

# IATAC: A Smart Predictor to Turn-off L2 Cache Lines

JAUME ABELLA

*Universitat Politècnica de Catalunya-Barcelona*

ANTONIO GONZÁLEZ and XAVIER VERA

*Intel Barcelona Research Center, Intel Labs-UPC*

Published in  
*ACM Transactions on Architecture and Code Optimization*  
Vol 2, No. 1, March 2005 pp 55-77.

Presented by Anthony M. Castaldo

- Two Main Sources of Power Dissipation In a Processor
  - Dynamic Power (Has been most important)
  - Leakage (Growing at 5x rate of Dynamic per generation)
- Large caches occupy most area; responsible for most leakage
- Caches are getting larger
- L2 is largest on-chip cache, consumes most power
- Need to power them off!
- IATAC: 65% of L2 lines turned off
- IATAC: 2% performance hit

- Combine Caches of Different Voltages  
Abella and González [2003]
- Gated Supply Voltage  $V_{DD}$ , lost contents  
Powell et al. [2000]
- Drowsy Caches (reduced  $V_{DD}$  no loss)  
Require two supply lines, so higher overhead  
Flautner et al. [2002]
- Drowsy Caches (Gated Ground)  
Non-viable, requires too accurate a supply voltage  
Agarwal et al. [2002]
- SuperDrowsy Caches (One  $V_{DD}$ , no losses)  
more susceptible to soft errors  
Kim et al. [2004]

- Sub-Banking, Multiple Line Buffers, Bitline Segmentation  
Ghose and Kamble [1999]
- Vertical and Horizontal Cache Partitioning Su and Despain [1995]
- Dynamic or Static reconfiguration (cache size or associativity)  
**Dynamic:** Balasubramonian et al. [2002], Dropsho et al. [2002]  
**Static:** Albonesi [1999], Zhang et al. [2002, 2003]
- Power off L2 items if in L1 Li et al. [2002]
- Cache the L1 Cache (L1 Filter Cache) Kin et al. [1997]
- Encode / Compress frequently used values Yang and Gupta [2002]
- Way Predictors Inoue et al. [1999]
- Way Prediction and Remap for Non-Conflicting Access  
Powell et al. [2001] based on Batson and Vijaykumar [2001] and by Calder et al. [1996]
- **Switch off Cache Lines Not Expected To Be Re-Used**  
Very similar to IATAC Zhou et al. [2001], Kaxiras et al. [2001]

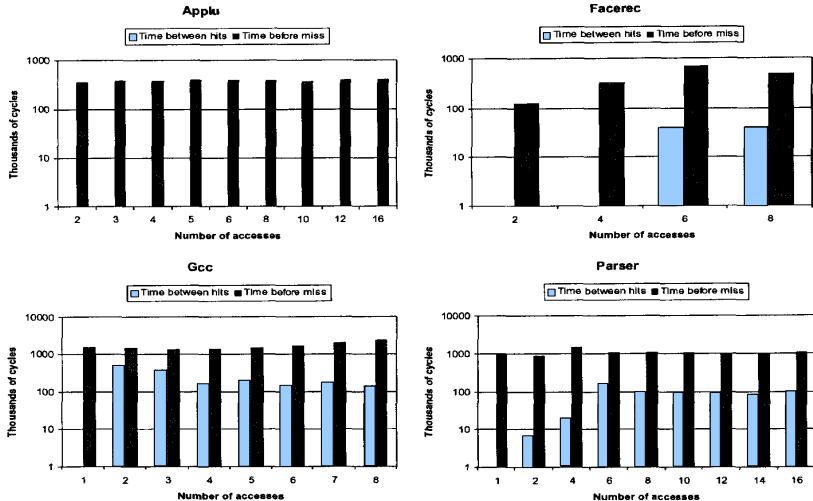


Fig. 1. Average time between hits to cache lines (*time between hits*) and the average time from the last access to replacement (*time before miss*) against varying numbers of accesses. The results correspond to four representative programs (note logarithmic scale).

- L1 filters out most Stride 0 access, so L2 deals with Stride  $> 0$ .
- The access pattern in one cache line tends to be duplicated in other lines.
- Local cache line predictors don't share information across multiple lines.
- Almost always, *time between hits*  $<$  *time between misses*.
- *time between hits* varies depending on how often a cache line is accessed.
- If we wait as long as time between hits WITHOUT a hit: Power off the line.

- Tag is always on, to detect prediction errors. Also, hits are counted even if line is powered off!
- Per Line Variables:
  - counter is incremented every access.
  - `thits` is the maximum time between consecutive hits.
  - elapsed cycles since line was last accessed.
  - decay cycles that must elapse to turn off line.
  - `wrong` a bit to indicate line was prematurely turned off.
  - `on/off` indicates whether line is powered up or down.
- Global Data, all per access count tracked:
  - `AcumCounters`: Number of cache lines replaced.
  - `GlobalDecay`: Decay required.
  - `MaxGlobalDecay`: Maximum of values to the right of a given access count.

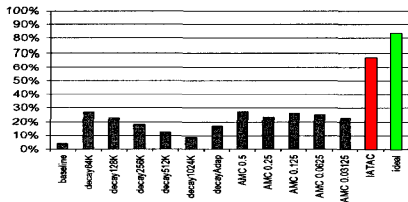
- Coarse Counters: Only tick every 1000 cycles. Heirarchical, saves energy.
- Two tested cache sizes: Large (4MB) and Medium (512KB).
- MIN\_COUNT: Ignore count if fewer recent hits than MIN\_COUNT. Typically 8.
- MAX\_COUNT: Decay value for AcumCounters. Cache size dependent; tested at 1024 (512KB) and 8192 (4MB).
- AcumCounters: All are halved if any hits MAX\_COUNT.
- Every hit on a tag: increment counter, check if wrong, set decay.
- Every miss on a tag: Use `thits` to update `GlobalDecay`, add to `AcumCounter` and adjust all if needed, then set line decay.

- Cache Decay: Hu et al. 2002, Kaxiras et al. 2001  
 A Local Information approach relying upon temporal clustering.
  - **DecayN** Fixed Decay Intervals: Turn off cache line if N cycles elapse with no access.
  - **DecayAdap** Adaptive Decay Intervals: Dynamically finds decay intervals for each cache line individually; but turns off tags and makes compounding mistakes.
- **AMCPF** Adaptive Mode Control: Zhou et al. 2001  
 Whole cache, but dynamic decay interval. Relies on tags remaining on to properly calculate ideal decay interval, and **PF** 'Performance Factor' as tuning parameter.
- CACTI 3.0 (timing, power and area model for cache memory)
- WATTCH (architecture-level power and perf simulator)
- Spec2000 benchmark suite, simulating 1B instructions for each benchmark after 100M instruction 'warmup' for caches.

## Exactly what it sounds like! Energy Used Multiplied by Gate Delay (Execution time)

- *It is so **EASY** to reduce Energy!*
  - Decrease Clock Period
  - Reduce Supply Voltage
  - Reduce Capacitance (use smaller transistors)
- But all of these *increase* Gate Delay (and thus Execution Time).
- Processors vary in Energy consumption by a factor of 100, but if we look at Energy x Delay, variance is reduced to single digits.
- EDP, and later  $ED^2P$ , are measures of *Energy Efficiency*.

Turned off cache lines (512 KB)



Turned off cache lines (4 MB)

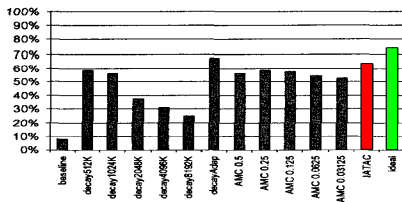
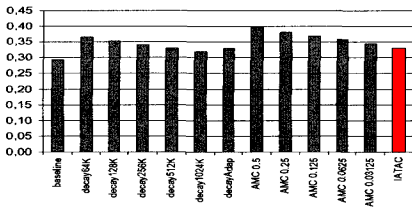


Fig. 6. L2 turn-off cache line ratio for the different mechanisms.

L2 miss ratio (512 KB)



L2 miss ratio (4 MB)

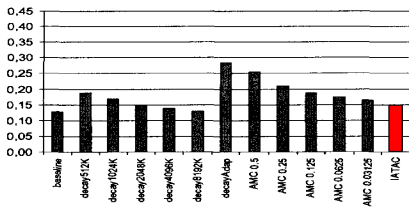


Fig. 7. L2 miss ratio for the different mechanisms.

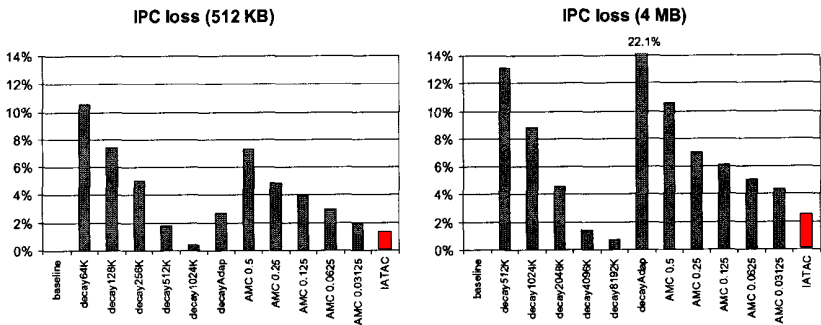


Fig. 5. IPC degradation for the different mechanisms.

- IATAC is first to use a global access history of a cache line to tailor the decay interval for that individual cache line
- IATAC outperforms all previous approaches since turn-off ratio is very high and performance loss is very low
- IATAC costs very little additional hardware
- IATAC provides best  $ED^2P$  across all mechanisms for different cache sizes
- There is room for future work in two ways. IATAC gets close to the ideal turn-off ratio but about half the ideal  $ED^2P$  improvement; because the 2% performance degradation has a disproportionate impact on  $ED^2P$ .
  - Similarly accurate predictors with some way of reducing performance degradation,
  - More accurate predictors would lead to reduced performance degradation.