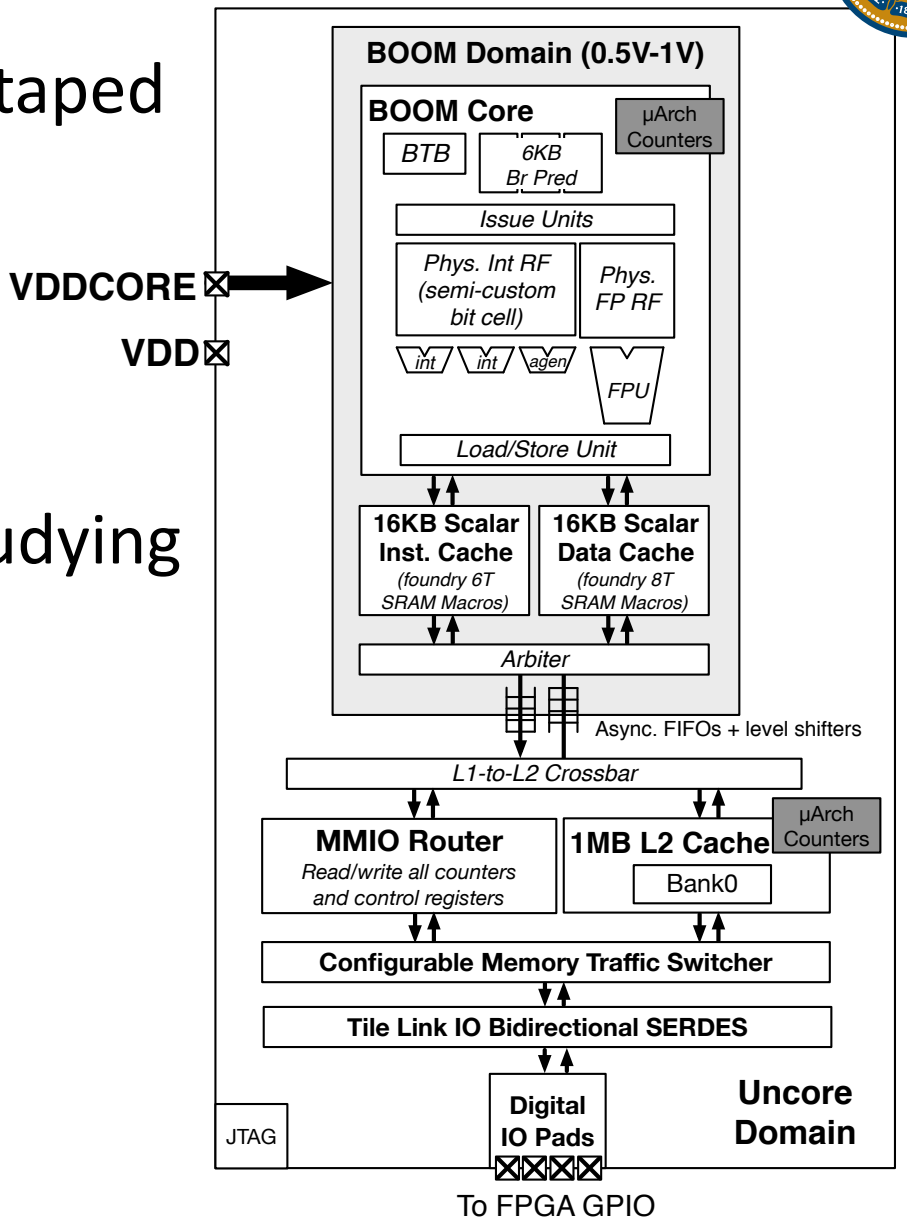
**OR**




```
exe_units += Module(new ALUExeUnit(is_branch_unit =
exe_units += Module(new ALUExeUnit(has_fpu = true
                                  , has_mul = true
                                  ))
exe_units += Module(new ALUExeUnit(has_div = true))
exe_units += Module(new MemExeUnit())
```

# Good news

- BOOM has (finally) been taped out!
  - 2017 Aug 15
- TSMC 28 nm HPM
- Academic test-chip
- BOOM is in support of studying other features

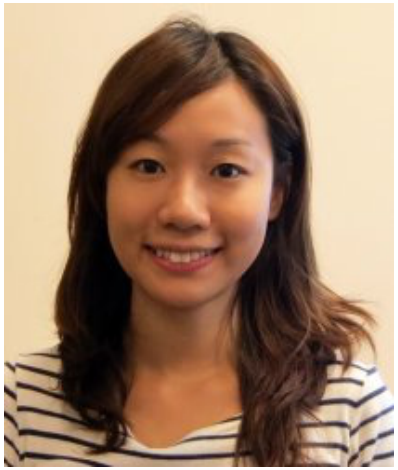


# BOOM v2

## taping out an out-of-order processor with a 2 person team in 4 months

Christopher Celio, Pi-Feng Chiu, Borivoje Nikolic, David Patterson, Krste Asanović

**2017 CARRV Workshop**




Berkeley Architecture Research

# Comparison: Industry

| Processor | SiFive U54 Rocket (RV64GC) | Berkeley BOOMv2 | UltraSPARC T2 | ARM Cortex-A9 | Intel Xeon Ivy |
|---|---|---|---|---|---|
| Language | Chisel | Chisel | Verilog | - | SystemVerilog |
| Core LoC | 8,000 | 16,000 | 290,000 | - | NDA |
| Total LoC | 34,000 | 50,000 | 1,300,000 | - | NDA |

# Comparison: Industry

| Processor | SiFive U54 Rocket (RV64GC) | Berkeley BOOMv2 | UltraSPARC T2 | ARM Cortex-A9 | Intel Xeon Ivy |
|---|---|---|---|---|---|
| Language | Chisel | Chisel | Verilog | - | SystemVerilog |
| Core LoC | 8,000 | 16,000 | 290,000 | - | NDA |
| Total LoC | 34,000 | 50,000 | 1,300,000 | - | NDA |
| **Foundry** | TSMC | TSMC | TI | TSMC | Intel |
| **Technology** | 28 nm (HPC) | 28 nm (HPM) | 65 nm | 40 nm (G) | 22 nm |
| **Core+L1 Area** | 0.54 $mm^2$ | 0.52 $mm^2$ 16kB/16kB | ~12 $mm^2$ | ~2.5 $mm^2$ | ~12 $mm^2$ core+L1+L2 |
| **Coremark/MHz** | 2.75 | 3.92 | 1.64* | 3.71 | 5.60 |
| **Frequency** | 1.5 GHz | 1.2 GHz** | 1.17 GHz | 1.4 GHz | 3.3 GHz |

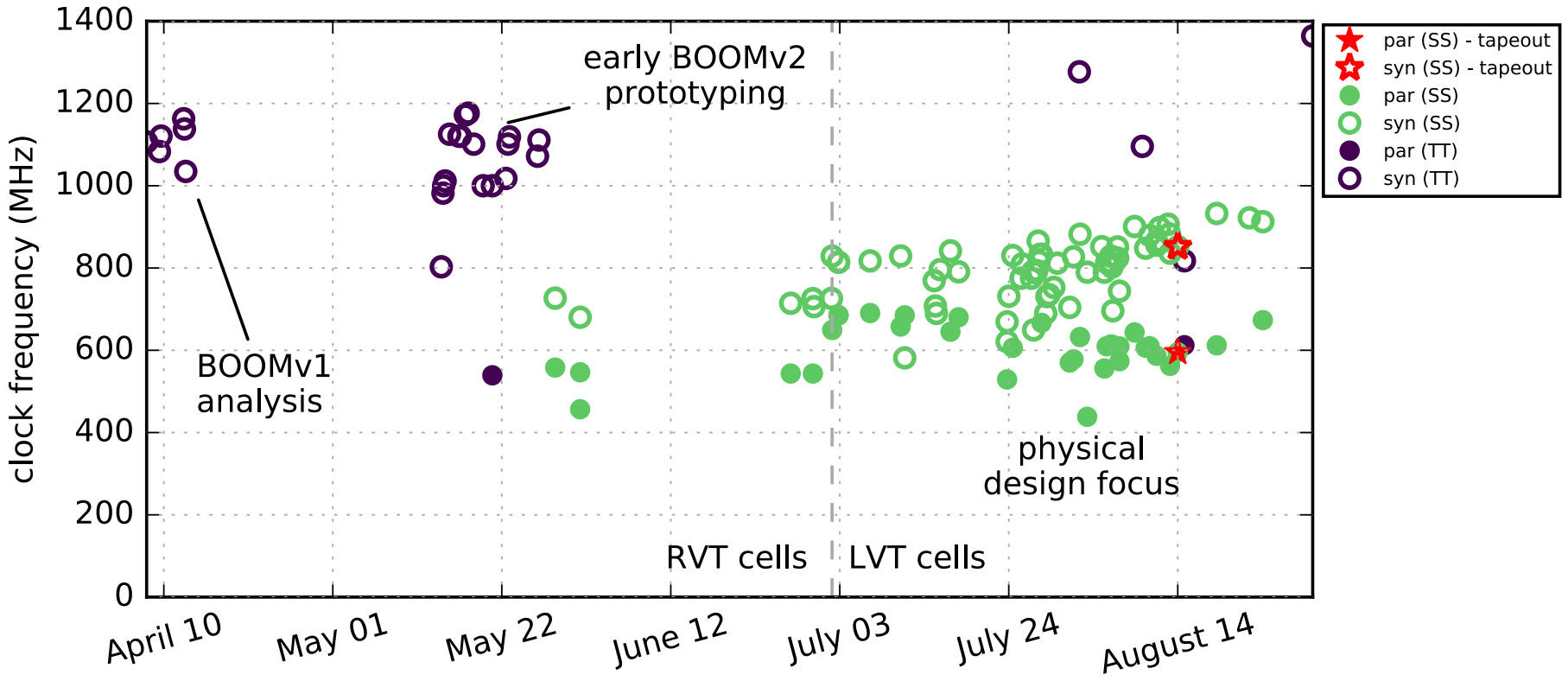
*From eembc.org. 32 threads/8 cores achieve 13 Cm/MHz.

**Estimated

# Limits of Educational Libraries

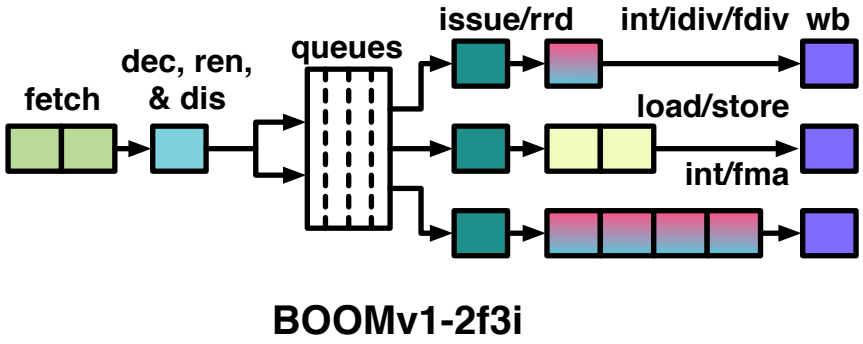
- BOOMv1
  - Educational 45 nm libraries
  - CACTI model for SRAMs
  - synthesized flip-flops for most non-cache memory arrays
  - good for catching small timing bugs
  - not accurate enough to justify sweeping redesigns
- BOOMv2
  - TSMC 28 nm HPM (high performance mobile)
  - LVT-based standard cells
  - memory compiler (one- and two-port)

# 4 months of agile tape-out

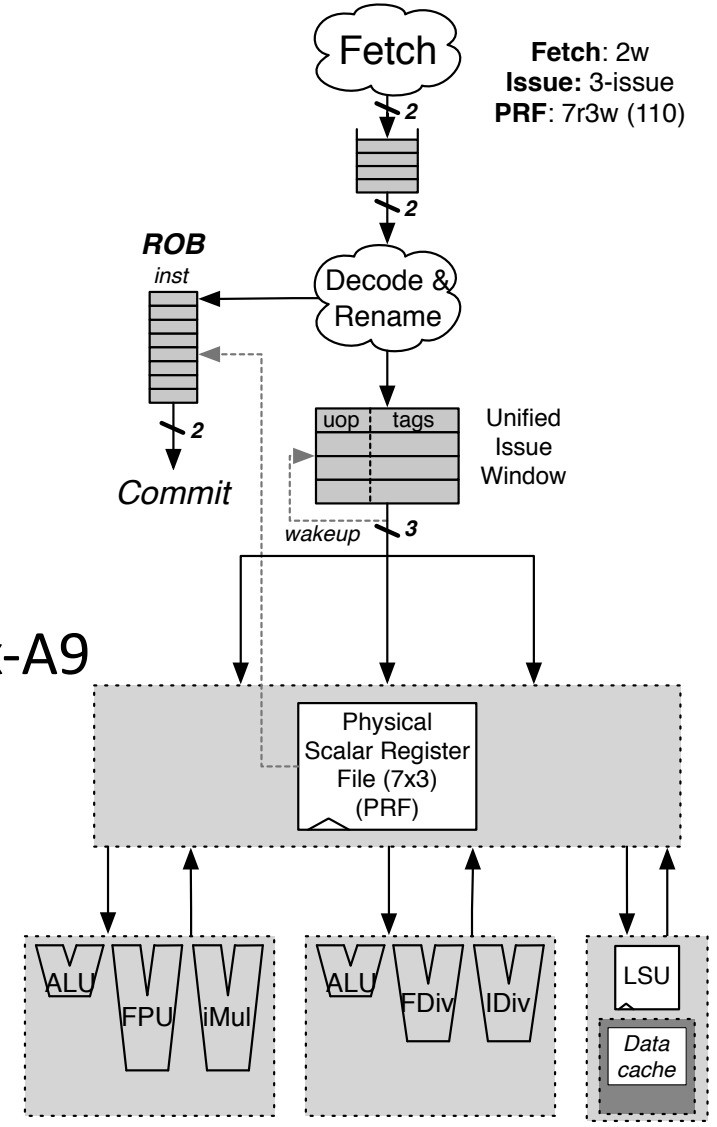



*ignore the Y-axis:
-- too many parameters/variables changing between each run
-- doesn't capture DRC violations

# BOOM v1

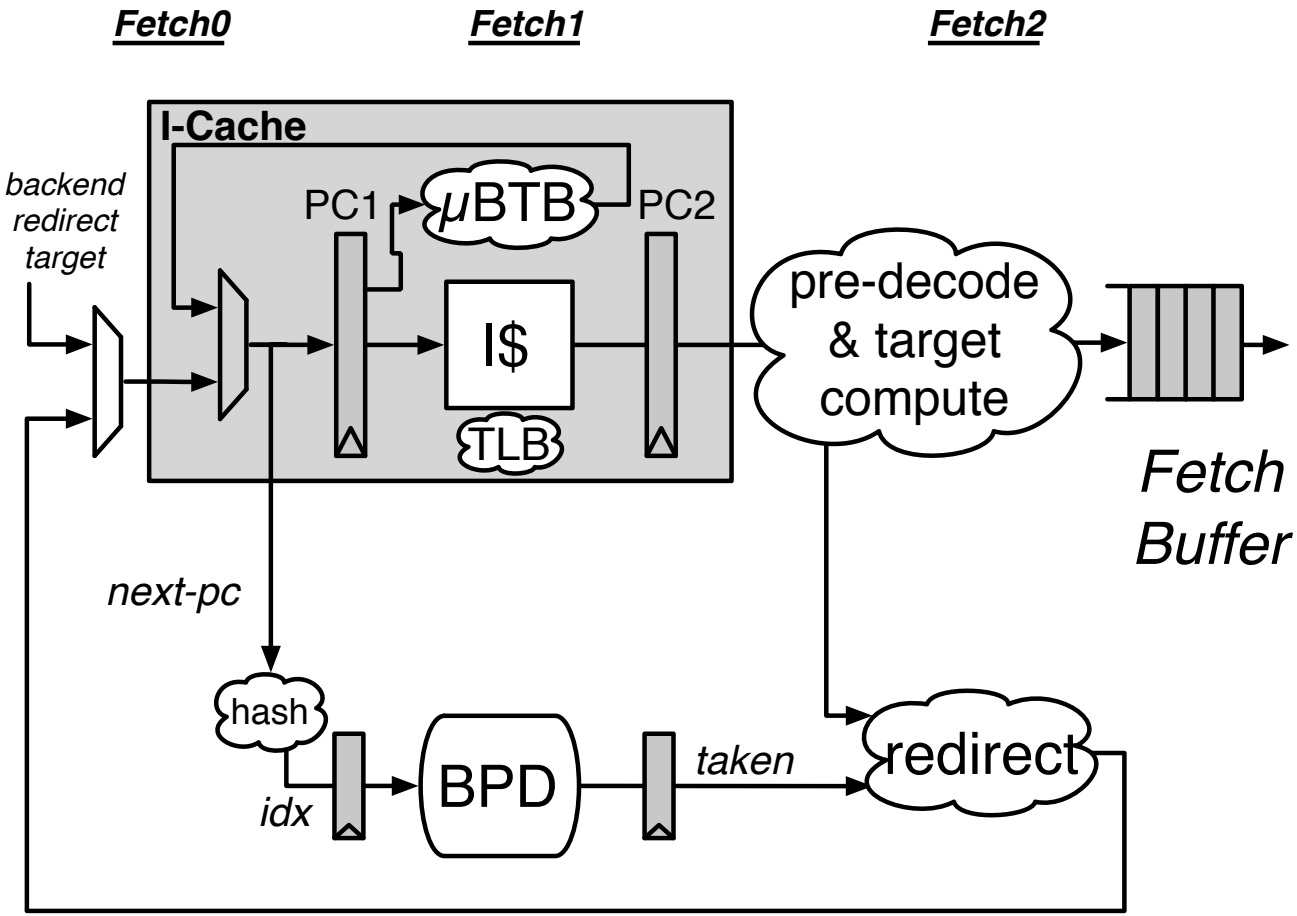



**BOOMv1-2f3i**

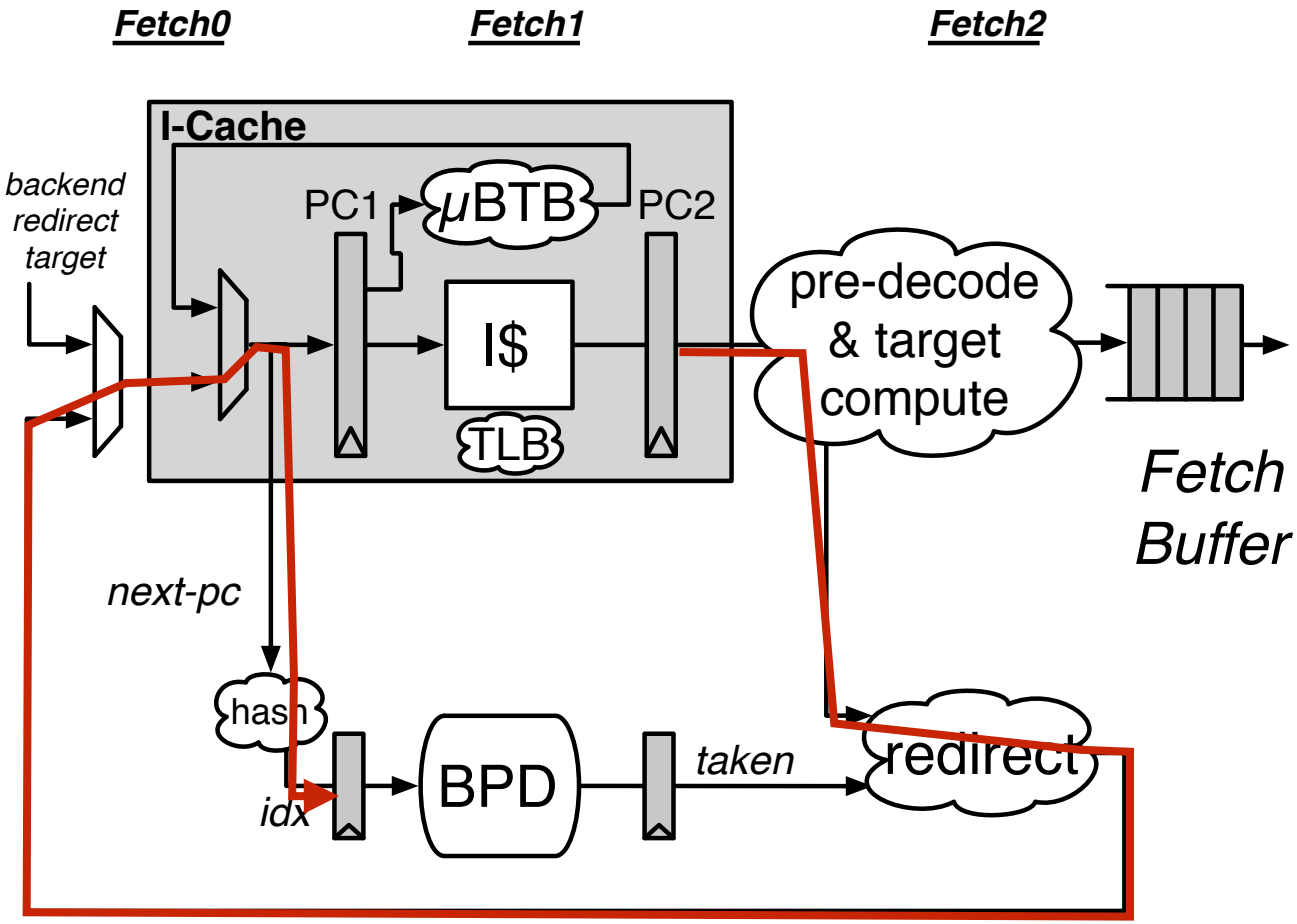



- short pipeline
  - inspired by R10K, 21264, Cortex-A9
- unified issue window

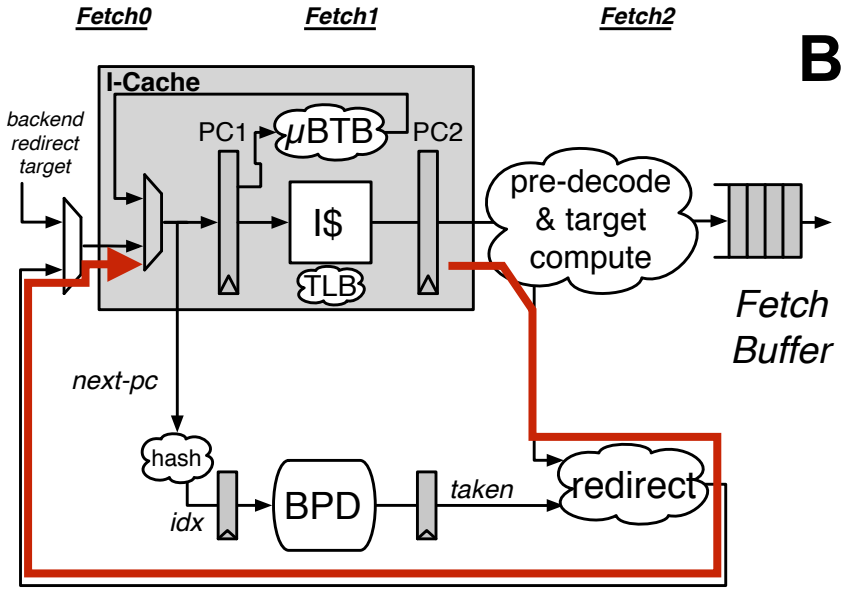
# BOOMv1: Frontend



# BOOMv1: Frontend

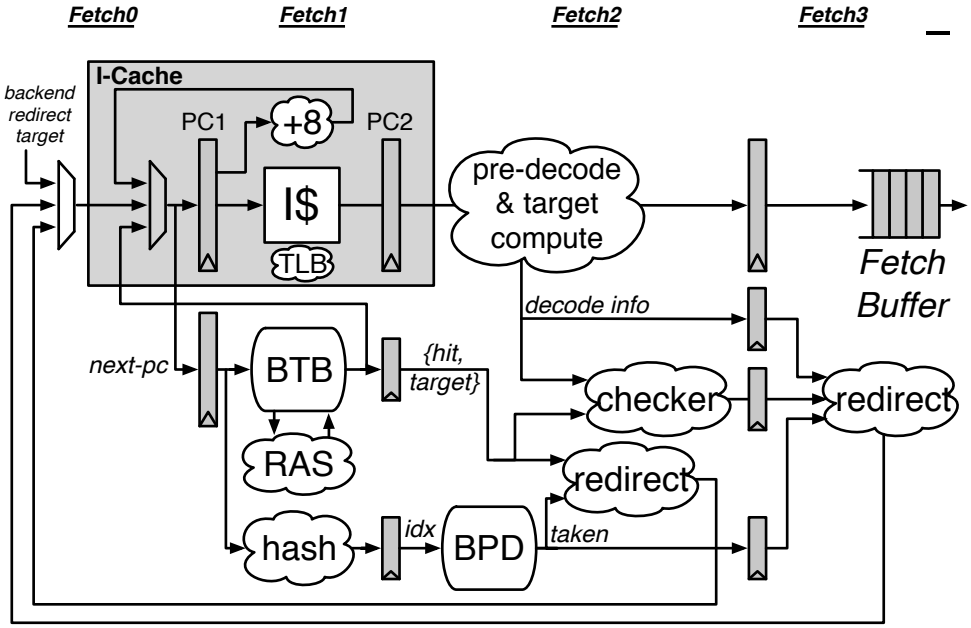


# Frontend




## BOOMv1

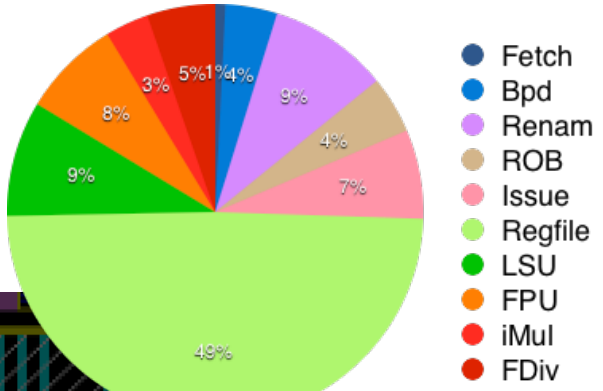
- BTB in SRAM
  - set-associative
  - partially tagged
  - Checker to verify integrity
- BPD (Conditional Predictor)
  - hash gets entire stage
  - redirect based on BTB
  - redirect pushed back (I$)

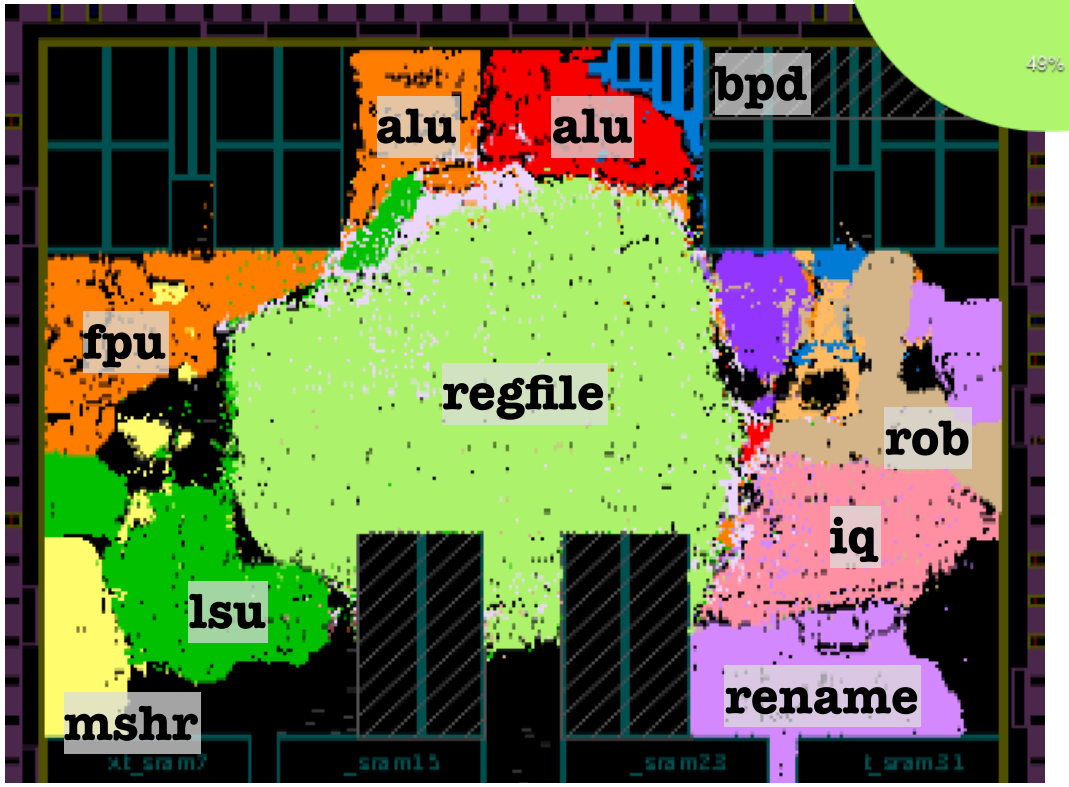



## BOOMv2

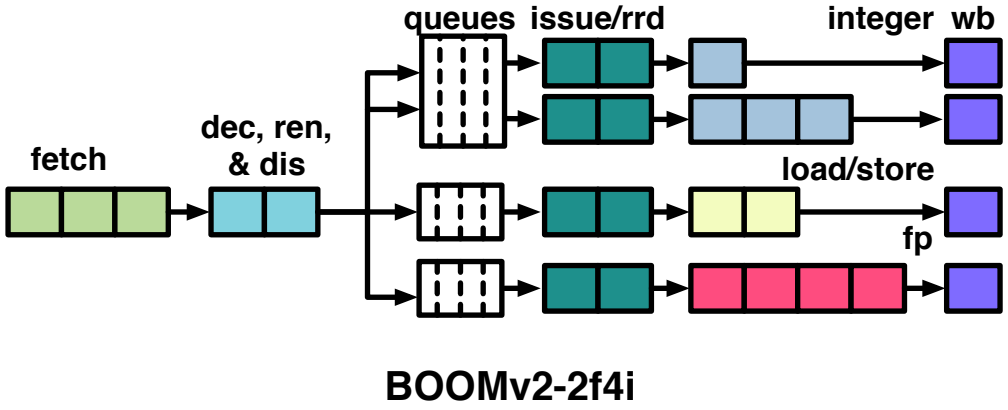
# Regfile: (the first P&R)

- Regfile blows up
- critical paths in issue-select, register read



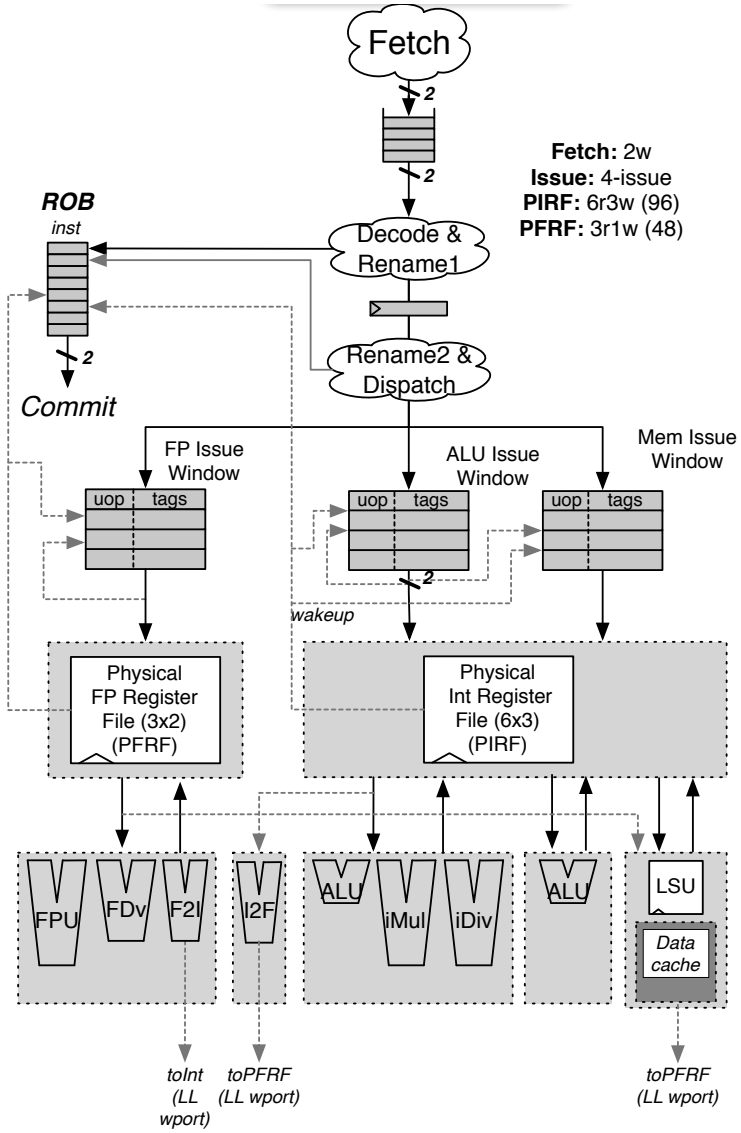



# BOOMv2



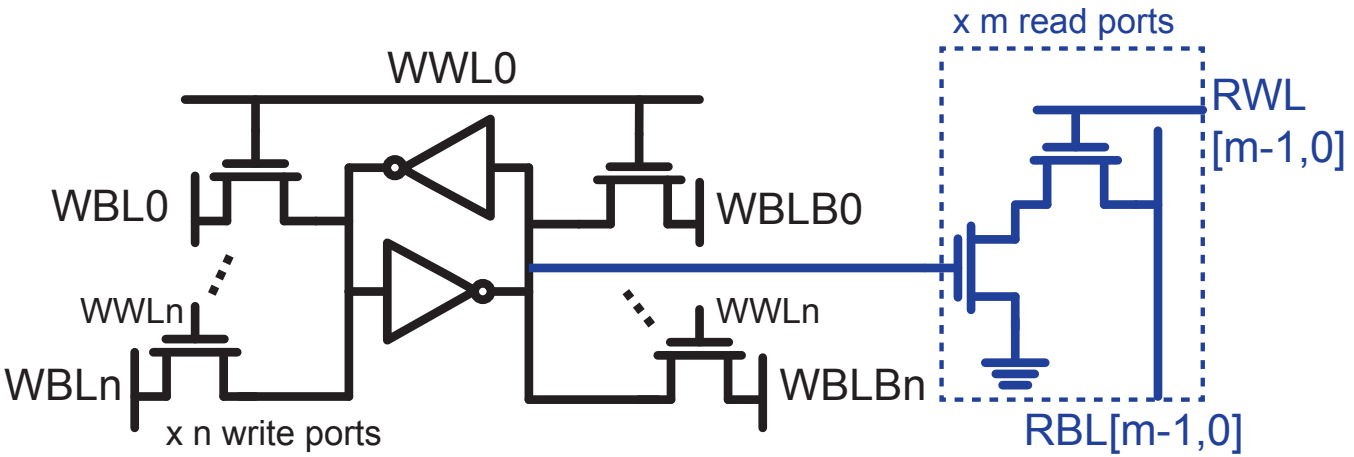

**BOOMv2-2f4i**

- split unified issue window (1 day)
- split physical register file (1 week)
- issue/register read separate stages (~1 day)
- 2 stage rename (~1 week)
- 3 stage fetch (3 weeks)



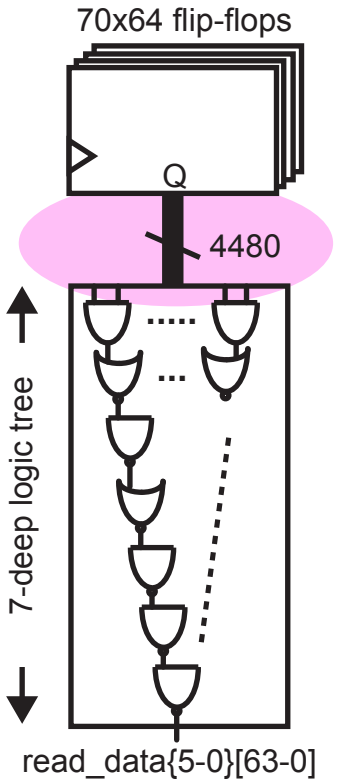
# Regfile Challenges

- foundational IP provide memories with single and dual-port memories
- companies build their own hand-crafted register files
- not enough labor in academia to support a customized register file

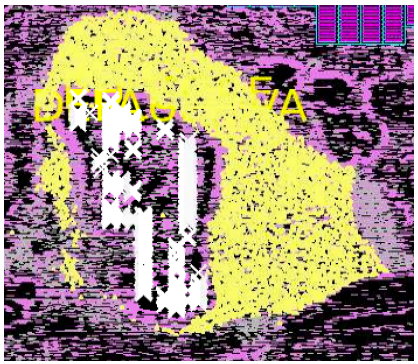


# Regfile

## synthesized

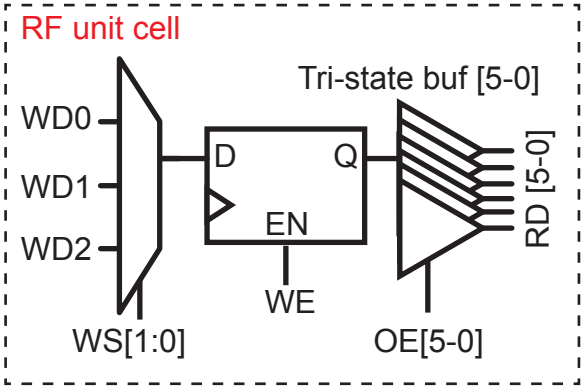



Congestion!

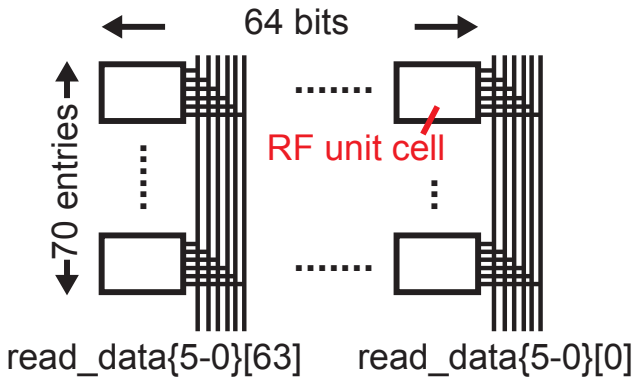


- Routing 4480 wires causes congestion issues

## array

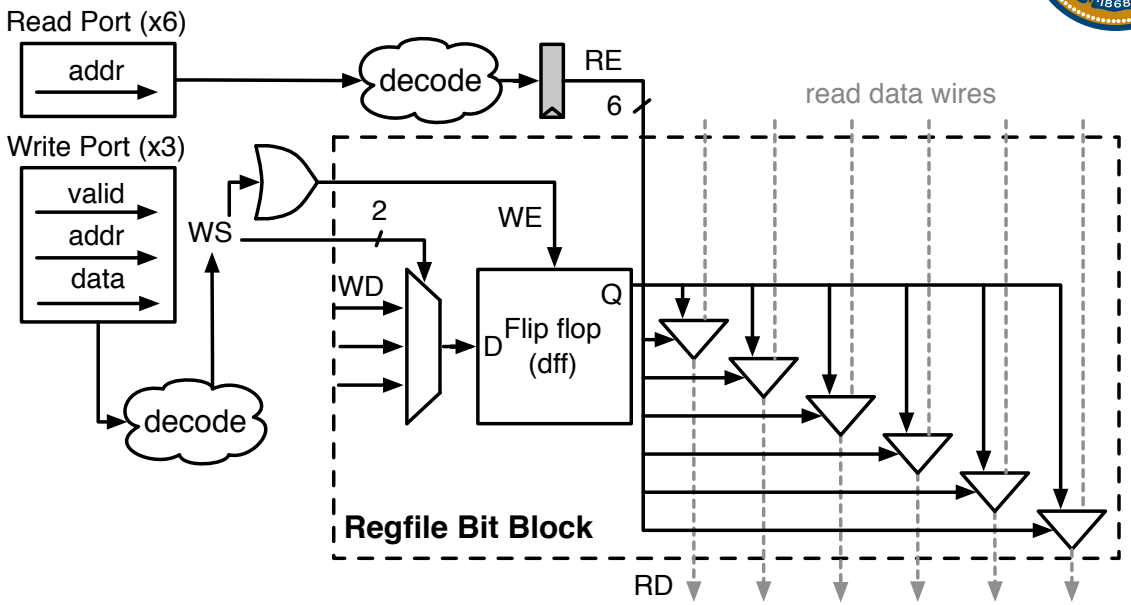



Tri-state buffer mimics the PG in a custom RF cell so that read data wire can be shared




- Reduce read_data routing wires to 384

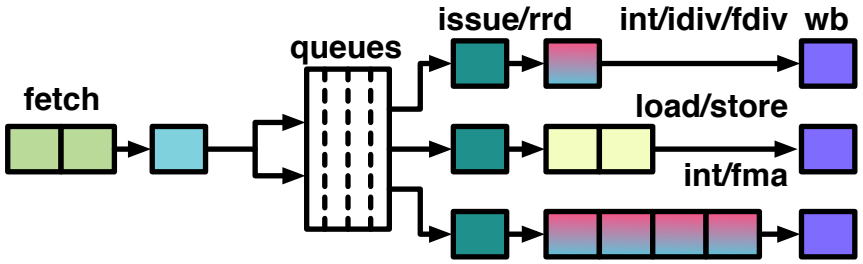
# Regfile: Our quick solution



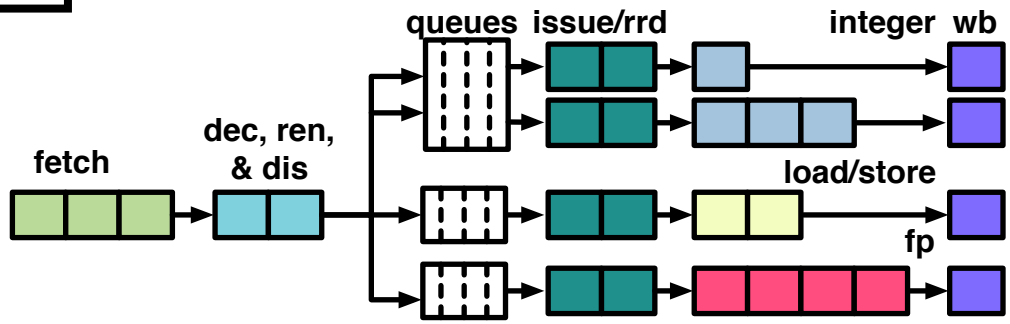

- hand-craft our own bit block out of standard cells
  - tri-state to drive read wires
  - hierarchical bitlines
- pre-place arrays of bit blocks
- let router handle the 18 wires per 1 flip-flop
- blackbox in Chisel
  - simulate control using Chisel model
  - verify at gate-level

# Before and After

| | BOOMv1 | BOOMv2 |
|---|---|---|
| BTB entries | 40 (fully-associative) | 64 x 4 (set-associative) |
| Fetch Width | 2 insts | 2 insts |
| Issue Width | 3 micro-ops | 4 micro-ops |
| Issue Entries | 20 | 16/20/10 |
| Regfile | 7r3w | 6r3w (int), 3r2w (fp) |
| Exe Units | iALU+iMul+FMA<br>iALU+fDiv<br>Load/Store | iALU+iMul+iDiv<br>iALU<br>FMA+fDiv<br>Load/Store |




BOOMv1-2f3i




BOOMv2-2f4i

# Results

# Results

- Hard to compute -- not apples to oranges
  - a lot of the work was about design rule checks, not performance

# Results

- Hard to compute -- not apples to oranges
  - a lot of the work was about design rule checks, not performance
- Roughly 25% decrease in clock period
- Roughly 20% decrease in Coremark/MHz
  - known issue from load-use delay

# Results

- Hard to compute -- not apples to oranges
  - a lot of the work was about design rule checks, not performance
- Roughly 25% decrease in clock period
- Roughly 20% decrease in Coremark/MHz
  - known issue from load-use delay
- Barely a win?
- Changes fixed DRC errors, geometry errors (shorts), so...

# Results

- Hard to compute -- not apples to oranges
  - a lot of the work was about design rule checks, not performance
- Roughly 25% decrease in clock period
- Roughly 20% decrease in Coremark/MHz
  - known issue from load-use delay
- Barely a win?
- Changes fixed DRC errors, geometry errors (shorts), so...
  - infinitely Faster! Better! Stronger!

# Agile Hardware Development

- RTL hacking can be very agile
  - ~6 minutes to compile, build, and run "riscv-tests" regression suite (10 KHz for Verilator simulator)
  - Chisel allows for quick, far-reaching changes
  - generator approach allows for late-binding design decisions
  - small changes, improvements (that don't affect floor plan) are agile

# Agile Hardware Development

- RTL hacking can be very agile
  - ~6 minutes to compile, build, and run "riscv-tests" regression suite (10 KHz for Verilator simulator)
  - Chisel allows for quick, far-reaching changes
  - generator approach allows for late-binding design decisions
  - small changes, improvements (that don't affect floor plan) are agile
- Physical design is a bottleneck
  - 2-3 hours for synthesis results
  - 8-24 hours for p&r results
  - too much value provided by manual intervention
  - reports are difficult to reason about
  - sadly can't throw this at a computer and get 7 good designs a week later

# Agile Hardware Development

- RTL hacking can be very agile
  - ~6 minutes to compile, build, and run "riscv-tests" regression suite (10 KHz for Verilator simulator)
  - Chisel allows for quick, far-reaching changes
  - generator approach allows for late-binding design decisions
  - small changes, improvements (that don't affect floor plan) are agile
- Physical design is a bottleneck
  - 2-3 hours for synthesis results
  - 8-24 hours for p&r results
  - too much value provided by manual intervention
  - reports are difficult to reason about
  - sadly can't throw this at a computer and get 7 good designs a week later
- Verification is another bottleneck
  - I can write bugs faster than I can find them

# Future Directions (for BOOM)

- A lot of improvements to make to BOOM
  - Clear direction of further IPC/QoR improvements

# Future Directions (for me)

Esperanto Technologies

"Esperanto Technologies designs high-performance energy-efficient computing solutions."
- riscv.org

- See Dave Ditzel's talk from yesterday to learn more
- RISC-V Founding Gold Member
  - on lots of working groups!
- We are committed to maintaining the BOOM open-source repository
- We're hiring...

# A 2-person tapeout takes a village!

- RISC-V ISA
  - very out-of-order friendly!
- Chisel hardware construction language
  - object-oriented, functional programming
- FIRRTL
  - exposed RTL intermediate representation (IR)
- Rocket-chip
  - A full working SoC platform built around the Rocket in-order core
- Thanks to:
  - Krste Asanović, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, 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, David A. Patterson, Brian Richards, Colin Schmidt, Wenyu Tang, Stephen Twigg, Huy Vo, Andrew Waterman, Angie Wang, and more...

# Questions?

## Funding 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 Lab industrial sponsors and affiliates Intel, Google, 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.*