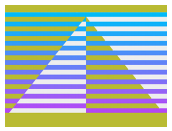
ILLINOIS TECH

PS-ORAM: Efficient Crash Consistency Support for Oblivious RAM on NVM

Gang Liu[1,2], Kenli Li[1], Zheng Xiao[1], ***Rujia Wang[2]***

[1] Hunan University, China

[2] Illinois Institute of Technology, USA



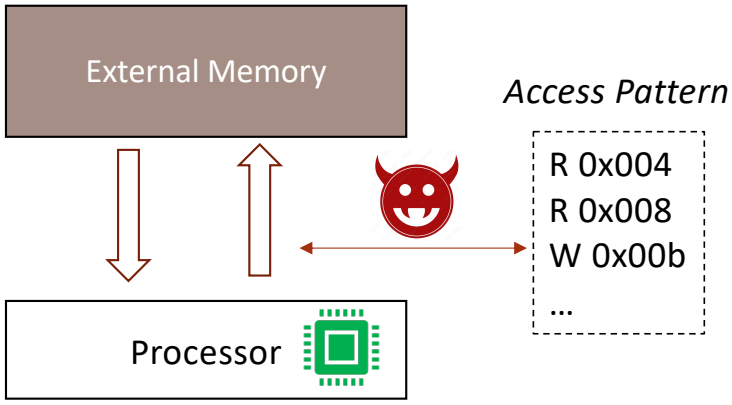
ISCA 2022

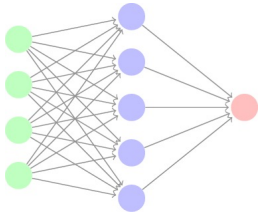
Outline

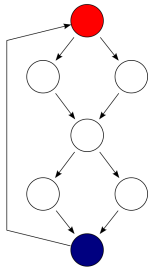
- Background
- Motivation
- Design of **PS-ORAM**
  - Architecture Design
  - Workflow
- Evaluation
- Summary

Background: access pattern side-channel leakage

- Threat Model
  - Trusted processor
  - Untrusted external memory/storage
  - An attacker may snoop the communication between memory and processor
- Encryption cannot hide memory access pattern
  - E.g., read/write intensities, frequencies, etc
  - Information may leak through the side-channel
- Example attacks:
  - Neural network architecture [1][2]
  - Control flow graph [3]



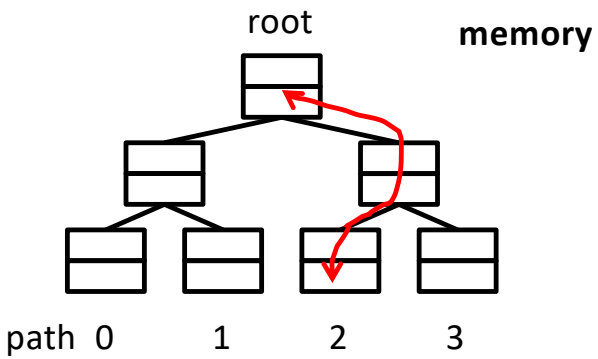




[1] Deepsniffer: A dnn model extraction framework based on learning architectural hints, ASPLOS'20
[2] Reverse engineering convolutional neural networks through side-channel information leaks, DAC'18
[3] HIDE: Hardware-support for Leakage-Immune Dynamic Execution, ASPLOS'04

Background: Oblivious RAM

- Tree-based Path Oblivious RAM (ORAM)[4]
  - ***L+1*** level
  - ***Z*** data blocks in each node



- **Basic steps:**
  1. Check Stash
  2. Access and Modify Position Map
  3. Load Path
  4. Update Stash
  5. Evict Path

Position Map

Stash

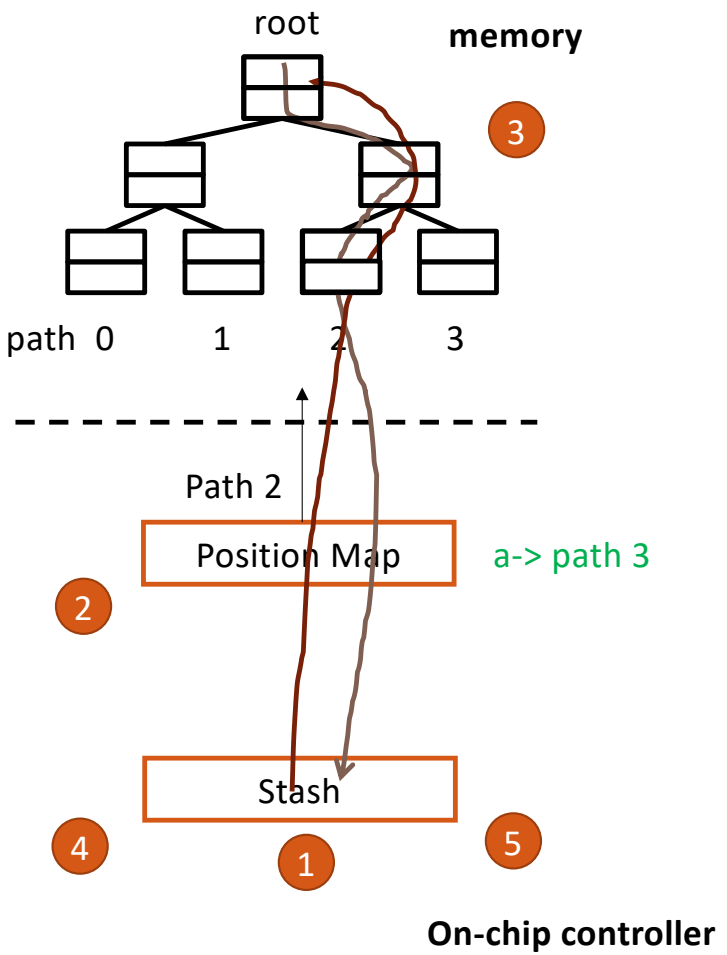
On-chip controller

[4] Path ORAM: an extremely simple oblivious RAM protocol, CCS'13

Background: Oblivious RAM

- Tree-based Path Oblivious RAM (ORAM)[4]
  - ***L+1*** level
  - ***Z*** data blocks in each node

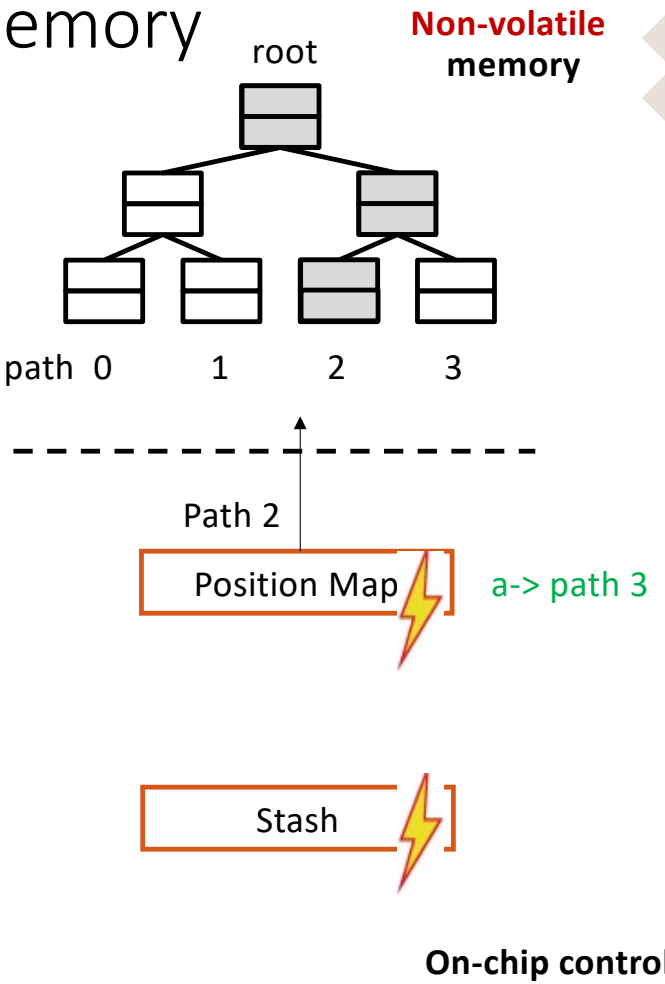
- **Basic steps:**
  1. Check Stash
  2. Access and Modify Position Map
  3. Load Path
  4. Update Stash
  5. Evict Path



[4] Path ORAM: an extremely simple oblivious RAM protocol, CCS'13

Challenges: ORAM with Persistent Memory

- What if the ORAM system is using persistent non-volatile memory (NVM)?
- If there is a crash:
  - What are the cases for data inconsistency?
  - How to enhance the system for ORAM crash consistency?
  - How to minimize performance and hardware overhead, while securing the system?

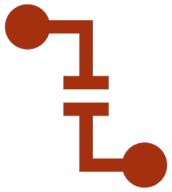


Common crash-consistent approaches

Software-based

(e.g., Logging and Copy-on-Write)

- Inefficient and slow for recovery;
- Memory space overhead;
- Compromise security.



Hardware-based

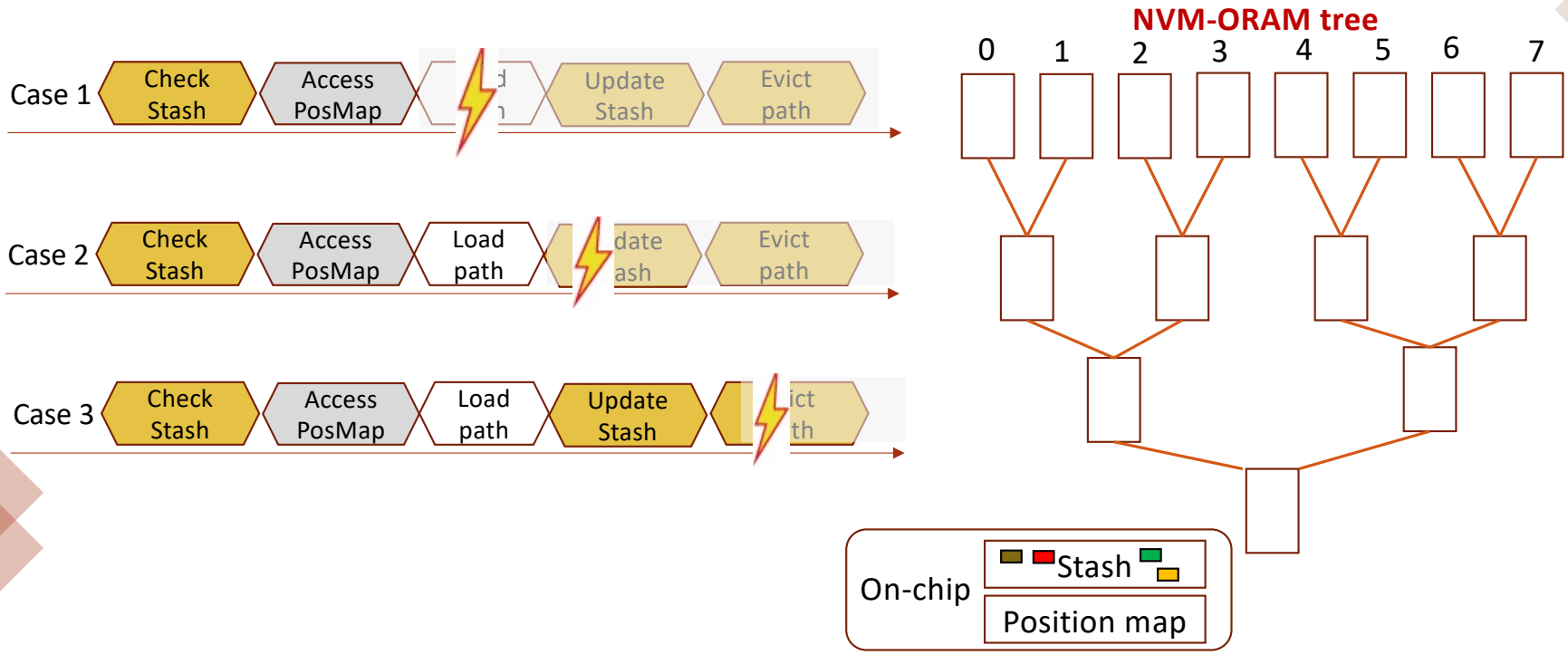
NVM-based register and cache are slower;

eADR is not common for all types of processor;

ADR cannot directly support ORAM.

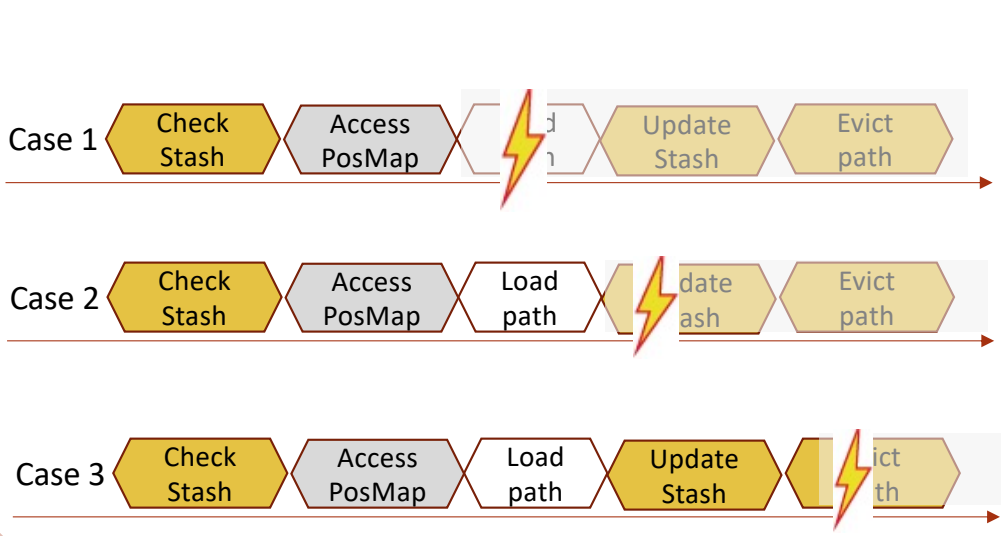
Crash Analysis (1)

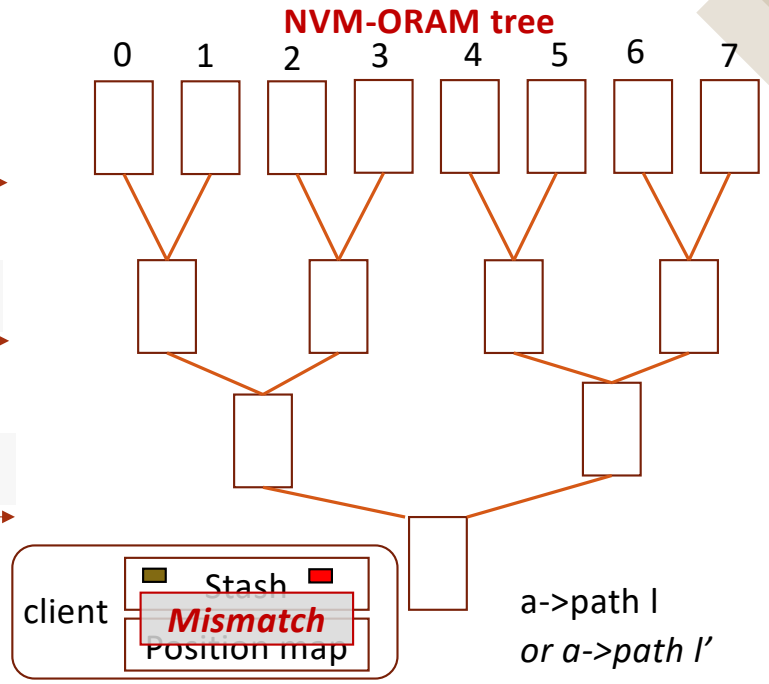
- *When the ORAM system crashes, it could cause the loss of modified data blocks stored in the stash.*



Crash Analysis (2)

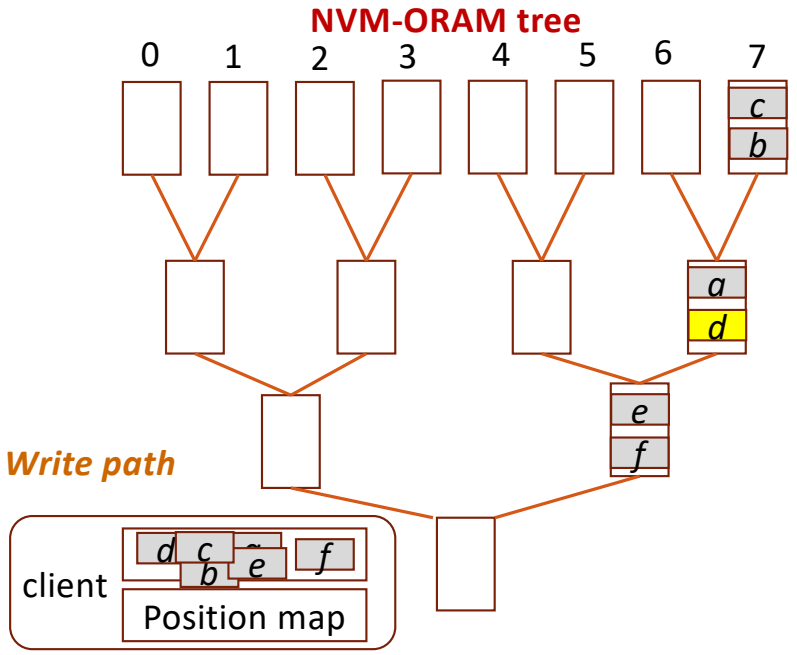
- *When the ORAM system crashes, the PosMap data that stores the path ids could be inconsistent with data blocks.*





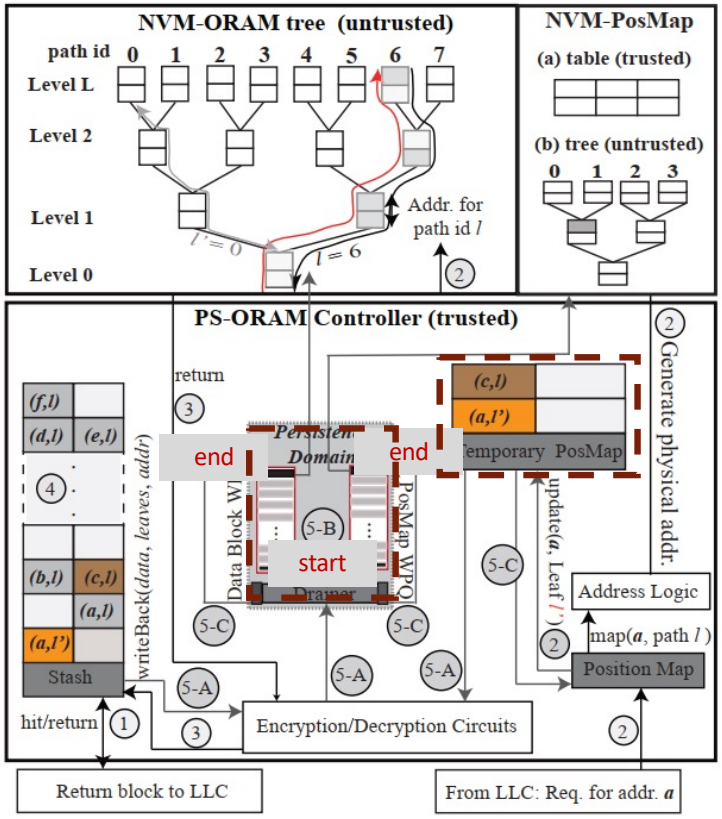
Crash Analysis (3)

- If the system crash during the write-back step, it may cause data blocks overwritten in NVM.



Crash-consistent ORAM: PS-ORAM Architecture

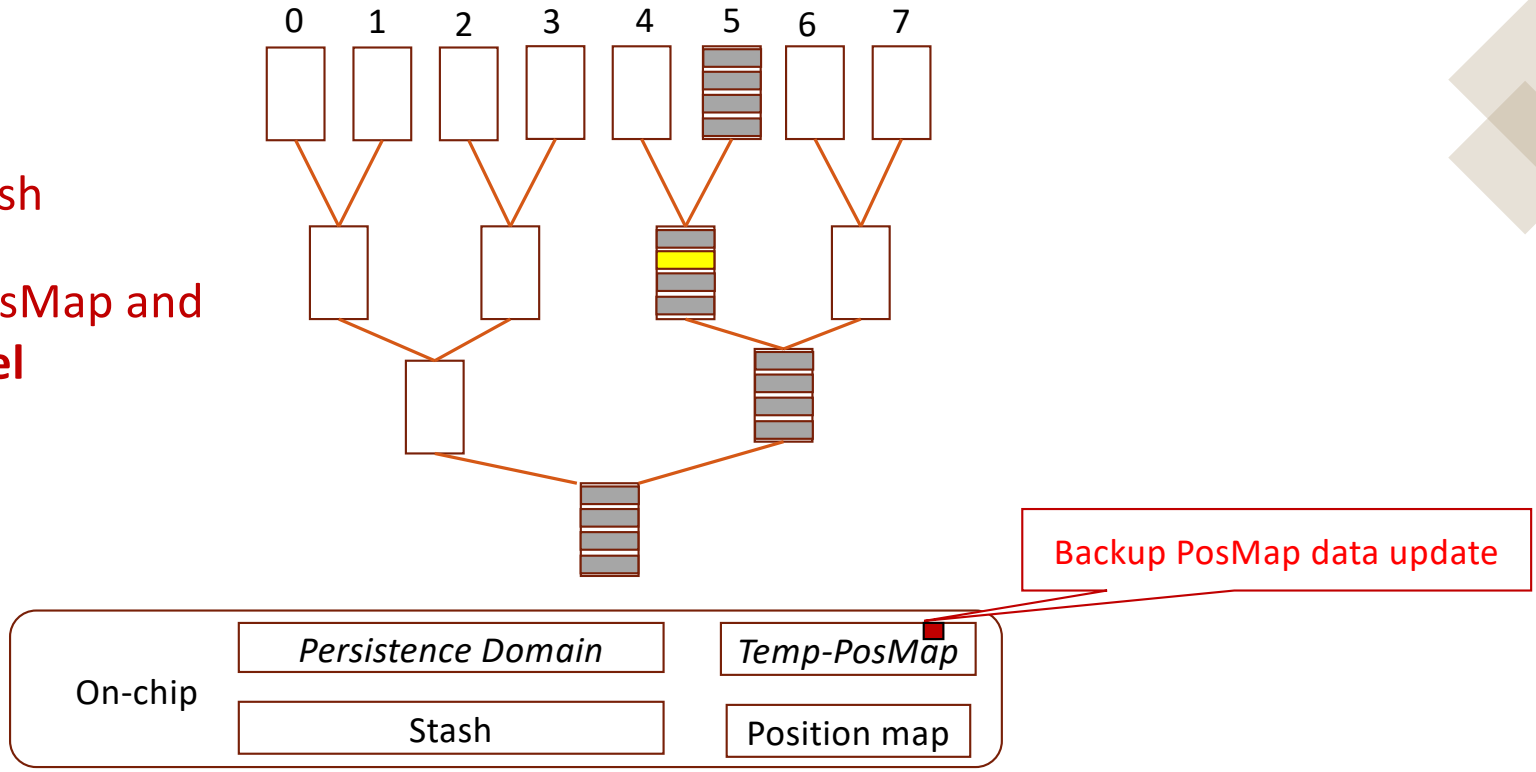
- ADR-type on-chip persistence domain
  - Write Pending Queues (WPQs)
  - Drainer
- ORAM controller modification
  - Temporary PosMap, storing dirty PosMap entries



PS-ORAM Workflow

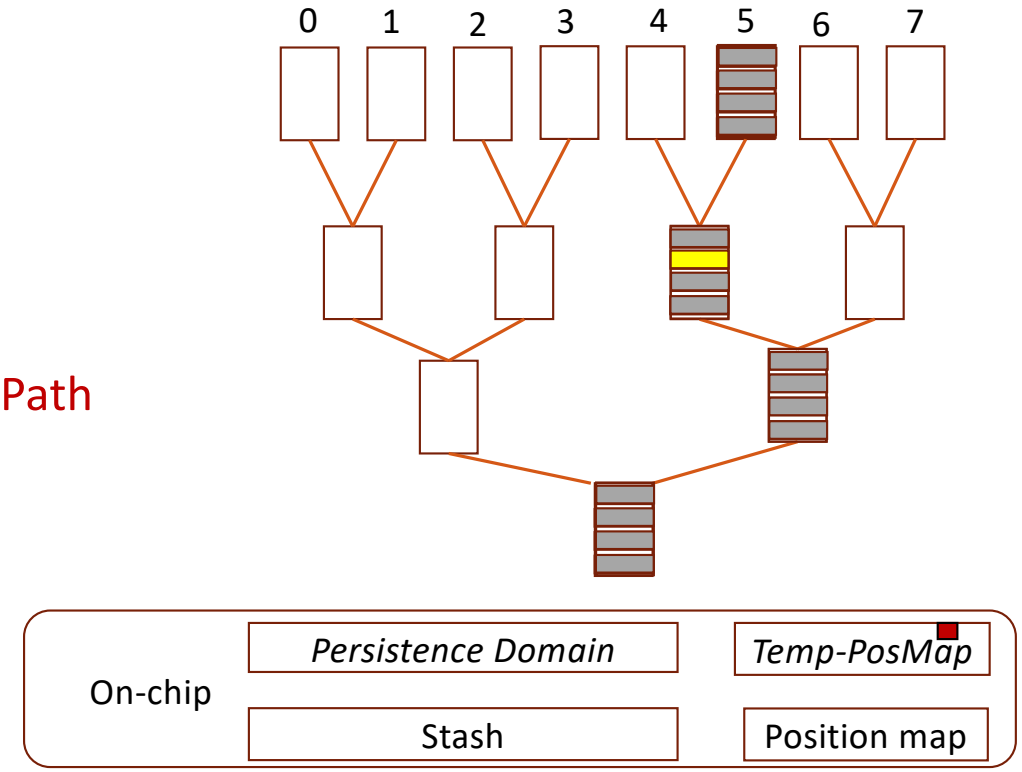
1. Check Stash

2. Access PosMap and **Backup Label**



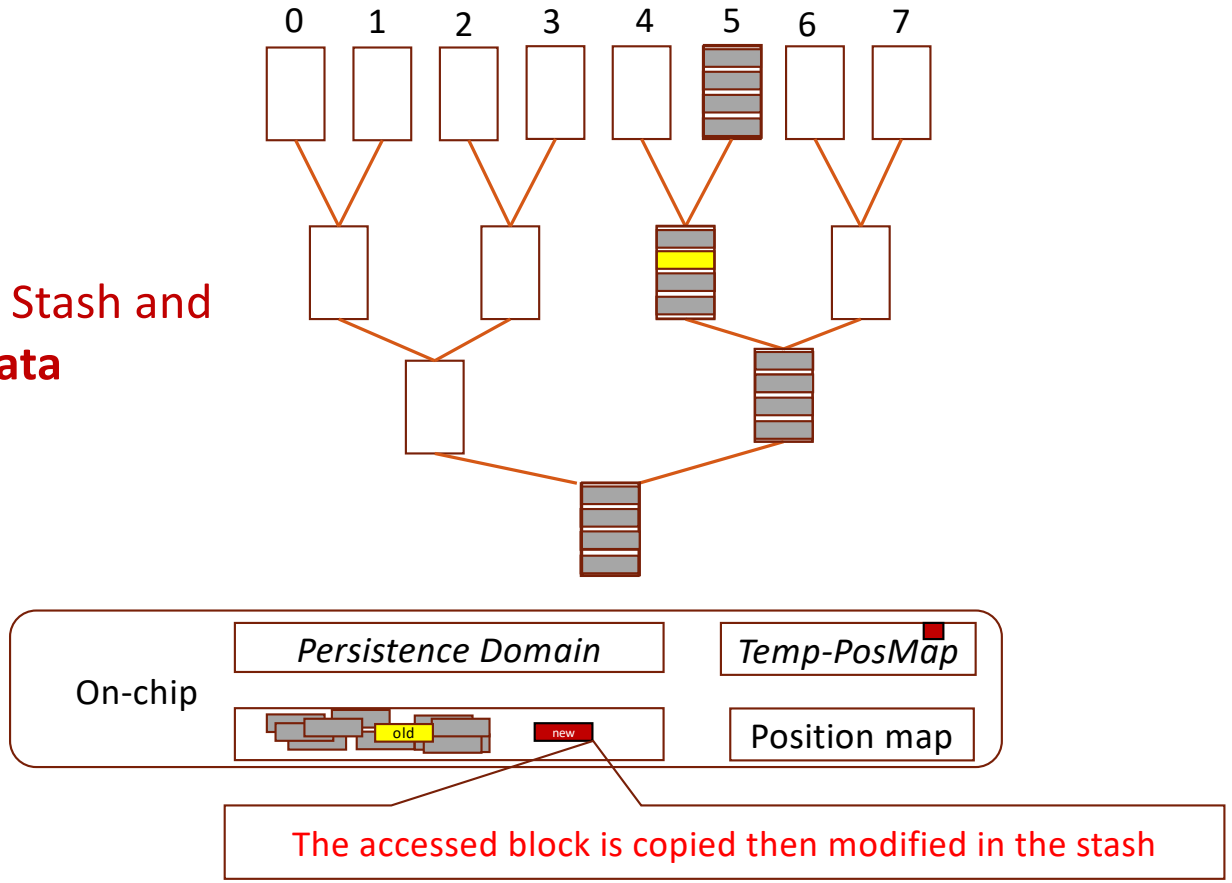
PS-ORAM Workflow

3. Load Path



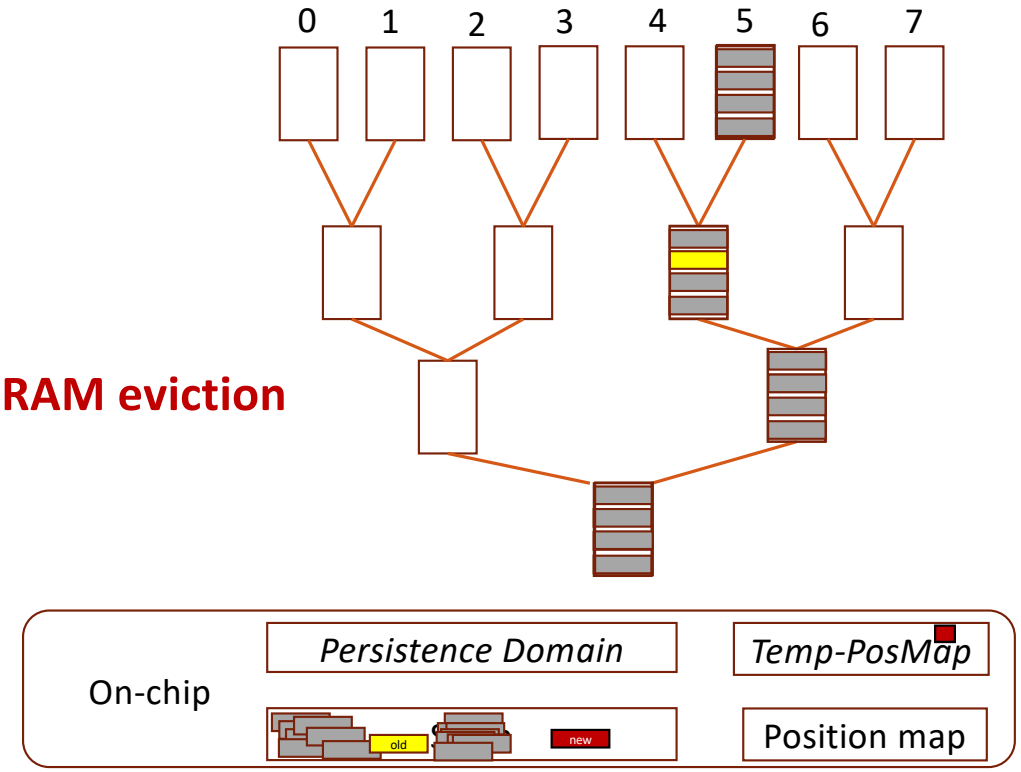
PS-ORAM Workflow

4. Update Stash and **Backup Data**



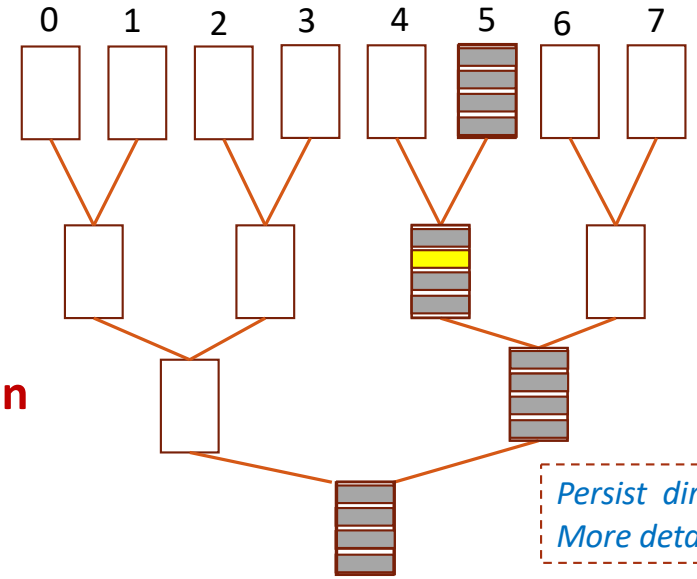
PS-ORAM Workflow

5. PS-ORAM eviction

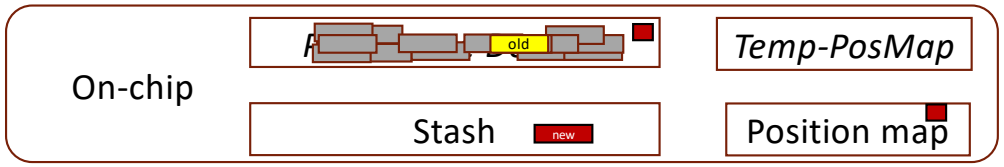


PS-ORAM Workflow

5. PS-ORAM eviction



Persist dirty PosMap entries in NVM
More details in paper



What PS-ORAM achieves

Low-cost data persistence

Persist data block without excessive additional write

Atomic metadata writeback

Synchronize metadata and data write back operations

Secure

No relaxation on threat model

Does not introduce additional side-channels

Evaluation Methodology

- Cycle-accurate gem5 simulator + NVMain 2.0 for the NVM-based main memory
- Workloads: SPEC06+ ORAM protocols

(a) On-chip processor and cache

Core type/frequency	in-order (1 core), 3.2 GHz
L1 I/D cache L1 read/write	32KB/32KB, 2-way LRU 2/2-cycle
L2 cache L2 read/write	1MB shared, 8-way LRU 20/20-cycle

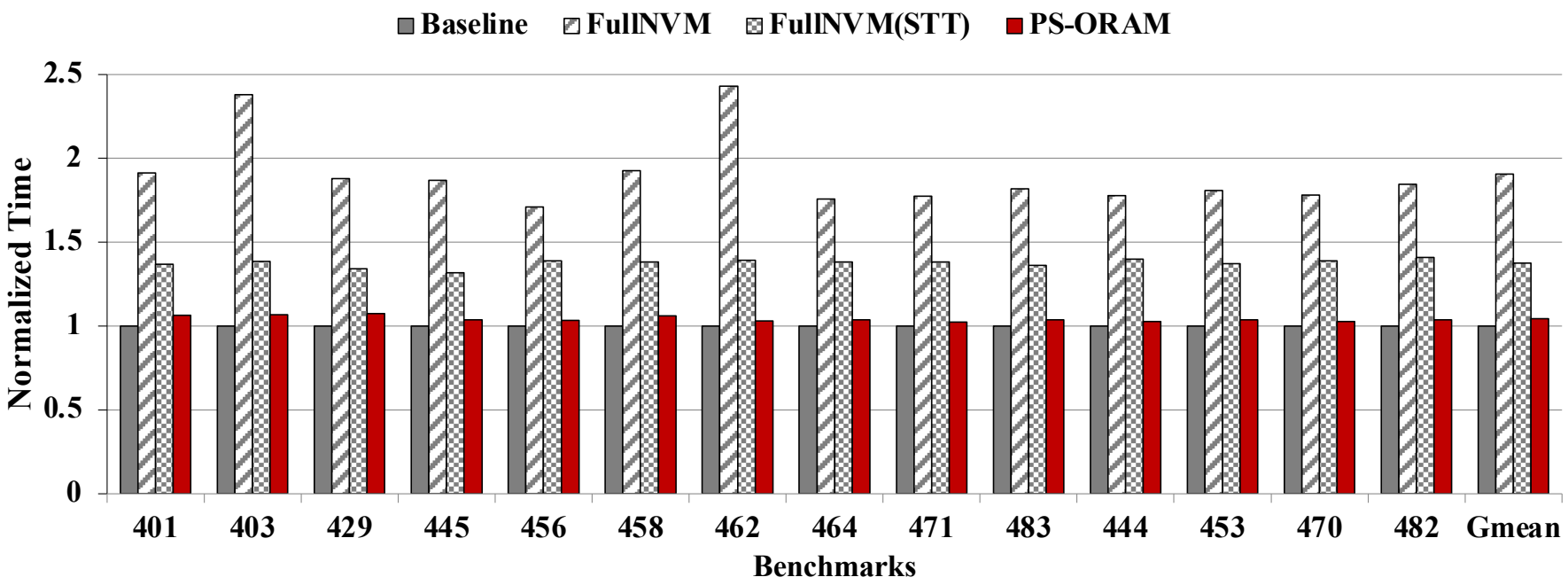
(b) Persistence domain

PCM	4GB, 400MHz , $tRCD$ /tWP /tCW D /tWT R/tRP /$tCCD$=48/60/4/3/1/2
STTRAM	4GB, 400MHz , $tRCD$ /tW P /tCW D /tWT R/tRP /$tCCD$ =14/14/10/5/1/2
WPQs	96/4-entry for PosMap WPQ, 96/4-entry for Data WPQ

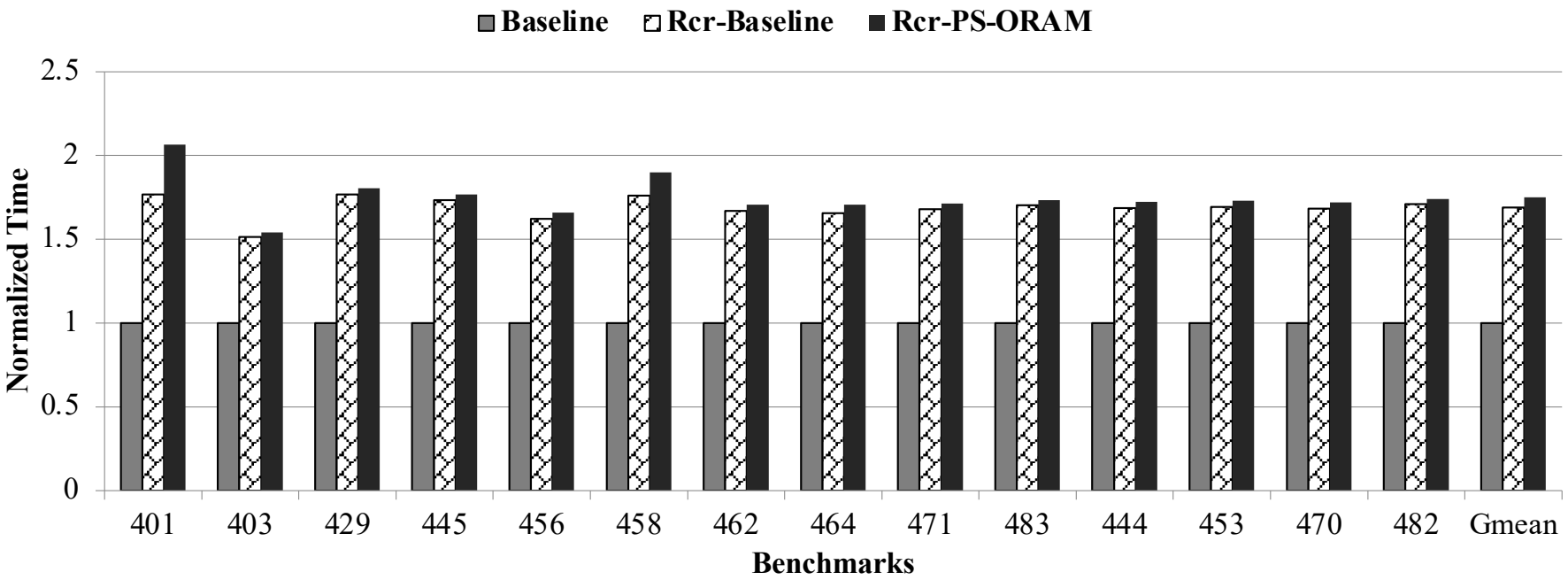
(c) ORAM controller

Data block size	64B
Data ORAM capacity	4GB (L = 23)
Block slots per bucket (Z)	4
Stash size (C)	200-entry
Temporary PosMap size (C_{tPos})	96-entry
AES-128 latency	32 cycles

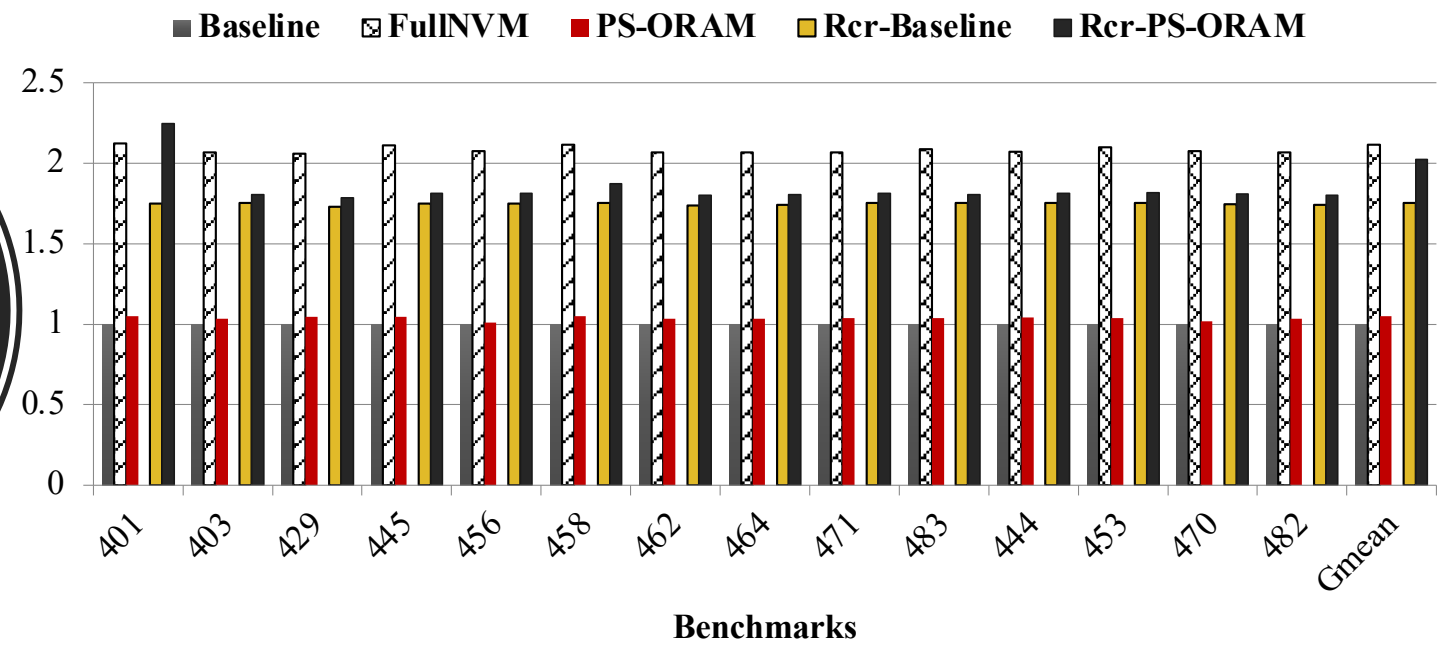
Overall Performance (non-recursive ORAM)



Overall Performance (recursive ORAM)



NVM write traffic comparison



Draining Cost Comparison of PS-ORAM and eADR-ORAM

Estimated draining energy and time cost for PS-ORAM vs. eADR

Technology	eADR		PS-ORAM(WPQ sizes)		Normalized to PS-ORAM (WPQ size=96 / 4)		
System	eADR-cache	eADR-ORAM	96-entries	4-entries	eADR-cache	eADR-ORAM	PS-ORAM
Energy	12.653mJ	2.286J	76.530uJ	2.83uJ	165X / 4471X	29870X / 807797X	1
Time	26.638us	4.817ms	161.134ns	6.713ns	165X /3968X	29894X / 717563X	1

Summary

- Crash analysis on ORAM built with persistent memory
- Challenging to adopt existing solutions to build crash-consistent ORAM
- Proposed PS-ORAM design
  - Moderate overhead architecture
  - Atomic persistence protocol
- Minimal performance overhead:
  - Non-recursive and recursive PS-ORAM only incurs 4.29% and 3.65% additional performance overhead compared to a system without crash consistency support.

Thank you

Questions?

Contact:
liug@hnu.edu.cn and
rwang67@iit.edu

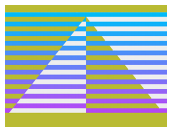
ILLINOIS TECH

PS-ORAM: Efficient Crash Consistency Support for Oblivious RAM on NVM

Gang Liu[1,2], Kenli Li[1], Zheng Xiao[1], ***Rujia Wang[2]***

[1] Hunan University, China

[2] Illinois Institute of Technology, USA



ISCA 2022