Reducing Cache Access Energy in Array-Intensive Applications[*] (Extended Abstract)

Mahmut Kandemir
Microsystems Design Lab
Pennsylvania State University
University Park, PA, 16802, USA

Ibrahim Kolcu
UMIST
P.O. Box 88, Sackville Street
Manchester, M60 1QD, UK

Cache memories are known to consume a large percentage of on-chip energy in current microprocessors. For example, [1] reports that the on-chip cache in DEC Alpha 21264 consumes approximately 25% of the on-chip energy. Both sizes and complexities of state-of-the-art caches play a major role in their energy consumption. Direct-mapped caches are, in general, more energy efficient (from a per access energy consumption viewpoint) as they are simpler as compared to set-associative caches, and require no complex line replacement mechanisms (i.e., there is no decision concerning which line has to be evicted when a new line is to be loaded).

While there exists a large body of compiler-based techniques to manipulate access pattern of a given code to improve its cache utilization, there are not many compiler techniques that try to improve cache energy consumption of a given code. Rather, in many cases, a reliance is placed upon the observation that optimizing cache locality also optimizes cache energy. This is true to some extent as optimizing locality (performance) of memory accesses reduces the activity between cache and off-chip memory, and consequently, decreases the number of writes into cache. Recent work (e.g., [2]) also shows that the classical performance-oriented compiler optimizations (e.g., loop-level transformations) can be very effective in reducing overall memory system energy.

This study goes beyond these performance-centric techniques, and proposes an energy-oriented optimization strategy that aims directly at reducing per access energy cost for direct-mapped data caches (rather than as a side effect of a performance-oriented optimization). Specifically, we have developed a compiler algorithm that uses access pattern analysis to determine those memory references that are certain to result in cache hits in a virtually-addressed direct-mapped data cache. After detecting such references, the compiler substitutes the corresponding load operations with energy-efficient loads that access only data array of the cache instead of both tag and data arrays. This tag access elimination, in turn, reduces the per access energy consumption for data accesses. This study makes the following contributions:

- We present an access pattern analysis strategy for detecting and isolating definite hits (i.e., hits that can be detected at compile time with a 100% accuracy) in a given application code.
- We use the result of this analysis to transform code to make explicit (in the code) the memory accesses which are responsible for these certain hits.
- We report experimental results showing the benefits of the proposed technique.

Our experimental results indicate that certain definite hits constitute a large percentage of total hits. They also show that even our most conservative strategy improves the cache energy consumption by 11% on the average. The overall memory system energy savings due to our approach are competitive with those obtained when set-associative caches are used.

References

[1] J. F. Edmondson et al. Internal organization of the Alpha 21164, a 300 MHz 64-bit quad-issue CMOS RISC microprocessor. *Digital Technical Journal*, Vol. 7, No. 1, 1995, pp. 119–135.

[2] M. Kandemir et al. Influence of compiler optimizations on system power. In *Proceedings of the 37th Design Automation Conference*, Los Angeles, California USA, June 5-9, 2000, pp. 304–307.

[*]This work was supported in part by NSF awards #0093082 and #0103583.