Light NUCA: a proposal for bridging the inter-cache latency gap

Darío Suárez†, Teresa Monreal†, Fernando Vallejo‡, Ramón Beivide‡, and Víctor Viñals†
†gaZ-DIIS-I3A
Universidad de Zaragoza, Spain
‡ATC
Universidad de Cantabria, Spain
HiPEAC Network of Excellence
Email: {dario, tmonreal, victor}@unizar.es
Email: fernando@atc.unican.es, ramon.beivide@unican.es

***Abstract*—To deal with the "memory wall" problem, microprocessors include large secondary on-chip caches. But as these caches enlarge, they originate a new latency gap between them and fast L1 caches (inter-cache latency gap). Recently, Non-Uniform Cache Architectures (NUCAs) have been proposed to sustain the size growth trend of secondary caches that is threatened by wire-delay problems. NUCAs are size-oriented, and they were not conceived to close the inter-cache latency gap. To tackle this problem, we propose Light NUCAs (L-NUCAs) leveraging on-chip wire density to interconnect small tiles through specialized networks, which convey packets with distributed and dynamic routing. Our design reduces the tile delay (cache access plus one-hop routing) to a single processor cycle and places cache lines at a finer-granularity than conventional caches reducing cache latency. Our evaluations show that in general, L-NUCA improves simultaneously performance, energy, and area when integrated into both conventional or D-NUCA hierarchies.**

I. Introduction

As technology scales, the latency gap between the processor and main memory widens forcing a size and latency increment in secondary caches. So, at the same time these large caches reduce the gap to main memory, they widen an inter-cache latency gap between them and fast L1 caches. To bridge this gap, two main approaches can be followed: reducing the latency of secondary caches or increasing the size of first level caches without compromising their latency and bandwidth.

Within the first approach, Kim *et al.* proposed Non-Uniform Cache Architectures [1]. NUCA connects individually accessible cache banks in a 2D-mesh. NUCA pioneers inter-bank block migration techniques, but has solely focused on large secondary caches. In respect to the second approach, Balasubramonian *et al.* provided evidence of the latency/size trade-off between L1 and L2 caches. They proposed a reconfigurable cache able to dynamically adjust its size to the working set [2]. However, this scheme only supports single-ported cells, and it may not be able to provide the high bandwidth that superscalar processors require.

The present work tries to close the latency gap between secondary on-chip caches and fast L1 caches by enlarging the cache accessible by the processor at low latencies without degrading bandwidth. Our proposal is based on a light dynamic NUCA formed by small cache banks that benefits from the fine granularity and working set adaptability of the Balasubramonian approach and from the non-uniform access time and block-migration techniques of NUCAs. To make feasible this idea, we have to fight against the reasons that, up to now, have prevented NUCAs to be used with small cache banks. Some of them follow.

To maximize performance, NUCA banks tends to be large [3]. Since NUCA network mechanisms have been designed for multi-megabyte caches, they focus on density rather than latency and bandwidth; e.g., the 2D-mesh network with wormhole routing requires at least one routing cycle before and after accessing any bank. Besides, NUCAs employ a single-injection point and shared banks with multiple cycle initiation rate that can stall the network in miss bursts.

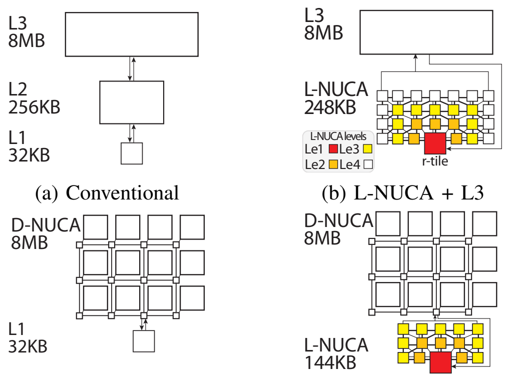
Since NUCA latency and bandwidth mostly depends on its networking, L-NUCA focuses on improving topologies, routing and packet delivery. In an L-NUCA cache, the L1 cache is surrounded by a set of small tiles connected by three on-chip networks specially tuned for different cache operations. L-NUCA tiles manage a cache access and one hop routing within a single cycle. This allows for placing blocks at latencies inversely proportional to their temporal locality at a finer granularity than conventional or NUCA hierarchies.

The rest of this paper is organized as follows. Section II presents the main features of L-NUCAs. Section III details the topologies, routing algorithms and flow control policies. Sections IV and V describe the experimental methodology and evaluate the results. Section VI comments on related work, and Section VII concludes the paper.

II. L-NUCA Overview and its Integration in the Cache Hierarchy

We have studied L-NUCA in two representative environments: a conventional 3-level hierarchy in which an L-NUCA replaces the L2 cache (Figs. 1(a) and 1(b)) and a D-NUCA hierarchy in which an L-NUCA is placed between the L1 and the D-NUCA (Figs. 1(c) and 1(d)).

The processor interfaces the L-NUCA through the *root-tile* (r-tile), which is a conventional L1 cache extended with the flow control logic required for sending and receiving blocks. The rest of tiles surround the r-tile and are interconnected only through local links in order to minimize wire delay. To simplify block migration, all the tiles share the same block size. Block search is efficiently performed by grouping tiles into growing size levels. For example, the L-NUCA of Fig. 1(b) has 4 levels, named Lei. The r-tile forms the first



(a) Conventional (b) L-NUCA + L3 (c) D-NUCA (d) L-NUCA + D-NUCA

Fig. 1. Cache hierarchies studied. L-NUCA levels from Le2 to Le4 are made of 8KB tiles.

level (Le1), the 5 tiles surrounding it the second one (Le2), and so on...

L-NUCA operation is simple yet efficient. Tiles only have knowledge of their local cache contents; hence, when the r-tile misses, it propagates the miss outwards to the Le2 tiles. If any of them hits, it sends the requested block back to the r-tile and stops propagating the miss. At the same time, the remaining Le2 missing tiles are sending out the miss request to their Le3 leaf tiles. In the next cycle, the requested Le3 tiles will miss and propagate the request to Le4. Eventually, when all Le4 tiles miss, the request is forwarded to the L3 cache. Incoming blocks from the L3 and tile hits are directly sent to the r-tile. After completing enough requests the r-tile becomes filled and in order to refill an incoming block from the L3, the r-tile will evict a victim block to an Le2 tile. The Le2 destination tile will repeat the eviction operation if the corresponding set is full; therefore, L-NUCA tiles act like a distributed victim cache similarly to the L2 in [4].

Because tile latency is set to one processor cycle, each L-NUCA level can be looking up for a different request during the same cycle. Hence, contrary to conventional caches, the number of in-flight requests increases with L-NUCA size increments. This fact together with the reduction on average cache latency—due to the better exploit of temporal locality—are the key to success of L-NUCAs.

III. Networks and Routing in L-NUCA

L-NUCA networks pursuit the following goals: (i) maximize the hit ratio, (ii) minimize the hit time, (iii) minimize the miss determination time, and (iv) match the miss rate bandwidth of the r-tile and keep it even though L-NUCA size increases.

Concerning (i), L-NUCA capacity is maximized by managing tile contents in exclusion. Conflict misses are reduced by placing no restriction in block mapping into tiles. The remaining goals impact the implementation of the three basic operations in L-NUCA: block search, block transport or service, and block replacement. Search operation requires a low-latency miss propagation network, (ii), along with a fast method for determining global misses, (iii). Both requirements together with goal (iv), call for integrating the tile cache access and one-hop routing within a single processor cycle. Transport operation requires a quick block delivery to the r-tile (ii). The replacement operation must contribute to goal (i), exploiting as much as possible the temporal locality.

A. Topologies

The small L-NUCA tiles (2 to 8KB) ensure a short inter-tile distance, so small pitch metal layers can be used to route a large number of wires among tiles. In order to leverage this on-chip wire-availability [5], we advocate to replace the NUCA 2D-mesh with 3 dedicated point-to-point networks with unidirectional links, one for each cache operation: the *Search*, *Transport*, and *Replacement* networks.

The *Search* network (Fig. 2(a)) uses a broadcast-tree for propagating miss requests very fast with the minimum number of links. Besides, the maximum distance is only increased by one hop when adding an L-NUCA level. A NUCA 2D-mesh network would double the number of required hops to reach all the tiles, would increase the number of links by more than 50%, and would add 2 hops to the maximum distance when adding a new level.

The Search network also collects global misses and forwards them to the next cache level. Global miss determination requires only to gather the miss status of all last-level tiles because when a tile experiences a hit it stops propagating the Search message. Global misses may be efficiently determined with a segmented miss-line similar to the hierarchical bit-lines of SRAMs [6]. We assume that this operation takes one-cycle after the last-level search.

The *Transport* operation (Fig. 2(b)), employs a 2D-mesh that offers multiple return paths to the r-tile ensuring path diversity and low contention even when the same tile hits during consecutive cycles.

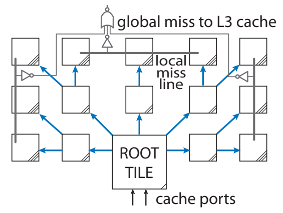
The *Replacement* network (Fig. 2(c)) connects tiles whose latencies differ in one cycle, except for the r-tile, by means of an irregular topology with the lowest possible degree[1]. This topology tries to maintain blocks ordered by their temporal locality and to keep them as much as possible; hence, when a level is added the distance from the r-tile to the upper corner tiles—the only tiles that evict blocks to the next cache level—increases by 3 hops.

Transport and replacement are completely decoupled in L-NUCAs. All hits are directly transported to the r-tile (multi-hop messages), and evictions operate like a "domino" (single-hop messages) among tiles. These evictions finish when an empty way is found or a block is evicted to the next cache level.

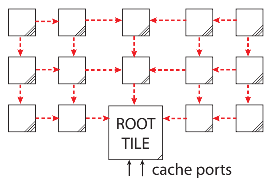
B. Headerless Messages, Distributed Routing, and Flow Control

Each network transfers its own message. Since the message destination is implicit in all the networks, and their topologies

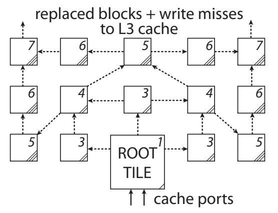
[1]The degree of a node is the sum of all its input and output links. Low degrees reduce network complexity.



(a) Broadcast-tree Search network.



(b) 2-D mesh Transport network.



(c) Latency-driven Replacement network.

Fig. 2. Network topologies for a 3-level L-NUCA. All links are unidirectional. In (c), values represent the tile latency assuming 1-cycle tiles. Tile latency includes search, tile access, and transport delay.

ensure that all output links are valid for every message, L-NUCA employs headerless messages reducing routing delay and the size of links, buffers, and crossbars.

Routing a Search message simply consists in sending it to all tile leaves. This "wired" multicast avoids the complexity of multicast in conventional routers [7].

Transport and Replacement networks route headerless messages with a dynamic distributed algorithm. Every node randomly selects an output link. This algorithm reduces contention in comparison to dimensional order routing where all the messages with the same source and destination take the same route.

In respect to flow control, it is not required by the Search network because Search messages cannot be blocked. Transport and Replacement networks rely on buffered flow control. Since links are message-wide, the flow control digits (flits) are the messages themselves. To avoid message dropping, L-NUCAs use store-and-forward flow control with On/Off back-pressure and two-entry buffers per link—round-trip delay between tiles is two cycles [5].

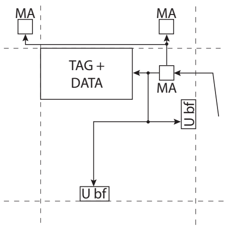
Since hit blocks move down toward the r-tile and evicted ones move up, we call respectively downstream (D buffers), and upstream (U buffers), to the Transport and Replacement flow control buffers.

The lack of cyclic dependencies among messages with the guarantee of message consumption ensures that L-NUCAs are deadlock-free, not requiring virtual channels.

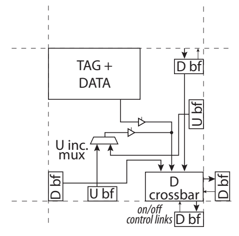
C. Parallel Cache Access and One-Hop Routing in a Single-Cycle

Now, we analyze the critical paths of Search, Transport, and Replacement operations, looking carefully at the most demanding timing path: the integration of a cache access plus one-hop transport routing within a single processor cycle. In the following, we will focus on the upper left corner tile in the second L-NUCA level as it has the maximum number of links a tile can require. Figures 3(a), 3(b) and 3(c) show each network components.

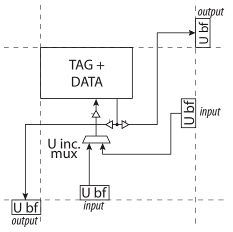
a) Search operation: Fig. 3(a). It begins when a tile receives a miss request in its Miss Address register (MA). The requested address is looked up simultaneously in the tag array and in the U buffers, which include an address comparator per entry (up to 4 per tile). Looking up in the U buffers is enough for finding blocks in transit across the Replacement network (if any) avoiding false misses.



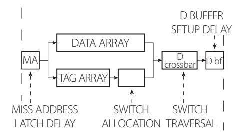
(a) Search network components.



(b) Transport network components.



(c) Replacement network components.



(d) Timing diagram of the critical path.

Fig. 3. Tile organization with the components involved in each operation for an example tile having the highest degree. MA, D bf, and U bf stand for Miss Address register, Downstream (Transport) buffer, and Upstream (Replacement) buffer, respectively.

If the cache or the U buffers hit, a Transport operation starts. Otherwise, the request is propagated to the MA registers of the neighbor leaf tiles. Tile look-up and miss propagation is done in a single cycle because cache miss determination takes less time than the whole cache access. For the small caches and low-associativity employed in L-NUCA tiles, the delay until the tag comparison represents roughly 80% of the total delay measured with Cacti 5.3 [8].

b) Transport operation: Fig. 3(b). It consists on routing hit blocks either from the tile D buffers or from the cache. In both cases, blocks are sent through the D crossbar and stored into a D buffer of a neighbour tile. In the rare event that a tile hits and all its output D channels are Off, the tile sends a contention-marked message through the Search network. When the marked Search message arrives to the global miss logic, it is returned to the r-tile restaring the Search operation. We have observed that this event rarely occurs.

c) Replacement operation: Fig. 3(c). It is only carried out during Search idle cycles and requires two, non necessarily consecutive, cycles. In the first one, the control logic checks whether any of the output U channels are On, if so a victim block is read out from the tile cache, sent through the selected output U channel, and stored in the corresponding output U buffer. In the second cycle, the incoming block is written in the cache from the input U buffer.

In summary, the critical timing path in an L-NUCA tile is a hit search followed by a transport operation. Existing caches and routers already accomplish both tasks sequentially in a very low number of FO4s [6], [9]. Since our network subsystem is much simpler and performs some routing tasks in parallel with the cache access, we believe that both tasks fit perfectly in a single cycle. To verify this assumption, let's consider the timing diagram of L-NUCAs, see Fig. 3(d), and compare it with the multiple stages that conventional virtual channel router performs [10]:

Decoding and *routing*. Headerless messages allow to remove this stage.

Virtual channel allocation. This stage is completely avoided because L-NUCAs do not employ virtual channels.

Switch allocation. This stage assigns output channels to input requesters. Its delay depends upon how many resources have to be assigned and how complex is the assignment algorithm. Since switch allocation depends only on the number of occupied D buffers and on the results of the miss address comparison (tag array access and U comparators), it can be performed in parallel with the data array access; therefore, its delay can be overlapped with the cache access.

Switch traversal. This stage sends messages through the crossbars and is not demanding because the 5 crossbar inputs (2 D buffers, 2 U buffers, and the cache) can be reduced to 3. Since content exclusion ensures that only a single copy of each block exists, hits can not simultaneously happen in the cache and in the U buffers. This favors the use of a cut-through crossbar reducing the number of inputs, latency, and energy [11].

Summarizing, L-NUCAs only add a low-latency *switch traversal* stage to the critical path, so we conclude that a cache access and one-hop routing can both fit within one processor clock cycle.

D. Coherence Support

To maintain coherence, L-NUCAs will require the same apparatus as a conventional L1 cache. Anyway, inclusion has to be enforced; e.g., by explicit invalidation. The coherency point would be the next cache level and its directory would include status vectors to check whether blocks are in the L-NUCA (similar to the Piranha CMP [4]).

IV. Experimental Framework

We have extended Simplescalar/Alpha 3.0d with: reorder buffer, issue windows, speculative wake-up and selective recovery, payload and register file stages, accurate timing models for caches, buses, network contention, and flow control. All caches use LRU replacement, and the 8MB L3 is similar to the Intel Core 2 [12]. D-NUCAs are modelled based upon the SS-performance configuration in [1]. For the rest of parameters see Table I.

We use all but one (*483.xalancbmk*[2]) SPEC CPU 2006 benchmarks with the input sets suggested in [13]. Simulation comprises a run of 100 million instruction after a warm-up of 200 millions following the SimPoint guidelines [14].

As regards delay, energy, and area, we assume a cycle time of 19 FO4s similar to the Intel Core2 Duo E8600 in a 32 nm technology [15]. Cache area, delay, and energy are estimated with Cacti 5.3 [8] improved to support access time as an optimization objective. L3 caches employ Low Operating Power (LOP) transistors and the rest of caches High Performance (HP) ones. Within our cycle time constraints the largest configuration found for the one-cycle L-NUCA tile was an 8KB-2Way-32B cache. The area and energy of routers have been estimated with Orion [16]. The transport crossbar delay was modelled with HSPICE to verify it satisfies the cycle time constraints.

V. Result Evaluation

To prove the effectiveness of L-NUCA caches, we evaluate them in terms of area, performance, and energy in two scenarios: backed by a conventional L3 and by a D-NUCA.

A. L-NUCAs vs. Conventional Hierarchies

The selected baseline configuration (*L2-256KB*) was the most performance in our exploration of the L2 design space for three-level conventional hierarchies. This baseline is compared to three L-NUCAs (2, 3, and 4 levels).

TABLE II
Conventional and L-NUCA areas.

	L1 + L2 /L-NUCA area (mm^2)	tile area (mm^2)	network area percentage (%)
L2-256KB	0.91	—	—
LN2-72KB	0.46	0.06	14.01
LN3-144KB	0.86		18.8
LN4-248KB	1.59		19.02

Regarding area (Table II), the L-NUCA low network overhead with their small size keep area under control; e.g., *LN3-144KB* saves a 5.3% compared to *L2-256KB* while overpassing its performance.

Regarding performance, Fig. 4(a) shows the IPC for all configurations. All L-NUCAs overpass *L2-256KB* with average gains ranging from 5.4% (*LN2-72KB*) to 6.22% (*LN4-248KB*) in Integer and from 14.3% (*LN2-72KB*) to 15.4% (*LN4-248KB*) in Floating Point. In L-NUCAs of 4 levels and beyond, performance increments do not pay-off for the rise in area.

L-NUCA gives consistent gains across benchmarks because it reduces the average latency of cache hits. To give insight into this reasoning, left part of Table III shows the average percentage of L-NUCA read hits relative to the *L2-256KB* ones. On average, 59.6% and 40.9% of L2 hits are serviced by the 5 Le2 tiles at a smaller latency. Besides, these hits do not suffer from contention as the right part of Table III proves. These two columns show the ratio between the average

[2]Its execution produced a known stack overflow problem.

TABLE I
ARCHITECTURAL AND NETWORK PARAMETERS.

Fetch/Decode width	4, up to 2 taken branches	Issue width	4(INT or MEM)+4 FP	Store Buffer size	48
Branch predictor	bimodal + gshare, 16 bit	Commit width	4	L2/L3 Write Buffer size	32/32
ROB/LSQ size	128/64	MSHR L1/L2/L3 size	16/16/8	TLB miss latency	30
INT/FP/MEM IW size	32/24/16	MSHR sec. misses	4	Branch misspred. delay	8

L1 cache / r-tile	32KB, 4 Way, 32B block size, parallel access mode, 2-cycle completion, 1-cycle initiation, write-through, 2 ports, read hit ener.: 21.2 pJ, leakage power: 12.8 mW
L2 cache	256KB, 8 Way, 64B block size, 4 cycle completion, 2 cycle initiation, serial access mode, copy-back, 1 port, read hit ener.: 47.2 pJ, leakage power: 66.9mW
L-NUCA cache tile	8KB, 2 Way, 32B block size, parallel access mode, 1-cycle completion and initiation, copy-back, 1 port, read hit ener.: 14 pJ, leakage power: 2.2 mW
L3 cache	8MB, 16 Way, 128B block size, 20-cycle completion, 15-cycle initiation, copy-back, 1 port, read hit ener.: 20.9 pJ, leakage power: 600 mW
D-NUCA cache	8MB, 8 sparse sets, 4 rows. Banks: 256KB, 2 Way, 128B block size, parallel access mode, 3-cycle completion and initiation, 1 port, read hit ener.: 131.2 pJ, leakage power: 33.5 mW. Links: 256 bits
Main memory	First chunk: 200 cycles, 4-cycle inter chunk, 16B wires

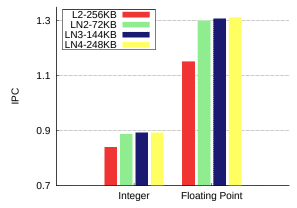
	D-NUCA	L-NUCA
Flit Size	32B	32B
Buffer size per channel	4 (virtual)	2 (physical)
Routing latency	1	1

	D-NUCA	L-NUCA
Flits per message	1-5	1
Virtual channels	4	—
MSHR size	16	16

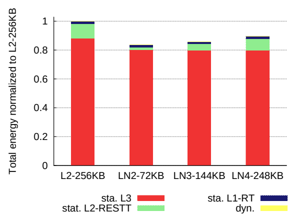
TABLE III
PERCENTAGE OF THE NUMBER OF READ HITS IN EACH L-NUCA LEVEL RELATIVE TO THE NUMBER OF READ HITS IN THE L2 OF *L2-256KB* AND AVERAGE TO MINIMUM TRANSPORT NETWORK LATENCY RATIO.

	Le2 / L2 (%)		Le3 / L2 (%)		Le4 / L2 (%)		All levels / L2 (%)	
	Int.	FP.	Int.	FP.	Int.	FP.	Int.	FP.
LN2-72KB	58.7	40.9	—	—	—	—	58.7	40.9
LN3-144KB	59.9	41.0	21.2	29.4	—	—	81.2	70.3
LN4-248KB	60.1	41.0	21.1	27.1	7.4	19.5	88.6	87.7
Average	59.6	41.0	21.2	28.3	7.4	19.5		

Avg / Min Transport Latency	
Int.	FP.
1.014	1.009
1.008	1.005
1.005	1.004



(a) IPC harmonic mean.



(b) Total Energy.

Fig. 4. Average IPC and Total Energy for conventional and L-NUCA configurations. RT stands for r-tile and RESTT for rest of tiles. Network energy consumption is included for the L-NUCA configurations.

transport latency (delay between a tile experiences a hit until the Issue Window is notified about the data availability) and the minimum transport latency (no contention) for all tested benchmarks. These values (less than 1.5% of increment) prove the effectiveness of the proposed topologies and the distributed dynamic routing.

Regarding energy, cache consumption is dominated by static energy in current technologies [17]. Therefore, L-NUCA performance improvements also entail energy savings because the savings in static consumption overpass the small increment in dynamic energy caused by inter-tile block migrations and search broadcasts.

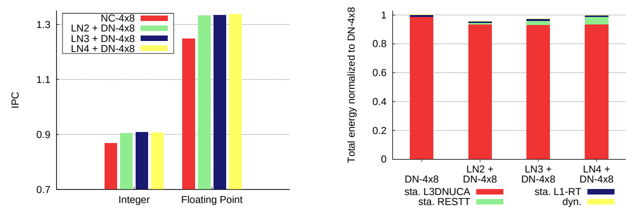
Fig. 4(b) shows the average energy consumption for all benchmarks normalized to the *L2-256KB* configuration. As expected, L3 static energy stands out above the rest, and the execution time reduction coming from L-NUCA saves roughly 10% of static L3 energy. In the overall picture, all L-NUCA configurations beat the baseline with savings ranging from 10.5% (*LN4-248KB*) to 16.5% (*LN2-72KB*).

Putting together area, performance, and energy results, the replacement of a 256KB L2 by an 128KB L-NUCA saves a 5.3% in area, improves IPC by 6.1% and 15% for Integer and Floating Point benchmarks, respectively, and reduces energy by 14.24%.

B. Integrating L-NUCAs with D-NUCAs

The 8MB D-NUCA baseline (*DN-4x8*) was selected after exploring its design space varying, among others, bank size, link width, number of sparse sets, number of rows, associativity, insertion policy, and injection bandwidth. *DN-4x8* includes 32 tiles as described in Table I.

Regarding performance, Fig. 5(a) compares Integer and Floating Point IPC of all configurations. The combination of L-NUCA + D-NUCA always improves performance, being the achieved gains almost flat across the tested L-NUCAs, roughly 4.5% and 7% for Integer and Floating Point benchmarks, respectively. Since D-NUCA organizations reduce the latency gap when compared to conventional hierarchies, the number of L-NUCA levels required to maximize performance decreases; therefore, two levels are enough, and with a slight 1.2% area increment, *LN2* + *DN-4x8* improves *DN-4x8* by 4.2% and 6.8% for Integer and Floating Point, respectively. Besides, gains are consistent across benchmarks and in 60% of them IPC improves by more than 10%.



(a) IPC harmonic mean. (b) Total Energy.

Fig. 5. Average IPC and Total Energy for L1 + D-NUCA and L-NUCA + D-NUCA configurations.

Regarding energy, Fig. 5(b) shows the total energy consumption relative to *DN-4x8*. Once again, the execution time reduction helps to decrease the energy consumption from 4.25% (*LN2 + DN-4x8*) to 0.2% (*LN4 + DN-4x8*). Interestingly, *LN2 + DN-4x8* saves 19.8% of dynamic energy with regards to *DN-4x8* because the added activity in the L-NUCA (8KB banks, simple networking) requires less energy than the dynamic activity removed in the D-NUCA (256KB banks, virtual channel routing).

Summarizing, if we can afford a negligible 1.2% area increment, adding an LN2-72KB to a D-NUCA hierarchy improves IPC by 4.2% and 6.8% for integer and floating point benchmarks, respectively and saves 4.25% in total energy consumption.

VI. Related Work

In addition to the related work presented in Section I, multiple authors have studied the on-die cache latency gap. For example, Beckmann and Wood proposed the use of transmission lines to reduce the wire delay [18]. Chishti *et al.* proposed NuRAPID decoupling the placement between tag and data [19]. Regarding NUCA improvements, Jin *et al.* proposed a novel router, a replacement algorithm, and a heterogeneous halo topology [7]. Muralimanohar and Balasubramonian introduced heterogeneity in the wires and in the topology with a mixed point-to-point bus network [3]. All these works focus on multi-megabyte caches, while L-NUCAs focus on size-reduced caches.

VII. Conclusions

The inclusion of large secondary on-chip caches for closing the latency gap between the processor and main-memory causes another latency gap between those large caches and the fast and small first level caches. This work tackles this problem by extending the cache size reachable by the processor at very low latencies. This objective is achieved by adapting NUCA caches to size-reduced caches.

One of the main novelties of L-NUCAs is its interconnection system. Three different networks have been conceived for the basic cache operations: search, transport, and replacement. All of them are based on short and scalable local links. Besides, it supports fast lookup and block delivery in a fully-associate structure maximizing hit ratios. Routing is implicit in all the networks minimizing cost and increasing message delivery. With this interconnection fabric, L-NUCA performs in parallel a cache access and one-hop routing in a single cycle.

Our detailed simulations show that, in general, L-NUCA improves simultaneously performance, energy, and area when integrated into both conventional or D-NUCA hierarchies.

Acknowledgment

The authors would like to thank Brian Greskamp, the members of the gaZ research group, and the anonymous reviewers for their suggestions on this paper. This work was supported in part by the Gobierno de Aragón grant "gaZ: Grupo Consolidado de Investigación", the Spanish Ministry of Education and Science grant TIN2007-66423, and the European Union Network of Excellence HiPEAC-2 (High-Performance Embedded Architectures and Compilers, FP7/ICT 217068).

References

[1] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *ASPLOS-X*, 2002.
[2] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, "Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures," in *MICRO*, 2000.
[3] N. Muralimanohar and R. Balasubramonian, "Interconnect design considerations for large NUCA caches," in *ISCA*, 2007.
[4] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Vergheseemph, "Piranha: a scalable architecture based on single-chip multiprocessing," in *ISCA*, 2000.
[5] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.
[6] D. W. Plass and Y. H. Chan, "IBM POWER6 SRAM arrays," *IBM J. of Research and Development*, vol. 51, no. 6, pp. 747–756, 2007.
[7] Y. Jin, E. J. Kim, and K. H. Yum, "A domain-specific on-chip network design for large scale cache systems," in *HPCA*, 2007.
[8] S. Thoziyoor, N. Muralimanohar, and N. P. Jouppi, "Cacti 5.0," HP Labs, Tech. Rep. HPL-2007-167, October 2007.
[9] A. Kumar, P. Kundu, A. P. Singh, L.-S. Pehy, and N. K. Jha, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *ICCD*, 2007.
[10] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *HPCA*, 2001.
[11] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *MICRO*, 2003.
[12] S. Gochman, A. Mendelson, A. Naveh, and E. Rotem, "Introduction to Intel Core Duo processor architecture," *Intel Tech. J.*, vol. 10, 2006.
[13] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite," in *ISCA*, 2007.
[14] G. Hamerly, E. Perelman, J. Lau, and B. Calder, "SimPoint 3.0: Faster and more flexible program phase analysis," *J. of Instr.-Level Parallelism*, vol. 7, 2005.
[15] Intel Corporation, "Intel® Core^{TM}2 Duo Processor E8600, http://ark.intel.com/cpu.aspx?groupId=35605."
[16] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *MICRO*, 2002.
[17] J. Wuu, D. Weiss, C. Morganti, and M. Dreesen, "The Asynchronous 24 MB On-Chip Level 3 Cache for a Dual-Core Itanium Architecture Processor," in *ISSCC*, 2005.
[18] B. Beckmann and D. Wood, "TLC: Transmission line caches," in *MICRO*, 2003.
[19] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Distance associativity for high-performance energy-efficient non-uniform cache architectures," in *MICRO'03*, 2003.