Shrinking L1 Instruction Caches to Improve Energy–Delay in SMT Embedded Processors

Alexandra Ferrerón–Labari, Marta Ortín–Obón, Darío Suárez–Gracia, Jesús Alastruey–Benedé, and Víctor Viñals–Yúfera

gaZ—DIIS—I3A, Universidad de Zaragoza, Spain
{ferreron, ortin.marta, dario, jalastru, victor}@unizar.es

Abstract. Instruction caches are responsible for a high percentage of the chip energy consumption, becoming a critical issue for battery-powered embedded devices. We can potentially reduce the energy consumption of the first level instruction cache (L1-I) by decreasing its size and associativity. However, demanding applications may suffer a dramatic performance degradation, specially in superscalar multi-threaded processors, where, in each cycle, multiple threads access the L1-I to fetch instructions.
We introduce iLP-NUCA (*Instruction Light Power NUCA*), a new instruction cache that substitutes the conventional L2, improving the Energy-Delay of the system. iLP-NUCA adds a new tree-based transport network topology that reduces latency and energy consumption, regarding former LP-NUCA implementations.
With iLP-NUCA we reduce the size of the L1-I outperforming conventional cache hierarchies, and reducing the overall consumption, independently of the number of threads.

1 Introduction

Superscalar execution cores demand a continuous instruction supply to feed their functional units. Delays due to instruction cache misses affect the instruction flow speed and instruction issue, and hence, performance. Simultaneous Multi-Threading (SMT) is a technique to hide long latency operations, such as cache misses, by the execution of several threads [19]. SMT aims to have all the functional units highly utilized by using a more powerful front-end (fetch unit) that supplies instructions from several threads. Consequently, the aggregated demand of instructions added by SMT makes on-chip instruction caches even more critical.

Instruction caches are responsible for a high amount of the energy consumption of the system. For example, StrongARM SA-100 and ARM 920TTM dissipate the 27% and 25% of the total power in the instruction cache, respectively [13],[15].

Ideally, we would like to have an instruction cache big enough to fit the footprint of the most demanding applications in order to increase their hit rate. However, bigger caches come at the expense of higher access latencies and higher energy consumption per access. Thus, there is a complex trade-off between size, on the one hand, and latency and energy consumption, on the other hand.

Reducing the size of the first level instruction cache (L1-I)—as Qualcomm MSM8960 does [6]—will potentially benefit the system from the energy perspective, but it could affect the cache hit rate, harming performance, and the energy consumption of the second level cache could increase, counteracting other savings.

Recent works propose to dynamically reconfigure the L1 size and associativity, so only part of the ways or part of the sets are active in a given moment [18],[22]. With such an approach, the cache adapts to the behavior of the applications and operates within the optimal size. Other works add a small structure close to the L1-I that captures part of its accesses, increasing the fetch speed, reducing the energy consumption, or both [3],[10].

All of these techniques need to either modify the interface between the first-level cache and the processor, or add additional hardware/software support. We propose a different approach not requiring to modify the interface between the L1-I and the processor. We reduce the L1-I size and associativity, and replace the L2 cache with an iLP-NUCA (*instruction Light Power NUCA*), which minimizes performance degradation and reduces energy consumption.

Our experiments running SPEC CPU2006 show that, for the same L1-I size, iLP-NUCA performs better and consumes less energy than a conventional instruction-only L2. Alternatively, by using iLP-NUCA we can shrink the L1-I from 16KB to 8KB and keep performance, regardless of the number of threads. Specifically, we reduce the energy consumption by 21%, 18%, and 11%, reaching 90%, 95%, and 99% of the ideal performance for 1, 2, and 4 threads, respectively.

This work makes the following contributions:

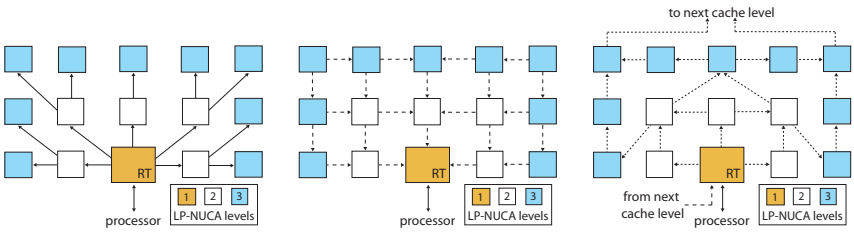
- We apply a new NUCA organization to the instruction hierarchy. To the best of our knowledge, this is the first work proposing a specific NUCA cache for the instruction hierarchy in both single thread and SMT environments.
- We adapt LP-NUCA for instruction hierarchies (iLP-NUCA). We propose a new instruction-driven transport network-in-cache for iLP-NUCA, based on a tree topology, which improves previous designs. This new topology simultaneously reduces the effective service latency of cache blocks, the number of links, and the energy consumed by the network.
- We propose to shrink L1-I size and associativity to save energy, and maintain an acceptable performance by replacing a conventional L2 with iLP-NUCA. The additional area iLP-NUCA provides, by reducing the L1-I, offers on-chip area for the implementation of other components such as accelerators.

The rest of the paper is organized as follows. Section 2 explores iLP-NUCA. In Section 3 the methodology we followed is explained. Section 4 collects the main results. Section 5 presents the related work, and Section 6 concludes.

2 Instruction Light Power NUCA

2.1 LP-NUCA Overview

LP-NUCA [16] extends the conventional L1 capacity by surrounding it with one or more levels of small cache tiles interconnected by specialized networks, as shown in Fig. 1.



(a) Search network. (b) Transport network. (c) Replacement network.

Fig. 1. 3-level LP-NUCA with its three networks-in-cache.

The first level of LP-NUCA (root-tile, RT) corresponds to a conventional L1 cache with added network logic. The second level of the conventional hierarchy is replaced by a tiled structure whose effective access time increases gradually. The LP-NUCA design relies on three specialized networks-in-cache: search, transport, and replacement, allowing for a simple implementation, low latency, and high bandwidth. The search network (Fig. 1a) uses a broadcast-tree to propagate miss requests from the RT to the rest of the LP-NUCA levels. The transport network (Fig. 1b) employs a 2-D mesh to deliver hit cache blocks from the tiles to the RT. The replacement network (Fig. 1c) connects tiles by means of an irregular topology to place recently evicted blocks close to the RT, and exploit their temporal locality. Please refer to [16] for further details about the design and implementation of LP-NUCA.

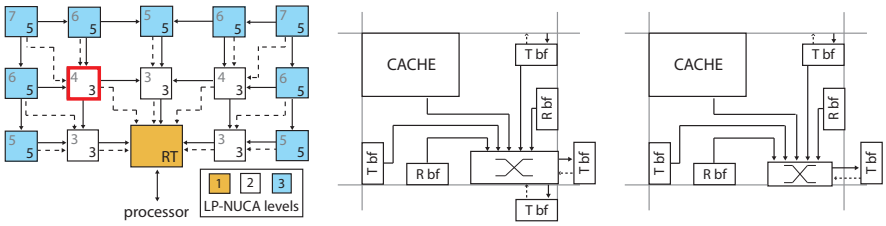
LP-NUCA has never been tested for instruction hierarchies; however, it is a good candidate because it could exploit the temporal code locality, reduce the average memory access time, and be more energy-efficient. Besides, it does not modify the interface between the L1 and the processor. We call the structure adapted to instruction hierarchies iLP-NUCA (*instruction Light Power NUCA*). The following section describes the improvements we propose to successfully adapt LP-NUCA to instruction hierarchies.

2.2 Instruction-Driven Tree-Based Transport Network

Instruction and data hierarchies differentiate in two key aspects. First, the miss instruction bandwidth demand from L1/RT is lower than the corresponding miss data bandwidth demand. In SMT environments (without taking into account the prefetches), the maximum number of requests in iLP-NUCA equals the number of threads being executed. Second, an instruction fetch missing in the first cache level stalls the corresponding thread, preventing the front-end to supply instructions to the pipeline. The L1-I miss resolution time becomes critical forcing the re-evaluation of the search and transport networks.

Previous designs of LP-NUCA utilized a bufferless search network with no flow control, having little margin for latency improvement, and a 2-D mesh for the transport network, because it provides high bandwidth with its path diversity. Figure 2a shows the 2-D mesh topology (solid lines). The numbers in the upper

left corner of the tiles represent the round-trip latencies (time since injecting a request into LP-NUCA until the block returns to the processor), assuming a one-cycle tile access time. The main components of the transport network (Fig. 2b) are a switch and two transport buffers (Tbf) that contain hit blocks on their way to the RT. The replacement buffers (Rbf) contain victim blocks waiting for replacement. Hits can occur also in the replacement buffers and, therefore, they must be accessed during a search operation. The transport network relies on buffered flow control, and uses store-and-forward flow control with on/off back-pressure (dashed lines in Figs. 2b and 2c) and two-entry buffers per link.



(a) 2-D Mesh (solid lines) and tree-based (dashed lines) transport networks.

(b) 2-D mesh transport components.

(c) Tree-based transport components.

Fig. 2. Transport networks (a)—2-D mesh (solid lines, latencies in upper left corner) and tree-based (dashed lines, latencies in bottom right corner)—and their respective components, (b) and (c), for the example tile (thick box).

A preferable transport network for instructions would offer lower latency, even if that comes at the expense of lower bandwidth. Figure 2a shows in dashed lines an instruction-driven transport network topology based on a minimum degree tree. With this topology the benefit is twofold. On the one hand, we decrease the number of links between tiles to just one transport output link, decreasing the crossbar delay and area (Fig. 2c). On the other hand, latency becomes uniform within each level and most of the tiles decrease their hit latency (numbers on bottom right corner in Fig. 2a), as we need less hops to reach the RT. The latter also reduces the energy consumed by the network when delivering blocks to the RT. Thus, we expect to reduce the average hit latency of blocks in comparison with previous LP-NUCA designs, and the energy consumption of the transport network. This latency reduction could allow us to dramatically shrink the RT (L1-I) because misses can be served faster from the next cache level.

The novel topology increases the number of inputs of the RT multiplexer from 3 to 5 without harming performance. Cycle time remains unchanged because the multiplexer is not in the critical path (it works in parallel with the instruction cache banks).

Finally, the original replacement network offers a good block replacement distribution and, therefore, iLP-NUCA will use the same network topology.

3 Methodology

3.1 Processor Model and Simulation Environment

We model the system summarized in Table 1. Our model is based on state-of-the-art high performance embedded systems such as IBM/LSI PowerPC 476FP [12], NetLogic XLP864 [7], and Freescale QorIQ AMP T2080 [4]. Our system has a more powerful memory hierarchy, and supports 4 hardware threads.

We compare two alternative instruction cache hierarchies. The first one is a conventional setup, named CV, made up of two dedicated instruction/data cache levels, L1-I/D and L2-I/D, and a third level shared by instructions and data, L3. The second one, named iLP, replaces L1-I and L2-I by a three-level iLP-NUCA, that is, L1-I is replaced by the root-tile, and L2-I by two levels of tiles (Le2 and Le3) interconnected as explained in Sect. 2. iLP also replaces L1-D and L2-D by the original data LP-NUCA. We model a perfect data path with L1-D/LP-NUCA caches that always hit, so that result variations only come from instruction activity. Section 4.3 removes this constraint to examine the interaction of iLP-NUCA with the data hierarchy.

We have developed *SMTScalar*, a cycle-accurate execution-based simulator based on SimpleScalar 3.0d for Alpha ISA [2]. SMTScalar heavily extends SimpleScalar to support detailed microarchitectural models, highly configurable memory hierarchies, simultaneous multi-threading execution, and iLP-NUCAs.

Table 1. Simulator micro-architectural parameters. BS, AM, lat, and init stand for block size, access mode, latency, and initiation latency, respectively.

Clock Frequency	1 GHz	Fetch/Decode/Commit	4
ROB/LSQ	64/32 entries	Int/FP/Mem IW	24/16/16 entries
Functional units	3 int ALUs, 2 mem ports, 3 FP units	STB/L2–iLP-NUCA WB/L3 WB	32/16/16 entries
Issue width	4(Int+Mem) + 2FP	TLB miss latency	30 cycles
Branch Predictor	bimodal + gshare	Miss. branch penalty	6 cycles
L1/L2/L3 MSHR	8/8/4 entries	MSHR second. misses	4
Baseline L1/RT [a]	32KB, 4Way, 32B BS, parallel AM, 2-cycle lat, 1-cycle init		
L2-I/L2-D	512KB, 8Way, 32B BS, serial AM, 4-cycle lat, 2-cycle init		
iLP-NUCA rest tiles	32KB, 2Way, 32B BS, parallel AM, levels: 3 (448KB)		
L3	4MB, eDRAM, 16Way, 128B BS, 14-cycle lat, 7-cycle init		
Main Memory	100 cycles/4 cycle inter chunk, 16 Byte bus		

[a] L1: write-through; RT: copy-back; L1/RT: write-around and 2 ports.

We estimate cache access latencies and energy consumption assuming 32nm LSTP (Low STandby Power) technology with Cacti 6.5 [14]. For iLP-NUCA networks-in-cache we derive latency and consumption figures by scaling the data obtained from a real 90nm layout [16].

3.2 Workloads

We use the full SPEC CPU2006 benchmark suite [9], but 483.xalancbmk (which could not be executed in our environment). For each program we simulate 100M representative instructions that were selected following the SimPoints methodology [8]. Caches and branch predictor are warmed up during 200M instructions for one thread experiments. Multi-thread experiments (2 and 4 threads) are multiprogrammed. We utilize *last* as simulation ending policy and collect statistics for the first 100M instructions of each thread. For two thread experiments we run all the benchmarks combinations. For four thread experiments, we assure results are representative by using statistical sampling and taking enough samples to reach 97% of confidence level and less than 3% error [17],[20].

3.3 Metrics

We use two system oriented performance metrics for multiprogrammed workloads: IPC throughput (i.e., committed user instructions summed over all threads divided by total elapsed cycles[1]) and fairness [5], according to the formulas:

$$IPC\ throughput = \sum_{i=1}^{n} IPC_i \qquad fairness = \frac{\min\limits_{i}\left(\frac{CPI_i^{MT}}{CPI_i^{ST}}\right)}{\max\limits_{i}\left(\frac{CPI_i^{MT}}{CPI_i^{ST}}\right)}$$

where IPC, CPI, ST and MT refer to instructions per cycle, cycles per instruction, single thread, and multi-threaded execution, respectively.

Regarding energy consumption, we report the total energy consumed by the cache hierarchy. We compute the Energy-Delay and Energy-Delay^2 products taking into account the energy consumption of the memory hierarchy[2] and the execution time (delay). We follow the Li *et al.* approach for SMT environments [11], and account for all the energy consumed until the last thread commits 100M instructions.

4 Evaluation

4.1 Impact of the Tree-Based Transport Network

The new tree-based network-in-cache reduces the tiles degree regarding the former 2-D mesh. The transport components (see Fig. 2c) are the two input transport buffers and the switch. From the hardware implementation of LP-NUCA [16], we know that each buffer accounts for one third of the energy consumption, and so does the switch. Our new network simplifies the switch removing one output link. From the original layout we estimate that these changes reduce the components energy consumption by 20%.

[1] The use of IPC throughput is safe because there are no synchronization instructions.

[2] In general, we have observed that the processor activity implementing iLP-NUCA decreases slightly, due to a reduction in the speculation depth.

We compare the former 2-D mesh with the tree-based transport network in a system with a 3–level iLP-NUCA (4-way, 32KB root-tile).

Our results show that the tree-based transport network reduces the average service latency by 8%, with bigger gains in tiles located on more distant levels. Thus, the effectiveness of the new network strongly depends on the amount of reused blocks that those levels capture, which tends to be low except in some benchmarks, such as 447.dealII. Finally, the tree-based network does not experience contention because the number of simultaneous requests is low.

Energy savings are more noticeable when executing several threads. With 2 threads the new transport network consumes 4.7% less energy than the former 2-D mesh; with 4 threads energy savings reach 6%.

4.2 Instruction Cache Energy/Performance Trade-offs

We compare two alternative instruction cache hierarchies (CV and iLP), as explained in Sect. 3.1. In this section we assume a perfect L1 data cache, so that result variations will only come from instruction activity. We are also interested in analyzing the sensitivity of CV and iLP to a possible L1-I/RT downsizing. Hence, we reduce the L1-I/RT size and associativity from 4-way, 32KB (baseline) to 2-way 16KB, 8KB, and 4KB. Next, we show results of energy, performance, and fairness for all the above configurations, considering the system loaded by one, two, or four threads.

Energy Consumption. Figure 3 shows the total energy consumption for 1, 2, and 4 threads. Each bar corresponds to one configuration, and represents the added energy consumption of all the executed workloads. We plot the dynamic energy of the different caches accessed by instructions (L1-I, L2-I, and L3), and group the total static energy on top. In order to analyze the results in Fig. 3 we first highlight some obvious facts; namely, i) the dynamic energy of L1-I and RT matches, as both caches have a very similar complexity; ii) the static energy is almost constant for a given number of threads, except for the smallest L1-I cache in the CV system loaded with 4 threads, due to its significant slowdown; and iii) the L3 cache spends a negligible amount of dynamic energy, because hitting in L2-I is the norm. Bearing this in mind, to gain an insight into the best choice we should concentrate on the energy transfer between the L1-I and the L2-I as we change L1-I size and number of threads.

In configurations with 32KB 4-way L1-I/RT, the dynamic energy consumed by the L2-I or by the levels 2 and 3 of iLP-NUCA (Le2+Le3 in Fig. 3) is similar and very small, though it increases as the number of threads grow. We can conclude that the instruction footprints of most applications fit in the L1-I cache.

As we shrink the L1-I/RT size, the energy per access decreases and so does the total energy of L1-I/RT. However, smaller L1-I/RT caches fail to capture the required footprint and the number of misses increase. In turn, the growing number of first-level misses increases the dynamic energy consumption in L2-I or Le2+Le3, and, eventually, the total energy consumption increases. Underloaded

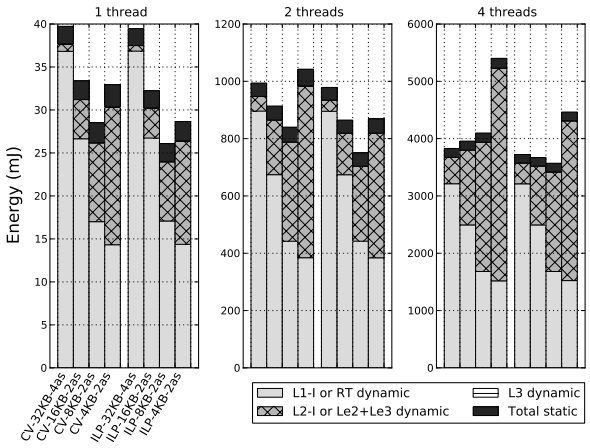


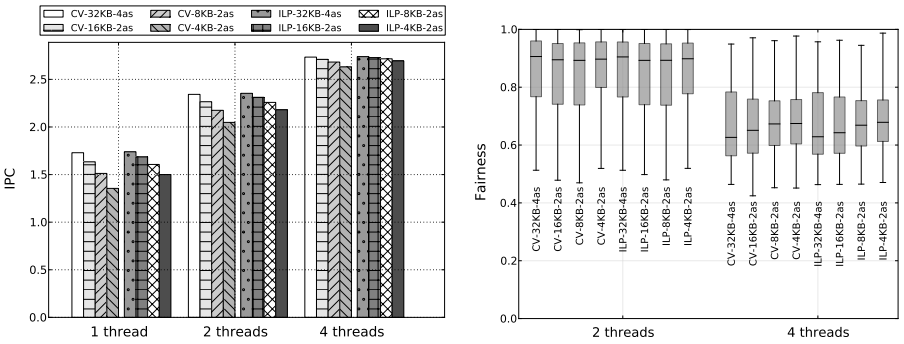
Fig. 3. Energy consumption of several CV and iLP cache configurations—labeled with type (CV vs. iLP) and L1-I/RT size and associativity—for 1, 2, and 4 threads.

systems, executing one or two threads, present a similar behavior: overall energy reduction for 16KB and 8KB L1-I/RT caches, and overall increase for 4KB. However, in the fully loaded system, when executing four threads, as the first level shrinks and the working set moves to the second level, the behavior of CV and iLP clearly departs. CV hierarchies do not allow L1-I reduction, because the energy transfer between levels is always detrimental for the overall spending. In contrast, iLP hierarchies benefit from RT reduction up to 8KB. All in all, for any configuration and number of threads, iLP-NUCA always consumes less energy than the conventional hierarchy, and the preferable configuration is an iLP-NUCA with a 2-way 8KB RT.

Performance and Fairness. Figure 4a shows the IPC throughputs harmonic mean and Fig. 4b the fairness distributions for the considered configurations.

When reducing the first-level size the miss ratio increases, and performance lowers in both CV and iLP systems. With iLP-NUCA the performance degradation is lower: for the same size of L1-I/RT, iLP-NUCA performs better, regardless of the number of threads. In the conventional system loaded with a single thread, as L1-I reduces to 16KB, 8KB, and 4KB, the IPC throughput decreases 5.8%, 12.5%, and 20%, respectively. In the iLP-NUCA system, performance also decreases, yet the gradient is less and the 32KB IPC throughput is slightly better.

When considering two threads, the former observations hold, but now the sensitivity to the first-level cache size is softened by the ability of SMT to hide long latency operations by interleaving useful work from other threads. In the conventional system loaded with two threads, as L1-I reduces to 16KB, 8KB, and 4KB, the IPC throughput reduces 3%, 6.3%, and 10.5%, respectively, while in the iLP system the reductions are 1.1%, 3.2%, and 6%, respectively.



(a) Average IPC throughput. (b) Fairness distribution.

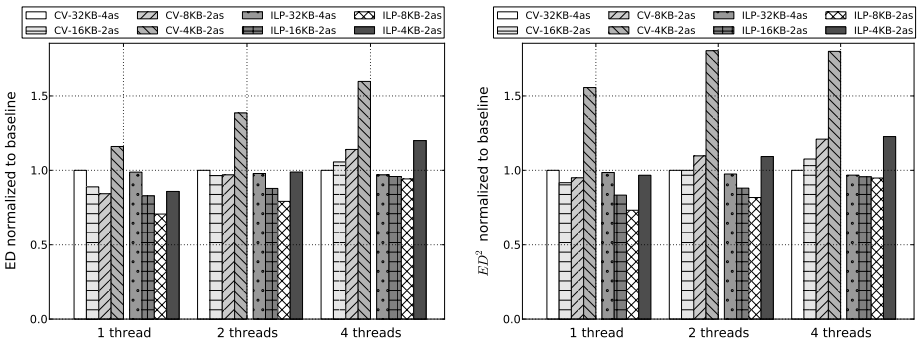
Fig. 4. Harmonic mean of IPC throughputs and fairness distribution—labeled with type (CV vs. iLP) and L1-I/RT size and associativity.

With four threads differences are smaller. The occupancy of the functional units is higher and there are more threads to hide the processor stalls due to instruction misses. As the first level shrinks, iLP-NUCA keeps performance degradation below 1.5%, while conventional caches lose up to 3.5%, on average.

As expected, since we keep the first-level cache latency constant, from a performance standpoint it is always better to select the largest size. However, if we can afford some losses when the system is underloaded (one or two threads), an iLP system with 16KB or 8KB will keep an acceptable performance and use less energy. For example, an iLP system implementing a 8KB RT would reach the performance of a CV system with a 16KB L1-I, but saving 21%, 18%, and 11% of energy, for 1, 2, and 4 threads, respectively.

Regarding the fairness among threads, let us consider Fig. 4b where each candlestick represents the minimum, the quartile 25, the median, the quartile 75, and the maximum of the fairness distribution. A fairness figure close to 1.0 means an even resource distribution. Lower figures mean unfair execution, where some threads are slowed down much more than others. As we can see in Fig. 4b, the fairness distribution for two and four threads is quite similar in CV and iLP systems. We can conclude that implementing an iLP system is not detrimental at all from a fairness standpoint.

Energy-Delay and Energy-Delay^2. Figure 5 shows the Energy-Delay and Energy-Delay^2 products normalized to the baseline (CV-32KB-4ass). Configurations with iLP-NUCA present better (lower) ED and ED^2 results, independent of the number of threads. We find bigger differences when the L1-I cache is smaller and the pressure on the next levels higher. Again, the optimal configuration would be an iLP-NUCA with a 2-way, 8KB RT, whose normalized ED values are 0.70, 0.8, and 0.94, for 1, 2, and 4 threads respectively. ED^2 values are 0.73, 0.82, and 0.95, respectively.



(a) Energy–Delay product.

(b) Energy–Delay^2 product.

Fig. 5. ED (Fig. 5a) and ED^2 (Fig. 5b) products for 1, 2, and 4 threads. Values are normalized to baseline (CV-32KB-4ass).

4.3 Evaluating iLP-NUCA with a Non-ideal Data Cache Hierarchy

We evaluated our proposal with a non-ideal data path, using a conventional L1-D/L2-D pair, and the original data LP-NUCA. We just summarize some results due to the lack of space. The benefit now is twofold: we reduce the fetch latency on the instruction side (iLP-NUCA), and we reduce the load latency as well as increasing the store bandwidth on the data side (LP-NUCA). This allows for energy reductions up to 28%, 28%, and 27% for 1, 2, and 4 threads, respectively, while achieving greater IPC throughput, on average, regardless of the number of threads. Thus, former conclusions hold, encouraging the use of iLP-NUCA. A 2-way, 8KB RT is reinforced as the best candidate for our workloads.

5 Related Work

Several works have addressed the instruction cache energy consumption problem by adding hardware structures between the processor and the first-level instruction cache (L1-I) to capture instruction fetches [3],[10],[21]. These proposals add extra hardware structures between the processor and the instruction cache. They reduce the amount of access to the L1-I, but at the expense of introducing extra fetch latency when a miss to these structures occurs. On the contrary, we keep the interface between the processor and the L1-I, and these designs could be implemented in front of our system.

Reconfigurable caches try to adapt dynamically to applications requirements obtaining great energy reductions, but they modify the cache structures [1],[18],[22]. A variable size or associativity allows the hosting of applications or program phases with big working set requirements and reduces the energy consumption when the requirements are small. Apart from complexity, a reconfiguration drawback appears when predictions are wrong and the cache becomes

under- or over-sized. Nonetheless, merging reconfiguration with iLP-NUCA would be an interesting future work, for example switching on/off iLP-NUCA levels.

These works do not consider multi-threaded applications, where several threads that run together and share a small structure (such as a filter or victim cache) could interfere each other by evicting useful blocks of other threads.

6 Conclusions

Current high-performance embedded processors require a powerful instruction cache hierarchy with low energy consumption. Most designs rely on large first level instruction (L1-I) caches that, for many applications and a low number of executing threads, spends an undue amount of energy. However, shrinking L1-I caches naively harms the performance of many applications, specially when the processor becomes loaded up to its maximum number of threads. In order to simultaneously improve the performance and energy consumption of the instruction cache hierarchy, we propose iLP-NUCA, a structure that replaces a conventional L1-I/L2-I cache pair by several levels of richly interconnected cache tiles. We provide iLP-NUCA with a new transport network-in-cache, reducing blocks average service latency by 8%, and the energy consumption of the network.

We compare our proposal with a state-of-the-art conventional three level cache hierarchy, where L1-I and L2-I are dedicated to instructions, and L3 is shared. From our experiments we can conclude that iLP-NUCA performs better and consumes less energy for the same L1-I/RT size, independently of the number of threads. Furthermore, iLP-NUCA achieves the performance of a conventional hierarchy with a double sized L1-I, saving a great amount of energy. For instance, if we compare a conventional 2-way, 16KB L1-I against an iLP-NUCA with a 2-way, 8KB RT, we found equal performance and energy savings of 21%, 18%, and 11% for 1, 2, and 4 threads, respectively.

Acknowledgments. This work was supported in part by grants TIN2010-21291-C02-01 (Spanish Government, European ERDF), gaZ: T48 research group (Aragón Government, European ESF), Consolider CSD2007-00050 (Spanish Government), and HiPEAC-3 NoE. The authors would like to thank Manolis Katevenis for his suggestions about the transport network.

References

1. Albonesi, D.H.: Selective cache ways: on-demand cache resource allocation. In: Proc. of the 32nd Ann. ACM/IEEE Int. Symp. on Microarchitecture. pp. 248–259 (1999)
2. Austin, T., Burger, D.: The simplescalar tool set, version 2.0. Tech. Rep. CS-TR-97-1342, University of Wisconsin Madison (1997)
3. Bellas, N., Hajj, I., Polychronopoulos, C., Stamoulis, G.: Architectural and compiler techniques for energy reduction in high-performance microprocessors. IEEE Trans. on Very Large Scale Integration Systems 8, 317–326 (June 2000)

4. Byrne, J.: Freescale drops quad-core threshold. Microprocessor Report 26, 10–12 (July 2012)
5. Gabor, R., Weiss, S., Mendelson, A.: Fairness and throughput in switch on event multithreading. In: Proc. of the 39th Ann. IEEE/ACM Int. Symp. on Microarchitecture. pp. 149–160 (2006)
6. Gwennap, L.: What's inside the Krait. Microprocessor Report 26, 1–9 (June 2012)
7. Halfhill, T.R.: Netlogic broadens XLP family. Microprocessor Report 24, 1–11 (August 2010)
8. Hamerly, G., Perelman, E., Lau, J., Calder, B.: SimPoint 3.0: Faster and more flexible program analysis. In: Journal of Instruction Level Parallelism (2005)
9. Henning, J.L.: SPEC CPU2006 benchmark descriptions. SIGARCH Comput. Archit. News 34, 1–17 (2006)
10. Kin, J., Gupta, M., Mangione-Smith, W.: The filter cache: an energy efficient memory structure. In: Proc. of the 30th Ann. IEEE/ACM Int. Symp. on Microarchitecture. pp. 184–193 (1997)
11. Li, Y., Brooks, D., Hu, Z., Skadron, K., Bose, P.: Understanding the energy efficiency of simultaneous multithreading. In: Proc. of the 2004 Int. Symp. on Low Power Electronics and Design. pp. 44–49 (2004)
12. LSI Corporation: PowerPC™ processor (476FP) embedded core product brief, `http://www.lsi.com/DistributionSystem/AssetDocument/PPC476FP-PB-v7.pdf` (January 2010)
13. Montanaro, J., Witek, R., Anne, K., Black, A., Cooper, E., Dobberpuhl, D., Donahue, P., Eno, J., Farell, A., Hoeppner, G., Kruckemyer, D., Lee, T., Lin, P., Madden, L., Murray, D., Pearce, M., Santhanam, S., Snyder, K., Stephany, R., Thierauf, S.: A 160 MHz 32 b 0.5 W CMOS RISC microprocessor. In: Proc. of 1996 IEEE Int. Solid-State Circuits Conference. Digest of Technical Papers. pp. 214–215, 447 (1996)
14. Muralimanohar, N., Balasubramonian, R., Jouppi, N.: Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In: Proc. of the 40th Ann. IEEE/ACM Int. Symp. on Microarchitecture. pp. 3–14 (2007)
15. Segars, S.: Low power design techniques for microprocessors. ISSCC Tutorial note (February 2001)
16. Suárez, D., Dimitrakopoulos, G., Monreal, T., Katevenis, M.G.H., Viñals, V.: LP-NUCA: Networks-in-cache for high- performance low-power embedded processors. IEEE Trans. on Very Large Scale Integration systems 20, 1510–1523 (August 2012)
17. Suárez, D., Monreal, T., Viñals, V.: A comparison of cache hierarchies for SMT processors. In: Proc. of the 22nd Jornadas de Paralelismo (2011)
18. Sundararajan, K.T., Jones, T.M., Topham, N.: Smart cache: A self adaptive cache architecture for energy efficiency. In: Proc. of the Int. Conference on Embedded Comp. Systems: Architectures, Modeling, and Simulation. pp. 41–50 (July 2011)
19. Tullsen, D.M., Eggers, S.J., Levy, H.M.: Simultaneous multithreading: maximizing on-chip parallelism. In: Proc. of the 22nd Ann. Int. Symp. on Comp. Arch. pp. 392–403 (1995)
20. Wackerly, D., Mendenhall, W., Scheaffer, R.L.: Mathematical Statistics with Applications. Brooks/Cole Cengage Learning, 7th edn. (2008)
21. Yang, C.L., Lee, C.H.: Hotspot cache: joint temporal and spatial locality exploitation for i-cache energy reduction. In: Proc. of the 2004 Int. Symp. on Low power electronics and design. pp. 114–119 (2004)
22. Zhang, C., Vahid, F., Najjar, W.: A highly configurable cache for low energy embedded systems. ACM Trans. Embed. Comput. Syst. 4, 363–387 (May 2005)