ECE 5730
Memory Systems
Spring 2009

Cache Power Management

Cornell University

Announcements

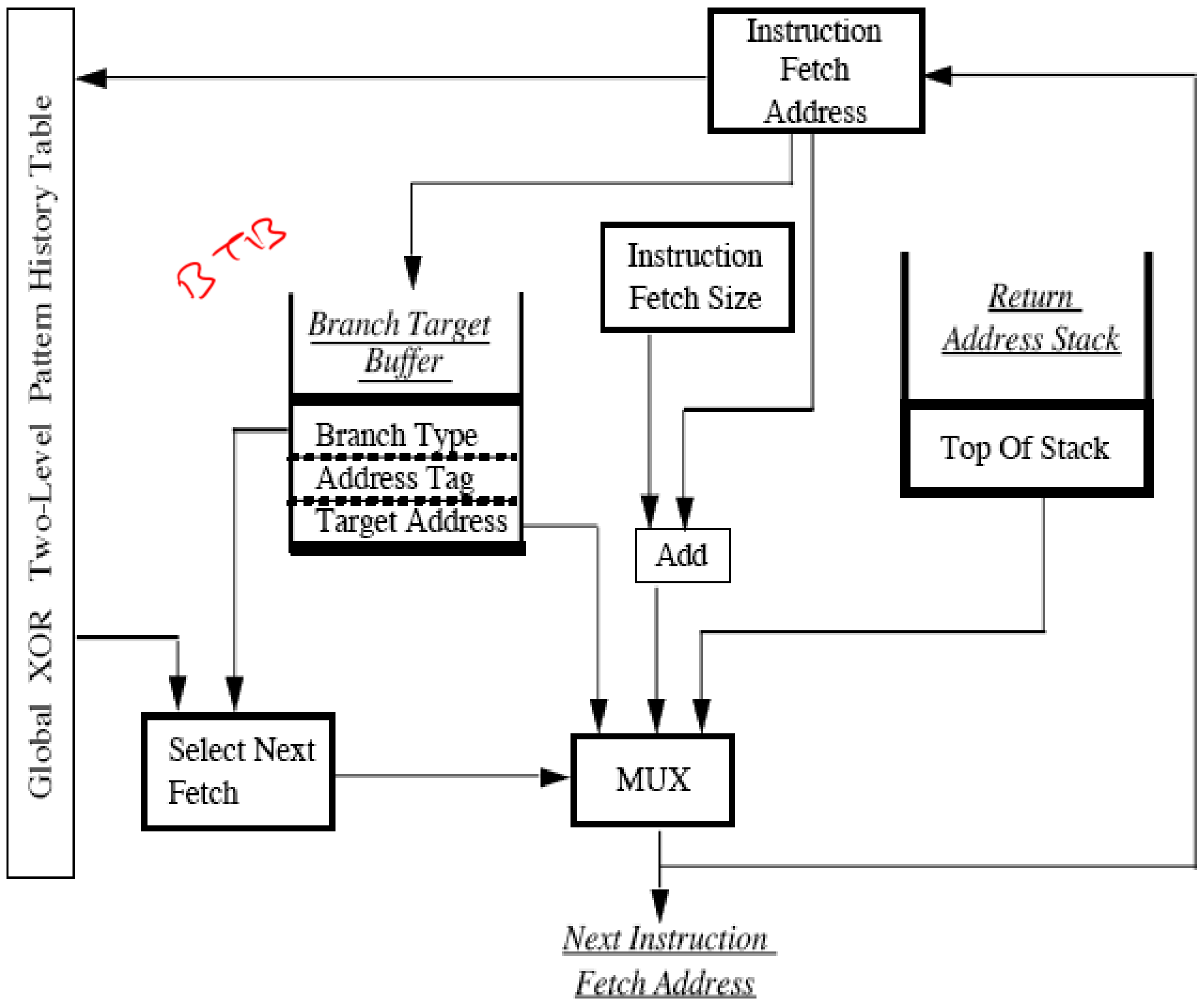
- Quiz 5 on Tuesday

- Quiz 4
  - Average = 8.9/10

- No office hours today

I-cache Way Prediction

- Conventional instruction fetch mechanism

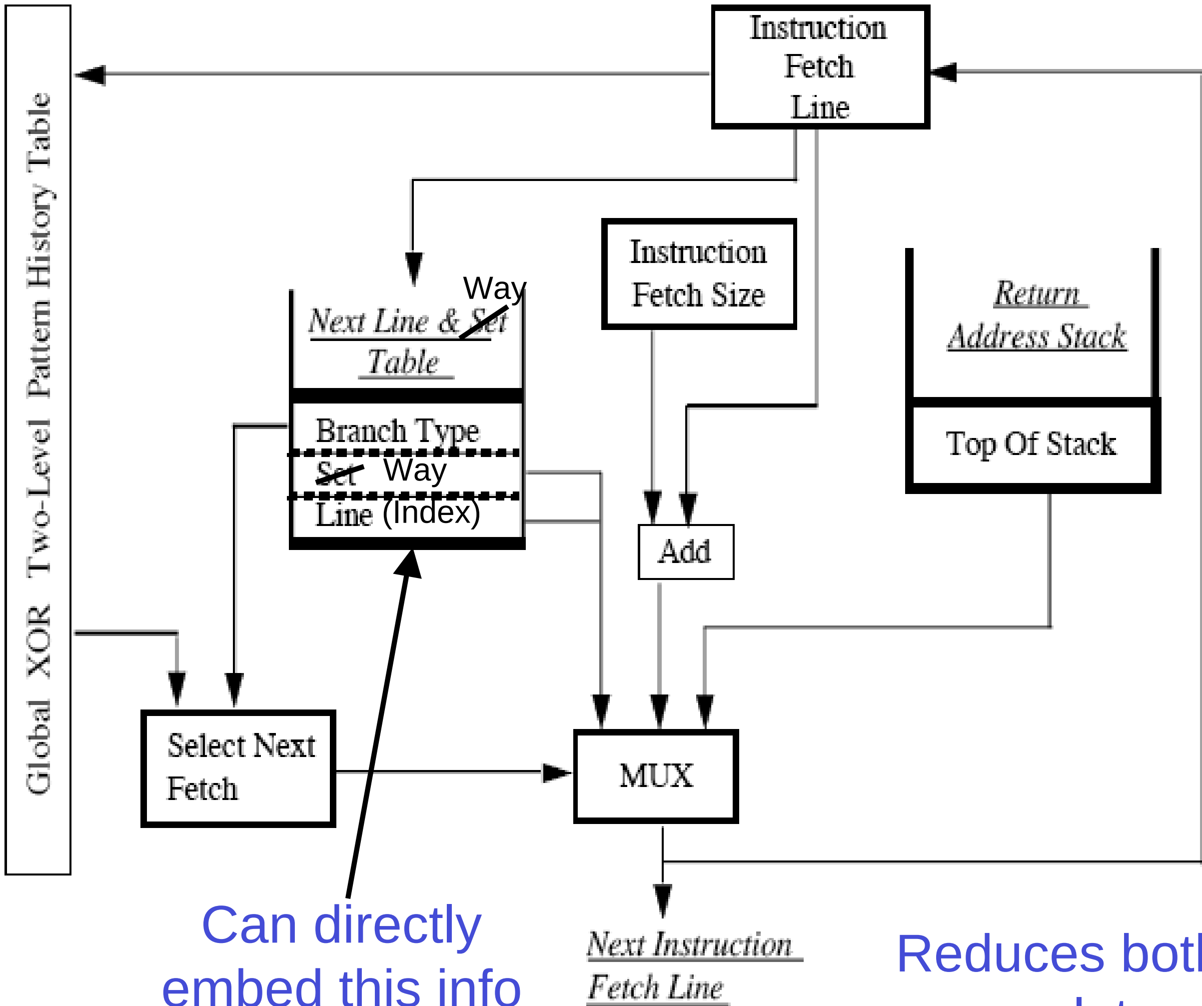


[Calder95]

I-cache Way Prediction

↙ BTB does not predict cache stuff. This scheme fixes that by replacing the BTB

- **Predicting the way, and the next line *index***



Can directly embed this info into the I-cache

Reduces both access latency *and* energy!

[Calder95]

Cache Power Management

- Circuit techniques
  - Transistor sizing, multi-V_t, low-swing bitlines, etc
- Microarchitecture techniques
  - Static techniques: banking, phased tag/data access, way prediction, CAM-tags
  - today Dynamic techniques: gated-Vdd, cache decay, drowsy caches
- Compiler techniques
  - Data partitioning to enable sleep mode

Dynamic Techniques for Leakage

- **Static microarchitectural techniques address dynamic power by reducing activity**

- **But lots of leaking idle cache rows!**

leakage current causes power loss (up to 50% in a cache)

- **Three example microarchitectural approaches**
  - ***Gated-Vdd***
    - **Gate the supply-to-ground path** → adds more transistors in the way to minimize leakage
  - ***Cache decay***
    - **Same gating mechanism as Gated-Vdd but different control policy**
  - ***Drowsy caches*** → prevents dynamic power loss (no misses since data is retained)
    - **Reduce the Vdd in order to retain cell state**

↳ drop the voltage, retains the data

doesn't retain data

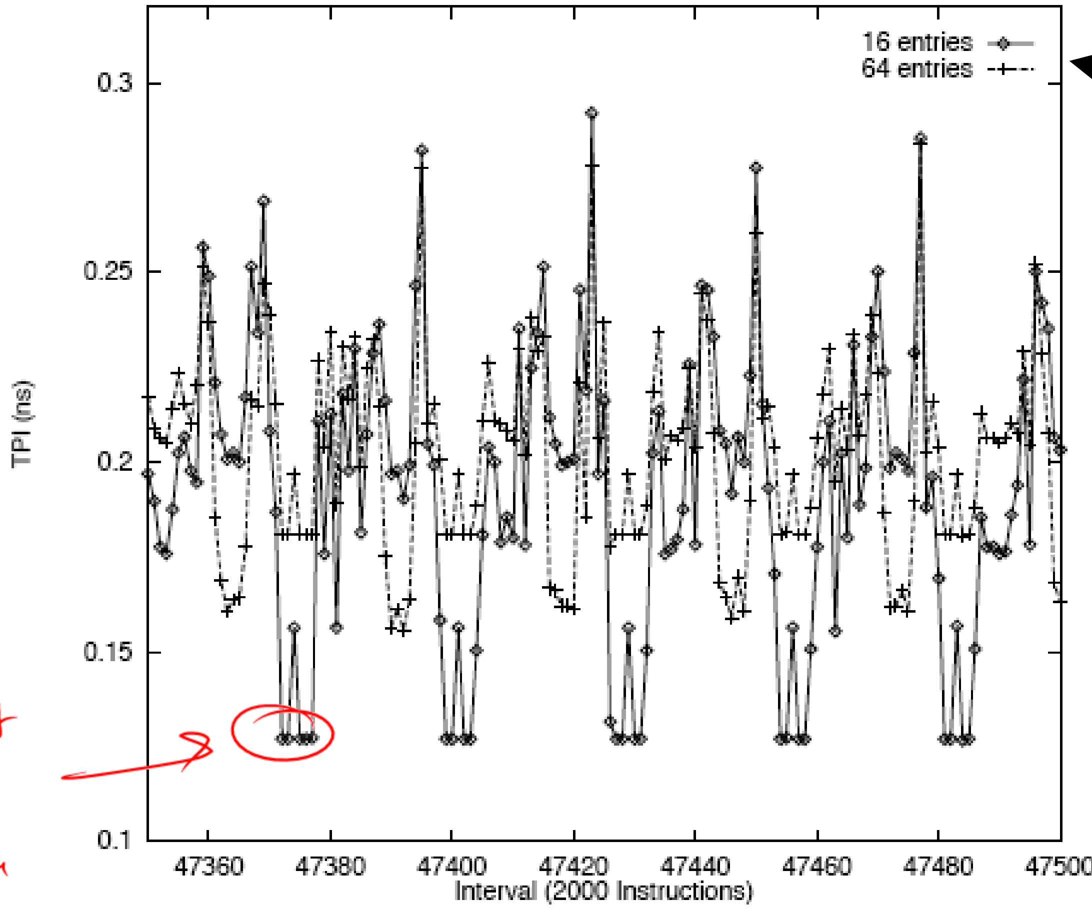
prevents static power loss, but dynamic power loss on miss.

still some wakeup latency, but faster than a miss

Resizable Hardware

- **Demand for hardware resources varies among applications, and within a single application**

time per instruction



issue queue entries

Issue queue length really is dependent on workload

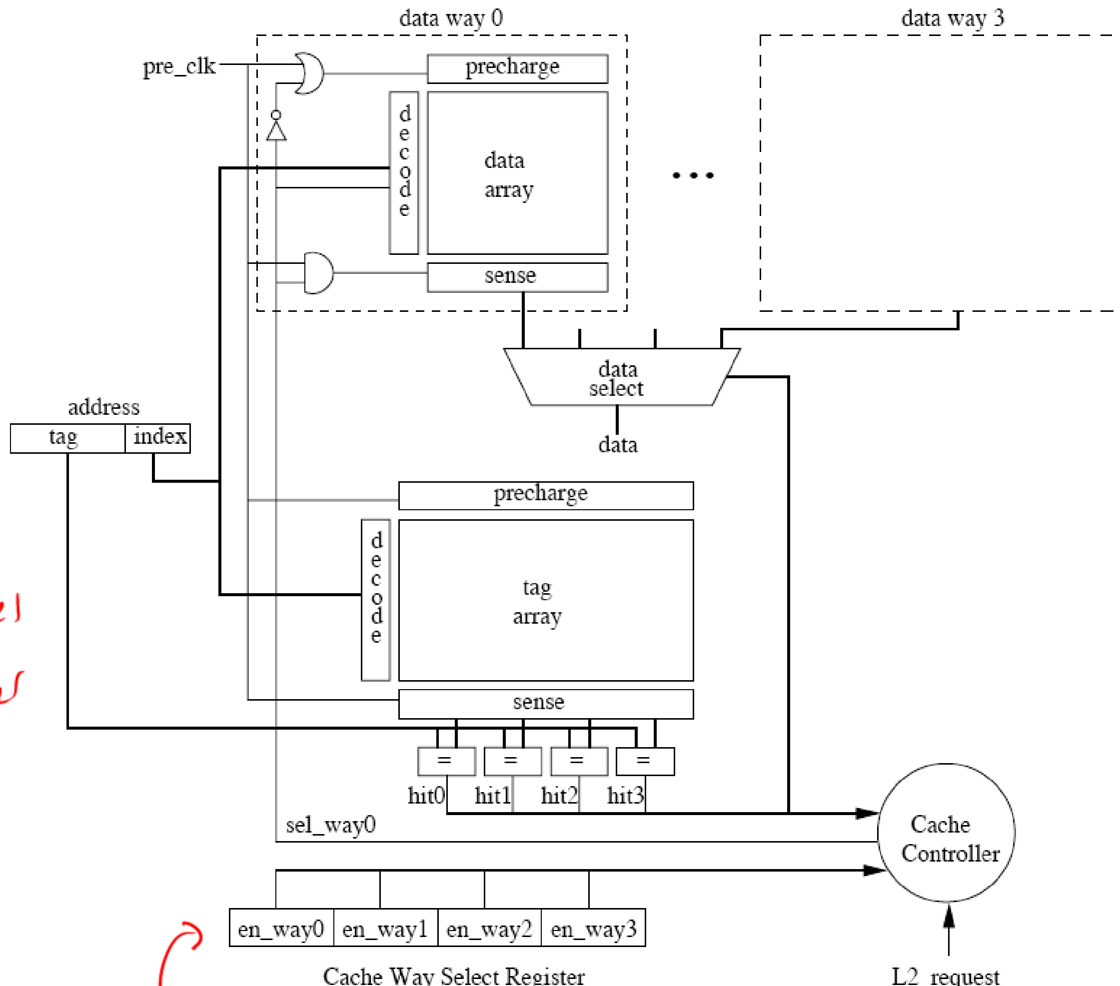
might want to use 16 entry queue here, for example

instruction intervals

[Albonesi98]

Resizable Caches

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



this addresses dynamic power

enable bits for cache ways

could thus switch from direct-mapped cache all the way to 4-way set associative.

[Albonesi99]

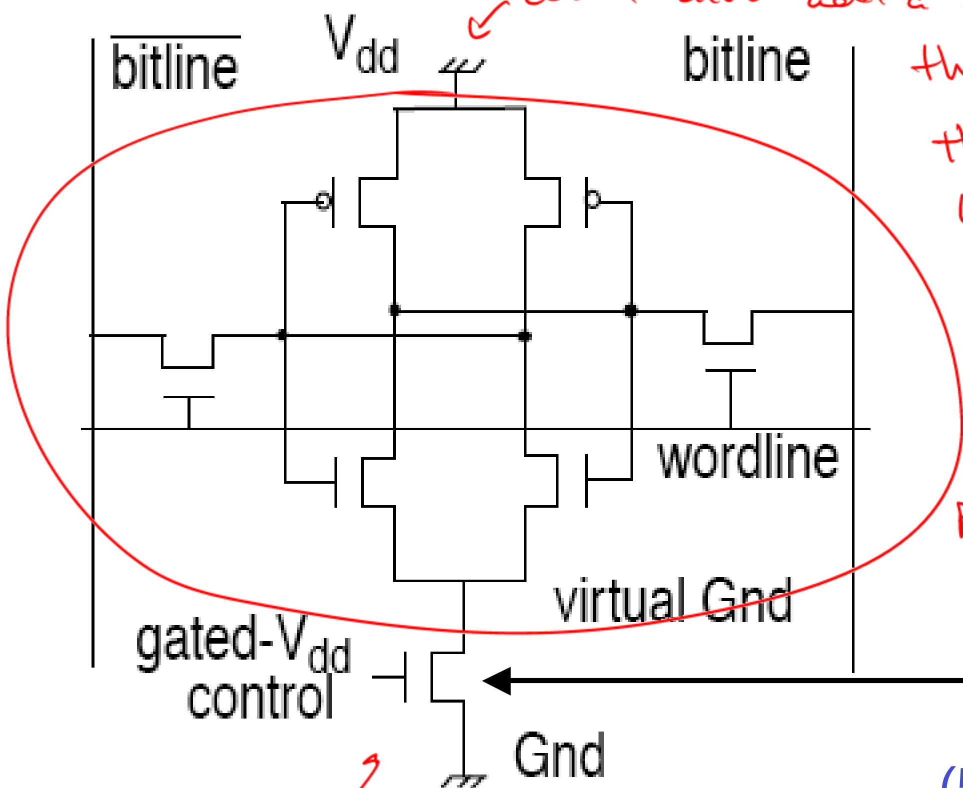
Gated-Vdd

finer grained control by set instead of way as in previous slide

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

could also add a series PMOS here

Even though the foot transistor leaks, by adding another series transistor you add series "resistance" thus limiting the leakage current



this is implemented in the ~2009 era Intel L2 caches (atom, core2)

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

standard 6T SRAM

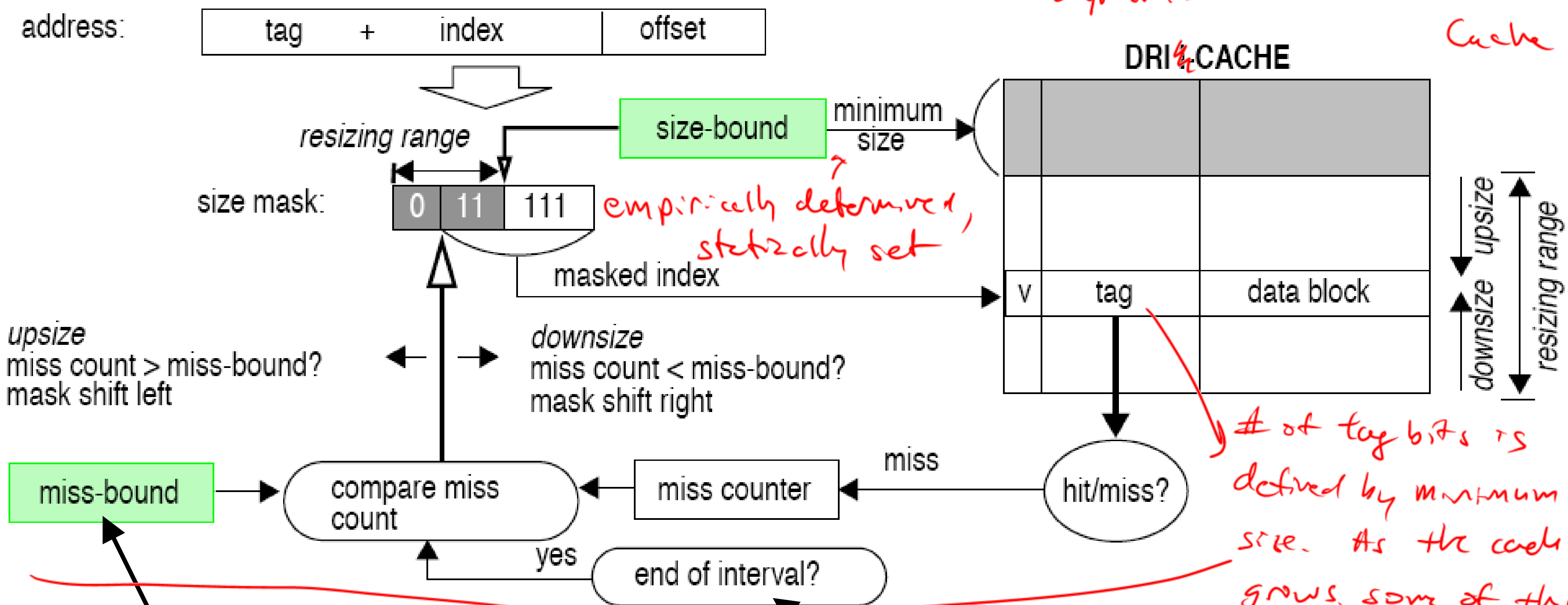
shared among cells in same row (5% total area cost)

add a foot transistor

[Powell00]

Gated-Vdd Microarchitecture

Dynamic Resizeable Instruction Cache



empirically determined, statically set

# of tag bits is defined by minimum size. As the cache grows, some of these tag bits are useless overhead

threshold above/below which cache is upsized/downsized

number of instructions between resizings

the control scheme watches misses to increase or decrease the size of the cache.

need more space → turn on fast transistor and vice versa

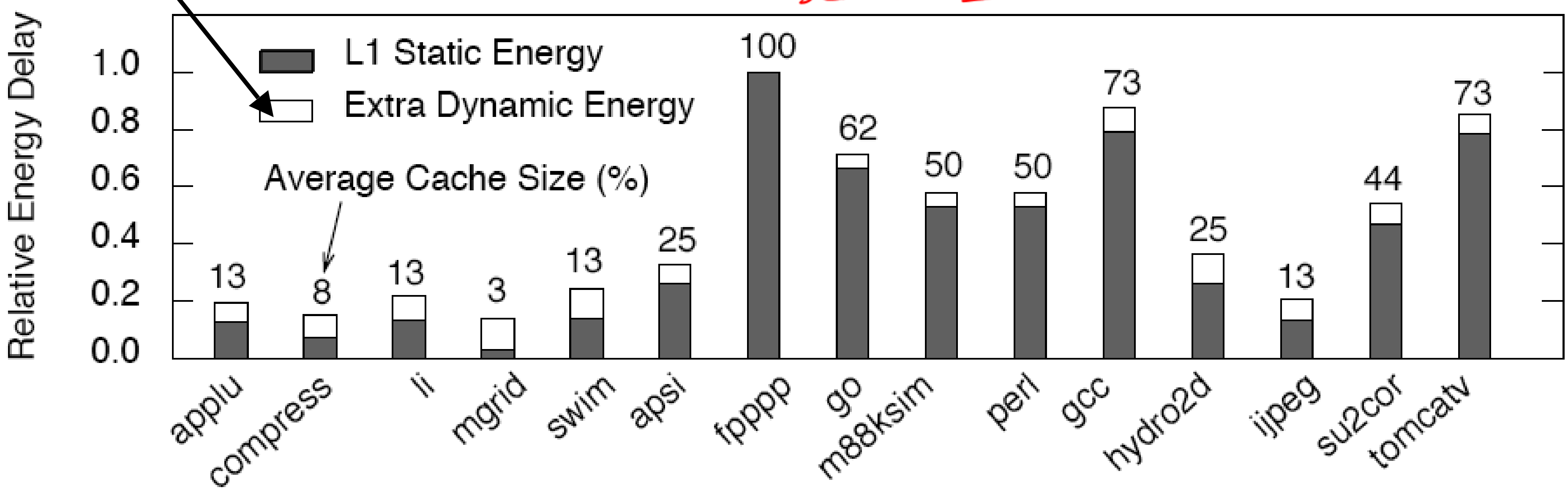
[Powell00]

Gated-Vdd I-cache Effectiveness

essentially performance/energy

due to additional misses

32kB I-cache



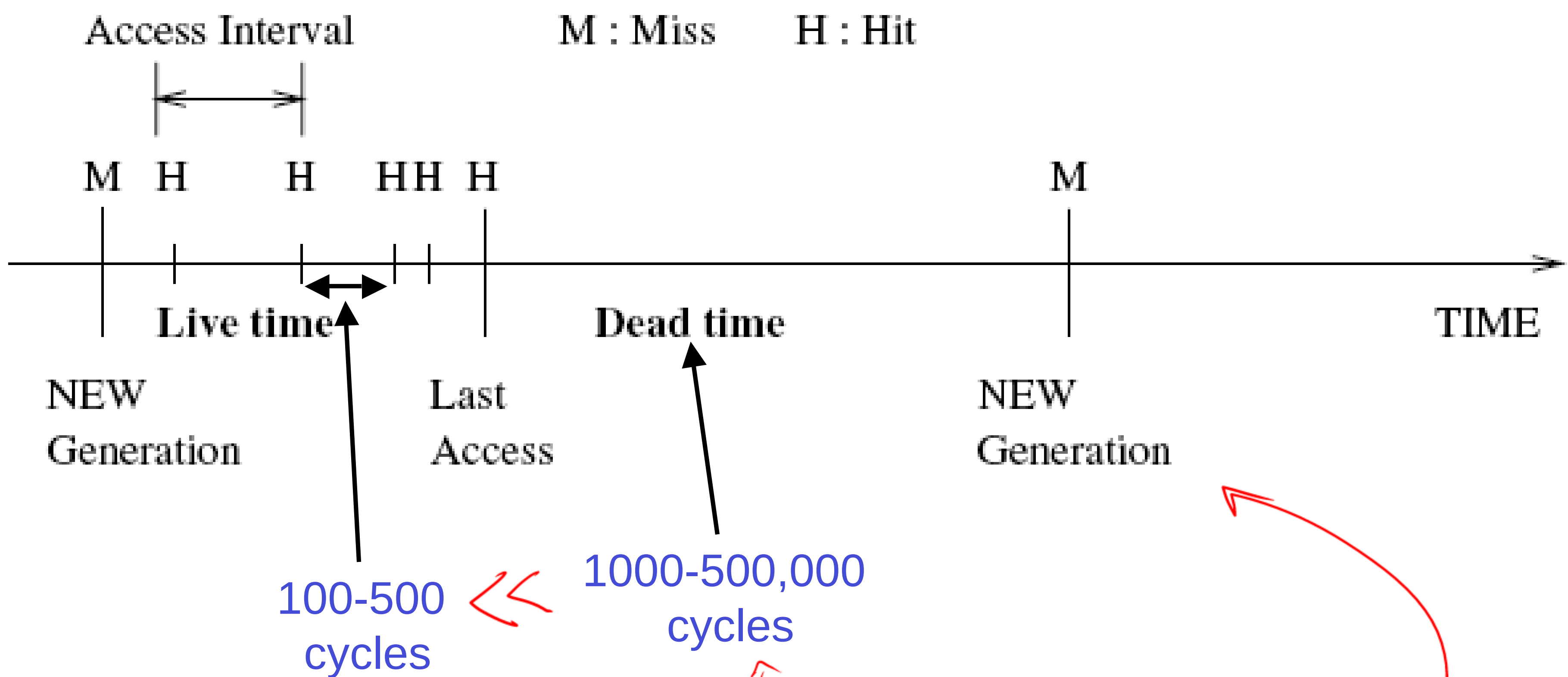
[Powell00]

over time, observe the behavior of things in a cache.

access typically occurs in bursts

Cache Decay

- Exploits *generational behavior* of cache contents



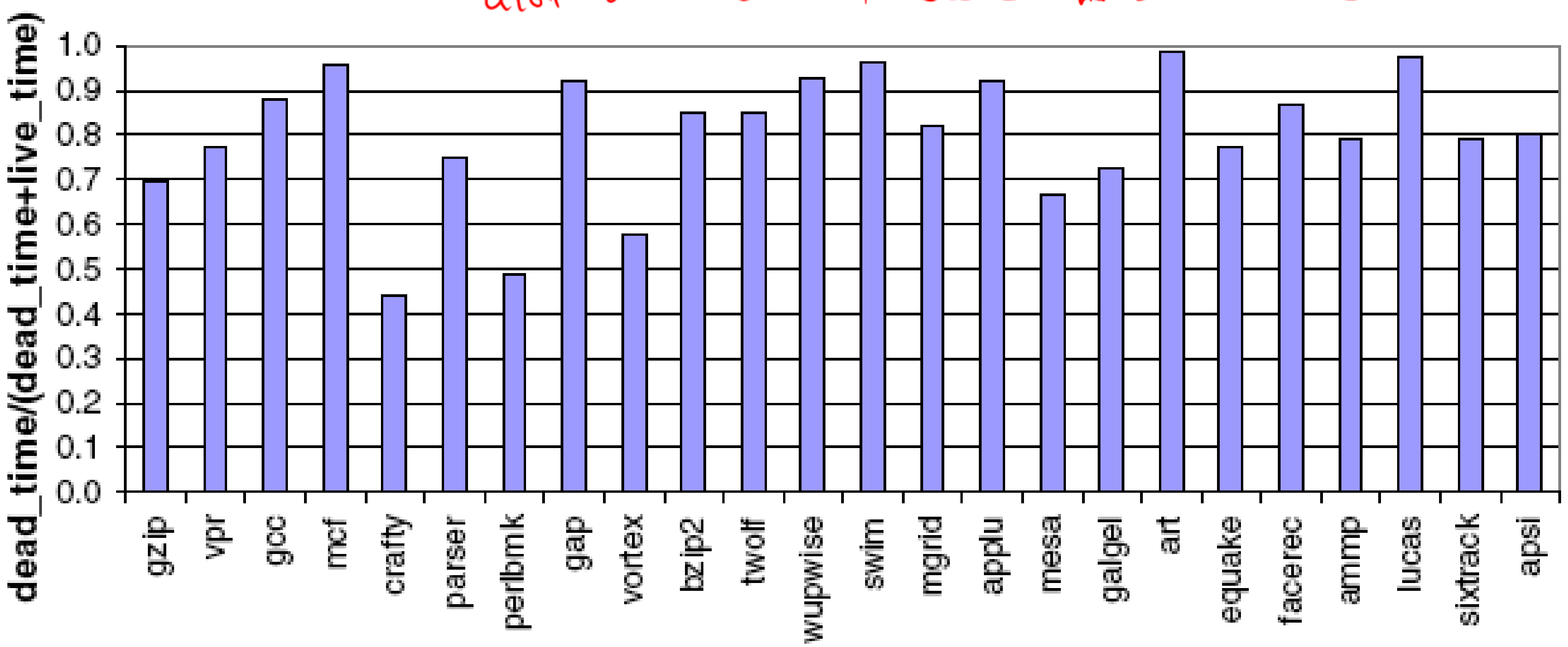
exploit this dead time, as you're probably going to miss anyway

[Kaxiras01]

Cache Decay

- **Fraction of time cache lines are "dead"**

alot of the time cache lines are dead

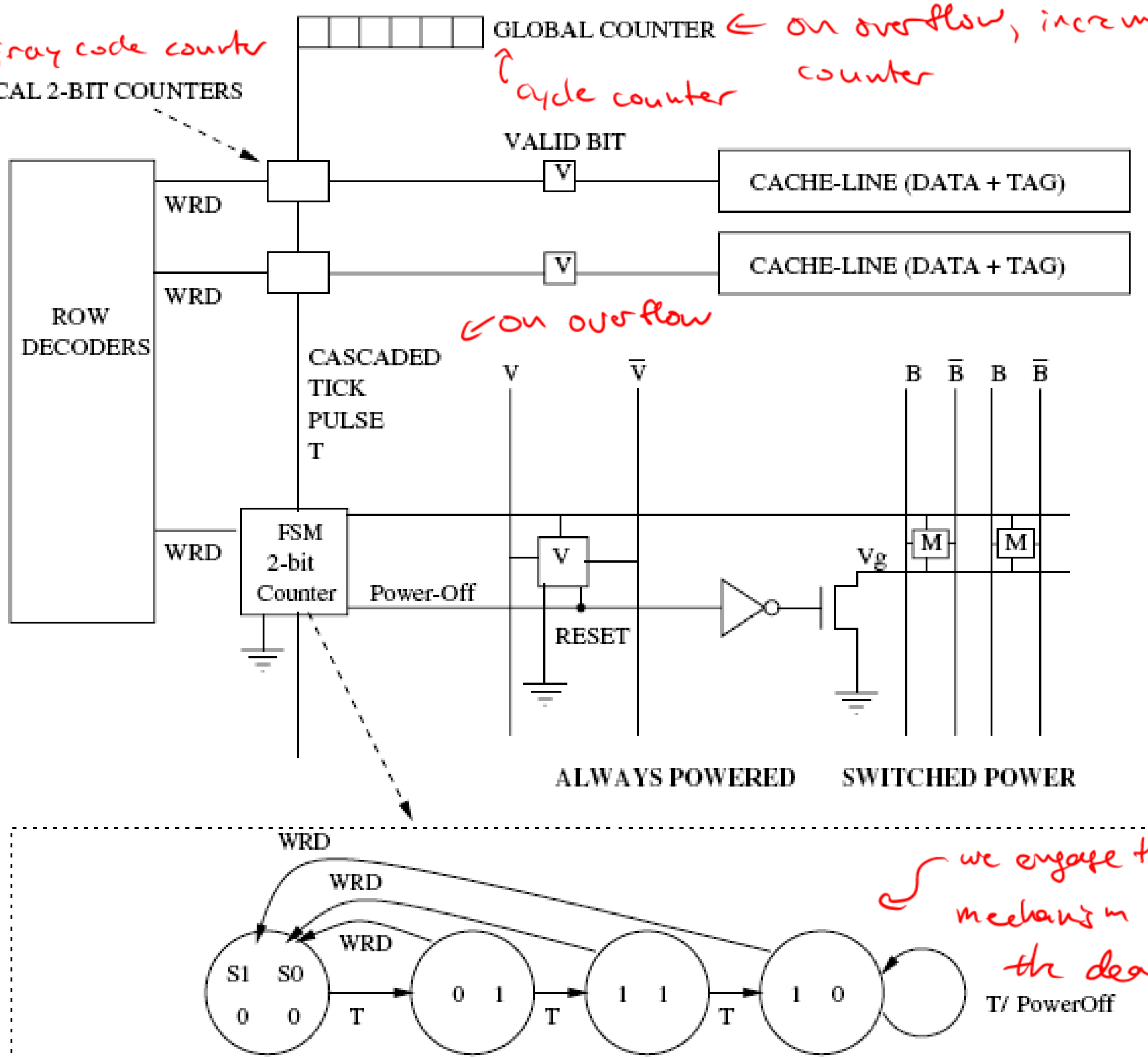


32KB L1 D-cache

[Kaxiras01]

gray code: only one bit changes on update

Cache Decay Implementation



State Diagram for 2-bit (S1,S0), saturating, Gray-code counter with two inputs (WRD, T)

[Kaxiras01]

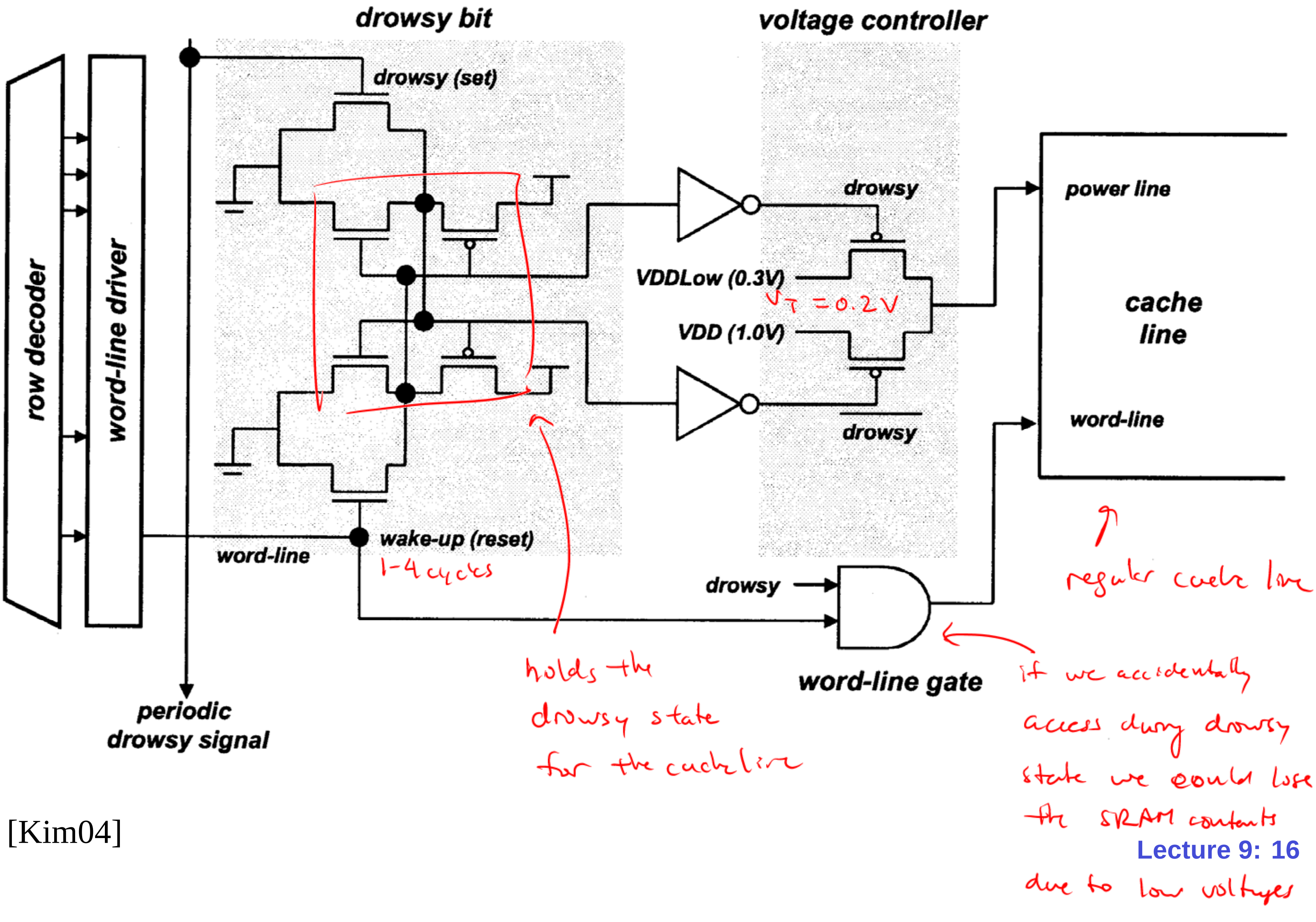
turn it back on, on access

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**
  - exponential
  - **Leakage drops superlinearly with reduced Vdd**
  - **Cell can be fully restored in a few cycles**
  - misprediction on drowsy, not hit/miss
  - **Much lower misprediction cost than gated-Vdd, but noise susceptibility and less reduction in leakage** vs gated Vdd

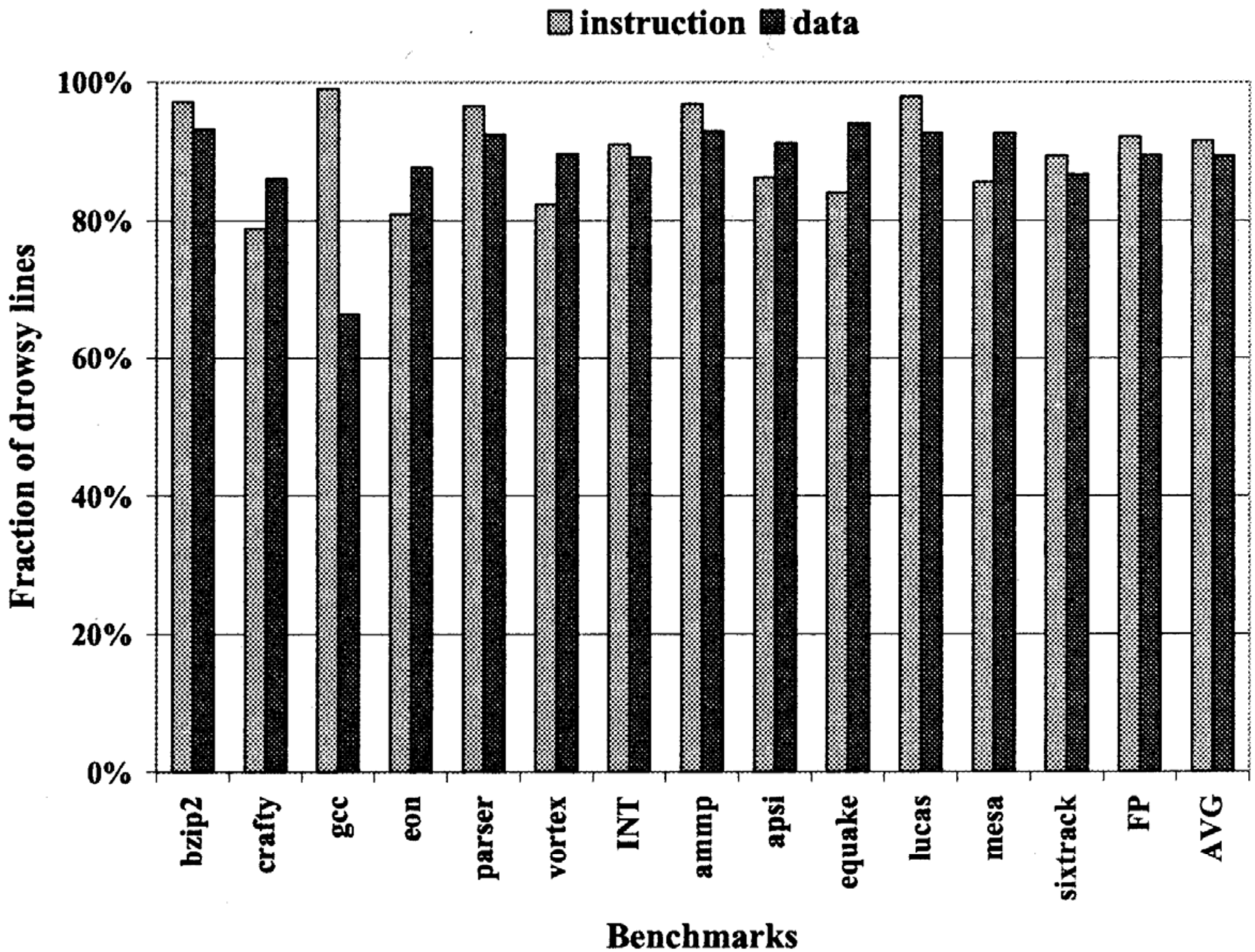
Drowsy Cache Organization



[Kim04]

Drowsy Cache Effectiveness

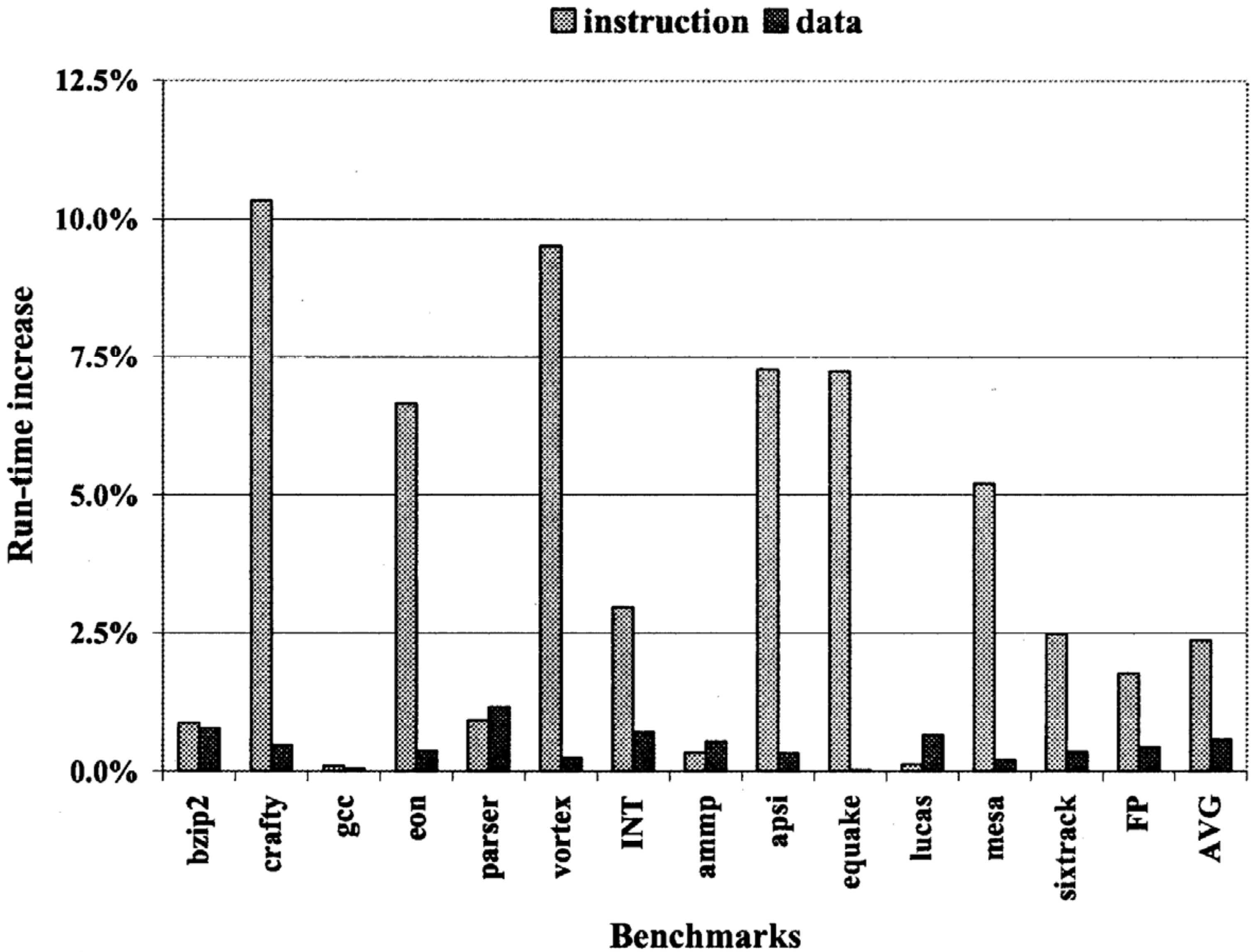
(assumed 1-cycle wakeup time)



drowsy cache turns drowsy also

[Kim04] 32KB L1 caches 4K cycle drowsy period

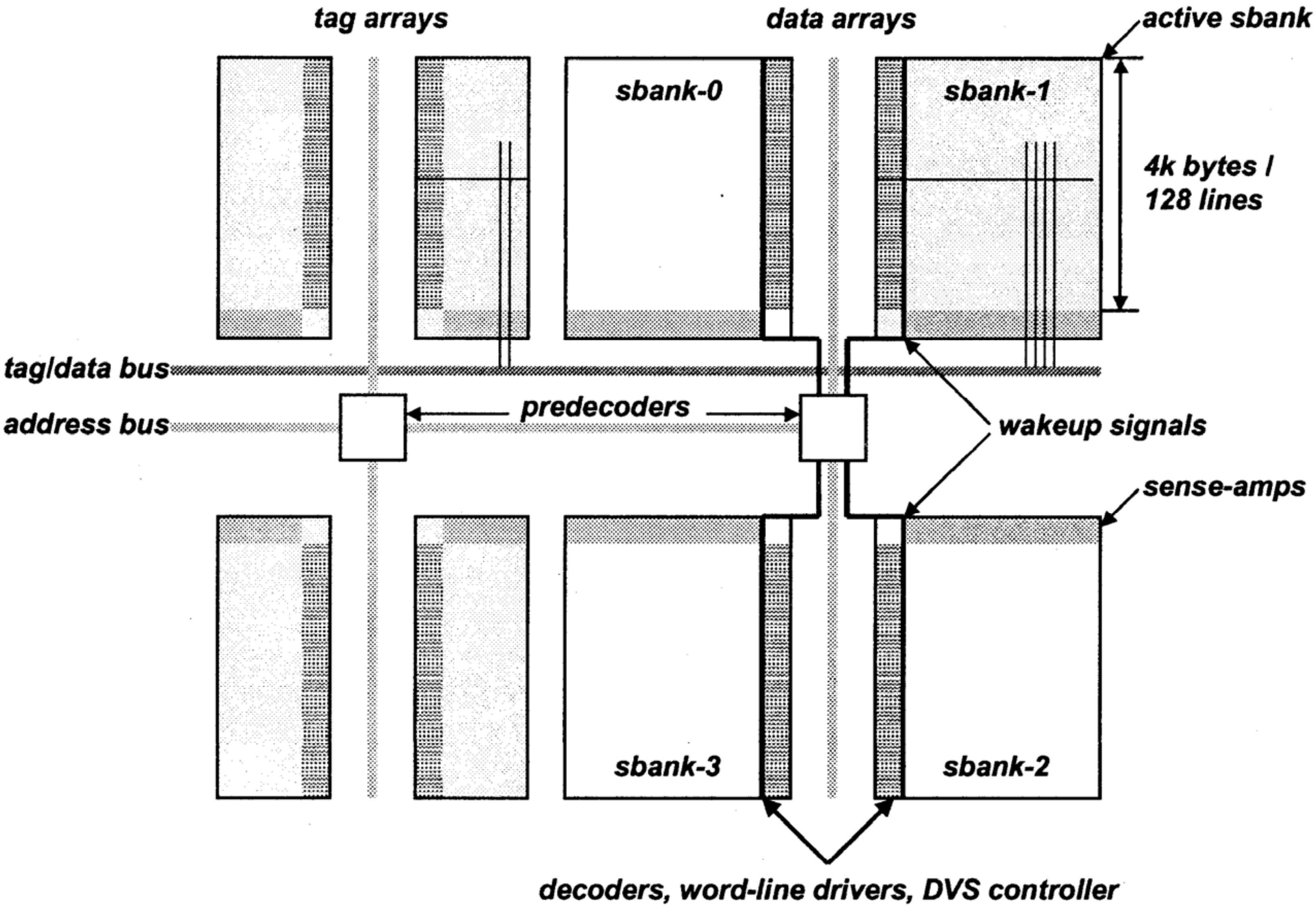
Drowsy Cache Performance Cost



drowsy cache is bad for I-cache. more temporal locality for data, must do something new for I cache

[Kim04]

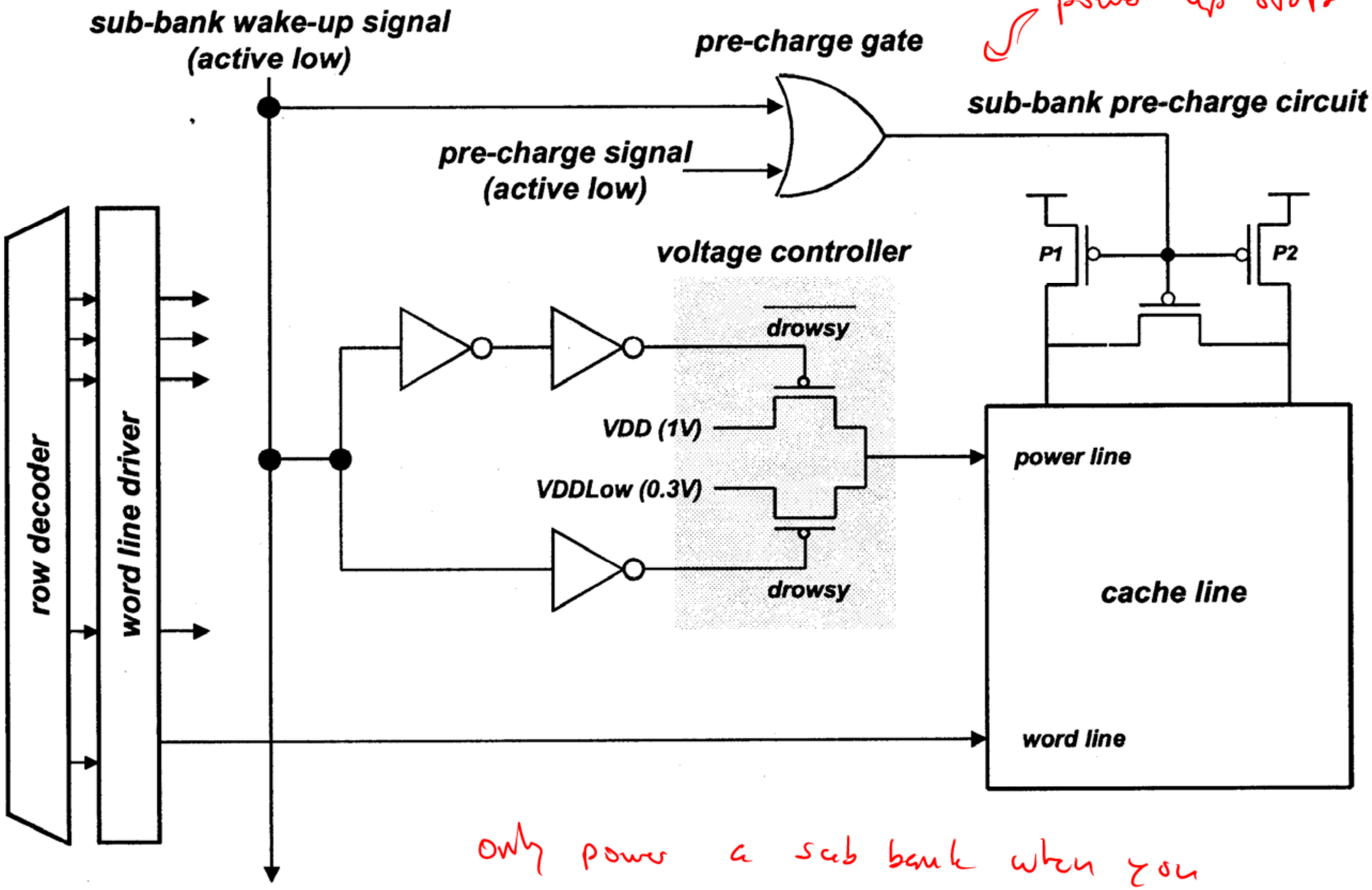
Sub-banked Drowsy I-cache



[Kim04]

Sub-banked Drowsy I-cache

power up stuff

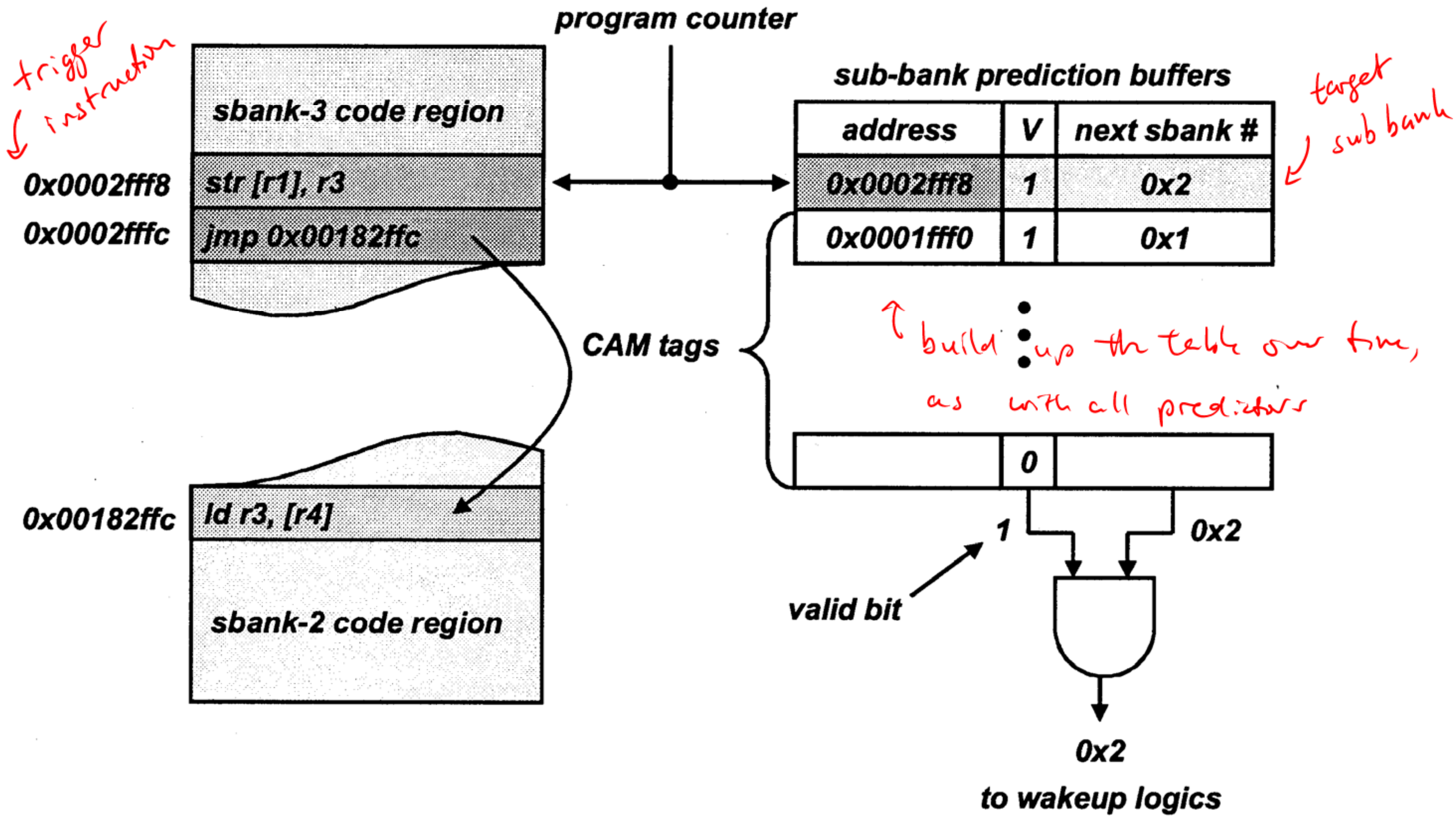


only power a sub bank when you need to. need prediction

[Kim04]

Next Sub-bank Prediction

- Predict fetch transition to a new sub-bank

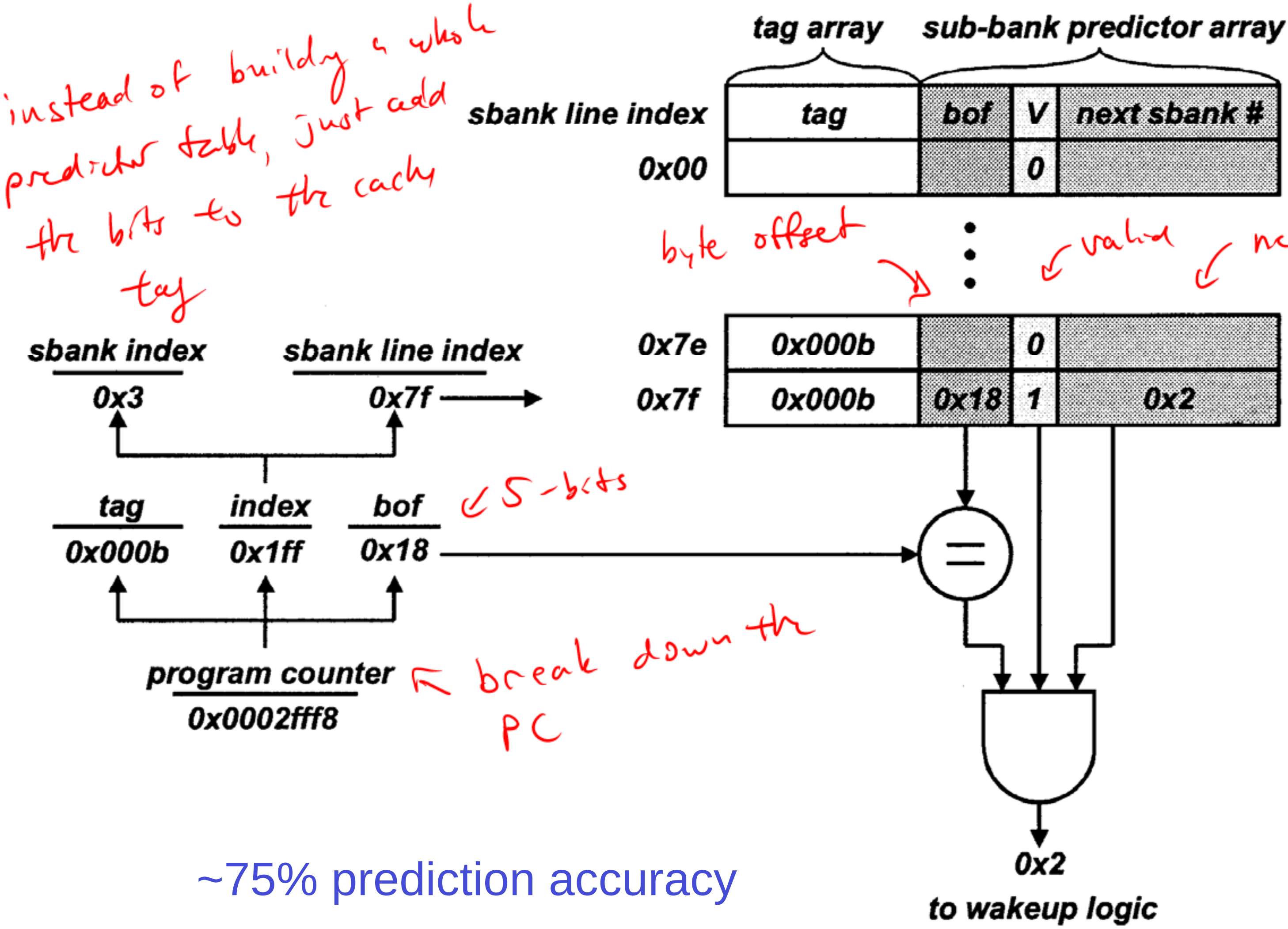


[Kim04]

Next Sub-bank Prediction

- Can embed predictions with cache tags

instead of building a whole predictor table, just add the bits to the cache tag



byte offset

valid

next sub bank

5-bits

break down the PC

~75% prediction accuracy

[Kim04]

slightly worse performance

lots more leakage

Drowsy D-cache vs. D-cache Decay

FP	Run-time impact (%)		Norm. leakage (%)		INT	Run-time impact (%)		Norm. leakage (%)	
	drowsy	decay	drowsy	decay		drowsy	decay	drowsy	decay
ammp	0.54	0.12	34 (22)	81	bzip2	0.77	0.38	34 (22)	35
applu	0.50	0.00	37 (26)	41	crafty	0.46	1.08	39 (28)	114
apsi	0.32	0.02	35 (23)	31	eon	0.36	0.13	38 (26)	38
art	0.61	0.01	38 (27)	44	gap	0.46	0.90	32 (20)	31
equake	0.03	0.04	33 (21)	10	gcc	0.06	0.00	53 (45)	96
facerec	0.15	0.03	32 (20)	23	gzip	2.07	1.23	34 (22)	85
fma3d	0.14	0.87	32 (20)	141	mcf	0.66	0.04	34 (22)	35
galgel	0.31	0.00	51 (42)	68	parser	1.17	1.75	35 (22)	63
lucas	0.66	0.00	34 (22)	25	perl	0.87	6.97	36 (24)	224
mesa	0.21	0.09	34 (22)	22	twolf	0.87	0.29	35 (22)	36
mgrid	0.49	0.00	40 (28)	42	vortex	0.25	0.78	36 (25)	126
sixtrack	0.34	0.18	39 (27)	56	vpr	0.89	0.84	36 (24)	76
swim	0.61	0.00	37 (25)	34	avg	0.57	0.64	37 (25)	59
wupwise	0.20	0.01	34 (22)	50					

[Kim04]

with high Vt transistors

Drowsy I-cache vs. I-cache Decay

FP	Run-time impact (%)		Norm. leakage (%)		INT	Run-time impact (%)		Norm. leakage (%)	
	drowsy	decay	drowsy	decay		drowsy	decay	drowsy	decay
ammp	0.00 (99)	1.83	25	28	bzip2	0.83 (52)	0.69	24	8
applu	0.00 (99)	0.22	24	10	crafty	3.75 (71)	6.26	26	118
apsi	2.01 (67)	0.14	25	25	eon	2.54 (70)	0.31	25	39
art	0.00 (99)	0.21	24	1	gap	1.93 (70)	0.55	25	40
equake	0.71 (87)	0.07	25	19	gcc	0.03 (78)	2.37	24	15
facerec	0.39 (76)	0.26	24	5	gzip	1.00 (87)	0.24	25	6
fma3d	4.52 (74)	0.08	26	54	mcf	0.19 (81)	0.23	24	3
galgel	0.00 (27)	0.04	24	1	parser	0.51 (72)	0.44	24	8
lucas	0.01 (97)	0.00	24	2	perl	2.34 (62)	1.84	25	68
mesa	0.67 (86)	0.37	25	23	twolf	0.30 (74)	2.98	24	56
mgrid	0.01 (99)	0.10	24	19	vortex	4.34 (63)	8.43	26	147
sixtrack	1.23 (86)	0.45	25	15	vpr	0.00 (99)	0.16	24	5
swim	0.00 (99)	0.28	24	9	avg	0.79 (74)	0.99	25	23
wupwise	0.79 (79)	0.32	24	20					

% accuracy of the predictor

[Kim04]

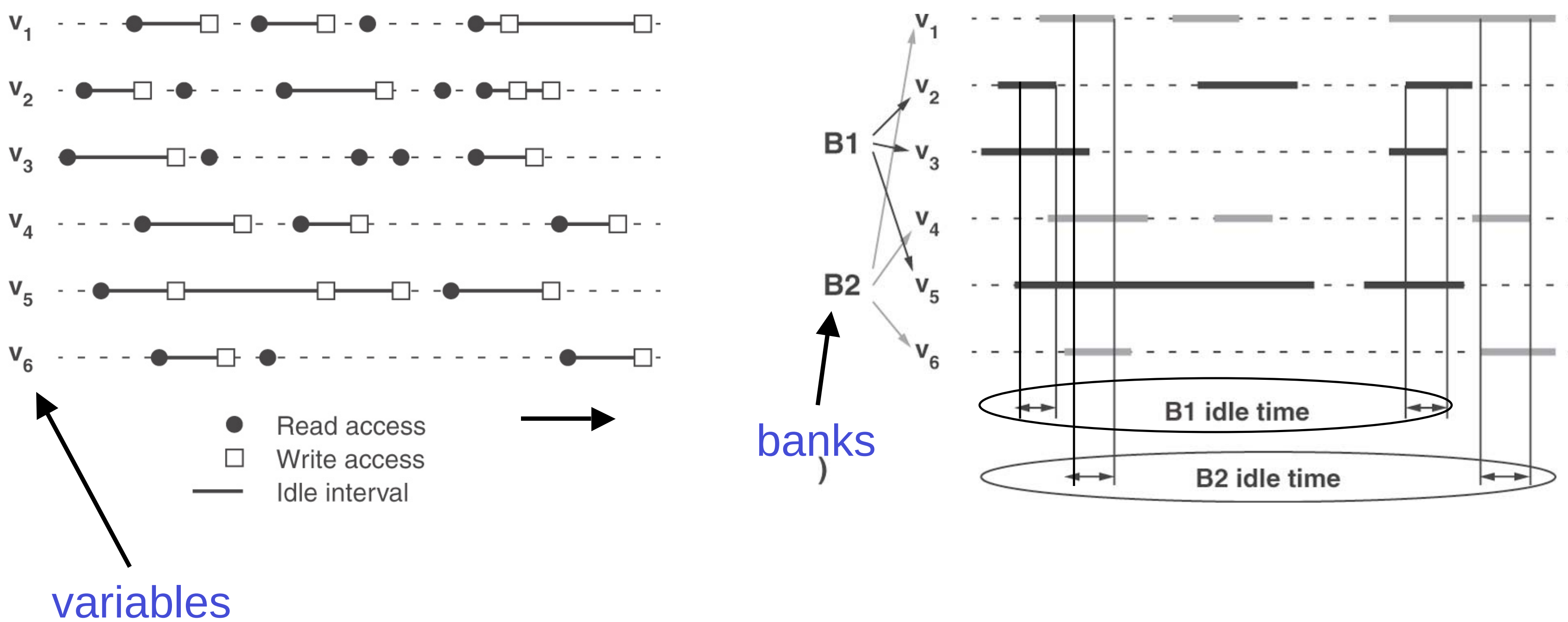
Cache Power Management

- Circuit techniques
  - Transistor sizing, multi-V_t, low-swing bitlines, etc
- Microarchitecture techniques
  - Static techniques: banking, phased tag/data access, way prediction, CAM-tags
  - Dynamic techniques: gated-Vdd, cache decay, drowsy caches
- Compiler techniques
  - Data partitioning to enable sleep mode

⇒ add sleep enable/disable instructions to an ISA
drowsy

Compiler-based Data Partitioning

- **Multiple D-cache banks, each with sleep mode**
- **Lifetime analysis used to assign commonly idle data to the same bank**



Next Time

Cache Case Studies