WiP: Isolating Speculative Data from Microarchitectural Covert Channels

Kristin Barber, Anys Bacha*, Li Zhou, Yinqian Zhang, Radu Teodorescu

Department of Computer Science and Engineering

The Ohio State University

http://arch.cse.ohio-state.edu

*University of Michigan

Motivation

- Speculative execution is pervasive in OOO processors
- Fundamental to performance

conditional branches
exceptions
speculative store bypass
value speculation

- Transient execution leave traces in the micro architectural state
- Covert channel used to leak information to attacker
- We now know it is vulnerable to a wide range of attacks

Spectre-v1 [12]
Spectre-v1.1 [11]
Spectre-v1.2 [11]
Spectre-v2 [12]
Spectre-v3 (Meltdown) [14]
Spectre-v3a [2]
Spectre-v4 [7]
LazyFP/Restore [20]
ret2spec [15]
Foreshadow [4]
NetSpectre [18]
SMoTherSpectre [3]

- We have to re-invent speculation with security in mind and little performance impact

Outline

- Threat Model
- Design
- Evaluation
- Conclusion

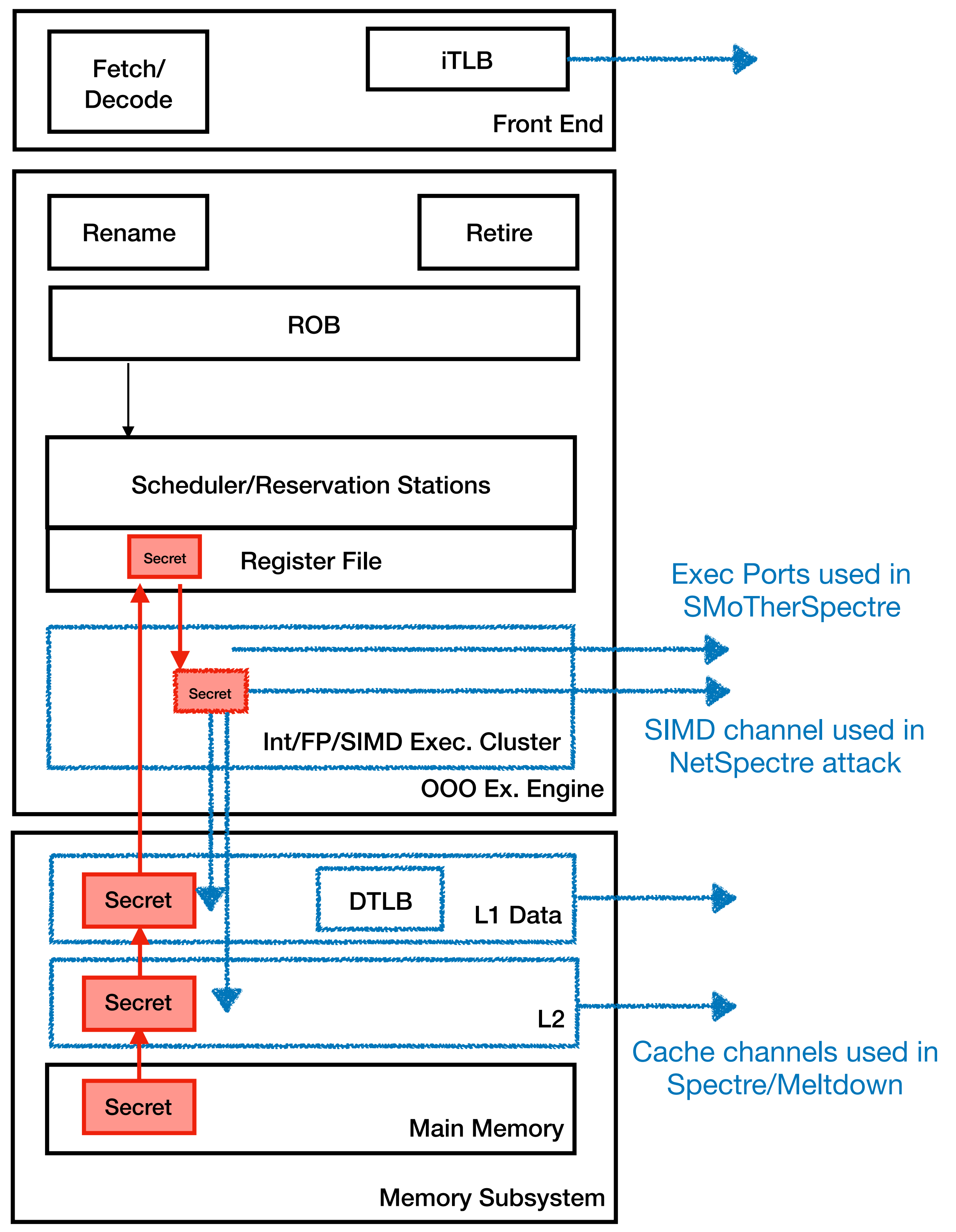
Outline

- Threat Model
- Design
- Evaluation
- Conclusion

Threat Model

- Attack vectors:
  - — Illegal memory access (Meltdown)
  - — Illegal control flow (Spectre)
- Sensitive data resides anywhere in the memory hierarchy
  - — Accessed through a transient/mis-speculated instruction
- Any covert channel can be used to exfiltrate secret data:
  - — Caches, ALUs, SIMD units, TLBs, etc.
- Out-of-scope: leakage of data retrieved through non-transient instructions

OOO Processor



Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

Existing Solutions

- Software-only mitigation solutions

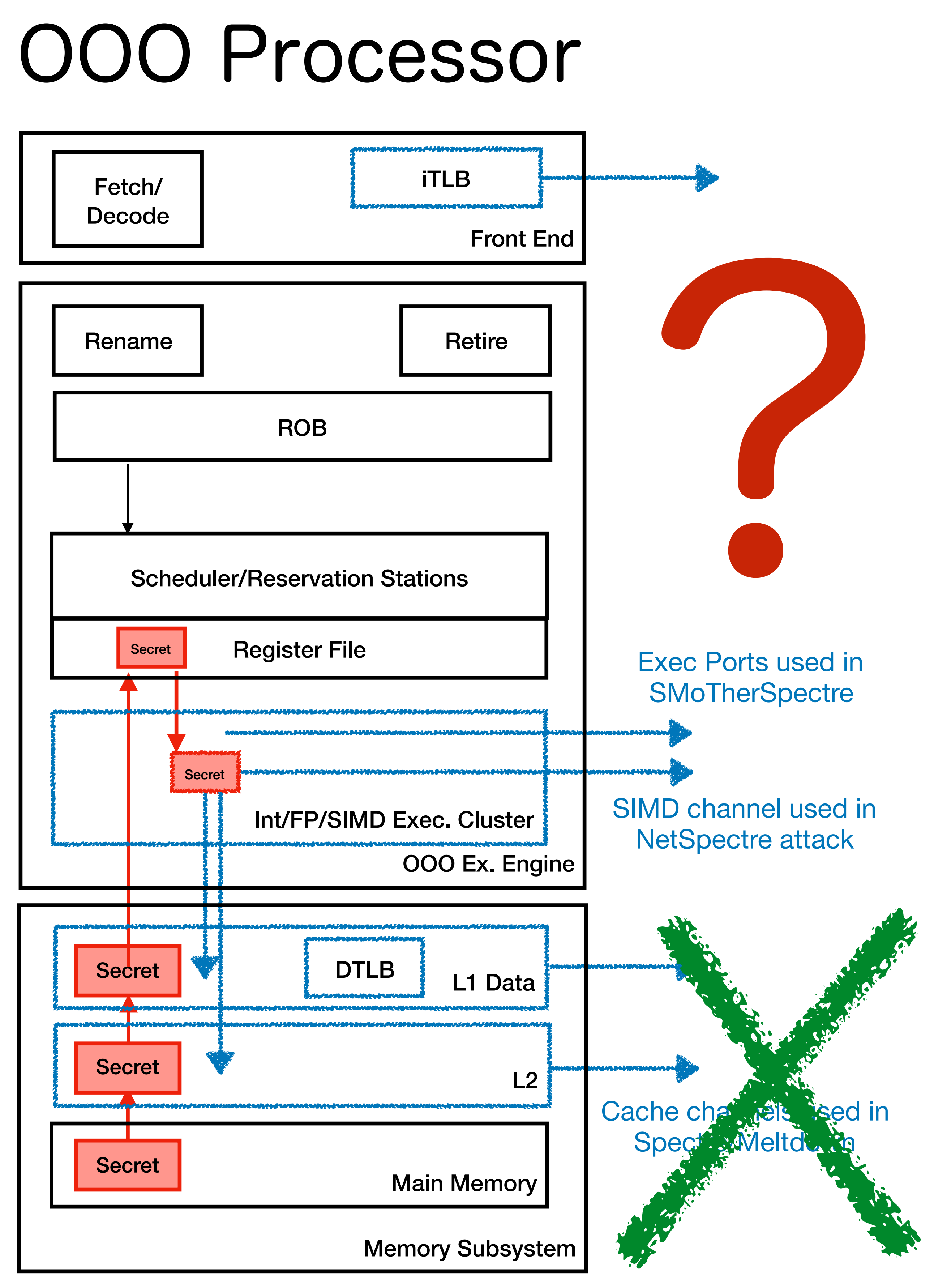
  — Generally very high overhead for good coverage

  — Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases.

- Existing hardware solutions

  — Lower-overhead, better coverage

  — Mostly focused on closing specific covert channels


OOO Processor
Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown
?

Outline

- Threat Model
- Design
- Evaluation
- Conclusion

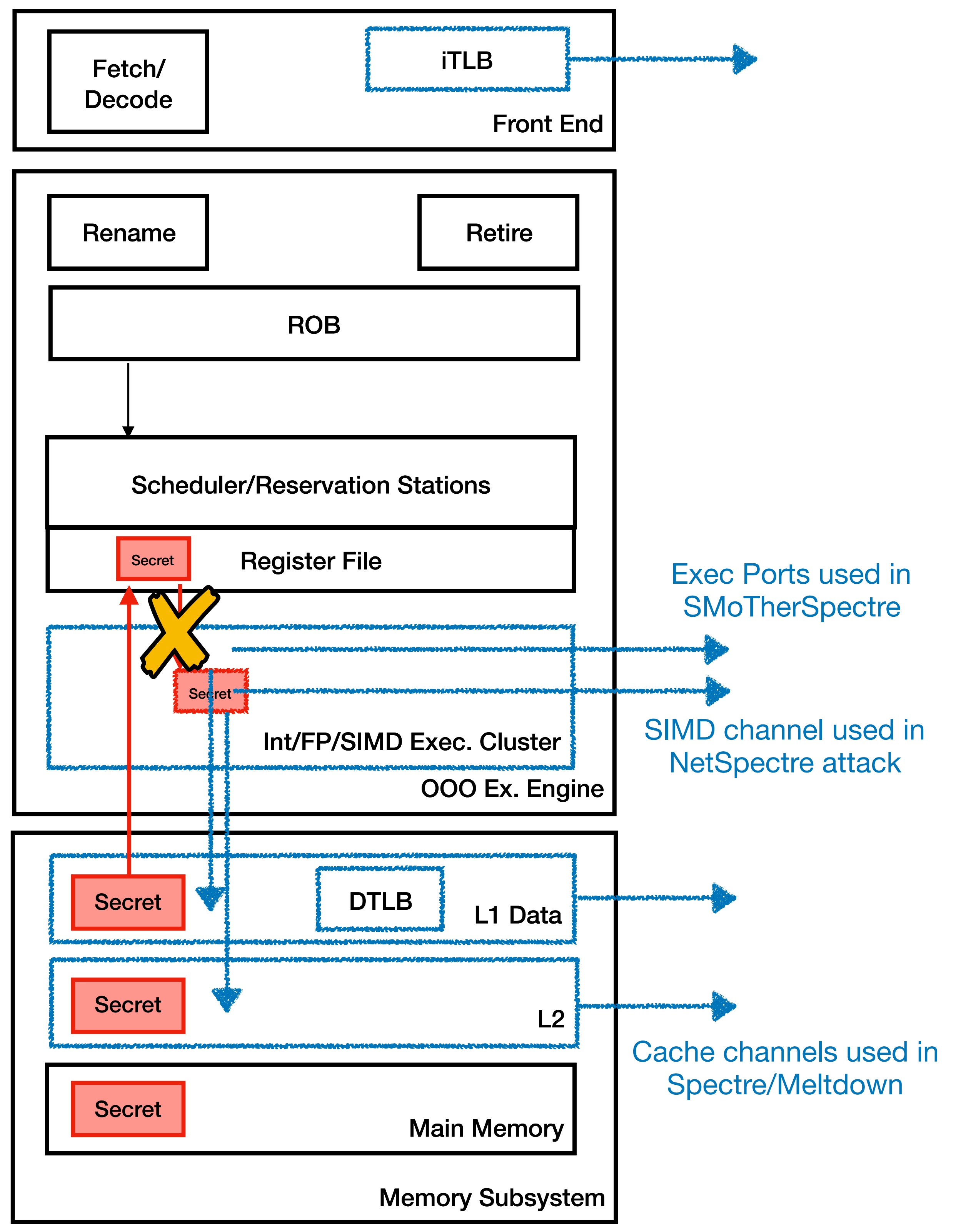
Defense Strategy: Main Idea

A more general solution that prevents covert channel formation

Key Observation:

- All covert channels have dependences on secret data
- Restrict speculative data use by dependent instructions

OOO Processor

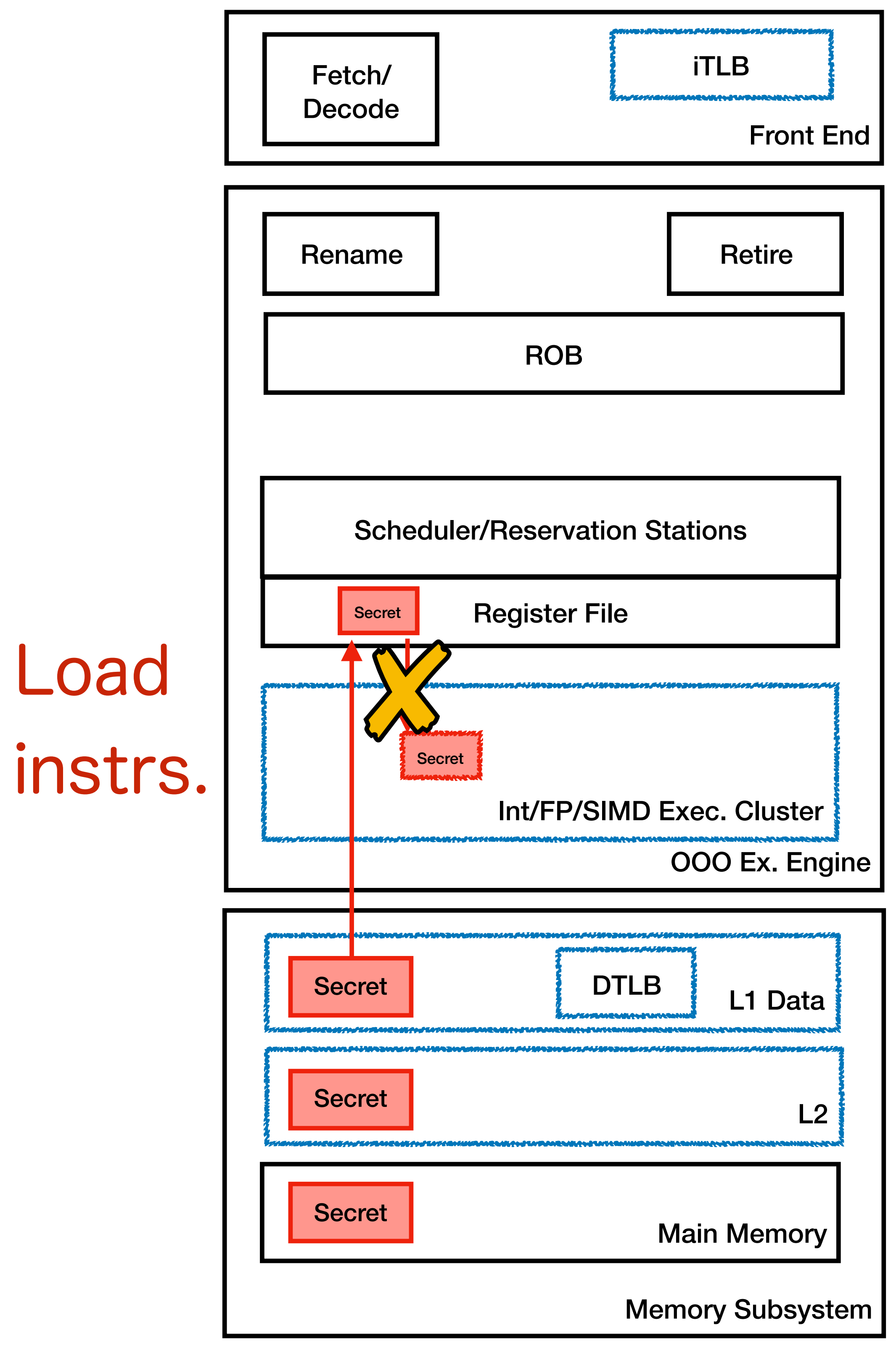


Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

Defense Strategy: Main Idea

- Preventing covert channel formation:
  - — Monitor speculative status of Load instructions
  - — Forward data to dependents only when "safe"
- What we consider "safe" is implementation-dependent
  - — Two schemes, different performance/security tradeoffs

OOO Processor



Outline

- Threat Model
- Design
  - Conservative Scheme
  - Optimized Scheme
- Evaluation
- Conclusion

COMPUTER ARCHITECTURE RESEARCH LAB

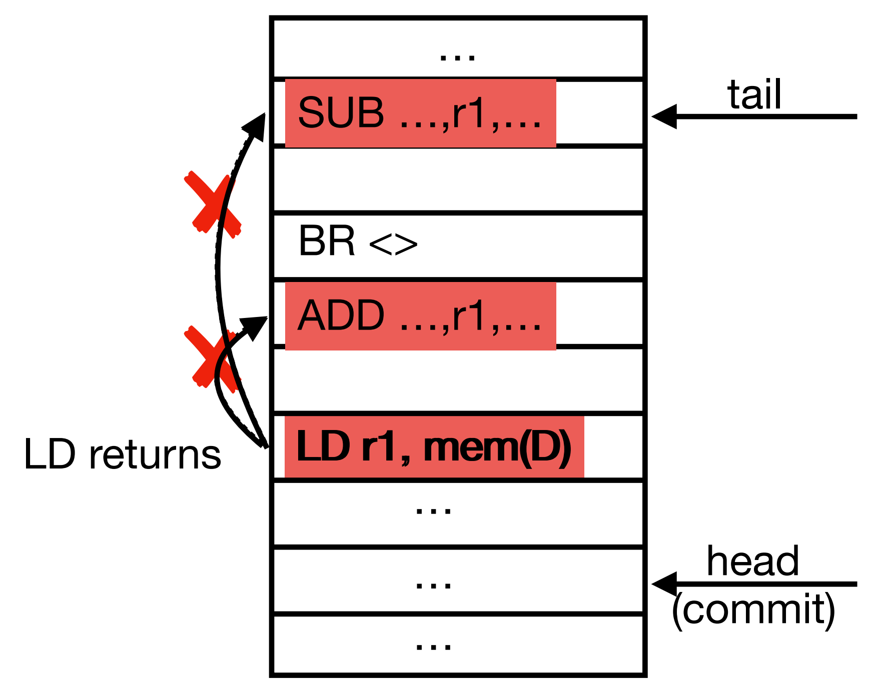
Outline

- Threat Model
- Design
  - Conservative Scheme
  - Optimized Scheme
- Evaluation
- Conclusion

Conservative Scheme

- Speculative Loads are issued and executed normally
- When data returns from memory (cache)
  - — Register file is updated
  - — Delay forwarding data to dependent instructions
- When Load commits, forward to dependents
- All data guaranteed to be non-speculative before use
- Downside: relatively large performance impact

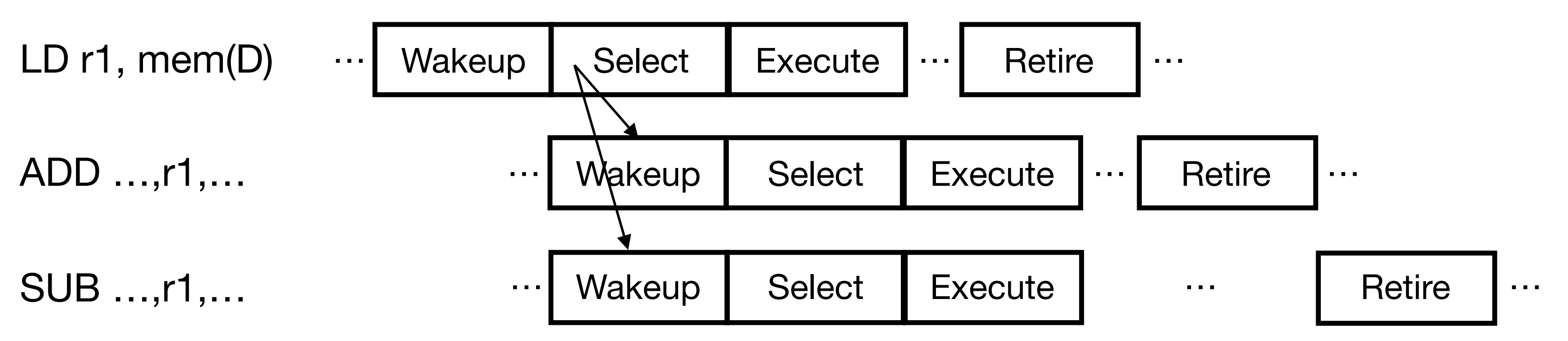
Reorder Buffer



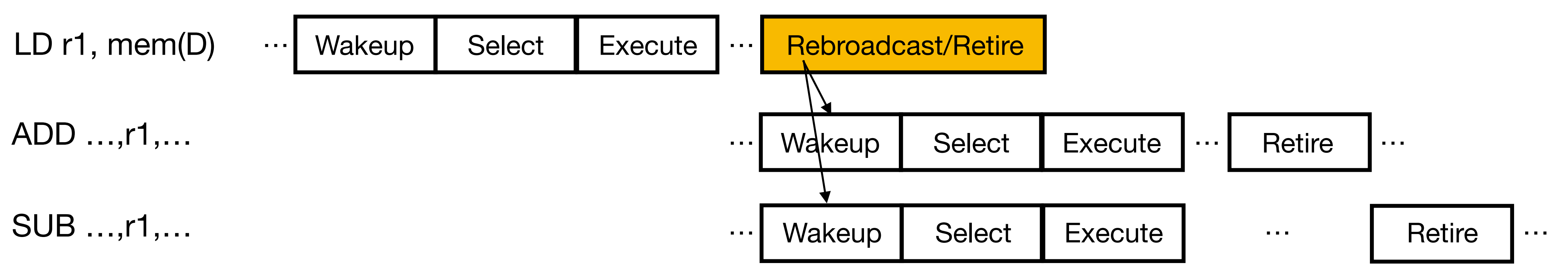
Conservative Scheme

- Impact on Wakeup/Select Logic
- Baseline: LD-dependents speculatively woken up on Select
  - — Assuming that LD will be hit
- Wakeup delayed until retirement

Baseline Wakeup/Select/Execute/Retire Pipeline



Wakeup/Select/Execute/Retire Pipeline



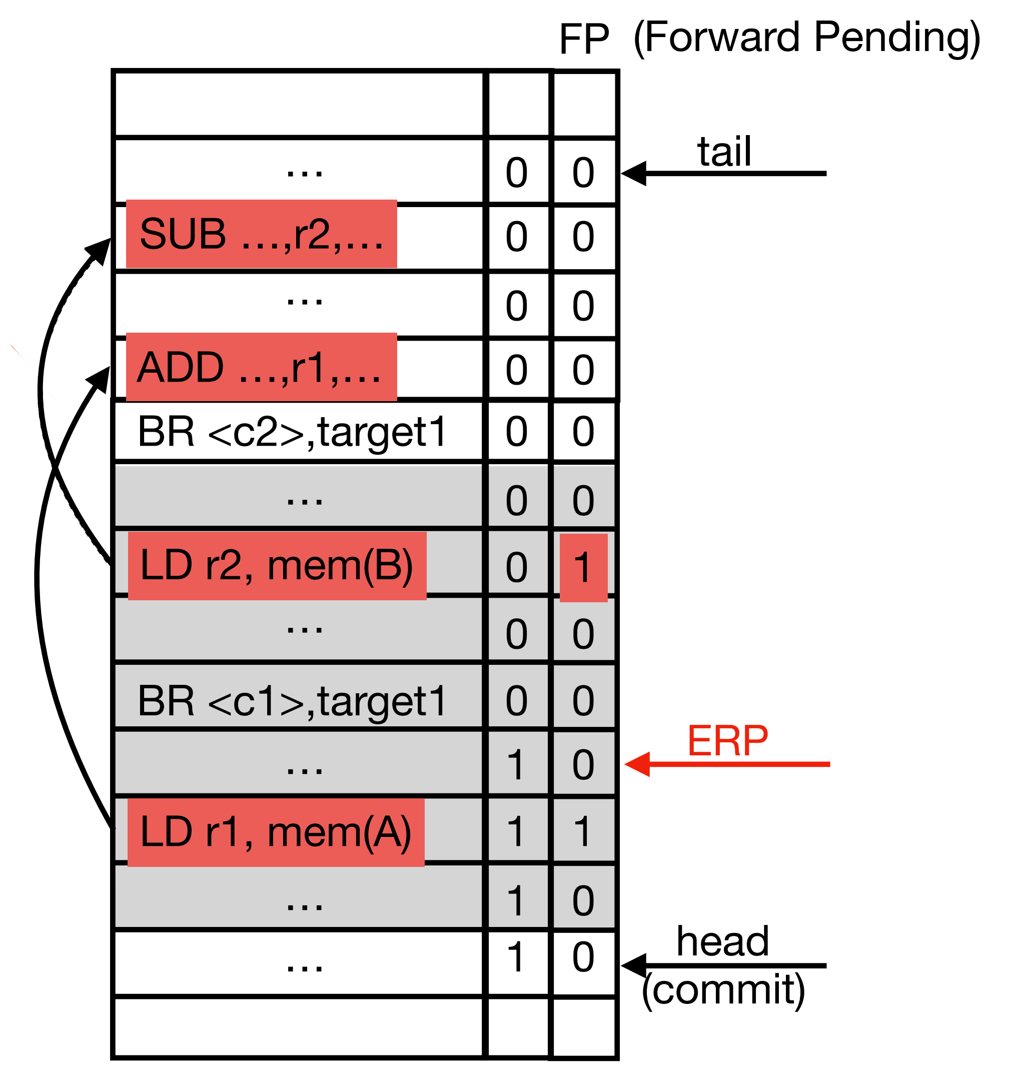
Outline

- Threat Model
- Design
  - Conservative Scheme
  - Optimized Scheme
- Evaluation
- Conclusion

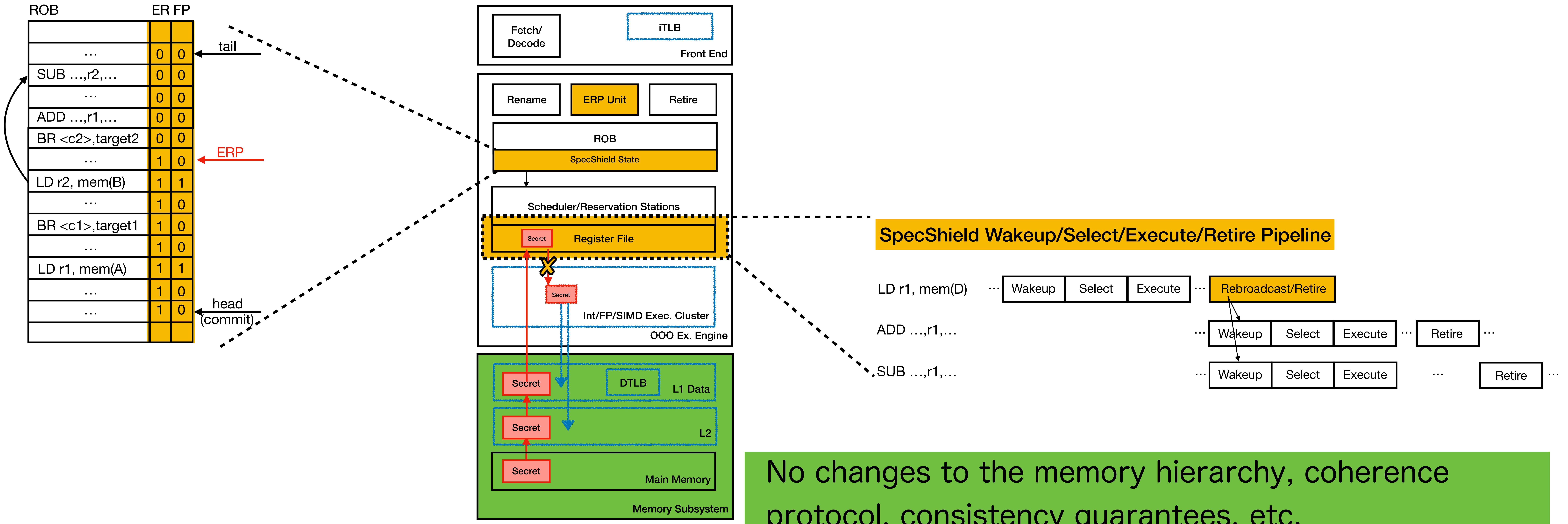
Optimized Scheme

- Observation: Most Loads are safe earlier than retirement
- Define Early Resolution Point (ERP)
    - — All older branches have resolved
    - — All older loads and stores have had addresses computed
    - — No branch mispredictions or memory-access exceptions
- Forwarding of speculative data allowed after ERP
- Much lower performance impact, equivalent security

Reorder Buffer



SpecShield Hardware Support



No changes to the memory hierarchy, coherence protocol, consistency guarantees, etc.

🟨 SpecShield Changes/Additions

Outline

- Threat Model
- Design
- Evaluation
- Conclusion

COMPUTER ARCHITECTURE RESEARCH LAB

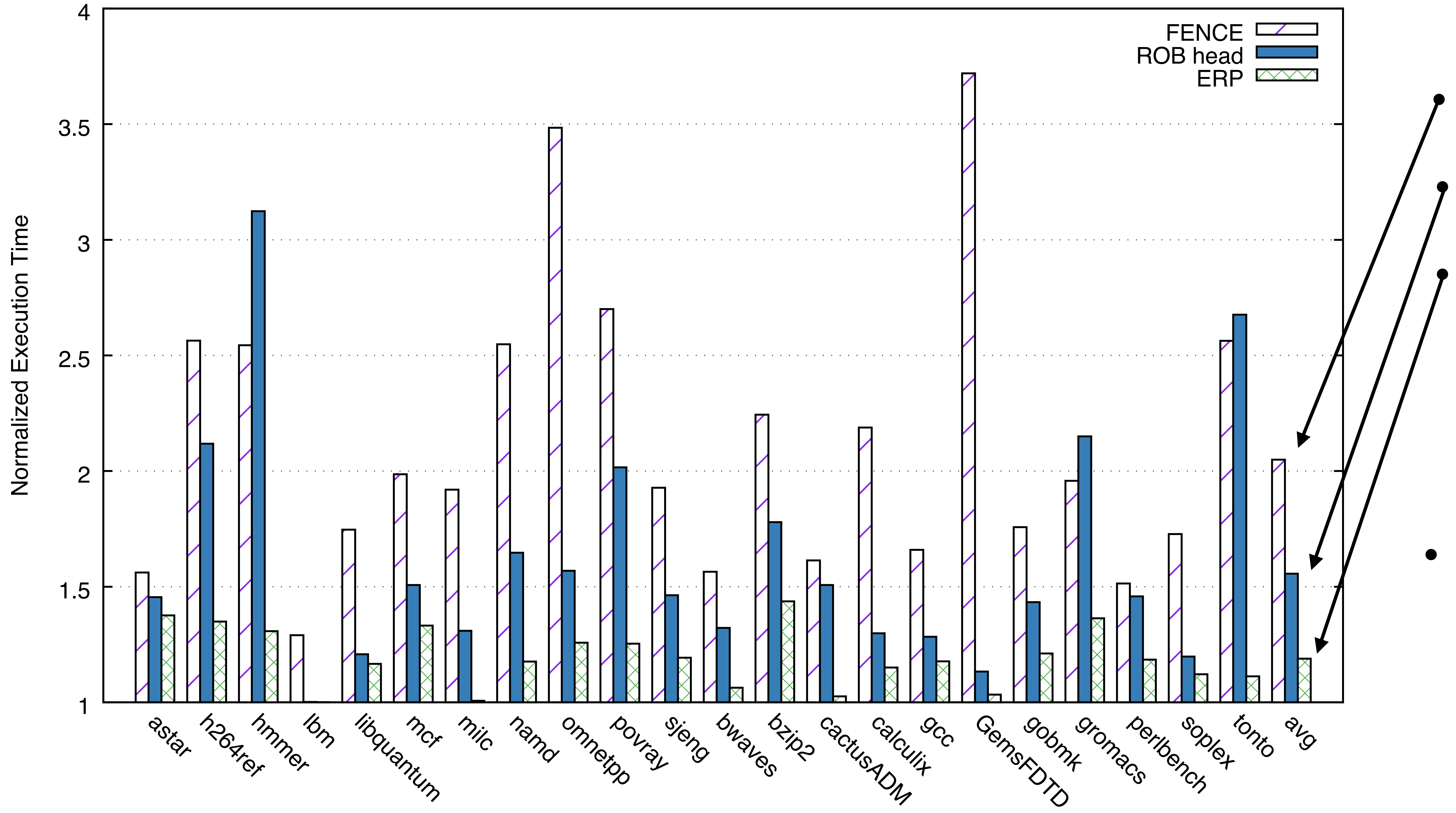
Evaluation Methodology

Experimental Platform

- Simulator: gem5, full-system mode, Ubuntu 14.04 OS
- Benchmarks: spec2006, reference input set

CPU Architecture			
CPU Clock	2GHz	LSQ Entries	32
L1 ICache	32KB (4-way)	IQ Entries	64
L1 DCache	32KB (8-way)	BTB Entries	4096
L2 Cache	2MB (16-way)	dTLB Entries	64
Issue Width	8	iTLB Entries	64
ROB Entries	192	FP Registers	256
Branch Predictor	LTAGE	Int Registers	256

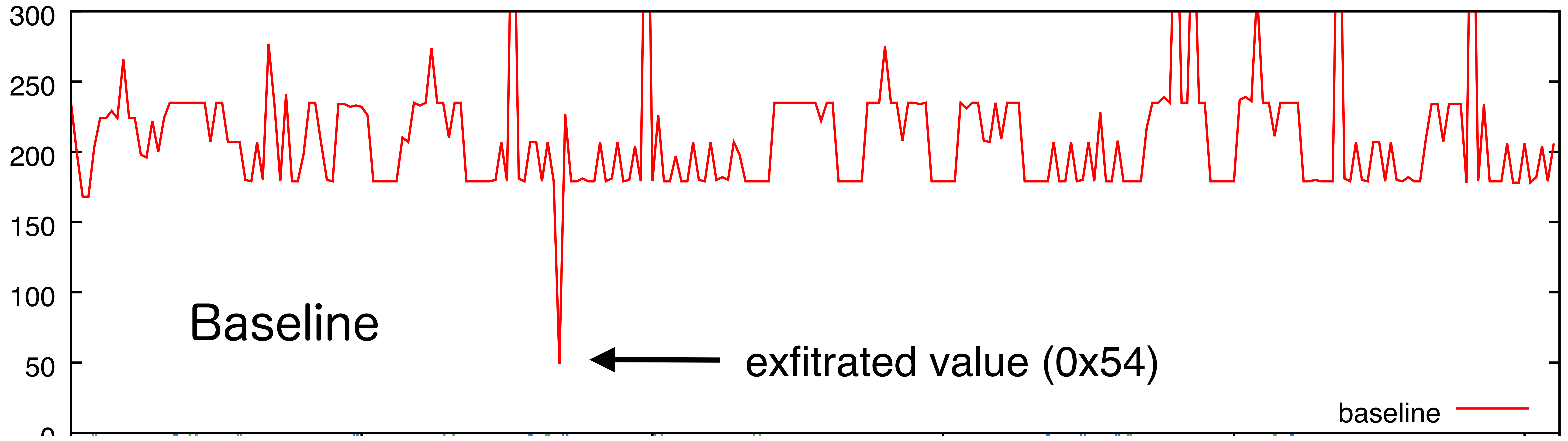
Performance



- LFENCE > 200%
- SpecShieldSTL 55%
- SpecShieldERP 18%

- Benchmarks with low miss rates most impacted

Cache Latency for Spectre Attack



- Spectre-attack, using cache as covert channel
- Exfiltrated value visible in access latency

- Secret value no longer appears in the cache channel

Conclusions

- Microarchitectural framework for preventing transient execution attacks on arbitrary memory
- SpecShield is more general
  - — Unlike prior work that has focused on closing specific covert channels, SpecShield controls all speculative data-flow within the pipeline, preventing channel formation.
- SpecShield is easier to implement
  - — No changes to the memory hierarchy, coherence protocol, consistency guarantees, etc.
- Possibility for flexibility and security policies