

Replicating and Mitigating Spectre Attacks on an Open Source RISC-V Microarchitecture

CARRV 2019 – June 22nd, 2019 - Phoenix, Arizona

Abraham Gonzalez, Ben Korpan, Jerry Zhao, Ed Younis

Krste Asanović

University of California, Berkeley

Berkeley
UNIVERSITY OF CALIFORNIA



| **Berkeley
Architecture
Research**



Outline



- Motivation
- Open-source Approach to Hardware
 - BOOM: Berkeley Out-of-Order Machine
- Replicating Spectre Attacks on BOOM
- Implementing a Speculation Buffer
 - Comparisons
 - Implementation
- Conclusion

Motivation

Exploits Everywhere



Researchers discover seven new Meltdown and Spectre attacks

Experiments showed that processors from AMD, ARM, and Intel are affected.



By [Catalin Cimpanu](#) for [Zero Day](#) | November 14, 2018 -- 14:44 GMT (06:44 PST) | Topic: [Security](#)

SPOOKY ACTION AT A DISTANCE —

New Spectre attack enables secrets to be leaked over a network

It's no longer necessary to run attacker code on the victim system.

PETER BRIGHT - 7/26/2018, 2:40 PM

Intel LazyFP vulnerability: Exploiting lazy FPU state switching

June 6 2018

Beyond Spectre: Foreshadow, a new Intel security problem

Researchers have broken Intel's Software Guard Extensions, System Management Mode, and x86-based virtual machines.

Speculative Store Bypass explained: what it is, how it works

May 21, 2018 | [Jon Masters](#), chief ARM architect, Red Hat

Researchers discover SplitSpectre, a new Spectre-like CPU attack

Why are Spectre-style attacks hard?



Attack Scenarios

- User process attacks kernel
- User process attacks user space
- Intra-process sandbox escape
- User process attacks enclaves
- Remote timing attacks
- ...

Covert Channels

- Changes in cache state
- Power consumption
- Resource contention (FPUs, buffers)
- ...

Spectre Variations

Leakage Mechanisms

- Conditional branch
- Indirect jump
- Return instructions
- Speculative store bypass
- Data speculation
- ...

Target CPUs

- ARM
- Intel
- AMD
- RISC-V
- ...

Mitigation Approaches



InvisiSpec/SafeSpec: Blocking unsafe loads from altering the data cache

DAWG: Partition data cache between security domains

StealthMem/CATalyst: Hide visibility of a secure memory region

Context-based fencing: Dynamically stop speculation in secure code

Compiler-inserted fencing: Statically analyze program for Spectre-vulnerable snippets

Lots of interesting approaches, but how to compare them?

Use them together?

M. Yan, et. al. 2018. InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy. In MICRO.
K. N. Khasawneh, et. al. 2018. Safespec: Banishing the spectre of a meltdown with leakage-free speculation. Archived.
V. Kiriansky, et. al. 2018. DAWG: A Defense Against Cache Timing Attacks in Speculative Execution Processors. In MICRO.
T. Kim, et. al. 2012. STEALTHMEM: System-Level Protection Against Cache-Based Side Channel Attacks in the Cloud. In USENIX.
F. Liu, et. al. 2016. CATalyst: Defeating last-level cache side channel attacks in cloud computing. In HPCA.
M. Taram, et. Al. 2019. Context-Sensitive Fencing: Securing Speculative Execution via Microcode Customization. In ASPLOS.
Microsoft. 2018. Microsoft's compiler-level Spectre fix shows how hard this problem will be to solve. In Ars Technica.

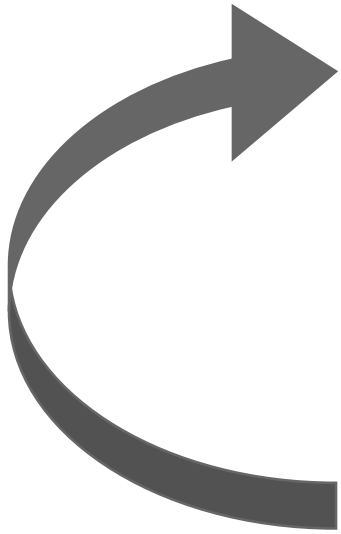
Open-source Approach to Hardware

Open-source HW + Agile Design Tools + Fast Simulation/Emulation = Security?

Large proliferation of open-source software stacks,
cores, and simulation/design infrastructure



Security benefits from open-source work



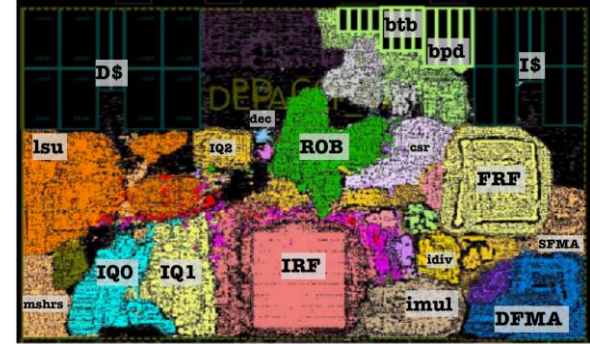
1. Think of new security mitigation/exploit
2. Use open-source RTL to start implementation
3. Quickly iterate through design development with easy, fast, and free tooling
4. Open-source work and have others scrutinize or use your work

BOOM: The Berkeley Out-of-Order Machine

BOOM Overview



- Open-source, out-of-order, superscalar RISC-V core
- Runs RISC-V ISA RV64GC
- Linux-capable - boots Fedora + Buildroot
- Silicon-proven - taped out
- ~18K LoC of open-source Chisel RTL
- Highly parameterizable and configurable
- Full integration with Rocket Chip, FireSim, HAMMER



```
19 / mosh-client (tmux)
x 19 / mosh-client (tm... 361
3526.switch0 (Detached)
2 Sockets in /var/run/screen/S-centos.
[centos@ip-172-30-2-207 ~]$ TERM=linux ssh root@172.16.0.2
root@172.16.0.2's password:
# cat /proc/cpuinfo
hart      : 0
isa       : rv64imafd
mmu       : sv39
uarch     : ucb-bar,boom0

# ping twitter.com
PING twitter.com (104.244.42.1): 56 data bytes
64 bytes from 104.244.42.1: seq=0 ttl=42 time=0.406 ms
```

CHISEL

J. Bachrach, et. al. 2012. Chisel: constructing hardware in a scala embedded language. In DAC.
K. Asanovic, et. al. 2016. The Rocket Chip Generator. Technical Report.
S. Karandikar, et. al. 2018. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In ISCA.
E. Wang, et. al. 2018. Hammer: Enabling Reusable Physical Design. In WOSET.

BOOM Microarchitecture



Fetch
(4 cycles)

Decode
and
Rename

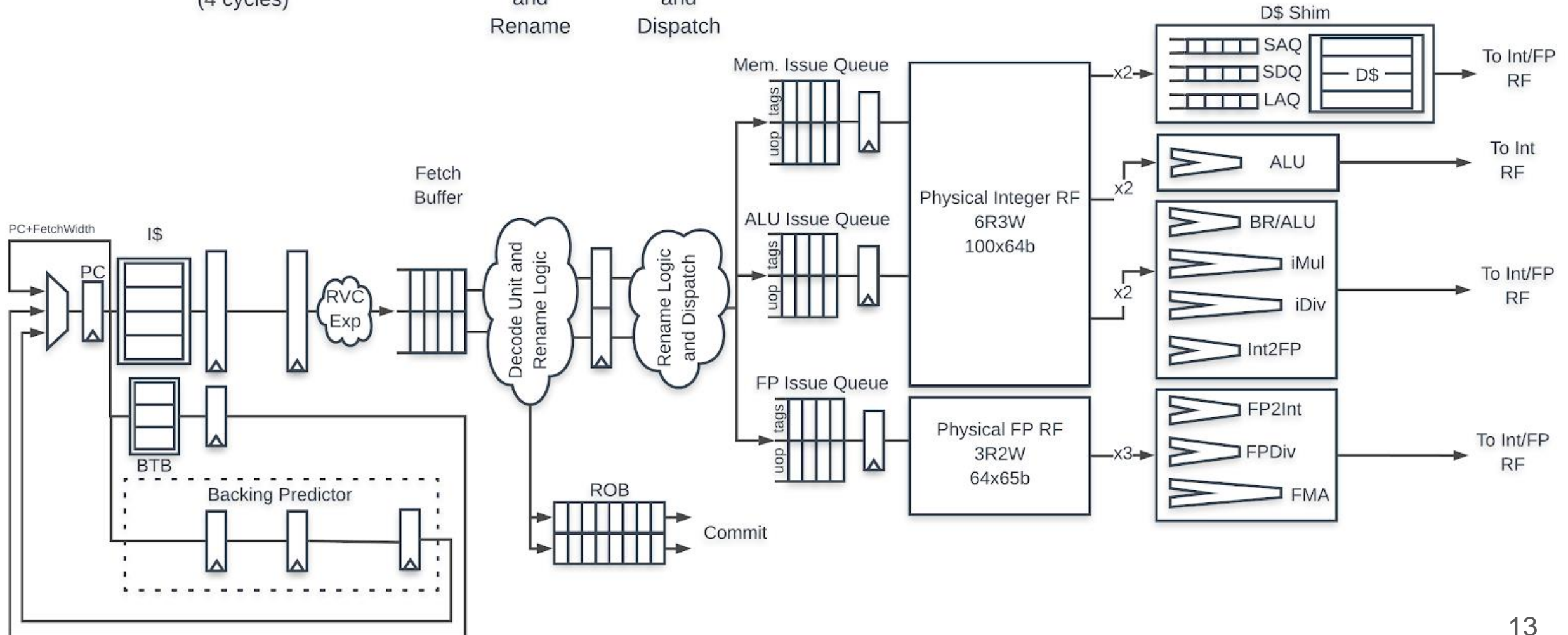
Rename
and
Dispatch

Issue

Register Read

Execute

Writeback



Replicating Spectre Attacks

Spectre v1 Overview



Speculation:

- Performance-seeking behavior of modern processors
- Execute instructions before we know they will commit

Side-channel:

- Microarchitectural state which holds interacts with program execution
- Caches, TLBs, power...

Typical Spectre attack:

1. Setup processor to misspeculate in victim code (e.g. train branch predictors)
2. Misspeculation leaks secret into a side channel
3. Attacker recovers secret from side channel

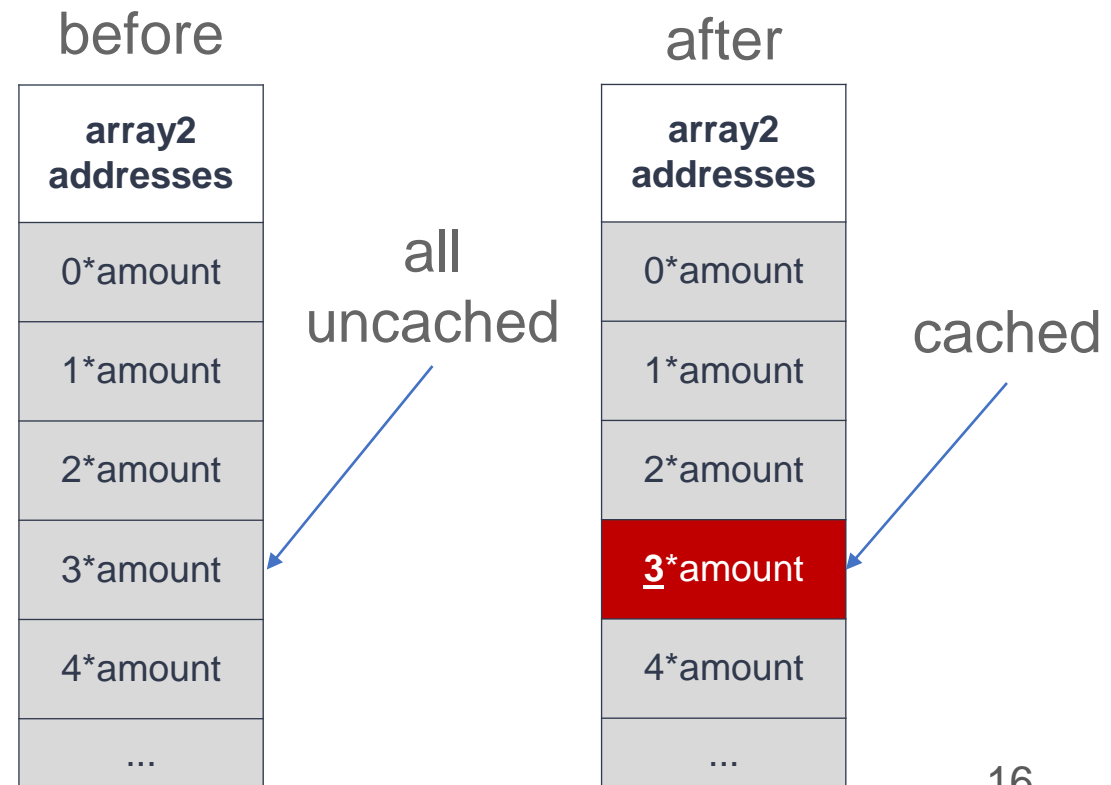
Spectre v1 Example



Steps:

1. Access *if* statement multiple times correctly (predict *if* to fall-through)
2. Give $x > \text{array1_sz}$
3. Predict the *if* to be true and bring in *secret* and *array2* value
4. Use the time difference between cached and uncached lines to determine *secret*
5. Repeat!

```
if (x < array1_sz):  
    secret = array1[x]  
    out = array2[secret * amount]
```



Components Needed – With BOOM?



- Branch Prediction
 - Set associative BTB and GShare branch predictors
- Speculative Execution
 - Out-of-order execution and branch kill masks for speculative execution
- Caching
 - L1 data cache and a outer memory set to the latency of an L2 cache
- Cache Manipulation
 - Custom-made L1 data cache *clflush*

BOOM provides all the elements to replicate Spectre!

Spectre v1 Running on FireSim



The screenshot shows the FireSim website. At the top is a navigation bar with the FireSim logo (a flame icon) and links for About, Publications, Docs, Blog, GitHub, Twitter, and Mailing List. The main content area has a dark background with a pattern of hexagons. It features the text "Easy-to-use, FPGA-accelerated Cycle-accurate Hardware Simulation of RISC-V Systems in the Cloud" and a "Get Started!" button. Below this, there is a section titled "What is FireSim?" which describes FireSim as an open-source cycle-accurate, FPGA-accelerated scale-out computer system simulation platform developed in the Berkeley Architecture Research Group in the Electrical Engineering and Computer Sciences Department at the University of California, Berkeley. It also mentions that FireSim is capable of cycle-exactly simulating from one to thousands of multi-core compute nodes, derived directly from

Implementing a Speculation Buffer

Protecting Data Caches

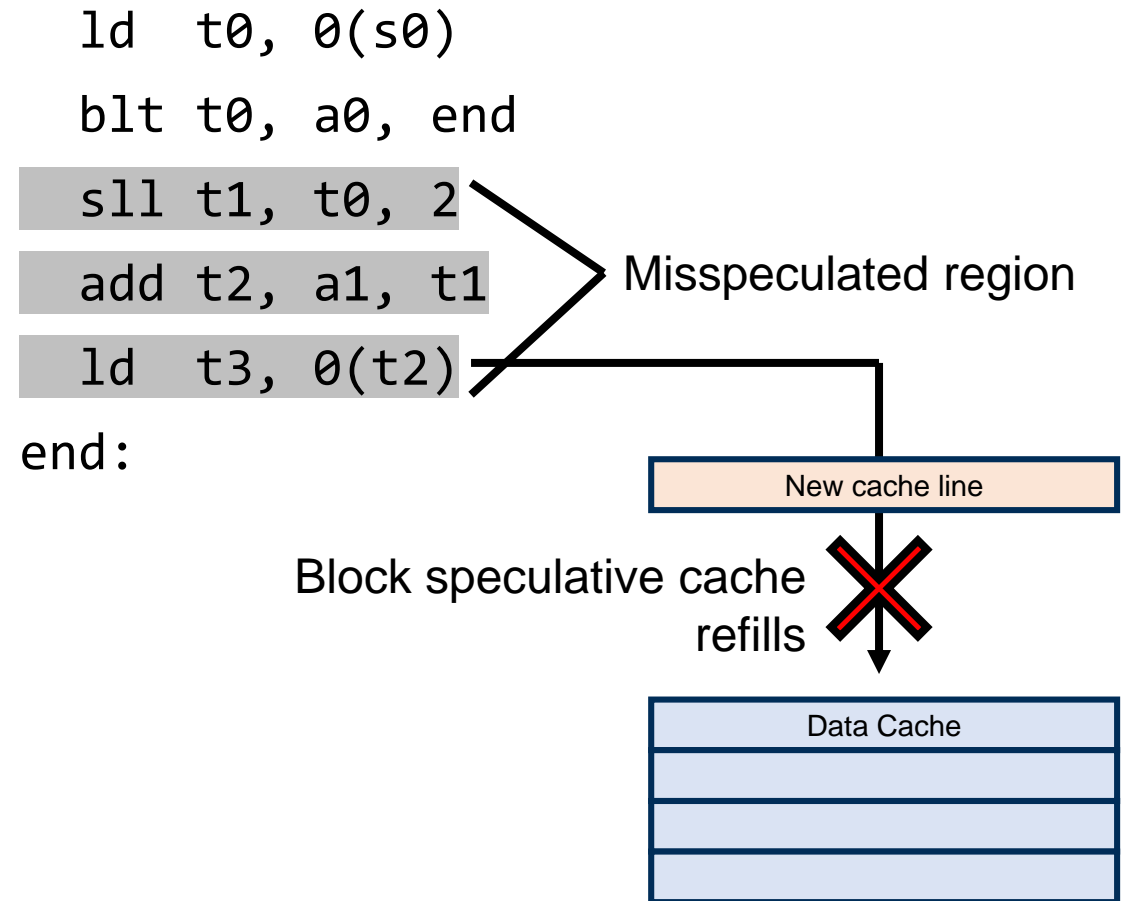


Problem: Load refills are not subject to architectural guarantees

- Misspeculated loads leave **side-effects**, creating a side-channel

Solution: Treat the data cache as an architectural structure

- Only alter the cache state when instructions **commit**
- Implement a working prototype in BOOM RTL



Prior Work



InvisiSpec

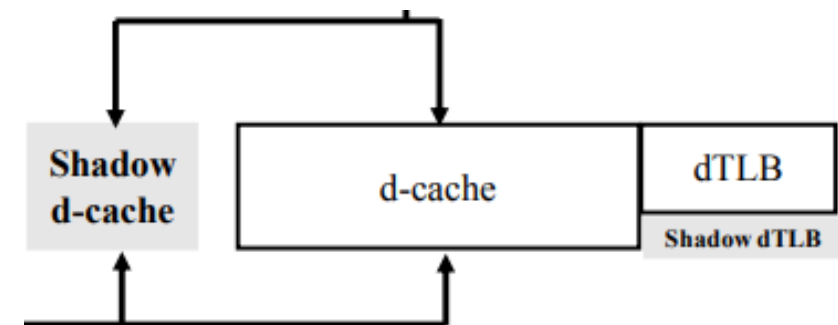
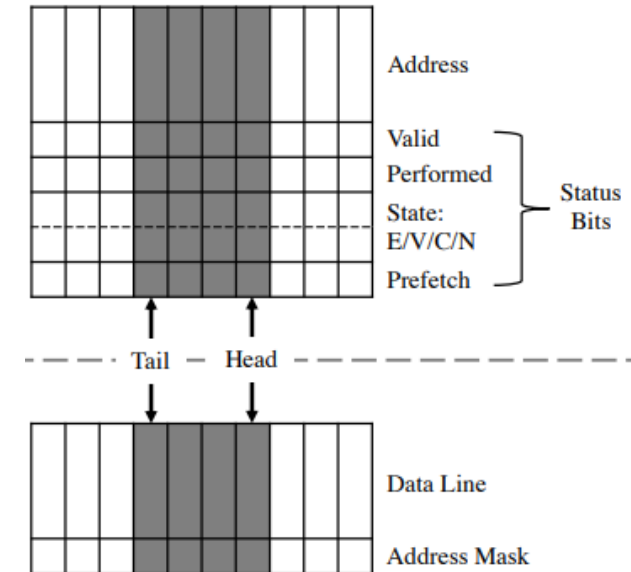
- Per load-queue-entry speculation buffer
- Speculation-aware cache-coherence policy

Safespec

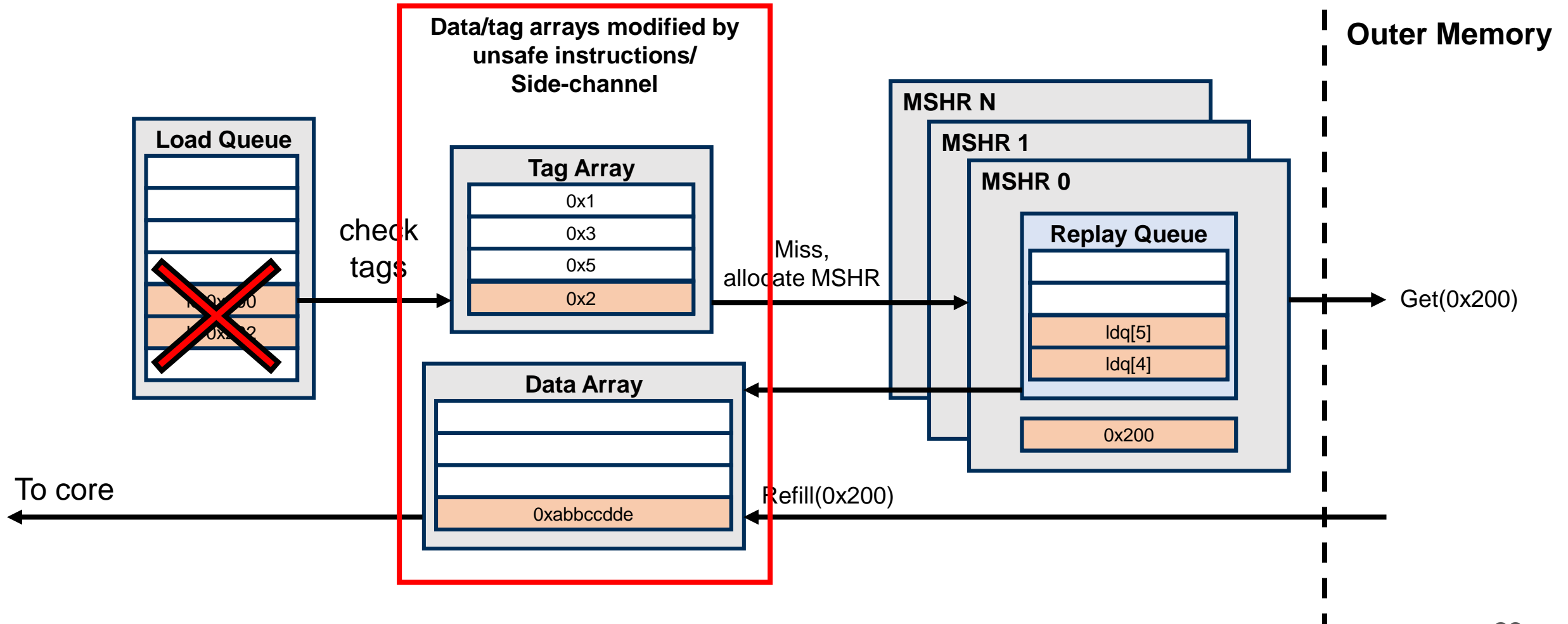
- Speculation-depth sized “shadow structures”
- Protect DCache, ICache, TLBs

BOOM Speculation Buffer:

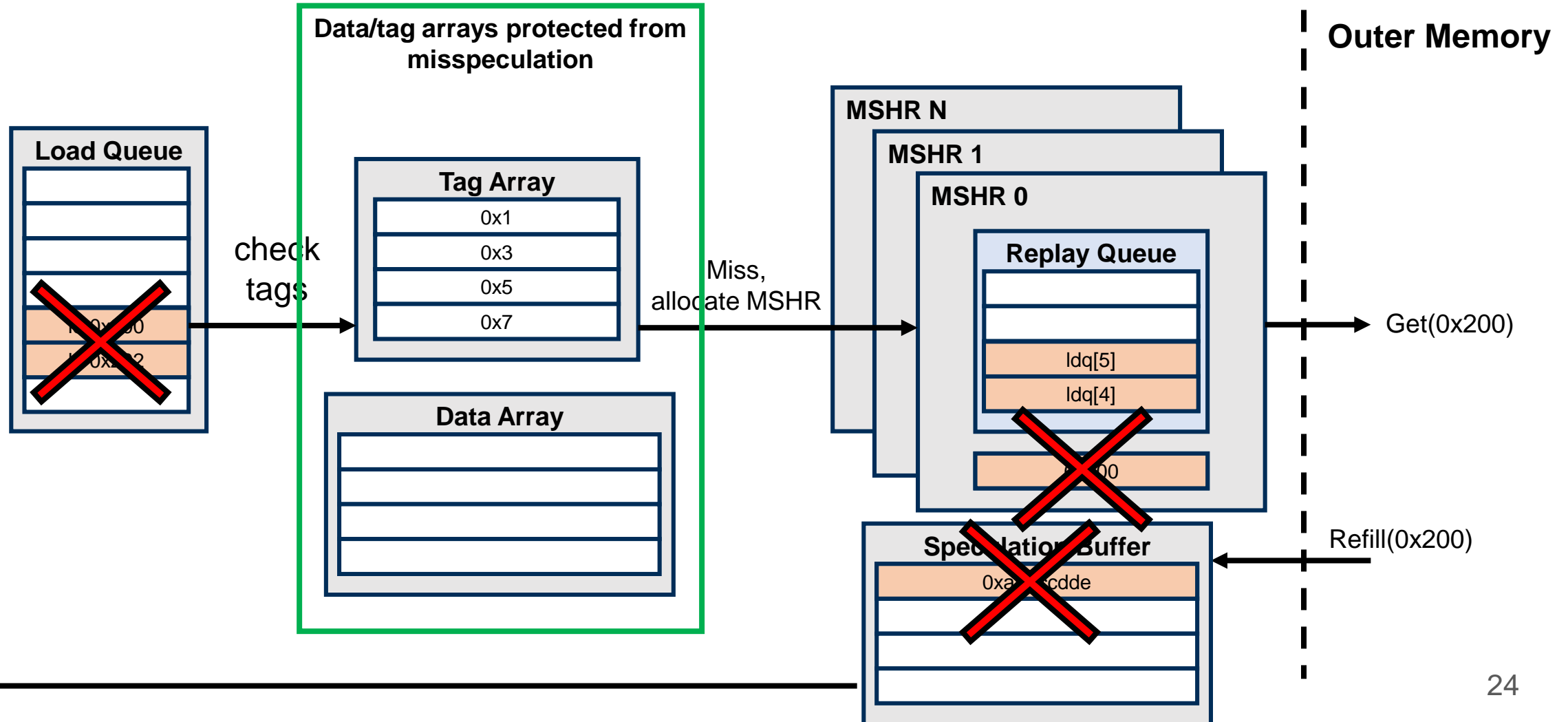
- Hold speculated loads in **line-fill-buffers**



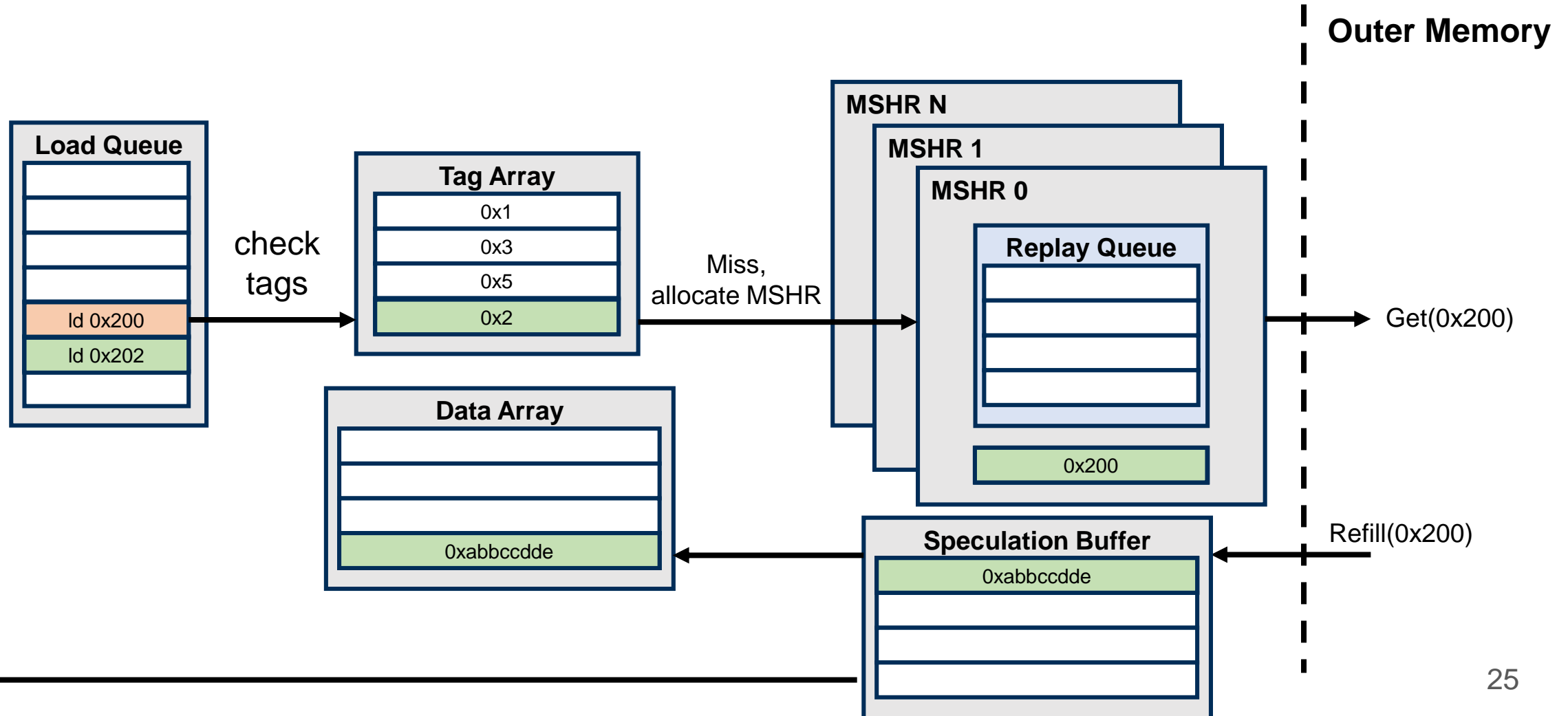
Life of a Misspeculated Load



Blocking Misspeculated Loads



Blocking Misspeculated Loads



Blocking Misspeculated Loads



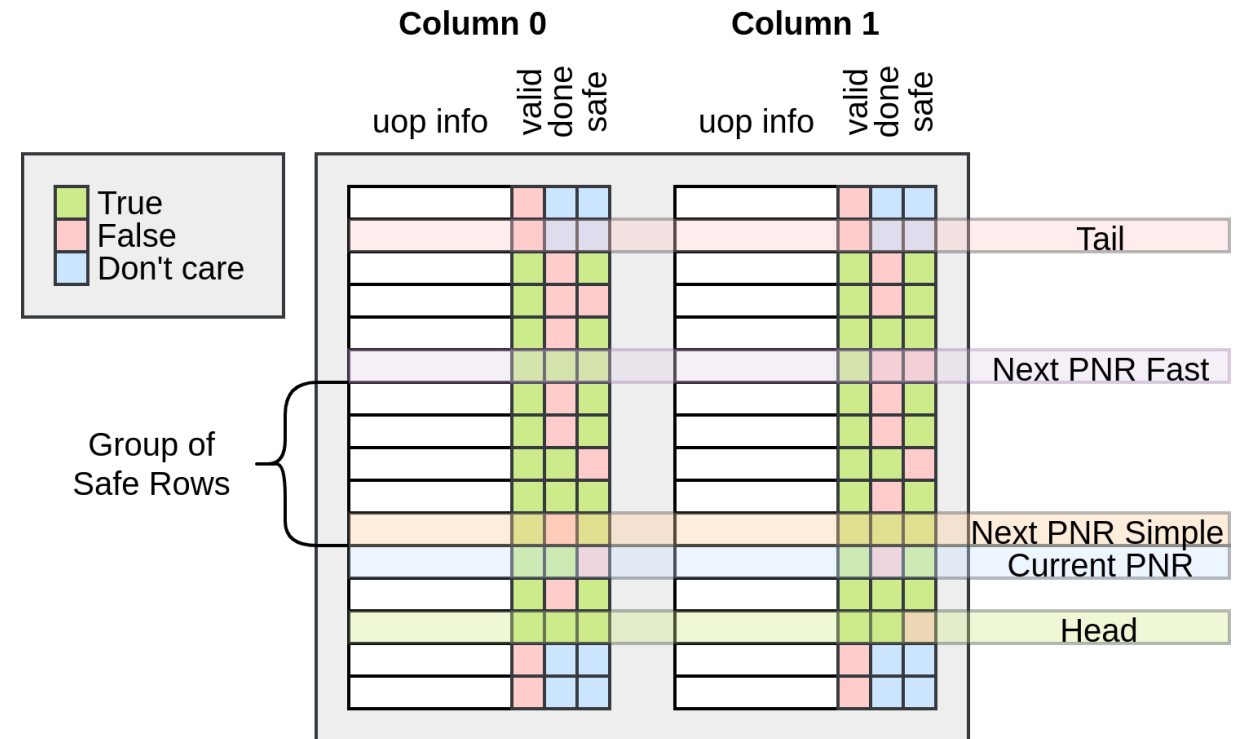
- Load refills wait in the buffer until one of their misses has committed
- Stall writeback until one of the following occurs
 - A load-miss to that line has committed OR
 - A store-miss hits that line (stores are non-speculative)
- If all load misses to that line were misspeculated, discard it
- Bypass loads out of the load-fill-buffer
 - Subsequent loads “see” the data in the DCache
 - Minimizes performance penalty

Committing Loads



When to commit load refills to the DCache?

- When the ROB commits the load?
 - Most secure.
 - Huge performance penalty for load misses
- When the load is free from branches?
 - Does not consider exceptions/interrupts
 - Minimal performance penalty
- When the load reaches the **point-of-no-return**
 - New ROB pointer, tracks instructions which are guaranteed to commit



Speculation Buffer Results



1 month implementation time

Microbenchmarks

- Set of assembly routines to test edge cases

Dhrystone results

- Original: 2176 dps
- W. Speculation buffer: 2216 dps
- Impact: ~2% better IPC

Preliminary physical results in TSMC 45nm

- ~3% larger area

<u>Benchmark</u>	<u>Version of BOOM</u>		
	<u>Normal</u>	<u>With Speculation Buffer</u>	<u>% Difference</u>
Non-speculative LD misses to same sets	540 cycles	640 cycles	-19%
Non-speculative LD misses to different sets	264 cycles	297 cycles	-11%
MSHR evicted speculative LD misses	48 cycles	67 cycles	-40%
Dhrystone	2176 dps	2216 dps	+2%

Comparison



	InvisiSpec	SafeSpec	BOOM Speculation Buffer
Implementation Platform	Custom GEM5	Marssx86	BOOM RTL
Buffer size	Additional cacheline * load-queue-size	Additional cacheline * speculation depth	Repurposed line-fill- buffers
Commit condition	Wait for branch OR Wait for non-speculative	Wait for branch OR Wait for commit	Wait for point-of-no-return
Physical design feedback	CACTI estimates	CACTI estimates	Trial TSMC 45nm implementation
Protected components	L1D, LLC, multicores	L1D, L1I, TLBs	L1D
Performance impact	-22% performance	+3% performance	+2% performance

Conclusion

Conclusion



Demonstrated application of RISC-V ecosystem towards secure hardware

- Working demonstrations of Spectre attacks on a RISC-V core
- RTL of Spectre mitigation available in an open-source core

Continue improving BOOM security

- Secure other structures: TLBs, ICache, LLC, branch predictors
- Enable secure enclave execution

BOOMv3 Tapeout + More Attacks

- Planning to add Speculation Buffer and CSRs to enable/disable it
- More attacks with different predictors/structures (TAGE, RAS, etc)

Questions?

Thanks CARRV19!

Contact: {abe.gonzalez,bkorpan,jzh,edyounis,krste}@berkeley.edu

Links:

- *Core*: boom-core.org
- *Github*: github.com/riscv-boom
- *FireSim*: fires.im
- *HAMMER*: github.com/ucb-bar/hammer

Thanks:

- Chris Celio, David Kohlbrenner
- UCB ADEPT Lab