HidFix: Efficient Mitigation of Cache-based Spectre Attacks via Hidden Rollbacks

Arash Pashrashid, Ali Hajiabadi and Trevor E. Carlson

National University of Singapore

42nd IEEE/ACM International Conference on Computer-Aided Design (ICCAD '23)

Bottom Line Up Front

- **Problem**: mitigating speculative execution attacks in modern CPUs

- Prior work:
  - High performance overhead: 30% to 200% overhead
  - Non-secure: introducing opportunities for new attacks

- Our work, HidFix:
  - Almost zero performance overhead
  - Same/stronger security guarantees

Fundamental Security Problem of Modern CPUs

- Decades-long focus of computer architects: CPU performance

- Aggressive CPU optimizations have resulted in fatal security vulnerabilities affecting almost all modern processors

SPECTRE MELTDOWN MDS Hertzbleed DOWNFALL Platypus COLLIDE+POWER

Cache-based Spectre Attacks

- Focus of this work: SPECTRE targeting speculative execution
- Example: Spectre via Prime+Probe

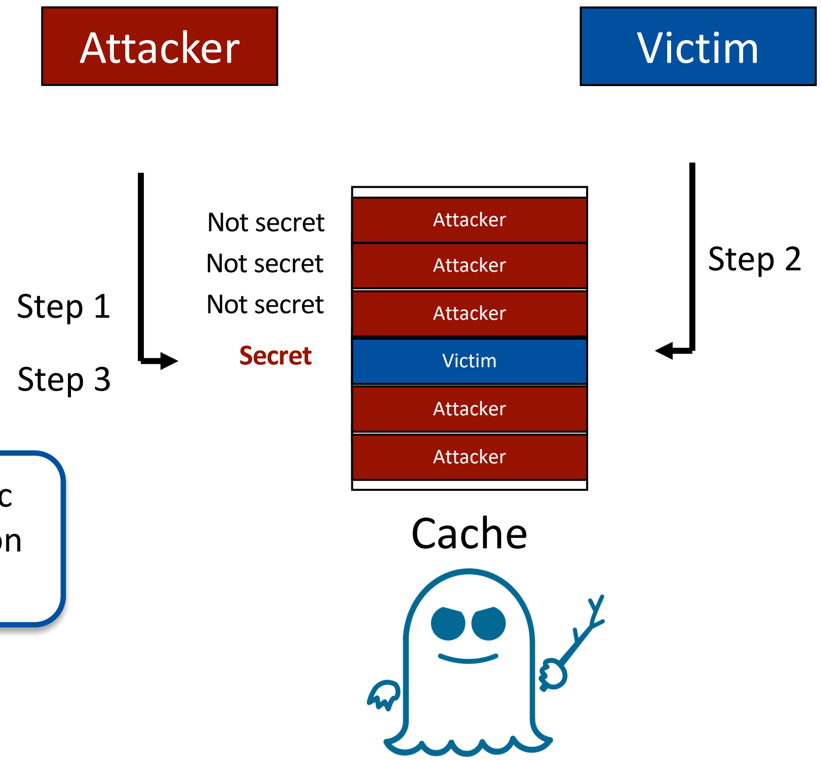
```
prime_cache_state()
```
① Initializes the system for attack

```
if (index < size_A)
```
② Misspeculation

```
    secret = load A[index]
```
Load Secret — Speculative Access

```
    SideChannel[secret]
```
Secret-dependent trace in the cache — e.g., access specific cache lines based on the secret value

```
probe_cache_state()
```
③ Extracting the Secret — e.g., load latency



How to Mitigate Spectre?

Restriction-based

Isolation

Access Obfuscation

High Performance Overhead

Invisible Speculation

Undo Speculation

High Performance Overhead

Vulnerable to new SCAs

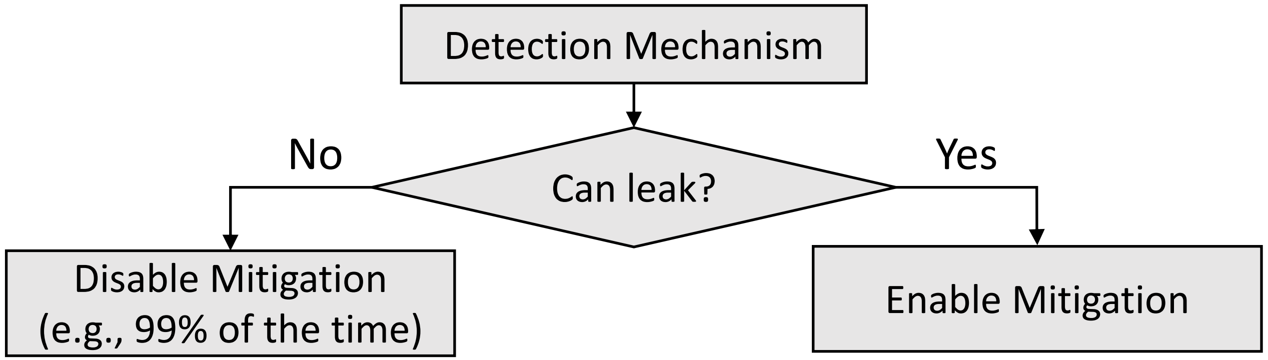
Cache Randomization

Medium Performance Overhead

Limited Protection

These solutions are always on and incur unnecessary performance overheads, even when protection is not required

Detection + Mitigation Approach



Existing detectors

- ML-based
- Cyclic-based
- μarch-based

\+

Existing Mitigations

(capable of selective enable)

Restriction-based	Invisible Speculation
Undo Speculation	Cache Randomization
Isolation	Access Obfuscation

Are the Current Possible Combinations of Detection + Mitigations Reliable?

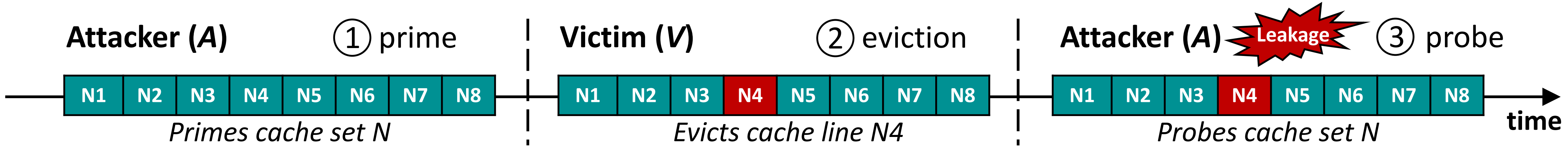
1. Cyclone + Isolation / Access Obfuscation — Bypassed → **BenignInterfere Attack**
2. Ideal Cyclic-based + Isolation / Access Obfuscation — Bypassed → **SingleProbe Attack**
3. Ideal Detection + Isolation / Access Obfuscation — Bypassed → **SinglePrime Attack**

1. Spectify reference

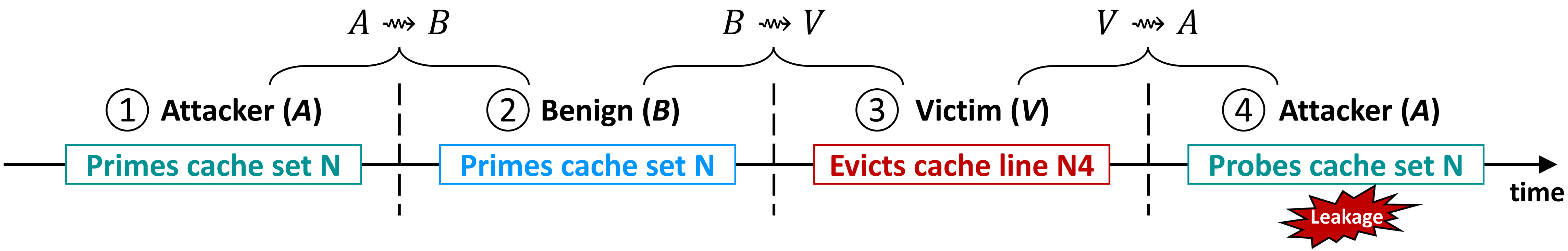
Attack 1: Bypassing Cyclone Detection

1. BenignInterfere | 2. SingleProbe | 3. SinglePrime

Cyclone[1]: detects $A \rightsquigarrow V \rightsquigarrow A$ as an attack **if no third party interfere**



BenignInterfere attack: bypassing the Cyclone by using third party interfere: $A \rightsquigarrow B \rightsquigarrow V \rightsquigarrow A$



Goal: Detection/Mitigation Co-design

- Blindly combining detection and mitigation is not effective and robust

- Our goal is to **co-design detection and mitigation** to achieve a solution that:

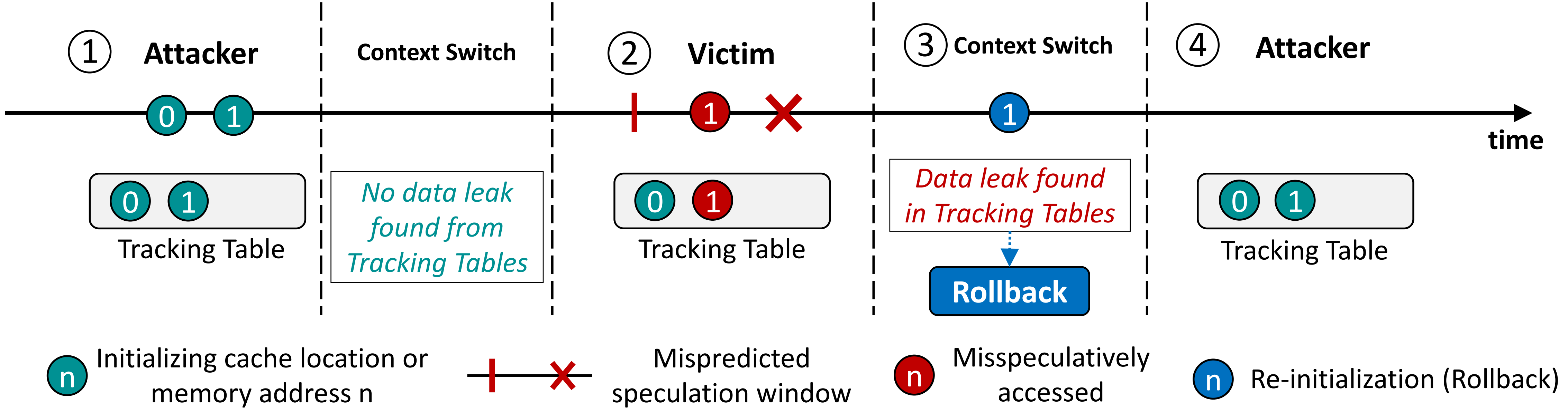
1. Accurately spots the speculatively leaked data through the cache

2. Reverts the data leaks before a potential attacker has a chance for extraction

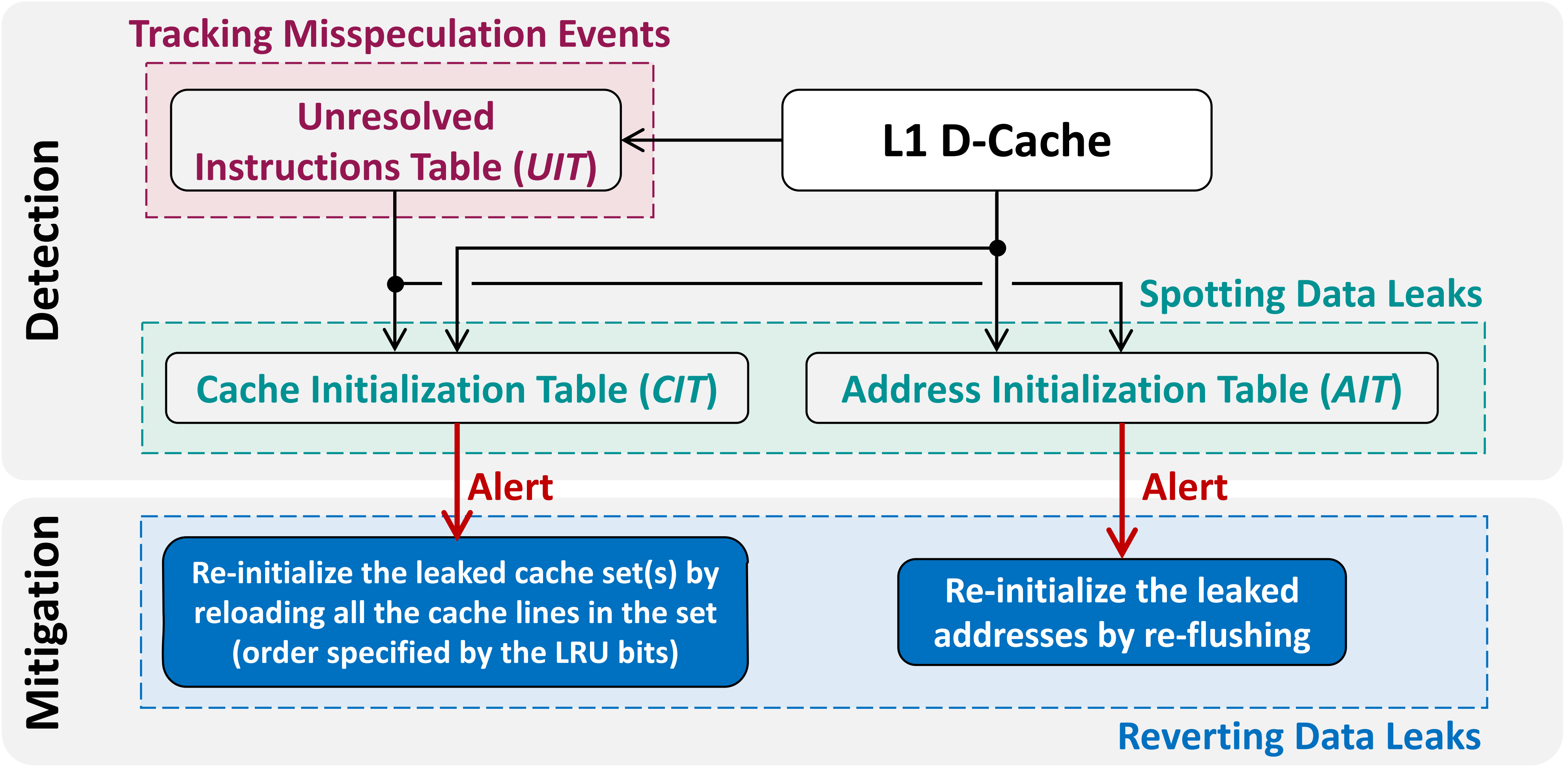
3. Minimizes performance and efficiency overheads, while comprehensively blocking all the leaks

HidFix Methodology

1. Spotting Speculative Data Leaks
   - *Leak Condition 1*: Cache location/memory address initialized by a potential attacker
   - *Leak Condition 2*: Misspeculatively accessed by a potential victim
2. Reverting Misspeculative Data Leaks
   - Re-initializing the cache locations and memory addresses that have misspeculatively leaked



HIDFIX Microarchitecture



Experimental Setup

- Simulation:
  - gem5 in syscall emulation (SE) mode
  - CACTI 6.5 for power and area overheads
- Benchmarks:
  - **Benign programs:** SPEC CPU2006 benchmark suite
  - **Malicious programs:** Spectre-v1 (Prime+Probe, Flush+Reload, Flush+Flush), prior attacks breaking ML-based detectors, our own new attacks
  - **Representatives:** ELFies as executable representative with a region size of 100M instructions

L1d/i size	32KB, 8-way
L2 size	256KB, 8-way
L3 size	1MB, 16-way
Fetch/dispatch/commit width	8/8/8
Branch Predictor	TAGE-SC-L-8KB
RF (INT/FP) size	256/256
LQ/SQ/IQ/ROB size	32/32/96/192
AIT/CIT size	512/512
UIT size	16

gem5 Configuration
(Skylake-like processors)

Security Evaluation

- Experimentally tested attacks
- HIDFIX shows 100% accuracy to spot misspeculative data leaks in known Spectre attacks:
  - Spectre Proof-of-Concept (PoC) attacks
  - ML evasive attacks (Spectify[1])
  - Our new attacks in this work
- HIDFIX rollbacks do not introduce new side effects to create SCAs
  - E.g., prior work[2] shows that E/M→S coherence state changes can introduce new vulnerabilities; *HIDFIX does not introduce such transitions*
  - Full security analysis in the paper

Attacks		Detection Accuracy	Mitigated?
Spectre PoC	Spectre-v1 (Prime+Probe)	100%	✓
	Spectre-v1 (Flush+Reload)	100%	✓
	Spectre-v1 (Flush+Flush)	100%	✓
Specify	Expanded-Spectre-NOP	100%	✓
	Expanded-Spectre-Mem	100%	✓
	Benign-Program-Spectre	100%	✓
This work	BENIGNINTERFERE	100%	✓
	SINGLEPROBE	100%	✓
	SINGLEPRIME	100%	✓

Performance Evaluation

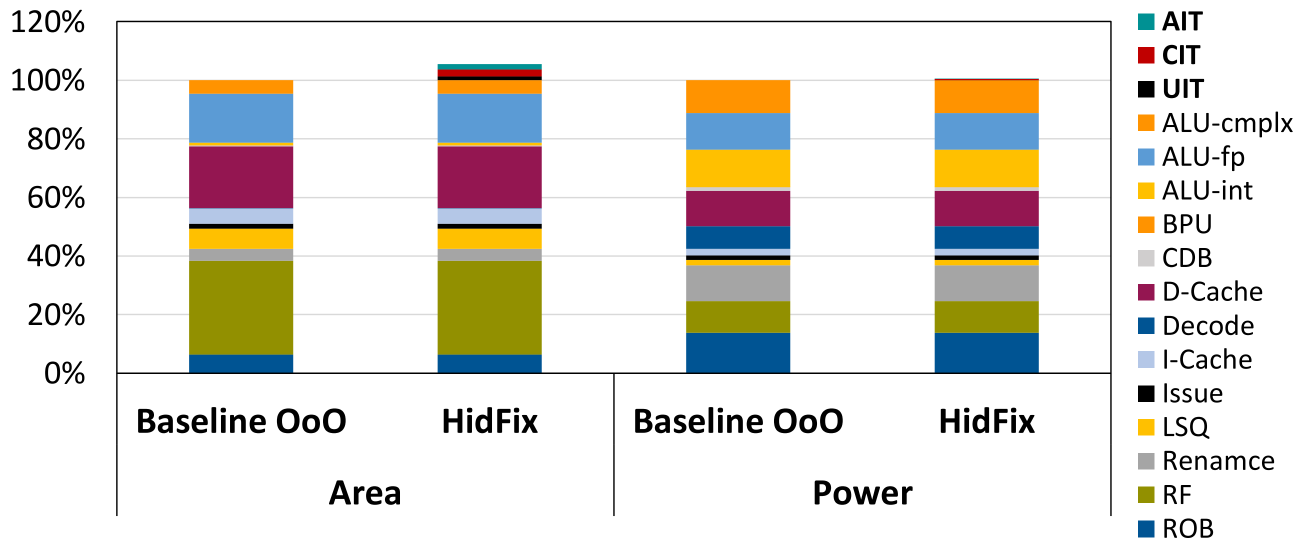
Application (SPEC CPU2006)	#leaks Detected	#cycles Baseline OoO core	#cycles HidFix core	Performance Overhead (%)
zeusmp	44	65,516,404	65,523,444	0.0107%
bwaves	4	110,485,539	110,486,179	0.0006%
bzip2	6	76,260,838	76,261,798	0.0013%
cactus	0	121,449,812	121,449,812	0.0000%
gamess	32	53,436,713	53,441,833	0.0096%
gcc	15	303,419,510	303,421,910	0.0008%
gobmk	7	97,271,448	97,272,568	0.0012%
libquantum	0	144,772,205	144,772,205	0.0000%
mcf	86	435,546,173	435,559,933	0.0032%
omnetpp	8	171,908,584	171,909,864	0.0007%
soplex	6	256,567,930	256,568,890	0.0004%
Average[1]	**18.45**	**135,986,161**	**135,989,670**	**0.0025%**

NUS National University of Singapore

geomean is used for performance overhead and number of cycles, and arithmetic mean for the number of leaks

Power and Area Overheads

- Power overhead: 0.5% over the baseline OoO core
  - Overheads come from CIT, AIT, and UIT tables
- The area overhead: 5.6% over the baseline core



Conclusion

- Blindly combining detections and mitigations is not sufficient
  - We present three new attacks to demonstrate the ineffectiveness of existing techniques
- HIDFIX: Co-designing detection and mitigation strategies
  - Near-zero performance overhead
  - End-to-end mitigation without the limitations of prior work
  - Not introducing new side effects resulting in new SCAs
  - Low area and power overheads


Thanks for your attention

HIDFIX: Efficient Mitigation of Cache-based Spectre Attacks via Hidden Rollbacks

Arash Pashrashid, Ali Hajiabadi and Trevor E. Carlson

National University of Singapore

42nd IEEE/ACM International Conference on Computer-Aided Design (ICCAD '23)