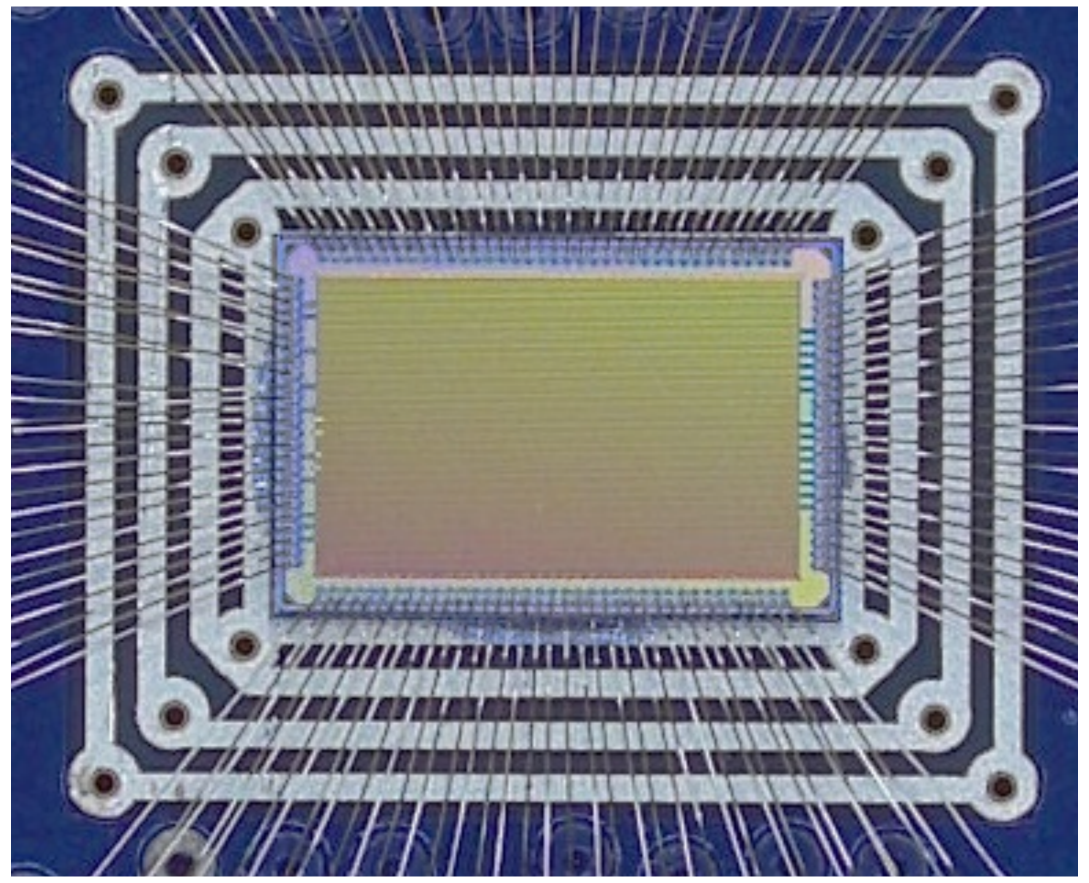
BROOM

An open-source out-of-order processor with resilient low-voltage operation in 28nm CMOS

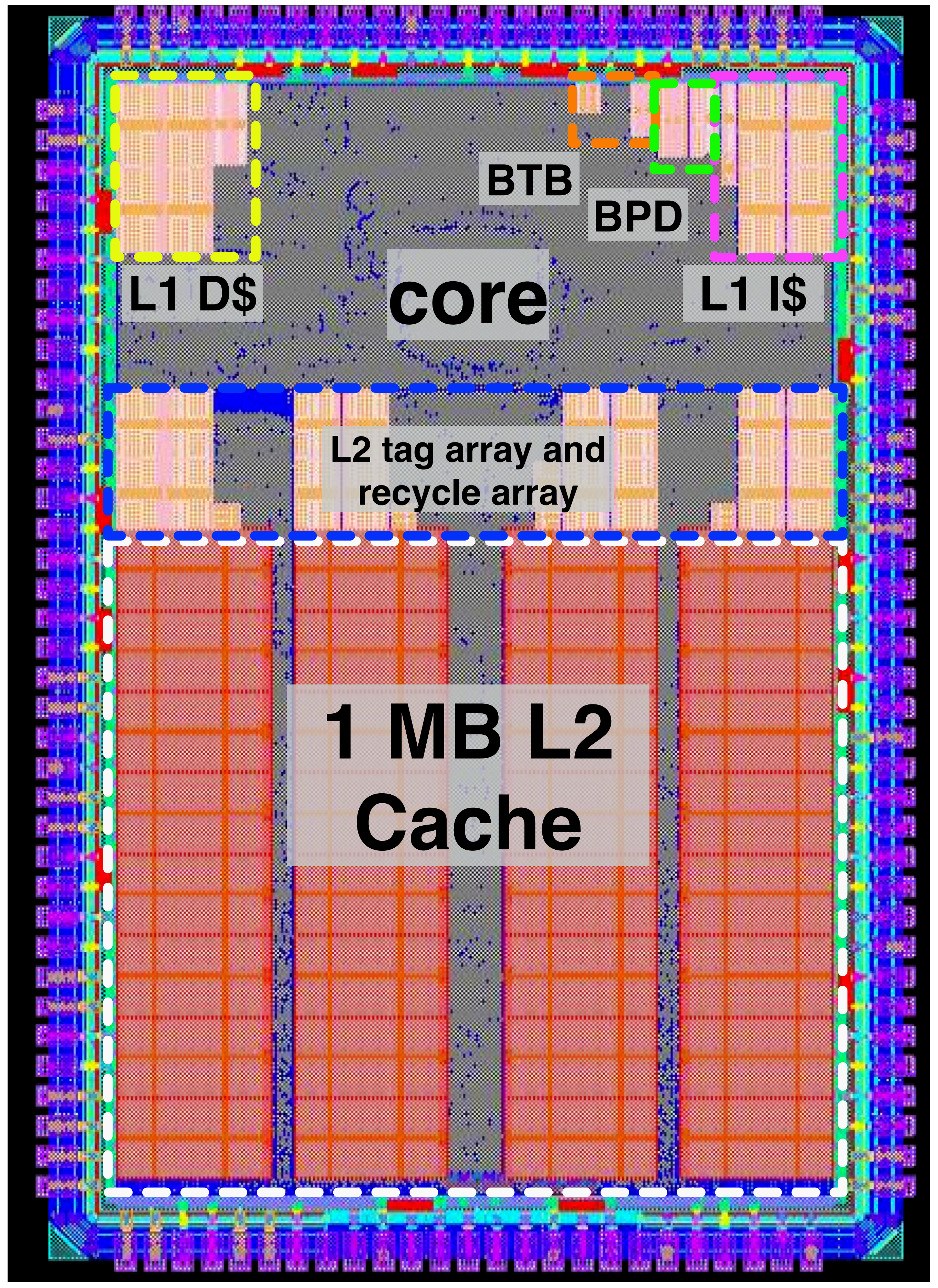


Christopher Celio, Pi-Feng Chiu,

Krste Asanović, David Patterson, and Borivoje Nikolic

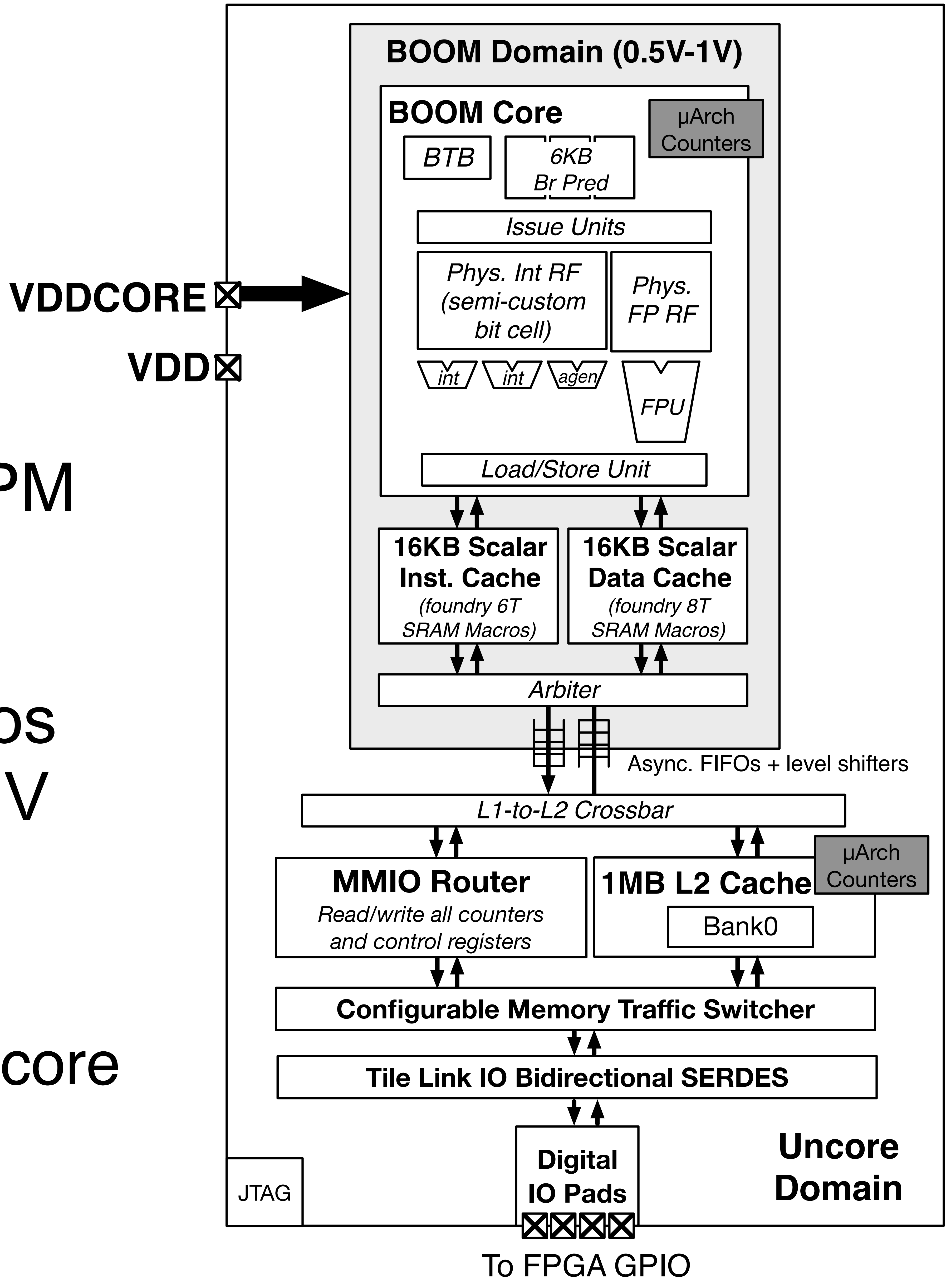
Hot Chips 2018

BROOM Chip (Taped out Aug 2017)



TSMC 28 nm HPM
6 mm^2
417k std cells
73 SRAM macros
1.0 GHz @ 0.9 V

- Open-source superscalar out-of-order RISC-V core
- Resilient cache for low-voltage operation

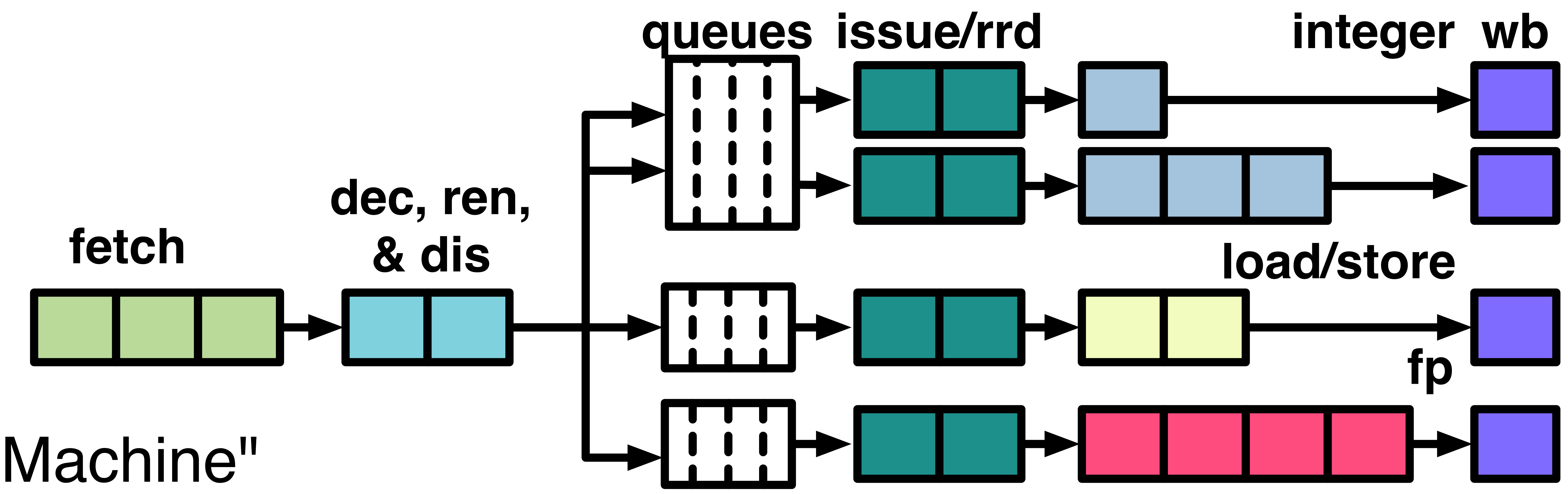


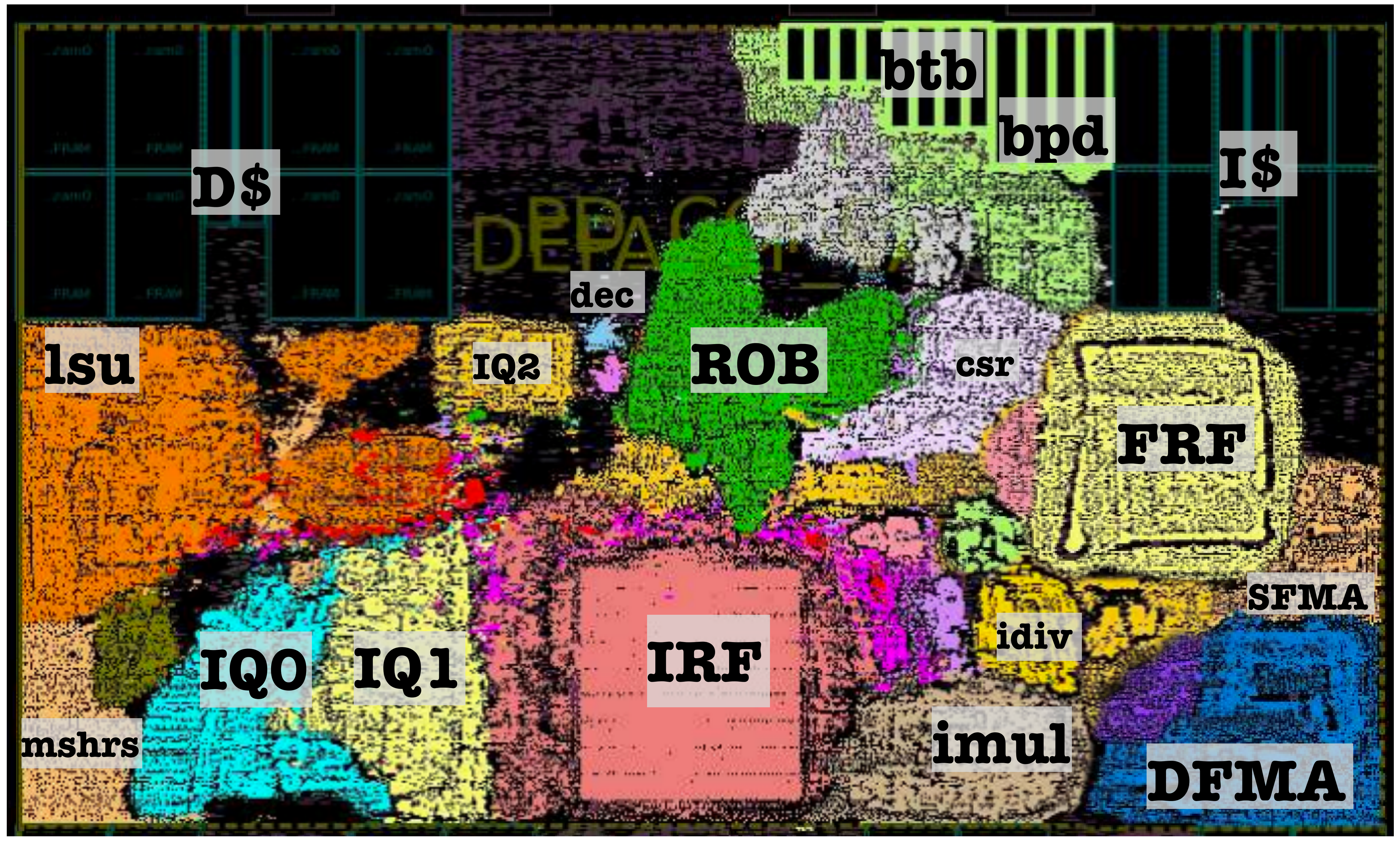
BOOM Domain (0.5V-1V)
BOOM Core
μArch Counters
BTB
6KB Br Pred
Issue Units
Phys. Int RF (semi-custom bit cell)
Phys. FP RF
int
int
agen
FPU
Load/Store Unit
16KB Scalar Inst. Cache (foundry 6T SRAM Macros)
16KB Scalar Data Cache (foundry 8T SRAM Macros)
Arbiter
VDDCORE
VDD
Async. FIFOs + level shifters
L1-to-L2 Crossbar
MMIO Router
Read/write all counters and control registers
1MB L2 Cache
μArch Counters
Bank0
Configurable Memory Traffic Switcher
Tile Link IO Bidirectional SERDES
JTAG
Digital IO Pads
Uncore Domain
To FPGA GPIO

Outline

- What is BROOM?
- The RISC-V BOOM Core
- Micro-architectural-level assist techniques
  - Line Disable (LD)
  - Line Recycle (LR)
  - Dynamic Column Redundancy (DCR)
  - Bit Bypass with SRAM (BB-S)
- The Agile Design Experience
- Chip Implementation
- Low Voltage Experimental Results
- Future Directions

What is BOOM?



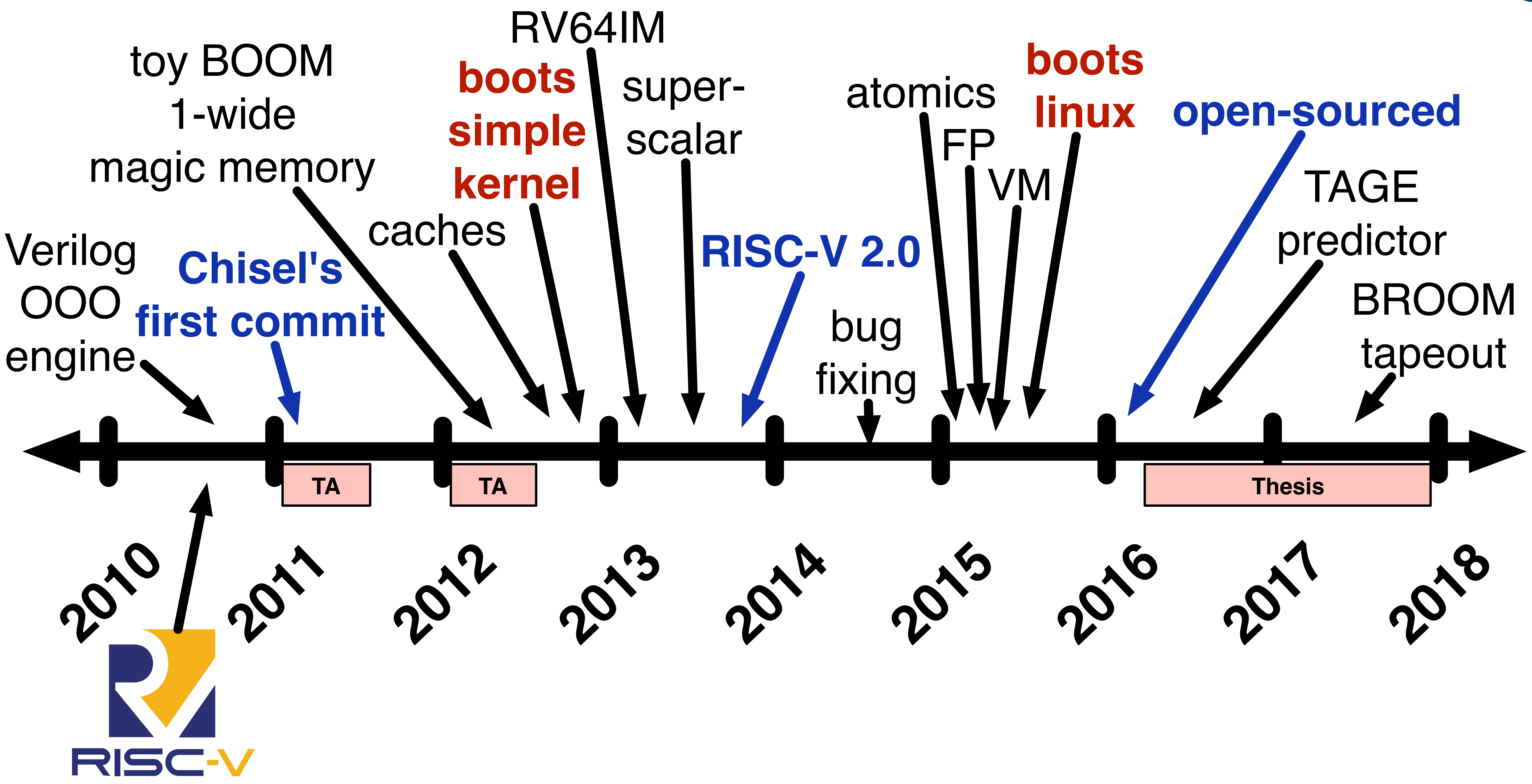


- "Berkeley Out-of-Order Machine"
- out-of-order
- superscalar
- implements **RV64G**, boots Linux
- It is synthesizable
- it is open-source
- written in **Chisel** (16k loc)
- It is parameterizable generator
- built on top of Rocket-chip SoC Ecosystem

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

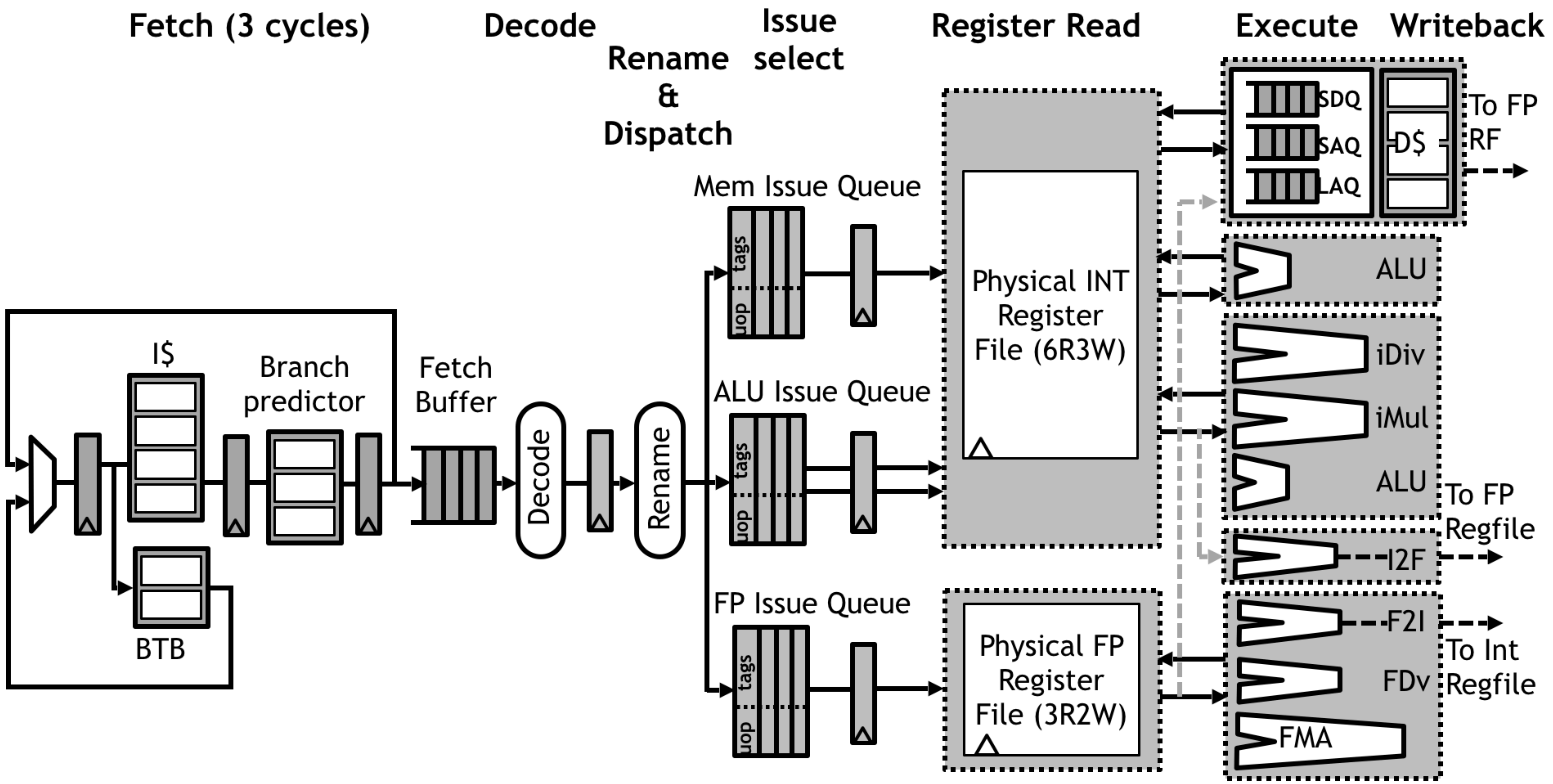
ASPIRE UC Berkeley

Timeline (the path to Hot Chips)



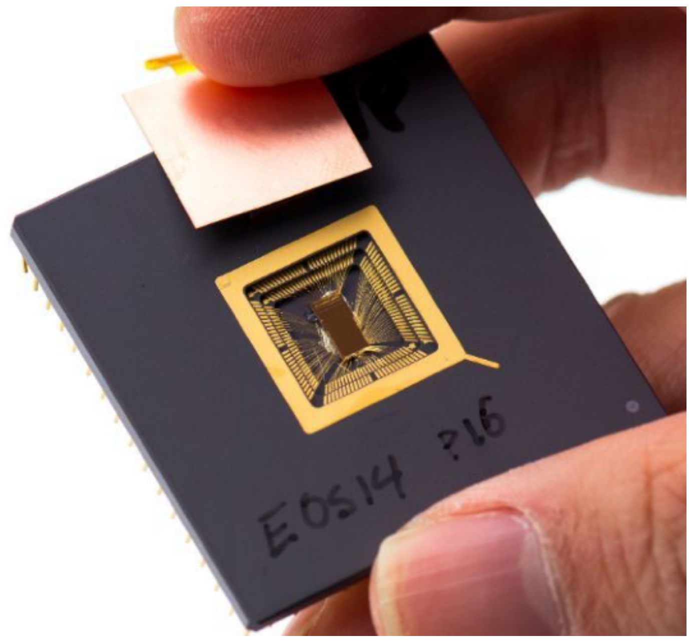
ASPIRE

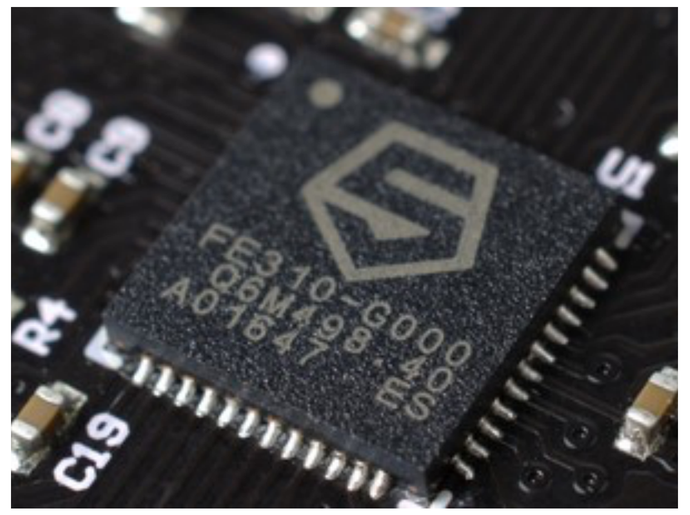
The BOOM Core

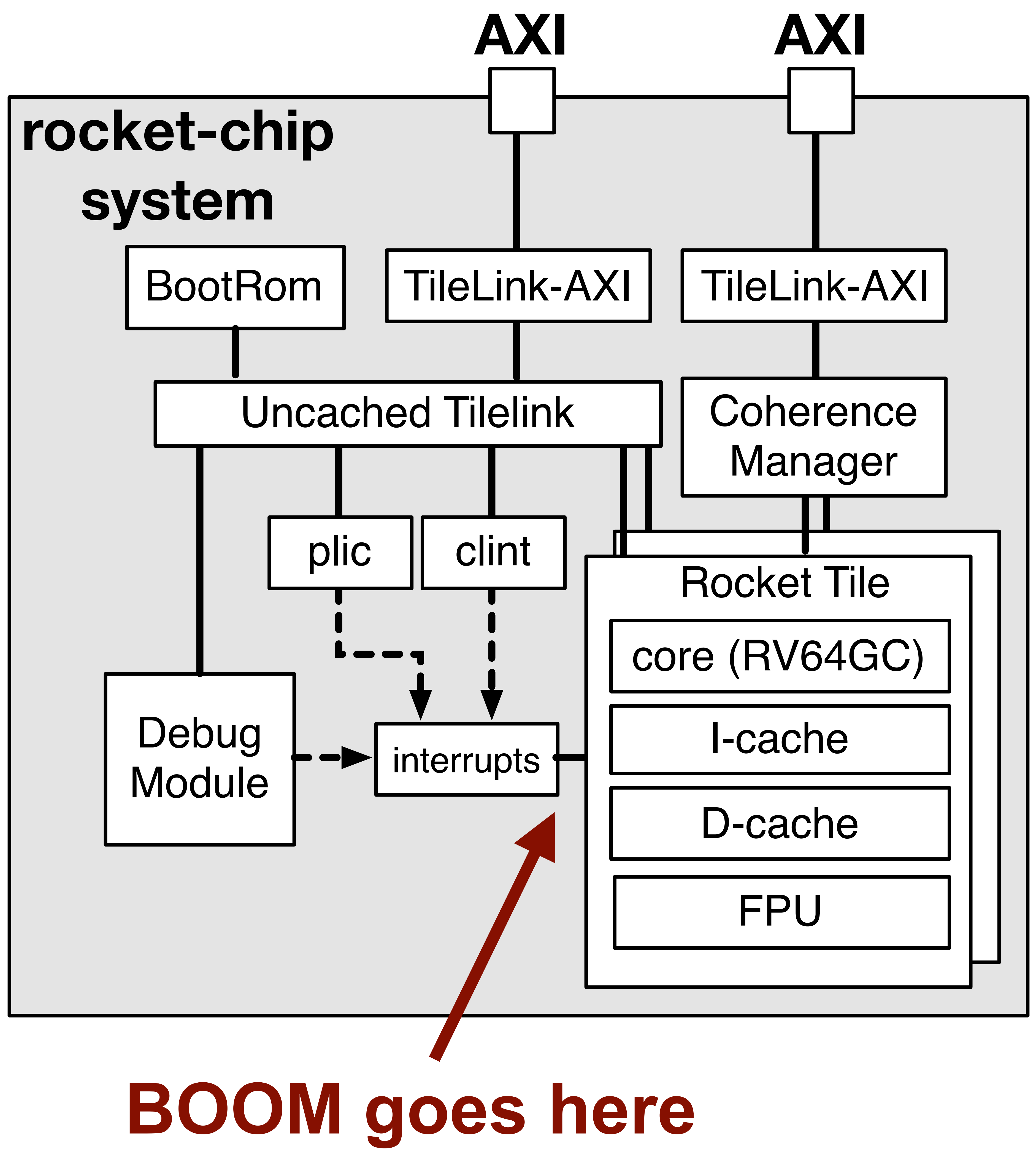


Leveraging Open-source RTL

- The Rocket-chip SoC Generator
- Started in 2011
- Taped out ~~10~~ (~~13~~?) *17* times by Berkeley + many others
- 6,016 commits
- 64 contributors
- Commercial quality
- Replace standard in-order core with BOOM
- Leverage Rocket-chip as a library of processor components







https://github.com/freechipsproject/rocket-chip

Core Comparison

Processor	SiFive U54 Rocket (RV64GC)	Berkeley BOOMv2 (RV64G)	OpenSPARC T2	ARM Cortex-A9	Intel Xeon Ivy
Language	Chisel	Chisel	Verilog	-	SystemVerilog
Core LoC	8,000	16,000	290,000	-	-
SoC LoC	34,000	50,000	1,300,000	-	-

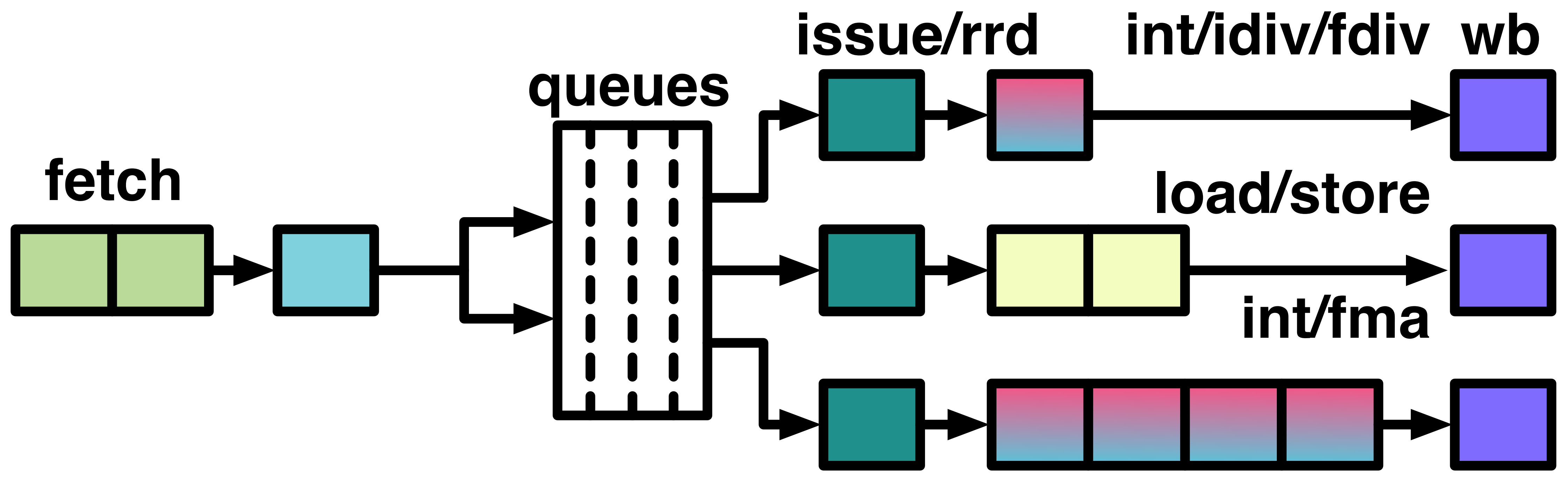
Core Comparison

Processor	SiFive U54 Rocket (RV64GC)	Berkeley BOOMv2 (RV64G)	OpenSPARC T2	ARM Cortex-A9	Intel Xeon Ivy
Language	Chisel	Chisel	Verilog	-	SystemVerilog
Core LoC	8,000	16,000	290,000	-	-
SoC LoC	34,000	50,000	1,300,000	-	-
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	3.77	1.64*	3.71	5.60
Frequency	1.5 GHz	1.0 GHz	1.17 GHz	1.4 GHz	3.3 GHz

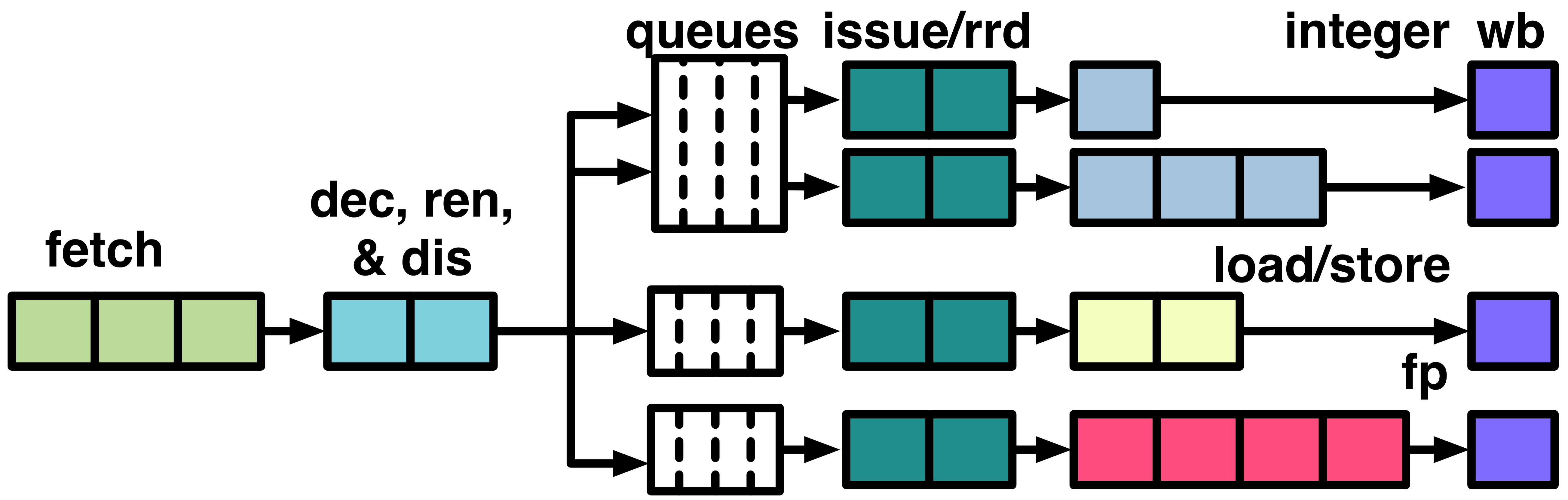
*From eembc.org. 32 threads/8 cores achieve 13 Cm/MHz.

Case study: how agile is the BOOM generator?

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

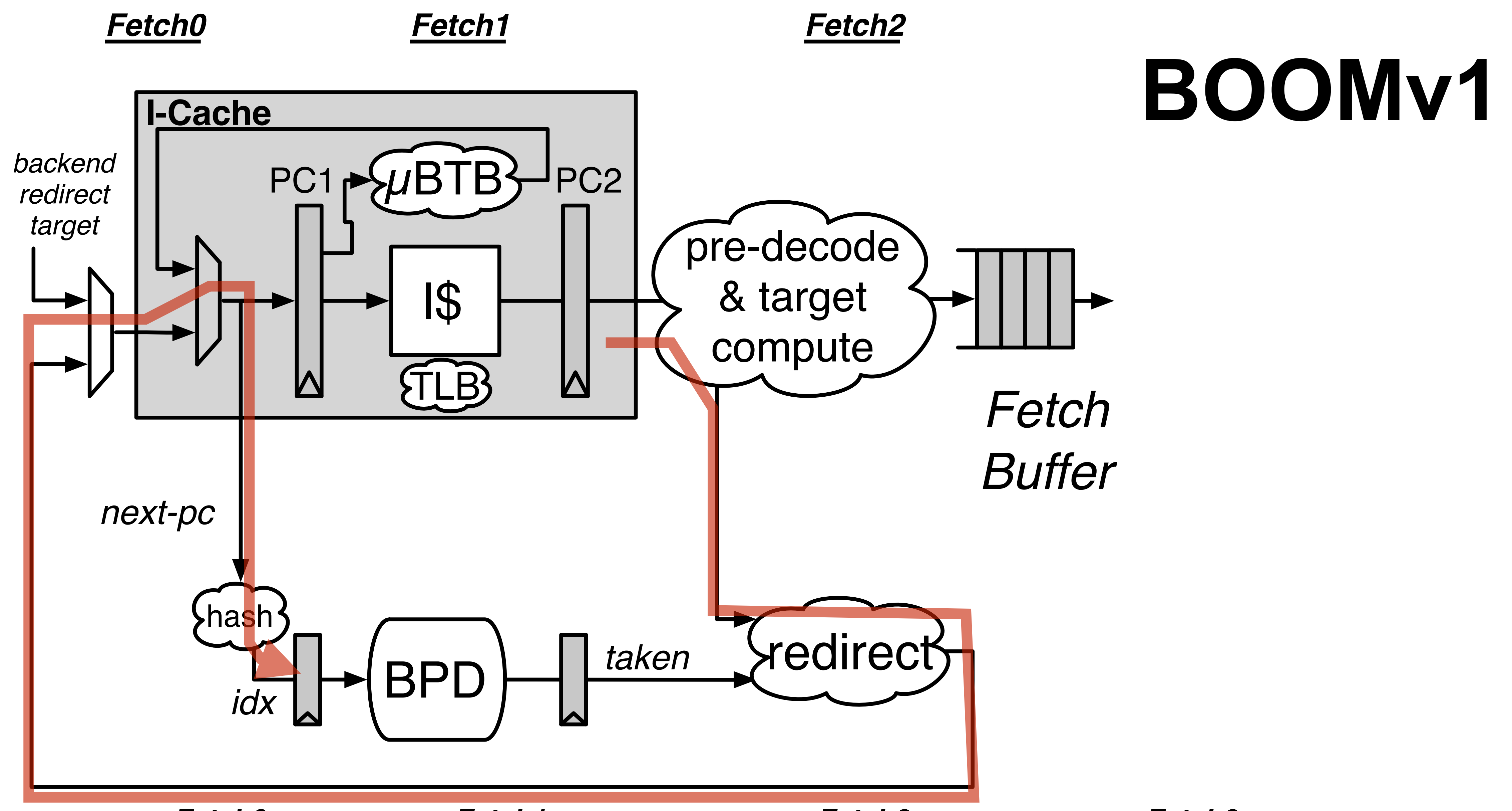


BOOM v1 (April 2017)

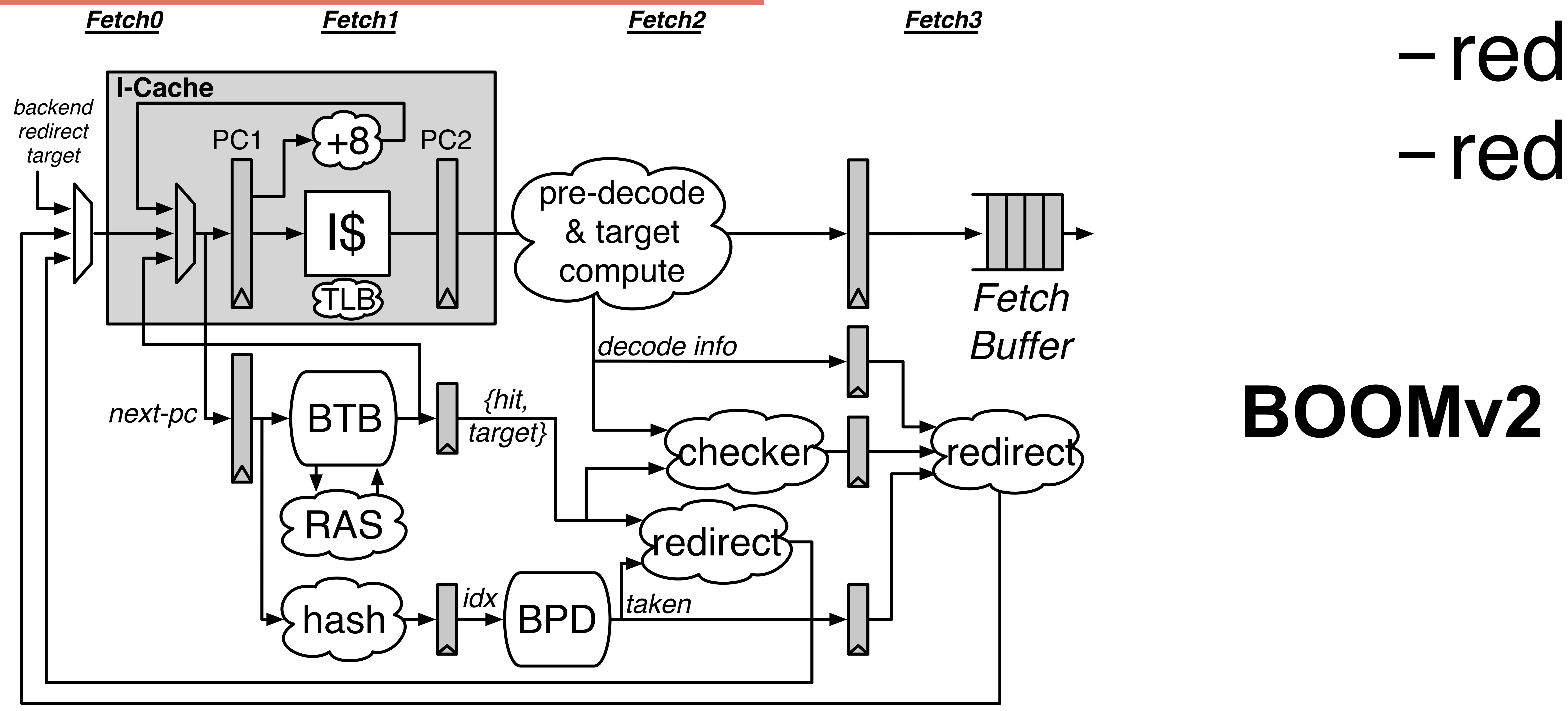


BOOM v2 (Aug 2017)

Frontend Design Changes

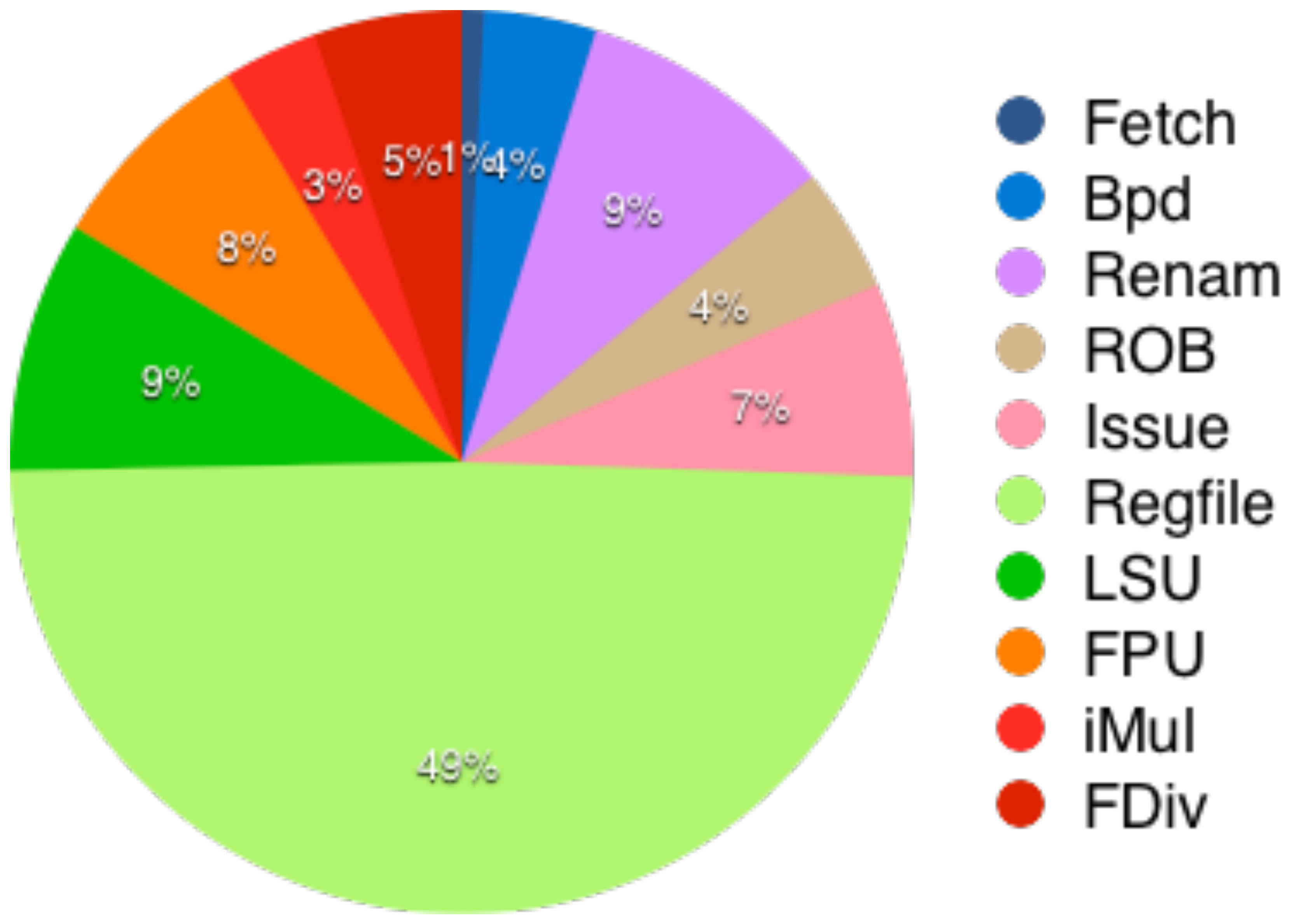


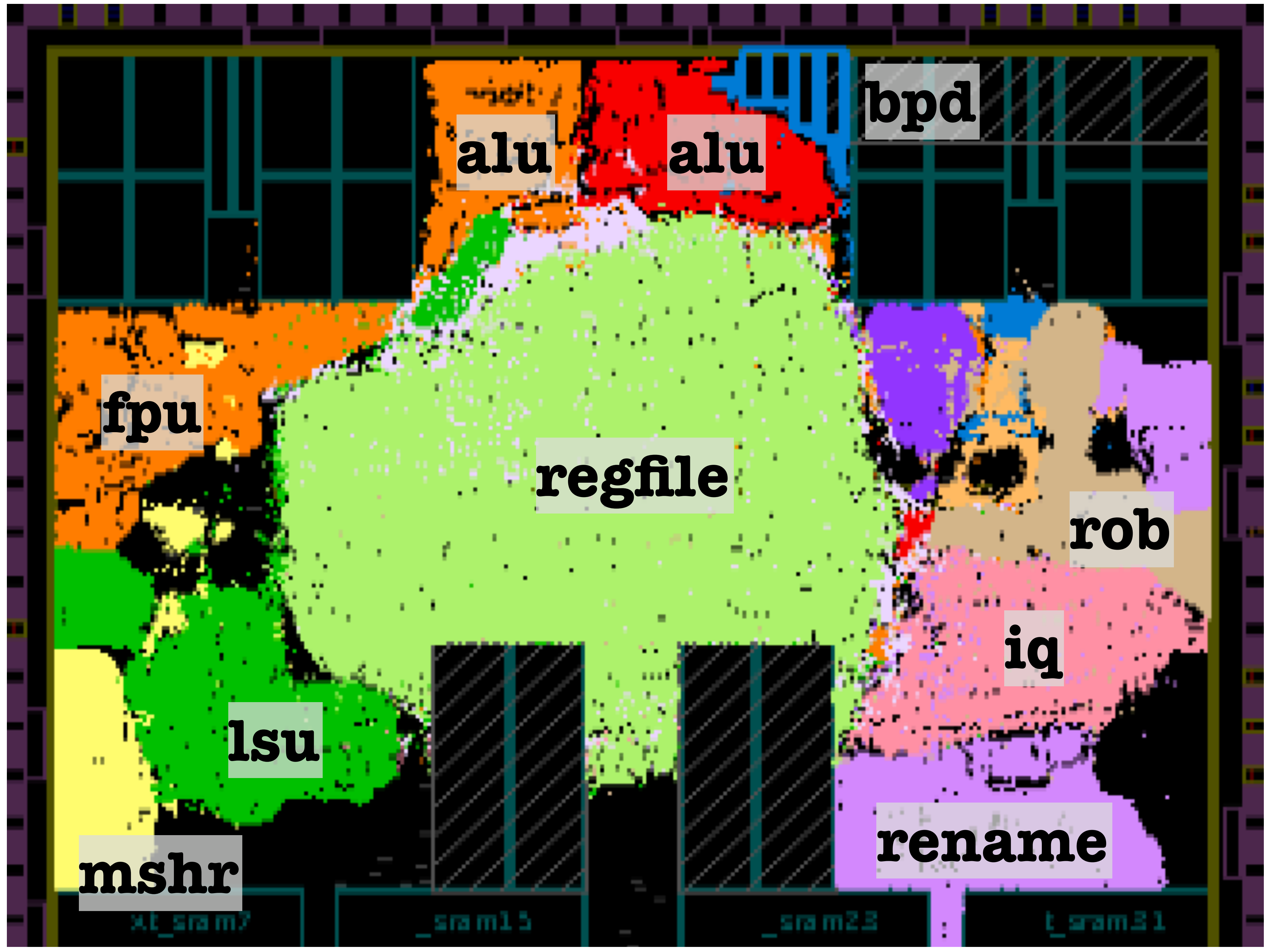
- BTB in SRAM
  - set-associative
  - partially tagged
  - Checker to verify integrity
- BPD (Conditional Predictor)
  - hash gets entire stage
  - redirect based on BTB
  - redirect pushed back (I$)



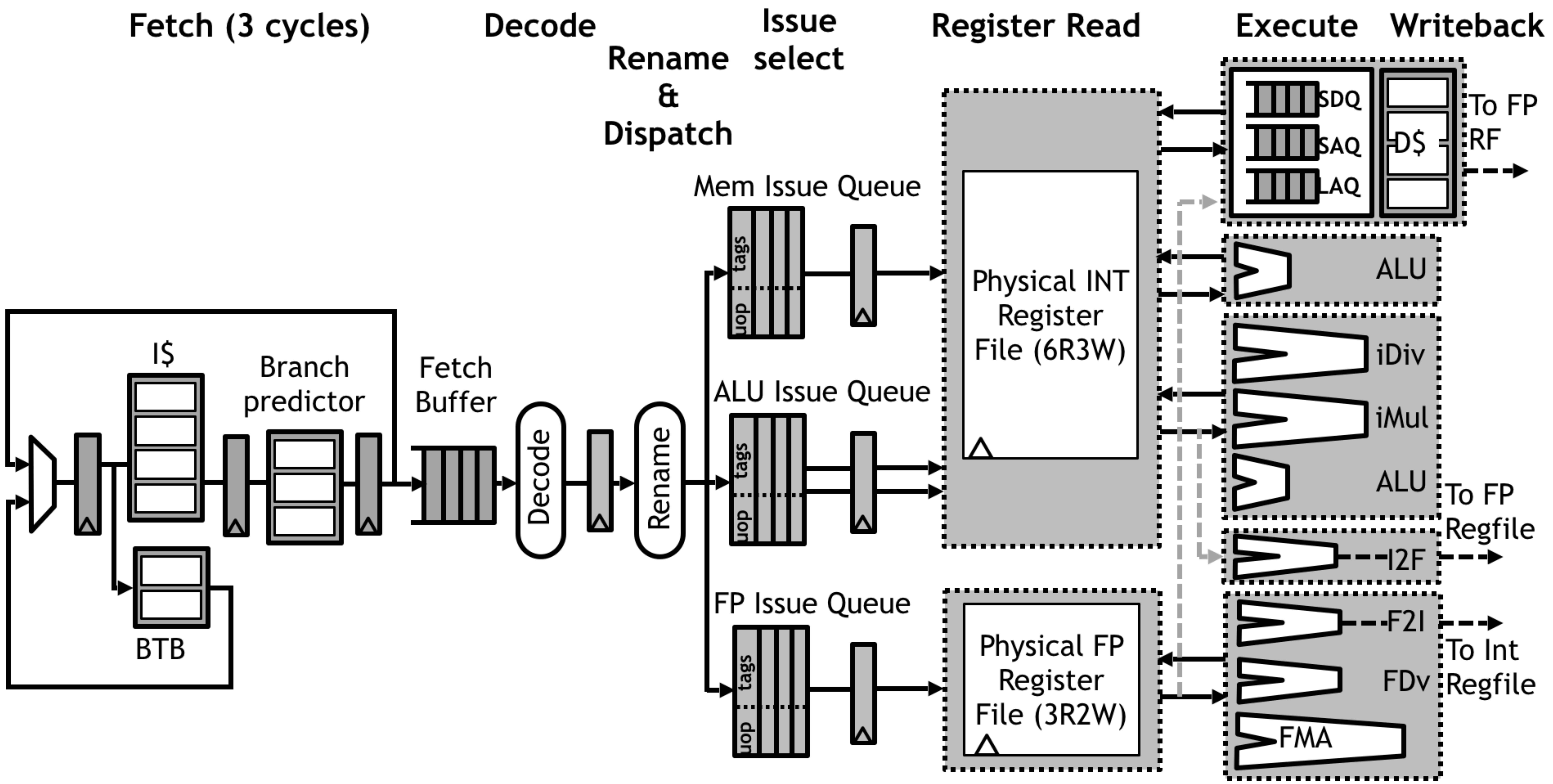
Building a Register File (the first P&R)

- BOOMv1 -- 7r3w with 110 registers (INT/FP)
- Initial Regfile design was infeasible for layout
- critical paths in issue-select and register read
- Not DRC/LVS clean



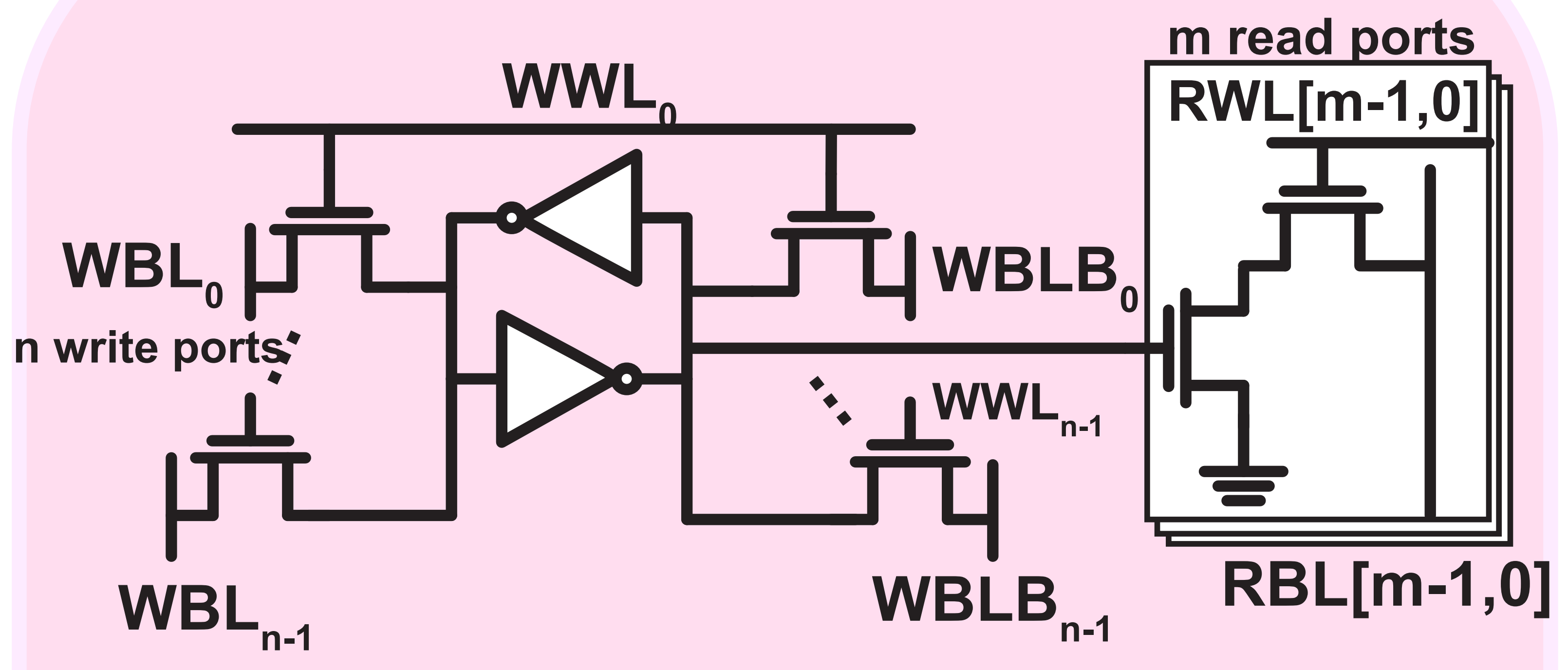


Splitting the Regfile and Issue Queues



Multi-port Register File for Design Exploration

Transistor-level



Advantage

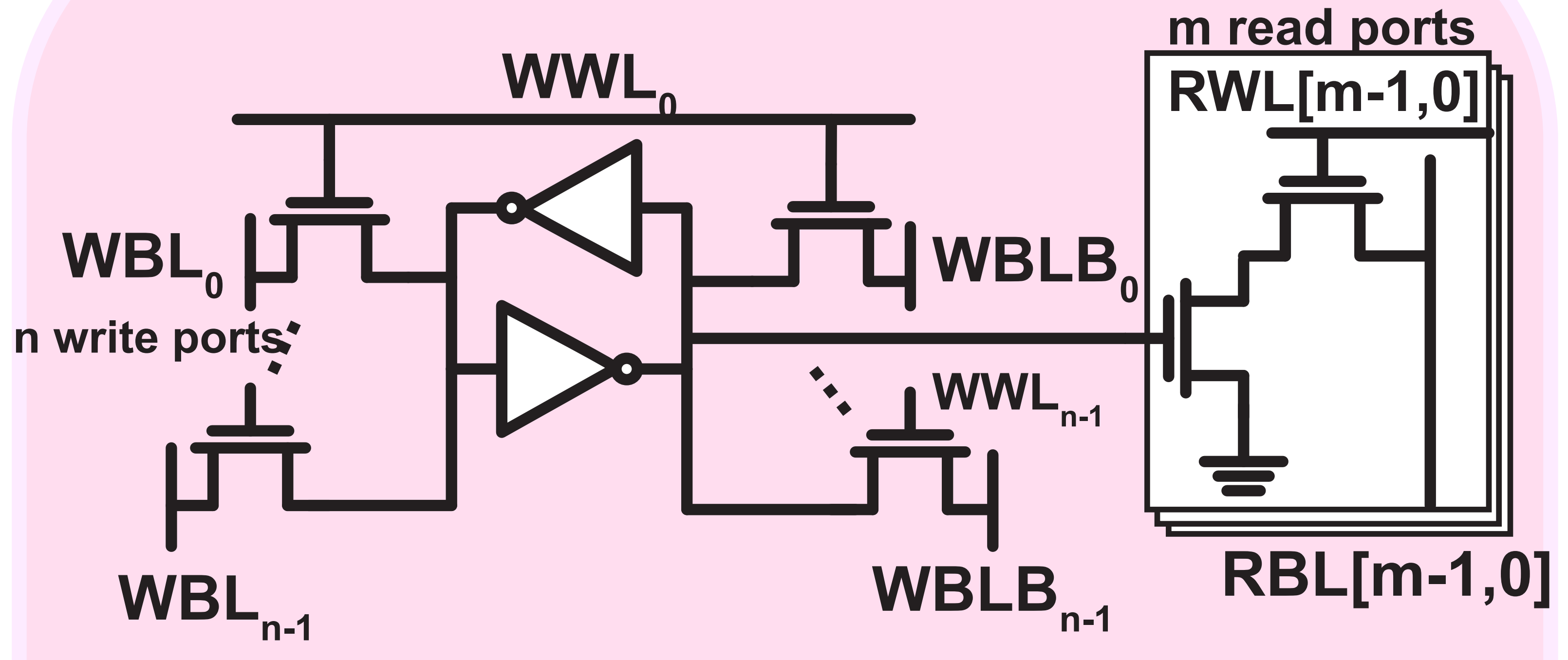
- Compact area
- Higher performance

Challenge

- Long design cycle
- Difficult for architecture design exploration

Multi-port Register File for Design Exploration

Transistor-level



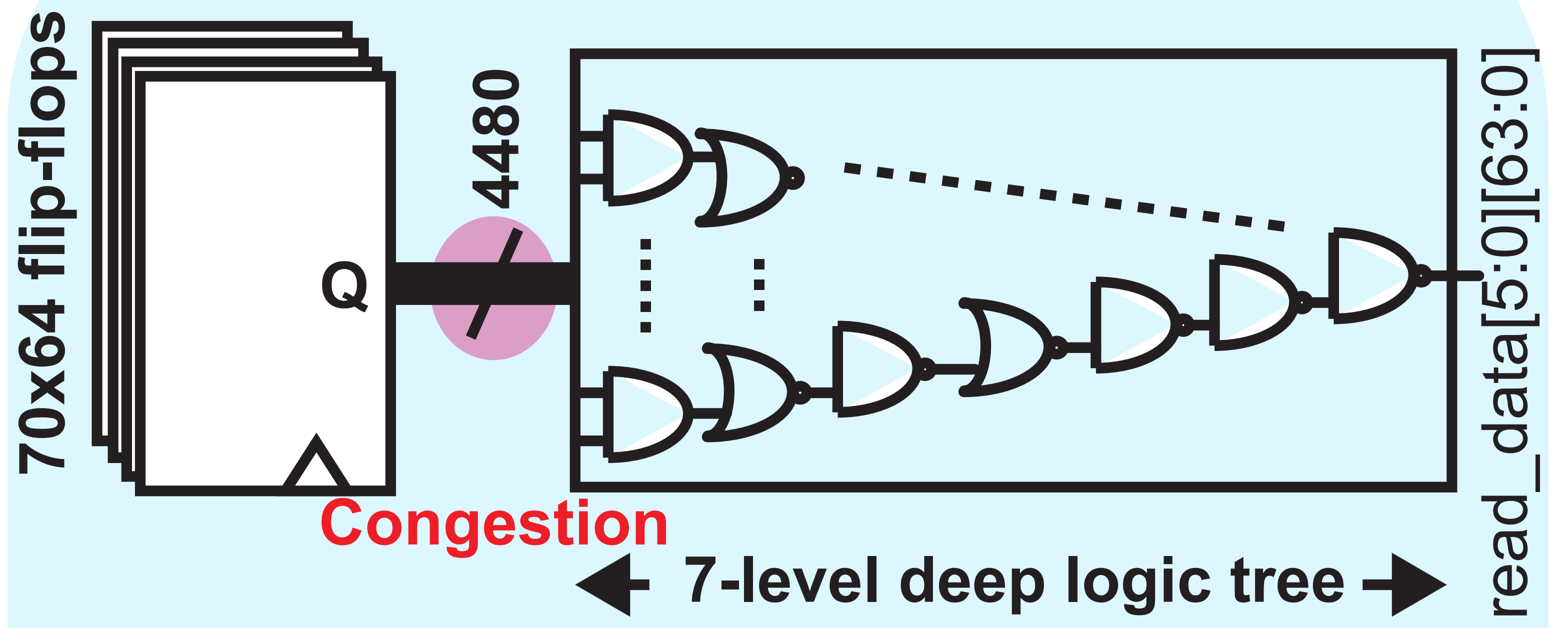
Advantage

- Compact area
- Higher performance

Challenge

- Long design cycle
- Difficult for architecture design exploration

RTL



Advantage

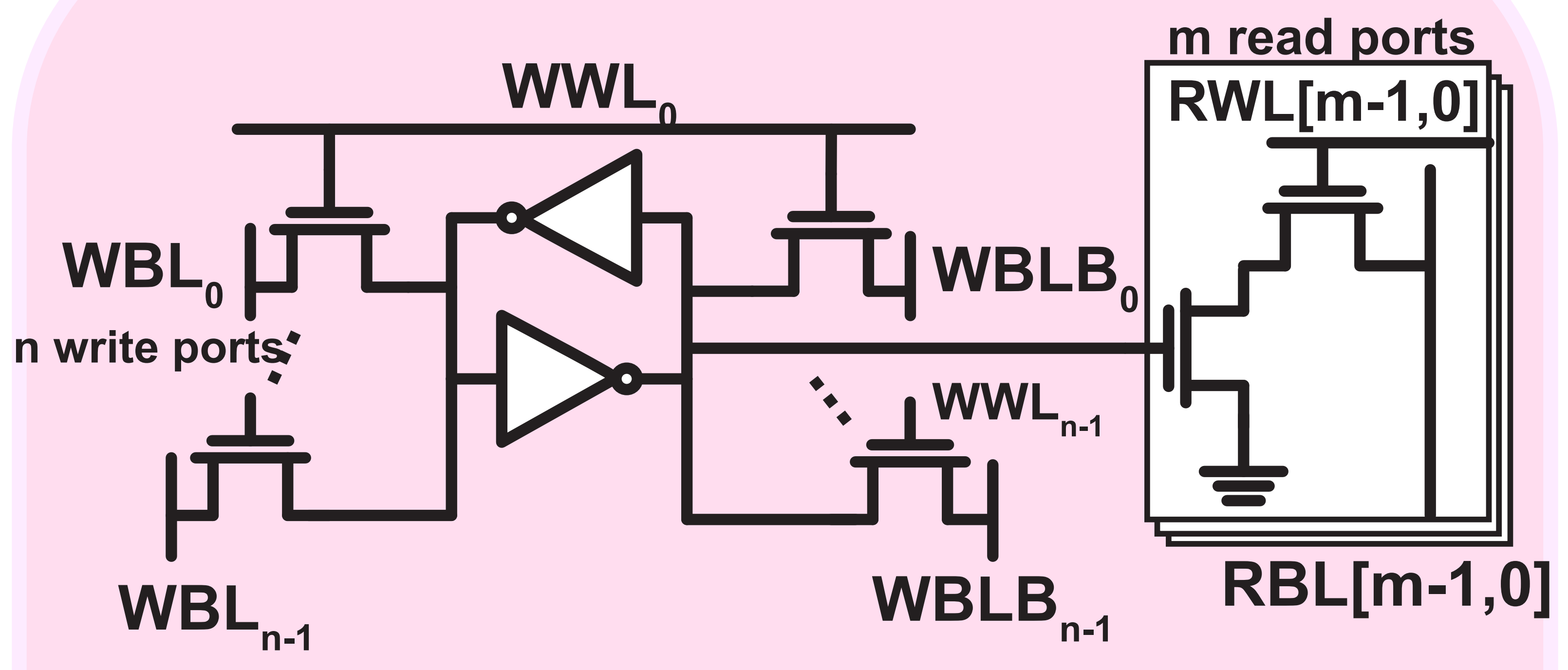
- Low design effort
- Rapid design exploration

Challenge

- Large area
- Bad performance
- Routing congestion

Multi-port Register File for Design Exploration

Transistor-level



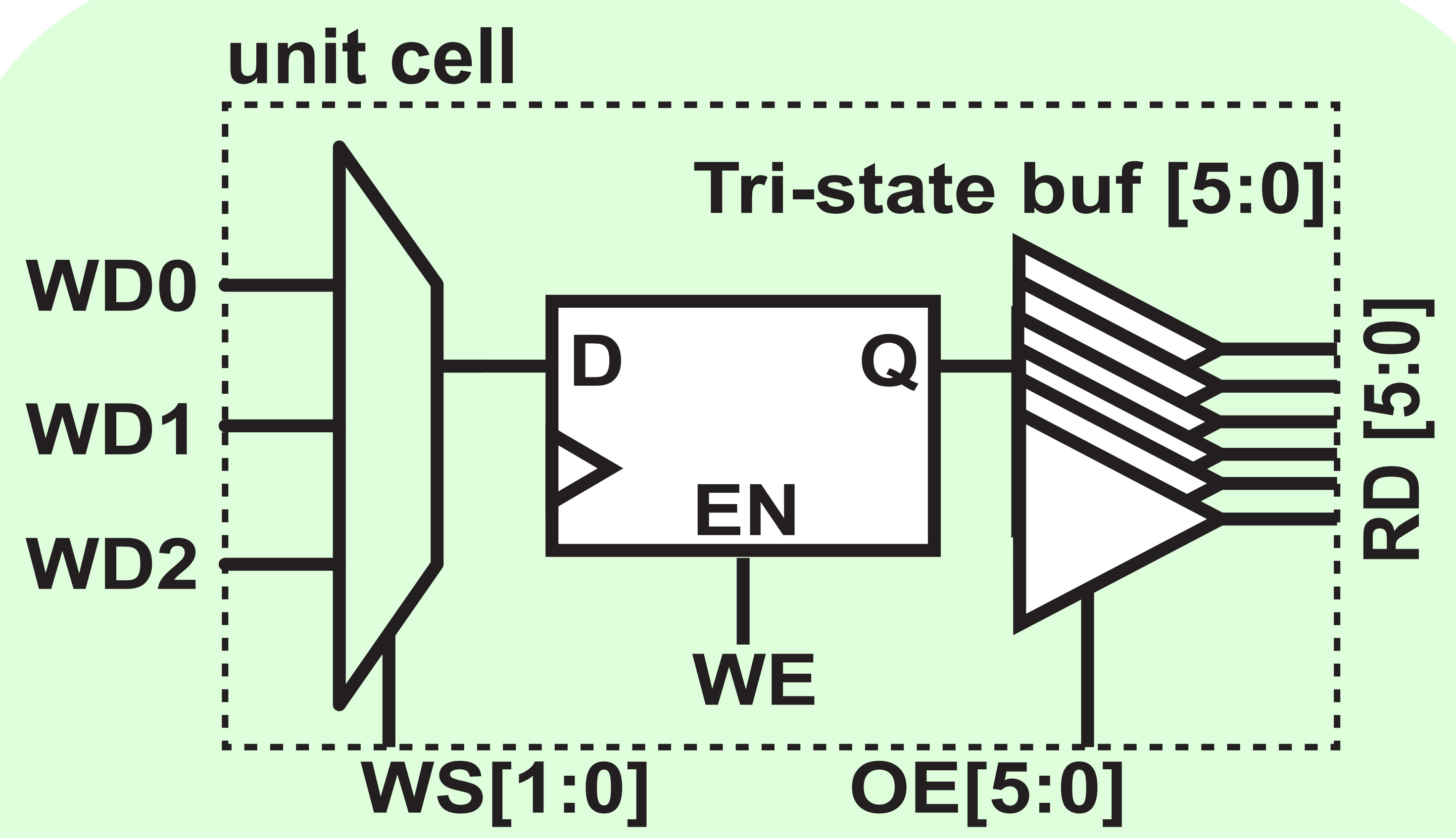
Advantage

- Compact area
- Higher performance

Challenge

- Long design cycle
- Difficult for architecture design exploration

Gate-level



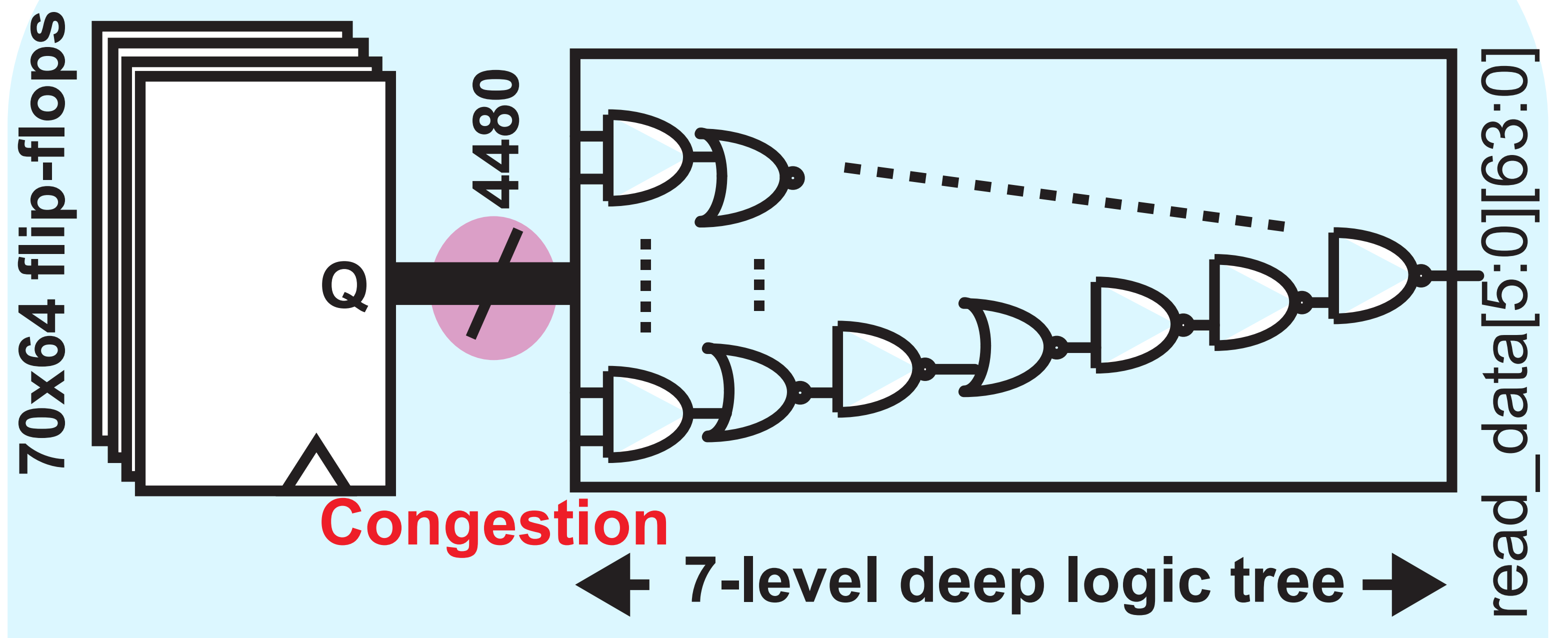
Advantage

- Rapid design exploration
- Shared read wires solve routing congestion

Challenge

- Guided place-and-route for area/performance optimization

RTL



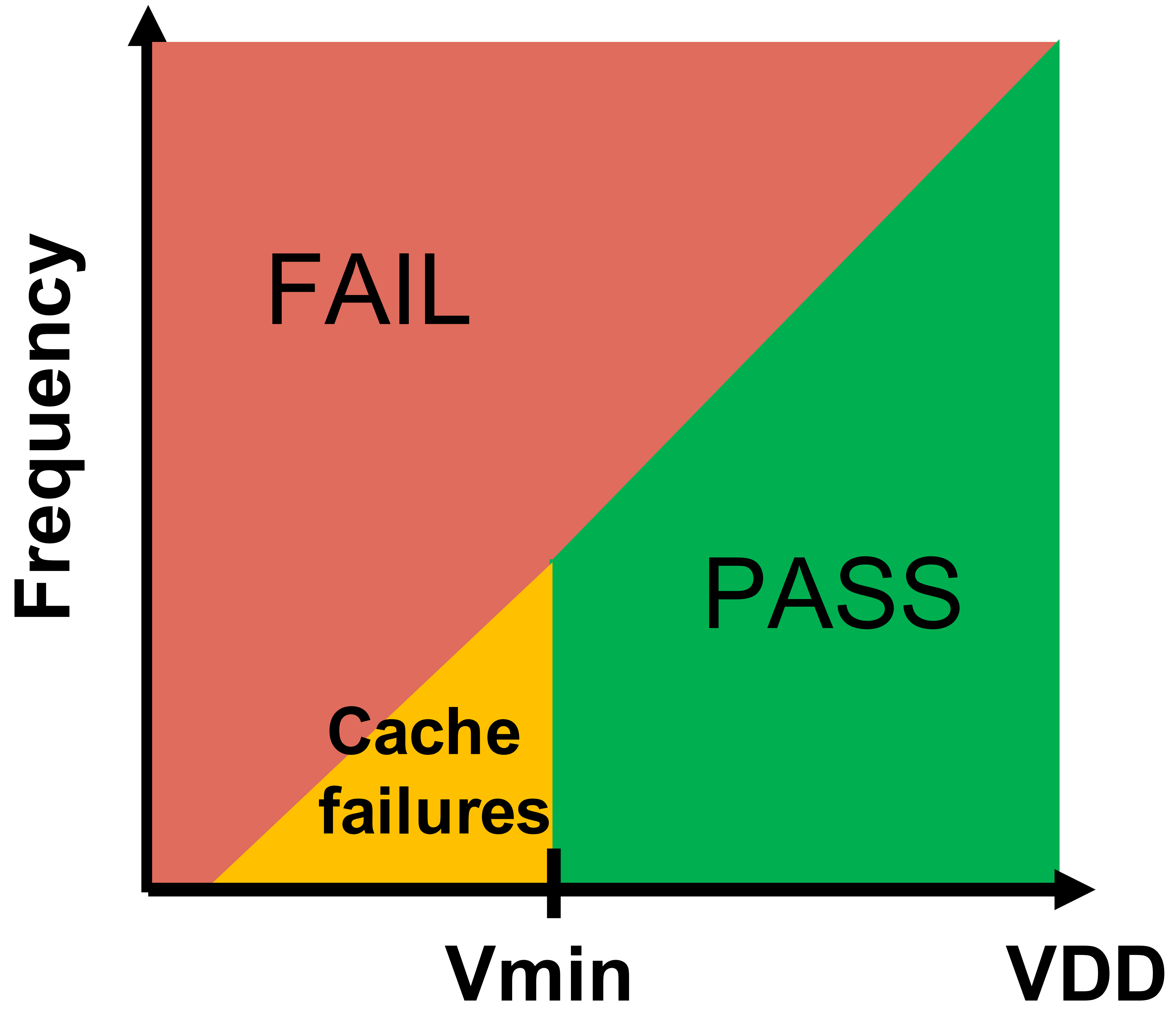
Advantage

- Low design effort
- Rapid design exploration

Challenge

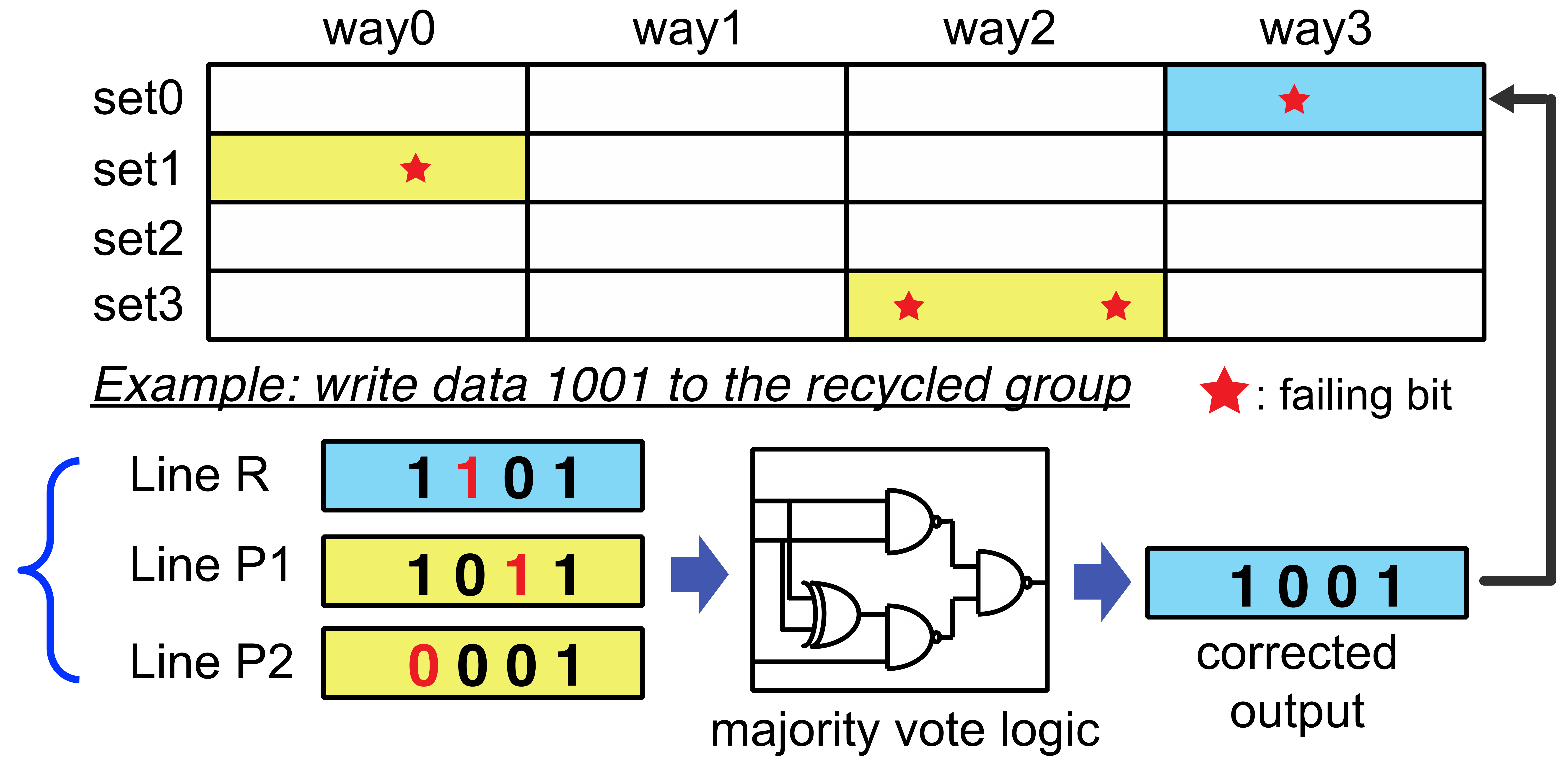
- Large area
- Bad performance
- Routing congestion

Resilient Cache for Energy Efficiency



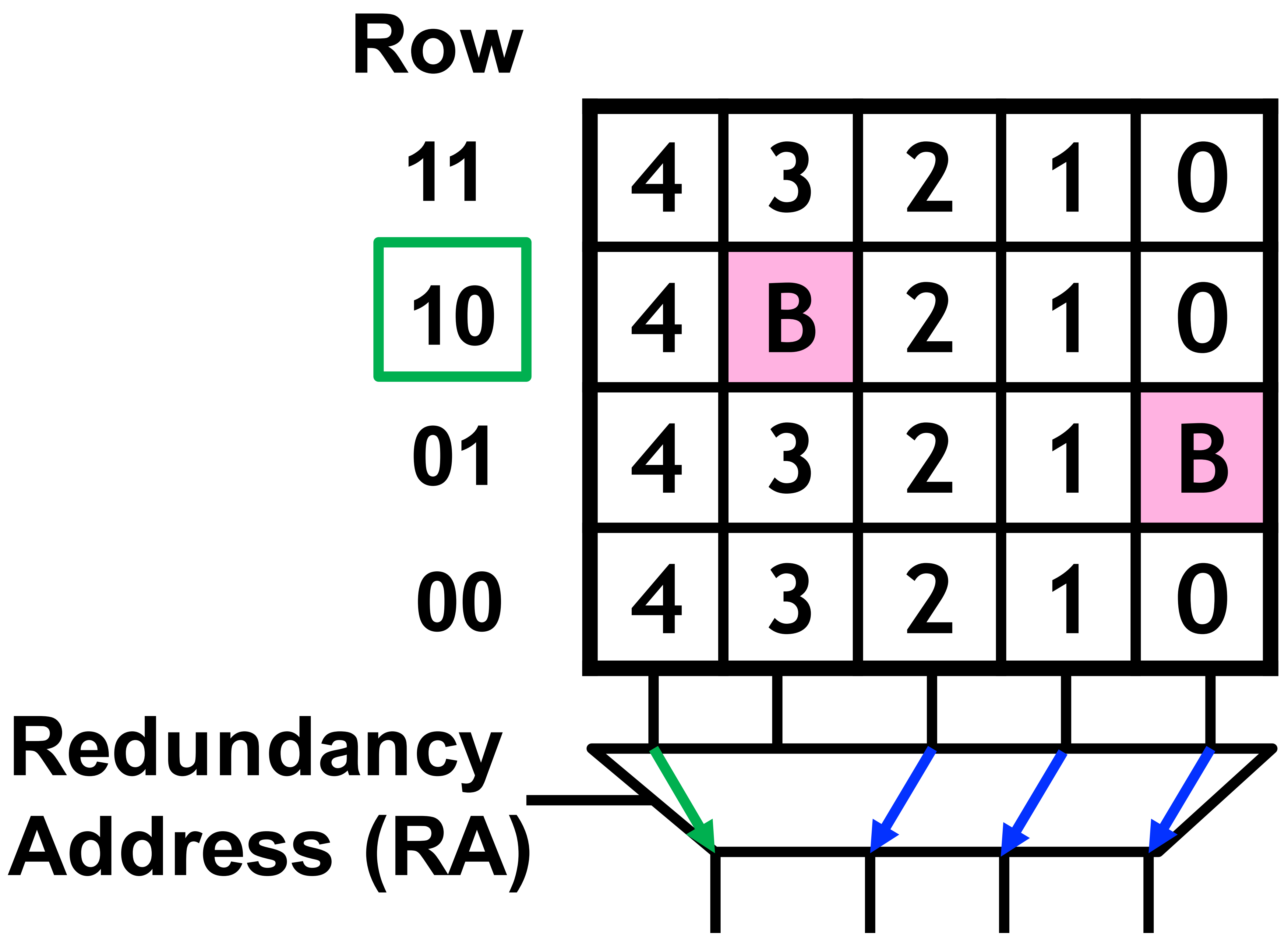
- Low-voltage operation improves energy efficiency
- SRAM-based cache fails at low voltages
- 50 mV reduction in VDD increases BER by 10x
- Architecture-level assist techniques can tolerate errors to reduce Vmin
- Only require RTL changes

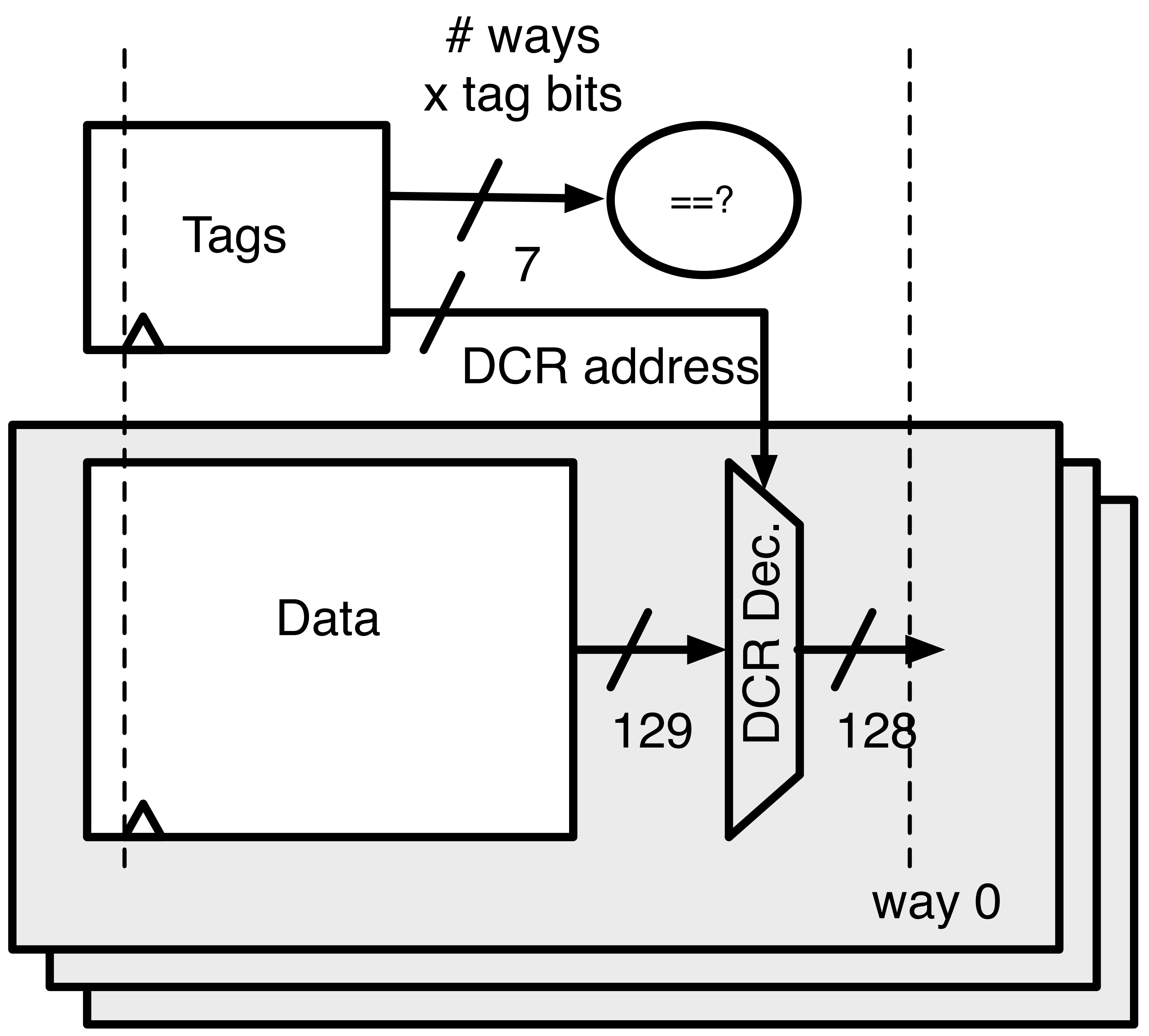
Line Recycling (LR)



- Line Disable (LD) avoids errors by disabling the faulty cache line
- Group three faulty lines with no error at the same column
- Two patch lines (line P1/P2) used to repair the recycled line (line R)
- A majority vote of line R/P1/P2 corrects the data output

Dynamic Column Redundancy (DCR)

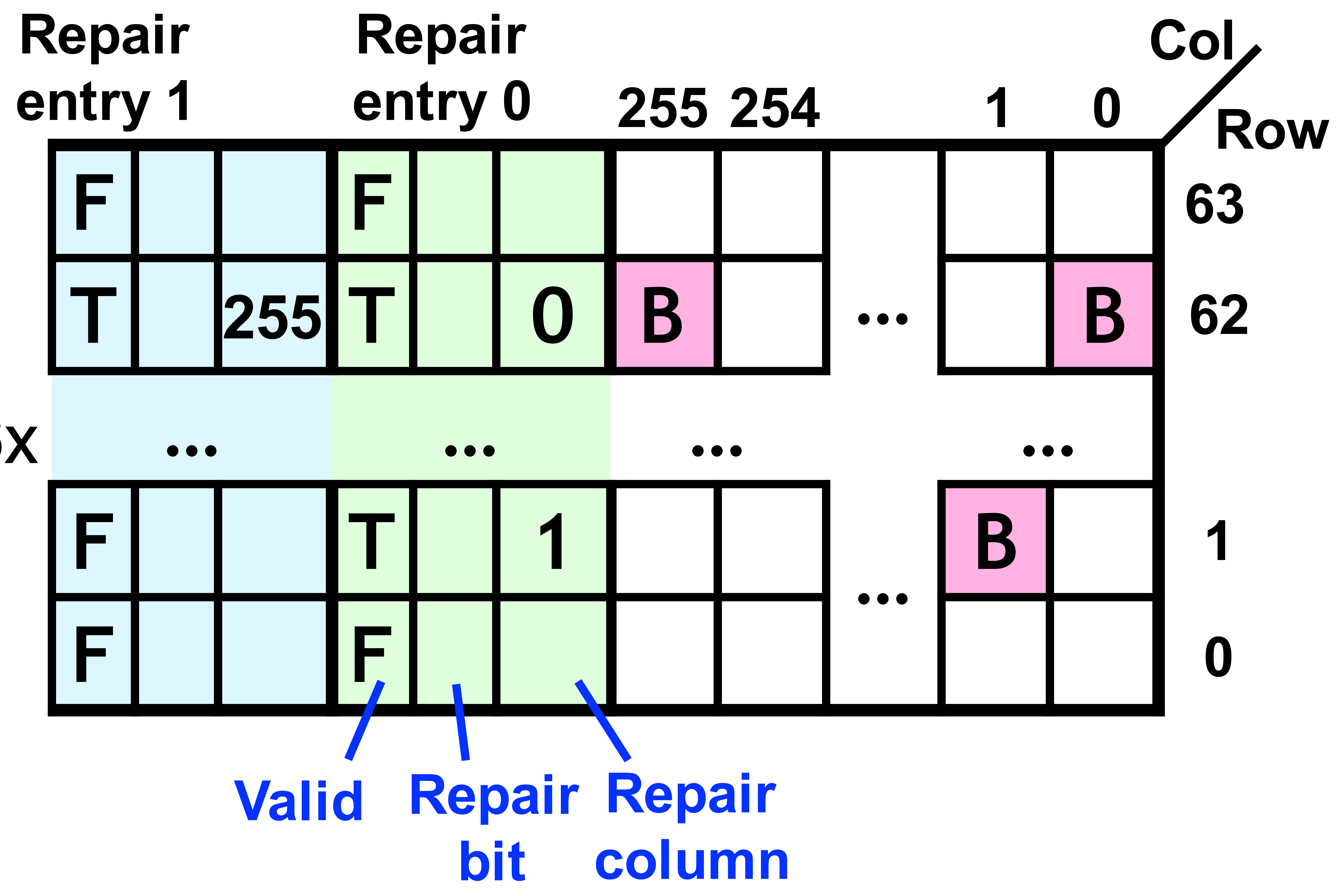




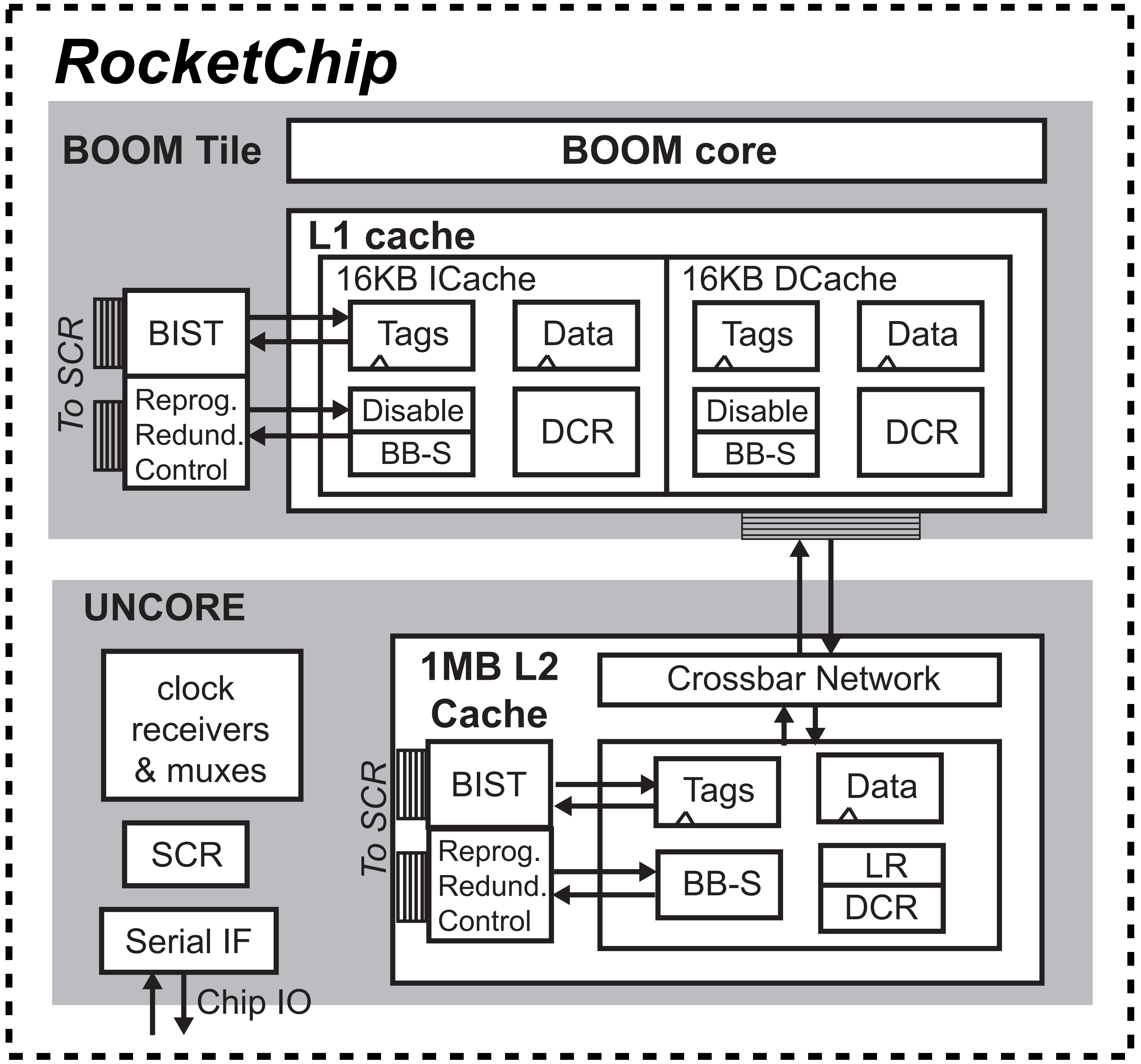
- DCR dynamically selects a different multiplexer shift to avoid the error according to the redundancy address (RA)
- Fix 1 bit per set
- Require LD/LR to handle multi-bit errors

Bit Bypass with SRAM (BB-S) for tag

- Expand tag arrays to store error entries
- Correct more error with less area
  - Fix 2 bits per row
  - An 8T SRAM bitcell is 6x smaller than a flip-flop
  - Shared decoder and peripheral circuit
  - Area overhead: 8.6% in tag arrays



Cache Resiliency Techniques



Technique	Protected cache	Timing overhead	Area overhead
LR	L2 data	Small §	0.77%
LD	L1/2 data	Small	0.2%
DCR	L1/2 data	Small	1.1% †
BB-S	L1/2 tag	Small	0.9% †
SECDED	L1/2	Large	10.9% ‡

§Require 3 additional cycles
†numbers are reported for L2
‡1repair/64bit
*Data portion is 86.2% of cache area,
tag portion is 11% of cache area

ASPIRE UC Berkeley

4 months of agile tape-out

clock frequency (MHz)
1400
1200
1000
800
600
400
200
0
April 10
May 01
May 22
June 12
July 03
July 24
August 14
2017
early BOOMv2 prototyping
BOOMv1 analysis
physical design focus
RVT cells
LVT cells
par (SS) - tapeout
syn (SS) - tapeout
par (SS)
syn (SS)
par (TT)
syn (TT)

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

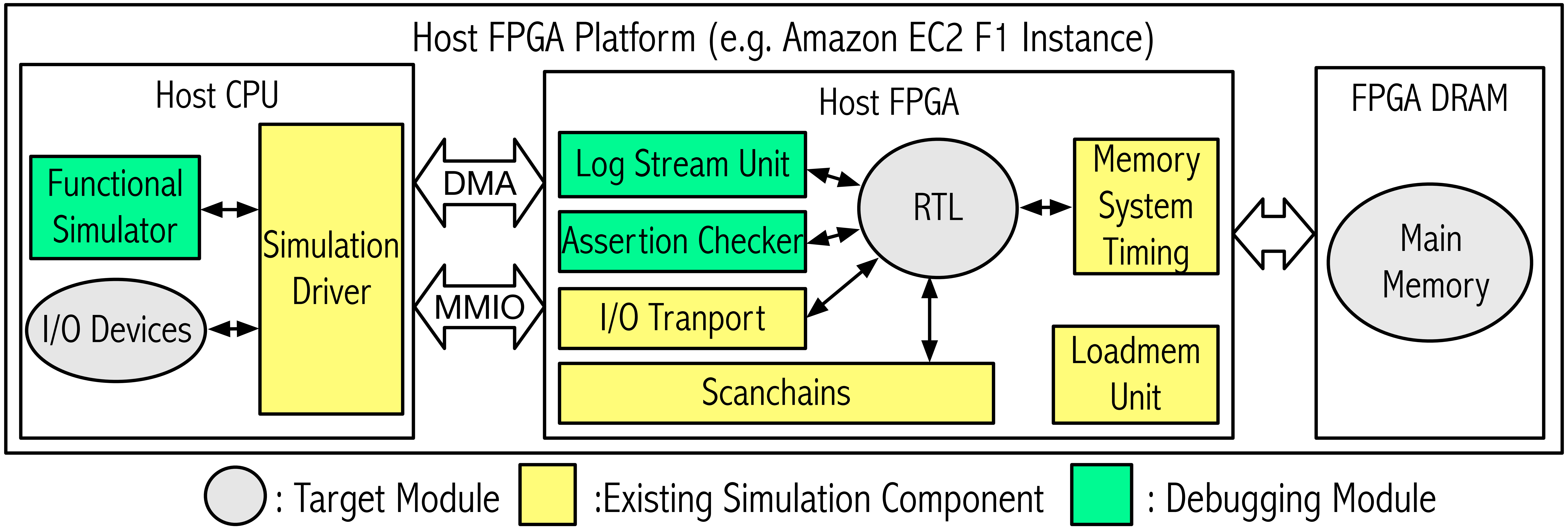
Agile Hardware Development

- 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
  - RTL and PD are tightly coupled
- Verification is a bottleneck
  - I can write bugs faster than I can find them

Verification

- Directed tests and a randomized torture generator.
- Verilator/VCS/FPGA simulation at RTL.
- VCS for post-gl/par simulation.
- Speculative OOO pipelines are difficult to get good coverage on.
  - Need tests that build up a lot of speculative state.
  - Need tests that cover OS- and platform-level use-cases.
- Assertions are king.

DESSERT: Debugging RTL Effectively with State Snapshotting for Error Replays across Trillions of Cycles



- Co-simulate, find bugs, and get waveforms from Cloud FPGA-based simulation!
- Donggyu Kim, et. al. CARRV 2018
- https://carrv.github.io/2018/papers/CARRV_2018_paper_10.pdf

Incorrect Jump Target

- 401.bzip2 (assertion error at 500 billion cycles)
  - JAL jumps to wrong target.
  - Due to improper signed arithmetic.
  - 2-3 year old bug.
  - 3 hours of FPGA time.

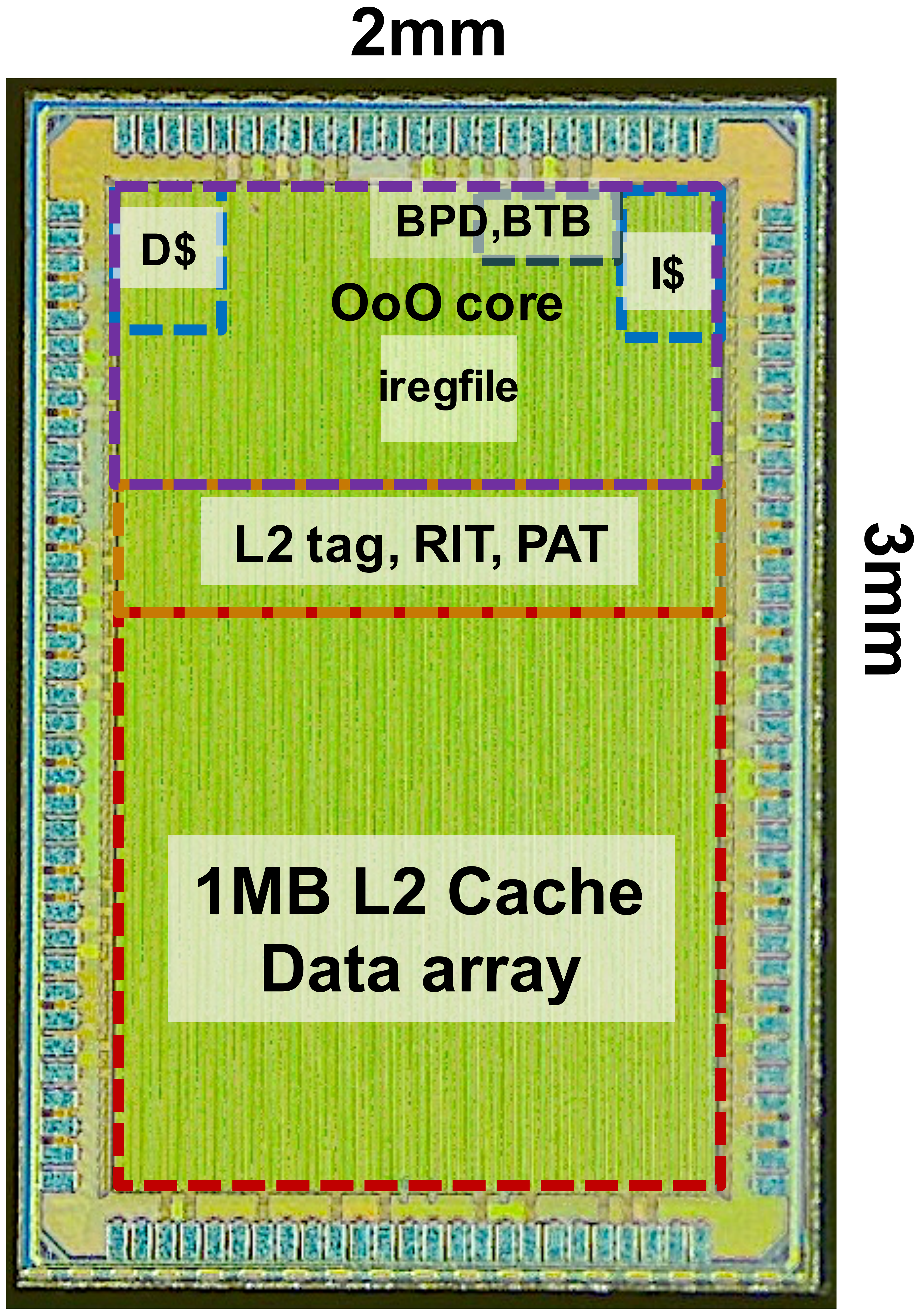
Incorrect Jump Target

- 401.bzip2 (assertion error at 500 billion cycles)
  - JAL jumps to wrong target.
  - Due to improper signed arithmetic.
  - 2-3 year old bug.
  - 3 hours of FPGA time.
  - Would require 39 years of Verilator simulation to find.
  - DESSERT found this via a synthesized assertion.

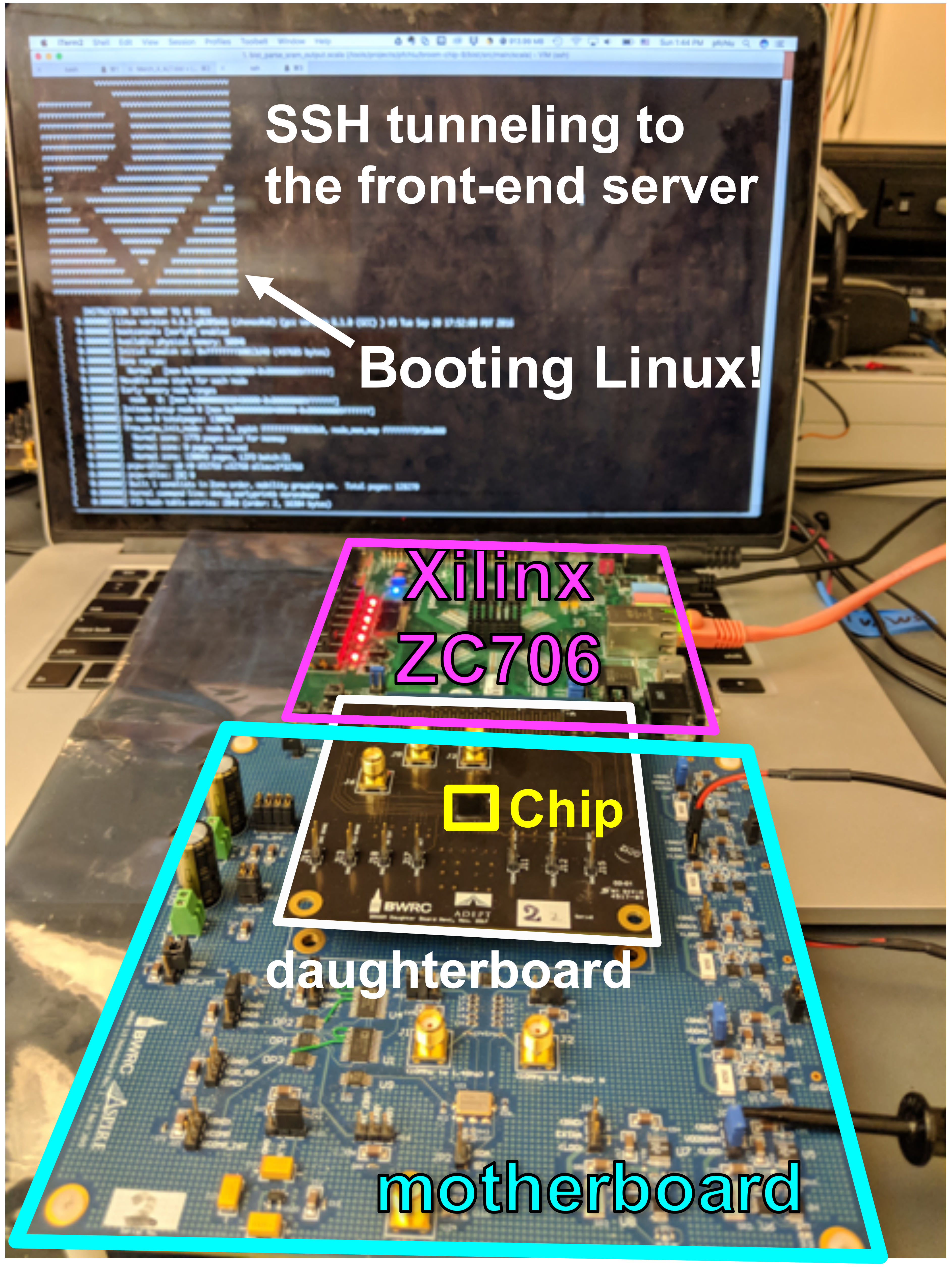
ASPIRE UC Berkeley

Chip Implementation

Chip Summary	
ISA	RISC-V RV64IMAFD with Sv39
Fetch Width	2 insts
Issue Width	3 micro-ops
Issue Entries	16 (i) 20 (m) 10 (f)
Regfile	6R3W (int), 3R2W (fp)
Exe Units	iALU+iMul+FMA iALU+fDiv Load/Store
L1 I/D Cache	4-way, 16KB
L2 Cache	8-way, 1MB



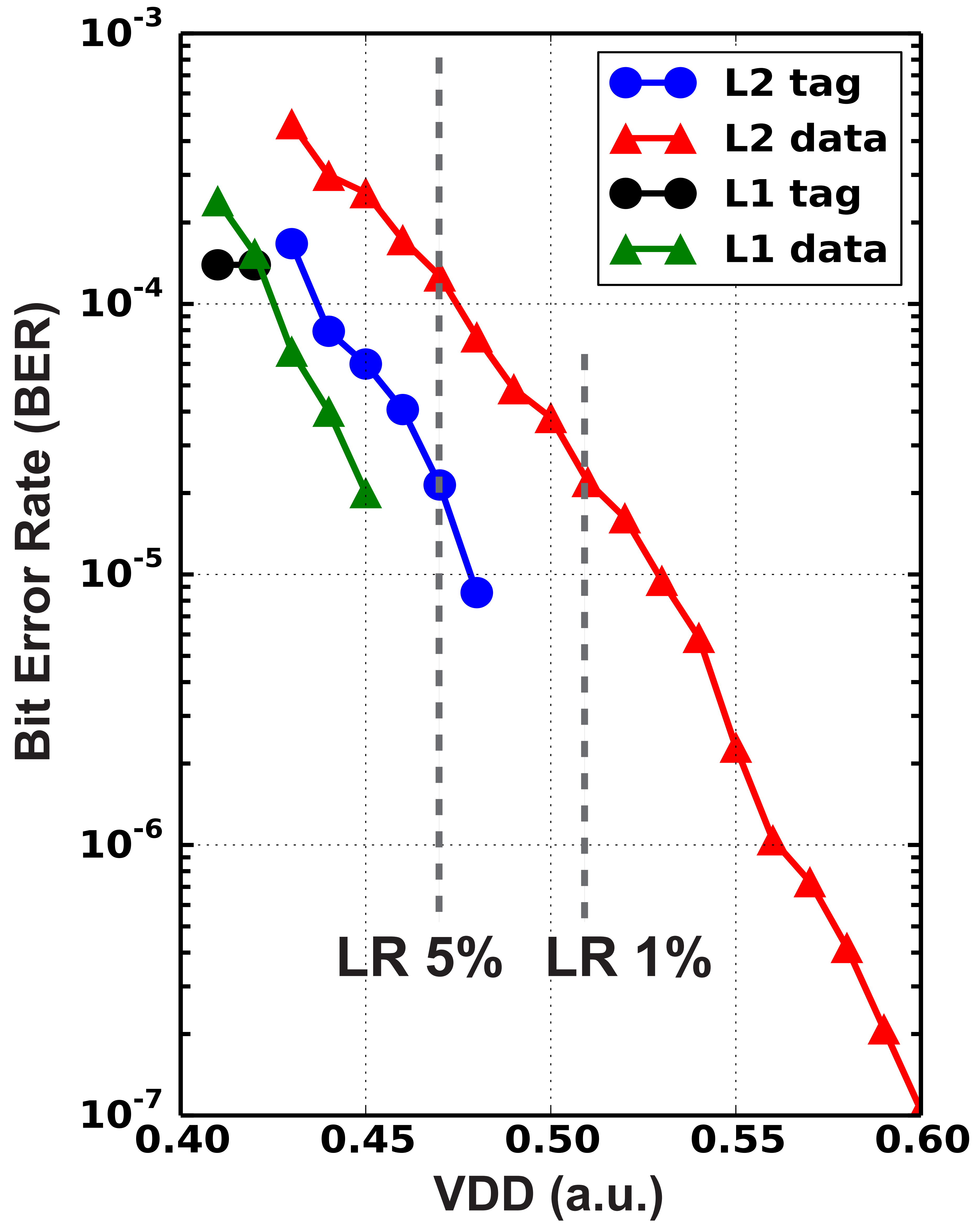
Experimental Setup

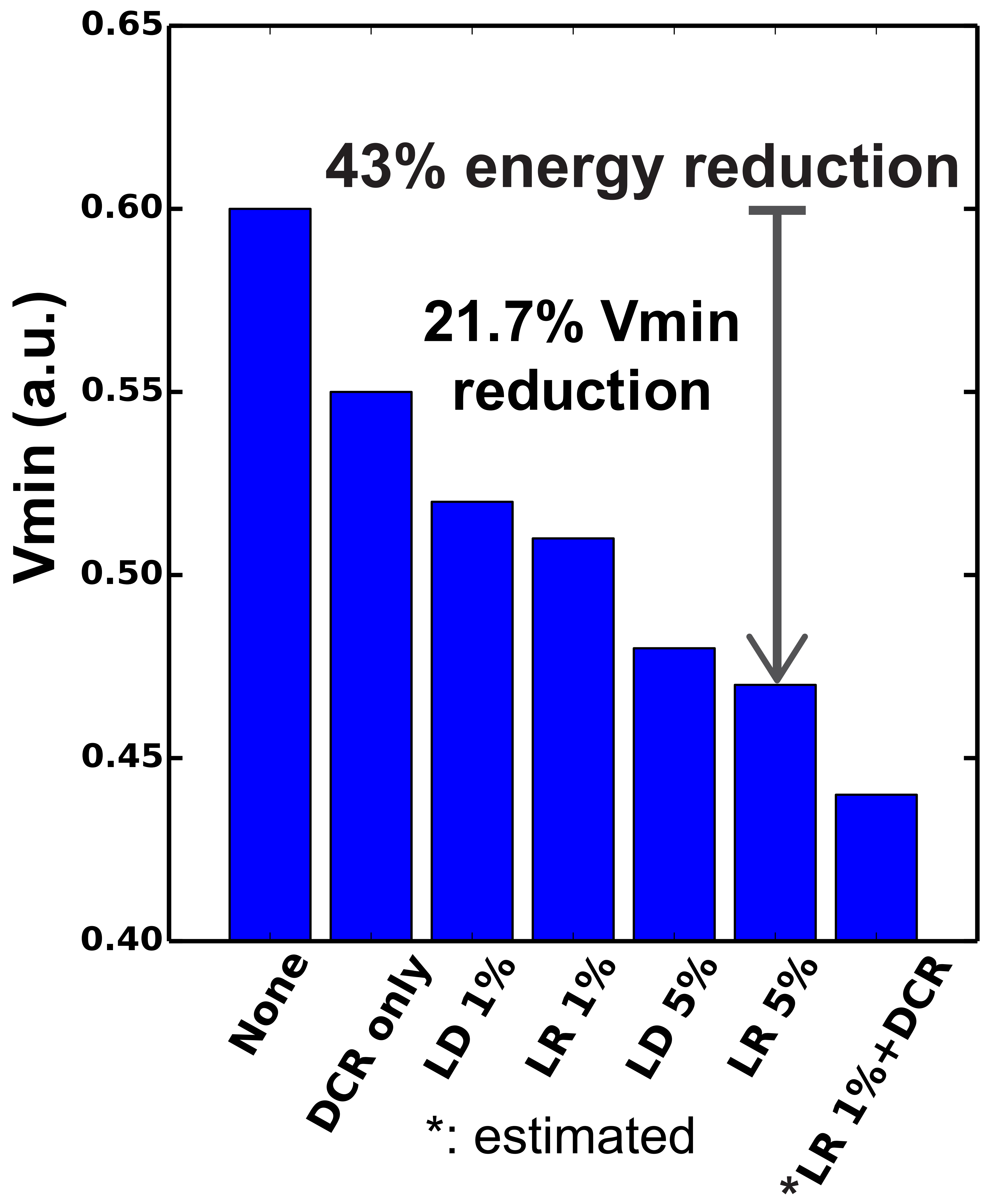


- Chip-on-board (COB) package
- Voltage and clock generation on the motherboard
- Cortex A9 on ZC706 works as the front-end server
- Boot Linux

Performance	
Clock frequency	1GHz @0.9V
	320MHz @0.6V
Coremark/MHz	3.77
Instruction Per Cycle	1.11 (@Coremark)

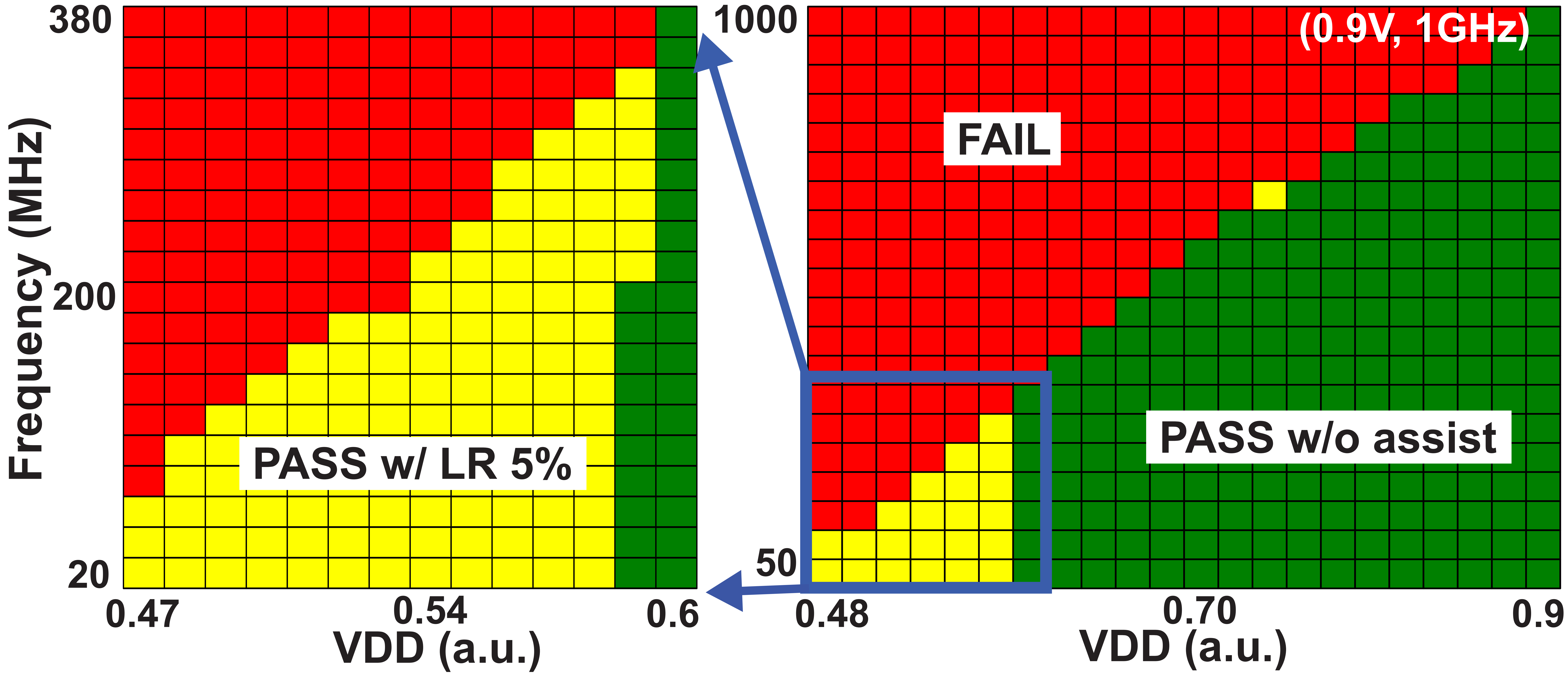
Bit Error Rate and Vmin reduction





Operating voltage and frequency

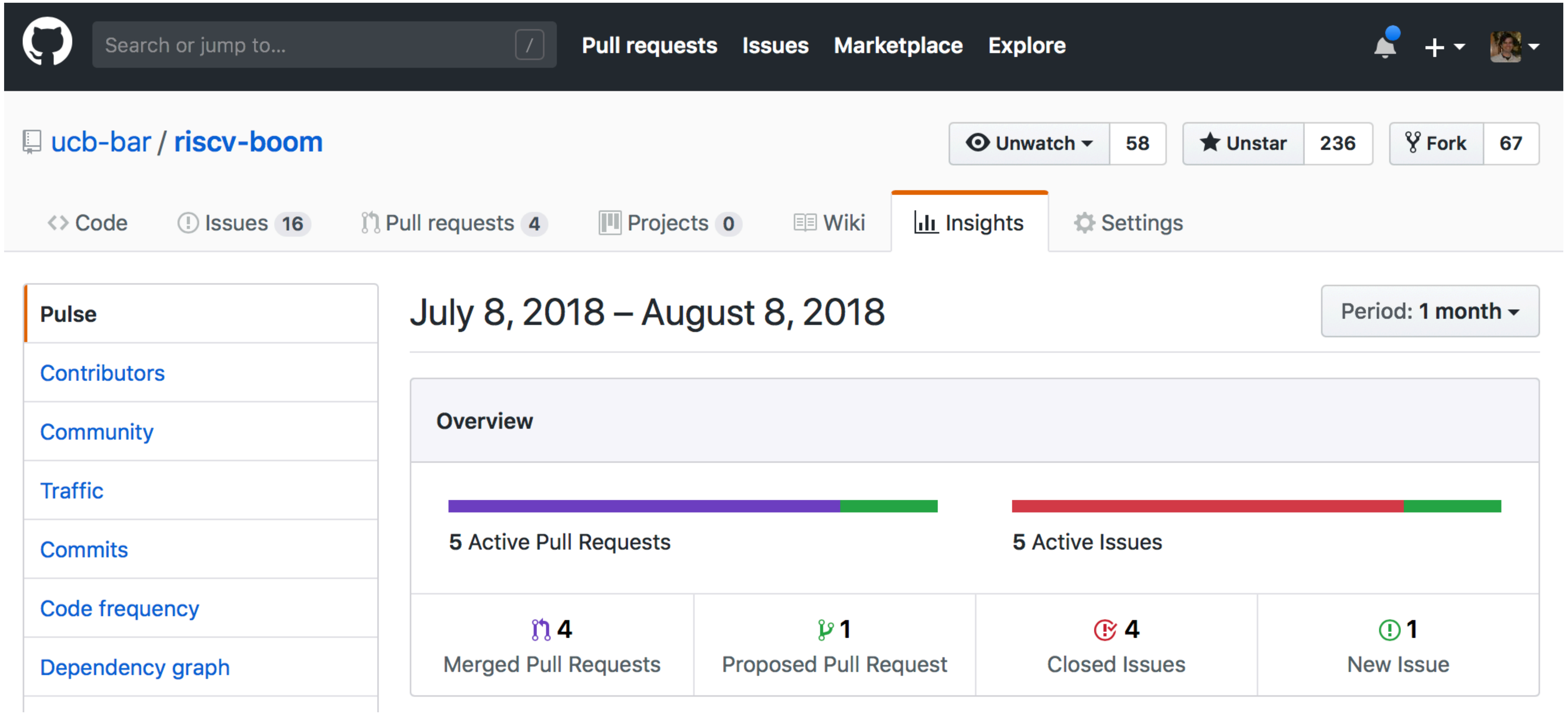
Benchmark: vvadd



- With LR and 5% loss of L2 cache capacity, Vmin is reduced to 0.47V@70MHz
- 2.3% increase in L2 misses, but only 0.2% degradation in IPC

Future Directions

- Pi-Feng and Chris have graduated!
- BOOM will continue to be supported and improved.
  - github.com/ucb-bar/riscv-boom



July 8, 2018 – August 8, 2018

Overview

5 Active Pull Requests

5 Active Issues

4 Merged Pull Requests	1 Proposed Pull Request	4 Closed Issues	1 New Issue

Lots of opportunities for using the BOOM core

Lots of opportunities for using the BOOM core

- Agile Methodology Research
  - How to verify complex IP?
  - How do you measure/predict RTL performance, area, power?

Lots of opportunities for using the BOOM core

- Agile Methodology Research
  - How to verify complex IP?
  - How do you measure/predict RTL performance, area, power?
- Software Studies
  - Hardware/software co-design.
  - High visibility of very long-running applications.

Lots of opportunities for using the BOOM core

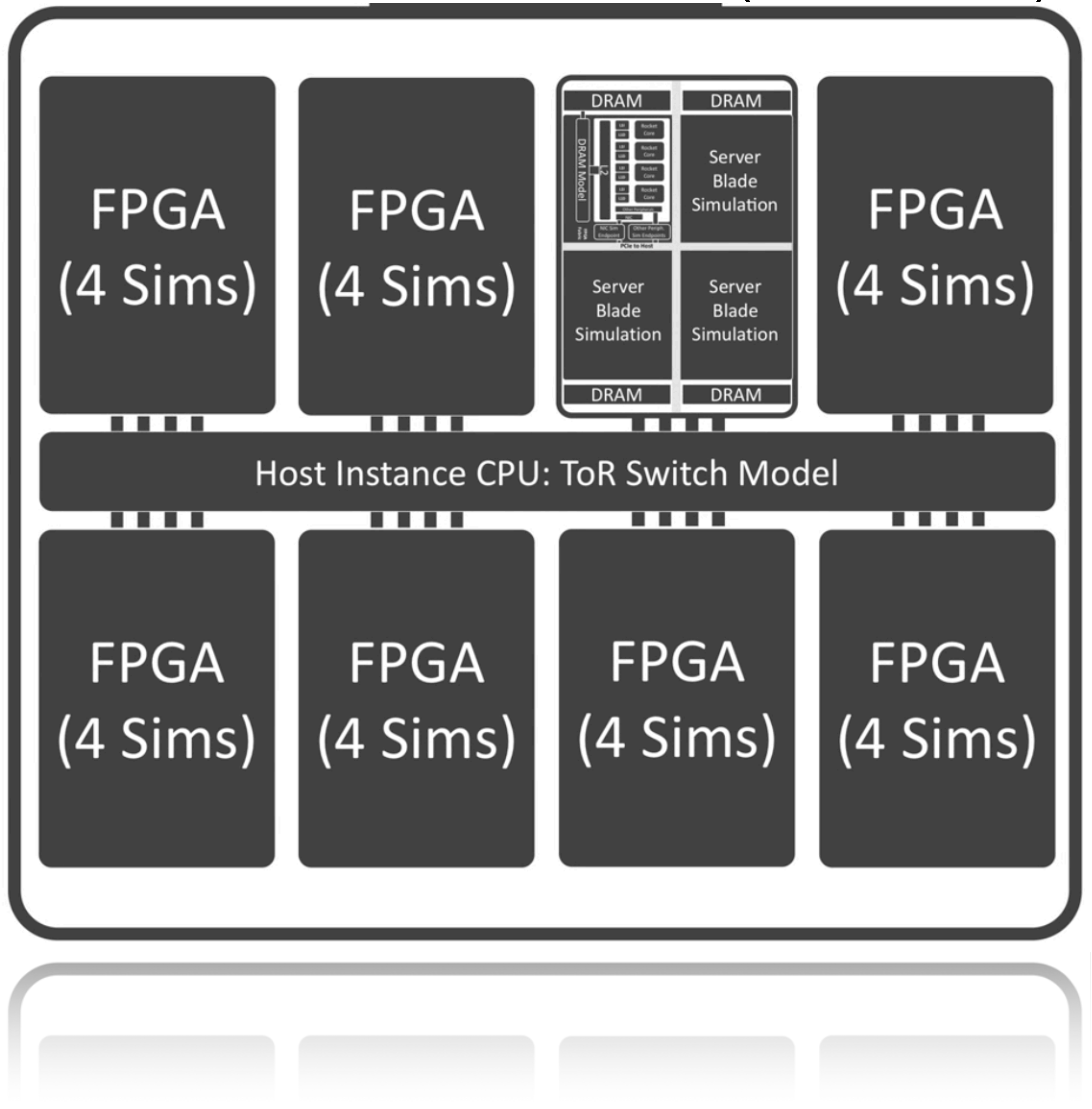
- Agile Methodology Research
  - How to verify complex IP?
  - How do you measure/predict RTL performance, area, power?
- Software Studies
  - Hardware/software co-design.
  - High visibility of very long-running applications.
- Security Research
  - New class of speculation-based attacks.
  - How to attack?
  - How to defend?
  - Evaluate cost of changes to branch predictors, caches, and more.

Lots of opportunities for using the BOOM core

- Agile Methodology Research
  - How to verify complex IP?
  - How do you measure/predict RTL performance, area, power?
- Software Studies
  - Hardware/software co-design.
  - High visibility of very long-running applications.
- Security Research
  - New class of speculation-based attacks.
  - How to attack?
  - How to defend?
  - Evaluate cost of changes to branch predictors, caches, and more.
- New RISC-V extensions
  - Variable-length vector.
  - Managed-language support.

FireSim now supports BOOM!

Shown: a 32 node rack (128 cores)



- FireSim is a open-source cycle-accurate FPGA-accelerated simulation tool that runs on Amazon EC2 F1
- Chisel RTL is automatically transformed into cycle-accurate FPGA simulator
- Peripheral device support:
  – UART, Disk, Ethernet NIC, easy to add more
- **Boot Linux on a multi-core BOOM with 16 GB DDR3, UART, Ethernet NIC in the cloud for 50 cents/hour at ~100 MHz**
- FireSim is available at:
  – **https://fires.im**
- ISCA 2018 Paper:

https://sagark.org/assets/pubs/firesim-isca2018.pdf

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:
  - 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, Jerry Zhao, and more...

Thank you!

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.*