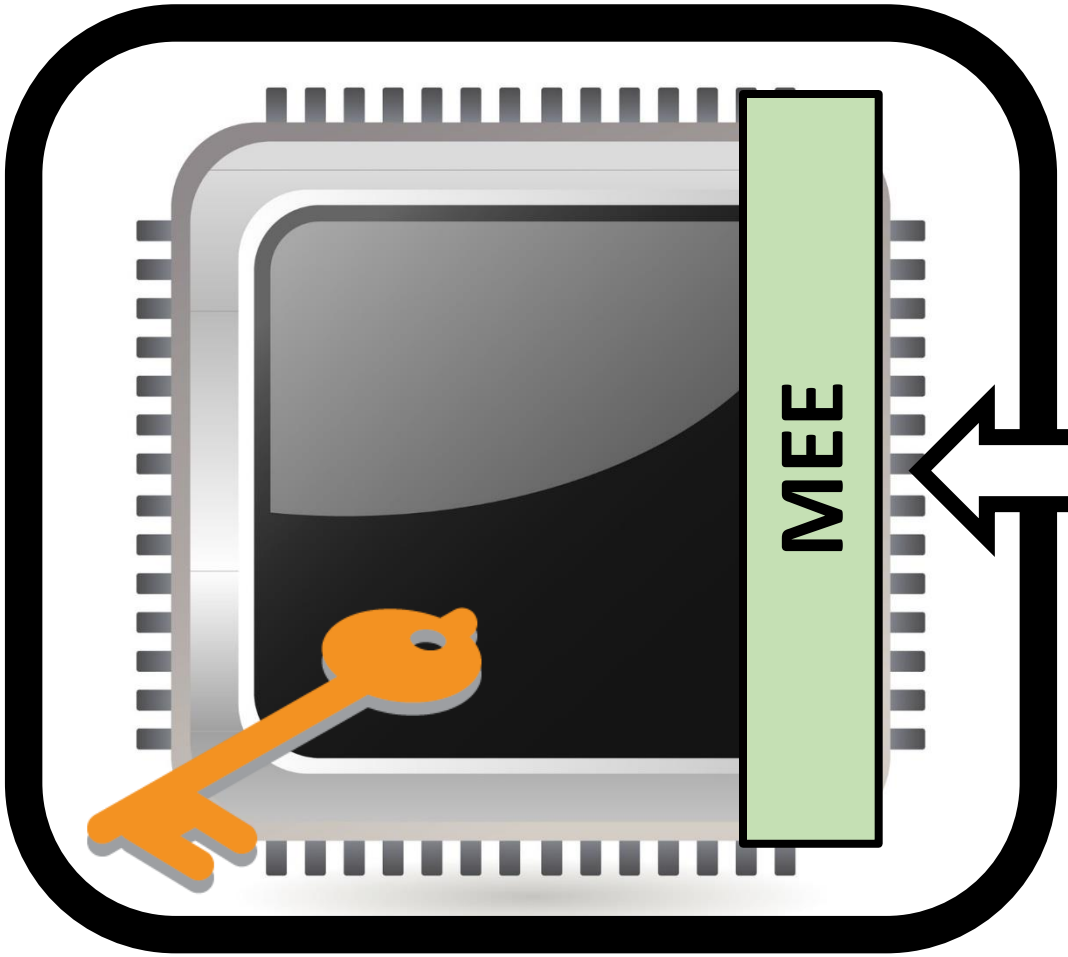
The Ascend Secure Processor

Christopher Fletcher
MIT

Joint work with

- Srini Devadas, Marten van Dijk
- Ling Ren, Albert Kwon, Xiangyao Yu
- Elaine Shi & Emil Stefanov
- David Wentzlaff & Princeton Team (Mike, Tri, Jonathan, Alexey, Yaosheng)
- Omer Khan

Last talk: Intel SGX



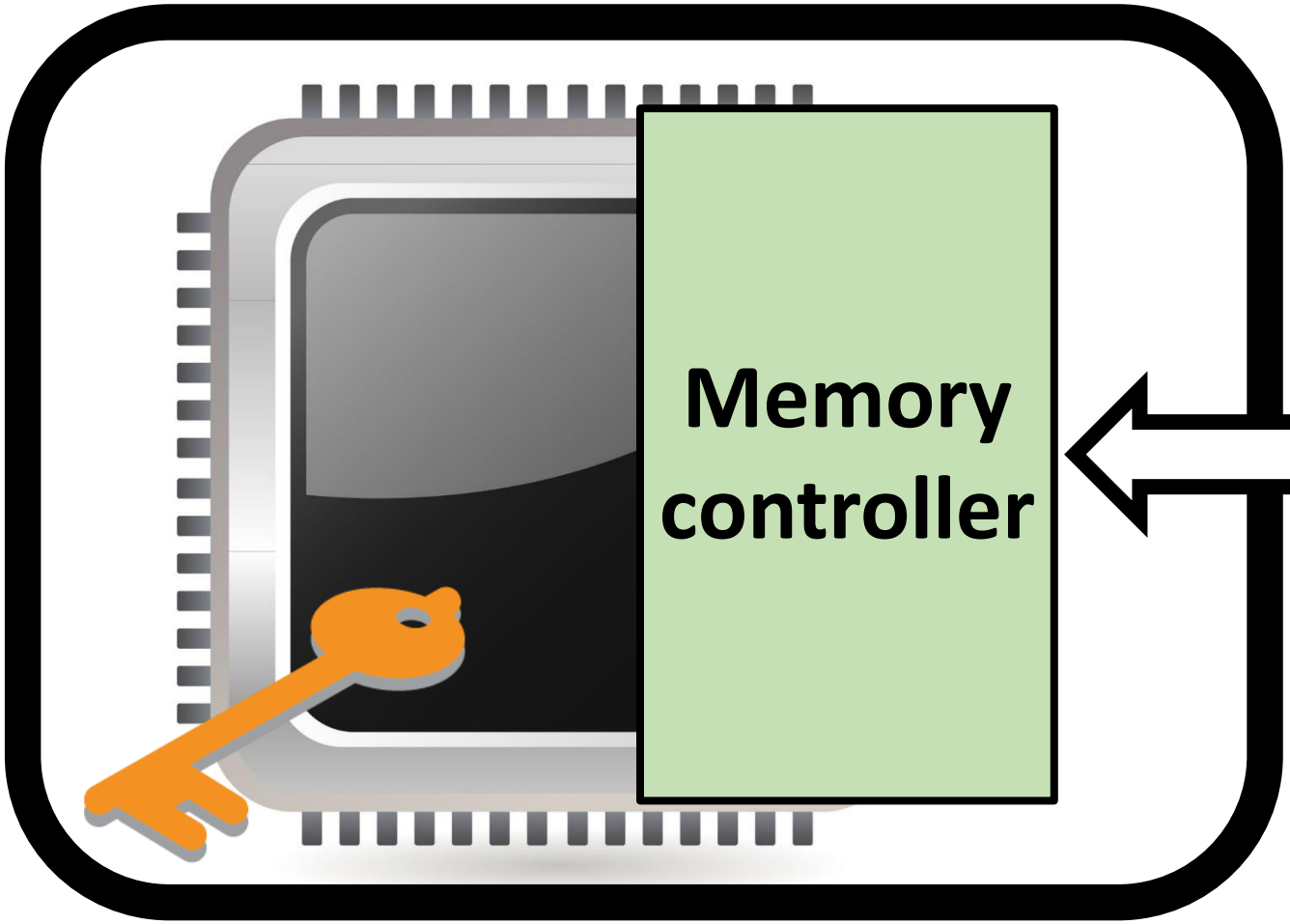
Data

Integrity

Address

Timing

Ascend Processor



This talk

Data Integrity

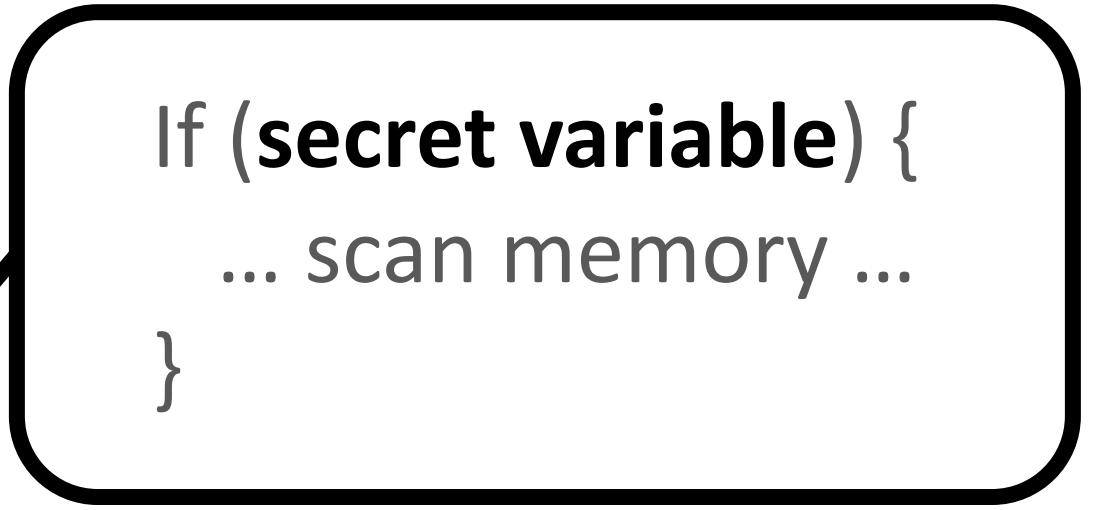
Address

Timing

Outline

- Motivation + Oblivious RAM (ORAM) primer
- ORAM in Hardware
- Demo ☺

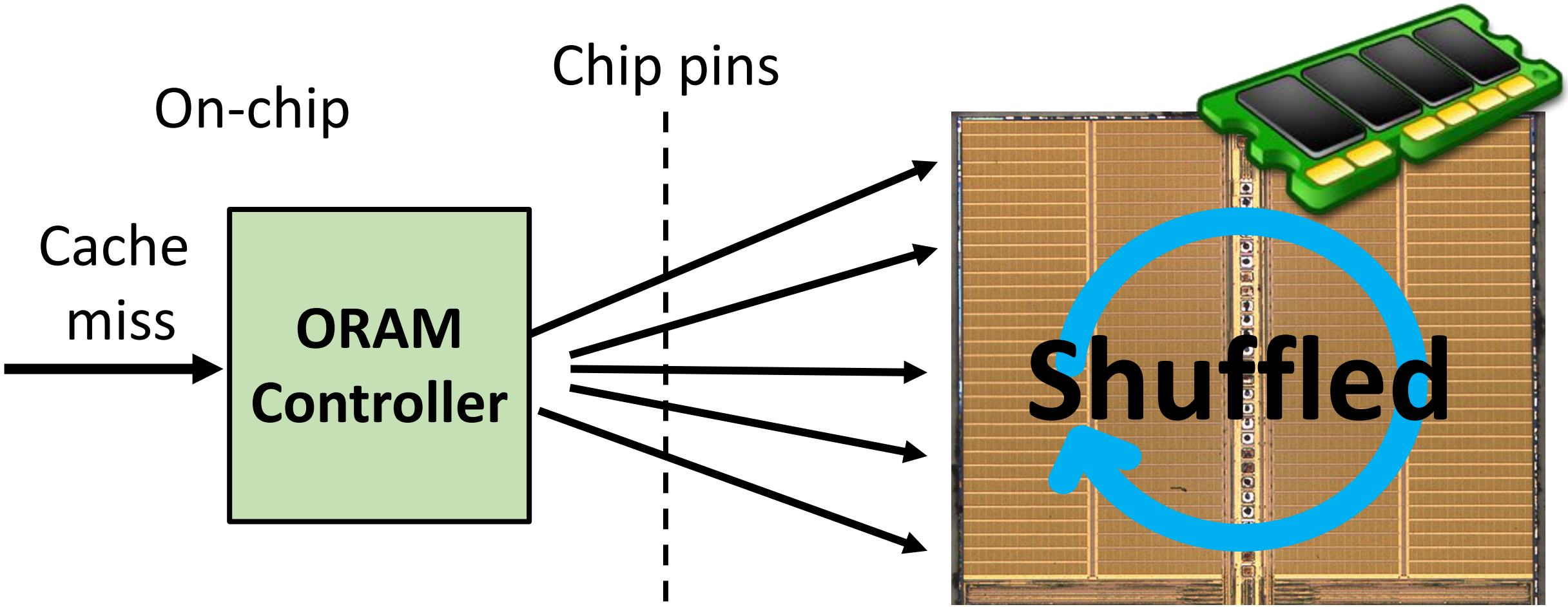
Op	Address	Time
R	0	1
W	1	10
R	5	15
R	6	16
R	7	17



Binary search

- SGX broken through page faults [Xu et al.'15]
- Shared library usage [Zhuang et al.'04]
- Search queries [Islam et al.'12]

Oblivious RAM (ORAM) [Goldreich-Ostrovsky'96]



Provably removes all access pattern leakage

ORAM security definition

- Access is 3 tuple: (**op** = Read/write, **address**, **data**)

- Consider access sequences A and A’

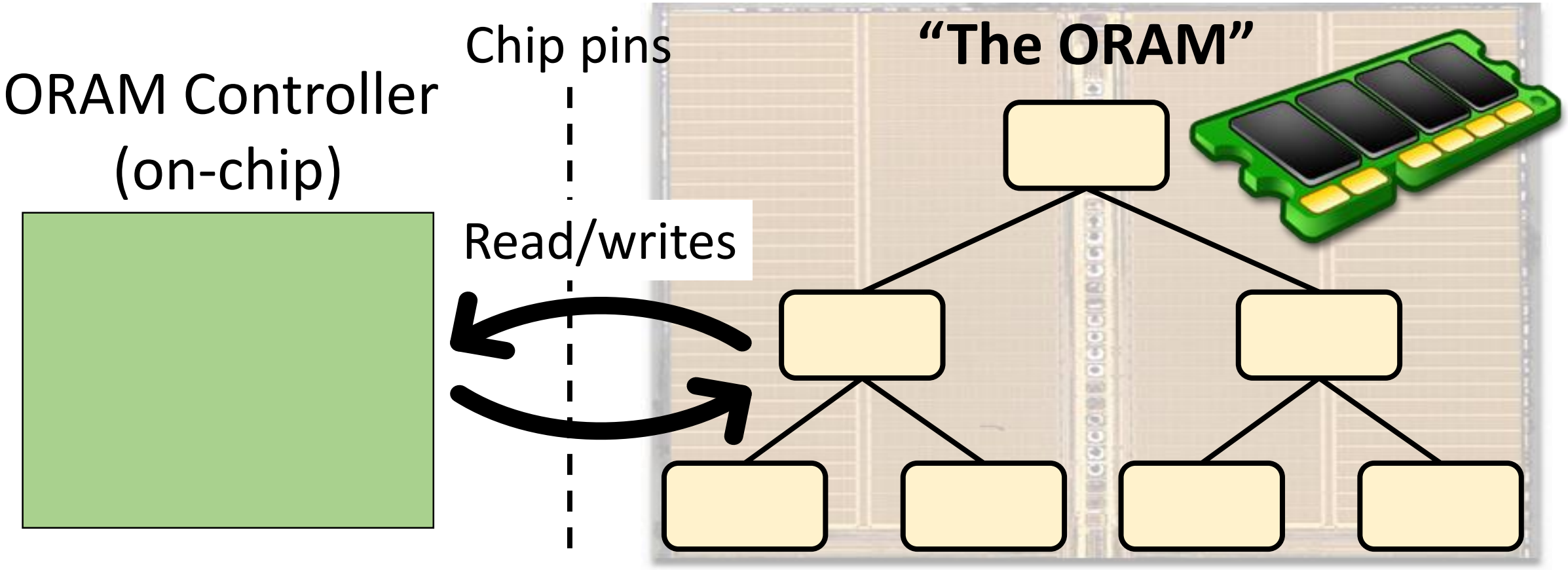
  $A = [(op_1, address_1, data_1), \quad (op_2, address_2, data_2), \ldots]$

  $A' = [(op_1', address_1', data_1'), \quad (op_2', address_2', data_2'), \ldots]$

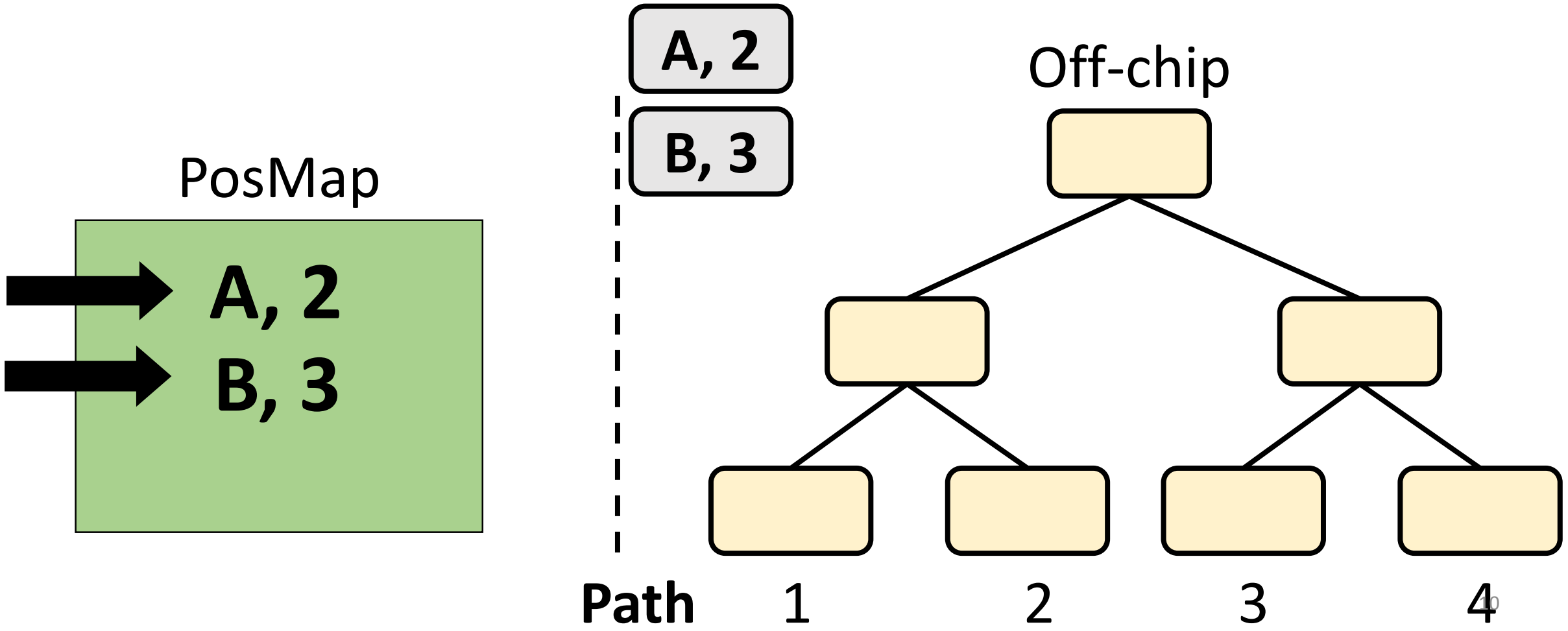
- If $|A| == |A'|$

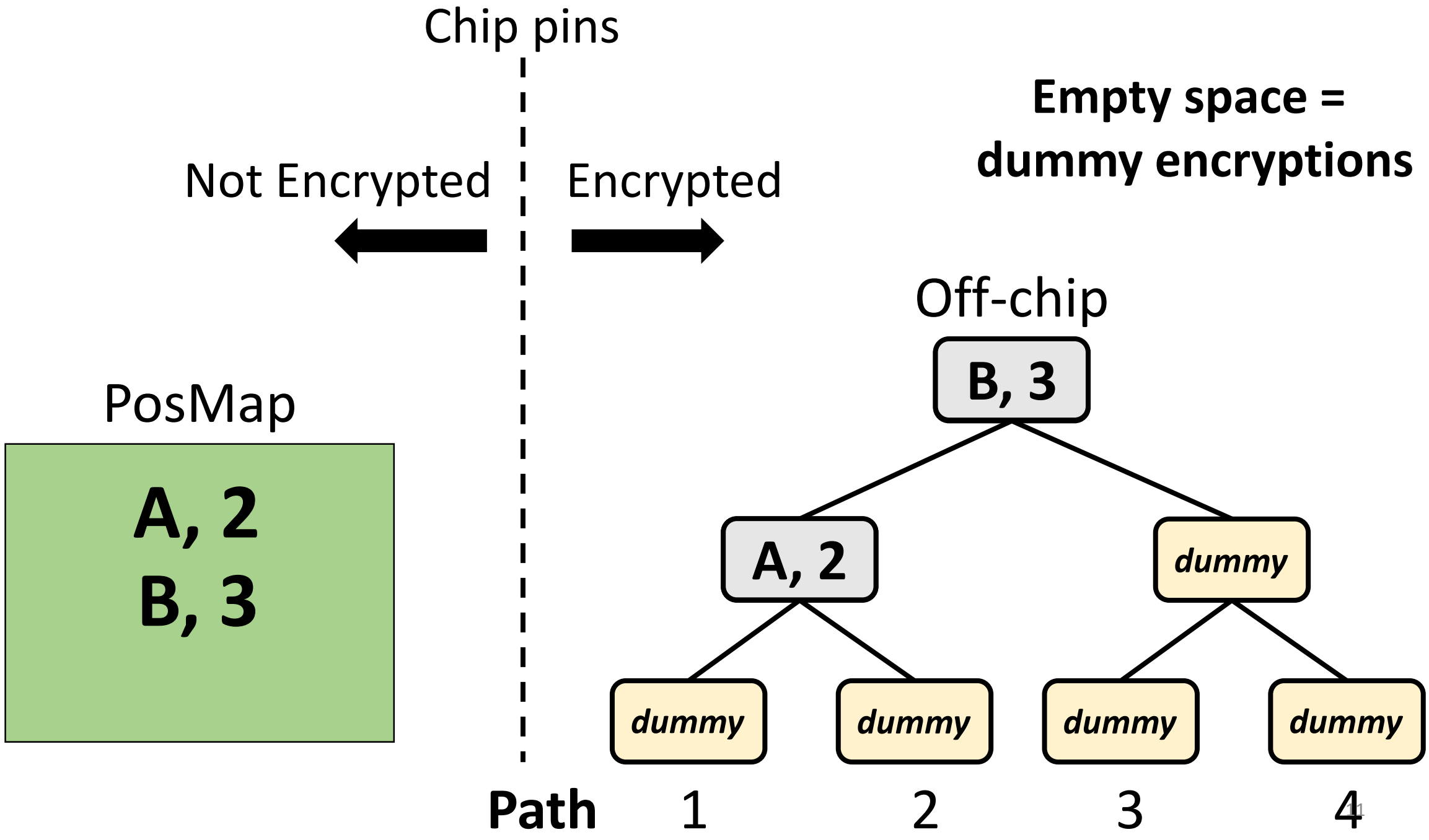
  then $ORAM(A) \approx ORAM(A')$

Path ORAM [CCS'13]



Block assigned to ***random*** path.
Block ***lives*** *on* that path.



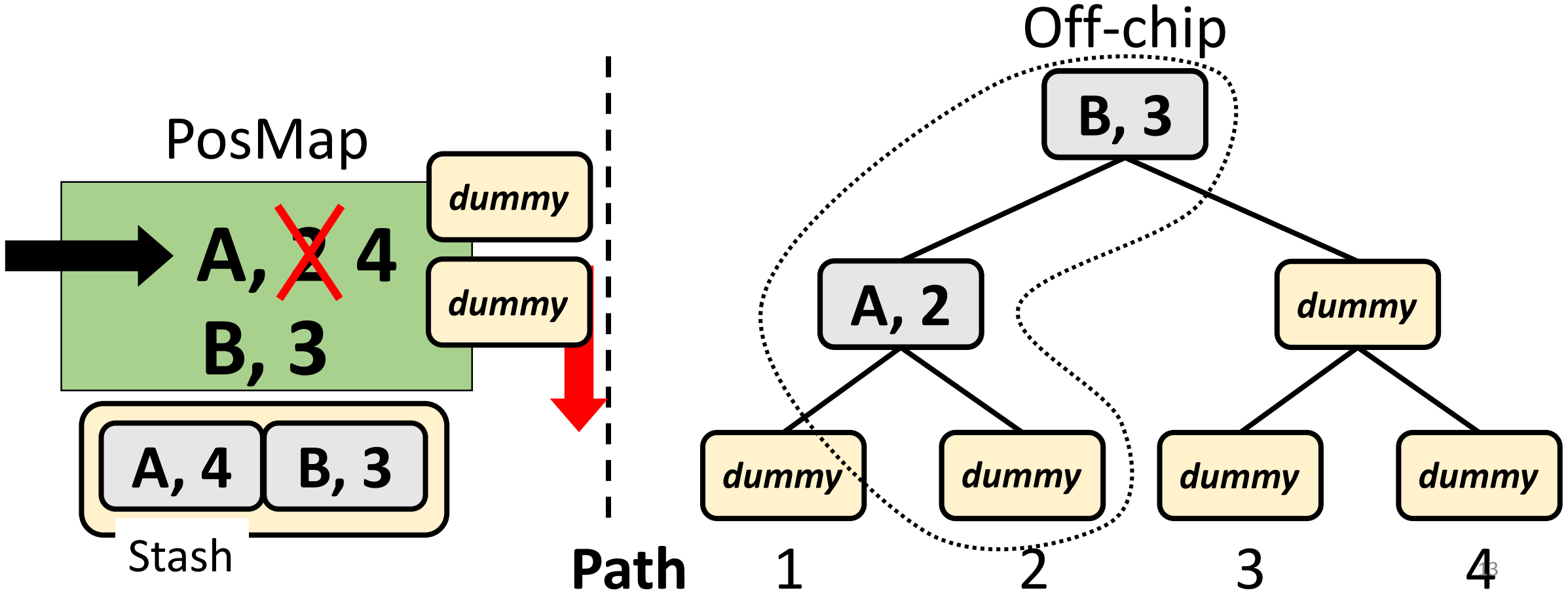

Chip pins
Not Encrypted
Encrypted
Empty space = dummy encryptions
Off-chip
PosMap
A, 2
B, 3
B, 3
A, 2
dummy
dummy
dummy
dummy
dummy
Path
1
2
3
4

Path ORAM Access:

Read+write the path the block is assigned to.

Path ORAM Access:

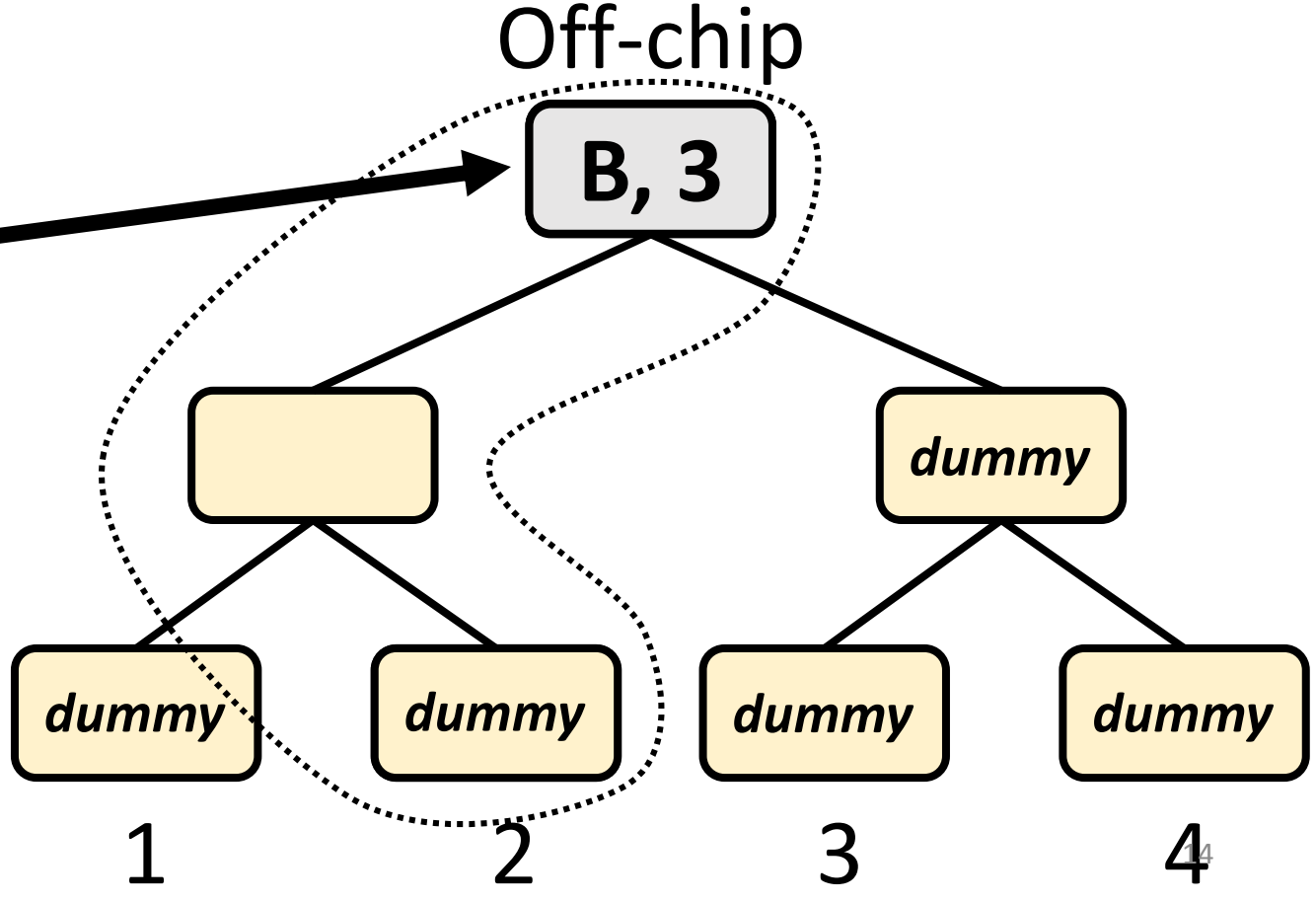
Read+write the path the block is assigned to.

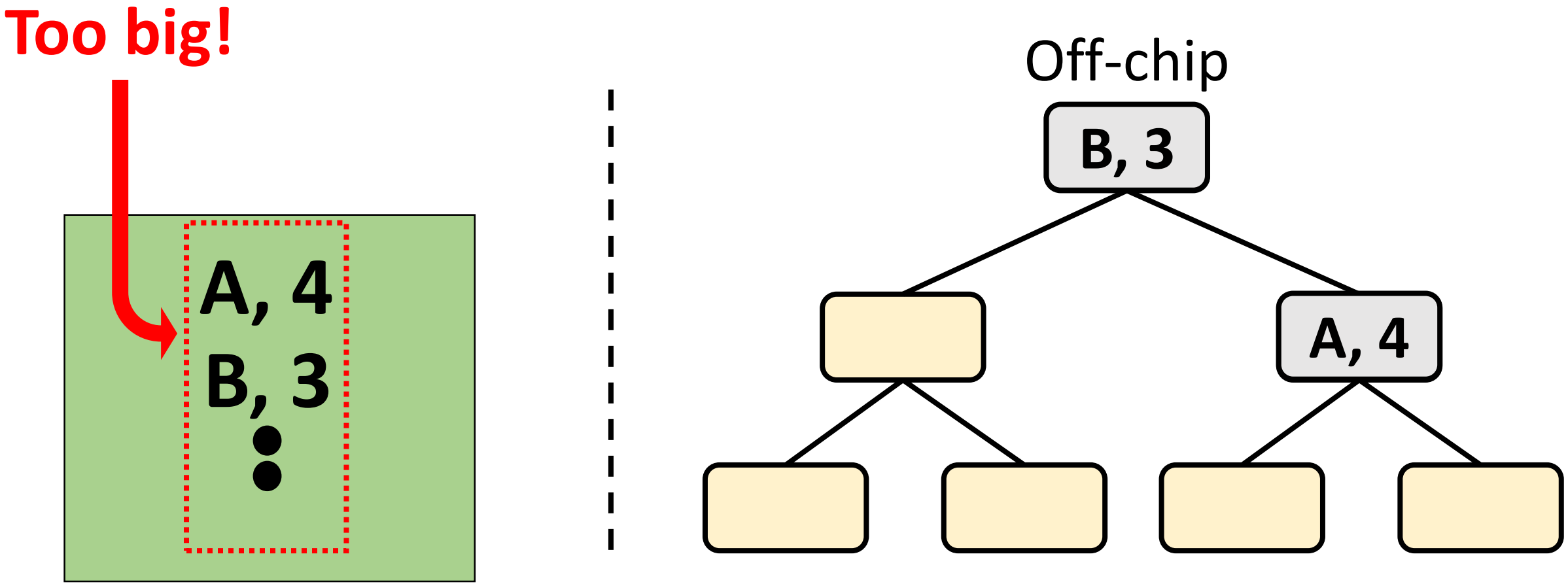


Typically, 4 slots per bucket “Z=4”

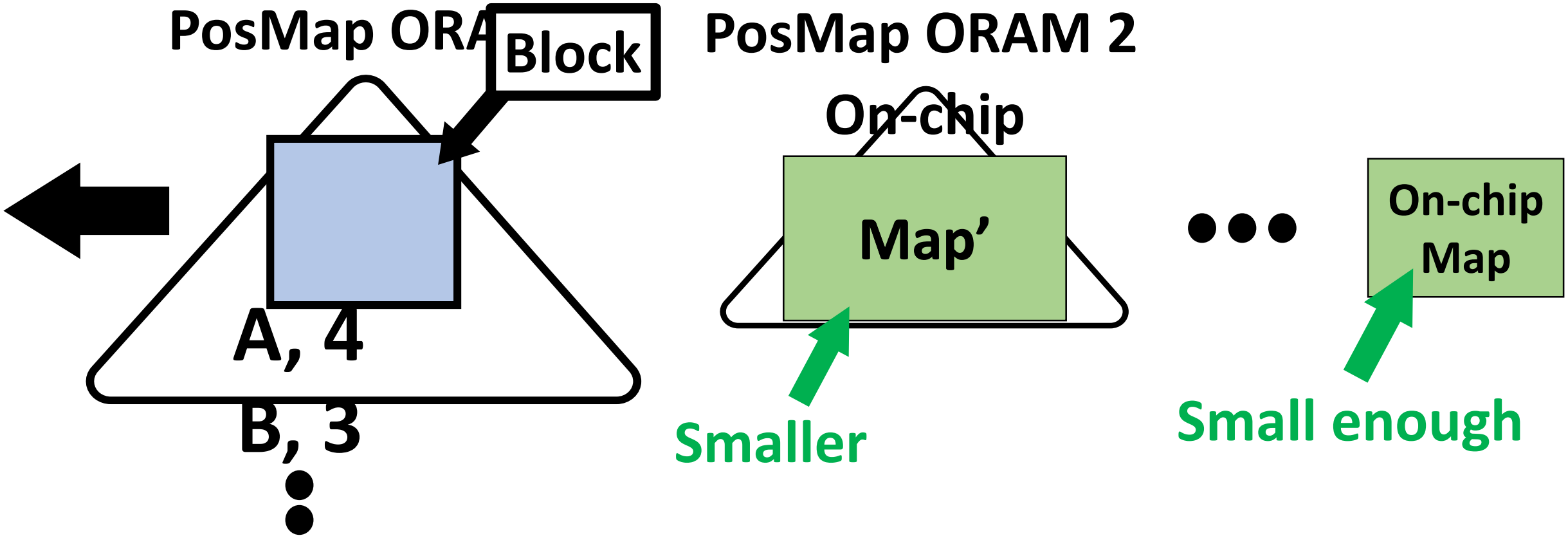
Z=1

…for simplicity

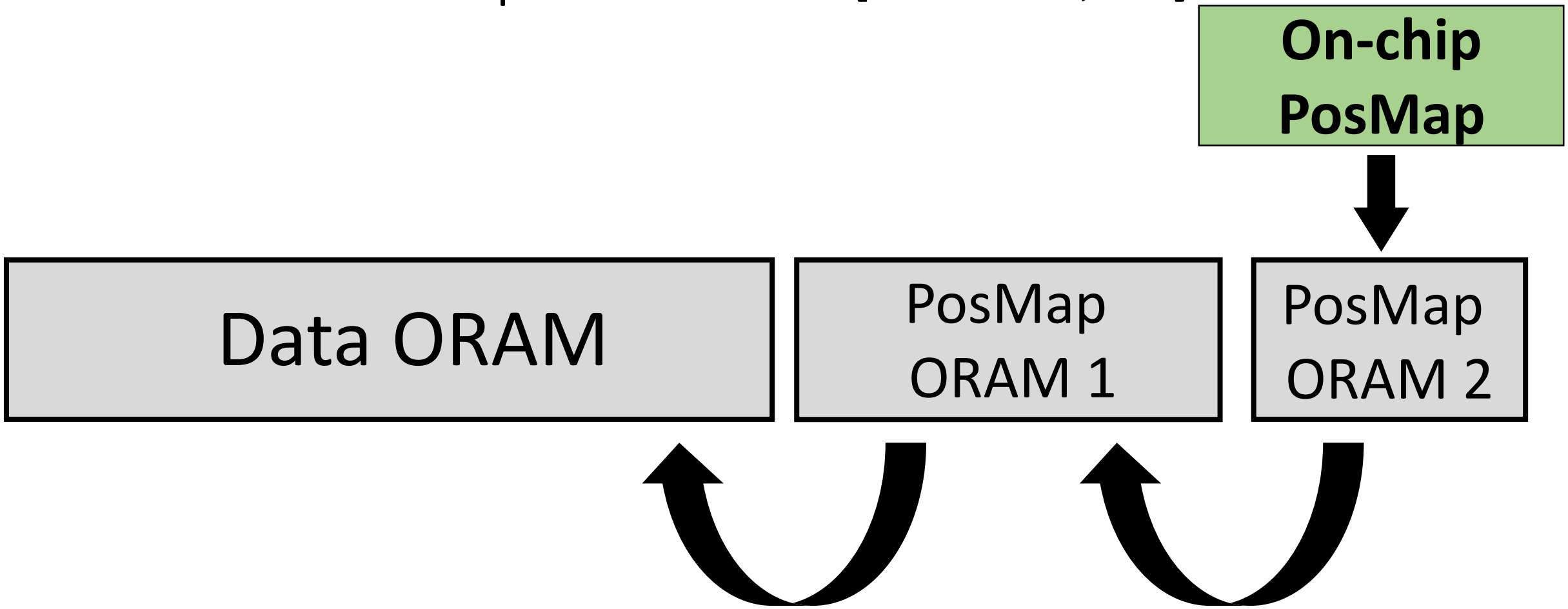



Too big!
A, 4
B, 3
Off-chip
B, 3
A, 4

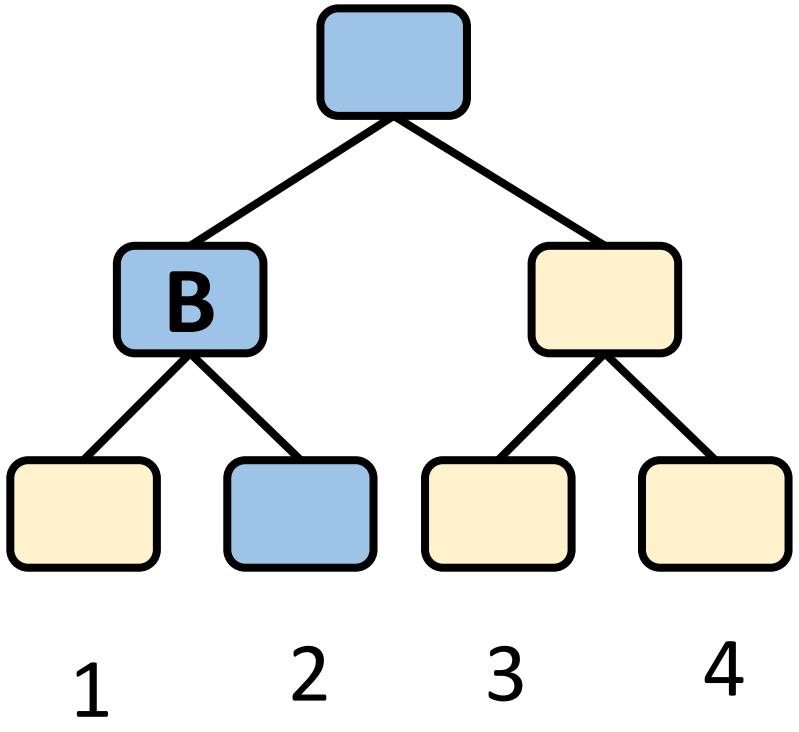
Map recursion [Shi et al., 11]



Map recursion [Shi et al., 11]



Path ORAM summary



Blocks assigned to paths.

Access block: Read+write path.

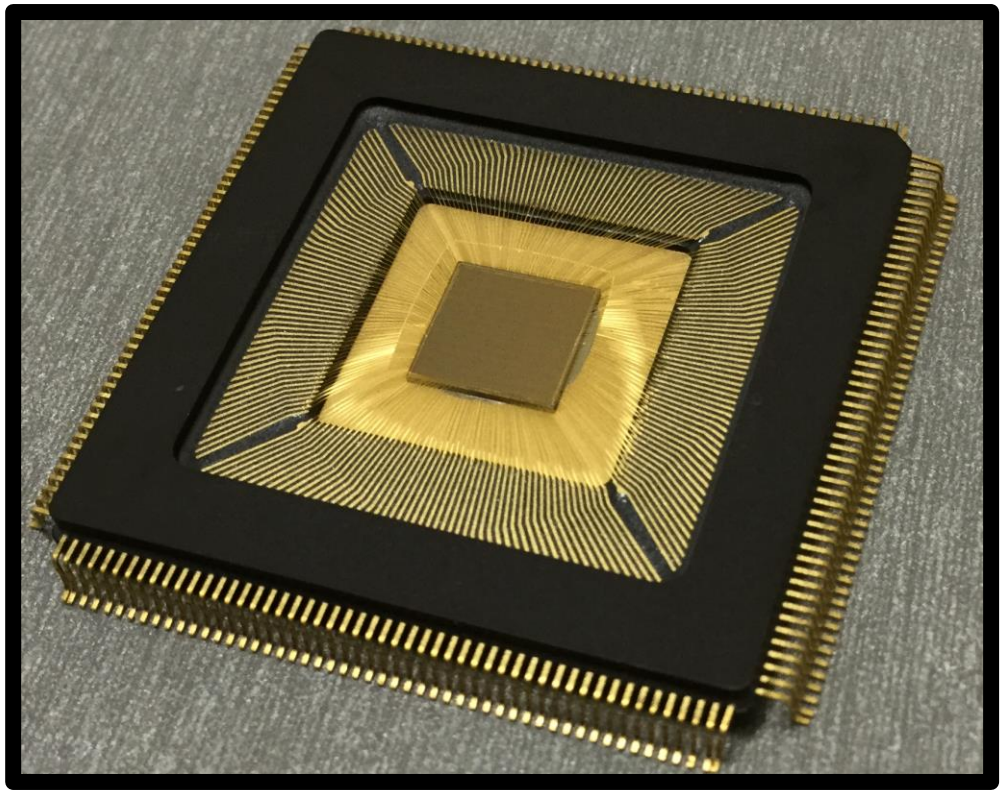
Adversary sees: random paths.

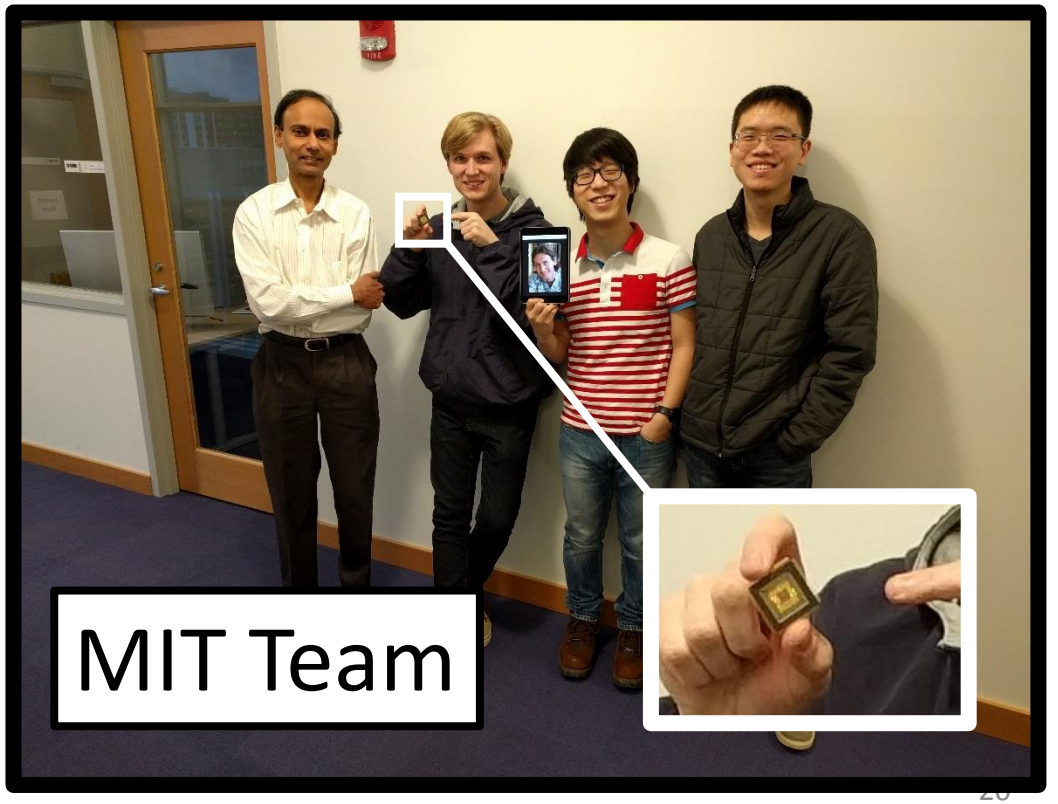
ORAM in Hardware

First ORAM in silicon

Ascend in silicon

- Collaboration with David Wentzlaff's group @ Princeton

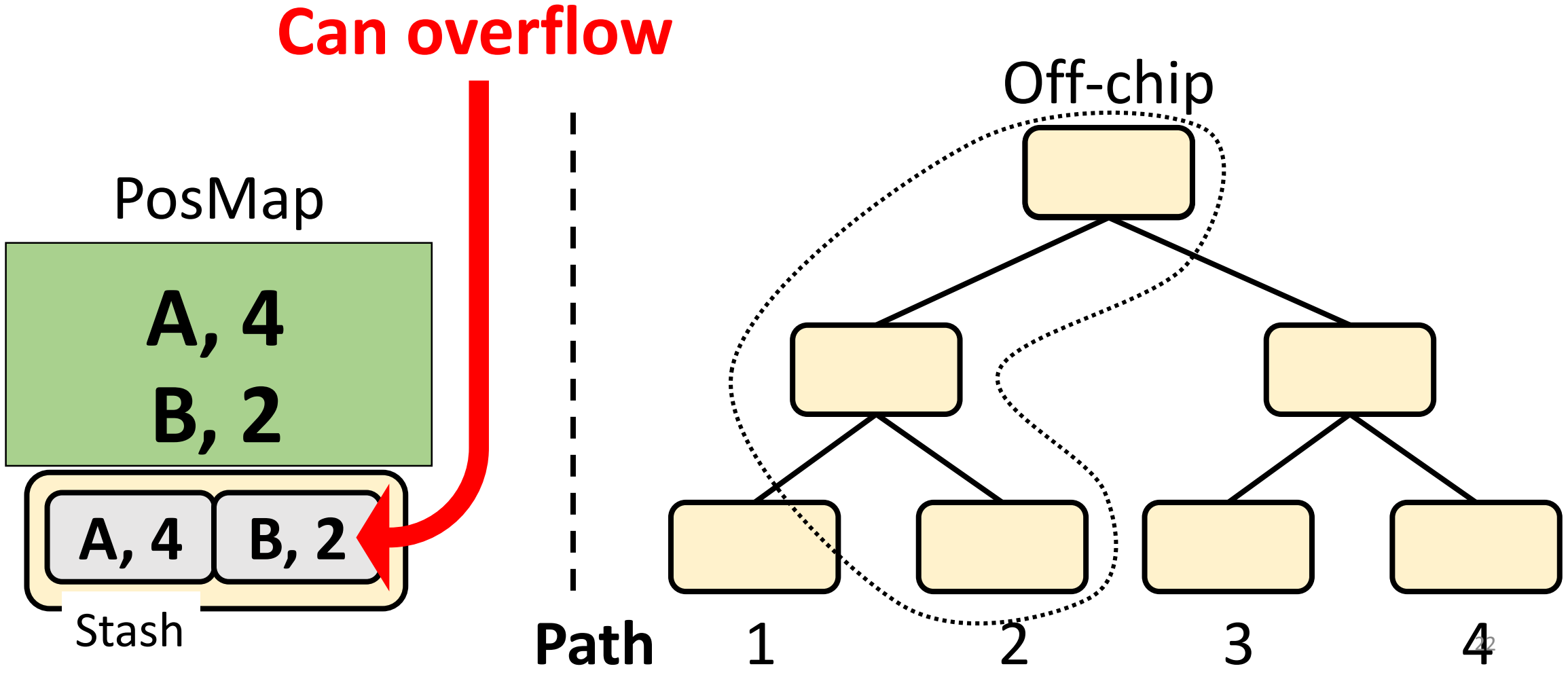




First silicon fully functional
@ 500 MHz & .9 V

Design (Verilog)
Open Source

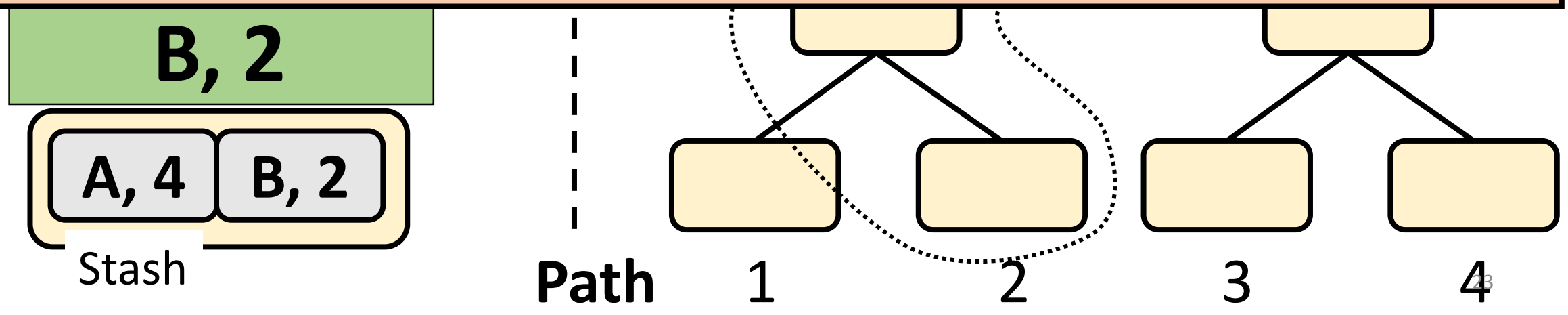
Blocks must live on assigned path or in stash.



Blocks must live on assigned path or in stash.

Off-chip

Bottleneck in prior work [Maas et al. ‘13]
Causes **3 X** avg. slowdown on SPEC.



Bit blasted stash eviction [FCCM'15]

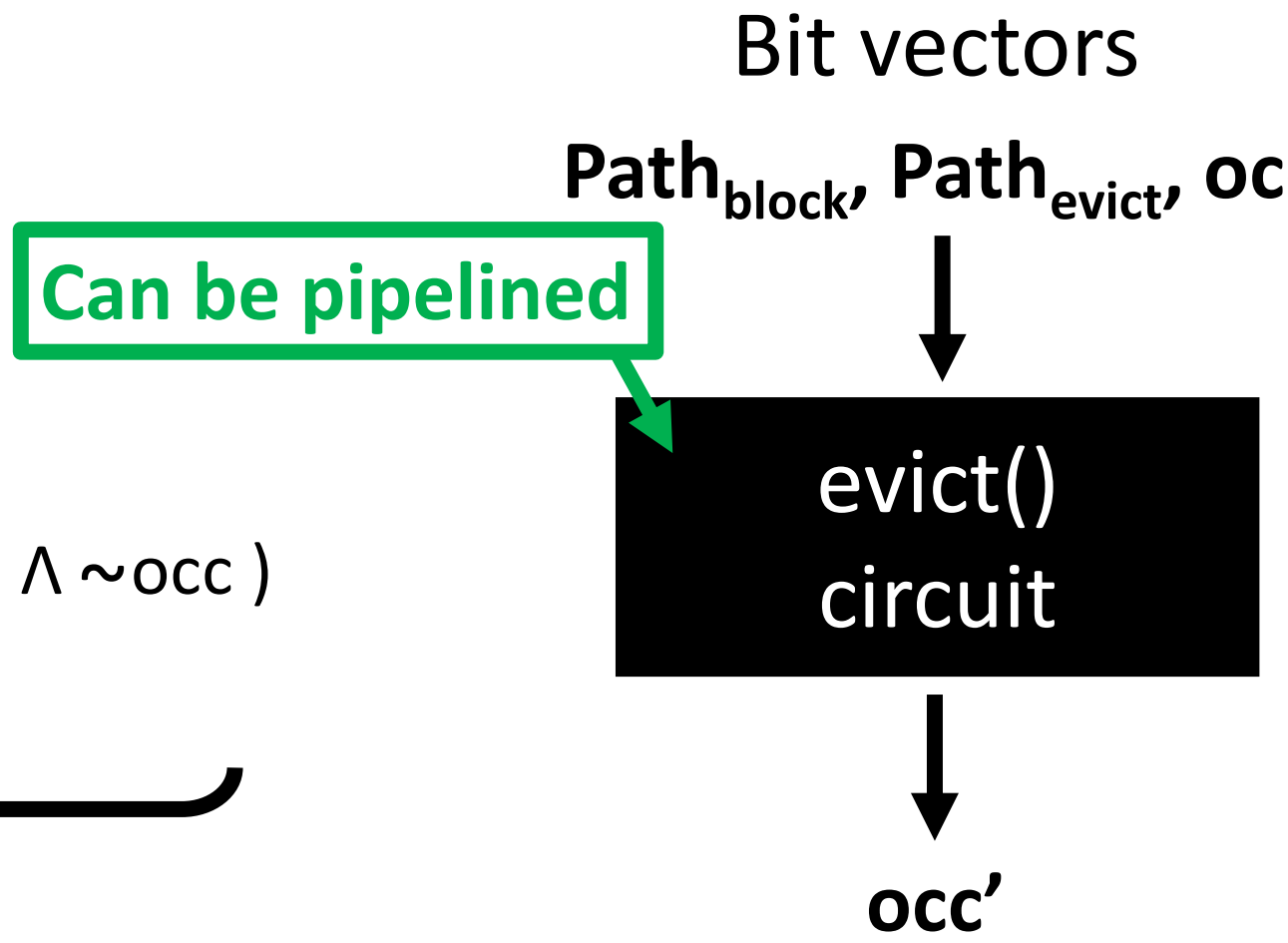
def evict($Path_{block}$, $Path_{evict}$, occ**):**

$t_1 = Path_{block} \oplus Path_{evict}$

$t_2 = bit\_reverse(((t_1 \Lambda - t_1) - 1) \Lambda \sim occ)$

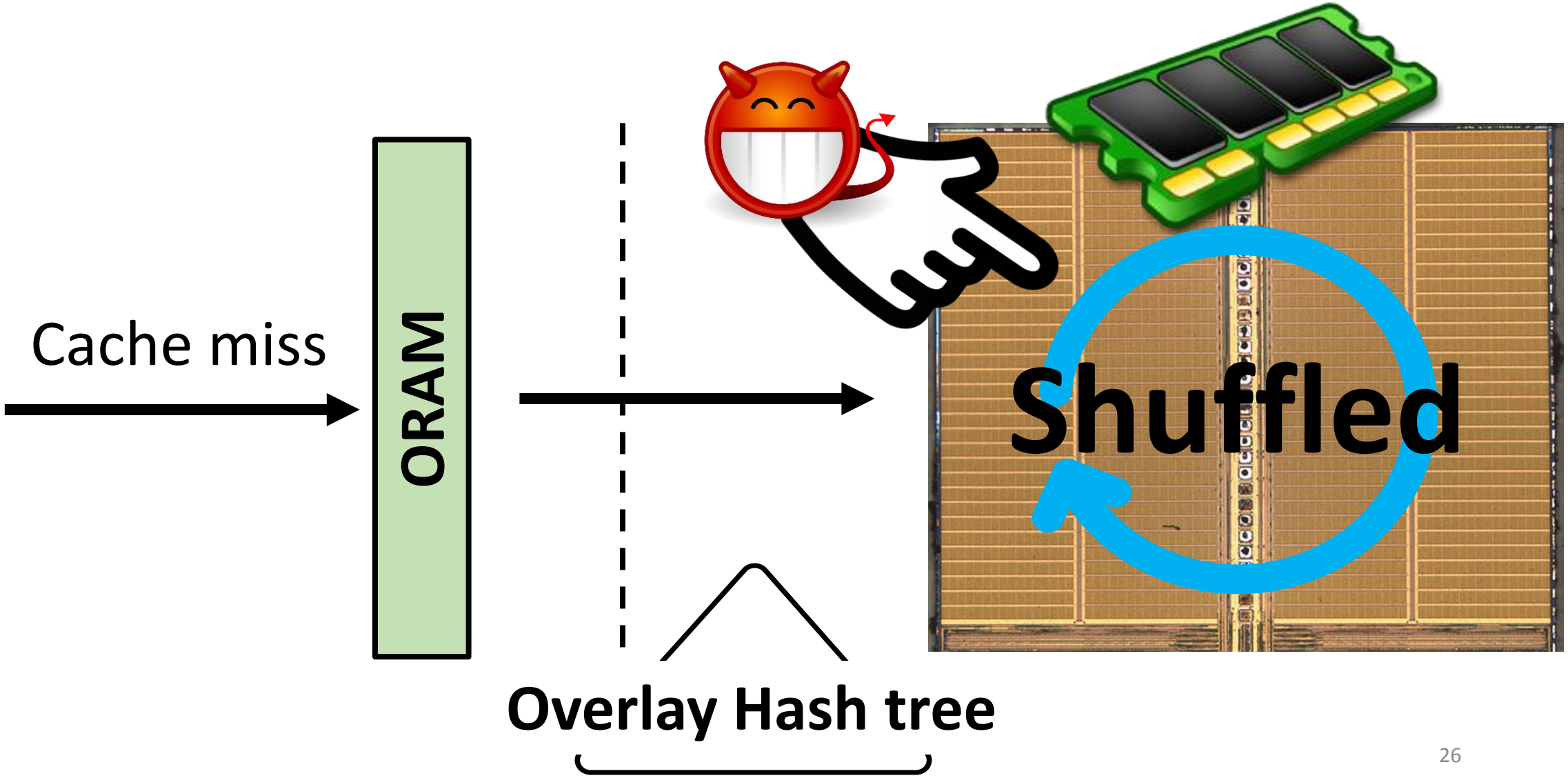
ret $bit\_reverse(t_2 \Lambda - t_2)$

≈ greatest common prefix



Simple design, no performance bottleneck.

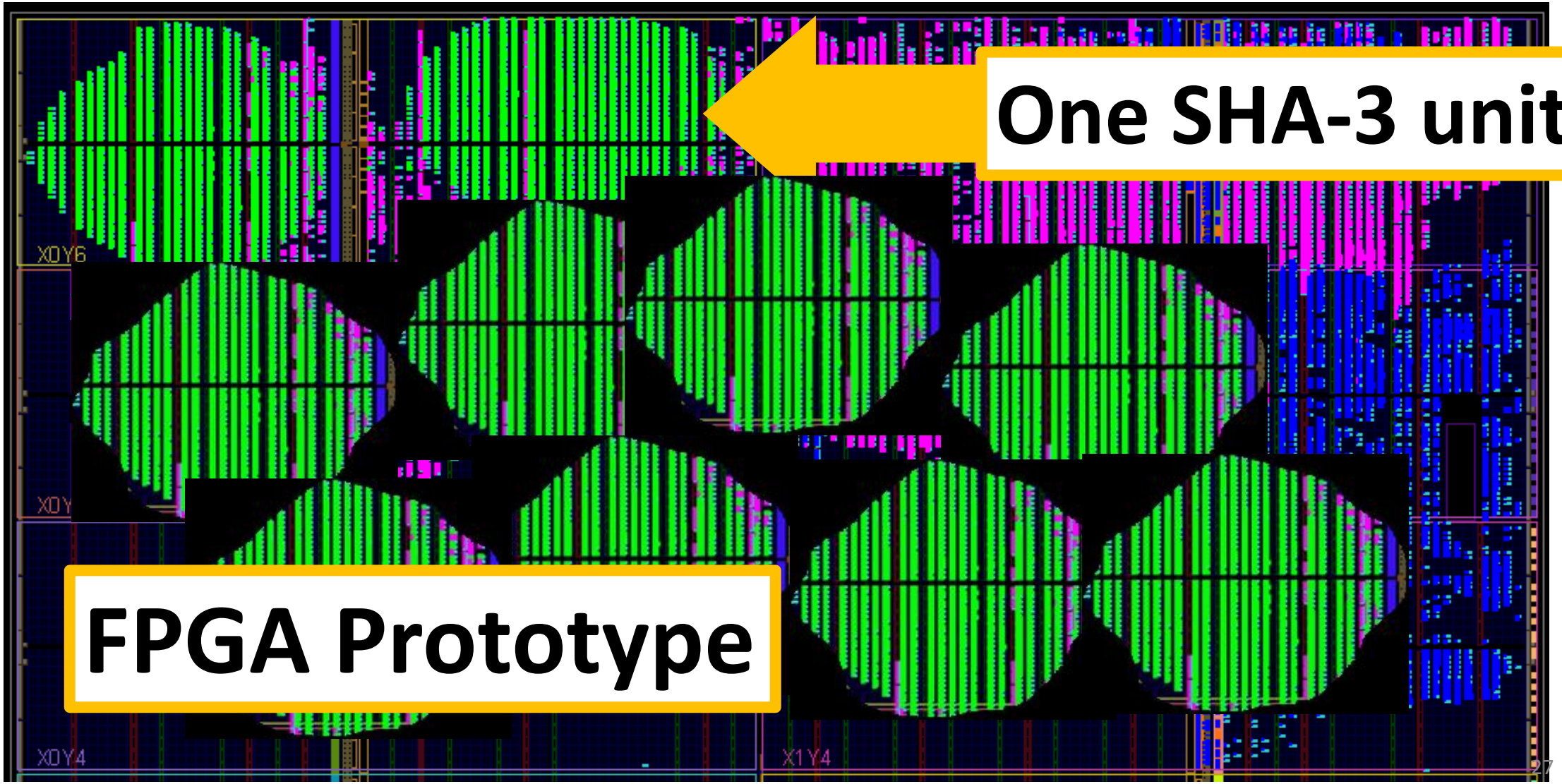
Integrity protection for ORAM



ORAM logic

Test harness

SHA-3



Cheap Integrity Scheme [ASPLOS'15]

- Per-block MAC

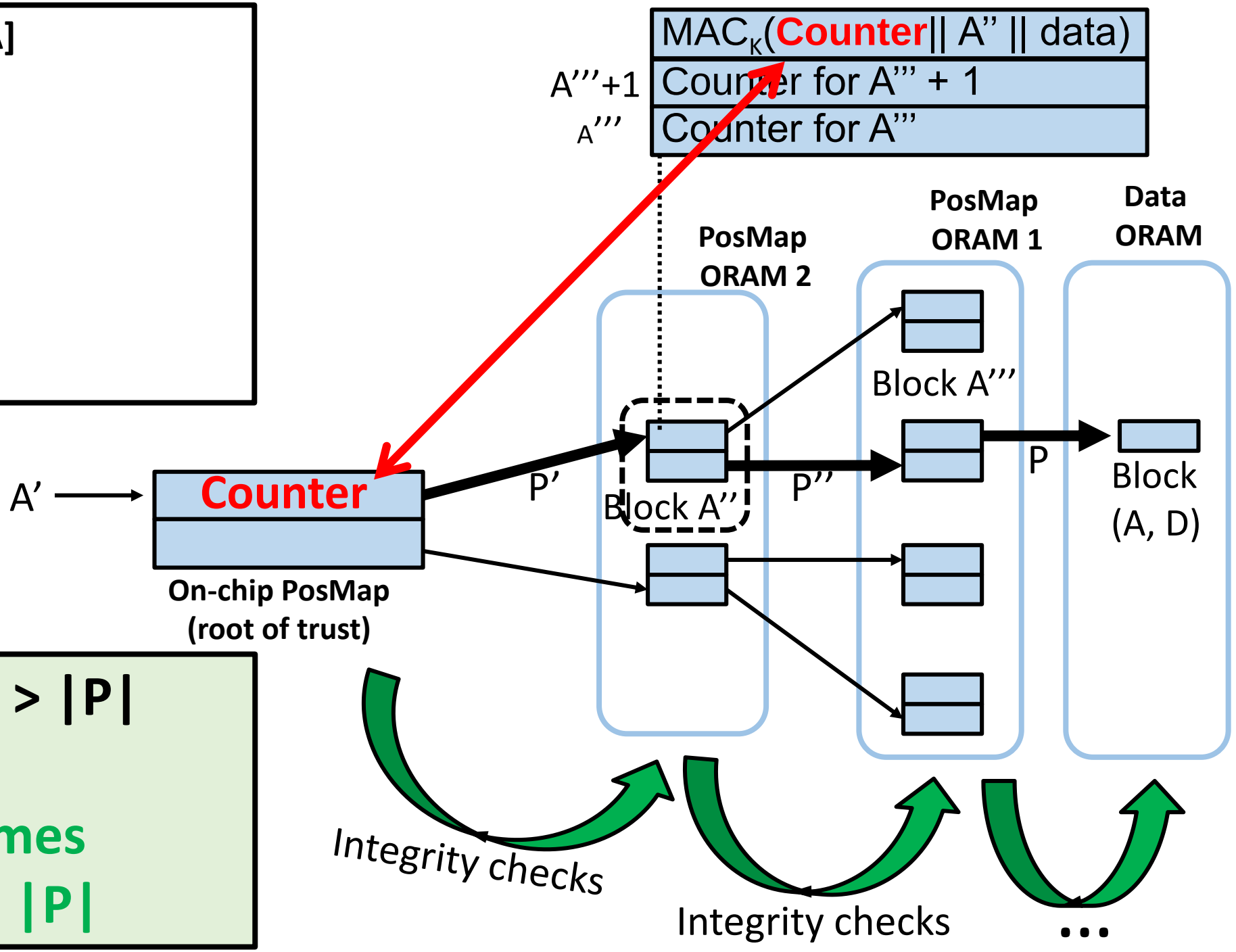
  {**Block data**, **Hash(Block data**, Block addr, counter**)**}

- **Good:** Hash 1 block, **NOT** path
- **Bad:** Need to store counters on-chip

Replace entries in map with counters!

Want: Path P = PosMap[A]

Algorithm:



Problem: |C| > |P|

More schemes to get |C| < |P|

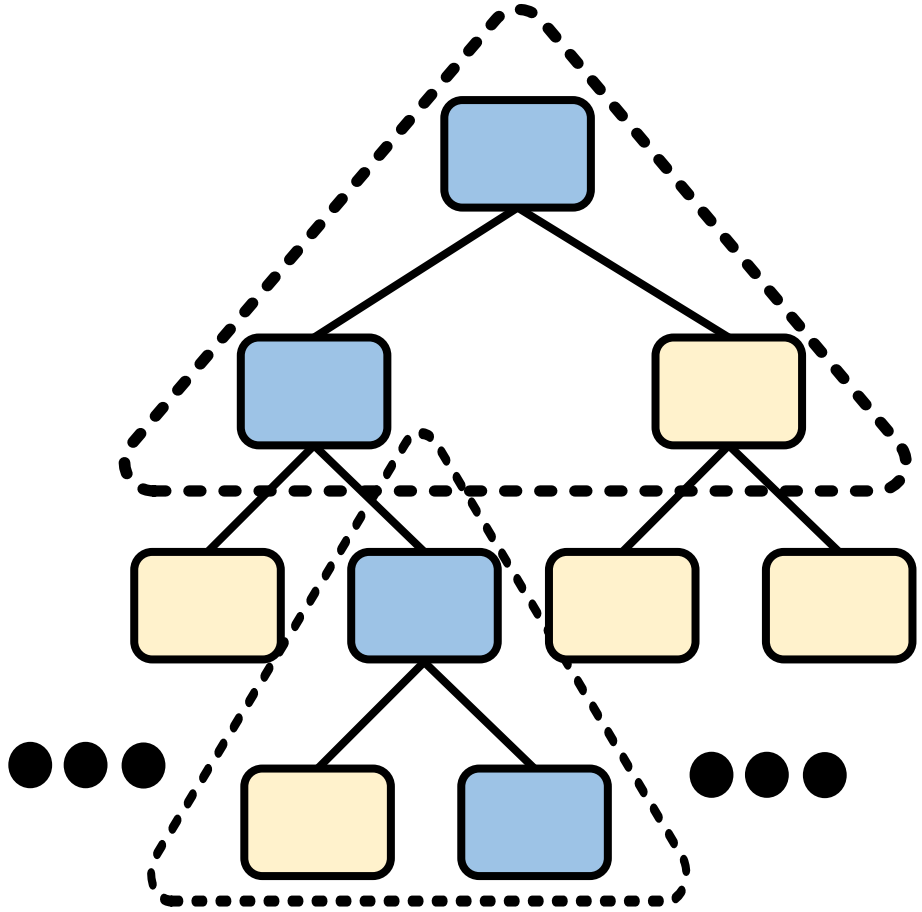
Cheap Integrity Scheme [ASPLOS'15]

Result:

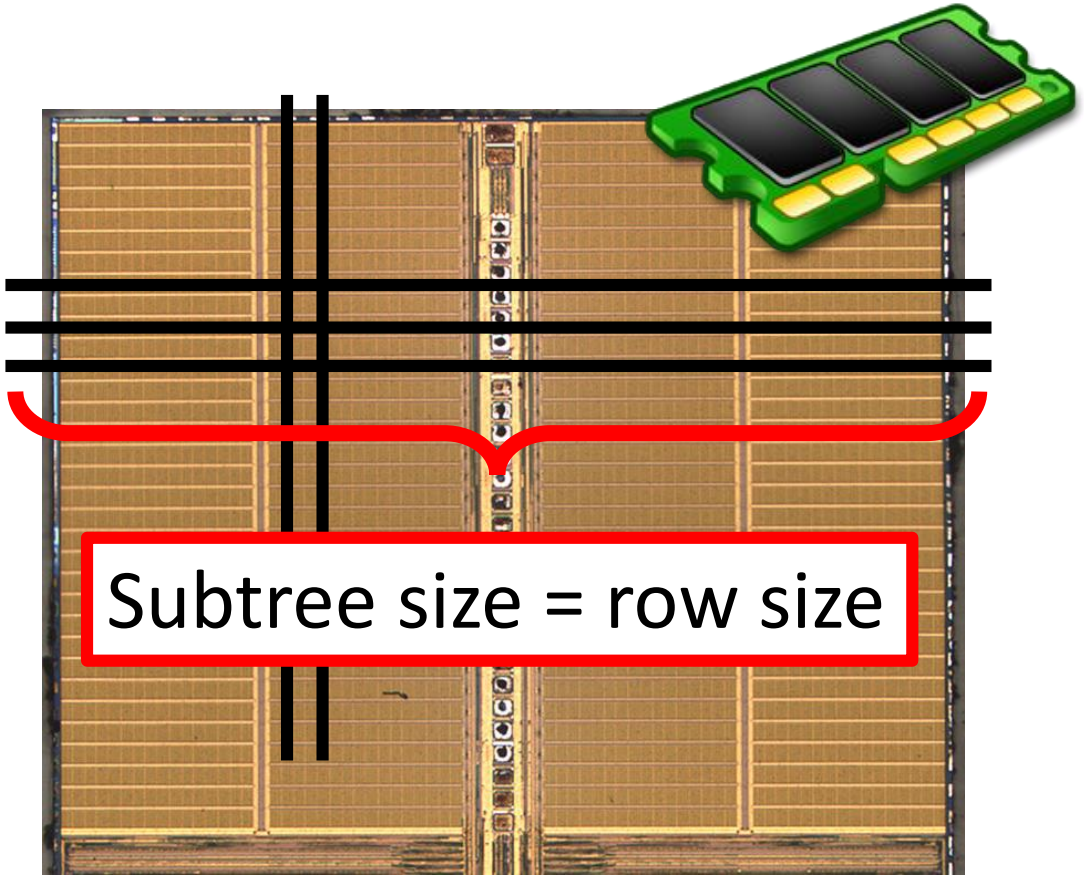
Hashing decreased by **68 X**, simple design

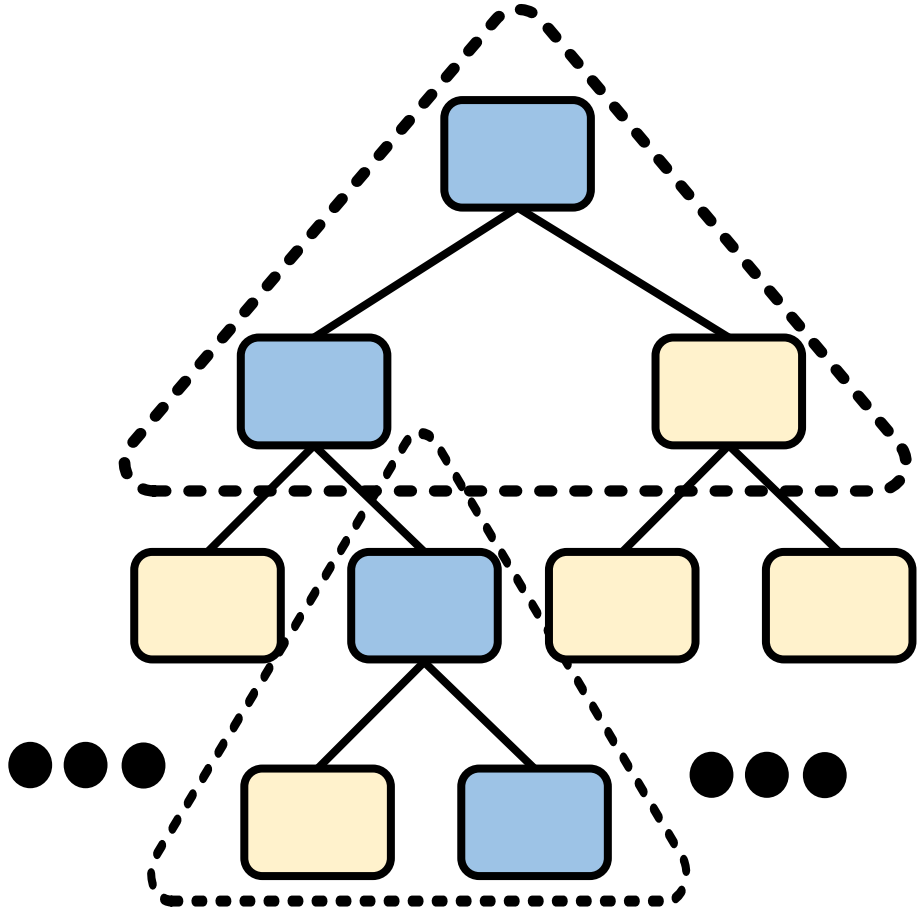
ORAM randomizes data layout.

Computer architecture assumes *data locality.*



ISCA'13





 # Row misses:

$\sim$ tree height

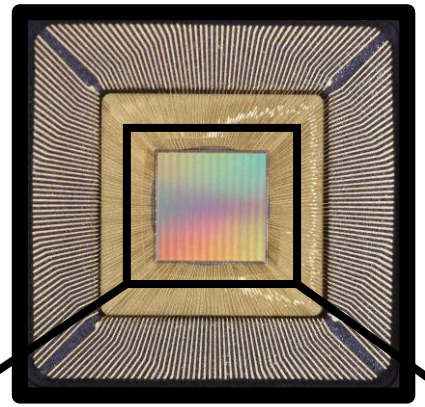
$\Downarrow$

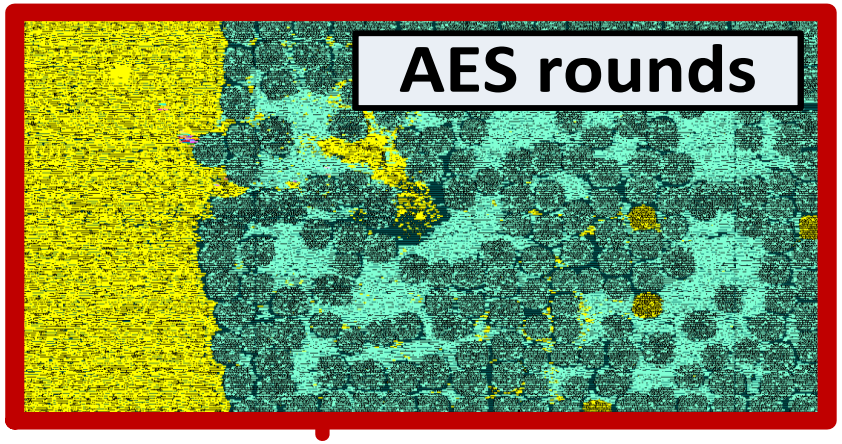
$\sim \dfrac{\text{tree height}}{\text{subtree height}}$

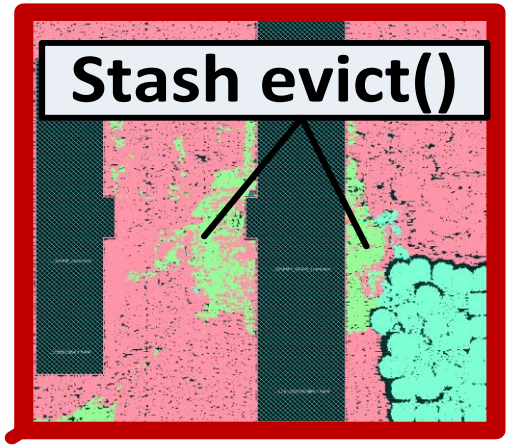
Row misses:

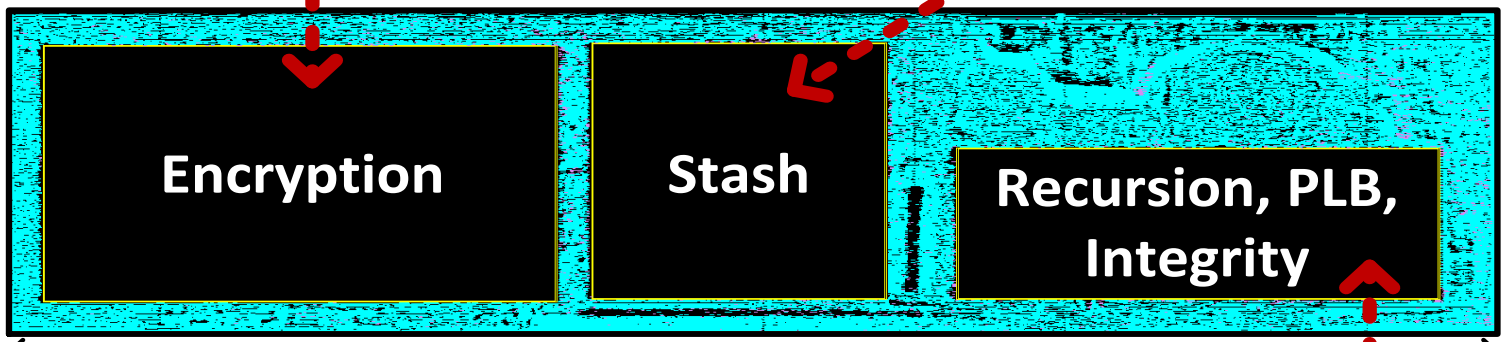
60% overhead ⟹ 13% overhead

460M transistors



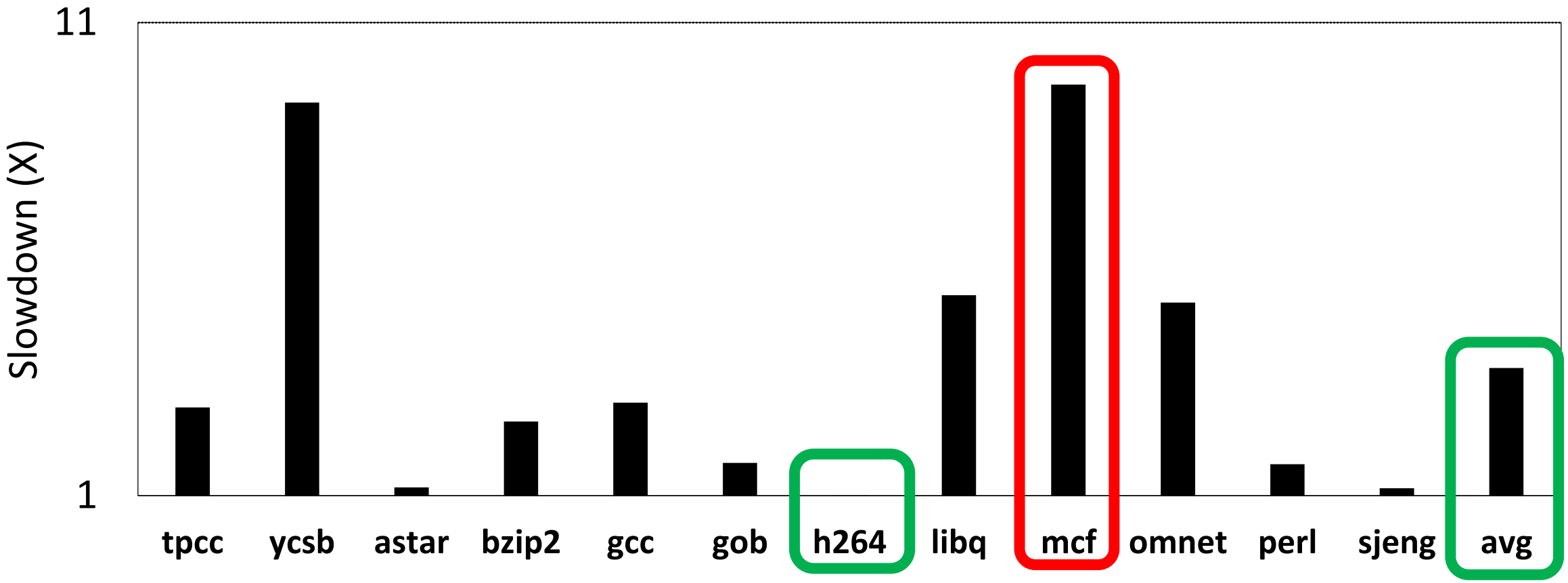






Slowdown vs. insecure

2 DRAM channels, In-order core,
2-level cache hierarchy, 1 MByte last-level cache
ORAM = 1208 cycles / tree lookup



Demo

Backup