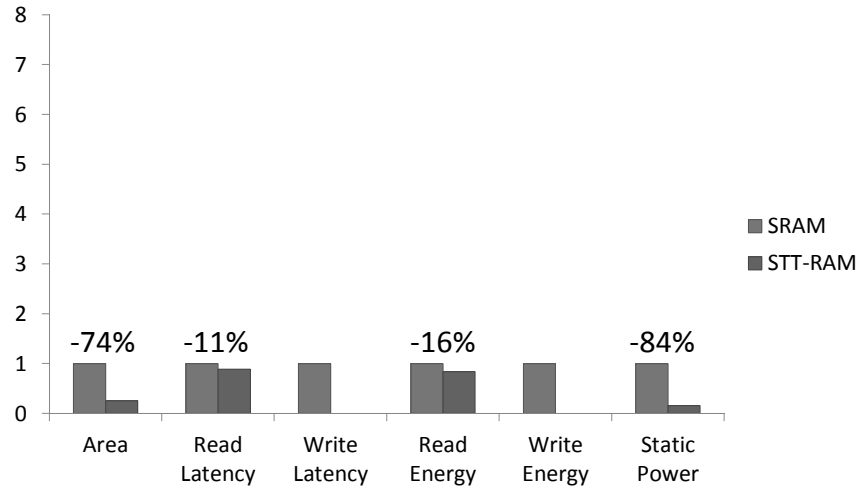
Low Energy STT-RAM Cache with Dead Write Prediction

Junwhan Ahn[1], Sungjoo Yoo[2], and Kiyoung Choi[1]

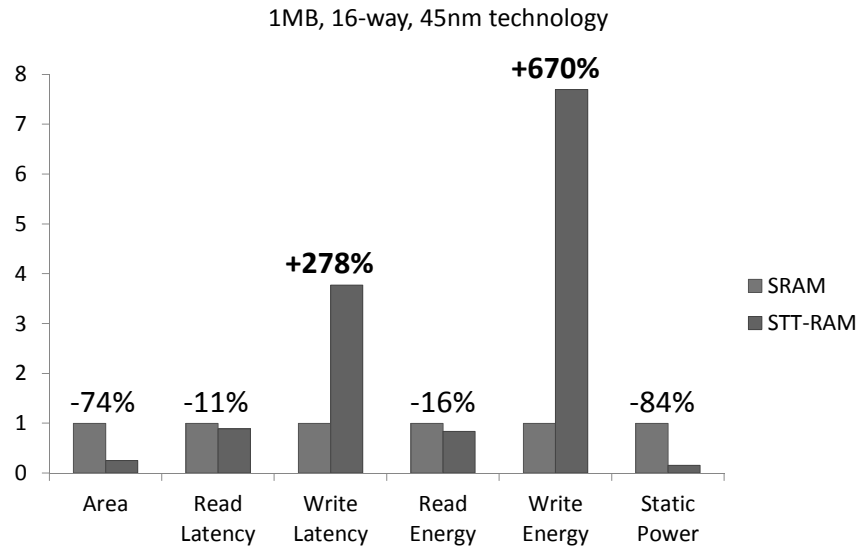
[1]Seoul National University
[2]POSTECH

SRAM Cache vs. STT-RAM Cache

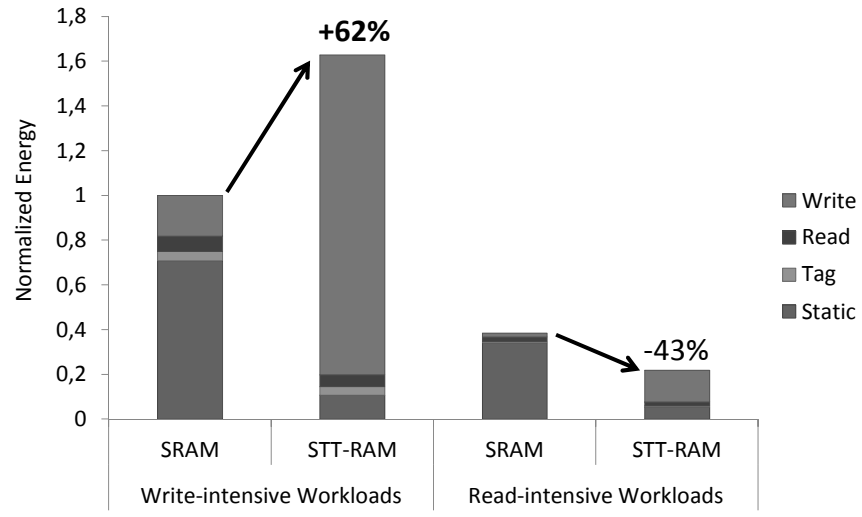
1MB, 16-way, 45nm technology



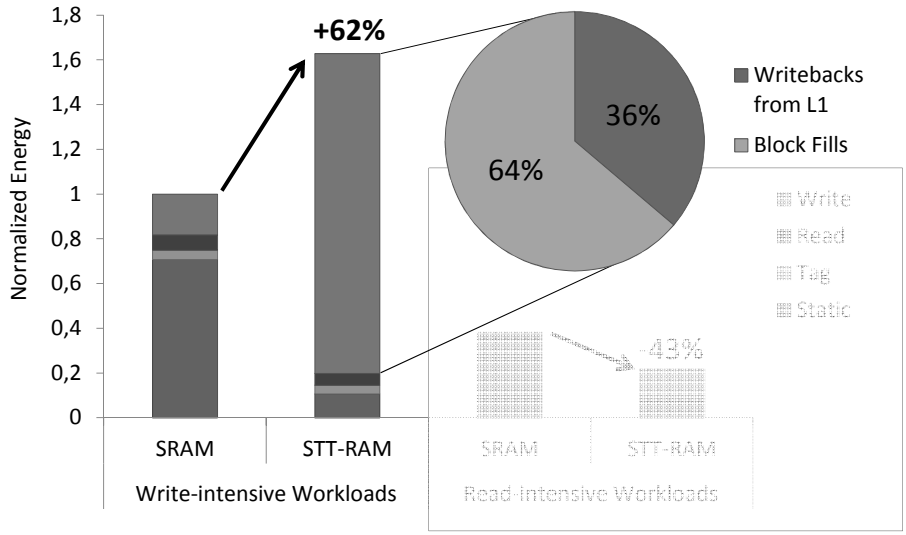
SRAM Cache vs. STT-RAM Cache



Impact of Inefficient Writes



Impact of Inefficient Writes



Key Concept

- Objectives
  - Reduce write energy incurred by both block fills and writebacks from lower-level caches
  - No modification to either LLC design or STT-RAM devices

Key Concept

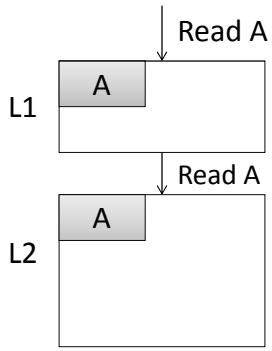
- Objectives
  - Reduce write energy incurred by both block fills and writebacks from lower-level caches
  - No modification to either LLC design or STT-RAM devices
- Observation: → block fills + writebacks
  - Most of the writes can bypass the LLC without extra cache misses → dead writes
    (e.g., a fill for a block that will never be accessed again)
  - Bypassing dead writes reduces write energy consumption

When Do Writes Become Dead?

- Dead-on-arrival fill
  - A fill operation for a block that will be accessed only once during its lifetime

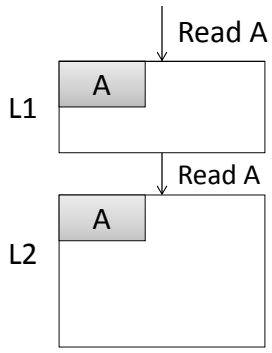
When Do Writes Become Dead?

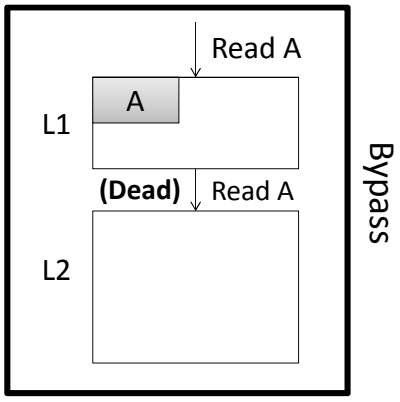
- Dead-on-arrival fill
  - A fill operation for a block that is accessed only once during its lifetime



When Do Writes Become Dead?

- Dead-on-arrival fill
  - A fill operation for a block that is accessed only once during its lifetime



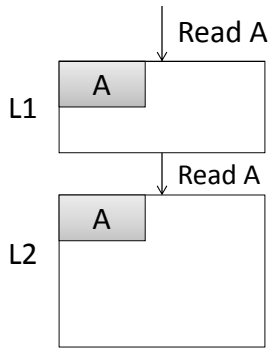


When Do Writes Become Dead?

- Dead-value fill
  - A fill operation initiated by a read miss, where the target block will be (over)written back right after the fill

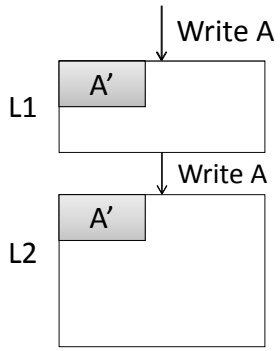
When Do Writes Become Dead?

- Dead-value fill
  - A fill operation initiated by a read miss, where the target block will be (over)written back right after the fill



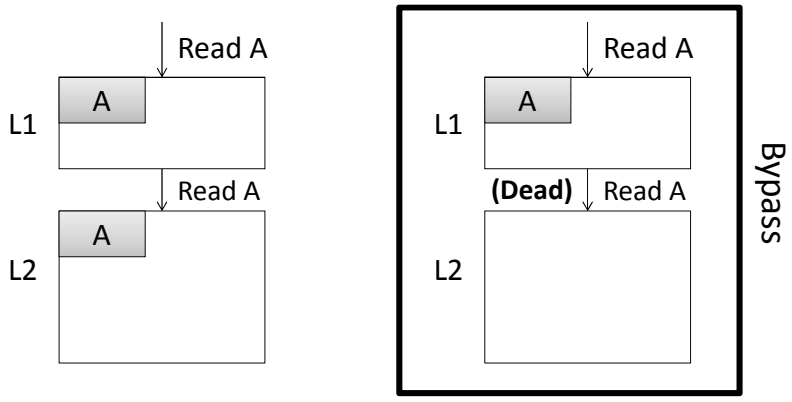
When Do Writes Become Dead?

- Dead-value fill
  - A fill operation initiated by a read miss, where the target block will be (over)written back right after the fill



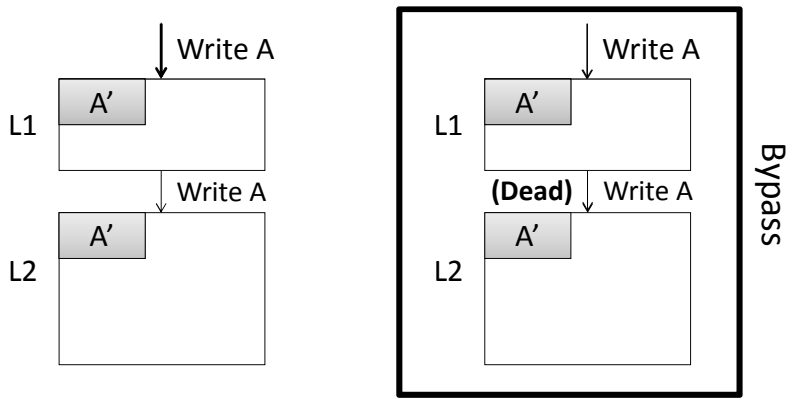
When Do Writes Become Dead?

- Dead-value fill
  - A fill operation initiated by a read miss, where the target block will be (over)written back right after the fill



When Do Writes Become Dead?

- Dead-value fill
  - A fill operation initiated by a read miss, where the target block will be (over)written back right after the fill

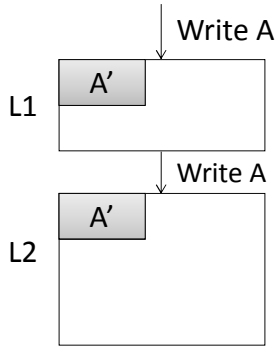


When Do Writes Become Dead?

- Closing write
  - A writeback operation that will be the last access to the block

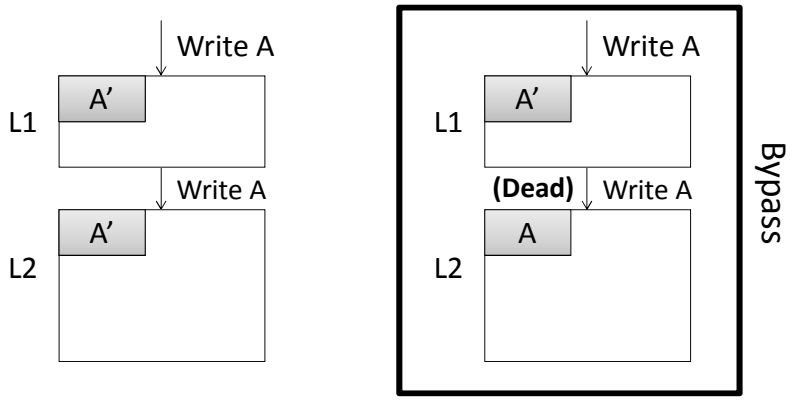
When Do Writes Become Dead?

- Closing write
  - A writeback operation that will be the last access to the block

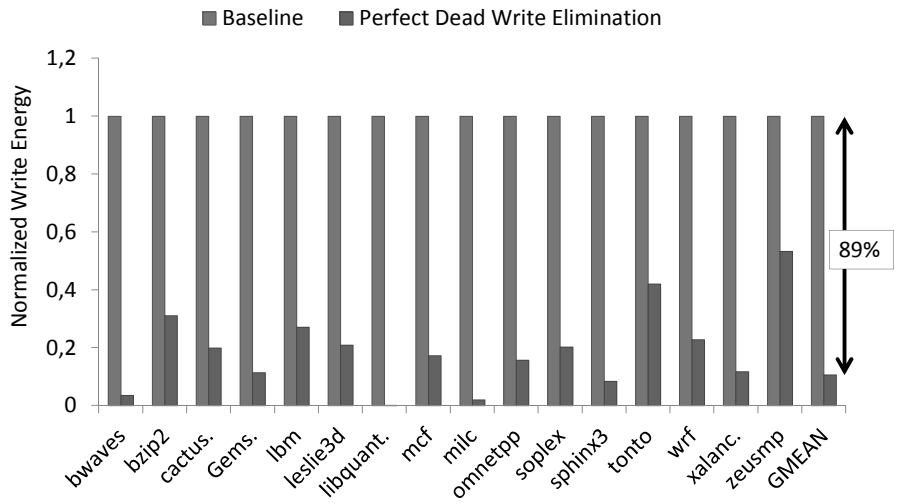


When Do Writes Become Dead?

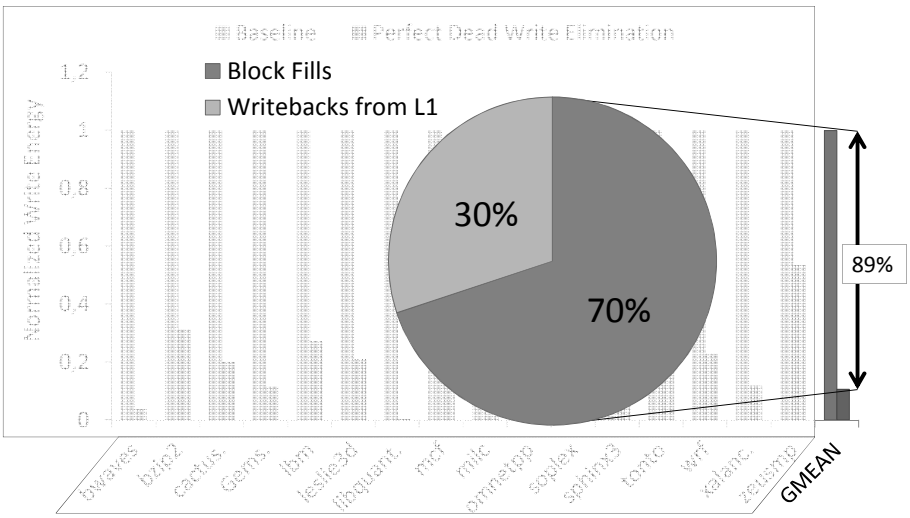
- Closing write
  - A writeback operation that will be the last access to the block



Potential



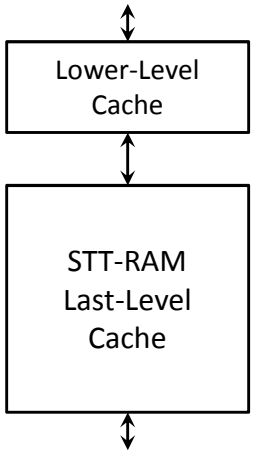
Potential



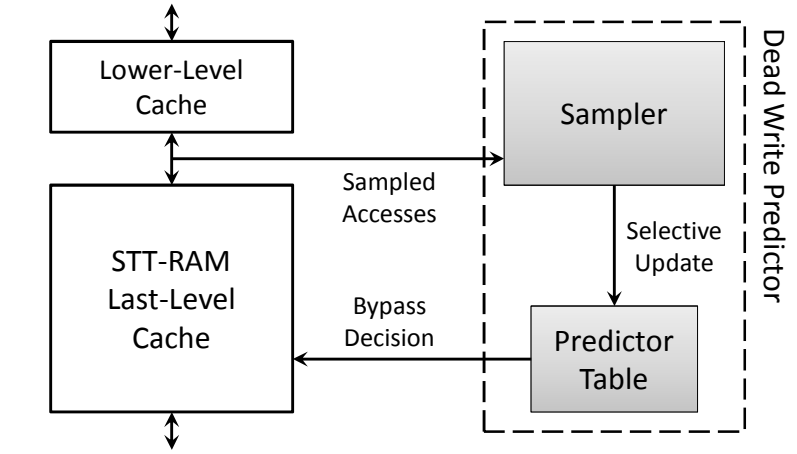
Dead Write Classification

- Dead-on-arrival fill
  - A fill operation for a block that will be accessed only once during their lifetime
- Dead-value fill
  - A fill operation initiated by a read miss, where the target block will be (over)written back right after the fill
- Closing write
  - A writeback operation that will be the last access to the block
- Identifying dead writes requires future knowledge

Dead Write Prediction

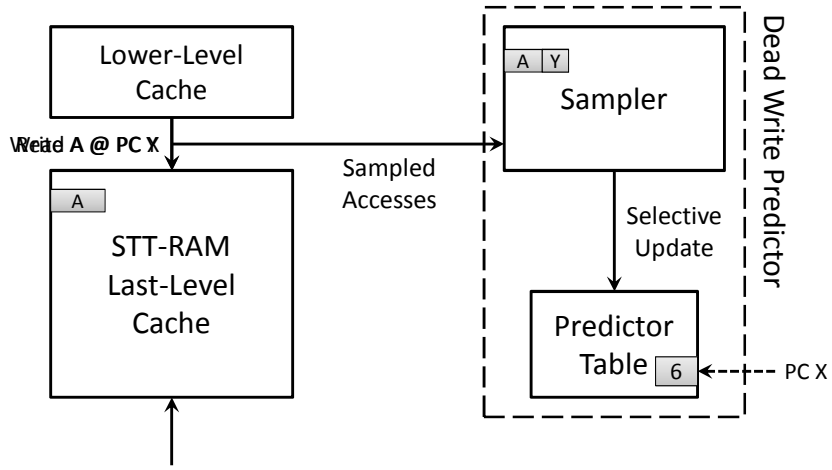


Dead Write Prediction



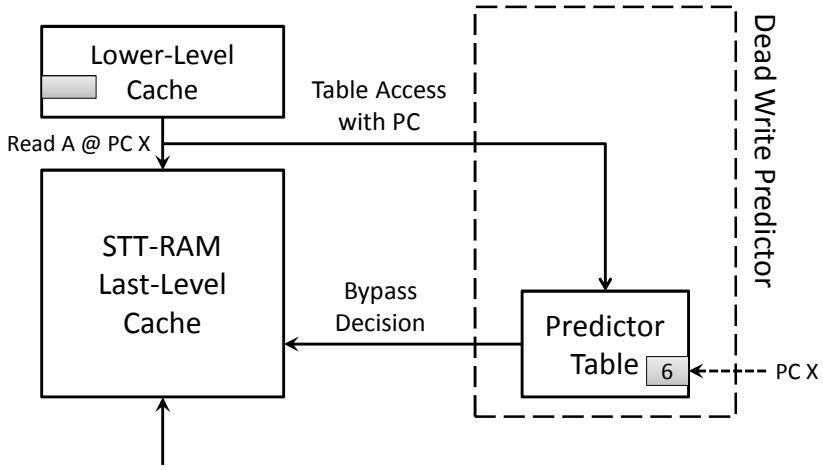
Dead Write Prediction **A**ssisted **S**TT-RAM **C**ache **A**rchitecture (DASCA)

Dead Write Prediction



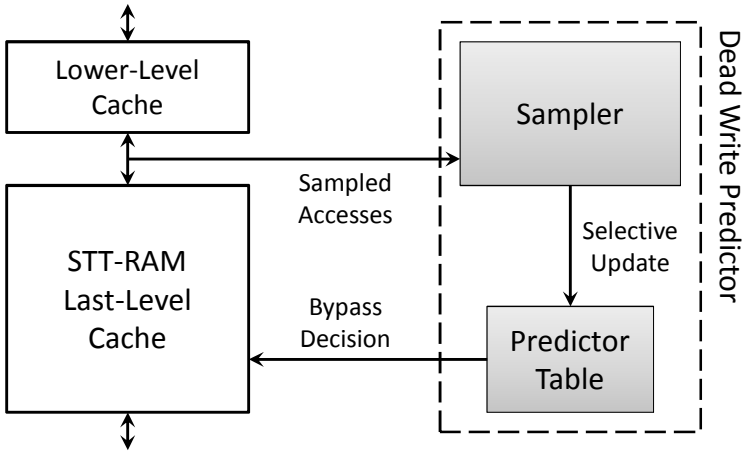
Predictor Update

Dead Write Prediction

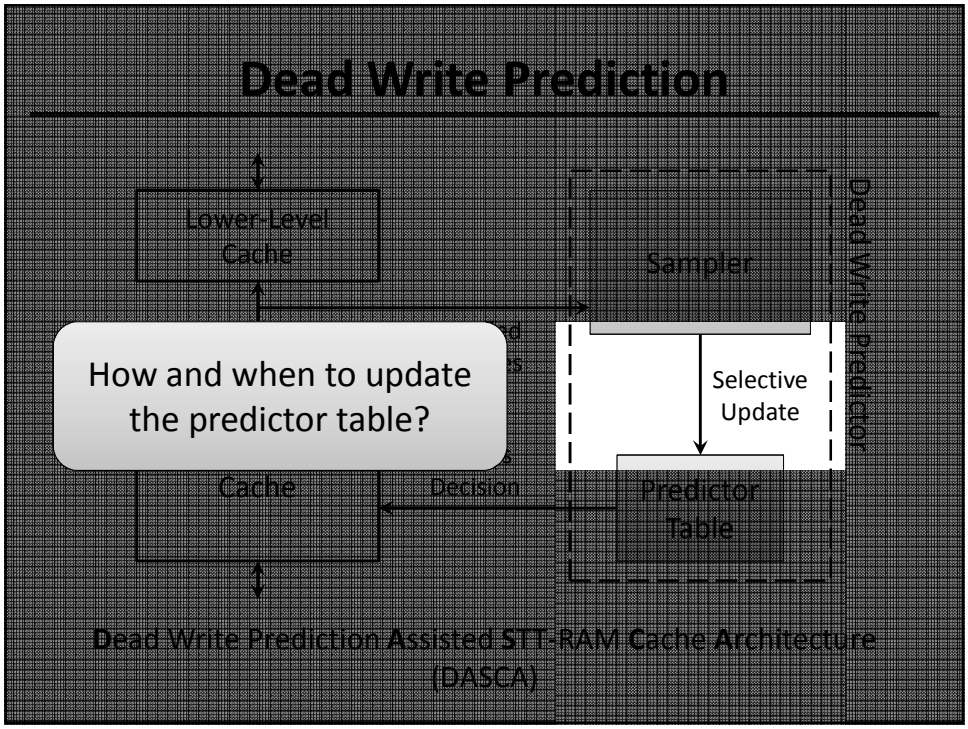


Bypass Decision

Dead Write Prediction



Dead Write Prediction **A**ssisted **S**TT-RAM **C**ache **A**rchitecture (DASCA)



Predictor Update Mechanism

Current Access Type

Last Access Type	Read Hit	Write Hit	Eviction
Read Hit			
Read Miss			
Write Hit			
Write Miss			

Predictor Update Mechanism

Current Access Type

Last Access Type	Read Hit	Write Hit	Eviction
Read Hit			
Read Miss			Dead (DOA Fill)
Write Hit			
Write Miss			

Predictor Update Mechanism

Current Access Type

Last Access Type	Read Hit	Write Hit	Eviction
Read Hit			
Read Miss		Dead (Dead-Value Fill)	Dead (DOA Fill)
Write Hit			
Write Miss			

Predictor Update Mechanism

Current Access Type

Last Access Type	Read Hit	Write Hit	Eviction
Read Hit			
Read Miss		Dead (Dead-Value Fill)	Dead (DOA Fill)
Write Hit			Dead (Closing Write)
Write Miss			Dead (Closing Write)

Predictor Update Mechanism

Current Access Type

Last Access Type	Read Hit	Write Hit	Eviction
Read Hit			
Read Miss	Live	Dead (Dead-Value Fill)	Dead (DOA Fill)
Write Hit	Live	Live	Dead (Closing Write)
Write Miss	Live	Live	Dead (Closing Write)

Predictor Update Mechanism

Current Access Type

Last Access Type	Read Hit	Write Hit	Eviction
Read Hit	ignore	ignore	ignore
Read Miss	Live	Dead (Dead-Value Fill)	Dead (DOA Fill)
Write Hit	Live	Live	Dead (Closing Write)
Write Miss	Live	Live	Dead (Closing Write)

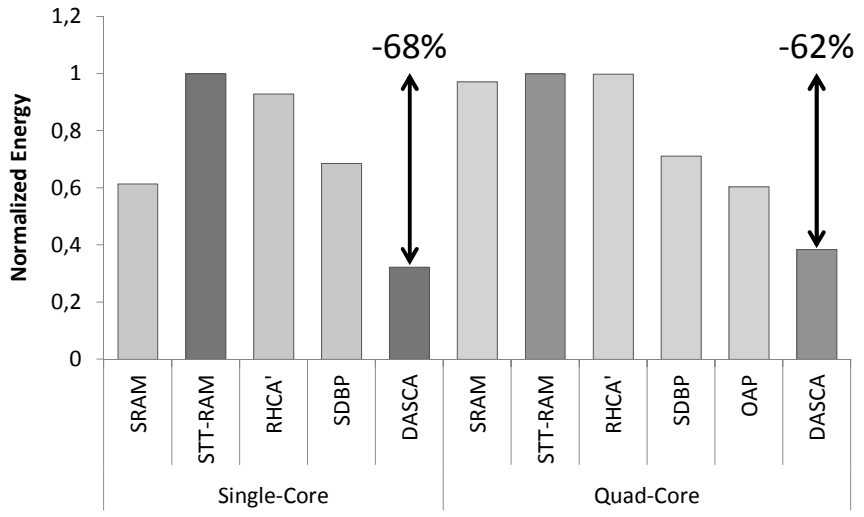
Evaluation

- Cycle-accurate x86-64 simulator based on Pin
  - 3GHz, four-issue, out-of-order cores
  - 32KB 4/8-way, 64B-block, private L1 I/D-caches
  - 1MB per core, 16-way, 64B-block, shared L2 cache
  - DDR3-1600, two channels, FR-FCFS, open row policy
- Workloads
  - Single-core: 16 write-intensive SPEC CPU2006 workloads
  - Quad-core: multiprogrammed SPEC CPU2006, PARSEC 2.1

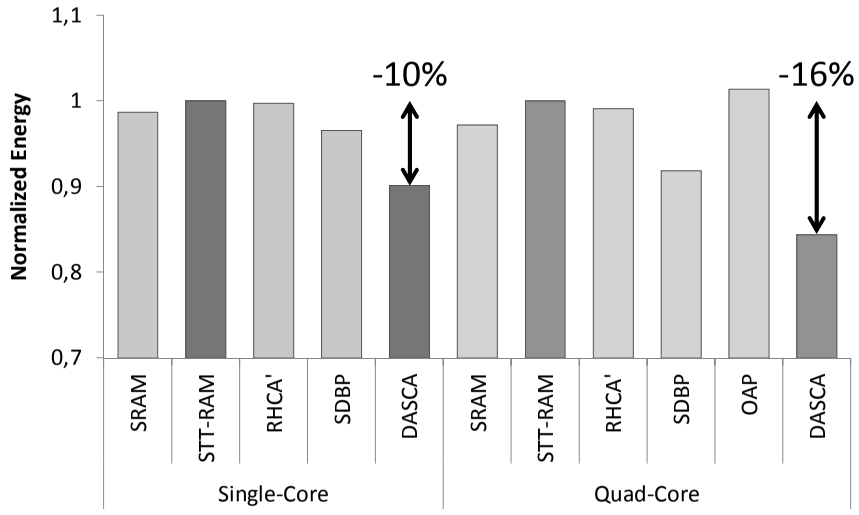
Comparison

- Region-based Hybrid Cache Architecture [Wu'09]
  - Each set composed of 4 SRAM ways & 12 STT-RAM ways
- Sampling Dead Block Predictor [Khan'10]
  - Not targeted for STT-RAM caches
  - Bypassing fill operations for dead blocks
- Obstruction-aware Cache Management [Wang'13]
  - Per-core bypass decision for LLC interference mitigation

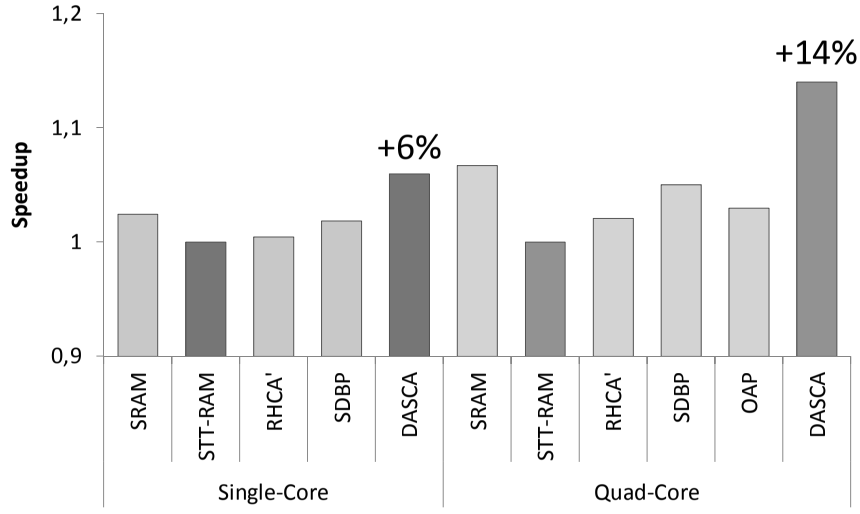
L2 Cache Energy



Main Memory Energy



Speedup



Conclusion

- STT-RAM last-level caches
  - Promising, but high write energy & long write latency
- Our solution: DASCA
  - Classification of dead writes
    - dead-on-arrival fill, dead-value fill, closing write
  - Dead writes are bypassed
- Evaluation for single-/quad-core systems
  - 68/62% reduction in LLC energy
  - 10/16% reduction in main memory energy
  - 6/14% improvement in system performance