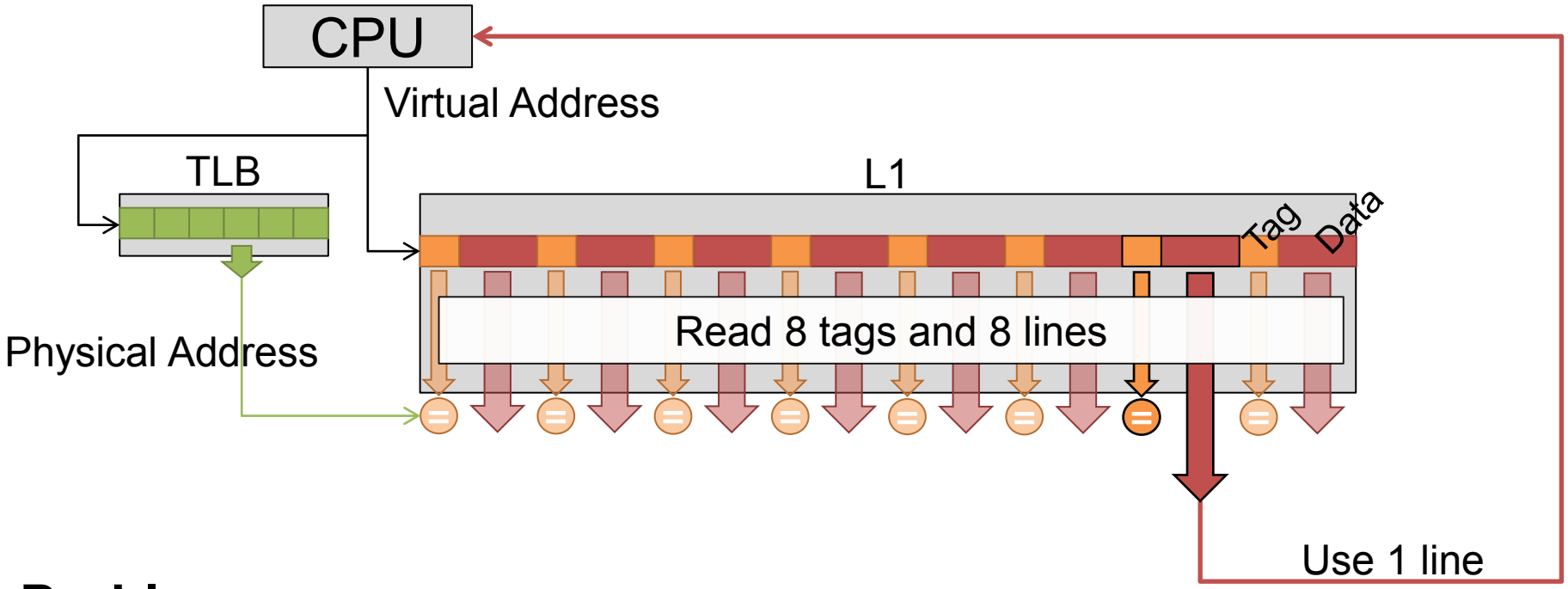
UPPSALA
UNIVERSITET

TLC: A Tag-Less Cache for Reducing Dynamic First Level Cache Energy

Andreas Sembrant, Erik Hagersten, David Black-Schaffer
Uppsala University, Sweden

Motivation: Standard VIPT L1

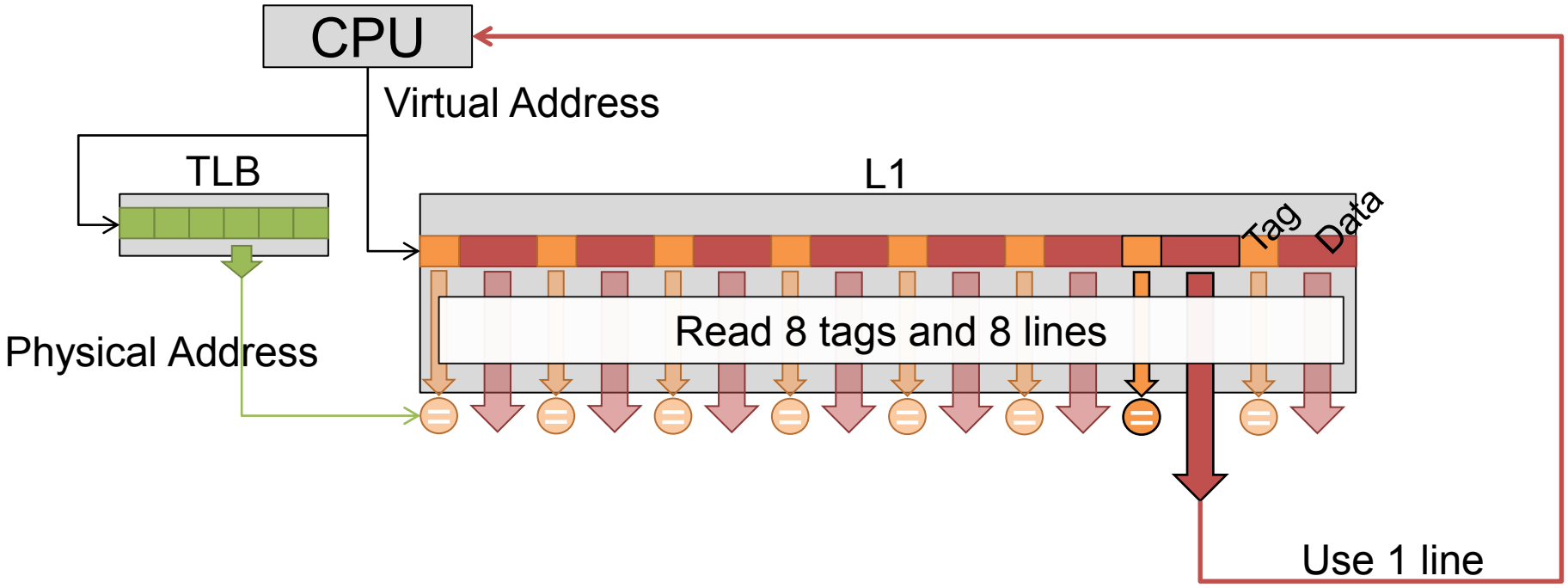


Problems:

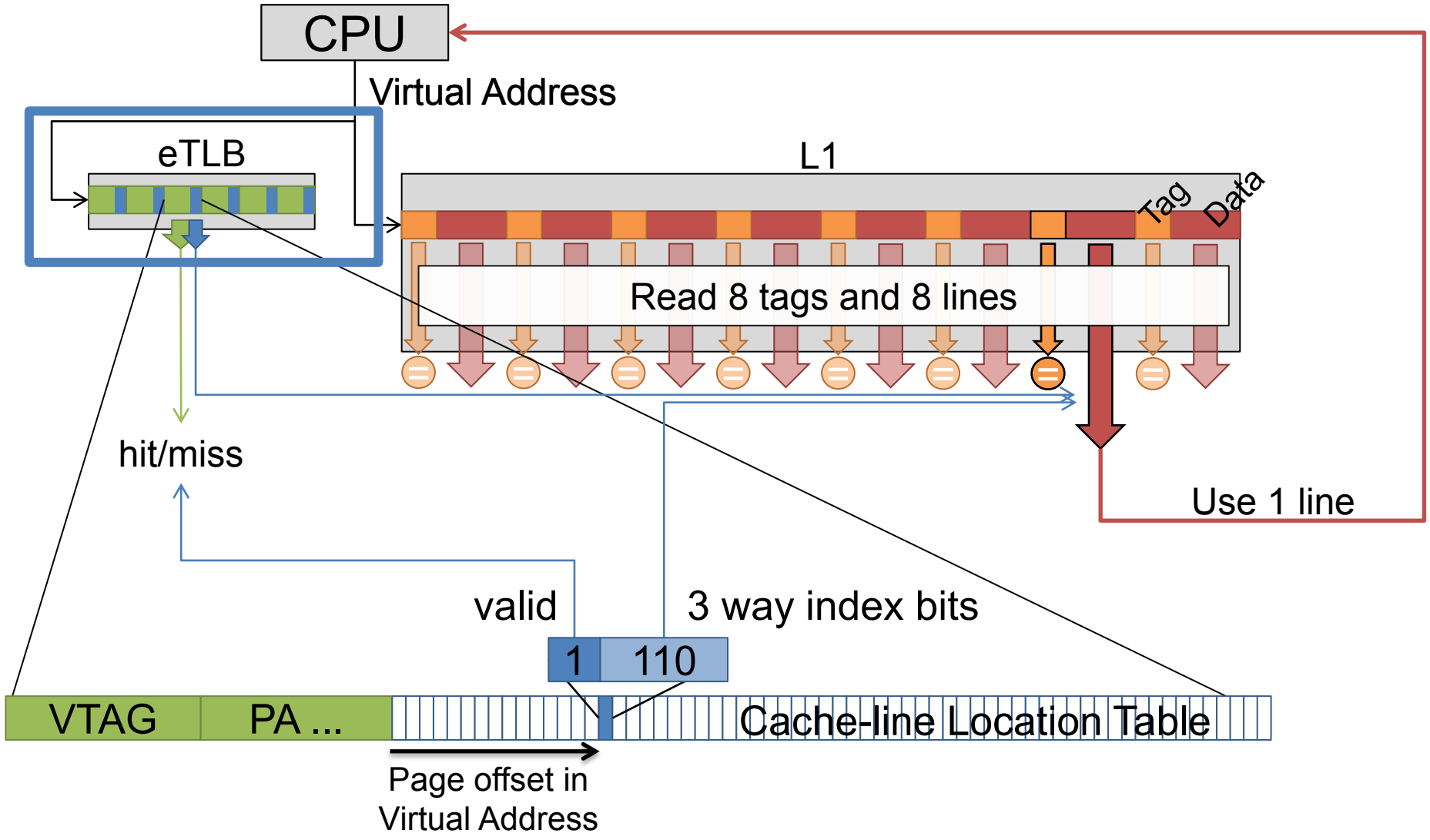
- What do we do?
  - Read the TLB, 8 tags and data from 8 cache lines
- What do we need?
  - Data from one cache line

Can we do better?

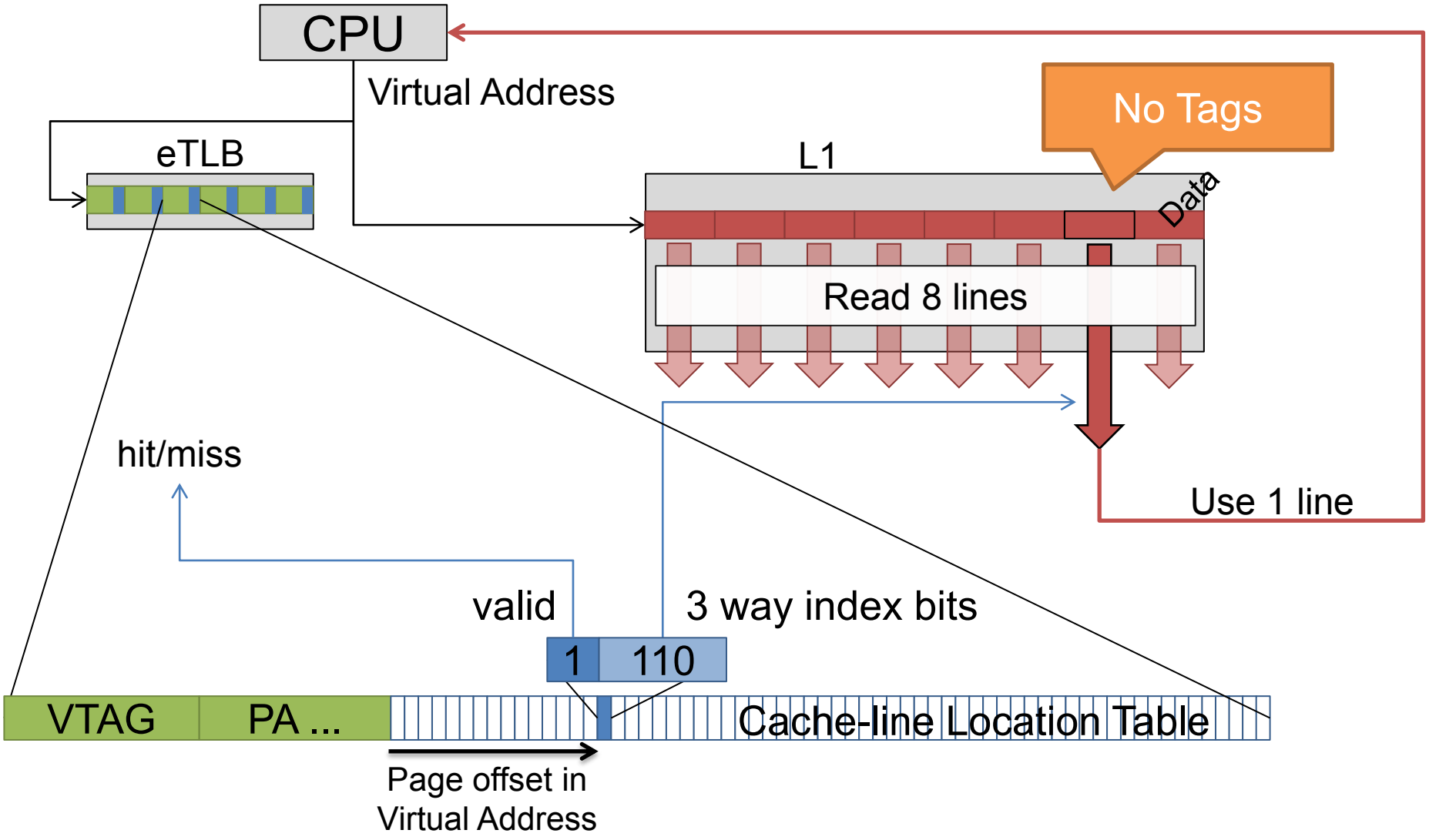
Tag-based → Tag-less Cache



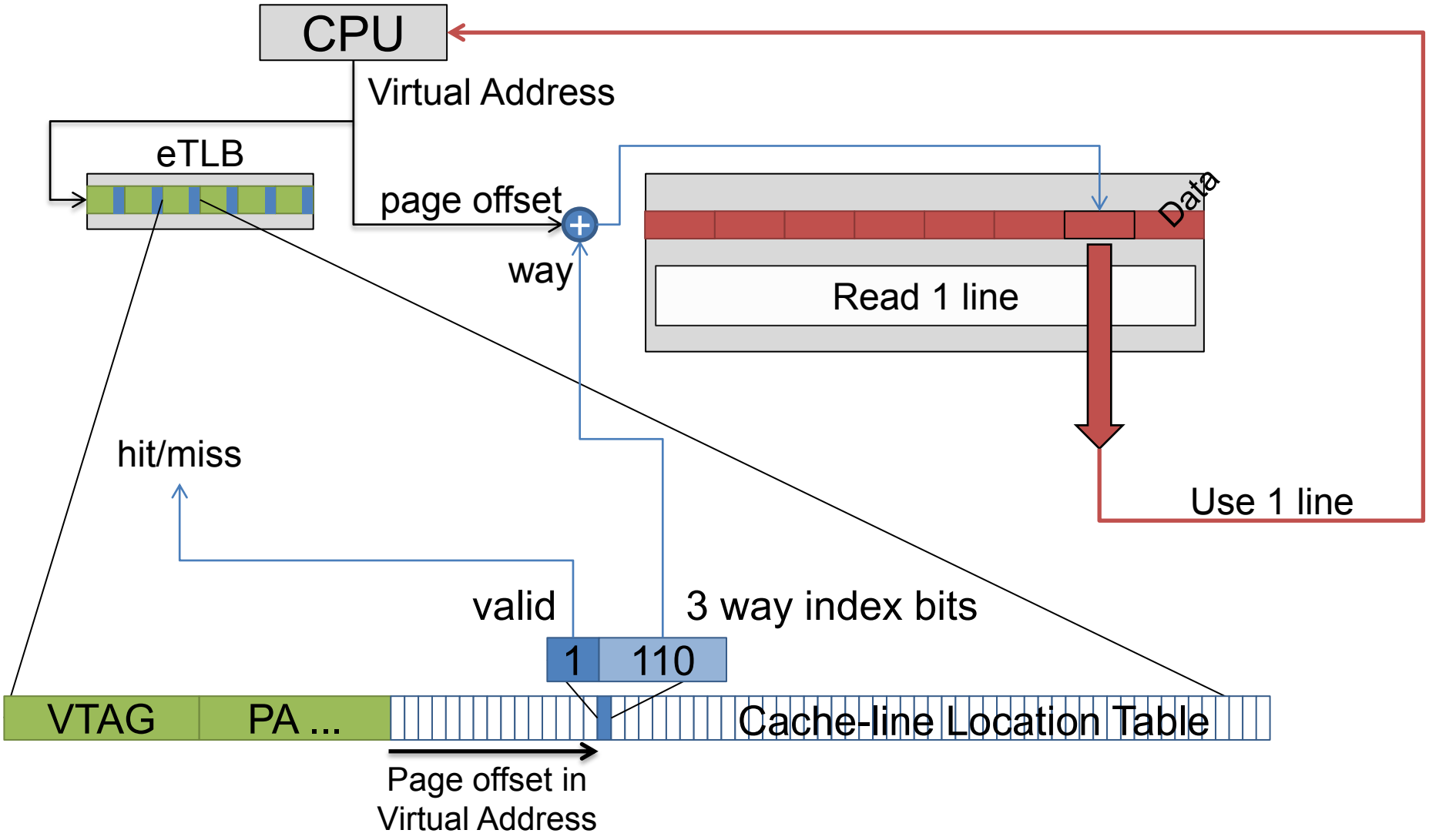
Tag-based → Tag-less Cache



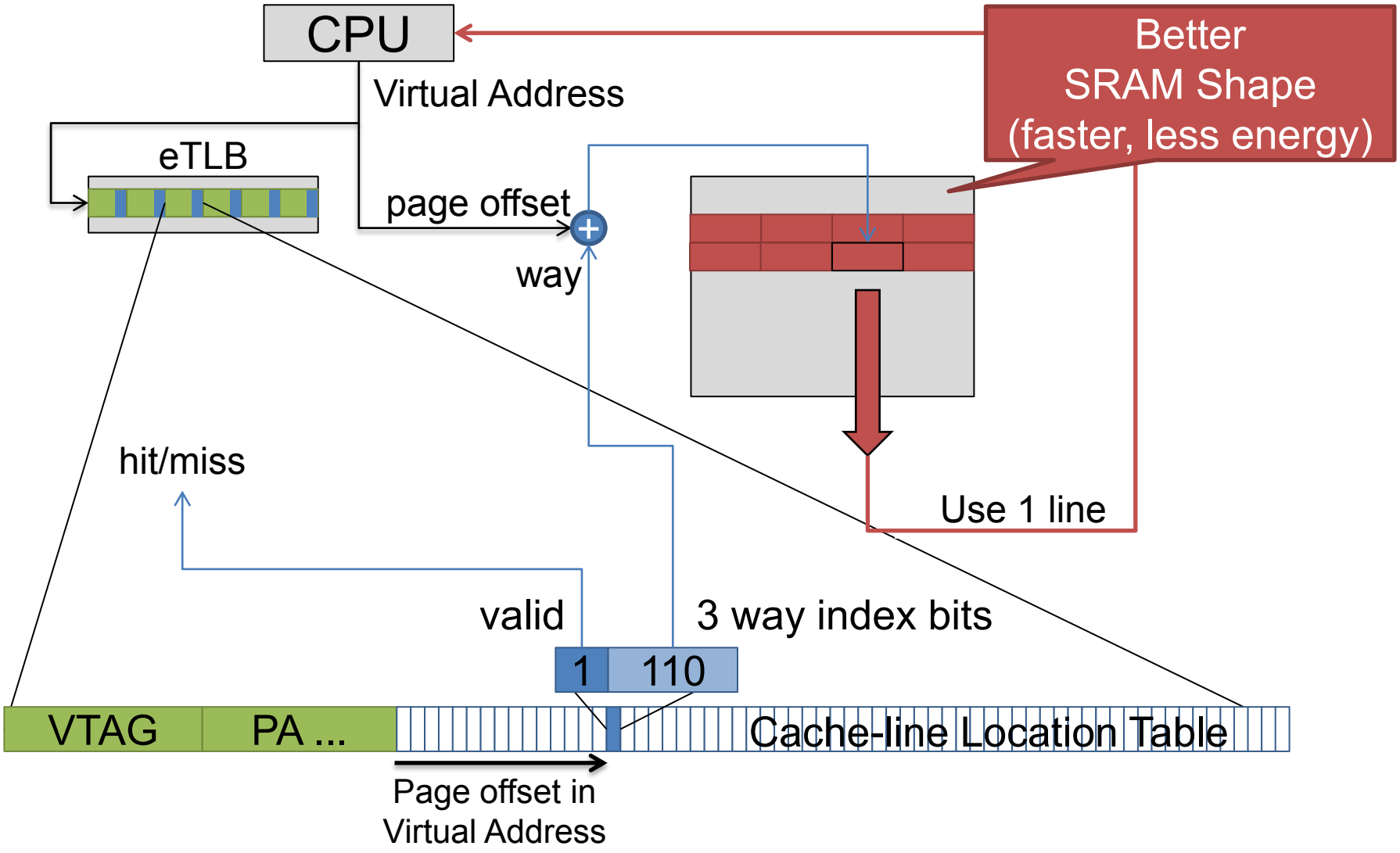
Tag-based → Tag-less Cache



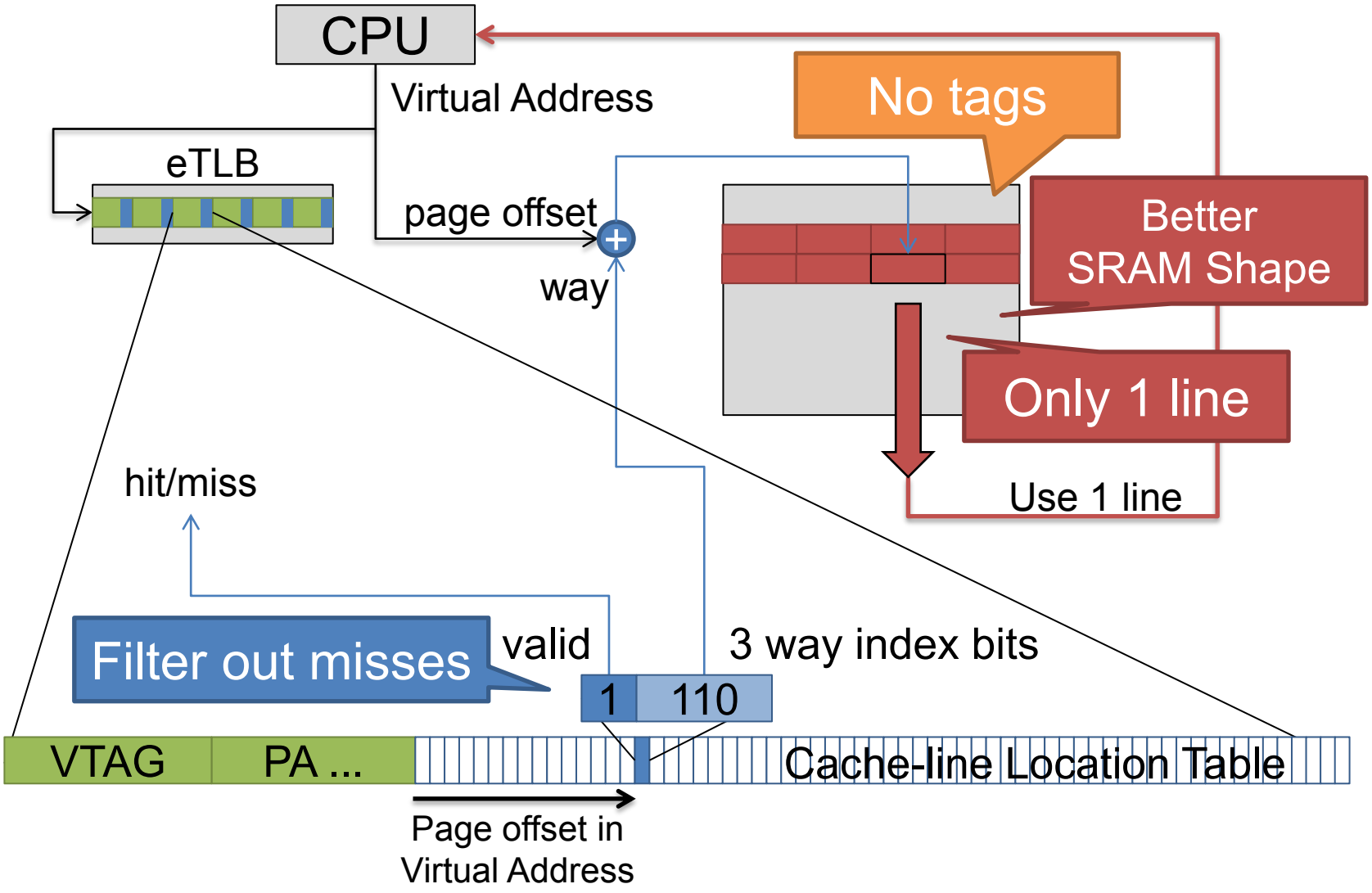
Tag-based → Tag-less Cache



Tag-based → Tag-less Cache



Tag-less Cache



Sanity Check (Bitcount)

L1: 32kB, 8-way **TLB:** 64 entry, 8-way

- **Storage / Area: +1%**
  - \+ Cache line location data in TLB
  - – No cache tags
- **Read hit energy: –64%**
  - – Don’t need the physical address from the TLB on hits
  - – No cache tags
  - – Only read the correct cache line

Slighty more storage (+1%), but many fewer bit reads (-64%)

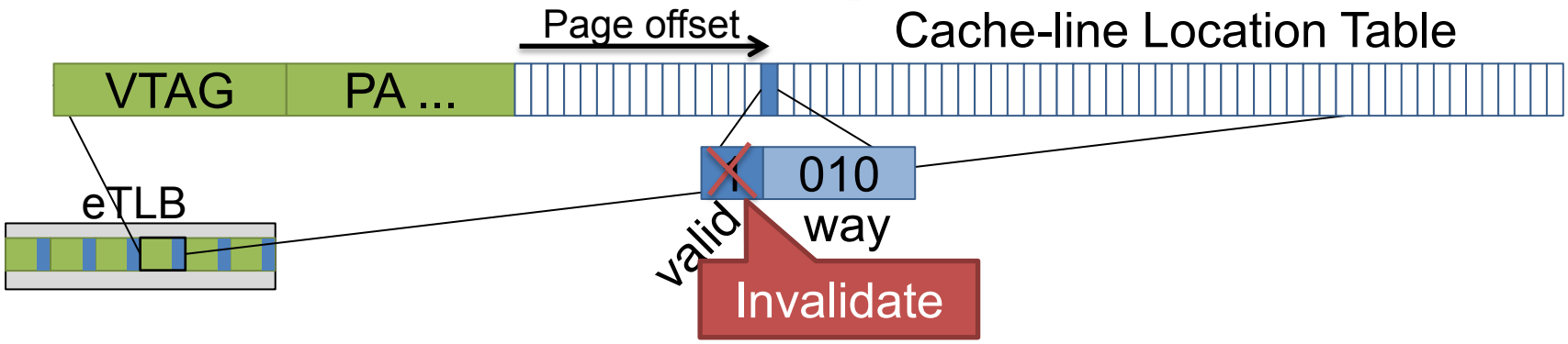
Using an eTLB, we can:

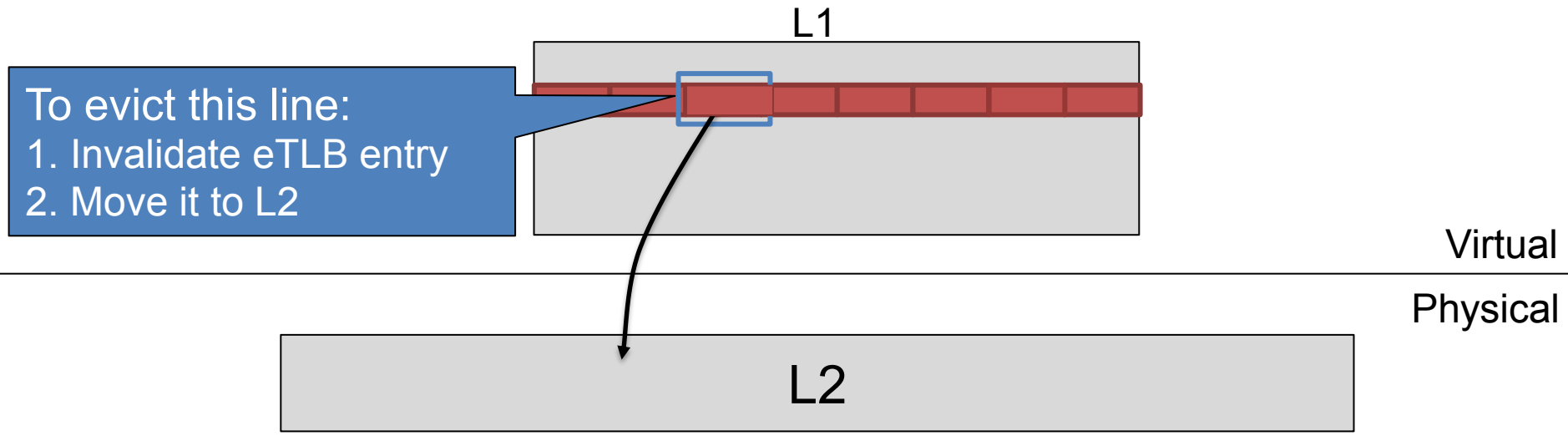
1. **Eliminate extra data-array reads**
   - *by determining the correct way from the TLB*
2. **Eliminate the tag-array**
   - *by avoiding tag comparisons*
3. **Filter out cache misses**
   - *by checking in the eTLB*
4. **Faster data-array SRAM shape**
   - *by only reading one way*

But …

- **We need new ways to handle:**
  1. Cache-line replacement
  2. eTLB replacement

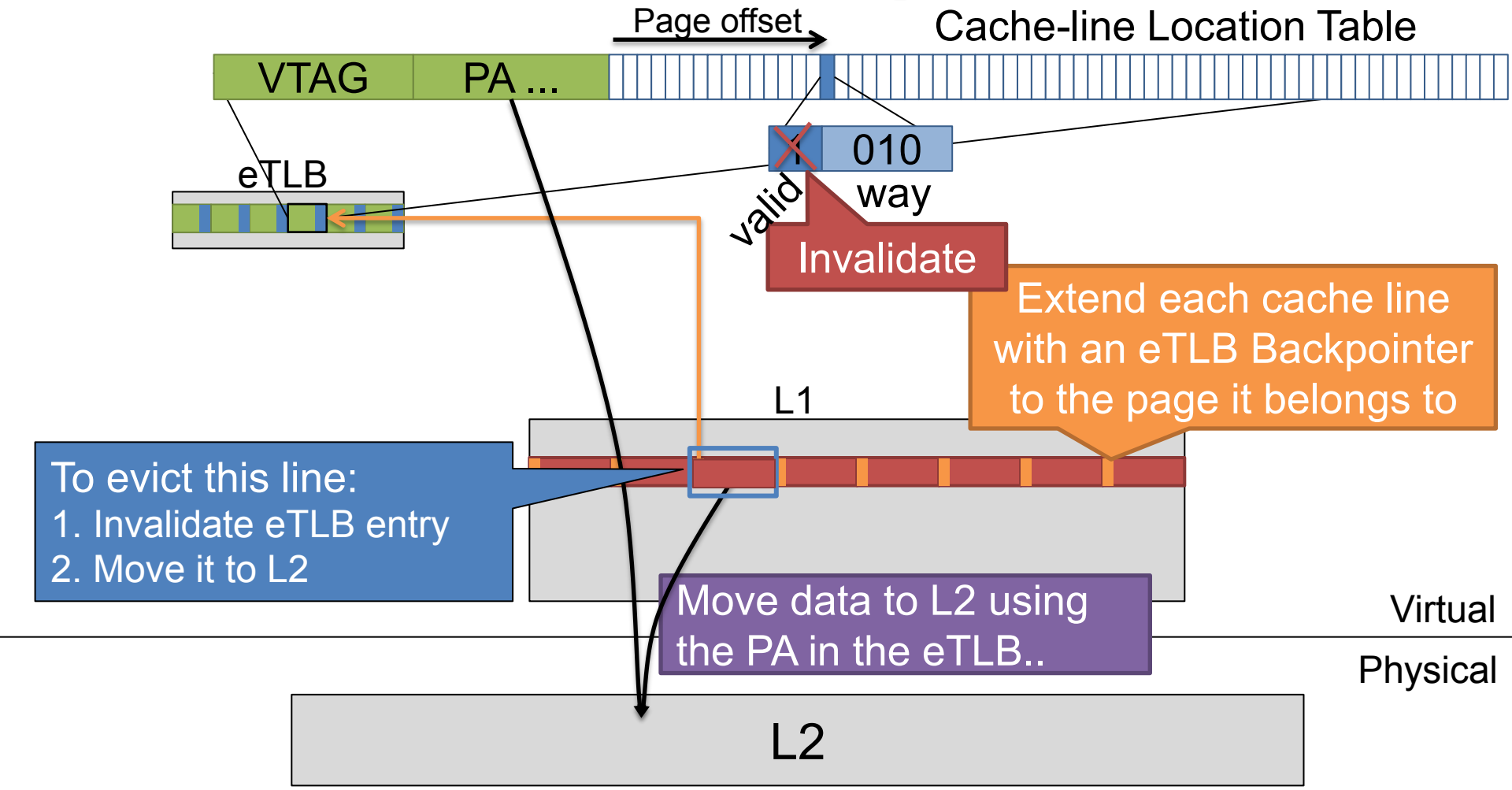
1. Cache-line Replacement





No tags → No way to find the right eTLB entry

1. Cache-line Replacement



1. Backpointer much smaller than a tag (6 vs. ~28bits)
2. Only used during replacement (infrequent)

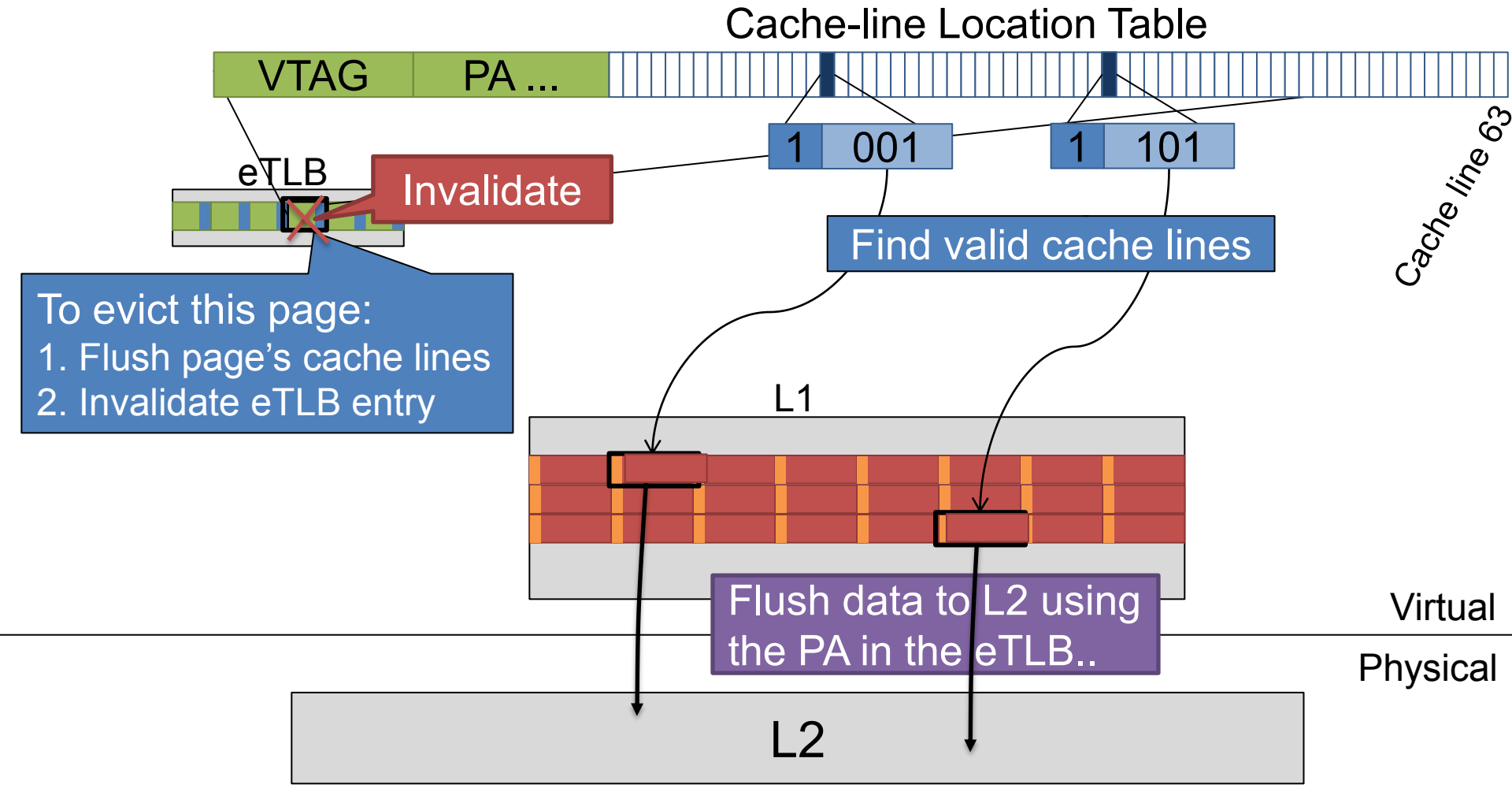
But …

- **To make this work we need to handle:**
  1. Cache-line replacement
  2. eTLB replacement

Problem: The only way to locate a cache line is via the eTLB.
- We can not keep data in the cache without an eTLB entry.

Solution: Need to flush the evicted eTLB entry's cache lines on eTLB replacement.

2. eTLB Replacement



Seems expensive?

This is rare!

1. **We do this rarely (SPEC2006):**
   - L1 miss ratio ~3% (use backpointers)
   - eTLB miss ratio ~0.3% (need to flush lines from a page)

2. **Off critical path (read hit)**

- **Optimize for the common case (hit) 97%**
  - Accesses minimum data on hits
  - Fast on hits
  - Low energy on hits

Improving TLC

1. eTLB Replacement Policy

Minimize forced cache line evictions

2. uPages (Sparse Data)

Minimize area when increasing eTLB entries

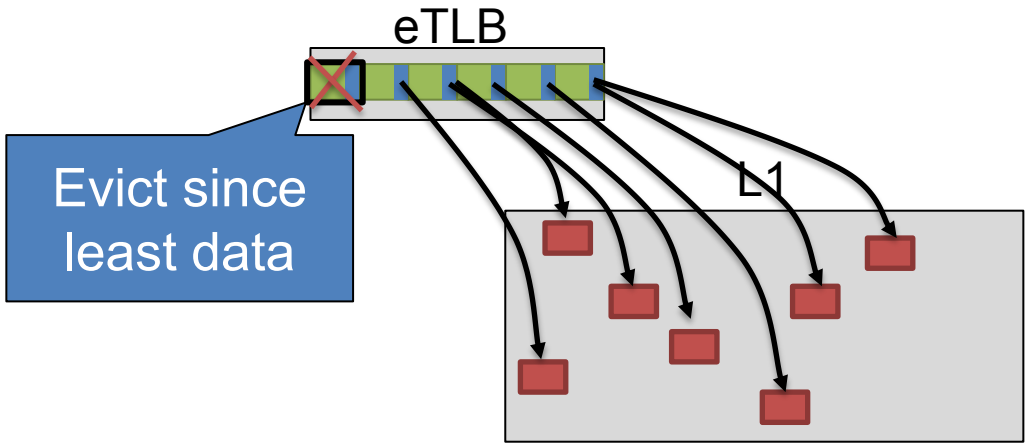
3. uPage Banking

Faster **eTLB → Data-array** communication

1. eTLB Replacement Policy

Problem: We can not keep data in the cache without an eTLB entry.

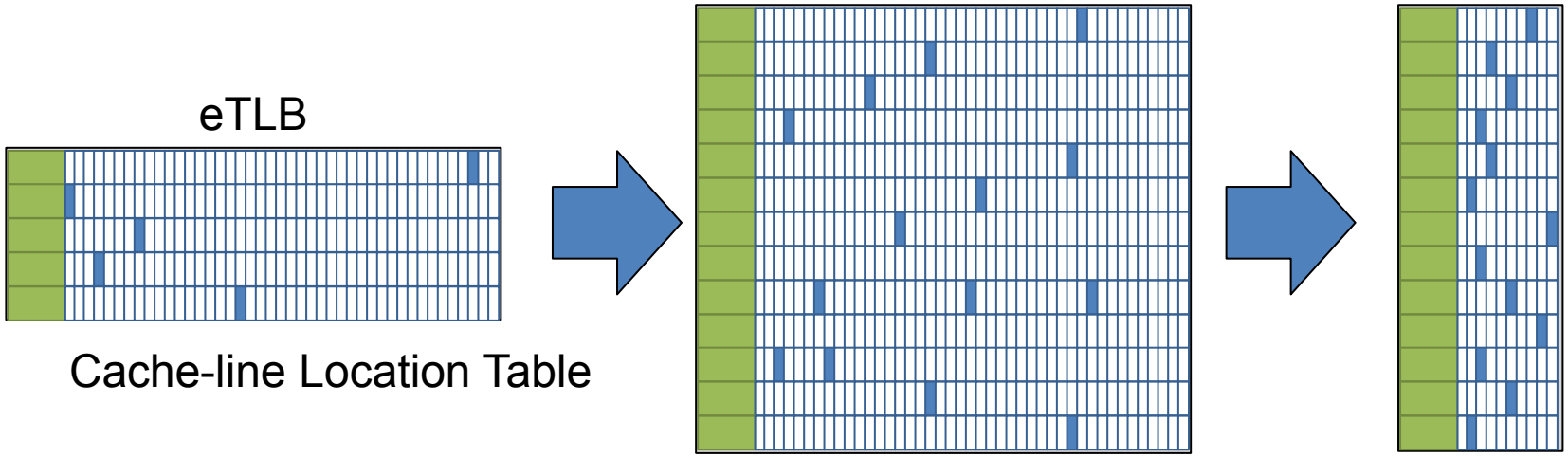
Evict the eTLB entry with least data in the cache.



2. uPages (Sparse Data)

Some applications with sparse access patterns need more eTLB entries

Micro pages (< 4kB)



Sparse data is limited to number of eTLB entries (< # L1 cache lines)

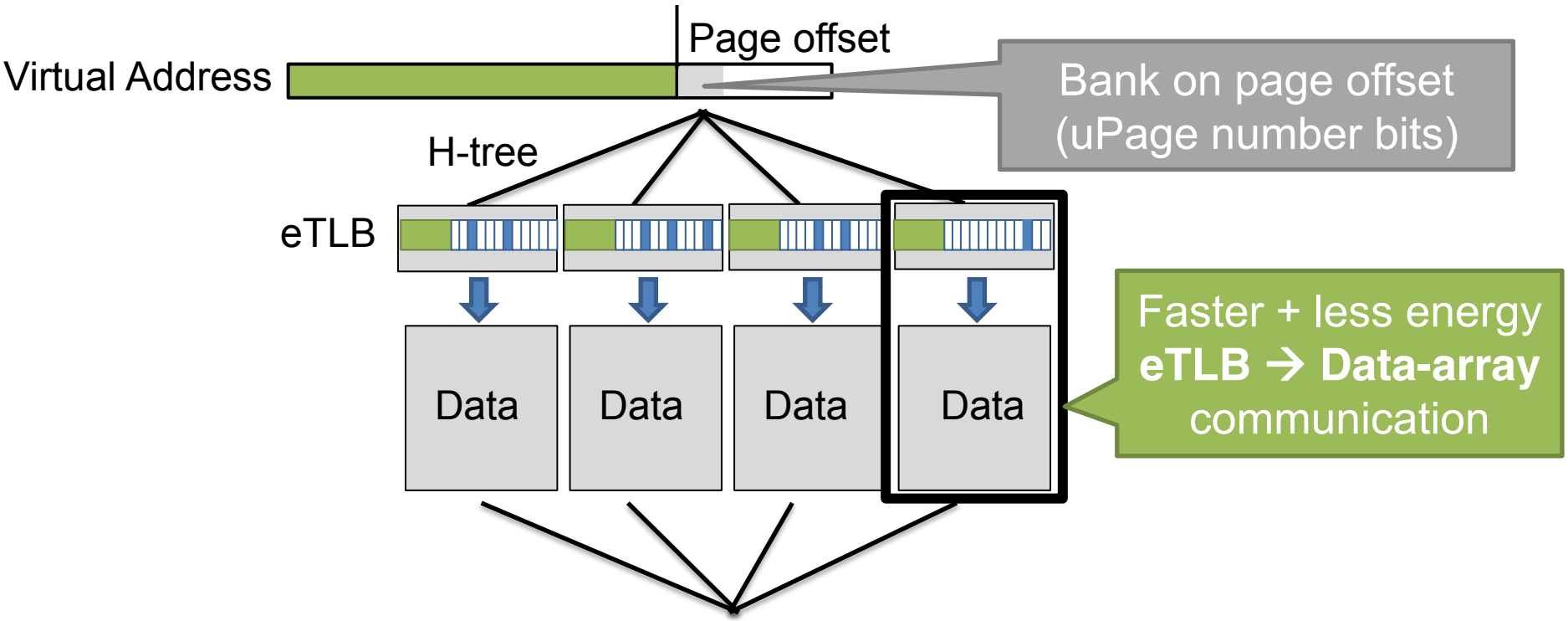
+ More eTLB entries
– Larger area
– Low CLT utilization

+ More eTLB entries
– More tags
+ Smaller CLT area
+ Higher utilization

In paper: Large Pages

3. uPage Banking

Use uPages to improve the eTLB → Data-array communication



Improving TLC

1. eTLB Replacement Policy

Minimize forced cache line evictions

2. uPages (Sparse Data)

Minimize area when increasing eTLB entries

3. uPage Banking

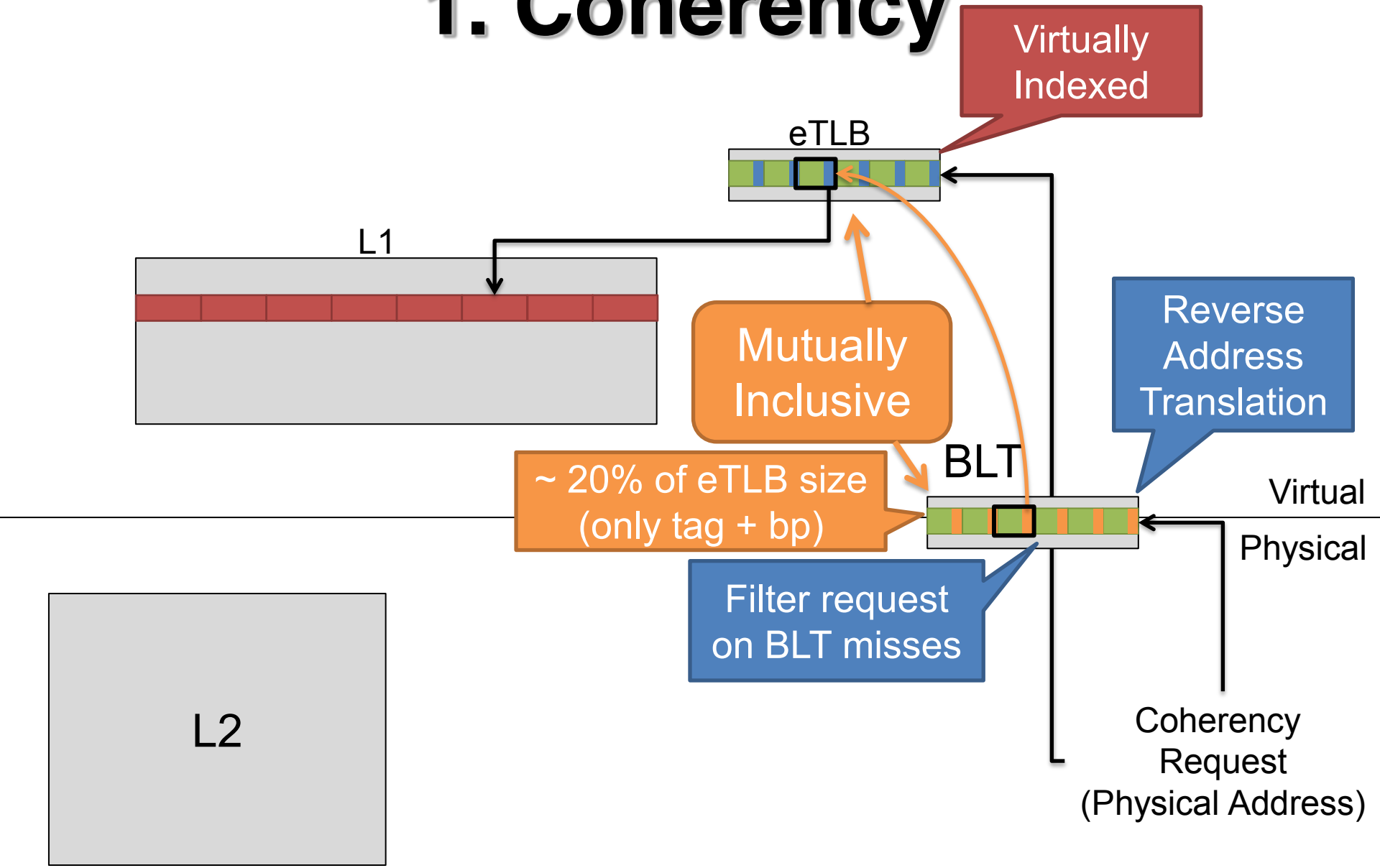
Faster **eTLB → Data-array** communication

Compatibility

- **Drop in replacement to a standard VIPT L1 cache**
  - No software modification
  - No processor core modification

- **Need to handle:**
  1. Coherency
  2. Synonyms

1. Coherency

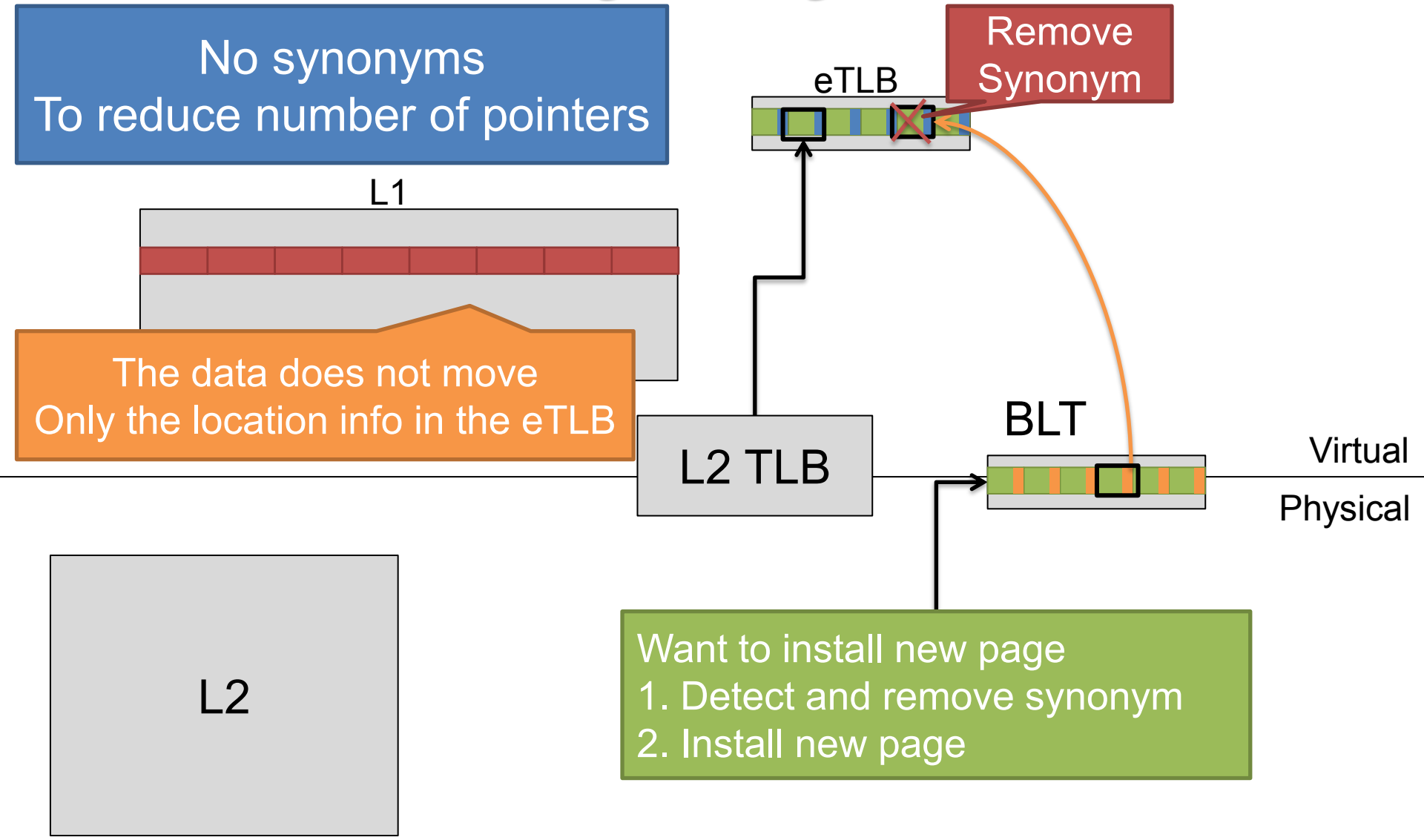

Virtually Indexed
eTLB
L1
Mutually Inclusive
Reverse Address Translation
~ 20% of eTLB size (only tag + bp)
BLT
Virtual
Physical
Filter request on BLT misses
L2
Coherency Request (Physical Address)

Compatibility

- **Drop in replacement to a standard VIPT L1 cache**
  - No software modification
  - No processor core modification

- **Need to handle:**
  1. Coherency
  2. Synonyms

2. Synonyms


No synonyms
To reduce number of pointers
Remove Synonym
eTLB
L1
The data does not move
Only the location info in the eTLB
L2 TLB
BLT
Virtual
Physical
L2
Want to install new page
1. Detect and remove synonym
2. Install new page

Compatibility

- **Drop in replacement to a standard VIPT L1 cache**
  - No software modification
  - No processor core modification

- **Need to handle:**
  1. Coherency (BLT to handle physical → virtual translation)
  2. Synonyms (No synonyms to reduce number of pointers)

Sanity Check 2 (Bitcount)

L1: 32kB, 8-way **TLB:** 64 entry, 8-way

- **Storage / Area:**
  - + Cache line location data in TLB
  - – No cache tags

	+1%
+ eTLB backpointers	+1%
+ BLT	+0.6%
	+2.6%
+ (64 → 256 entry 1kB uPage) eTLB	+8.4%
	+11%

Trade-off between area and optimizing for sparse data

- **Read hit energy: –64%**
  - – Don't need the physical address from the TLB on hits
  - – No change (Backpointer + BLT is off read hit path)
  - – Only read the correct cache line

More area (2.6–11%), but same energy reduction (-64%)

Access Time and Read Energy

- **CACTI 6.5 cache model**

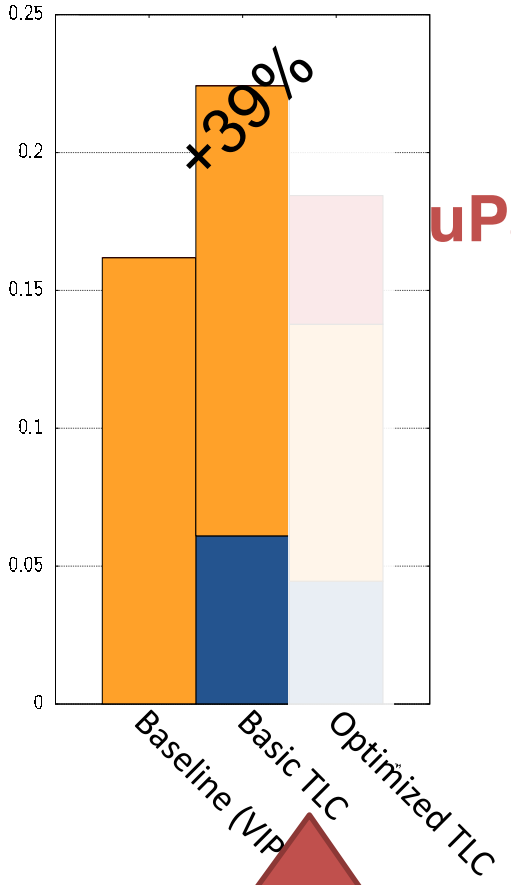
- **Here: 3 configurations**
  - Baseline (Standard VIPT cache)
    - 64 entry TLB
  - Basic TLC
    - 64 entry eTLB
  - Optimized TLC
    - Smart data-aware replacement, µPages, µPage-banking
    - 512 entry uPage eTLB

- **32kB L1 cache**

More configurations in the paper

Results: Read Hit

Access Time (ns)

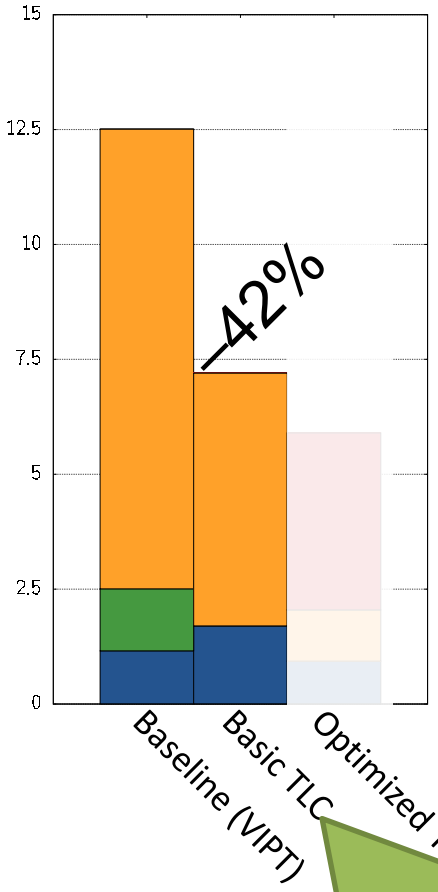


uPage-bank
H-Tree

Data

Tags
eTLB

Dynamic read energy (pJ)

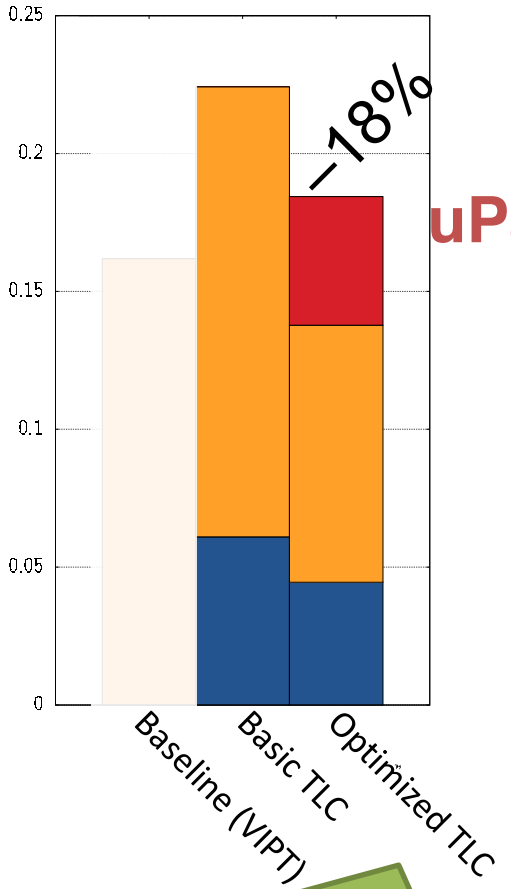


Slower due to eTLB

Less energy since no tags and only reading the correct way.

Results: Read Hit

Access Time (ns)

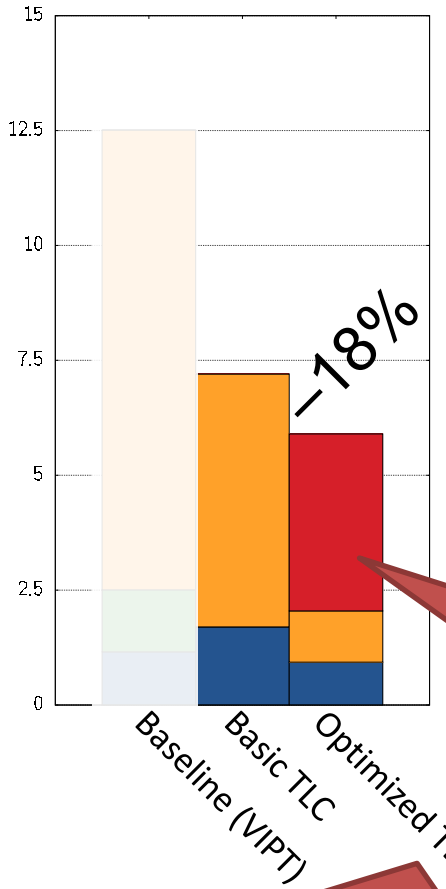


uPage-bank
H-Tree

Data

Tags
eTLB

Dynamic read energy (pJ)



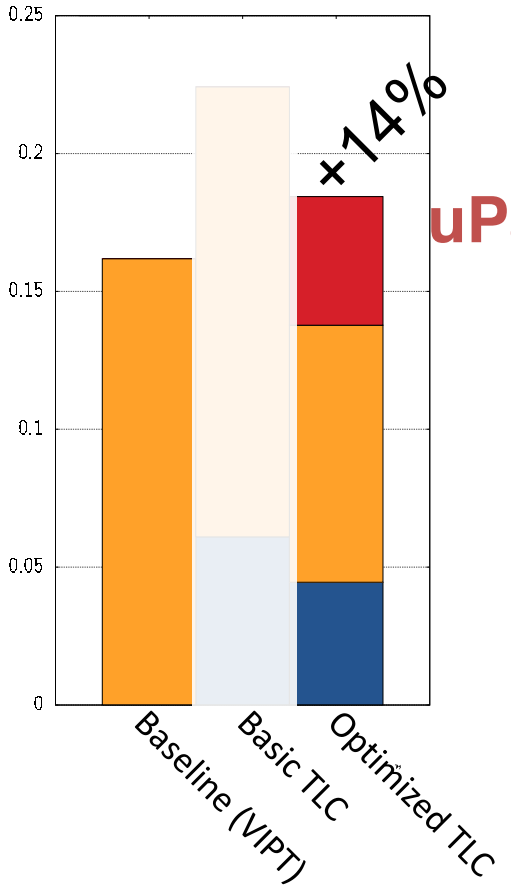
65% of the energy

Faster due to uPage-banking

But significant H-tree energy

Results: Read Hit

Access Time (ns)

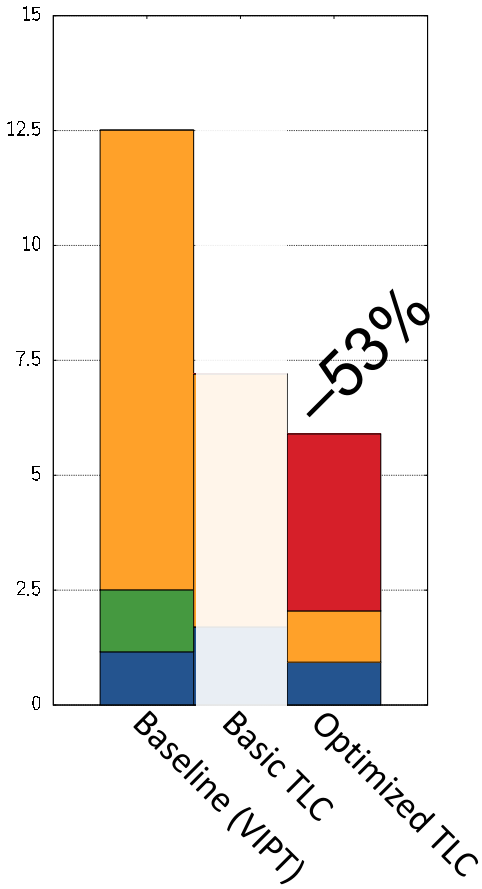


uPage-bank
H-Tree

Data

Tags
eTLB

Dynamic read energy (pJ)



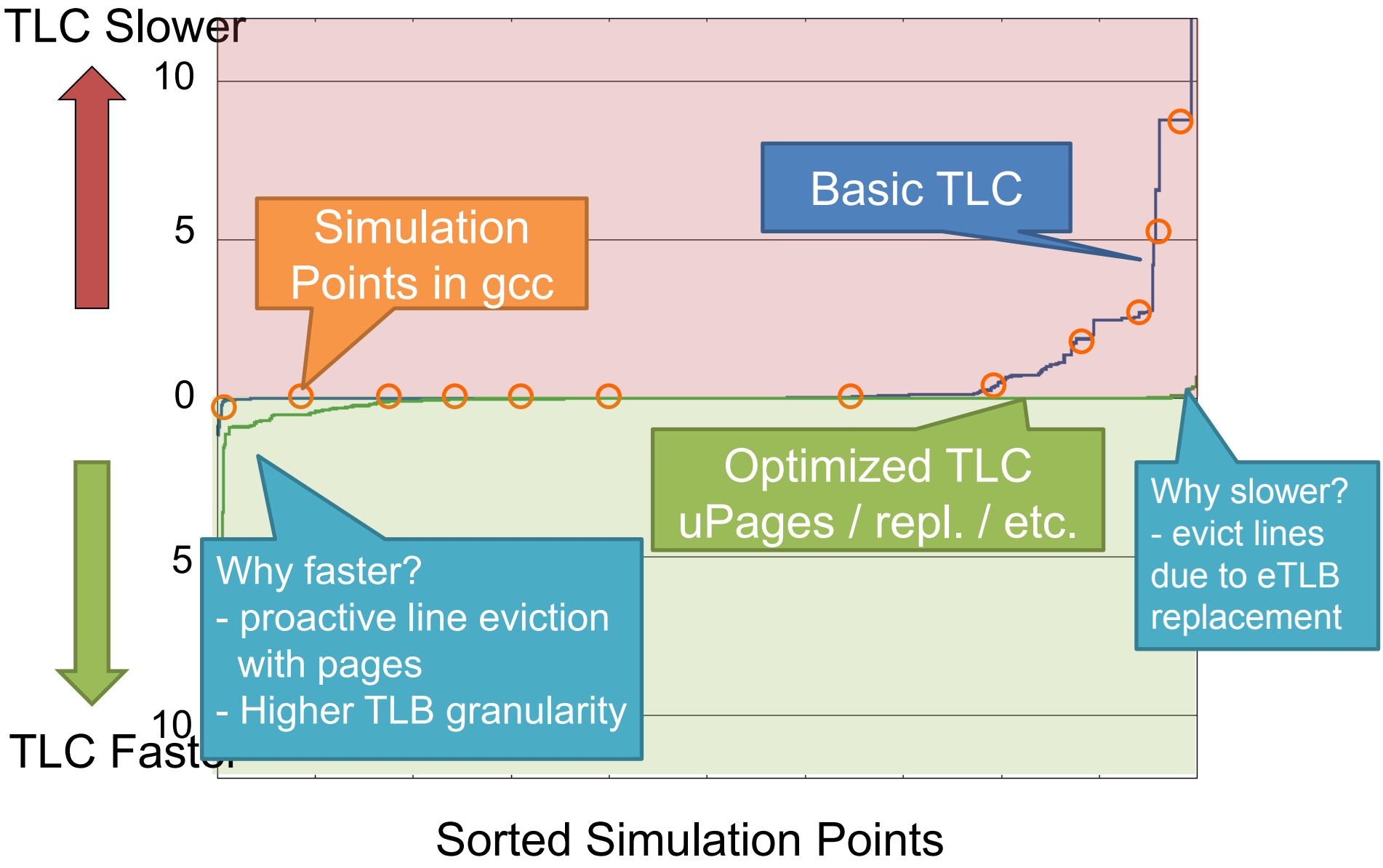
Comparable to VIPT

Much lower energy

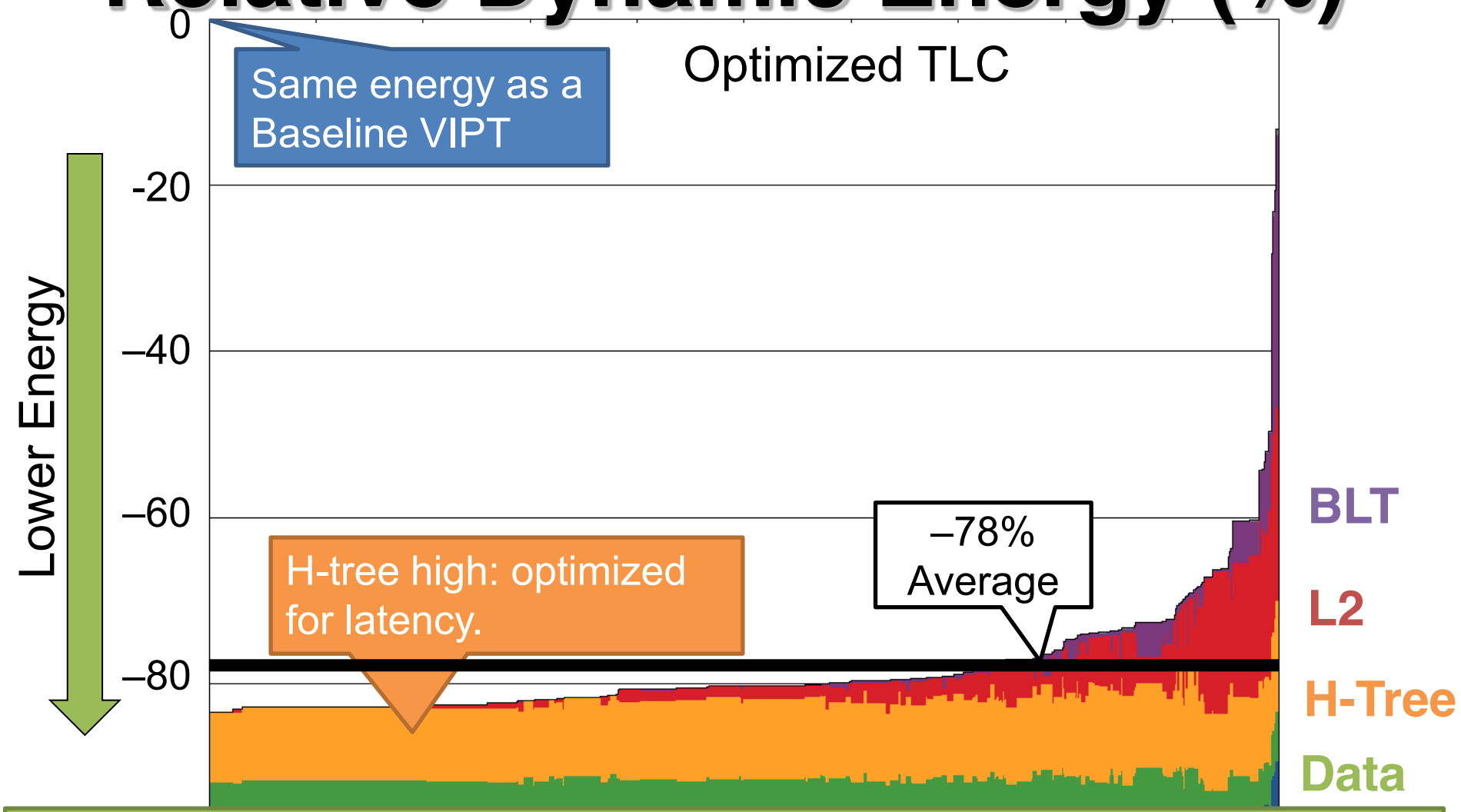
Runtime Performance and Energy

- Gem5 simulator
- SPEC2006
- Simulated SPEC's 307 longest phases
- Sorted the simulation points (phases) in ascending order (e.g. performance or energy)

Relative Performance (%)



Relative Dynamic Energy (%)



Reduce total L1D dynamic energy by 78% without hurting performance

Related Work

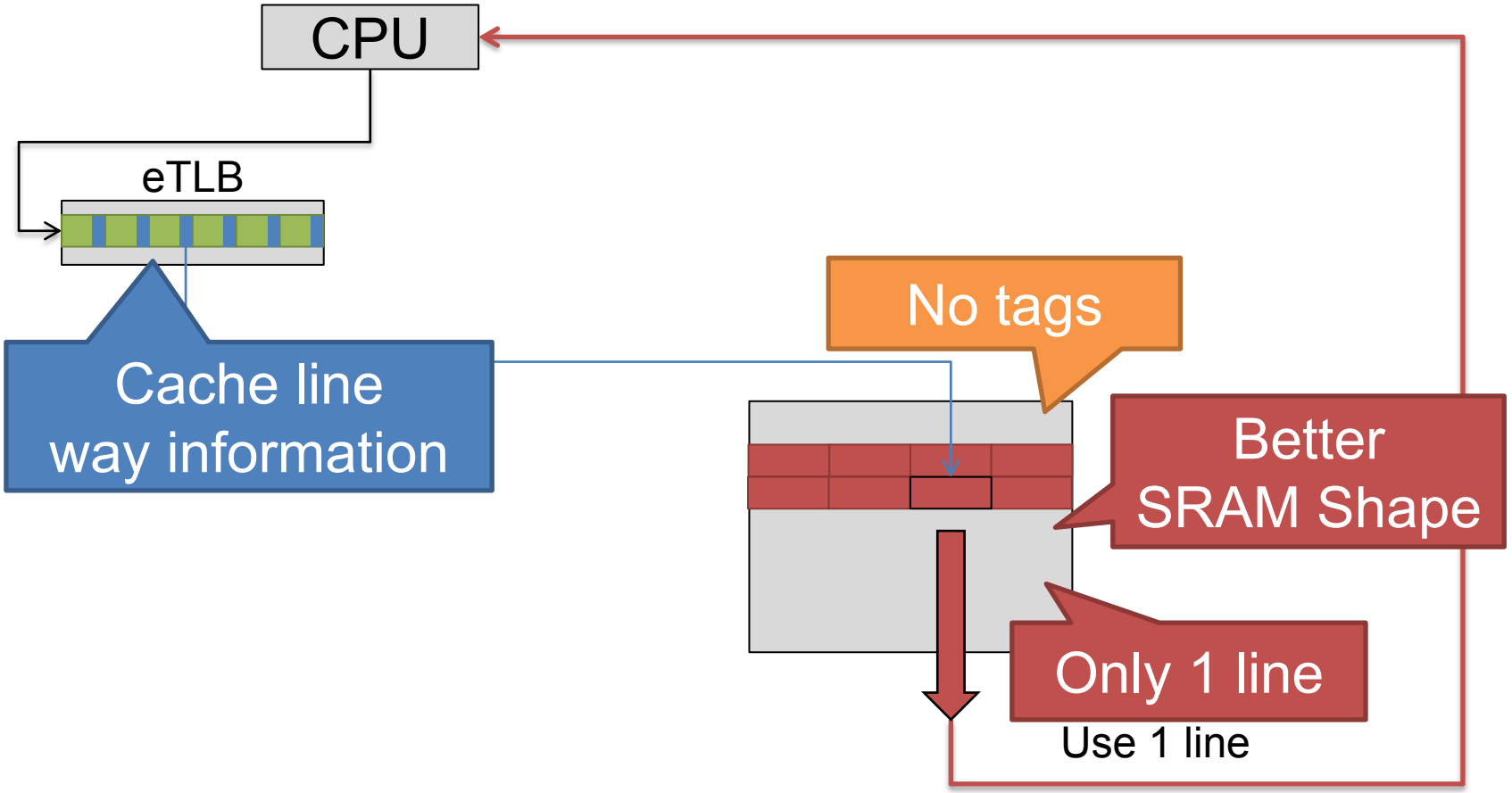
- **Way-prediction**
  - We eliminate tags
  - We always know the correct way from the eTLB
  - More in paper

- **Coarse-grained cache line tracking**
  - Region-tracker (Zebchuk) and Sector-cache (Seznec)
  - Focus on tag-area, coherency and L2/L3
  - We focus on L1 caches and dynamic energy

More in paper

- **eTLB page prediction**
  - Reduce eTLB energy even more
- **uPage optimizations**
  - Macro-page preloading to reduce TLB miss ratio
- **Super pages**
  - Split L2 TLB pages on load
- **Synonym optimizations**
  - Reduce cost of updating eTLB backpointers
- **More configurations**
  - Simulation results for different CPUs (ROB etc.)
  - More cache sizes (16kB, 32kB and 64kB)
  - VIPT with phased lookups

Summary



Reduce total L1D dynamic energy by 78% without hurting performance

Questions?