# CACHE POWER CONSUMPTION

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

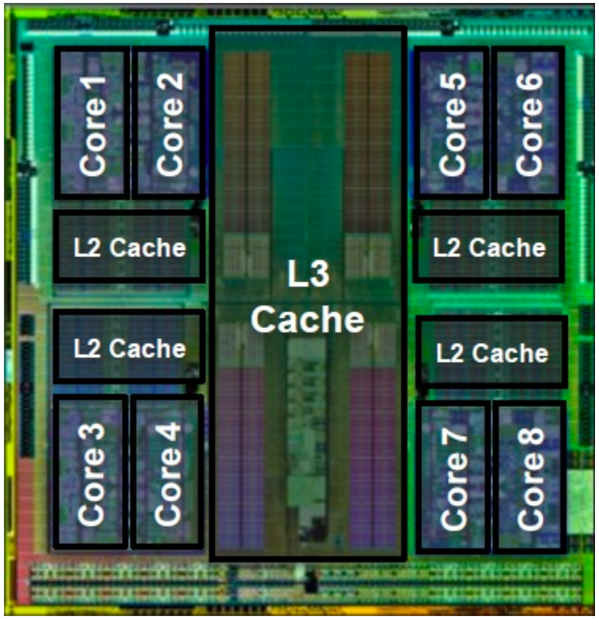
# Overview

- Upcoming deadline
  - Feb. 3rd: project group formation
- This lecture
  - Cache power consumption
  - Cache banking
  - Way prediction
  - Resizable caches
  - Gated Vdd/ cache decay, drowsy caches

# Main Consumers of CPU Resources?

- A significant portion of the processor die is occupied by on-chip caches
- Main problems in caches
  - Power consumption
    - Power on many transistors
  - Reliability
    - Increased defect rate and errors

**Example: FX Processors**




*[source: AMD]*

# Recall: CPU Power Consumption

- Major power consumption issues

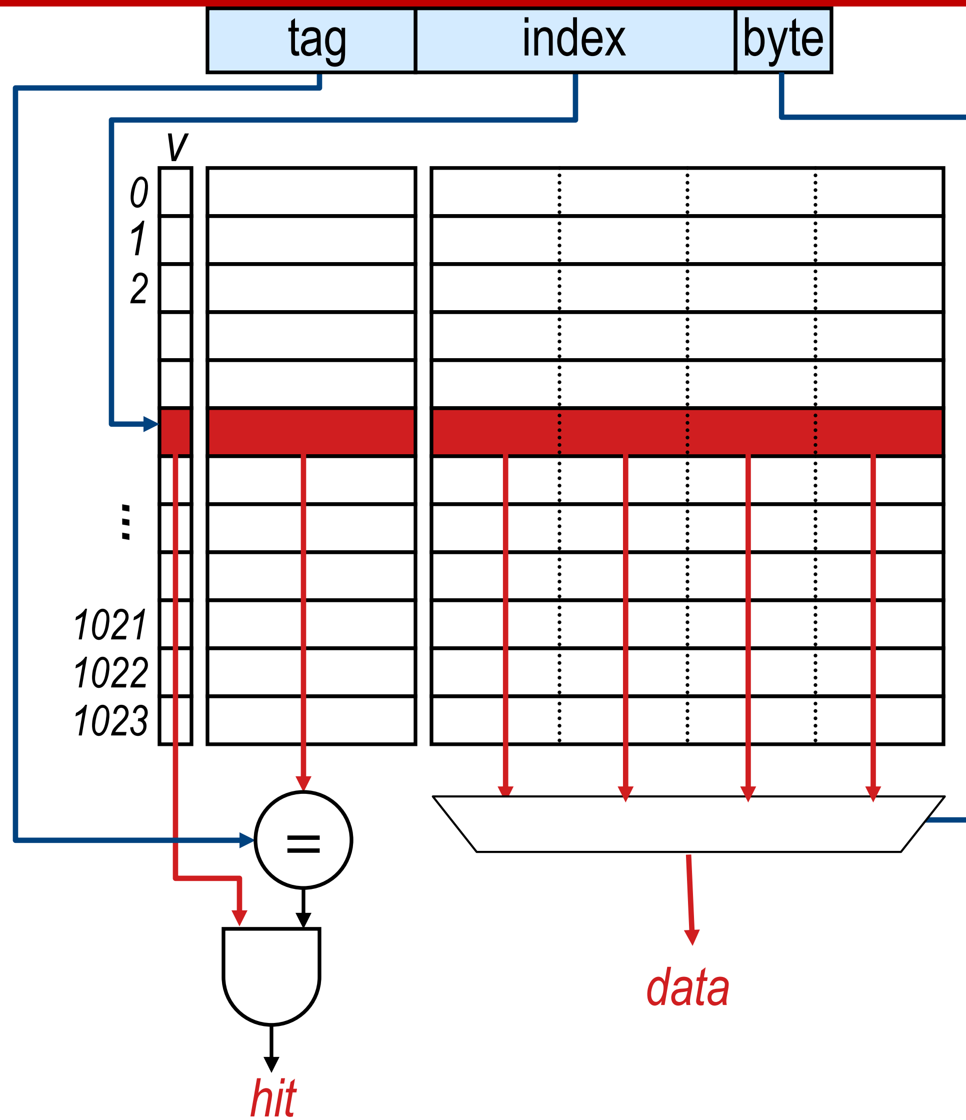
| Peak Power/Power Density | Average Power |
|---|---|
| ❑ Heat<br>○ Packaging, cooling, component spacing<br>❑ Switching noise<br>○ Decoupling capacitors | ❑ Battery life<br>○ Bulkier battery<br>❑ Utility costs<br>○ Probability, cannot run your business! |
| **Caches generate little heat (low activity factor)** | **Caches consume high average power (~1/3)** |

# Cache Power Management

- Circuit techniques
  - Transistor sizing, multi-Vt, low-swing bit-lines, etc.
- Microarchitecture techniques
  - Static techniques
    - banking, phased tag/data access, way prediction
  - Dynamic techniques
    - gated-Vdd, cache decay, drowsy caches
- Compiler techniques
  - Data partitioning to enable sleep mode

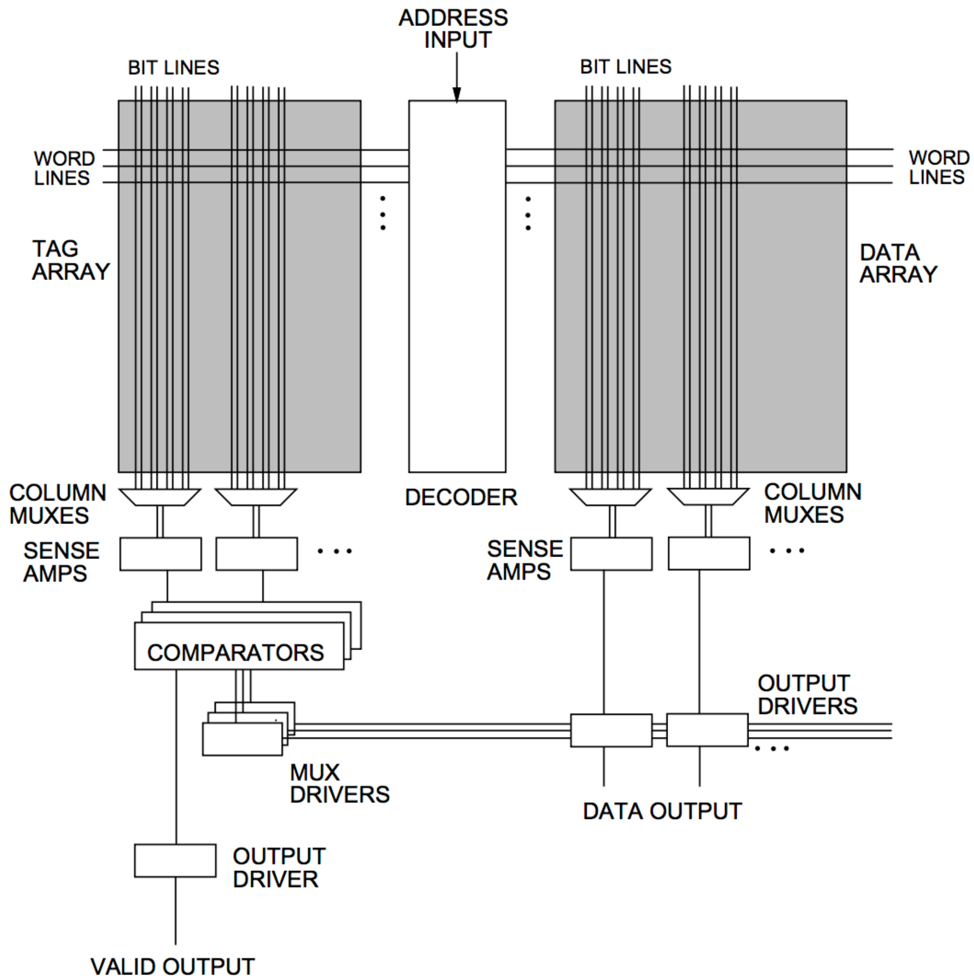
# Recall: Cache Lookup

- Byte offset: to select the requested byte
- Tag: to maintain the address
- Valid flag (v): whether content is meaningful
- Data and tag are always accessed



# Cache Architecture

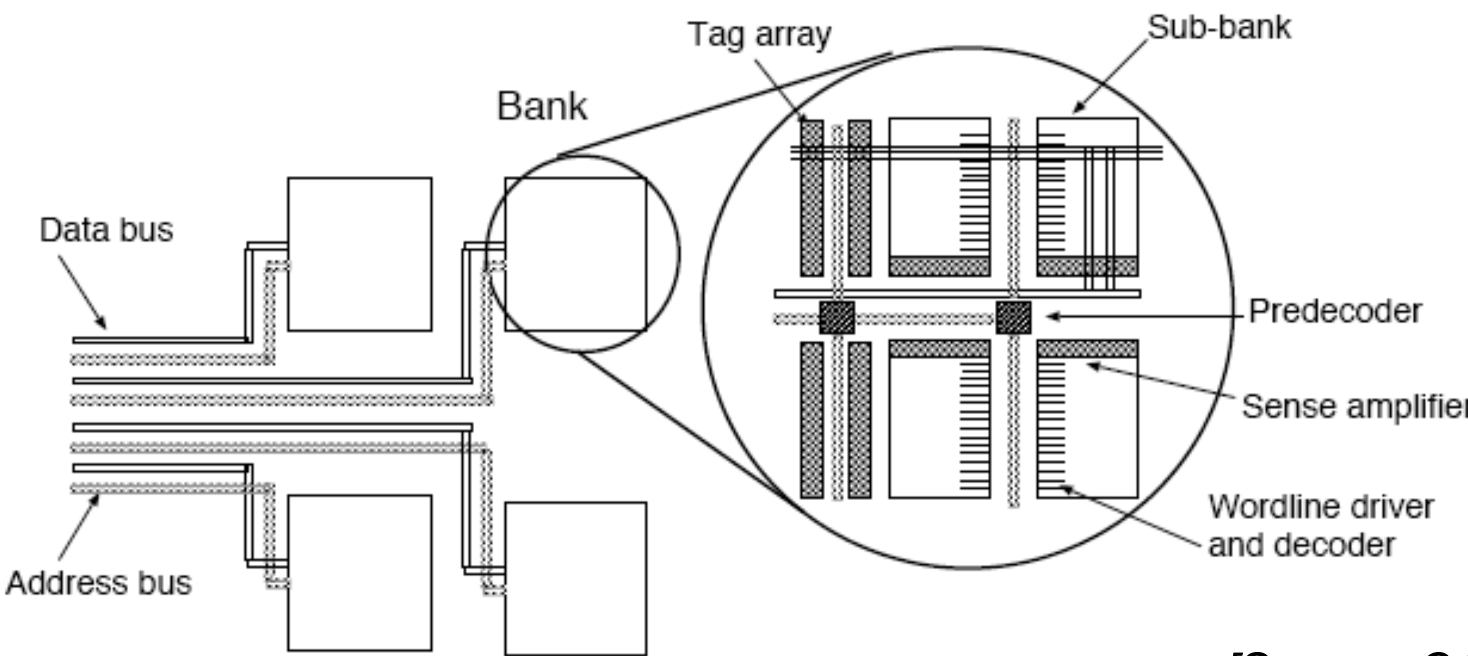
- Physical cache structure




*[CACTI 1.0]*

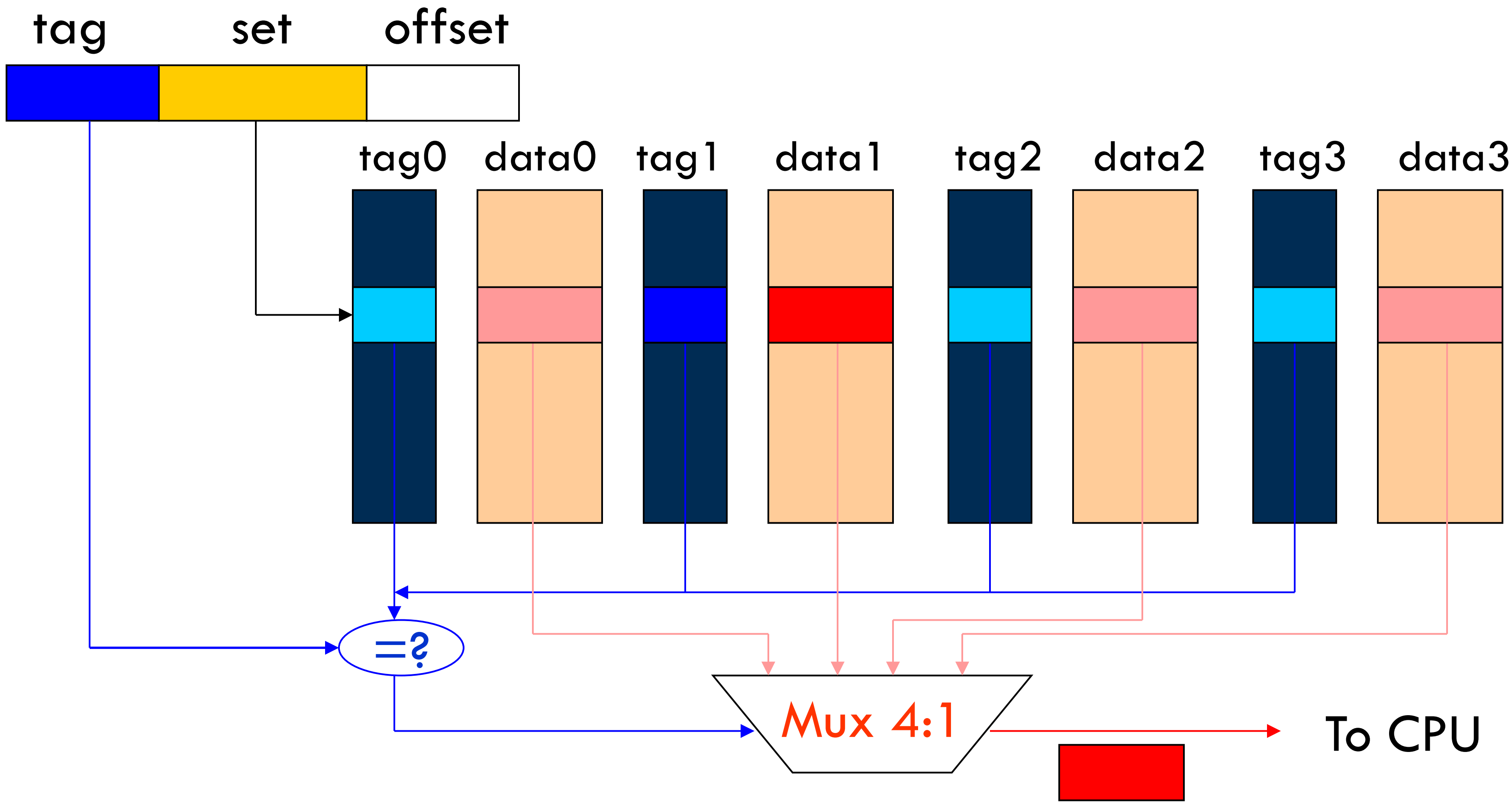
# Cache Banking

- Divide cache into multiple identical arrays
  - Static power: unused arrays may be turned off
  - Dynamic power: only the target arrays is accessed



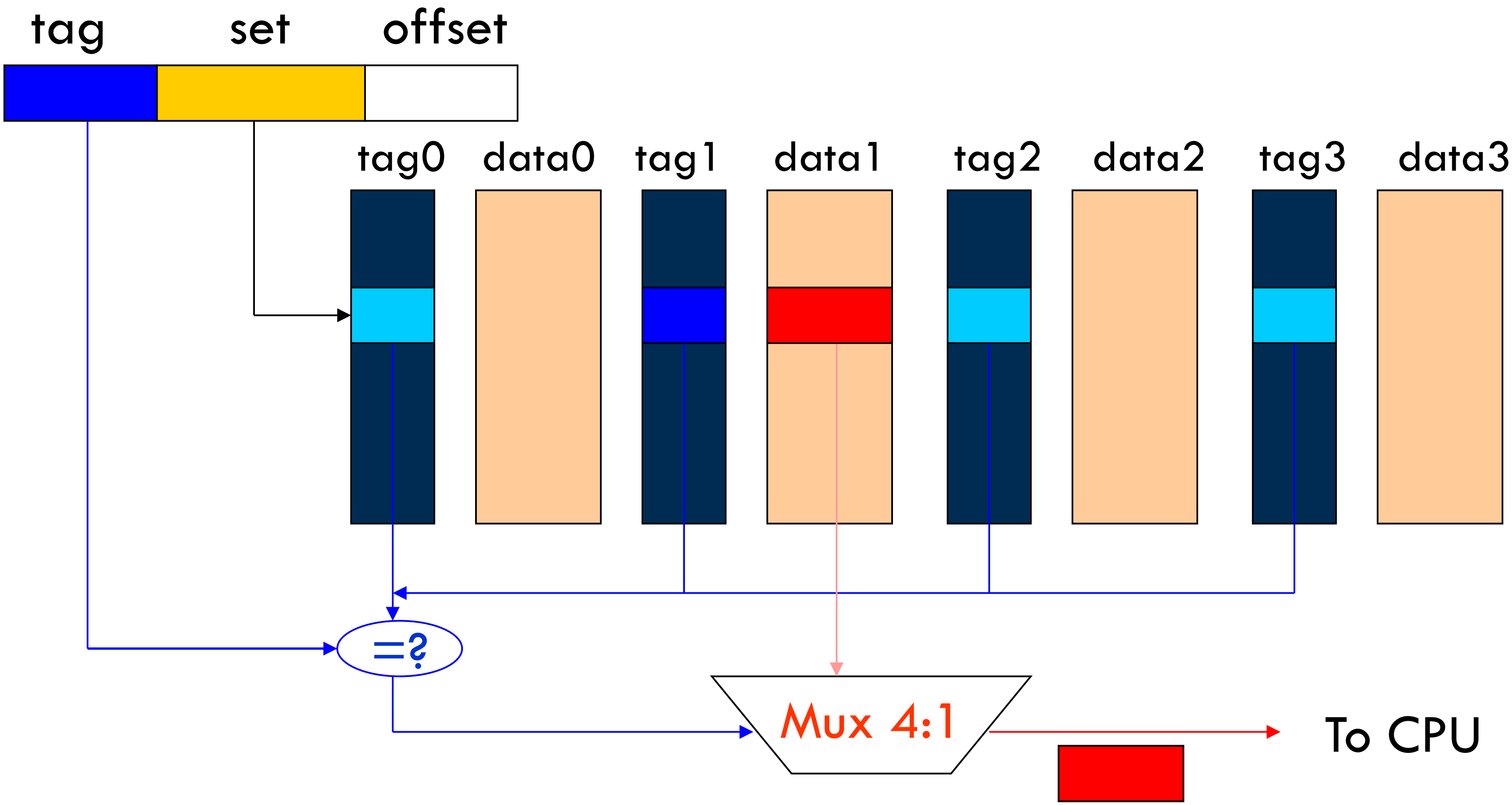

*[Source: CACTI]*

# Basic Set Associative Cache



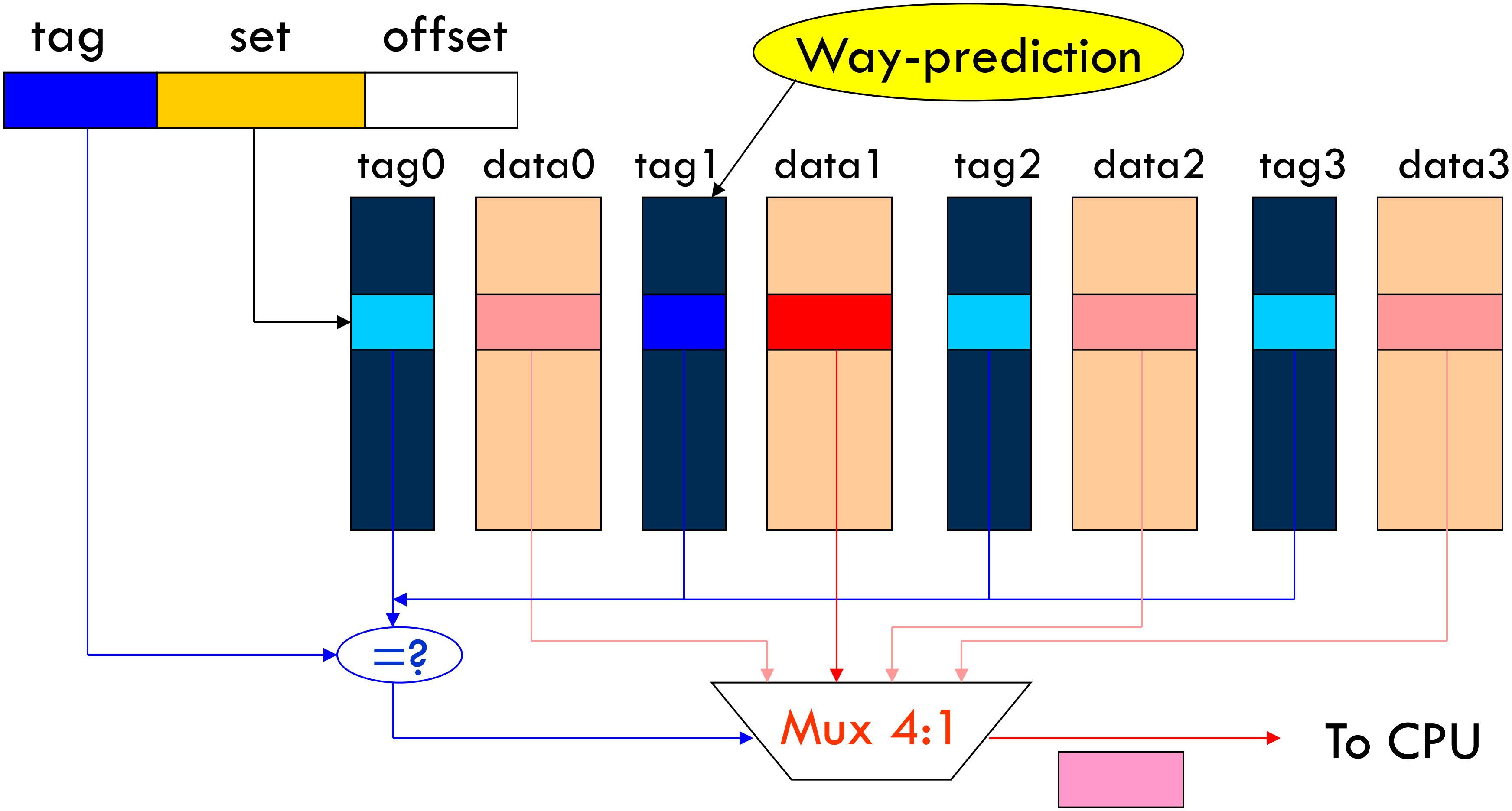

**Power per access: 4T + 4D**

# Phased N-way Cache




**Power per access: 4T + 1D**
**But access time increases**

# Way-prediction N-way Cache




**Correct prediction: 1T + 1D**
**Predict instead of sequential tag access**

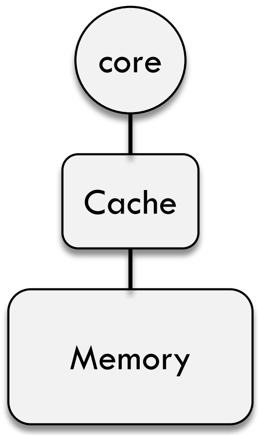
*[Powell02]*

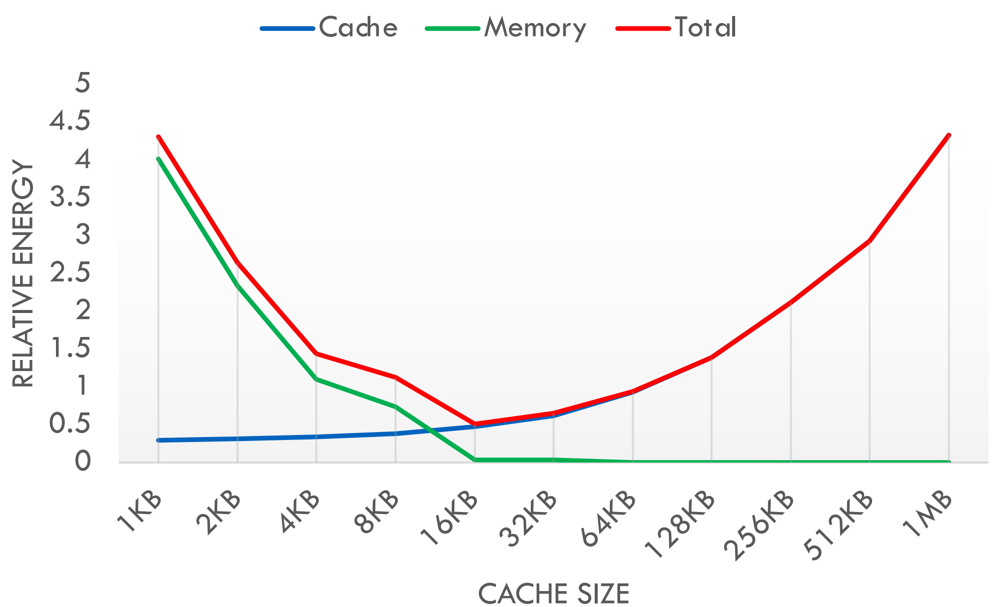
# Way Prediction Summary

- To improve hit time, predict the way to pre-set Mux
  - Mis-prediction gives longer hit time
  - Prediction accuracy
    - > 90% for two-way
    - > 80% for four-way
    - I-cache has better accuracy than D-cache
  - First used on MIPS R10000 in mid-90s
  - Used on ARM Cortex-A8
- Extend to predict block as well
  - "Way selection"
  - Increases mis-prediction penalty

# Cache Size

- Energy dissipation of on-chip cache and off-chip memory



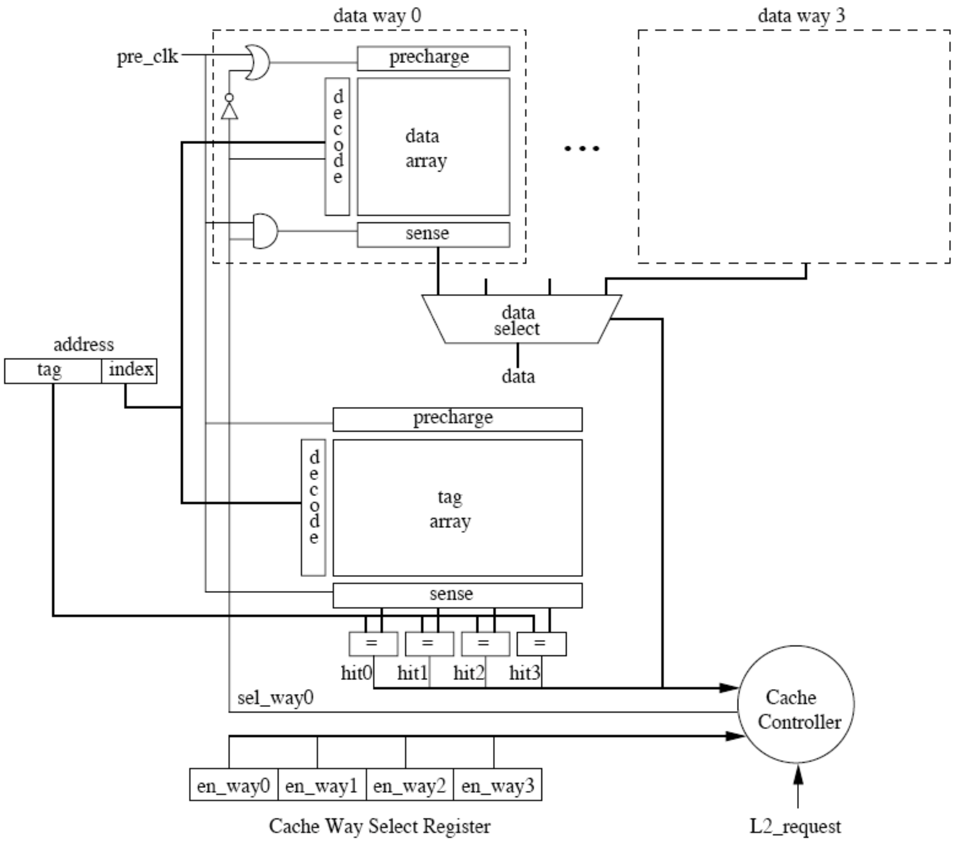




**Can we dynamically resize cache? Ways, sets, or blocks?**

*[Zhang04]*

# Resizable Caches

- Resizable caches turn off portions of the cache that are not heavily used by the running program

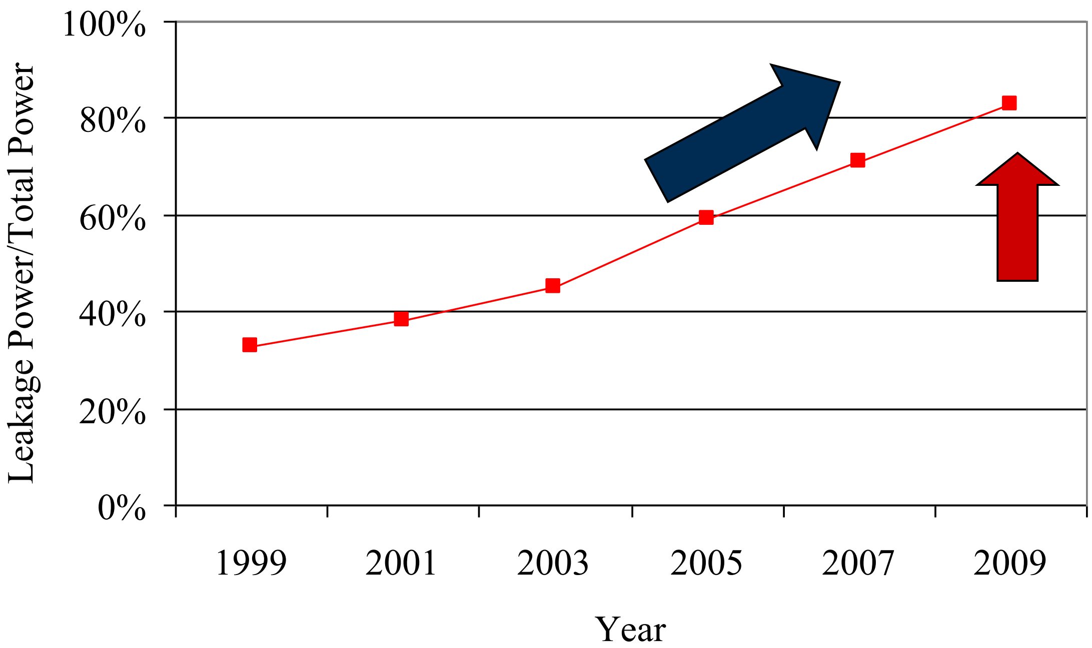



*[Albonesi99]*

# Leakage Power

- dominant source for power consumption as technology scales down

$$P_{leakage} = V \times I_{Leakage}$$



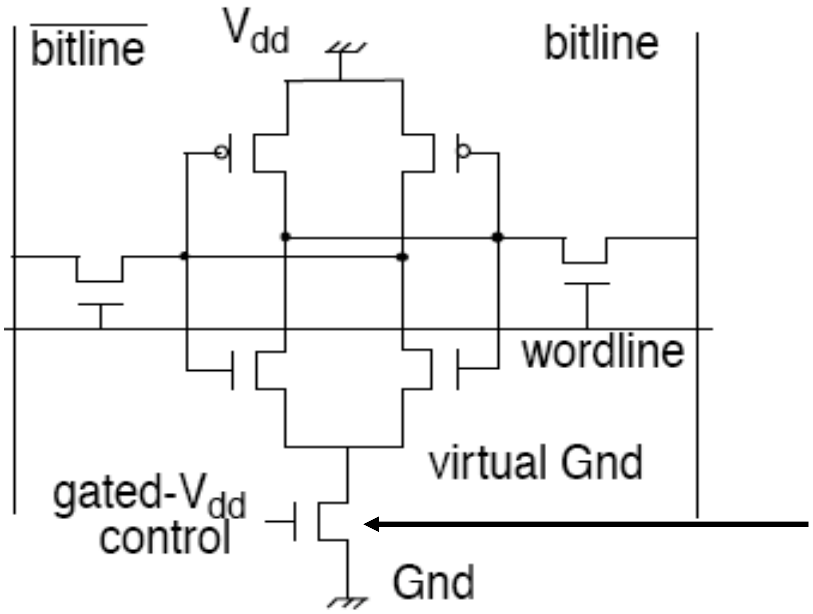

*[source of data: ITRS]*

# Dynamic Techniques for Leakage

- Three example microarchitectural approaches
  - Gated-Vdd
    - Gate the supply-to-ground path
  - Cache decay
    - Same gating mechanism but different control policy
  - Drowsy caches
    - Reduce the Vdd in order to retain cell state

# Gated Vdd

- Dynamically resize the cache (number of sets)
- Sets are disabled by gating the path between Vdd and ground ("stacking effect")

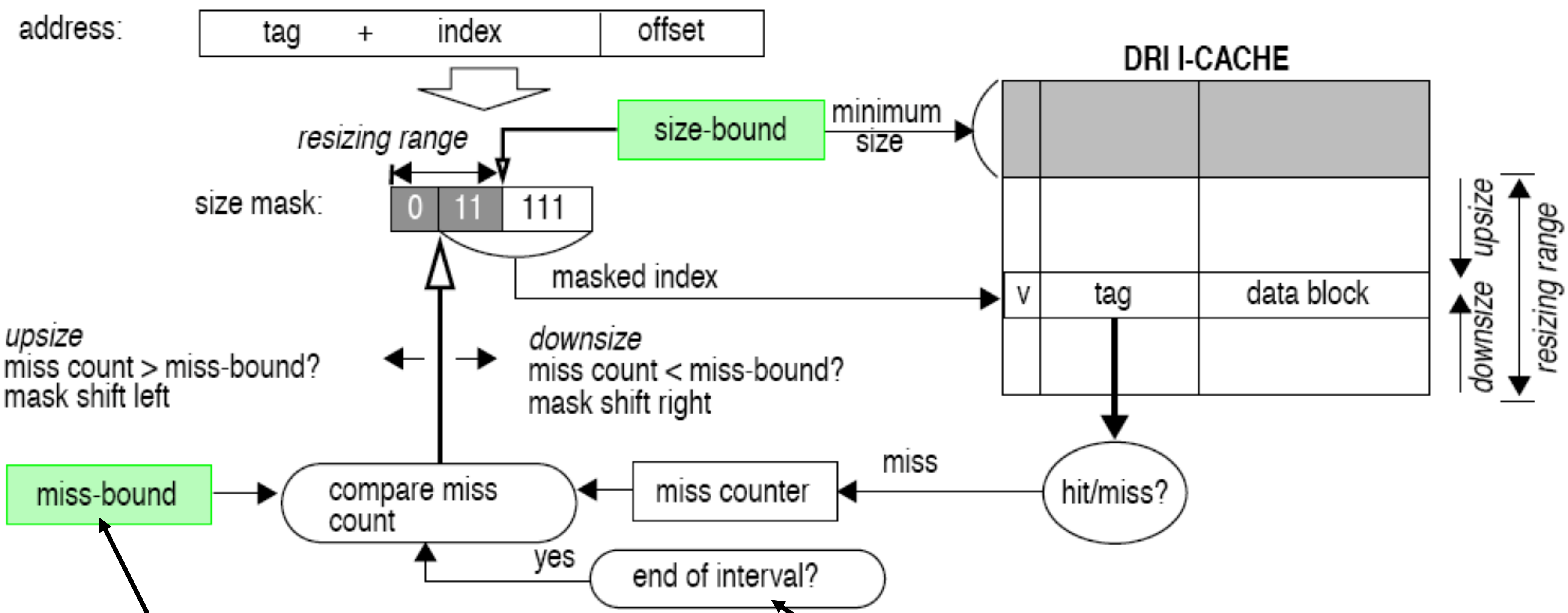



other possibilities, e.g., virtual Vdd (see paper)

shared among cells in same row (5% total area cost)

*[Powell00]*

# Gated Vdd Microarchitecture



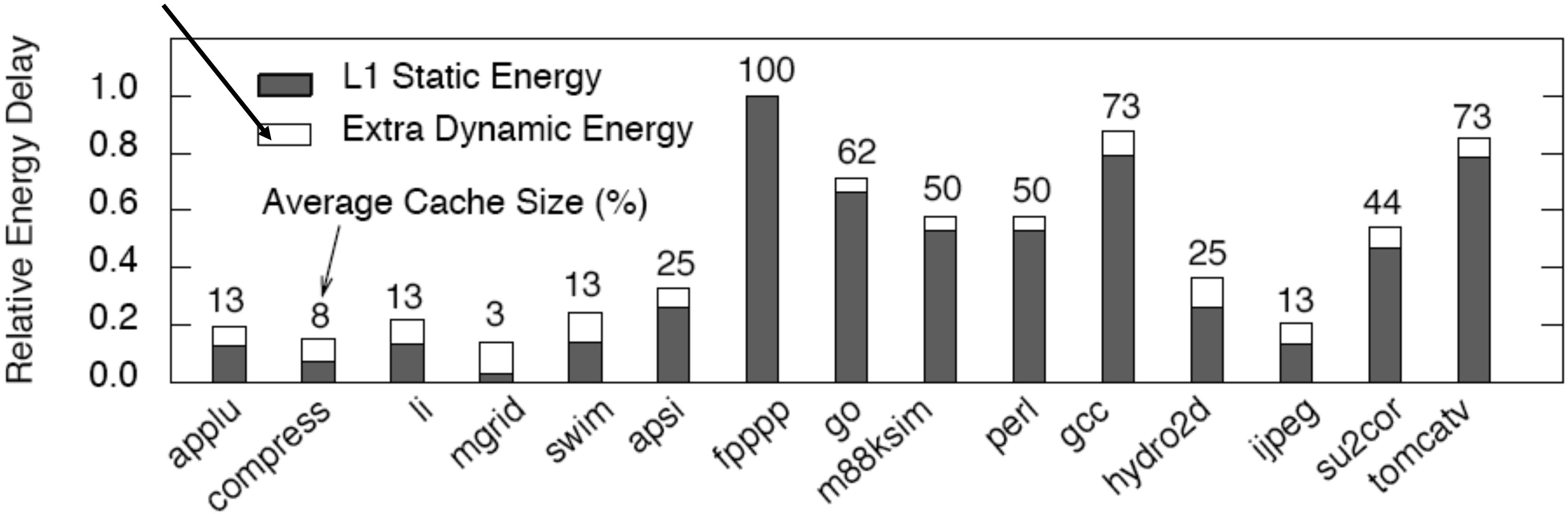

threshold above/below which cache is upsized/downsized

number of instructions between resizings

[Powell00]

# Gated-Vdd I$ Effectiveness

due to additional misses

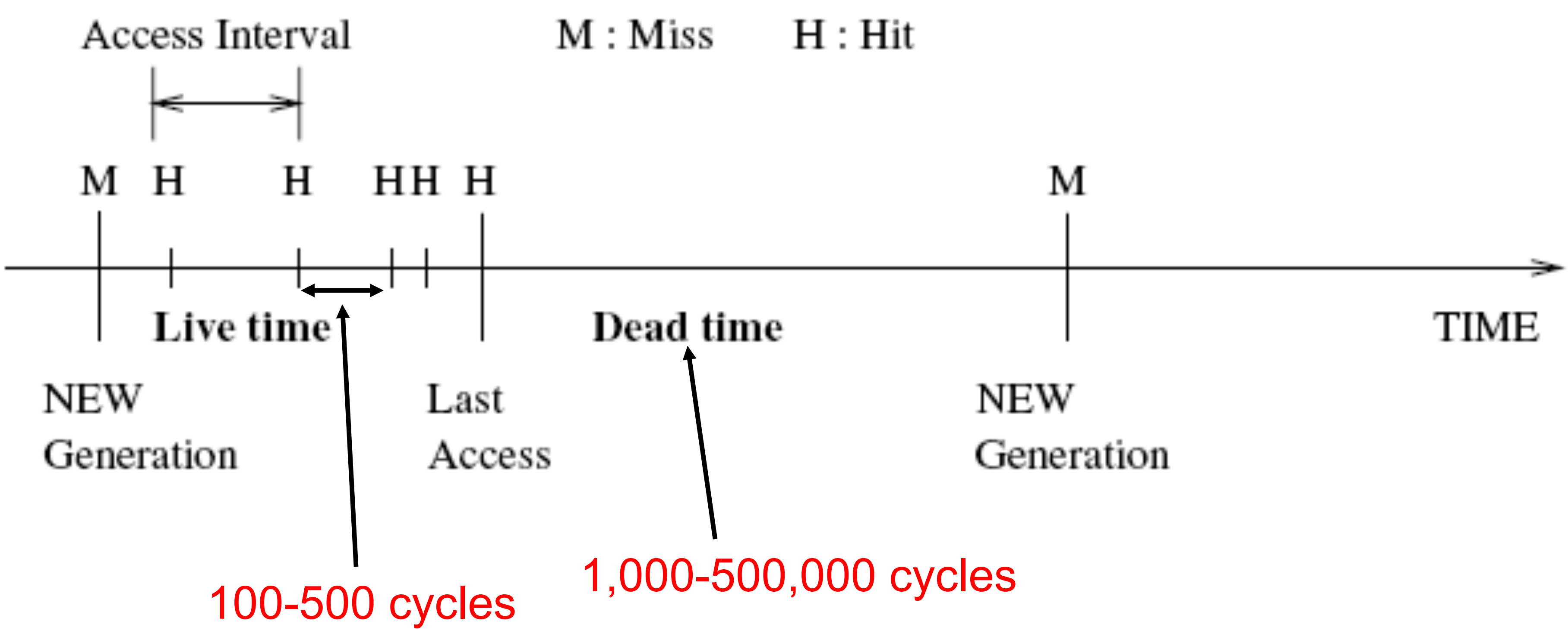



**High mis-predication costs!**

*[Powell00]*

# Cache Decay

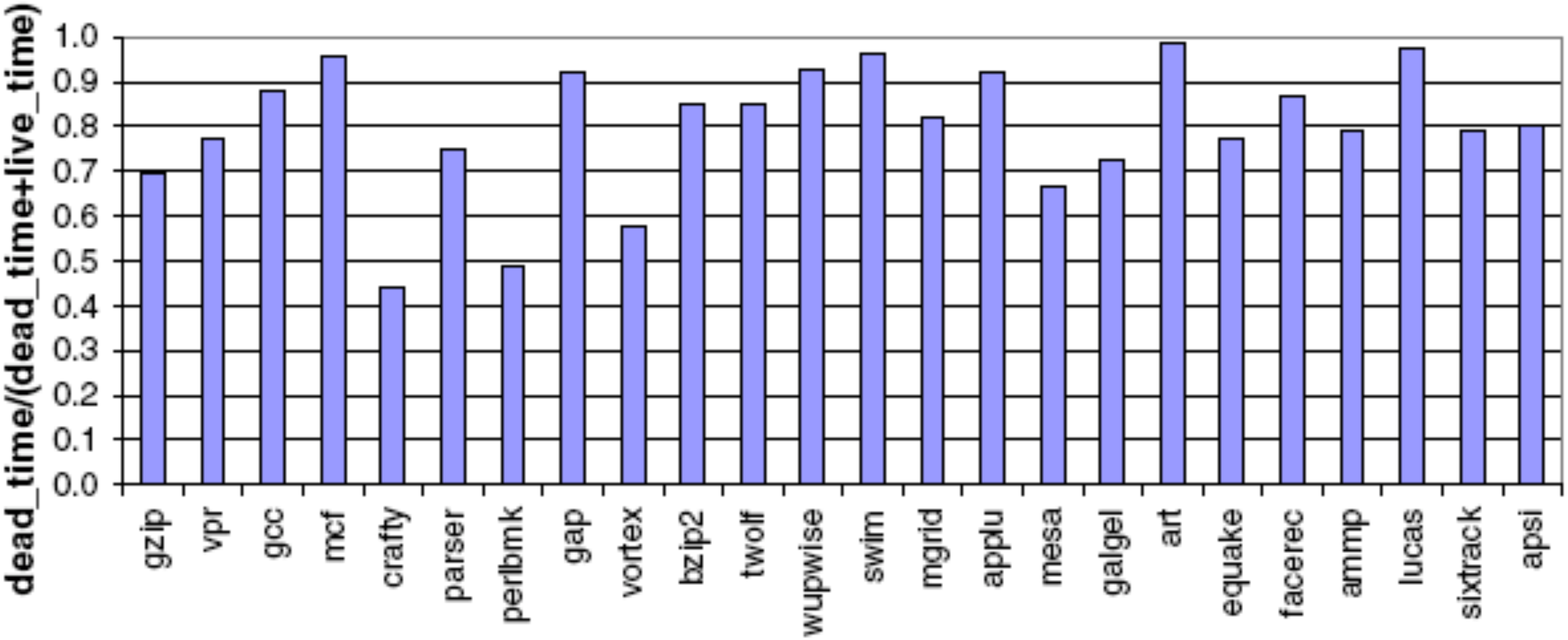
- Exploits generational behavior of cache contents




*[Kaxiras01]*

# Cache Decay

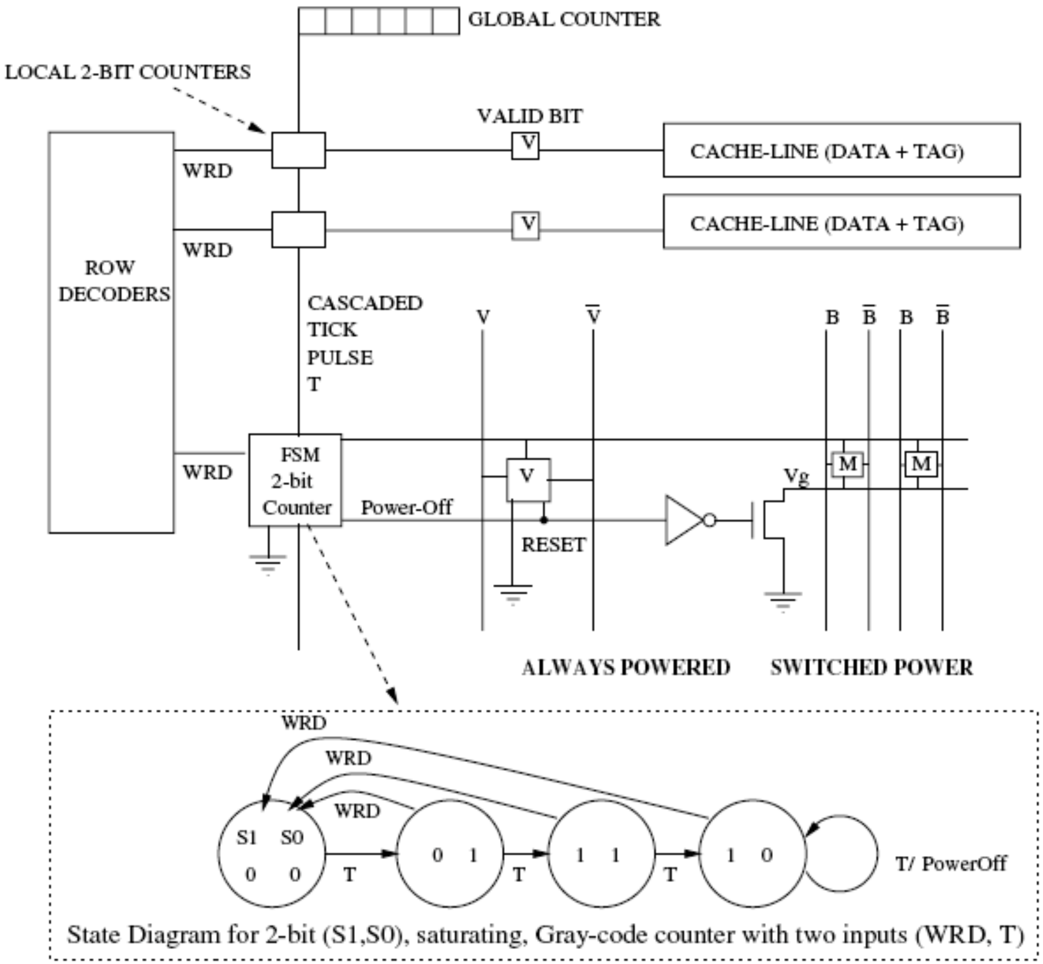
- Fraction of time cache lines that are "dead"




32KB L1 D-cache

*[Kaxiras01]*

# Cache Decay Implementation




**High mis-predication costs!**

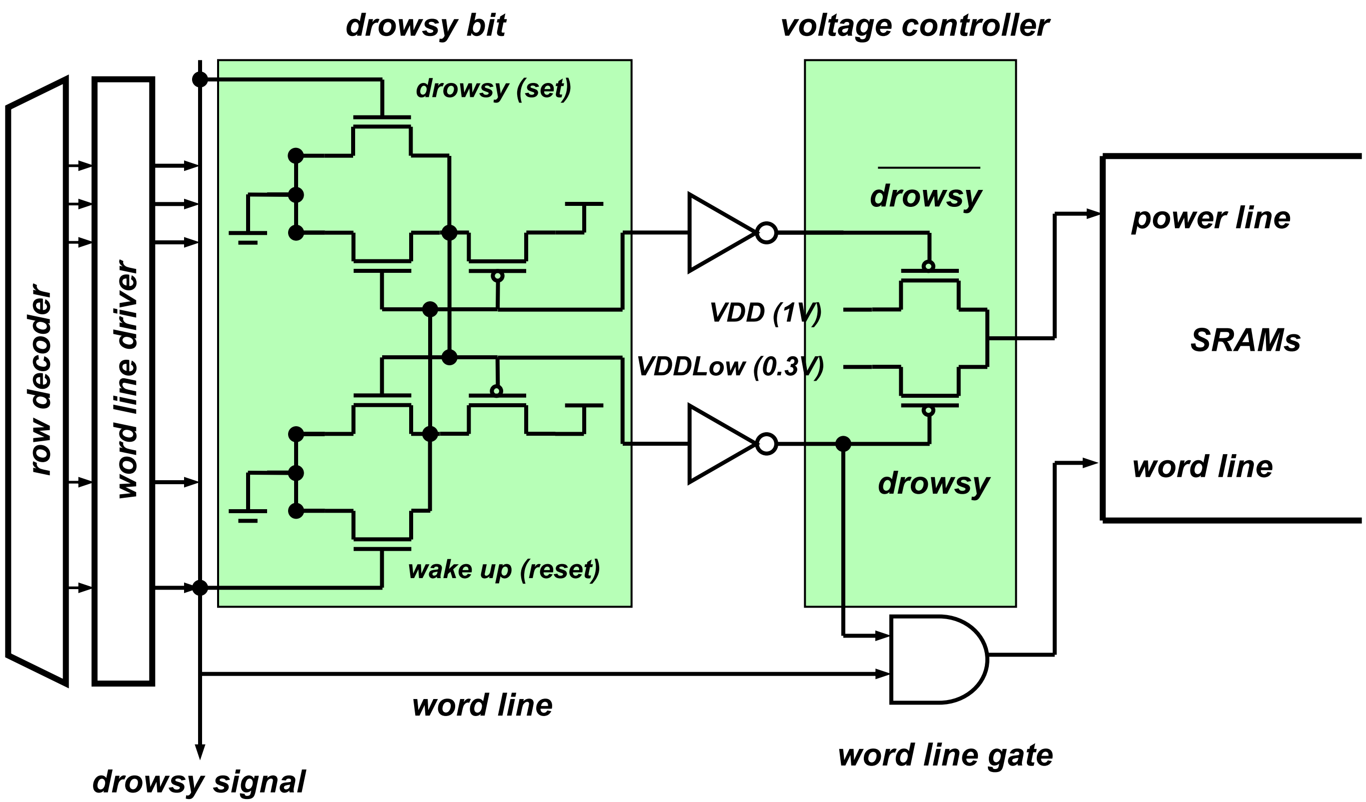
*[Kaxiras01]*

# Drowsy Caches

- Gated-Vdd cells lose their state
  - Instructions/data must be refetched
  - Dirty data must be first written back

- By dynamically scaling Vdd, cell is put into a drowsy state where it retains its value
  - Leakage drops superlinearly with reduced Vdd ("DIBL" effect)
  - Cell can be fully restored in a few cycles
  - Much lower misprediction cost than gated-Vdd, but noise susceptibility and less reduction in leakage

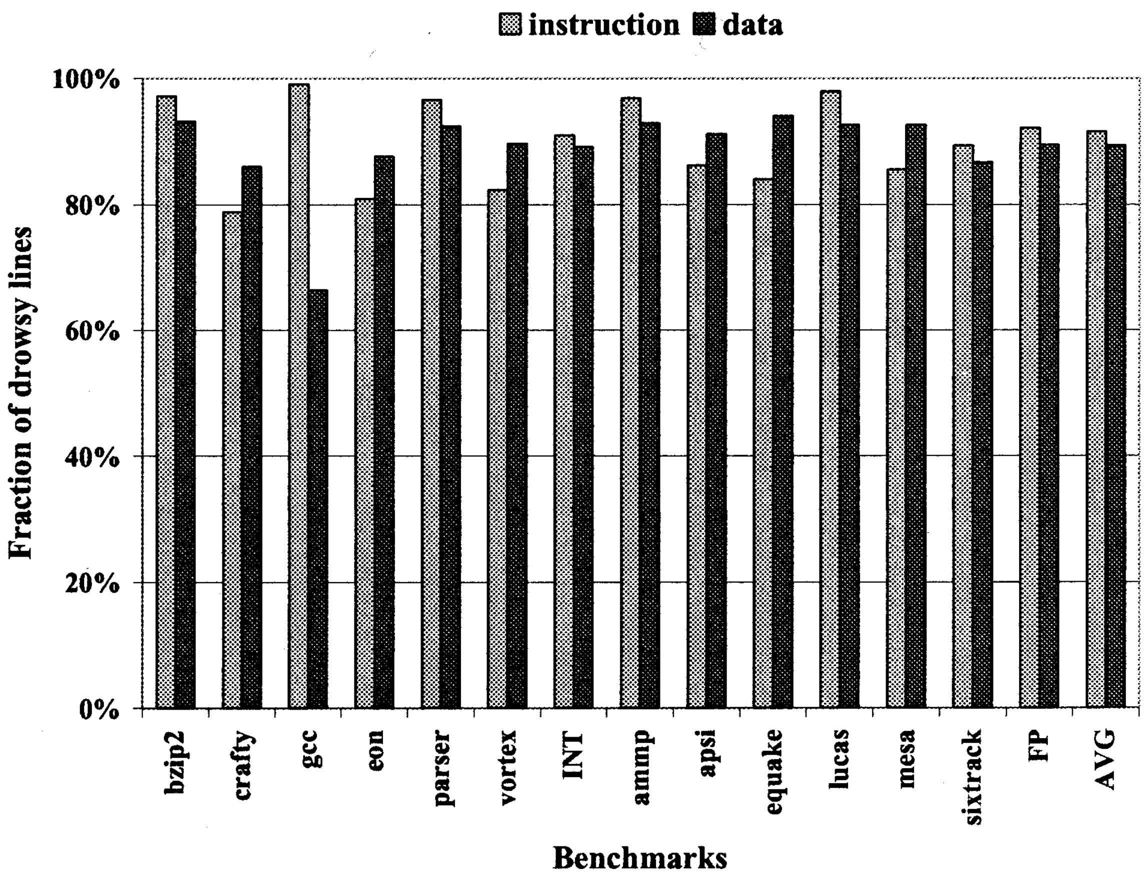
# Drowsy Cache Organization




**Keeps the contents (no data loss)**

*[Kim04]*

# Drowsy Cache Effectivenes

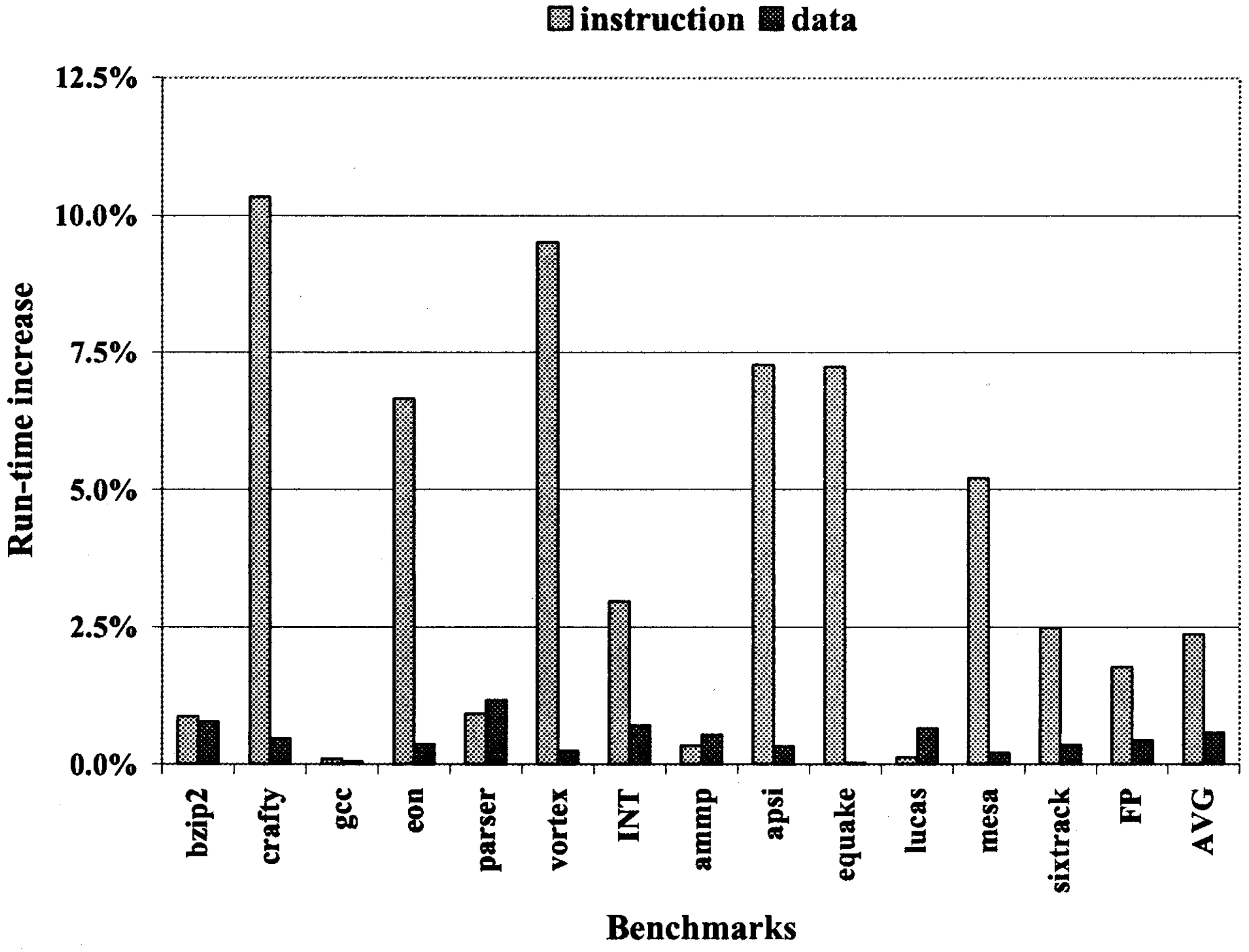



32KB L1 caches

4K cycle drowsy period

*[Kim04]*

# Drowsy Cache Performance Cost




*[Kim04]*