View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositorio Universidad de Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Trabajo Fin de Máster
Máster en Ingeniería de Sistemas e Informática

Efficient instruction and data caching for high-performance low-power embedded systems

Alexandra Ferrerón Labari

Director: Darío Suárez Gracia

Ponente: Jesús Alastruey Benedé

Grupo de Arquitectura de Computadores (GAZ)
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Curso 2011/2012
Septiembre 2012

Abstract

Although multi-threading processors can increase the performance of embedded systems with a minimum overhead, fetching instructions from multiple threads each cycle also increases the pressure on the instruction cache, potentially harming the performance/consumption ratio. Instruction caches are responsible of a high percentage of the total energy consumption of the chip, which for battery-powered embedded devices becomes a critical issue.

A direct way to reduce the energy consumption of the first level instruction cache is to decrease its size and associativity. However, demanding applications, and specially applications with several threads running together, might suffer a dramatic performance slow down, or even increase the total energy consumption of the cache hierarchy, due to the extra misses incurred.

In this work we introduce iLP-NUCA (*Instruction Light Power NUCA*), a new instruction cache that replaces the conventional second level cache (L2) and improves the Energy–Delay of the system. We provided iLP-NUCA with a new tree-based transport network-in-cache that reduces both the cache line service latency and the energy consumption, regarding the former LP-NUCA implementation.

We modeled in our cycle-accurate simulation environment both conventional instruction hierarchies and iLP-NUCAs. Our experiments show that, running SPEC CPU2006, iLP-NUCA, in comparison with a state–of–the–art high performance conventional cache hierarchy (three cache levels, dedicated L1 and L2, shared L3), performs better and consumes less energy.

Furthermore, iLP-NUCA reaches the performance, on average, of a conventional instruction cache hierarchy implementing a double sized L1, independently of the number of threads. This translates into a reduction of the Energy–Delay product of 21%, 18%, and 11%, reaching 90%, 95%, and 99% of the ideal performance for 1, 2, and 4 threads, respectively. These results are consistent for the considered applications distribution, and bigger gains are in the most demanding applications (applications with high instruction cache requirements). Besides, we increase the performance of applications with several threads without being detrimental for any of them. The new transport topology reduces the average service latency of cache lines by 8%, and the energy consumption of its components by 20%.

Resumen ejecutivo

Los procesadores multi-hilo pueden incrementar el rendimiento de sistemas empotrados con una mínima sobrecarga en área y energía. Buscar instrucciones de varios hilos cada ciclo, no obstante, incrementa la presión en la cache de instrucciones, lo que potencialmente puede afectar el ratio rendimiento/energía. Las caches de instrucciones además son responsables de un porcentaje muy alto del total de la energía consumida por el chip, aspecto crítico para dispositivos alimentados por baterías.

Una forma directa de reducir el consumo energético del primer nivel de cache de instrucciones es reducir su tamaño y asociatividad. Sin embargo, aplicaciones con demandas mayores, y especialmente aplicaciones multiprogramadas con varios hilos, pueden sufrir una drástica reducción del rendimiento, o incluso un incremento de la energía total consumida por la jerarquía de memoria cache, debido a los fallos adicionales.

En este trabajo introducimos iLP-NUCA (*Instruction Light Power NUCA*), una nueva cache de instrucciones teselada que reemplaza la cache convencional de nivel 2 (L2) y mejora el *Energy–Delay* del sistema. Además proponemos una nueva red de transporte para iLP-NUCA con topología en forma de árbol que reduce tanto la latencia media de servicio, como la energía consumida por la red de transporte, en comparación con implementaciones previas de LP-NUCA.

Hemos implementando en un entorno de simulación que modela en detalle los parámetros de la microarquitectura, jerarquías convencionales de cache de instrucciones e iLP-NUCAs. Nuestros experimentos demuestran que, ejecutando SPEC CPU2006, iLP-NUCA, en comparación con una jerarquía típica convencional de alto rendimiento (tres niveles de cache, L1 y L2 separados, L3 compartido), obtiene mayor rendimiento y consume menos energía.

Aún más, iLP-NUCA consigue alcanzar el rendimiento, en media, de una jerarquía convencional reduciendo el tamaño del primer nivel de cache de instrucciones a la mitad, independientemente del número de hilos. Esto se traduce en una reducción del producto *Energy–Delay* del 21%, 18% y 11%, alcanzando el 90%, 95% y 99% del rendimiento de una cache de instrucciones perfecta, con 1, 2 y 4 hilos, respectivamente. Estos resultados son consistentes para la distribución de las aplicaciones consideradas y las mayores mejoras se dan en las aplicaciones que más estresan la jerarquía de instrucciones. Además en aplicaciones con varios hilos, el incremento en rendimiento se consigue sin perjudicar a ninguna de las aplicaciones. La nueva red de transporte reduce la latencia media de servicio de bloques un 8%, y reduce el consumo energético de los componentes de la red un 20%.

Contents

1 Introduction 1
  1.1 Scope of the Project . . . 2
  1.2 Objectives . . . 2
  1.3 Masters Thesis organization . . . 2
  1.4 Contributions . . . 3
  1.5 Acknowledgements . . . 3

2 Instruction cache hierarchies for embedded systems 5
  2.1 State–of–the–art cache hierarchies for embedded systems . . . 5
  2.2 Performance and energy on a state–of–the–art embedded processor . . . 6
  2.3 Our proposal: instruction Light Power – NUCA . . . 8
    2.3.1 Tree-based transport network for iLP-NUCA . . . 9

3 Methodology 11
  3.1 Processor baseline . . . 11
    3.1.1 Memory hierarchy overview . . . 11
  3.2 Simulator . . . 12
  3.3 Workloads . . . 13
  3.4 Metrics . . . 13

4 Results 15
  4.1 Impact of the tree-based transport network . . . 15
  4.2 First level instruction caches: performance/energy trade-offs . . . 16
    4.2.1 Energy consumption . . . 16
    4.2.2 Performance . . . 17
    4.2.3 Energy-Delay . . . 19
    4.2.4 Summary . . . 23
  4.3 Putting all together: instructions and data . . . 23
  4.4 L1-I access latency . . . 23
  4.5 Results summary . . . 24

5 Related Work 25

6 Conclusion and future work 29
  6.1 Publications . . . 30
  6.2 Future Work . . . 30

References 33

A Project management **39**

B Simulation environment and methodology **41**
B.1 SMTScalar 41
B.1.1 The microarchitectural model 42
B.2 Simulation methodology 42

C SPEC CPU2006 characterization: instruction cache requirements **45**
C.1 Cache size and associativity implications in performance 45
C.1.1 L1-I misses per k-instruction 45
C.1.2 Performance evaluation: IPC 48

D Shared iLP-NUCA designs **51**

E ILP-NUCA: Cache de Instrucciones Teselada para Procesadores Empotrados **53**

List of Tables

2.1 Cache hierarchy parameters of representative current high-end processor samples (28-45 nm). 6

3.1 Simulator micro-architectural parameters. 11

List of Figures

2.1 Dynamic energy consumption of the memory hierarchy for several L1 sizes/associativities for 1, 2, and 4 threads. 7
2.2 3-level LP-NUCA. 8
2.3 2-D Mesh transport network topology and its components. 9
2.4 Tree-based transport network topology and its components. 10

3.1 Baseline system modeled: cache hierarchy detail. 12

4.1 Dynamic energy consumption of the memory hierarchy for several L1 sizes/associativities for 1, 2, and 4 threads. 17
4.2 IPC throughput and fairness distribution for several L1 sizes/associativities for 1, 2, and 4 threads. 18
4.3 ED and ED^2 products for 1, 2, and 4 threads, several configurations. Data cache always hits. 19
4.4 ED distribution for 1 thread applications. 20
4.5 ED distribution for 2 threads applications. 21
4.6 ED distribution for 4 threads applications. 22
4.7 ED and ED^2 products for 1, 2, and 4 threads, several configurations. 24

A.1 Schedule and tasks of the Masters Thesis. 40

B.1 Organization overview of the baseline simulated processor with a conventional three level cache hierarchy. 41
B.2 Simulation methodology workflow. 43

C.1 L1-I mpki for several cache sizes and associativities, SPEC CPU2006. 46
C.1 L1-I mpki for several cache sizes and associativities, SPEC CPU2006. 47
C.2 IPC for several cache sizes and associativities, SPEC CPU2006. 48
C.2 IPC for several cache sizes and associativities, SPEC CPU2006. 49
C.2 IPC for several cache sizes and associativities, SPEC CPU2006. 50

D.1 Search network topology for shared iLP-NUCA. 52
D.2 Transport network topology for shared iLP-NUCA. 52
D.3 Replacement network topology for shared iLP-NUCA. 52

Chapter 1 | Introduction

In the last years, embedded systems have evolved so that they offer functionalities we could only find before in high performance systems. Portable devices, including smartphones, tablets, or multimedia players, already have multiprocessors on-chip (such as Qualcomm MSM 8960, so called Krait [17], or ARM Cortex A15 MPCore [53]). These systems usually have several processor cores, often multi-threaded [52], and a powerful multi-level on-chip memory hierarchy. This memory hierarchy has a first private level of cache separated for instructions and data, and a last shared level. As most of these systems are battery-powered, the power consumption becomes a critical issue.

Achieving conflicting goals such as high performance and low power consumption is a high complexity challenge, where some proposals have been already made [32, 6, 29, 37, 24, 27, 50]. Suárez *et al.* proposed in [47] a new on-chip cache hierarchy, the Light Power NUCA (LP-NUCA), which is able to reduce the access latency taking advantage of Non-Uniform Cache Architectures (NUCA) properties. NUCA tackles the wire delay problem [1] in last level caches (LLC) by merging the second and third level caches into a mesh of cache banks, and enabling inter-bank block migration. In this way, blocks located in the banks close to the cache controller have a lower latency than those located further apart [31]. LP-NUCAs have been proved to be efficient for data hierarchies, achieving good performance and reducing the energy consumption.

On the other hand, instruction caches have different characteristics and requirements than data caches, which do not comply with the low-power embedded systems requirements, especially in SMT[2] environments [52, 51]. Instruction caches are accessed every cycle by one or more threads and its instruction flow supply influences the performance of the whole processor. Thus, it is not a surprise that instruction caches account for a high percentage of the total energy consumption of the chip. For example ARM 920TTM dissipates 25% of the power in its instruction cache [45], and ARM Strong instruction cache is responsible of 27% of the total energy consumption of the chip [38].

In this Master's Thesis, we do a review on modern instruction caches for SMT embedded systems. We study the goodness of adapting LP-NUCAs to the instruction cache hierarchy (iLP-NUCA). We propose a new design that improves previous LP-NUCA designs by modifying one of its networks-in-cache.

With this structure we can reduce the energy consumption and improve performance over a conventional high performance hierarchy independently of the number of threads. Furthermore, we achieve the same performance with smaller cache sizes, and we are able to reduce the energy consumption by 21%, 18%, and 11%, and we reach 90%, 95% and 99% of the ideal performance for 1, 2, and 4 threads, respectively.

---

[1] Driving a signal from the cache controller to the banks takes more time than the bank access itself.

[2] SMT stands for Simultaneous Multi-Threading.

1.1 Scope of the Project

This project has been developed in the Computer Architecture Group (*Grupo de Arquitecturas*) of the University of Zaragoza. This work was supported in part by grants TIN2010-21291-C02-01 (Spanish Government and European ERDF), gaZ: T48 research group (Aragón Government and European ESF), Consolider CSD2007-00050 (Spanish Government), and HiPEAC-3 NoE.

Since September 2011 I have a scholarship *Formación de Personal Investigador* (FPI) from *Ministerio de Innovación y Ciencia* (Spanish Government).

The results collected in this Master's Thesis correspond to the work developed since January 2012 to September 2012.

1.2 Objectives

The main objectives of this Master's Thesis are:

- Study the state-of-the-art in the design of instruction and data caches in SMT–CMP[3] environments for embedded superscalar processors.
- Extend the home made simulation environment *SMTScalar* to add instruction cache hierarchies.
- Implement iLP-NUCAs to efficient manage instructions and data in the simulation platform, and evaluate the new design by simulating representative workloads such as SPEC CPU2000/2006 [20, 21] .
- Study the possibilities of increasing performance and energy efficiency by adding content allocation polices based on hardware and/or software.

After an extensive review of state-of-the-art instruction caches for SMT embedded systems, we extended our simulation environment to support instruction cache hierarchies and iLP-NUCAs. We proved their functionality by the execution of the benchmark suite SPEC CPU2006.

Adapting this structure for CMP environments, as well as applying software based techniques for energy/performance efficiency, is right now ongoing work.

1.3 Master's Thesis organization

The remaining of this Master's Thesis is organized as follows: Chapter 2 explores iLP-NUCAs, our proposal to improve instruction cache hierarchies efficiency; in Chapter 3 the methodology we followed is explained; Chapter 4 collects the main results; Chapter 5 presents the related work and Chapter 6 concludes.

Also found as appendix:

A. Project management. It includes schedules and effort.

B. Simulation environment and methodology. This appendix details our simulation environment, *SMTScalar*, our simulated architecture, and our scientific workflow.

C. SPEC CPU2006 characterization: instruction cache requirements. This appendix presents a characterization of the instruction cache requirements of SPEC CPU2006 benchmark suite.

---

[3] CMP stands for Chip Multi-Processors.

D. Shared iLP-NUCA designs. This appendix presents some ongoing work about iLP-NUCAs shared by instructions and data.

E. iLP-NUCA: Cache de Instrucciones Teselada para Procesadores Empotrados. This appendix contains the paper that will be presented in the XXIII Jornadas de Paralelismo (JJPAR'12), Elche, September 2012.

1.4 Contributions

Summing up, the main contributions of this work are the following:

- We apply a new NUCA organization to the instructions hierarchy and evaluate it in a SMT environment. To the best of our knowledge, this is the first work that proposes a specific NUCA cache for the instructions hierarchy in single thread and SMT environments.
- We extended the simulation environment adding instruction cache hierarchies and iLP-NUCA caches. We followed a simulation methodology that automatizes the simulation process and allows to share resources and tools between the users of this and other simulation tools.
- We adapted LP-NUCAs for instructions hierarchies. We propose a new transport network-in-cache for LP-NUCA based on a tree topology that improves the previous designs. With this new topology we are able to reduce the effective service latency of blocks, we decrease the number of links (and therefore the nodes degree), and we reduce the energy consumed by the network.
- We observed that instruction caches are responsible of a high percentage of the energy consumption of the chip. Our experiments show that instruction caches are normally over–dimensioned for many representative workloads. We proposed to reduce the first level instruction cache size and associativity to save energy and keep an acceptable performance by backing–up the first level cache with an efficient structure, iLP-NUCA. We achieve the performance of a conventional hierarchy and reduce the energy consumption with smaller caches independently of the number of threads. Furthermore, the available area that iLP-NUCA provides by reducing the first level instruction cache offers space on-chip for the implementation of other components such as accelerators.
- The realization of this Master's Thesis delivered a poster presentation in an international conference, a paper submission and acceptance in a national conference, and the future submission of a paper to an international conference with the main results of this work.

1.5 Acknowledgements

First of all, I would like to thank to all the members of the Computer Architecture Group. Thanks to Chus for his revisions and support. Jorge, one day we will flatten the motivation curve. Specially, thanks to Darío, the most encouraging and positive person I have around, and the best director I could have.

For all the friends that support me every day. There are many coffees and many beers waiting for us. To Markus, who starts to be an expert in caches after hearing me day after day. Thanks for all your support and help.

This project is dedicated to my family, especially to my parents and my grandma.

Chapter 2 | Instruction cache hierarchies for embedded systems

This chapter presents the design space for instruction cache hierarchies and introduces our proposal (iLP-NUCA) to simultaneously improve performance and save energy in single thread and SMT embedded systems.

Superscalar execution cores demand a continuous instruction supply to feed their functional units [26, 46]. Instruction flows might be interrupted by dependencies between instructions, breaks in the control flow (due to branches), and wait states generated by cache misses. Delays due to the instruction cache hierarchy affect the instruction flow speed and the instruction issue, and hence, performance.

Simultaneous multi-threading (SMT) is a well-known technique to hide long latency operations by the execution of several threads [52]. SMT aims to have all the functional units highly utilized by using a more powerful front-end (fetch unit) supplying instructions from several threads. Although SMT can be implemented with a minimum energy and area overhead, it adds pressure to the front-end, and instruction caches become a critical component in the system design.

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

In the following sections we will present current design choices for instruction caches in commercial systems. We will evaluate in terms of performance and energy consumption the behavior of potential applications using SPEC CPU2006 [21] for a state–of–the–art processor. Finally, we will present our proposal, iLP-NUCA, and its enhancements over previous designs.

2.1 State–of–the–art cache hierarchies for embedded systems

Embedded systems cover a wide range of applications and domains. Table 2.1 collects information about representative commercial system-on-chips (SoCs) from several vendors. As we can see, instruction cache sizes vary between 16KB (Qualcomm MSM 8960 [17]) and 64 KB (Netlogics XLP864 [18]), while most of the caches are 32KB. Associativities of this processors usually vary between 2 and 8, being 4 a commonly adopted solution [53, 35, 3, 7].

Many of the embedded systems we can find nowadays such as smartphones, tablets, or multimedia players, are portable devices. These devices are usually powered by batteries, so energy consumption becomes more important, turning into a critical issue. Instruction caches are

Table 2.1: Cache hierarchy parameters of representative current high-end processor samples (28-45 nm).

Model, *Brand*	#threads/CPU	L1/CPU (KB) I	L1/CPU (KB) D	L2/CPU	L3 (MB)
QorIQ AMP T2080, *Freescale* [7]	2	32	32	512KB	512KB[a]
PowerPC 476FP, *IBM–LSI* [35]	1	32	32	512KB	2MB + 4 eDRAM
Cortex-A15 MPCore, *ARM* [53]	1	32	32	up to 4MB shared	no
XLP864, *NetLogic* [18]	4	64	32	512KB	16MB shared
MSM 8960, *Qualcomm* [17]	1	16[b]	16[b]	512KB	no
MIPS32 ProAptiv, *MIPS* [13]	>1	32-64[c]	32-64[c]	up to 1.3MB	no

[a] Platform cache.
[b] L0 cache 4KB.
[c] Optional 4KB–1MB scratchpad.

responsible of a high amount of energy consumption of the system. To cite some examples, the StrongARM SA-100 consumes the 27% of its total energy in the instruction cache [38], and ARM 920TTM dissipates 25% of the power in its instruction cache [45].

Some recent works claim that the applications executed in this kind of devices have high requirements on instructions and therefore would benefit from a big first level instruction (L1-I) cache [10, 16]. Others maintain that requirements of instruction caches are small [32, 50, 41, 33]. SPEC CPU2006 is a benchmark suite highly utilized by academia and industry, composed by 29 applications, including artificial intelligence, compilation, compression, and speech recognition, among others [21]. This kind of applications are commonly employed in the embedded domain. We consider that SPEC CPU2006 covers a wide range of applications of different disciplines and therefore can represent the behavior of this domain applications.

2.2 Performance and energy consumption evaluation on a state–of–the–art embedded processor

We run SPEC CPU2006 benchmarks with a typical state-of-the-art processor configuration including a 32KB 4–way set associative first level instruction cache, dedicated L2 cache (i.e., separated data and instruction L2 caches), and a shared L3 cache. We assume 32nm technology and run experiments with 1, 2, and 4 threads. For more details please refer to Chapter 3, Methodology. Our experiments show that only a small subset of applications from the suite take advantage of such a big instruction cache. In concrete, from the 28 benchmarks considered[1], half of them keep performance reducing the cache size and associativity by 2 (i.e., 16KB 2–way set associative cache). Nine of them even keep the same performance with a 4KB 2–way set associative cache.

However, those applications from the set that have a big instruction footprint experiment a huge slow-down. For example, 11 of the benchmarks slow down more than 20% when reducing the cache to 4KB 2–way.

When running 2 threads, 15% of the mixes keep performance when the cache is 4KB 2–way, and 30% does it when 16KB 2–way. Again the slow down of applications with high requirements is dramatic.

A more detailed characterization in terms of instruction requirements of SPEC CPU2006 can be found in Appendix C.

Although many applications fit in a small instruction cache, the performance loss of applications with high requirements is not acceptable to satisfy the user experience.

---

[1] 483.xalancbmk could not be executed in our simulation environment.

Instruction cache size and associativity have also a high impact on the energy consumption of the structure. Figure 2.1 shows the energy consumption of the memory hierarchy for 1, 2, and 4 threads. For the sake of clarity, we show they dynamic energy consumption of each structure, and group the total static energy consumed. The latter value is small in comparison with the total energy as we consider LSTP (Low STandby Power) technology. We keep all the parameters the same and vary the first level instruction cache size and associativity. We consider four configurations: CV–32KB–4ass (32KB, 4–way associative), CV–16KB–2ass, CV–8KB–2ass, and CV–4KB–2ass. Each bar corresponds to one configuration, and represents the sum of the total energy consumed by all the applications considered.

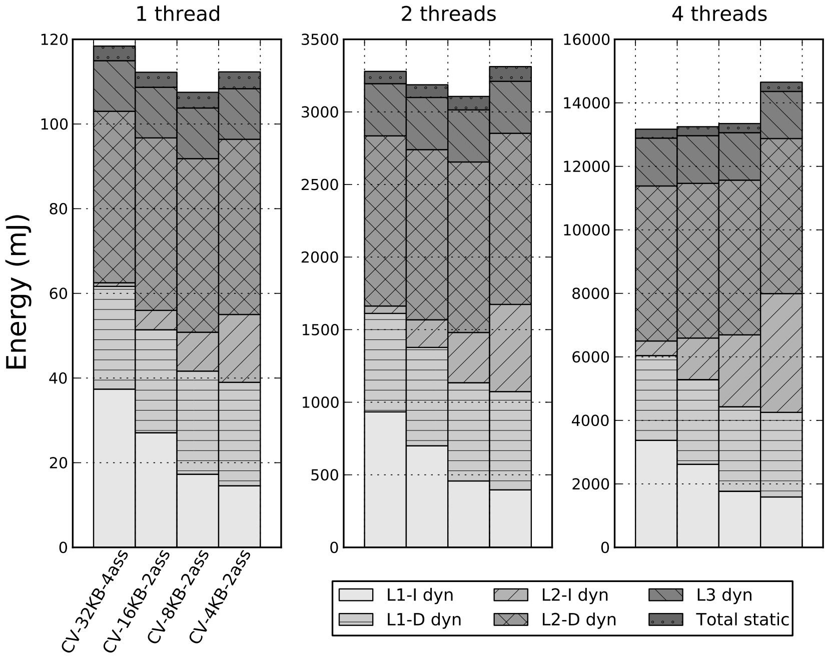


Figure 2.1: Dynamic energy consumption of the memory hierarchy for several L1 sizes/associativities for 1, 2, and 4 threads. Each bar corresponds to a configuration. Names of configurations include the size of the cache and the associativity. For example: CV–32KB-4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache.

If we focus on one thread, we can draw several conclusions. As we said before, many of the applications have a working set that fits within a 32KB 4–way set associative L1 instruction cache. This implies that all the requests hit in the first level (L1-I) and there is no need to fetch instructions from the next one (L2-I). The instruction cache consumes 32.5% of the memory hierarchy energy. The dynamic energy consumption of the L2-I is negligible (less than 1% of the total). As the L1-I cache shrinks, the dynamic energy per access also decreases (for example, with 4KB the percentage of energy consumption due to the L1-I is 13.5%). Per contra, the amount of blocks that the cache can hold is smaller, so the probability of a miss is higher. If we continue shrinking the L1-I cache there is a point where this reduction does not pay off. We can see that when the cache size is less than 8KB, the reduction on the energy consumption of the L1-I is smaller than the increase of the energy consumption of the L2-I, which at the end increases the total energy consumption of the system.

With two threads the scenario is similar, although the reductions are smaller as the cache requirements are bigger. If we consider four threads we can see how just reducing the cache to 16KB implies that the total energy consumption increases.

Potentially, reducing the first level instruction cache, as MSM 8960 does [17], is going to benefit the system from the energy point of view. But reducing the size of the L1-I will affect also its hit rate, which heavily affects performance. Yet the energy consumption of the L2-I cache will increase, counteracting the other savings.

Summing up, we would like to reduce the L1-I size and associativity to reduce the energy consumption of the system but without increasing the energy consumption of the L2-I, and losing as less performance as possible.

With this idea in mind, recent works proposed to dynamically reconfigure the L1 size and associativity so only part of the ways or part of the sets are used in a given moment [50, 8, 15]. With such approach the cache adapts to the behavior of the application and can operate with the optimal size. Other works proposed to add a small structure before or next to the first level cache to capture part of the accesses to the L1-I and increase the fetch speed, reduce the energy consumption, or both [32, 29]. A different approach is to rely on the compiler to map cache lines in a given structure or a given way/set of the cache [6, 27].

All these techniques need to either modify the interface between the L1-I and the processor, or add additional hardware or software support.

We propose a different approach. We will not modify the fetch unit or the L1-I structure, but we will optimistically reduce the size and associativity of the first level instruction cache. We will substitute the L2 cache by an improved and energy efficient structure, iLP-NUCA (*instruction Light Power NUCA*), based on the LP-NUCA design, which will minimize the performance degradation without increasing the energy consumption of the system. In the following section we will explain this structure in more detail.

2.3 Our proposal: instruction Light Power – NUCA

LP-NUCAs [47] extend the conventional first-level cache capacity by surrounding it with one or more levels of small cache tiles interconnected by specialized networks, as shown in Figure 2.2.

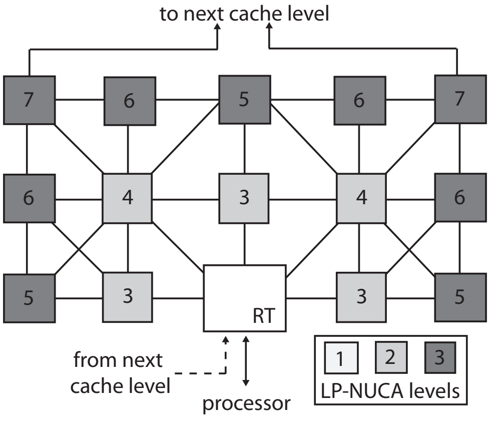


Figure 2.2: 3-level LP-NUCA. The numbers inside tiles represent the tile hit latency seen by the processor (assuming single-cycle tiles).

The first level of LP-NUCA (root-tile, RT) corresponds to a conventional L1 cache with additional control logic. We substitute the second level of the conventional hierarchy by a tiled structure whose effective access time increases gradually. The numbers inside tiles represent the minimum hit latency of a tile seen by the processor. We define the minimum hit latency of a tile as the time since we inject a request into the LP-NUCA until the block is actually served by that tile (i.e., received by the root-tile). The LP-NUCA design is characterized by the use of three specialized networks: search, transport, and replacement, which allow for a simple implementation and provide low latency and high bandwidth. Please refer to [47] for further details about the design and implementation of LP-NUCAs.

This structure has been proved to be efficient with data hierarchies, achieving high performance and reducing the energy consumption of the system.

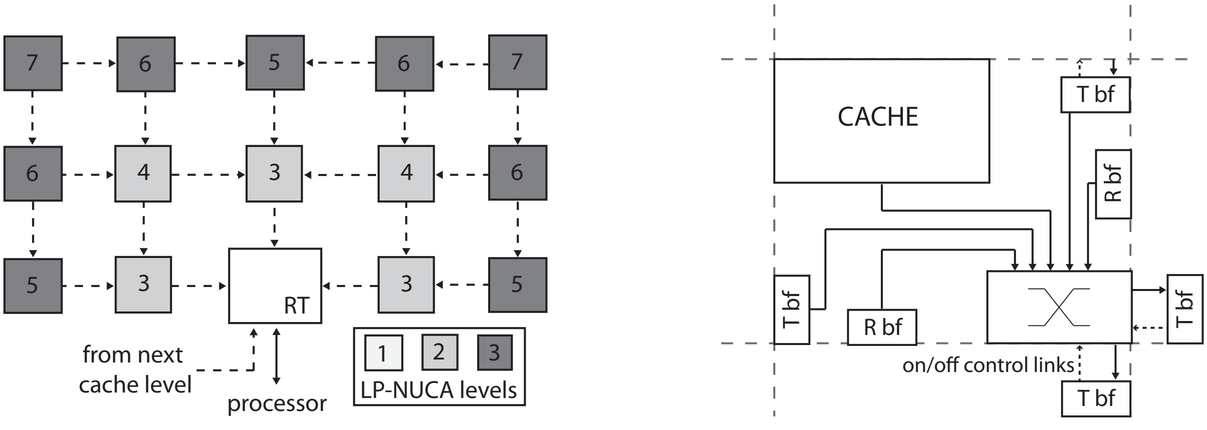
LP-NUCAs are good candidates for instruction hierarchies because they exploit the reuse (temporal locality), are energy-efficient structures, and provide a lot of bandwidth to the root-tile. Besides they do not change the interface between the L1 and the processor. We call this structure adapted to instruction hierarchies iLP-NUCA (instruction Light Power NUCA).

We expect that the use of iLP-NUCAs in the instruction hierarchy will allow for great reductions in energy consumption, at the same time that it will allow us to keep performance losses in an acceptable range.

In the following section we will describe the enhancements from the baseline design that we propose.

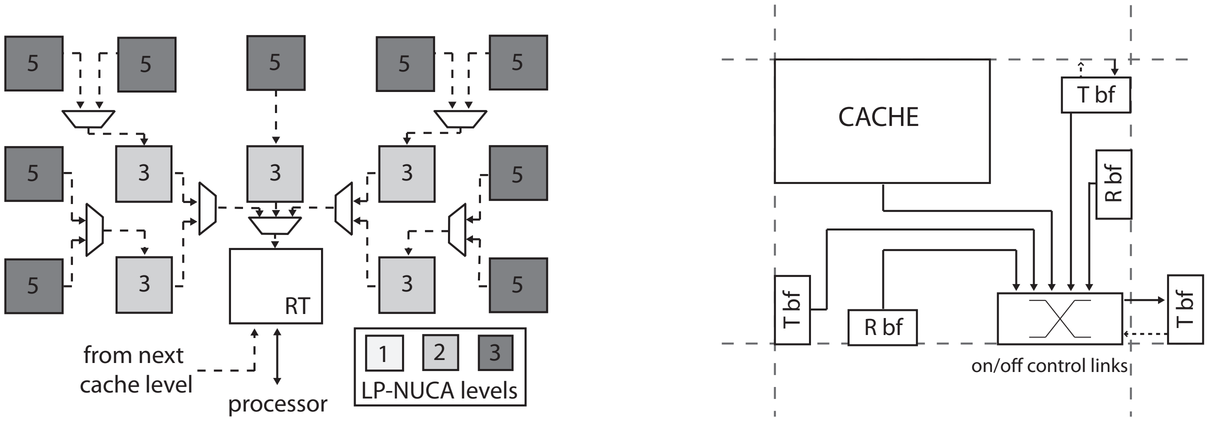
2.3.1 Tree-based transport network for iLP-NUCA

Previous designs of LP-NUCA utilized a 2-D mesh for the transport network because it provides high bandwidth with its multiple return paths to the root-tile. Figure 2.3 shows the 2-D mesh topology (Figure 2.3a) and the transport components inside a tile (Figure 2.3b). The numbers inside the tiles represent the tile latency assuming one-cycle tiles. This topology provides an increasing latency as we move further away from the root-tile. The main components of the transport network are the two transport buffers (Tbf) and the switch. In a given moment only three of the five inputs can be active in the crossbar (the contents among tiles are in exclusion, and therefore hits cannot happen simultaneously in the replacement buffers, Rbf in the Figure 2.3b, and the cache), which simplifies its design. We rely on buffered flow control, and use store-and-forward flow control with on/off back-pressure and two-entry buffers per link.



(a) 2-D Mesh transport network. **(b)** Transport components.

Figure 2.3: 2-D Mesh transport network and its components. The numbers inside the tiles represent the tile latency assuming one-cycle tiles. Tile latency includes search, tile access, and transport delay. The main components shown in Figure 2.3b are the transport buffers (Tbf), the replacement buffers (Rbf), the cache, and the switch.



(a) Tree-based transport network.

(b) Transport components.

Figure 2.4: Tree-based transport network and its components. The numbers inside the tiles represent the tile latency assuming one-cycle tiles. Tile latency includes search, tile access, and transport delay. The main components shown in Figure 2.4b are the transport buffers (Tbf), the replacement buffers (Rbf), the cache, and the switch. In comparison with the former 2-D Mesh, the number of output links has decreased and so does the complexity of the switch.

We already observed that the occupancy of the transport network links is very low [48]. Thus, we could decrease the path diversity in order to reduce the nodes degree. Figure 2.4 shows an alternative transport network based on a tree topology, and the transport components inside a tile (Figures 2.4a and 2.4b, respectively). With this topology the benefit is twofold. On one hand we decrease the links between tiles: now each tile only has one transport output link (before they had two), simplifying the crossbar design. On the other hand, we decrease the tile hit latency, especially for the tiles of higher levels, as we need less hops to reach the RT (which also means that we will consume less energy to deliver blocks to the root tile). Thus, we expect to reduce the effective service latency of blocks in comparison with previous LP-NUCA designs, and the energy consumption of the transport network.

The novel topology slightly increases the length of some wires, but it does not affect the tile cycle time because both search and replacement networks have less slack. It also increases the degree of the root-tile crossbar from 3 to 5 inputs without performance impact. To start the execution of a refill instruction, we only need to notify the MSHR (miss status holding register) which entry is going to be written. The MSHR will notify then the Issue Window that launches the refill instruction, and that is the time when the data will be necessary. We can take advantage of these cycles to pass through the root-tile crossbar.

From now on, we will assume that iLP-NUCA refers to the structure with this novel transport network.

Next chapters will show the detailed evaluation of a processor where we substitute the conventional L2 for a iLP-NUCA, and how it behaves in comparison with a conventional hierarchy.

Chapter 3 | Methodology

This chapter presents the methodology followed in this project. We briefly present the modeled processor, the simulation methodology and environment, the benchmarks utilized, and the metrics used to evaluate them. For more information please refer to Appendix B.

3.1 Processor baseline

For our experiments, our simulator models the system summarized in table 3.1. We based our model in typical state-of-the-art high performance embedded systems such as IBM/LSI PowerPC 476FP, NetLogic XLP864, and Freescale QorIQ AMP T2080 [35, 18, 7]. Our system has a more powerful memory hierarchy, and it is able to execute up to 4 threads simultaneously.

Table 3.1: Simulator micro-architectural parameters. BS, AM, lat, and init stand for block size, access mode, latency, and initiation rate, respectively

Clock Frequency	1 GHz	Threads	1/2/4
Fetch/Decode/Commit width	4	Issue width	4(IN+ME) + 2FP
INT/FP/MEM IW entries	24/16/ 16	ROB/LSQ entries	64/32
Branch Predictor	bimodal + gshare, 16 bit	Miss. branch penalty	6
L1/L2/L3 MSHR entries	8 / 8 / 4	MSHR secon. misses	4
TLB miss latency	30	Store Buffer/L2/L3 WB size[a]	32/16/16

Baseline L1/root-tile[b]	32KB–4Way–32B BS, write-through, 2-cycle lat, 1-cycle init
L2[c]	512KB–8Way–32B BS, serial AM, 4-cycle lat, 2-cycle init, copy-back
iLP-NUCA rest of tiles	32KB–2Way–32B BS, parallel AM, copy-back, levels: 3, total size: 448KB

L3	4MB eDRAM–16Way–128B BS, 14-cycle lat, 7-cycle init, copy-back
Main Memory	100 cycles/4 cycle inter chunk, 16 Byte bus

[a] L2, iLP-NUCA, and L3 Write Buffers coalesce entries.

[b] In root-tile, copy-back and write-around.

[c] L2 dedicated for data and instructions.

We estimate conventional cache access latencies and energy consumption assuming 32nm LSTP (Low STandby Power) technology with Cacti 6.5 [40]. Cacti is an integrated cache access time, cycle time, area, leakage, and dynamic power model. By integrating all these models together, cache trade-offs are all based on the same assumptions and, hence, are mutually consistent. Besides, the latencies provided by Cacti are typically lower than the ones achieved in commercial products, thus our assumptions are conservative. For iLP-NUCA latencies and energy estimations we use both CACTI 6.5 and a scaled 90nm layout [47].

3.1.1 Memory hierarchy overview

Figure 3.1 shows our baseline, a conventional three level cache hierarchy with dedicated first and second level caches, and with a third level shared between data and instructions. First level caches are true multiported (2 read/write ports). Second and third level caches are multi-banked, and have one read/write port. All caches use LRU (least recently used) but LP-NUCA, that employs LRF (least recently filled).

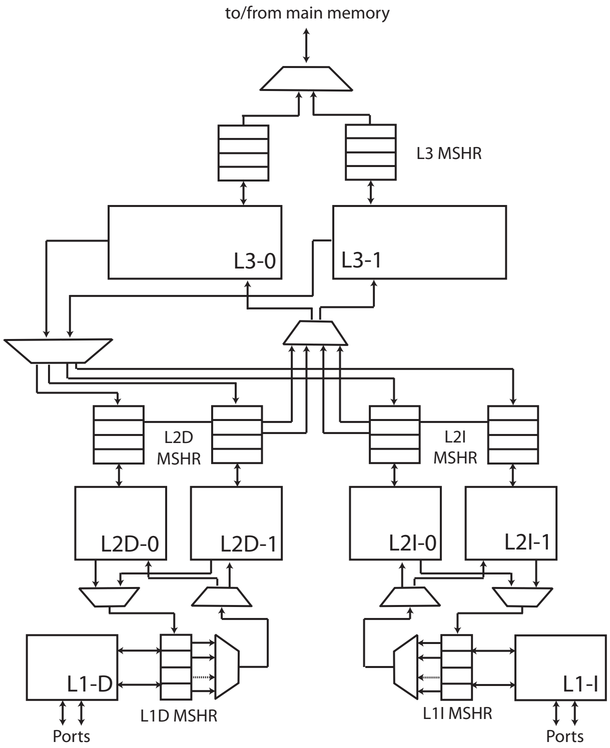


Figure 3.1: Baseline system modeled: cache hierarchy detail. For the sake of brevity, only the loads path is shown, omitting structures for stores.

L1 (baseline) and L2 caches are based on state-of-the-art commercial embedded systems (Table 2.1). For the shrunk L1-I configurations we decided to keep associativity in 2 as a compromise between energy consumption and performance in SMT applications [22].

We chose to model a separated L2 design for several reasons. First, we want to examine the behavior of the instructions and therefore we do not want data to interfere on it. In some of our experiments we will consider that the L1 data cache always hits. Second, dedicated caches offer higher performance, although they occupy more area. Finally, the implementation of a shared iLP-NUCA is not straightforward, and it is currently ongoing work.

We assume for the L2 serial access mode (tag array is accessed before the data array). We modeled with CACTI 6.5 L2 caches with smaller size (128 KB), and the results show that the size reduction does not affect either the access time or initiation rates (4 and 2 cycles, respectively). Reducing the L2 size would potentially reduce its energy consumption. However smaller L2 cache sizes might suffer higher miss rates, increasing the amount of accesses to the L3, and affecting both the performance of the system, and the energy consumption.

3.2 Simulator

We use *SMTScalar*, a cycle-accurate execution-based simulator based on SimpleScalar 3.0d for Alpha ISA [4]. SimpleScalar was heavily extended to support detailed microarchitectural models, highly configurable memory hierarchies, and simultaneous multi-threading execution for previous LP-NUCA works [48, 47]. We extended *SMTScalar* to add instruction cache hierarchies. We provided a realistic instruction fetch unit, and we evaluated the optimal values for the structures involved by both an extensive review of literature and state–of–the–art commercial processors, and experimentation with our simulation environment. More details about our simulated architecture and methodology can be found in Appendix B.

3.3 Workloads

We use the full SPEC CPU2006 [21] benchmark suite, but 483.xalancbmk (which we could not execute in our environment). For each program we simulate 100M representative instructions that were selected following the SimPoints methodology [19]. One thread experiments warm up cache and branch predictor during 200M instructions. Multi-thread experiments are multiprogrammed and we utilize *last* as simulation ending policy; there is no structure warming up and we only obtain statistics for the first 100M of instructions of each thread. For 2 SMT experiments we run all the combinations of benchmarks. For 4 SMT experiments, we assure the representativity of the results by following a methodology based on statistical sampling that allows for 97% of confidence level and between 3% error [25, 54, 49].

3.4 Metrics

We use performance metrics for multiprogrammed workloads including misses per k-instructions (mpki), average memory access latency (AMAT), IPC throughput (i.e., committed user instructions summed over all threads divided by total elapsed cycles[1]), and fairness [12]. IPC throughput and fairness are calculated according to the following formulas:

$$IPC\ througput = \sum_{i=1}^{n} IPC_i,\ fairness = \frac{\underset{i}{min}\left(\frac{CPI_i^{MP}}{CPI_i^{SP}}\right)}{\underset{i}{max}\left(\frac{CPI_i^{MP}}{CPI_i^{SP}}\right)}$$

where CPI refers to *instructions per cycle*, and MP and SP refers to single-thread and multi-thread execution, respectively.

Other metrics such as weighted speed up, ANTT (average normalized turnaround time), or STP (system throughput), were not utilized because we consider different baselines [9].

Regarding energy consumption, we report the total energy consumed as well as the Energy–Delay and Energy–Delay^2 products. We follow the Li *et al.* approach for SMT environments [34], and account for all the energy consumed until the last thread commits 100M instructions.

---

[1] The use of IPC throughput is save in our environment because we have no synchronization instructions.

Chapter 4 | **Results**

This chapter presents the Master's Thesis main results.

In previous chapters we have seen how from the energy point of view, we would like to have a small instruction cache. A small cache means that each access to the structure will consume less energy. However, from the performance point of view, some applications are heavily affected by reducing the available cache size. Increasing the execution time has a collateral effect, which is the increase of the static energy of the system.

In this chapter we will evaluate the new transport network for iLP-NUCA. We will see in more detail how small instruction cache sizes affect the energy consumption and the performance of the system. We will compare the behavior of a conventional cache hierarchy with our structure, the iLP-NUCA.

We will examine the system from the energy and the performance perspective, using the total energy consumed, and the IPC throughput and fairness, respectively. We will also use other metrics like mpki (misses per k-instruction) and AMAT (average memory access time). To summarize we will also consider Energy–Delay (ED) and Energy–Delay^2 (ED^2) products.

As we want to see how the system is influenced by instructions, we model a data cache that always hits. In this way data do not interfere in the system, and we can be sure that the performance and energy variations are due to the instruction caches. At the end of the chapter we will present the results modeling real data caches to have an overview of the behavior of the global system. Note that improving the private instruction caches reduce the pressure over shared last level ones, improving at the same time those applications that are data cache sensitive.

During this chapter, we will adopt the following name convention:

- CV refers to conventional cache hierarchies, and iLP to iLP-NUCA hierarchies.
- Configurations names, for the sake of brevity, have been shorten up. Each name is composed of three fields: category (CV/iLP), size of the first level instruction cache, and associativity of the above. For example: CV–16KB–2ass represents a conventional hierarchy where the L1-I size is 16KB and its associativity is 2.

We consider L1-I cache sizes ranging from 32KB 4–way set associative (so called baseline) till 4KB 2–way set associative. In any case we keep 2–way set associative caches because SMT takes advantage of the associativity [22], and it is a compromise between performance and energy consumption.

4.1 Impact of the tree-based transport network

The principal enhancement of iLP-NUCA from previous LP-NUCA designs is the new tree–based transport network. As we mention before, the main advantage of this network regarding the former 2-D mesh is the link count reduction per node, and the consequent degree reduction. The main transport components (see Figure 2.4) are the two input transport buffers and the switch. From the hardware implementation of LP-NUCA [47] we know that each buffer accounts for one third of the energy consumption, and so does the switch. Our new network will simplify the switch as we just have one output link. From the original layout we estimate that this reduction in the transport components will reduce their energy consumption by 20%. Although the path diversity is lower, the observed amount of traffic in the transport network lets us think that it will not experiment contention. This latter hypothesis is validated by our experimental results.

We compare the former 2-D mesh with the tree–based transport network with 3–level LP-NUCA/iLP-NUCA; instructions and data root-tiles are 32KB, 4–way set associative. Our experimental results show that the tree-based transport network is able to reduce by 8% the average service latency. As the bigger latency reductions are in the tiles located on the iLP-NUCA third level, the effectiveness of the new network is completely connected to the amount of reused that this level is able to capture. For example, in 447.dealII, the iLP-NUCA third level (data) capture 16.5% of hits (respect to the total hits in the second and third level). Our new network topology with its faster paths reduces the AMAT by 1.25%, and that translates into 1% of IPC improvement. However, other benchmarks like 450.soplex or 401.bzip2, which also have a high percentage of hits in the iLP-NUCA third level, have a little improvement on IPC (0.30% and 0.15% respectively). These applications present a high percentage of stores.

Energy benefits are more obvious when executing several threads. With 2 threads the new transport network consumes 7% less energy than the former 2-D mesh for data, and 4.7% less for instructions. With 4 threads the energy savings on the transport network energy consumption reach 6% for instructions, and 7% for data.

4.2 First level instruction caches: performance/energy trade-offs

We consider the processor presented in Chapter 3, with a three level cache hierarchy, where L1 and L2 are dedicated (i.e., separated for instructions and data), and a L3 is shared by instructions and data. We substitute the L1/L2 instruction caches for a 3 level iLP-NUCA. During this section we will consider that the data cache always hits.

4.2.1 Energy consumption

Figure 4.1 shows the energy consumption of different configurations for 1, 2, and 4 threads. Each bar corresponds to one configuration, and represents the sum of the energy consumption of all the applications considered. We only plot the dynamic and static energy of the instruction hierarchy.

As we saw before, the working sets of most of the applications executed fit in a 32KB 4–way set associative L1-I cache (see Appendix C for more details). Thus, the dynamic energy consumed by the L2-I is minimal. If we focus on one thread and we observed the evolution for conventional hierarchies (CV), we can see how when we decrease the L1-I size to 16KB 2–way set associative there is a huge decrease in the energy consumption. Still some of the applications fit in the cache, but there is a clear increase in the dynamic energy due to the L2-I. With 8KB the same tendency might be observed, but with 4KB the decrease of the dynamic energy consumption of the L1-I does not pay off the increase in the dynamic energy consumption of the L2-I. Thus the total energy consumption with 4KB is almost the same than with 32KB. iLP-NUCA configurations follow the same tendency. For the same L1-I size iLP-NUCA always consumes, on average, less energy than the conventional approach.

With two threads we can extract similar conclusions, still some observations should be made. Fetching from two threads increases the pressure on the L1-I cache, and the aggregated instruction footprint is less likely to fit in a small L1-I cache. Therefore the contribution of the L2-I respect to

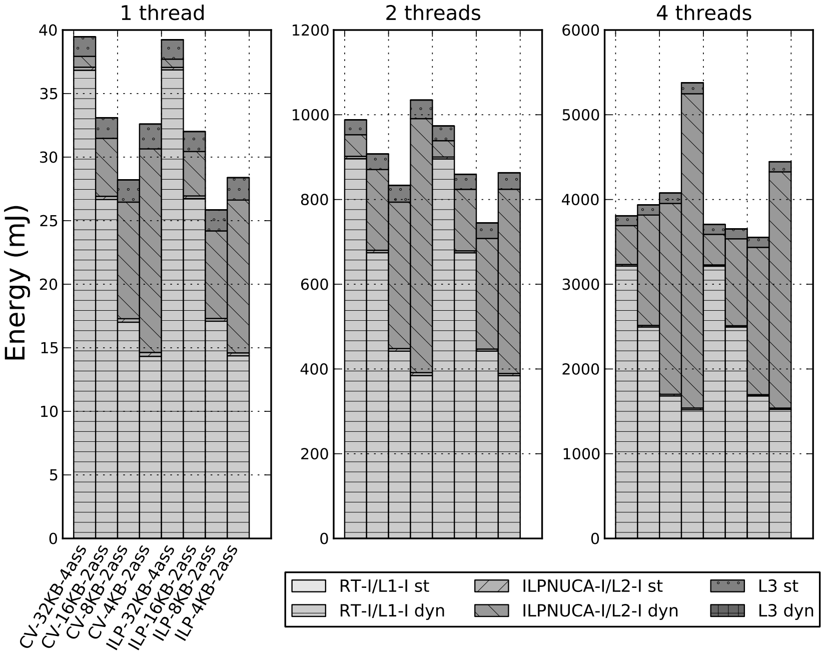


Figure 4.1: Dynamic energy consumption of the memory hierarchy for several L1 sizes/associativities for 1, 2, and 4 threads. Each bar corresponds to a configuration. Names of configurations include the size of the cache and the associativity. For example: CV–32KB–4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. Each bar represents the sum of the total energy consumed by all the applications considered. Data caches always hit.

the total energy consumption is higher. A L1-I cache of 4KB 2–way set associative in comparison with the 8KB 2–way set associative cache, decreases the energy consumption of the L1-I, on average, by 13%, but increases the energy consumption of the L2-I by 73.5%. iLP-NUCA energy consumption is increased by 66.5% in this case.

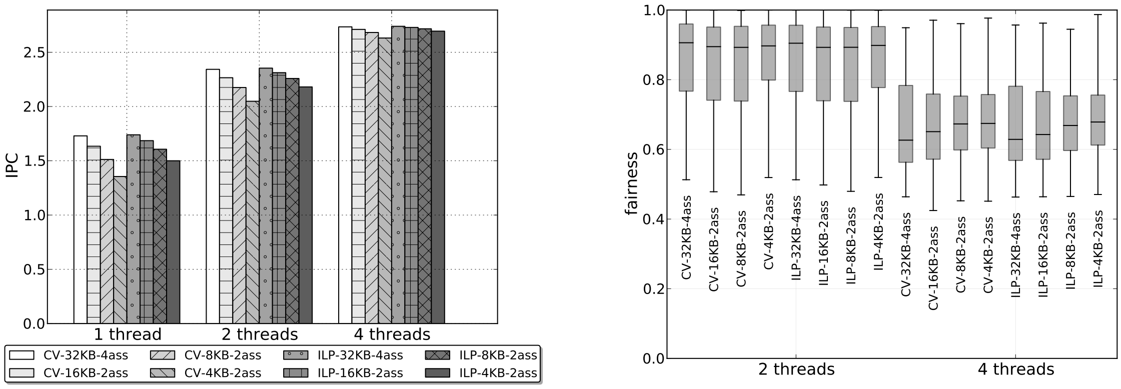
With four threads the added pressure gets too big and a conventional hierarchy cannot decrease the energy consumption. iLP-NUCA is able to decrease the energy consumption except for a 4KB 2–way set associative cache, where the dynamic energy of the iLP-NUCA experiments a huge increase.

All in all, for the same configuration of L1-I, iLP-NUCA always consumes less energy than the conventional hierarchy, independently of the number of threads. From the energy point of view, on average, a L1-I of 16KB or 8KB is preferable than 32KB. These values offer a L1-I cache capacity where a large part of the applications footprints fit, consuming less energy per access.

4.2.2 Performance

Figure 4.2 shows the average (harmonic) IPC throughput (Figure 4.2a) and fairness distribution (Figure 4.2b) of the system for different configurations. As in previous figures, each bar in Figure 4.2a corresponds to a configuration, following the name convention, and represents the average IPC throughput of the applications considered.

Although decreasing the L1-I provides large energy savings, it has a huge impact on IPC throughput. If we consider Figure 4.2a, and we focus on one thread, we see how when a conventional hierarchy shrinks the L1-I from 32KB to 16KB the average IPC throughput slows



(a) IPC throughput (average) for 1,2, and 4 threads, several configurations.

(b) Fairness distribution for 2 and 4 threads, several configurations.

Figure 4.2: IPC throughput and fairness distribution for several L1 sizes/associativities for 1, 2, and 4 threads. Each bar corresponds to a configuration. Names of configurations include the size of the cache and the associativity. For example: CV–32KB-4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. In Figure 4.2a each bar represents the average (harmonic) IPC throughput of all the applications considered. In Figure 4.2b the candlestick represents the minimum, the quartile 25, the median, the quartile 75, and the maximum of the distribution. In both cases data caches always hit.

down by 5.8%. Smaller cache sizes imply a slow down of 12.5% and 20% of performance for 8KB and 4KB respectively. Performance also decreases in a system implementing iLP-NUCA, yet this slow down is much smaller. With the same size than a conventional hierarchy, on average, iLP-NUCA performs better. If we compare with the conventional baseline (32KB–4ass), iLP-NUCA speeds up 0.61% with the same size, and slows down 2.8%, 7.6%, and 13% for sizes 16KB, 8KB, and 4 KB, respectively.

It the applications fit in the L1 cache, little difference in terms of performance can be observed. Remember that iLP-NUCA root-tile is equivalent to a conventional L1, and we do not change the interface between the L1 and the processor. When the applications do not fit in the L1 cache and start to stress the next level of the hierarchy (smaller cache sizes) is when we can take advantage of the iLP-NUCA structure. That is why the bigger differences in performance are observed for smaller L1-I cache sizes.

With two threads the slow down is soften by the ability of SMT to hide long latency operations with useful work from other threads. Nevertheless, the same tendency might be observed: the performance loss with conventional caches is bigger than that with iLP-NUCA, naming 3%, 6.3%, and 10.5% of slow down for CV–16KB–2ass, CV–8KB–2ass, and CV–4KB–2ass, respectively, against a speed up of 0.5% for iLP–32KB–4ass, and slow down of 1.16%, 3.2%, and 6% for iLP–16KB–2ass, iLP–8KB–2ass, and iLP–4KB–2ass, respectively.

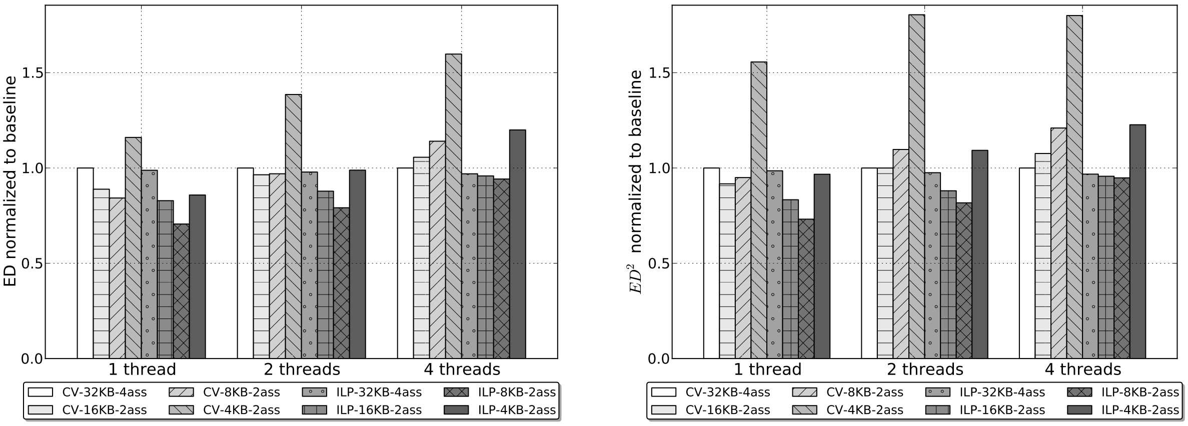
With four threads the difference is smaller. The occupancy of the functional units is higher and there are more threads to hide the stalls in the processor. iLP-NUCA keeps performance degradation in less than 1.5% while conventional caches lose 3.5% for 4KB.

In conclusion, iLP-NUCA achieves on average better performance than conventional data caches for the same L1-I configuration, regardless the number of threads. Even more, iLP-NUCA is able to achieve the same performance than a conventional hierarchy with half of the size (except for 16KB–2ass, where the performance degradation is 3% for 1 thread and less than 1% for 2 threads, if we compare iLP–16KB-2ass with CV–32KB–4ass).

Figure 4.2b shows the fairness distribution of the system. Each candlestick represents the minimum, the quartile 25, the median, the quartile 75 and the maximum of the distribution. The differences between cache organizations are very small and, in general, each iLP-NUCA configuration has a higher minimum. For 4 threads it might be surprising that smaller cache sizes present higher values of fairness. We calculate the fairness as the relative slow down of each application respect to the speed of that application running alone, with the same configuration, in the system. As applications perform, on average, worse as the cache size shrinks, the relative slow down is lower, and consequently the values of fairness higher. The conclusion we can extract from this graph is that iLP-NUCA increases performance without being detrimental to any program.

4.2.3 Energy-Delay

Figure 4.3 shows Energy-Delay and Energy-Delay^2 products normalized to the baseline (CV–32KB–4ass) for the different configurations considered, for 1, 2, and 4 threads. ED and ED^2 are the lower–the–better metrics.



(a) Energy–Delay product for 1, 2, and 4 threads, several configurations. Values are normalized to baseline (CV–32KB–4ass). Data cache always hits.

(b) Energy–Delay^2 product for 1, 2, and 4 threads, several configurations. Values are normalized to baseline (CV–32KB–4ass). Data cache always hits.

Figure 4.3: ED (Figure 4.3a) and ED^2 (Figure 4.3b) products for 1, 2, and 4 threads. Each bar corresponds to one configuration. Values are normalized to baseline (CV–32KB–4ass). Names of configurations include the size of the cache and the associativity. For example: CV–32KB–4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. Data cache always hits. ED and ED^2 are the lower–the–better metrics.

Bearing in mind the energy consumption and performance results we have seen, it seems clear that iLP-NUCA will present better results in the combined metric than a conventional hierarchy.

Both ED and ED^2 show a similar tendency. With one thread, ED decreases for both conventional and iLP-NUCA caches for 16KB and 8KB, although iLP-NUCA relative values are always smaller. With a 4KB L1-I, the conventional system presents a ED of 1.15 while iLP-NUCA keeps under the baseline (0.85).

With two threads we observe a similar behavior, yet the relative ED does not decrease that much. Again, iLP-NUCA relative values are smaller than conventional ones, and for the smallest cache size considered iLP-NUCA has a ED of 0.98.

With four threads the tendency changes. As we shrink the L1-I cache the relative ED of the conventional hierarchy increases. iLP-NUCA however presents a similar behavior for different cache sizes, but for 4KB, where the ED becomes 1.20.

Similar conclusions can be drawn for ED^2, yet the impact of the performance degradation (delay) affects more the conventional hierarchy.

In conclusion, in terms of ED and ED^2, iLP-NUCA on average performs better than a conventional hierarchy independently of the L1-I size and the number of threads.

Untill now we have shown the average values of ED and ED^2. To examine in more detail the behavior of our structure we will see the distribution of ED for all the applications considered. We want to see how iLP-NUCA performs against the conventional hierarchy. To do that, we take as baseline the best configuration of conventional caches: for 1 thread, 2 threads and 4 threads the best conventional configuration (on average) are CV–8KB–2ass, CV–16KB–2ass, and CV–32KB–4ass, respectively.

Figure 4.4 contains three graphs. The upper one presents the distribution of ED normalized to the 1 thread baseline (CV–8KB–2ass). The graph in the middle shows the absolute ED. The graph in the bottom of the figure shows the L1-I mpki (misses per k–instruction). We plot the baseline CV–8KB–2ass, and three of the iLP-NUCA configurations: iLP–16KB–2ass, iLP–8KB–2ass, and iLP–4KB–2ass. Benchmarks are sorted according to the L1-I mpki of the baseline.

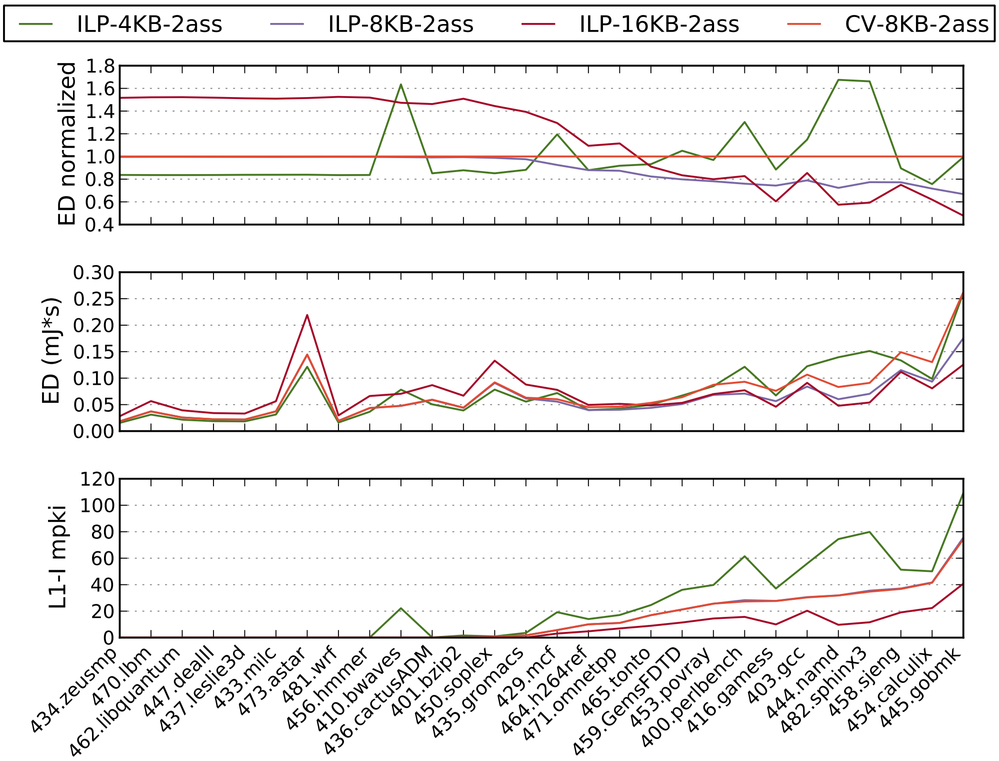


Figure 4.4: ED distribution for 1 thread applications. Names of configurations include the size of the cache and the associativity. For example: CV–32KB–4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. Data cache always hits. The upper graph represents ED values normalized to the best conventional configuration (baseline CV–8KB–2ass). The graph in the middle represents the absolute ED values. ED is a the lower–the–better metric. The lower graph represents the L1-I mpki. Benchmarks are sorted according to the L1-I mpki of the baseline (CV–8KB–2ass).

If we pay attention to the upper graph we can see how for the same L1-I cache size (8KB, 2–way set associative) iLP-NUCA always have the same or a smaller relative ED value (below 1). The performance (in terms of ED) of the iLP-NUCA is better when the mpki values of the applications are bigger. The lower graph sorts the SPEC CPU2006 benchmarks from small values to big mpki values for the baseline configuration. The first 11 applications on the left hand side of the graph fit in a 8KB cache (their mpki is close to 0). Therefore the energy consumption of the 8KB cache is smaller than the 16KB cache, as each access to the cache consumes less energy. When the application does not fit anymore in the 8KB instruction cache, 16KB configuration takes advantage of its larger capacity and reduces its relative ED. The right hand side of the graph shows applications with high mpki. We can see how iLP-NUCA performs better in this kind of applications as it can exploit its full potential to reduce the ED. The behavior of the 4KB 2–way set associative cache is too irregular due to the small capacity.

Figures 4.5 and 4.6 are the equivalent graphs for 2 and 4 threads applications. Regarding two threads applications, again for the same L1-I cache size iLP-NUCA shows a relative ED below the baseline. Also the difference increases as we move to the applications that show a higher aggregated L1-I mpki (right hand side of the graph). If we reduce the cache size to 8KB 2–way set associative, for applications with low L1-I mpki we achieve a large reduction in ED, as the aggregated footprint fits in the first level cache. In some applications with high L1-I mpki values ILP–8KB–2ass has a relative value over the baseline. Nevertheless, in most of the applications iLP-NUCA reduces the relative ED product with an instruction cache with half of the capacity. Thus, the reduction in ED is consistent among all the applications with two threads.

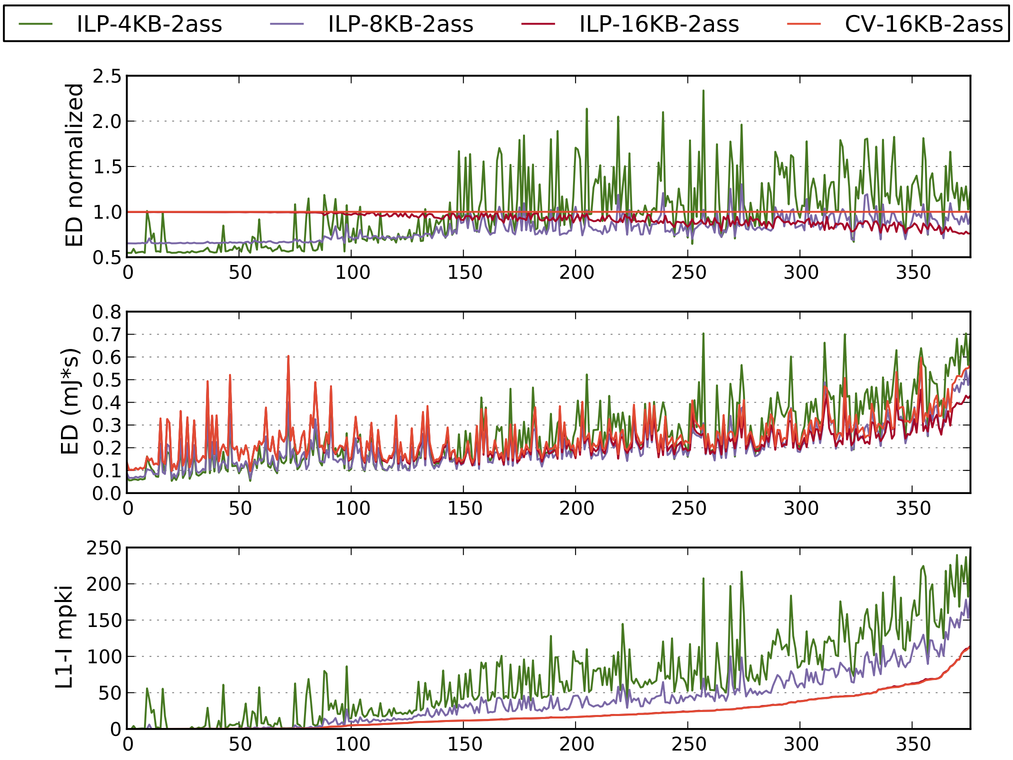


Figure 4.5: ED distribution for 2 threads applications. Names of configurations include the size of the cache and the associativity. For example: CV–32KB–4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. Data cache always hits. The upper graph represents ED values normalized to the best conventional configuration (baseline CV–16KB–2ass). The graph in the middle represents the absolute ED values. ED is a the lower–the–better metric. The lower graph represents the L1-I mpki. Benchmarks are sorted according to the L1-I mpki of the baseline (CV–16KB–2ass).

Figure 4.6 shows all the mixes considered in our experiments for 4 threads applications. The best conventional configuration in this case is a L1-I cache of 32KB 4–way set associative, so we consider the iLP-NUCA configurations iLP–32KB–4ass, iLP–16KB–2ass, and iLP–8KB–2ass. 4KB 2–way set associative first level instruction cache shows a very irregular pattern and it is too small to hold the aggregated instruction footprint of four applications. The same pattern as in previous figures can be observed. For the same size, iLP-NUCA performs better in terms of ED, and this differences are clearer as the aggregated L1-I mpki increases (applications on the right hand side of the graph). Shrinking the cache size to 16KB improves the ED product for those applications with low L1-I mpki. The relative ED product of this configuration (ILP–16KB–2ass) in general is below the baseline with some exceptions that rarely go beyond 10% increase, and usually reaches 20% decrease, until mix 350, approximately. Then the values swing around the baseline without big differences until approximately mix 500, where the relative ED is increased in some points by 20%. For most of the applications iLP-NUCA with a 16KB 2–way set associative instruction cache improves the ED of a conventional hierarchy with a 32KB 4–way set associative first level instruction cache. For more than half of the applications, iLP-NUCA with a L1-I 8KB 2–way set associative cache also outperforms a conventional hierarchy with four times more capacity and the double of the associativity.

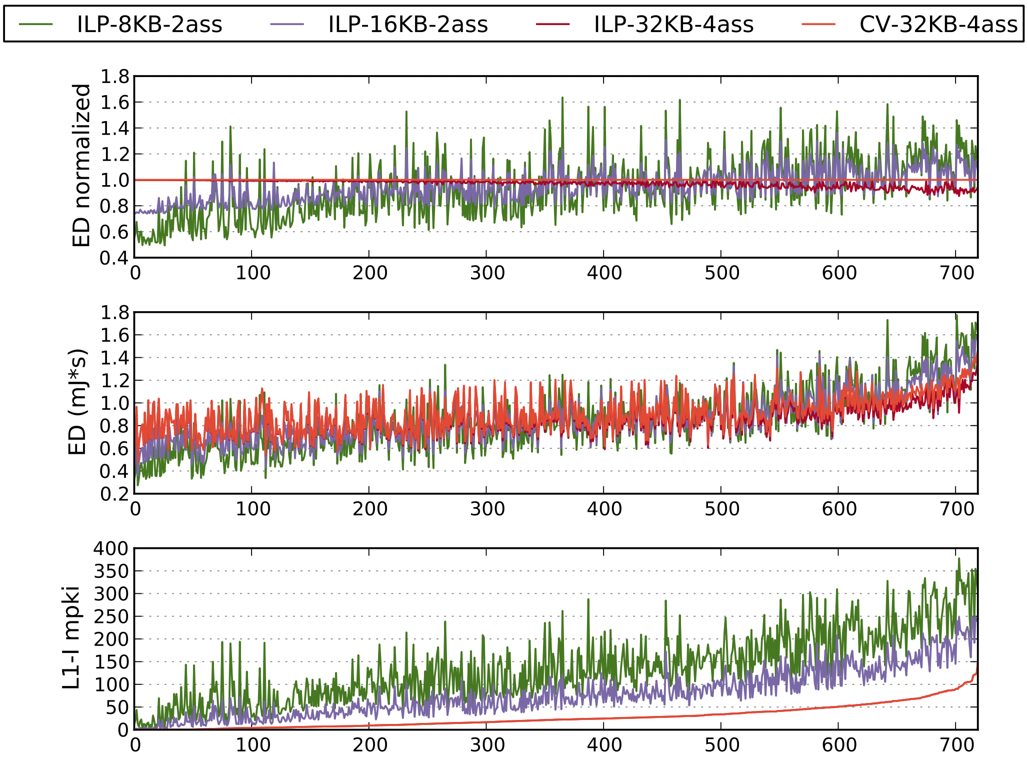


Figure 4.6: ED distribution for 4 threads applications. Names of configurations include the size of the cache and the associativity. For example: CV–32KB–4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. Data cache always hits. The upper graph represents ED values normalized to the best conventional configuration (baseline CV–32KB–4ass). The graph in the middle represents the absolute ED values. ED is a the lower–the–better metric. The lower graph represents the L1-I mpki. Benchmarks are sorted according to the L1-I mpki of the baseline (CV–32KB–4ass).

4.2.4 Summary

To summarize the results presented in this section, we can conclude that iLP-NUCAs are able to improve performance over conventional designs when the L1-I has the same size, and yet reduce the energy consumption. Even more, iLP-NUCA with a root-tile of 8KB, 2–way set associative achieves, on average, the performance of a conventional hierarchy with a 16KB, 2–way set associative cache, independently of the number of threads, and reduces the energy consumption by 21%, 18%, and 11%, for 1, 2, and 4 threads, respectively. With a 8KB, 2–way set associative root-tile, iLP-NUCA reaches 90%, 95%, and 99% performance of an ideal first level instruction cache. We have seen also that these results are consistent among the applications and that iLP-NUCA outperforms conventional caches specially in environments with high L1-I mpki (misses per k-instruction).

4.3 Putting all together: instructions and data

The results we have presented so far consider that data caches always hit. In this section we will examine the behavior of the complete system considering the same memory hierarchy model: a three level cache hierarchy, with L1 and L2 dedicated caches (i.e., separated for instructions and data), and shared L3. In iLP-NUCA configurations we substitute each L1/L2 for a 3 level iLP-NUCA (one iLP-NUCA for data, and one iLP-NUCA for instructions). The detailed micro-architectural parameters can be found in Table 3.1 (Chapter 3). To describe the behavior of the complete system we chose ED and ED^2 metrics.

Figure 4.7 shows Energy-Delay (Figure 4.7a) and Energy-Delay^2 (Figure 4.7b) products normalized to the baseline (CV–32KB–4ass) when the data cache is not perfect. The graphs show the same tendency as before and similar conclusions can be derived. A conventional hierarchy reduces ED and ED^2 when we reduce the L1-I cache to 16KB and 8KB for 1 thread applications, and increases for 4KB. With 2 and 4 threads shrinking the L1-I cache translates into bigger ED and ED^2 values. iLP-NUCA, on the other hand, keeps ED and ED2 below the conventional baseline and improves for 16KB and 8KB. The ED and ED^2 values increase for for 4KB L1-I but still they are below the conventional ones.

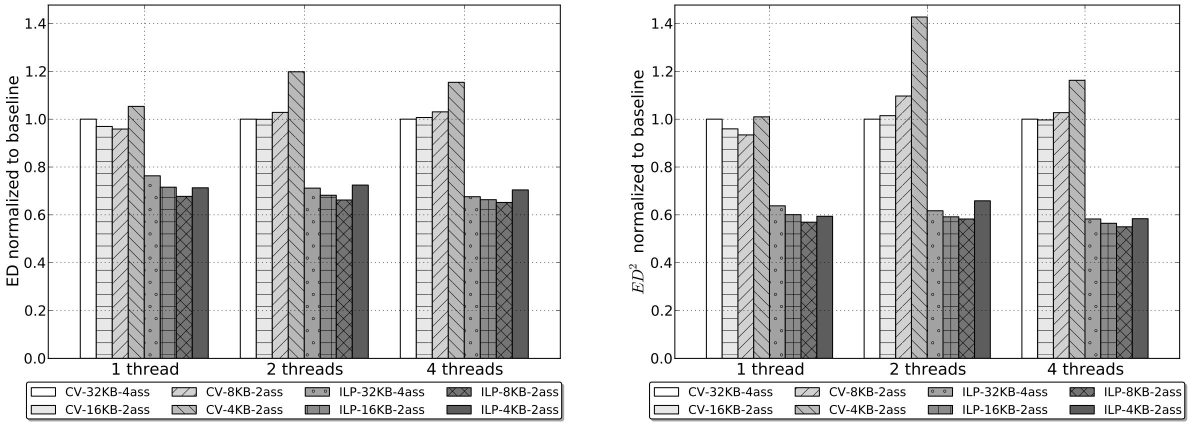
Several observations must be made. There are two factors that influence the drastic reduction in Energy–Delay.

- The energy savings. The energy savings are double by the combination of instruction iLP-NUCA and data iLP-NUCA.
- The performance increase. iLP-NUCA and its new transport network provide cache lines with a minimum latency taking great advantage of their temporal reuse. We detected that benchmarks with high slow downs in performance are bounded by stores. The higher iLP-NUCA bandwidth helps to increase the stores speed.

In conclusion, the system take advantage of faster instructions paths, faster data paths, and higher bandwidth to commit stores.

4.4 L1-I access latency

The decrease on the L1-I cache size and associativity influences the access latency and initiation rate of the cache. We modeled the different cache configurations with CACTI 6.5 [40] and our experiments show that for the given technology and cycle time (LSTP, 1 GHz) the access latency of the cache is never less than 2 cycles.



(a) Energy–Delay product for 1, 2, and 4 threads, several configurations. Values are normalized to baseline (CV–32KB–4ass).

(b) Energy–Delay^2 product for 1, 2, and 4 threads, several configurations. Values are normalized to baseline (CV–32KB–4ass).

Figure 4.7: ED (Figure 4.7a) and ED^2 (Figure 4.7b) products for 1, 2, and 4 threads. Each bar corresponds to one configuration. Values are normalized to baseline (CV–32KB–4ass). Names of configurations include the size of the cache and the associativity. For example: CV–32KB–4ass means conventional hierarchy, 32KB 4–way set associative L1-I cache. ED and ED^2 are the lower–the–better metrics.

Assuming that technology allows for accessing the L1-I cache in 1 cycle, we run experiments for different configurations of L1-I with both conventional and iLP-NUCA based hierarchies. Our results show that, as expected, the lower latency improves the performance of the system. However, the tendency remains, and the same conclusions extracted for 2-cycle access latencies can be observed. Furthermore, an instruction cache with 1-cycle access latency could request a cache line per cycle to the L2 in case of a batch of misses. iLP-NUCA can inject one miss per cycle and offers more bandwidth than a conventional hierarchy. Thus we also expect iLP-NUCA to perform better in this kind of scenario.

4.5 Results summary

To sum up, we have seen that first level instruction caches are responsible of a high percentage of the total energy consumption. We observed that by reducing the size and associativity of the L1-I we consume less energy, but at the expense of a non acceptable performance degradation. We proposed to utilize for instructions hierarchies LP-NUCAs with a new transport network (iLP-NUCAs). This new transport network effectively reduces the average service latency of the blocks and the energy consumption of the network. We also saw how iLP-NUCA achieves better performance than conventional hierarchies for the same configuration of L1-I, independently of the number of threads, and without being detrimental for any of them. Furthermore, it can achieve the same performance than a conventional hierarchy with smaller L1-I cache sizes, which allows for great energy reductions. In any case we observed that the optimal results in terms of Energy–Delay, on average, are those given by a 8KB 2–way set associative L1-I cache. The results are consistent for data, where we observed a twofold benefit due to the instructions and the data iLP-NUCAs.

Chapter 5 | **Related Work**

Several works addressed the instruction cache energy consumption problem by adding hardware structures that are able to capture a great amount of instruction fetches. Kin *et al.* proposed the Filter cache, a small direct mapped conventional cache placed before the first level instruction cache (L1-I) that trades off performance for power [32]. If the instruction request hits in the filter cache, the access is faster and consumes less energy. However, if the fetched instruction misses in the filter cache, we incur in a penalty delay for accessing the L1-I. Thus, although a 256-byte direct mapped filter cache achieves a 58% energy reduction, the performance loss is about 21%.

To cope with the extra delay incurred when a miss occurs in the filter cache, Jung *et al.* proposed to add a small cache (EM-cache) next to the L1-I instead of before [29]. In this way, some of the accesses will go to the L1-I and some of them will save energy by accessing only the EM-cache. The key idea is to detect if following instructions access the same cache block than the last instruction fetched. In this case, we can directly access the EM-cache. However, if we want to capture a bit more complicated patterns, we need to add instructions to the ISA and rely on the compiler to indicate the hardware when to access each structure.

In a similar way, Bellas *et al.* proposed the L-cache [6]. The L-cache contains instructions within a loop, so that iterations of the loop will fetch instructions from the L-cache rather than from the instruction cache, saving great amount of energy. As the compiler decides which blocks are kept in the L-cache, the performance loss in much lower. The energy savings are directly related to the amount of large blocks within loops, and the amount of times that those blocks are executed. Therefore this technique works well for floating point workloads, but the energy savings are lower in the case of integer workloads.

Trace caches follow a similar idea, although their main concern is increasing performance. Dynamic instruction streams are kept in an additional hardware structure to avoid noncontiguous fetches [43]. They were implemented in the Pentium 4, where they store already decoded micro-operations, or translations of complex x86 instructions, which had a positive impact by reducing the energy consumption of the machine [44].

All these proposals (and others like [57, 56]) add extra hardware structures between the processor and the first level instruction cache. Their goal is to reduce the amount of accesses 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 clear the interface between the processor and the L1 instruction cache, and therefore, all these designs are orthogonal to ours, and they could be easily implemented in front of our system.

Besides, these works do not consider multi-thread environments, where threads interfere with each other and may pollute the filter structure.

Aragon *et al.* proposed a technique to reduce the power consumption of CAM-based instruction cache designs with high associativity [2]. The key observation is that not all the instructions fetched are executed due to taken branches. They implement a prefetch mask to avoid fetching instructions from a block that will not be used.

Associativity tends to have a strong impact on power consumption on modern high-performance low-power embedded systems. First level caches of such systems usually have associativity varying from 2 to 8. For example: 2–way for ARM Cortex A15 [53], 4–way for IBM PowerPC 476FP [35] and ARM Cortex A9 [3], or 8–way for the Freescale QorIQ AMP T2080 [7]; in comparison with the 32-way associative L1-I cache used in Aragon's work.

Furthermore, CAMs generally have about twice the area of SRAM cells and tend to consume a large amount of power because the matchlines are heavily loaded and have an activity factor close to 1 [55]. These structures are preferred for TLBs (translation lookaside buffers), as they require a high-speed table look-up. They are also especially popular in network routers (where they are very useful for IP classification and forwarding).

Although SMT get more benefit from associativity than from size [14], our experiments conclude that instruction caches with associativities between 2 and 4 work well while they keep a reasonable energy consumption.

Other proposals also exploit the varying requirements of instruction caches not only among programs, but also among phases of the same program, to reduce the cache energy consumption.

There are some techniques to selectively place some of the unused cache lines into a low-leakage, state-preserving or switched-off state [11, 30, 42]. These approaches are straightforward applicable to our system and can account for additional energy savings.

Ishihara *et al.* proposed to relax the cache uniformity constraint to reduce the dynamic power consumption and the leakage power [24]. They developed both a non-uniform cache architecture which varies the number of ways per set, and a code placement algorithm to reduce cache accesses and cache misses. Other similar works that focus on analyzing cache requirements at run-time are [1, 8, 15, 58].

A recent proposal from Sundararajan *et al.* is Smart cache, a reconfigurable cache that adapts dynamically to the requirements of each program/phase [50]. They proposed a way to dynamically reconfigure the cache parameters (size and associativity) at run-time based on the behavior on the application with a machine learning decision tree model. This model is applicable to all the cache levels. They reduce the energy-delay product with an overall performance degradation of 2%.

All these proposals imply important design changes in the structure of the caches. The key advantage is that the variable L1-I size or associativity allows to capture big working sets of demanding applications and at the same time reduce energy when the requirements are small. The main performance losses come when the prediction of a certain optimal size is wrong (smaller than it should) and the L1-I cannot capture all the accesses. iLP-NUCA works as a distributed victim cache which backs up the L1-I. We believe that the combination of a reconfigurable cache approach and our iLP-NUCA would achieve high energy reductions while keeping performance.

Other proposals take advantage of the compiler to place blocks in an specific location of the cache. Jones *et al.* proposed to place frequently executed code at the start of the program binary and explicitly place these instructions in a particular way [27]. Thus, in a given access, the processor would be able to detect that the request is aimed to the selected instructions, and only one way and one set will need to be accessed. Similar proposals are [36, 23].

All these approaches are orthogonal to our design, as we do not modify the interface between the processor and the first level cache.

Victim caches [28] have also been studied as energy saving structures [37, 5]. Memik *et al.* proposed to put a victim cache before the L2 and enhance its behavior by a predicting schema. This schema would tell us if a block is not in the victim cache, or per contra is likely to be there [37]. In this way they can save some of the accesses to the victim cache that will not produce a hit.

Our experiments show that a conventional victim cache (8-16 entries) is not enough to capture the instruction footprint and produce a significant improve in performance. This work does not consider SMT applications neither. Several threads running together and sharing a small victim cache would interfere each other by evicting other threads victim blocks. Besides, iLP-NUCA is far more complex than a conventional victim cache and its networks-in-cache offer a fast communication with the L1 cache and lots of bandwidth.

Chapter 6 | **Conclusion and future work**

Although multi-threading processors can increase the performance of embedded systems with a minimum overhead, fetching instructions from multiple threads each cycle also increases the pressure on the instruction cache. Instruction caches are responsible of a high percentage of the total energy consumption of the chip, which for battery-powered embedded devices becomes a critical issue.

In this Master's Thesis we proposed to reduce the size of the first level instruction cache to achieve significant energy reductions. We proposed iLP-NUCA, an energy efficient tiled structure, that substitutes a conventional first and second level caches, for the instruction hierarchy, We provide iLP-NUCA of a new transport network-in-cache that reduces the average service latency of blocks 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 and L2 are dedicated (separated for instructions and data), and L3 is shared. To evaluate both proposals we implemented them in our simulation environment, providing it of instruction cache hierarchies and a realistic fetch unit.

We run experiments of representative applications (SPEC CPU2006) for the two structures (conventional and iLP-NUCA), varying the L1-I cache size and associativity, for 1, 2, and 4 threads.

From these experiments we can extract the following conclusions:

- First level instruction caches energy consumption represents an important percentage of the total energy consumed by the memory hierarchy.
- By reducing the L1-I cache size we can achieve great energy reductions. However, we suffer from a non acceptable performance degradation.
- In a system with ideal data caches, iLP-NUCA in comparison with a conventional hierarchy performs better and consumes less energy for the same L1-I size, independently of the number of threads. In this way iLP-NUCA achieves, on average, an Energy–Delay product (ED) relative to a conventional hierarchy with a 32KB, 4–way set associative L1-I cache, of 0.98, 0.83, and 0.70 keeping the same L1-I size, and shrinking the L1-I to 16KB, and 8KB for one thread applications. Two threads applications reduce the ED by more than 10% for 16KB and 8KB, and 4 threads applications by more than 5%. This ED improvements are consistent for the distribution of applications considered, being greater the benefits when the applications footprints increase the misses of the first level cache.
- We observe that when we model both instructions and data in the system the results are consistent.

In conclusion with iLP-NUCA we can implement a 8KB 2–way set associative first level instruction cache and get the same performance, on average, than a double sized conventional cache, independently of the number of threads. This translates into a reduction of the energy delay product of 21%, 18%, and 11%, and we reach 90%, 95%, and 99% of the ideal performance for 1, 2, and 4 threads, respectively.

6.1 Publications

The publications related to this Master thesis are the following:

- Alexandra Ferrerón, Darío Suárez and Víctor Viñals. Tiled instruction caches. *Poster.* Student Poster Session of the 7th International Conference on High-Performance and Embedded Architectures and Compilers. January 2012. *Best student poster award.*
- Alexandra Ferrerón, Marta Ortín, Darío Suárez, Jesús Alastruey and Víctor Viñals. iLP-NUCA: Cache de Instrucciones Teselada para Procesadores Empotrados. Prooceedings of the 23th Jornadas de Paralelismo (JJPAR'12). September 2012.
- Alexandra Ferrerón, Marta Ortín, Darío Suárez, Jesús Alastruey and Víctor Viñals. Shrinking L1 Instruction Caches to Improve Energy–Delay in SMT Embedded Processors. Future submission (September 2012) to 26th International Conference on Architecture of Computing System (ARCS 2013).

6.2 Future Work

This Master Thesis work lays the foundations for a PhD. dissertation. In the near future we would like to study in more detail the following ideas.

Some recent works claimed that the traditional benchmarks such as SPEC CPU do not represent the real behavior of applications [10, 16]. Characterizations of those new benchmarks suites point out that applications have high instruction requirements. iLP-NUCAs work well in environments where L1 caches present a high mpki (misses per k-instruction). Thus, we expect that in such applications iLP-NUCAs get performance benefits. We want to test our structure using new benchmarks like BBench [16] or EEMBC [41] to corroborate our hypothesis.

Mostly all portable devices comprise multicore processors. One of our priorities is to implement iLP-NUCAs in a multicore context. This implies a complete re-evaluation of the structure considering coherence, quality of service, etc. The characteristics of our structure let us think that we can also get benefit in this area.

Regarding the iLP-NUCA transport network, we saw that from the components of the transport network (the switch and the two buffers per tile), each buffer was responsible of one third of the energy consumption, and the other third was caused by the switch. With this tree-based transport network we could decrease the energy consumption by simplifying the switch design. The low utilization rates of the transport network would allow us to implement more aggressive techniques. A possible design would eliminate the buffers, and offer a direct way from each tile to the root-tile in just one hop. We want to explore this design where transport blocks can reach the root-tile in one cycle by traversing a bufferless tree-based network [39]. We want to sintetyze this new topology in hardware to test its functionality and quantify its benefits.

Finally, we have seen that there are not big opportunities for larger improvements in terms of performance and energy in first level instruction caches with hardware-only based techniques. We hardly believe that further energy savings are possible by hardware/software co-design techniques. Compilation techniques are a high candidate to improve the energy efficiency of our structure. In order to explore all this alternatives, we are going to collaborate with Professor Michael O'Boyle from the *Compiler and Architecture Co-Design Group* of the University of Edinburgh, where I will be doing an internship next year.

Bibliography

[1] David H. Albonesi. Selective cache ways: on-demand cache resource allocation. In *MICRO 32: Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture*, pages 248–259, Washington, DC, USA, 1999. IEEE Computer Society.

[2] Juan L. Aragón and Alexander V. Veidenbaum. Optimizing cam-based instruction cache designs for low-power embedded systems. *J. Syst. Archit.*, 54(12):1155–1163, December 2008.

[3] ARM. The ARM Cortex-A9 Processors, 2009. `www.arm.com/files/pdf/ARMCortexA-9Processors.pdf`.

[4] Todd Austin and Doug Burger. *SimpleScalar Tutorial (for tool set release 2.0)*. SimpleScalar LCC, 1997.

[5] R. Iris Bahar, Gianluca Albera, and Srilatha Manne. Power and performance tradeoffs using various caching strategies. In *Proceedings of the 1998 international symposium on Low power electronics and design*, ISLPED '98, pages 64–69, New York, NY, USA, 1998. ACM.

[6] N. Bellas, I.N. Hajj, C.D. Polychronopoulos, and G. Stamoulis. Architectural and compiler techniques for energy reduction in high-performance microprocessors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(3):317 –326, june 2000.

[7] Joseph Byrne. Freescale drops quad-core threshold. *Microprocessor Report*, 26(7):10–12, July 2012.

[8] Liming Chen, Xuecheng Zou, Jianming Lei, and Zhenglin Liu. Dynamically reconfigurable cache for low-power embedded system. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 5, pages 180 –184, aug. 2007.

[9] S. Eyerman and L. Eeckhout. System-level performance metrics for multiprogram workloads. *Micro, IEEE*, 28(3):42 –53, may. 2008.

[10] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '12, pages 37–48, New York, NY, USA, 2012. ACM.

[11] Krisztian Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proceedings of the 29th annual international symposium on Computer architecture*, pages 148–157. IEEE Computer Society, 2002.

[12] R. Gabor, S. Weiss, and A. Mendelson. Fairness and throughput in switch on event multithreading. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 149–160, Washington, DC, USA, 2006. IEEE Computer Society.

[13] J. Scott Gardner. Mips aptiv cores hit the mark. *Microprocessor Report*, 26(5):1–11, May 2012.

[14] R. Gonçalves, E. Ayguadé, and M. Valero. A simulator for SMT architectures: Evaluating instruction cache topologies. In *12th Symposium on Computer Architecture and High Performance Computing*, pages 279–286, 2000.

[15] Ann Gordon-Ross, Jeremy Lau, and Brad Calder. Phase-based cache reconfiguration for a highly-configurable two-level cache hierarchy. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, GLSVLSI '08, pages 379–382, New York, NY, USA, 2008. ACM.

[16] A. Gutierrez, R. G. Dreslinski, T. F. Wenisch, T. Mudge, A. Saidi, C. Emmons, and N. Paver. Full-system analysis and characterization of interactive smartphone applications. In *IEEE International Symposium on Workload Characterization*, pages 81–90, November 2011.

[17] Linley Gwennap. What's inside the krait. *Microprocessor Report*, 26:1–9, June 2012.

[18] Tom R. Halfhill. Netlogic broadens XLP family. *Microprocessor Report*, 24(7):1–11, 2010.

[19] Greg Hamerly, Erez Perelman, Jeremy Lau, and Brad Calder. Simpoint 3.0: Faster and more flexible program analysis. In *Proceedings of Workshop on Modeling, Benchmarking and Simulation*, 2005.

[20] John L. Henning. SPEC CPU2000: Measuring cpu performance in the new millennium. *Computer*, 33(7):28–35, 2000.

[21] John L. Henning. SPEC CPU2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, 2006.

[22] Sébastien Hily and André Seznec. Contention on 2^{nd} level cache may limit the effectiveness of simultaneous multithreading. Technical Report 1086, IRISA, fébrier 1997.

[23] Koji Inoue, Tohru Ishihara, and Kazuaki Murakami. Way-predicting set-associative cache for high performance and low energy consumption. In *Proceedings of the 1999 international symposium on Low power electronics and design*, ISLPED '99, pages 273–275, New York, NY, USA, 1999. ACM.

[24] Tohru Ishihara and Farzan Fallah. A non-uniform cache architecture for low power system design. In *Proceedings of the 2005 international symposium on Low power electronics and design*, ISLPED '05, pages 363–368, New York, NY, USA, 2005. ACM.

[25] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley & Sons, Inc., April 1991.

[26] M. Johnson. *Superscalar microprocessor design.* Prentice Hall series in innovative technology. Prentice Hall, 1991.

[27] Timothy M. Jones, Sandro Bartolini, Bruno De Bus, John Cavazos, and Michael F. P. O'Boyle. Instruction cache energy saving through compiler way-placement. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '08, pages 1196–1201, New York, NY, USA, 2008. ACM.

[28] Norman P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proceedings of the 17th annual international symposium on Computer Architecture*, ISCA '90, pages 364–373, New York, NY, USA, 1990. ACM.

[29] Changwoo Jung and Jihong Kim. Instruction cache organisation for embedded low-power processors. *Electronics Letters*, 37(9):554 –555, apr 2001.

[30] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th annual international symposium on Computer architecture*, pages 240–251. ACM Press, 2001.

[31] Changkyu Kim, Doug Burger, and Stephen W. Keckler. 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 systems (ASPLOS-X)*, pages 211–222. ACM Press, October 2002.

[32] J. Kin, Munish Gupta, and W.H. Mangione-Smith. The filter cache: an energy efficient memory structure. In *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, pages 184 –193, dec 1997.

[33] Chunho Lee, M. Potkonjak, and W.H. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, pages 330 –335, dec 1997.

[34] Yingmin Li, D. Brooks, Zhigang Hu, K. Skadron, and P. Bose. Understanding the energy efficiency of simultaneous multithreading. In *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*, ISLPED '04, pages 44–49, New York, NY, USA, aug. 2004. ACM.

[35] LSI Corporation. PowerPC$^{\mathrm{TM}}$ processor (476FP) embedded core product brief, `http://www.lsi.com/DistributionSystem/AssetDocument/PPC476FP-PB-v7.pdf`, January 2010.

[36] Albert Ma, Michael Zhang, and Krste Asanovic. Way memoization to reduce fetch energy in instruction caches. In *ISCA Workshop on Complexity Effective Design*. MIT, 2001.

[37] Gokhan Memik, Glenn Reinman, and William H. Mangione-Smith. Reducing energy and delay using efficient victim caches. In *Proceedings of the 2003 international symposium on Low power electronics and design*, ISLPED '03, pages 262–265, New York, NY, USA, 2003. ACM.

[38] J. Montanaro, R.T. Witek, K. Anne, A.J. Black, E.M. Cooper, D.W. Dobberpuhl, P.M. Donahue, J. Eno, A. Farell, G.W. Hoeppner, D. Kruckemyer, T.H. Lee, P. Lin, L. Madden, D. Murray, M. Pearce, S. Santhanam, K.J. Snyder, R. Stephany, and S.C. Thierauf. A 160 mhz 32 b 0.5 w cmos risc microprocessor. In *Solid-State Circuits Conference, 1996. Digest of Technical Papers. 42nd ISSCC., 1996 IEEE International*, pages 214 –215, 447, feb 1996.

[39] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 196–207, New York, NY, USA, 2009. ACM.

[40] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *MICRO 40: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 3–14, Washington, DC, USA, 2007. IEEE Computer Society.

[41] Jason A. Poovey, Thomas M. Conte, Markus Levy, and Shay Gal-On. A benchmark characterization of the eembc benchmark suite. *IEEE Micro*, 29(5):18–29, September 2009.

[42] Michael Powell, Se-Hyun Yang, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar. Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories. In *Proceedings of the 2000 international symposium on Low power electronics and design*, ISLPED '00, pages 90–95, New York, NY, USA, 2000. ACM.

[43] Eric Rotenberg, Steve Bennett, and James E. Smith. Trace cache: a low latency approach to high bandwidth instruction fetching. In *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 29, pages 24–35, Washington, DC, USA, 1996. IEEE Computer Society.

[44] Dave Sager, Desktop Platforms Group, and Intel Corp. The microarchitecture of the pentium 4 processor. *Intel Technology Journal*, 1:2001, 2001.

[45] S. Segars. Low power design techniques for microprocessors. ISSCC Tutorial note, February 2001.

[46] J. Shen. *Modern Processor Design: Fundamentals of Superscalar Processors*. McGraw-Hill Series in Electrical and Computer Engineering. McGraw-Hill Companies,Incorporated, 2004.

[47] D. Suárez, G. Dimitrakopoulos, T. Monreal, M. G. H. Katevenis, and V. Vi nals. LP-NUCA: Networks-in-cache for high- performance low-power embedded processors. *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, PP(99):1, August 2011.

[48] D. Suarez, T. Monreal, F. Vallejo, R. Beivide, and V. Viñals. Light NUCA: a proposal for bridging the inter-cache latency gap. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 530–535, 3001 Leuven, Belgium, Belgium, 2009. European Design and Automation Association.

[49] D. Suárez, Monreal T, and V. Viñals. A comparison of cache hierarchies for SMT processors. In *Proc. of the 22^{th} Jornadas de Paralelismo (JJPAR'11)*, 2011.

[50] Karthik T. Sundararajan, Timothy M. Jones, and Nigel Topham. Smart cache: A self adaptive cache architecture for energy efficiency. In *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2011.

[51] D. Tullsen, S. J. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm. Exploiting choice: instruction fetch and issue on an implementable simultaneous multithreading processor. In *Proceedings of the 23rd annual international symposium on Computer architecture*, ISCA '96, pages 191–202, New York, NY, USA, 1996. ACM.

[52] Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy. Simultaneous multithreading: maximizing on-chip parallelism. In *Proceedings of the 22nd annual international symposium on Computer architecture*, ISCA '95, pages 392–403, New York, NY, USA, 1995. ACM.

[53] Jim Turley. Cortex-A15 "Eagle" flies the coop. *Microprocessor Report*, 24(7):1–11, November 2010.

[54] Dennis Wackerly, William Mendenhall, and Richard L. Scheaffer. *Mathematical Statistics with Applications.* BROOKS/COLE CENGAGE Learning, 7th edition, 2008.

[55] Neil H. E. Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective.* Addison Wesley, 4th edition, 2010.

[56] Chia-Lin Yang and Chien-Hao Lee. Hotspot cache: joint temporal and spatial locality exploitation for i-cache energy reduction. In *Proceedings of the 2004 international symposium on Low power electronics and design*, ISLPED '04, pages 114–119, New York, NY, USA, 2004. ACM.

[57] Jun Yang and Rajiv Gupta. Energy-efficient load and store reuse. In *Proceedings of the 2001 international symposium on Low power electronics and design*, ISLPED '01, pages 72–75, New York, NY, USA, 2001. ACM.

[58] Chuanjun Zhang, Frank Vahid, and Walid Najjar. A highly configurable cache for low energy embedded systems. *ACM Trans. Embed. Comput. Syst.*, 4(2):363–387, May 2005.