九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Dynamic Management Technique to Mitigate Performance Degradation for Low-Leakage Caches

Komiya, Reiko
JSPS Research Fellow DC | Department of Informatics, Kyushu University | Department of Informatics, Kyushu University

Inoue, Koji
Department of Informatics, Kyushu University

Murakami, Kazuaki
Department of Informatics, Kyushu University

http://hdl.handle.net/2324/6376

出版情報：Proceedings of Cool Chips X, pp.173-184, 2007-04-19
バージョン：
権利関係：

KYUSHU UNIVERSITY

Dynamic Management Technique to Mitigate Performance Degradation for Low-Leakage Caches

Reiko Komiya, Koji Inoue, Kazuaki Murakami

Kyushu University

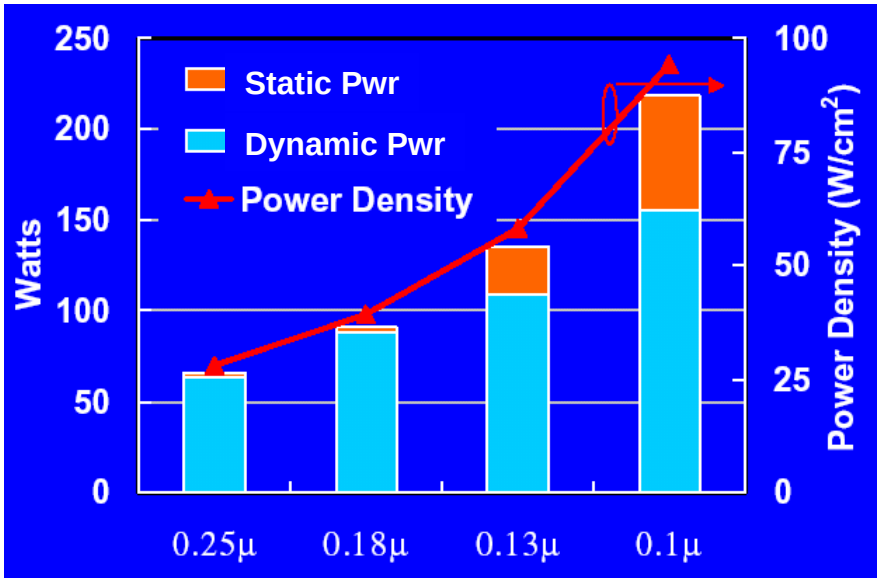
Outline

- Introduction
  - Leakage energy of cache memory
  - Conventional low leakage cache : Cache decay
- Problem of cache decay approach
- Solution: “Always-Active scheme”
- Results
- Conclusions

Introduction

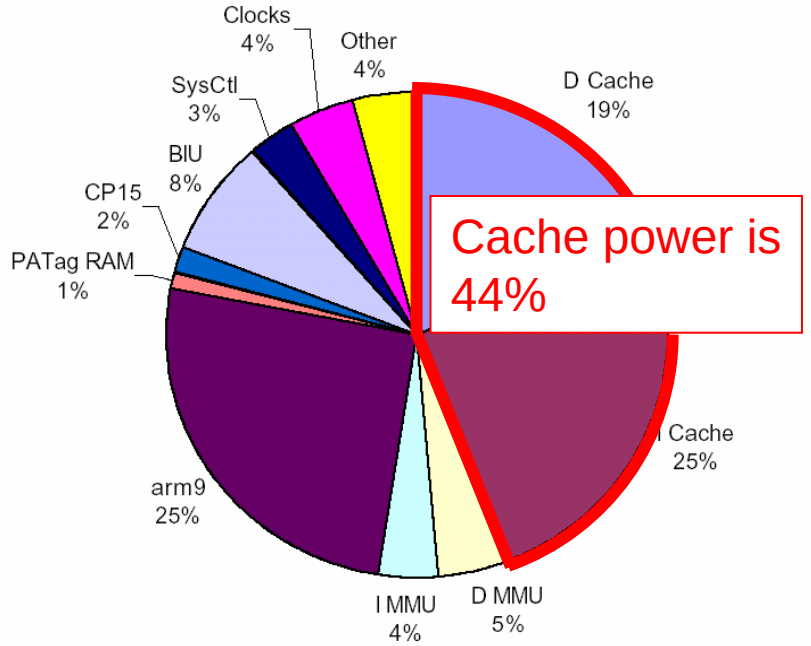
Energy consumption = Dynamic energy + Static energy

Breakdown of energy consumption in a processor family *1



Leakage energy increases with the progress of process technology

Power Analysis of ARM920T *2



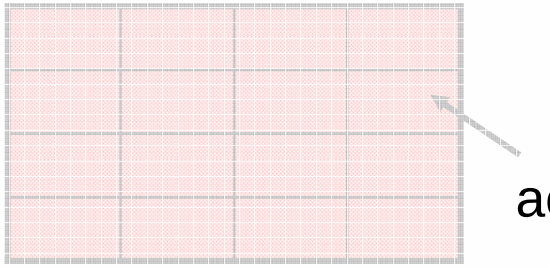
Cache leakage reduction is very important!!

*1 Fred Pollack (Intel Fellow): New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies, Micro32
*2 Simon Segars, “Low Power Design Techniques for Microprocessors,” ISSCC2001

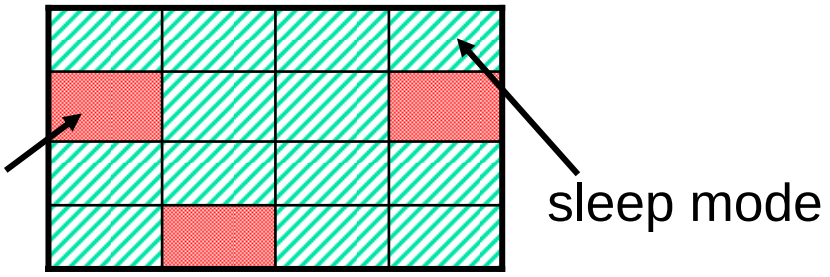
Normal cache & Conventional low-leakage cache

Normal cache

(doesn't support any leakage reduction technique)



Conventional low-leakage cache



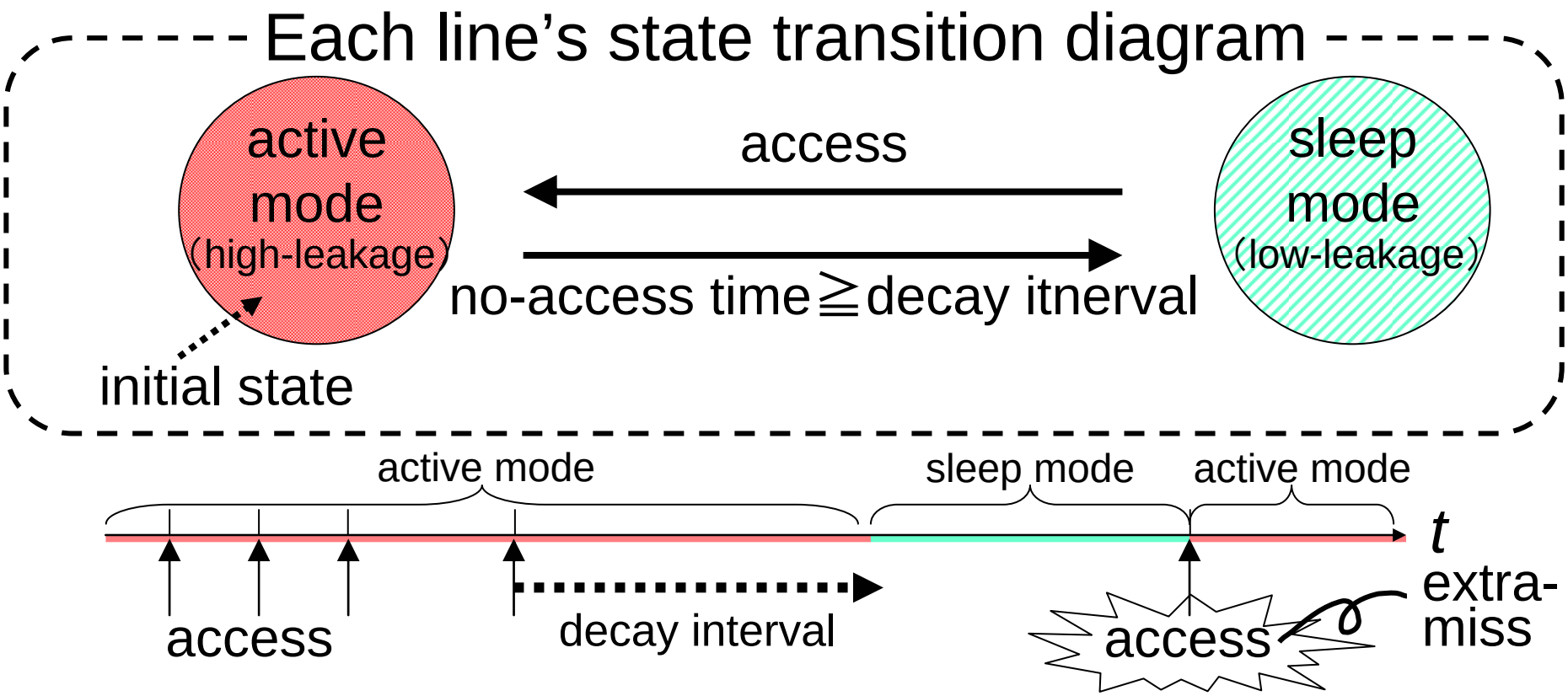
Two operation modes

- ✓active mode : high-leakage ☹ , preserve the data ☺
- ✓sleep mode : low-leakage ☺ , destroy the data ☹

majority	sleep mode	active mode
good	Leakage ↓	Performance ↑
bad	Performance ↓	Leakage ↑

Mode transition algorithm for cache decay

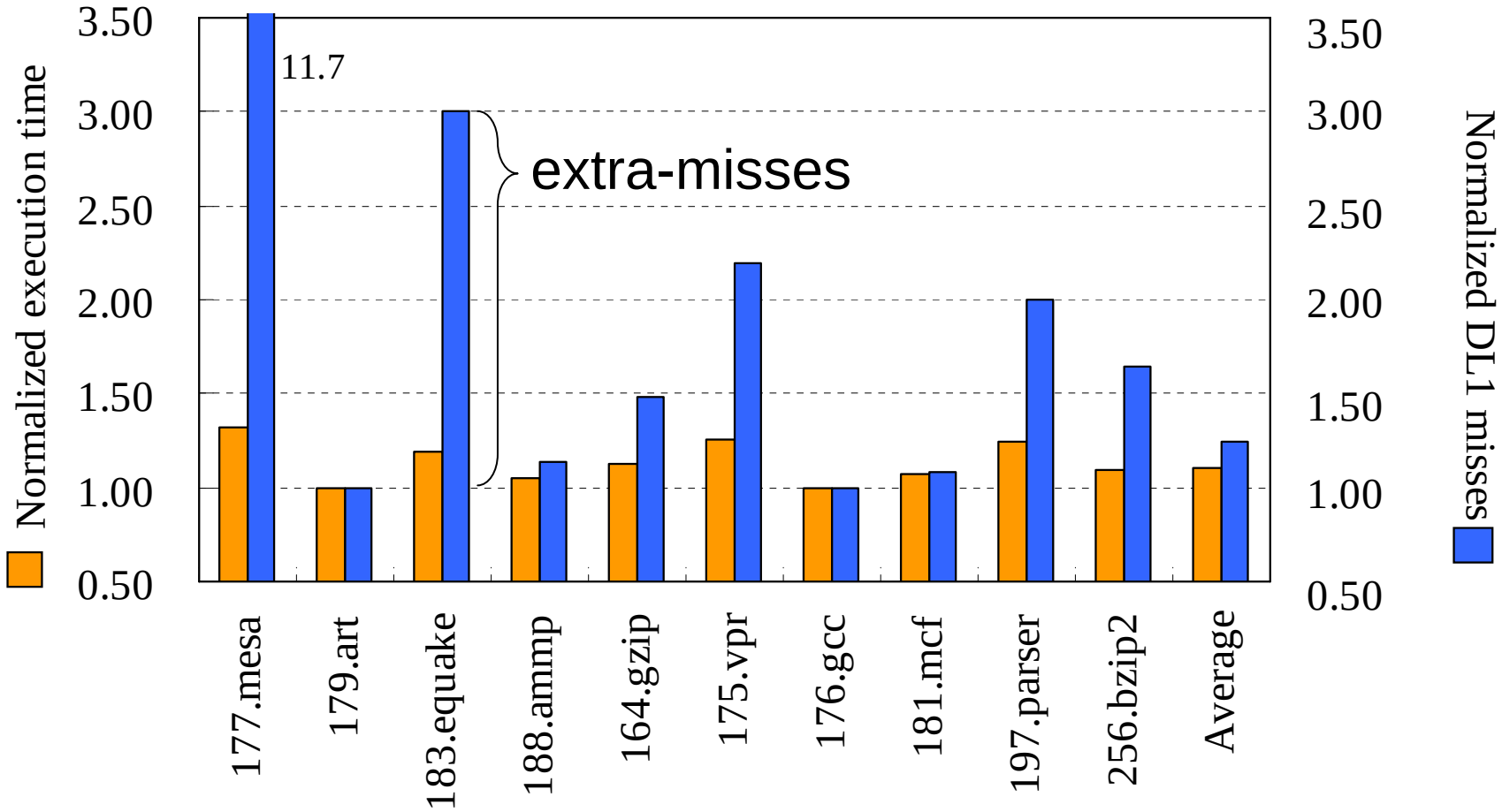
Cache decay: Conventional low-leakage cache



If destroyed data is referred to again, the number of misses increases

Extra-misses lead to performance degradation!!

Performance impact of extra-misses



Many extra-misses cause large performance degradation!

Our Goal

Achieve high-performance, low-leakage cache!

- Problem of conventional low-leakage caches
  - Degrades the performance due to extra-misses
- Our approach
  - Reduces extra-misses and improve performance
  - Prohibits some cache lines from going to sleep mode

Analysis of extra-misses

- Extra-Miss Density (*EMD*):

$$EMD_i = \frac{\text{number of extra-misses at the cache line } i}{\text{average number of extra-misses for all cache lines}}$$

- Example

Number of extra-misses at each cache line

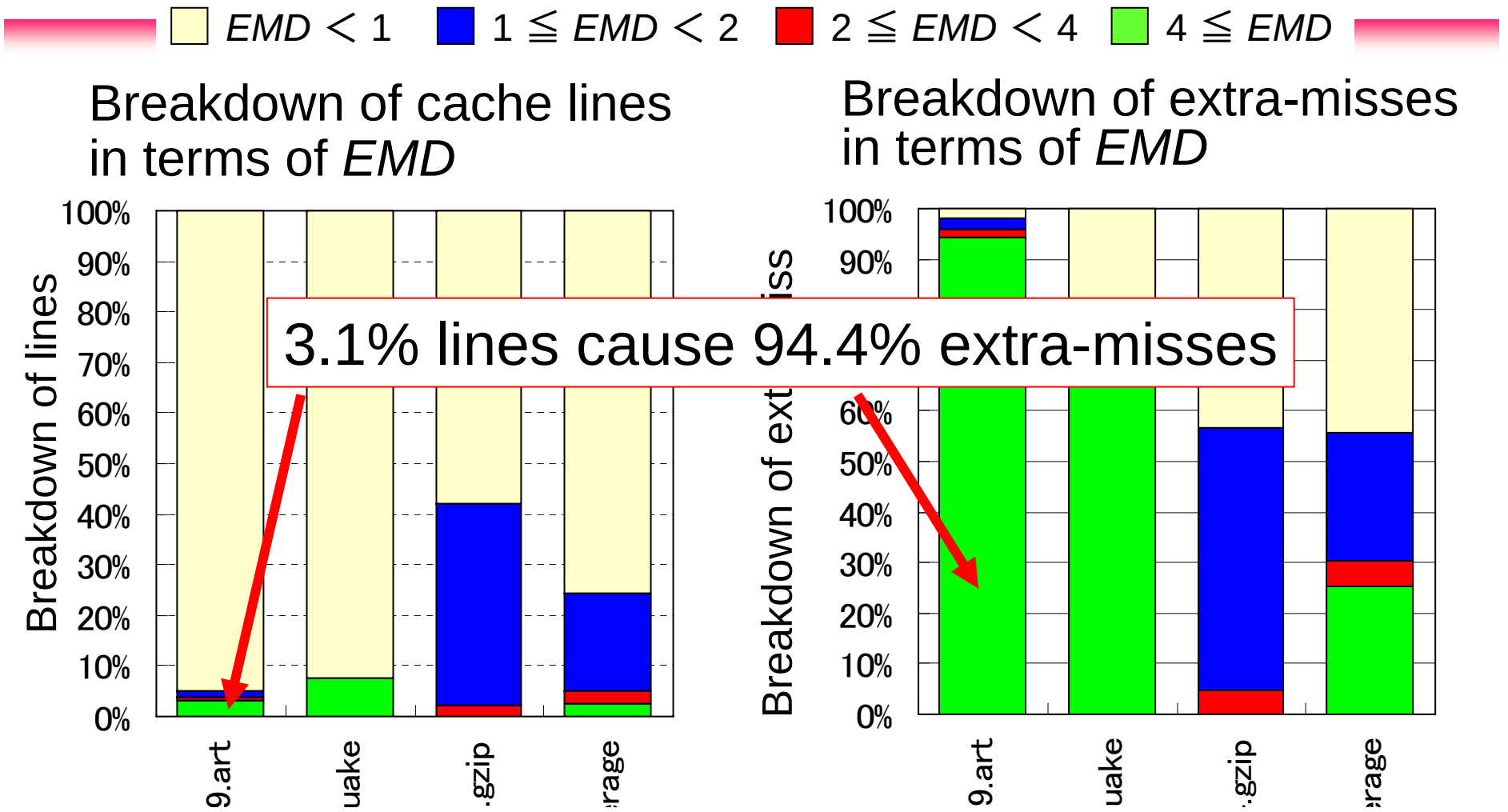
6	5	1
2	4	1
10	1	60

•Total number of extra-misses: 90
•Number of lines: 9
⇒ Average number of extra-misses: 10

EMD_6=1
EMD_7=0.1
EMD_8=6

Cache lines which often cause extra-misses have high *EMD* !

Characteristics of extra-misses



A small number of high *EMD* lines often produce extra-misses

Always-Active scheme

- Introduces “Always-Active mode (AA mode)”
- Identifies cache lines which would cause frequently extra-misses, and marks the line AA mode line
- Prohibits AA mode line from going to sleep mode
- Reduces extra-misses in AA mode line

How to identify AA mode lines

(a) Decay interval has passed since the last access

(b) Tag hits

- In the case of normal cache, this access would be hit
- In the case of cache decay, this access would be extra-miss

The accesses which satisfy (a) and (b) are called “extra-miss-candidate (EMC)”

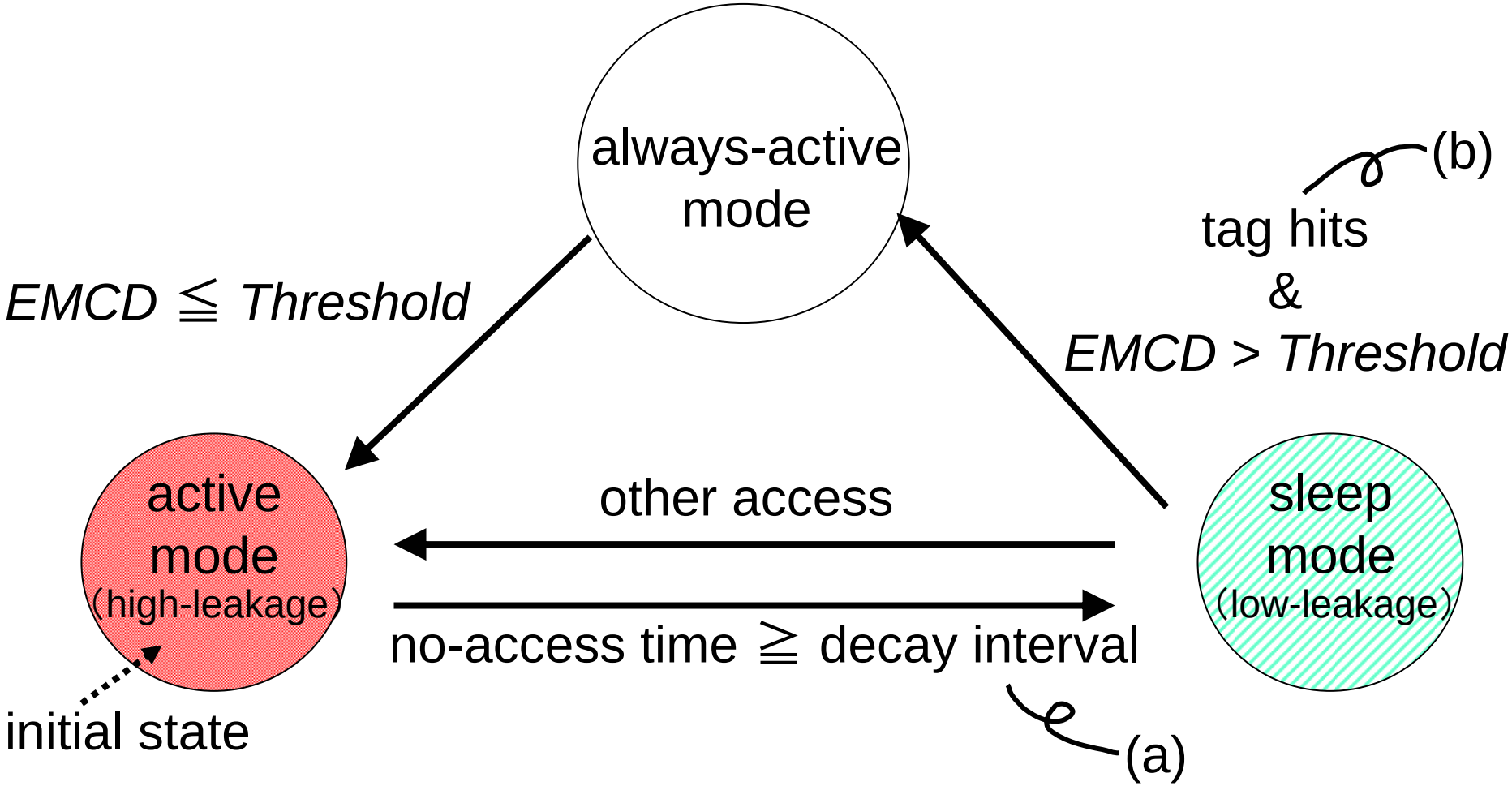
- If an access is EMC, we make the following judgment

$$EMCD_i = \frac{\text{number of EMC at the cache line } i}{\text{average number of EMC for all cache lines}} > \text{Threshold}$$

- If this equation is satisfied, the line transits to AA mode line

Mode transition algorithm for AA scheme

A line which causes EMC frequently ⇒ AA mode line



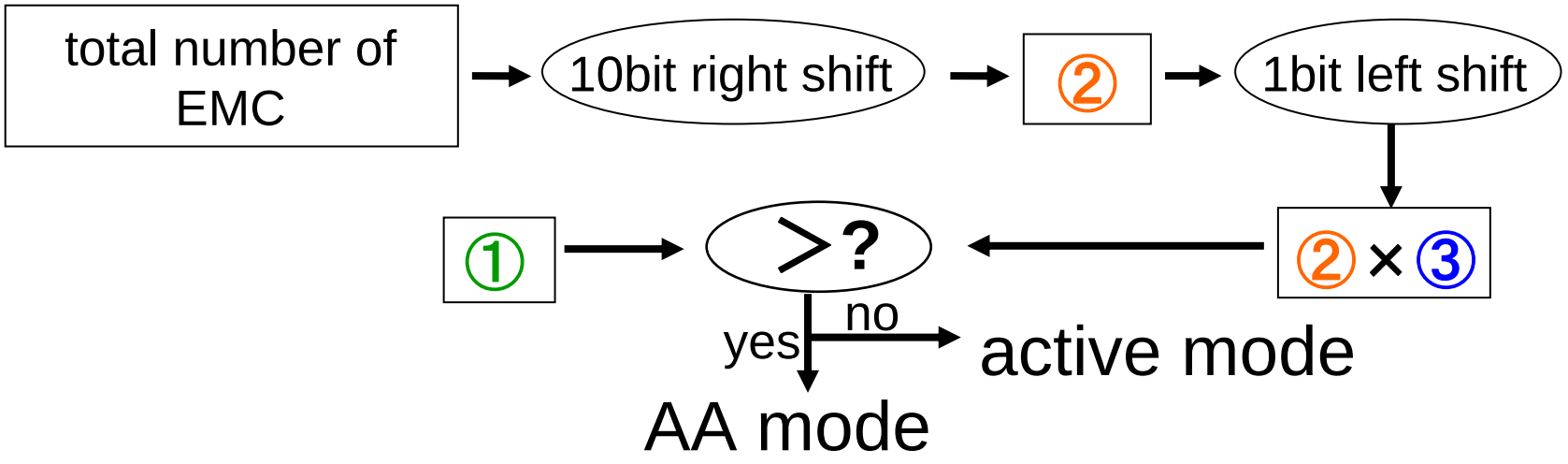
How to measure $EMCD_i$ dynamically

$$EMCD_i = \frac{\text{number of EMC at the cache line } i \quad \cdots ①}{\text{average number of EMC for all cache lines} \quad \cdots ②}$$

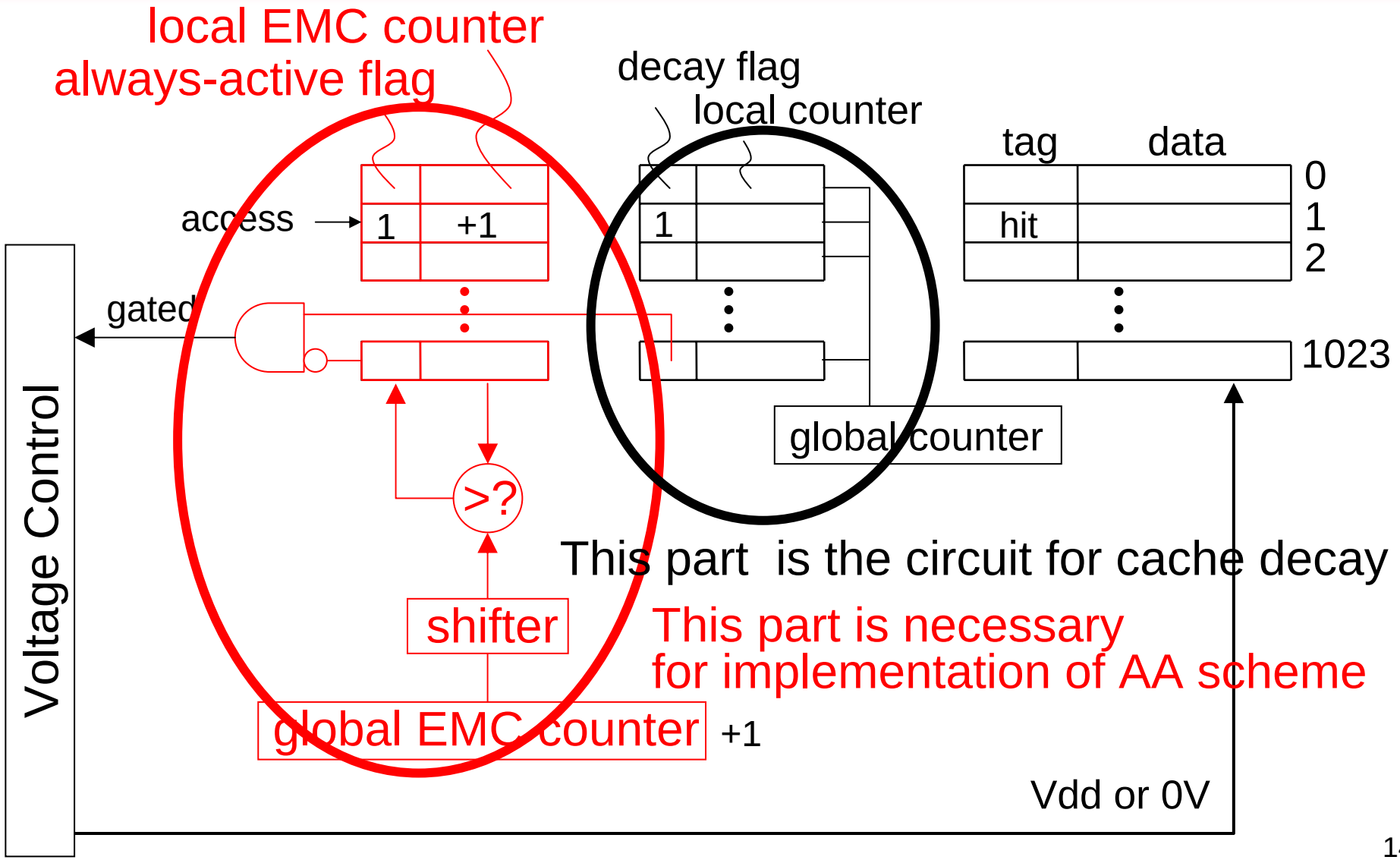
$$> \text{Threshold} \quad \cdots ③$$

⇒ ① > ② × ③ ・・・We judge this equation during program execution

Example）Number of cache lines = 1024 (=2^{10}), Threshold = 2 (=2^1)



Hardware implementation for AA scheme

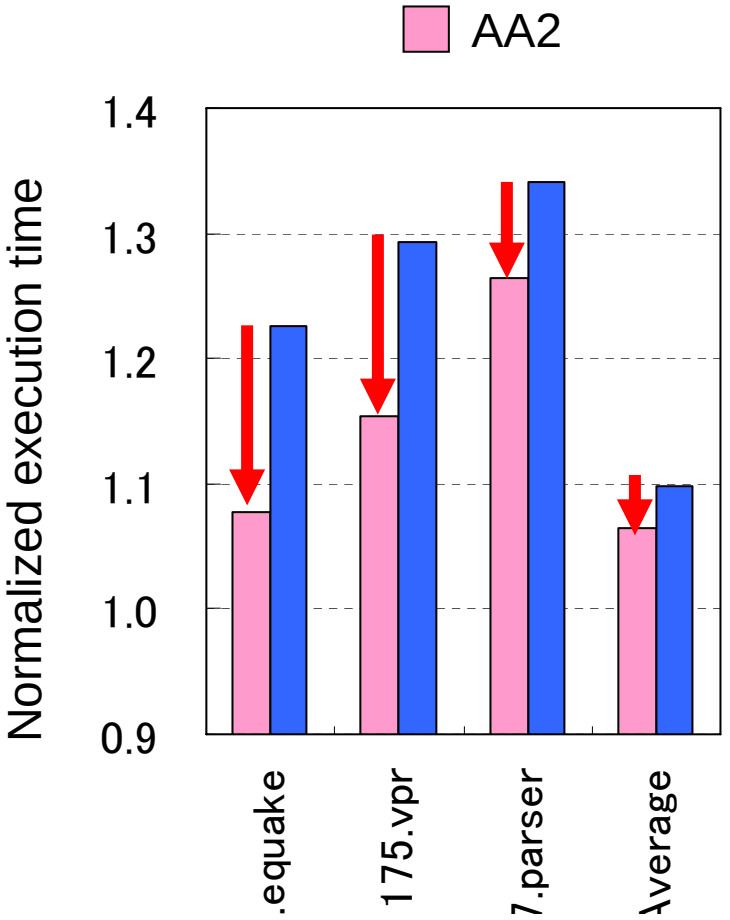


Experimental setup

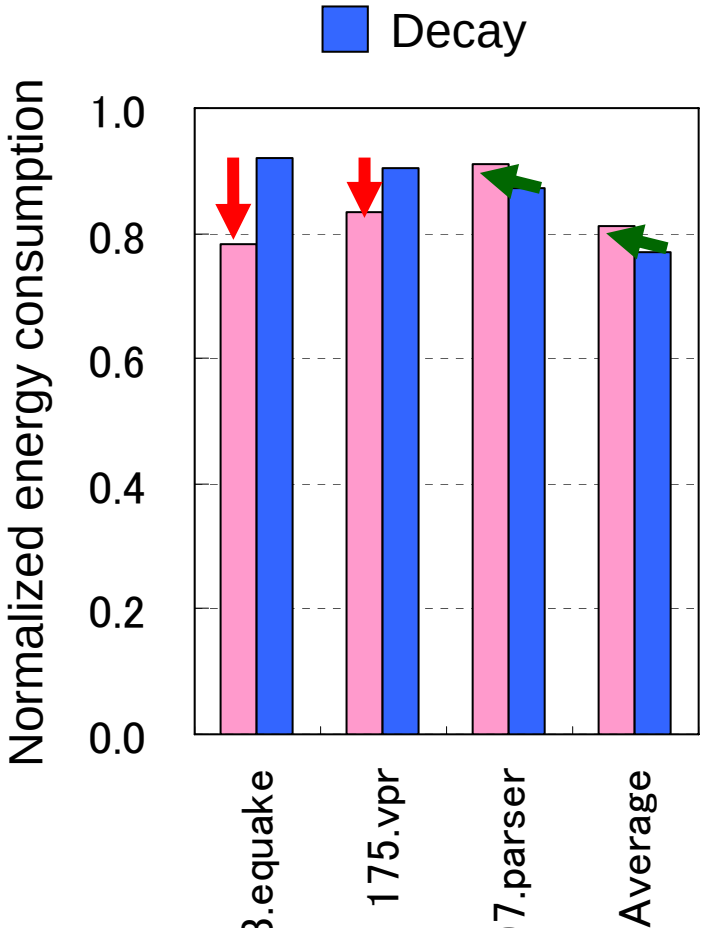
- Evaluation models
  - **Base**: normal cache
  - **Decay**: conventional low-leakage cache
  - **AA2**: cache decay with AA Scheme (threshold value=2)
- Cache configuration
  - L1 data cache
    - Cache size: 32KB
    - Associativity: 32way
    - Hit latency: 1 clock cycle
    - Miss penalty: 64 clock cycles
- Evaluation items
  - Normalized execution time (against Base model)
  - Normalized energy consumption (against Base model)

$E_{total} = LE_{L1} + DE_{L1} + DE_{memory}$

Performance & Energy consumption



AA2 improves performance in all benchmark programs



AA2 reduces or increases slightly energy

Conclusions

- We have proposed a high-performance, low-leakage cache: AA Scheme
  - Detects lines which cause EMC frequently at run time
  - Improves the performance by means of making the lines AA mode and prohibiting AA mode lines from going to sleep mode

- Evaluation results
  - Higher performance and lower energy consumption
  - Best case (183.equake):
    - Performance degradation: 22% →7%
    - Energy consumption: 13% reduction