Techniques for reducing power consumption in CMP NUCA caches

Pierfrancesco Foglia*, Francesco Panicucci°,
Cosimo Antonio Prete*, Marco Solinas*

** Dip. di Ingegneria dell'Informazione, Univerità di Pisa, via Diotisalvi, 2 56100 Pisa, Italy*
° IMT Lucca Institute for Advanced Studies, piazza San Ponziano, 6 55100 Lucca, Italy
Email: {foglia, prete, marco.solinas}@iet.unipi.it, francesco.panicucci@imtlucca.it

Members of HiPEAC, The European Network of Excellence on High Performance Embedded Architecture and Compilation

ABSTRACT

Current trend of technology scaling makes it possible to put a huge number of transistors on a single die. While dynamic power consumption can benefit from technology scaling, static power consumption get worse, thus making the latter the dominant factor of power consumption in future microprocessor systems. As on-chip cache memories require the most part of chip area and number of transistors, techniques have been developed to improve their power efficiency. In this paper, we analyze, in the context of CMP systems, how applications use cache banks depending on their locality, in order to motivate the opportunity of applying power-saving techniques to large L2 cache in a CMP environment.

KEYWORDS: power consumption, leakage, NUCA, CMP

1 Introduction

Power consumption is a major concern to computer architects. We can distinguish between two different kinds of power consumption: *dynamic power consumption*, which arises due to signal transitions, and *static power consumption*, due to gate leakage current that always flow on powered-on transistors [1, 2, 3]. While dynamic power consumption reduces with process technology scaling, static (leakage) power consumption increases [3]. Taking into account current technology scaling [8], static power increase is becoming more significant then dynamic power decrease, making the leakage currents the dominant source of power consumption [1, 2, 3].
Technology scaling makes also possible to design CMP systems [14], with more CPUs on a single-chip sharing a large on-chip L2 cache [7, 14]. As a large fraction of the total on-chip area and number of transistors is required by the L2 cache, such systems suffer of two main problems: *i*) *wire delay*: clock frequency increases as well as the delay in communication lines, thus signals need more clock cycles to be propagated on the chip, and this delay significantly affects performance [7, 15]; *ii*) *static power consumption*: the huge number of active transistors waste a lot static power [3]. To manage the wire delay problem, it could be adopted an architecture based on D-NUCA cache [7]. NUCA caches

[10] are bank partitioned caches with a non-uniform access time; D-NUCA are NUCA cache-based architectures that let data migrate among blocks to bring them near the CPU, thus limiting the wire delay problem. To reduce static power consumption, it is possible to adopt different solutions basing on different principles: *i*) defining a low-power state for each cache bank, and switching the lines between high- and low-power states basing on how frequently the lines are accessed by the processor (*state-preserving policy*); *ii*) switching off the cache banks that are no longer referred by the processor, or for which the time interval between two subsequent accesses is large enough (*state-destroying policy*) [3]. Basing on these considerations, two main techniques have been adopted: the first, *drowsy cache* [9], refers to a state-preserving policy, while the second, *gated-Vdd* [11], is a state-destroying solution.
In this paper, by considering the CMP architecture proposed by Beckmann and Wood [7], we investigate the behaviour of the access pattern to the D-NUCA banks, exploited by typical applications. Our preliminary results show how the hit distribution among the L2 banks may allow the adoption of static power saving strategies also in CMP systems.

2 Proposed techniques to reduce leakage power consumption

Several solutions have been proposed for reducing static-power consumption in large on-chip caches. Two main categories can be individuated: drowsy cache and gated-Vdd.
Gated-Vdd [11] technique turns off the transistors that are not used; as leakage power comes from transistors that are left on, gated-Vdd drastically reduces static power consumption; as a drawback, it doesn't preserve the state of the data, thus increasing the miss-rate if the same data are subsequently accessed. Drowsy cache [9] strategy makes use of multiple supply voltage, basing on a simple consideration: leaving the line fully on, consumes lot of power, turning it completely off causes the loss of the data. So this technique puts the cache line in a low-power state when it is not needed for a while, thus reducing static power consumption without loosing the data; as a drawback, the data can't be accessed while in this state, and there is a given wake-up time when switching the "drowsy" line from the low-power to the completely-on state. As the transistors are not completely turned off, drowsy cache saves less static power than gated-Vdd: if the data is not accessed for a very long time interval (or it is no longer accessed [1, 2]) gated-Vdd would perform better; if the data is accessed in a moderate time interval, drowsy-cache is the right solution because it avoids refetch penalty [3].
Kaxiras et al. [1, 2] adopt a gated-Vdd solution that turns off cache lines when a given number of cycles have elapsed since its last access. They pointed out that cache lines often store items that will be not used again: they call the time interval from the last hit to the line to the time it is evicted the *dead time*, and they aim to reduce power wasted on dead items in the cache.
Bardine et al. [4] proposed a solution that dynamically adapts cache size to the locality of the running application. The proposed architecture is a Dynamic-NUCA, that exploits, via prediction, the banks partitioning and the promotion mechanism to achieve high power efficiency. They dynamically turn off and on the cache ways depending on the locality of the application, using a gated-Vdd approach.
Tanaka et al. [5, 6] developed a technique that exploits gated-Vdd control per cache block and dynamic data compression scheme in the L2 cache. They make use of a compression-

decompression mechanism and allow data to be stored in L2 in reduced size, thus permitting the switching-off of the parts of the block that is no used.

3 Architecture, methodology and results

Our explorative evaluation is based on the architecture proposed in [7]: it is a CMP system with 8 CPUs which share a L2 D-NUCA cache. Figure 1 shows the architecture of the cache and how each CPU is connected to it. The L2 cache is composed by 256 banks, for a total storage capacity of 16 MB. The banks are *physically* organized in 16 *bankclusters*, shaped as shown in Figure 1. The migration policy allows data to migrate among bankclusters, with the purpose of bringing the most used data near the cpu that is referring them. Each bankcluster is equivalent to a way in conventional caches.

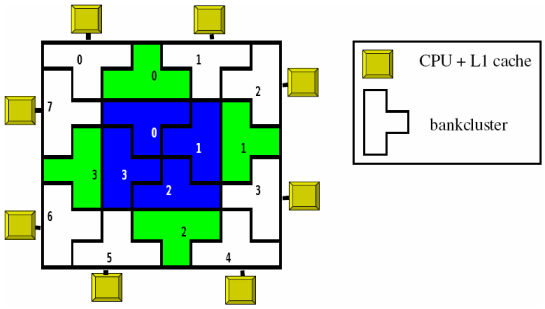

Figure 1: D-Nuca cache layout. Each "tetris" is a bankcluster: white tetris are bankcluster local of each cpu, light grey tetris are reffered to as intermediate, while dark tetris are the center ones. The search policy initially looks for the addressed block in the local bankcluster; in case of miss, the two nearest intermediate bankclusters are looked-up and finally the request is forwarded to the central bankclusters.

We analyze how D-NUCA banks and bankclusters are accessed, in order to motivate the use of a static power saving policy. Our experiments have been conducted using the Simics [16] simulator, together with GEMS [17]; we run Ocean and Barnes applications from the SPLASH-2 [12] benchmark suite, and applications from the SPEC2000 [13] suite. We considered conventional fast-forwarding phases; for both Ocean and Barnes, the simulations executed for 1B instructions, collecting statistical results every 40M instructions; in the case of the SPEC applications, we didn't consider the initial phase of data initialization in order to speed-up the execution, then the simulation executed for 200M instructions, also collecting statistical data every 40M instructions. Figure 2 shows the typical hit distribution among two different sampling points.

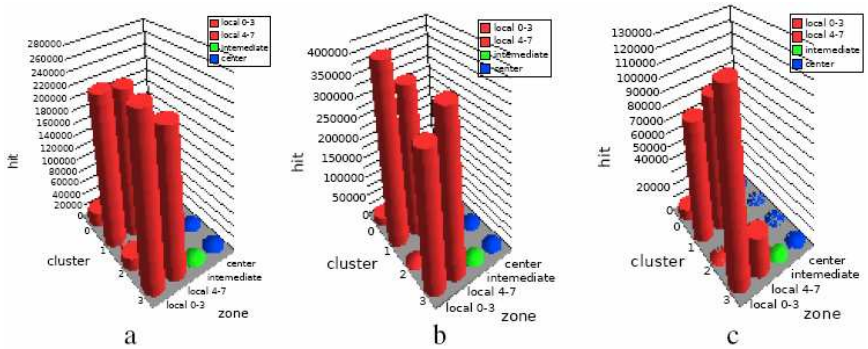

Figure 2: Number of hit in D-Nuca bankcluster: (a) Ocean (processes run on cpus 1,3,5,7) 40M instr; (b) Barnes (processes run on cpus 1,3,5,7) 40M instr; (c) mcf (cpu1), bzip2 (cpu3), art (cpu5) and parser (cpu7) 40M instr

We notice that the most part of the hits occur in bankcluster that are local to the CPUs in which the applications run, whereas the other bankcluster present a fewer number of accesses. This suggests that rarely used bankclusters might be turned off in those run interval in which the applications' working set doesn't take advantage of the entire cache capacity.

4 Conclusions

In this paper we discussed the need of reducing static power consumption in modern microprocessor systems. Focusing on a CMP environment with large L2 D-NUCA cache, we investigated how the cache is accessed by different applications. Results indicate that static power reduction techniques may be also applied to D-NUCA CMP systems. We plan to investigate the effectiveness of such techniques in the further step of our research.

5 Acknowledgements

This works is partially supported by the SARC project founded by European Union under the contract no. 27648.

6 References

[1] Kaxiras, S., Hu, Z., Martonosi, M., *Cache Decay: Exploiting Generational Behaviour to Reduce Cache Leakage Power*. In *Proceedings of the 28thIintern. Symp. on Computer Architecture*, Göteborg, Sweden, June 2001

[2] Hi, Z., Kaxiras, S., Martonosi, M., *Let Caches Decay: reducing leakage energy via exploitation of cache generational behaviour*. In *ACM Transaction on Computer System*, Vol. 20, Issue 2, May 2002

[3] Meng, Y., Sherwood, T., Kastner, R., *Exploring the limits of leakage power reduction in caches*. In *ACM Transaction on Architecture and Code Optimization*, Vol. 2, Issue 3, Sept. 2005

[4] Bardine, A., Foglia P., Gabrielli, G., Prete, C. A., Stenstrom, P., *Way-Adaptable D-Nuca Caches*. In *ACM SIGARCH Computer Architecture News*. To appear

[5] Tanaka, K., Matsuda, A., *Static energy reduction in cache memories using data compression*. In *IEEE Region 10 Conference on Convergent Technologies For The Asia-Pacific*, Hong-Kong, Nov. 2006

[6] Tanaka, K., Kawahara, T., *Leakage energy reduction in cache memory by data compression*. In *International Workshop on Advanced Low Power Systems*, 2007. To appear

[7] Beckmann, B. M., Wood, D. A., *Managing wire delay in Large Chip-Multiprocessor Caches*. In *Proceedings of the 37thIinternational Symposium on Microarchitecture*, Portland, Dec. 2004

[8] International Technology roadmap for Semiconductors. Semiconductor Industrial Association, 2005

[9] Flautner, K., Kim, N. S., Martin, S., Blaauw, D., Mudge, T., *Drowsy caches: simple techniques for reducing leakage power*. In *Proc. of the 29thIintern. Symp. on Computer Architecture*, Anchorage, Alaska, May 2002

[10] Kim, C., Bourger, D., Keckler, S. W., *An adaptive, Non-uniform cache structure for wire-delay dominated on-chip caches*. In *Proceedings of the 10th international conference on Architectural support for Programming Languages and Operating System*, San Jose, California, Oct. 2002

[11] Powell, M., Yang, S., Falsafi, B., Roy, K., Vijaykumar, T. N., *Reducing leakage in a high-performance deep-submicron instruction cache*. In *IEEE Trans. on Very Large Scale Integration Systems*, Vol.9, Issue 2, Sept. 2001

[12] Woo, S. C., Ohara, M., Torrie, E., Singh, J. P., Gupta, A., *The SPLASH-2 programs: characterization and methodological considerations*. In *Proc.s of the 22th Intern. Symp. on Computer Architecture*, S. Margherita Ligure, Italy, June 1995

[13] Standard Performance Evaluation Corporation http://www.spec.org

[14] Olukotun, K., Nayfeh, B. A., Hammond, L., Wilson, K., Chang, K., *The case for a single-chip Multiprocessor*., In *Proceedings of the 7th international conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, Massachusetts, United States, Oct. 1996

[15] Hammond, L., Nayfeh, B. A., Olukotun, K., *A single-chip multiprocessor*. In *IEEE Computer*, Vol. 30, Issue 9, Sept. 1997

[16] Virtutec Simics http://www.virtutech.com

[17] Winsconsin Multifacet GEMS Simulator, http://www.cs.wisc.edu/gems/