Architectural Exploration of Last-Level Caches targeting Homogeneous Multicore Systems

[1]Rodrigo Cataldo, [1]Guilherme Korol, [1]Ramon Fernandes, [2]Debora Matos, [1]César Marcon
[1]Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
[2]Universidade Estadual do Rio Grande do Sul (UERGS), Porto Alegre, Brazil
{rodrigo.cataldo, guilherme.korol, ramon.fernandes}@acad.pucrs.br; debora.motta@gmail.com, cesar.marcon@pucrs.br

***Abstract*— The Last-Level Cache (LLC) influences the overall system performance and power dissipation in multicore systems significantly. This paper evaluates five LLC architectures targeting execution time, dynamic and static power dissipation, and area consumption. They are measured using the widely adopted PARSEC benchmark suite for parallel shared-memory systems. Employing Gem5 full-system simulator and 32 nm technology characterization of the McPAT framework, this work had two interesting findings: (i) the shared LLC has the overall best performance under the PARSEC parallel workload, even for applications with less than 20% of shared data. (ii) A privately accessed cache can reduce up to 20 times the dynamic power dissipation on 32 nm technology and 25 times the area consumption when compared to shared-accessed caches.**

Keywords—multicore system; last-level cache; Gem5

I. INTRODUCTION

Shrinking process technologies and the power wall phenomenon have resulted in the development of multicore architectures to achieve the increasing demand of processing and bandwidth requirements [1]. Many interesting challenges arise in these architectures such as the appropriate sharing of on-chip resources across multiple cores and threads, as well as operating under a tight thermal design point [2]. In addition, highly complex heatsink are not suitable for such designs [3]. Caches are essential to hide main memory latency and, thus, to give support to highly efficient cores.

While the use of privately-accessed first cache level (L1) is predominant on multicore designs, for LLC, there is not a dominant architecture paradigm [1][4]. The trend of the recent commercial multicore (IBM POWER8, SPARC 64X and M7, Samsung Exynos 5) is employing shared LLC due to its efficiency of sharing data while many cores (Intel Xeon Phi, Tilera-Gx) focused on private designs due to its multi-hop interconnect nature.

Software-defined cache partitioning was proposed to adapt the LLC architecture for a given application set dynamically. However, only a set of cache parameter is available at the software level, which limits the potential of this approach and can even lead to an increase of execution time [5]. Therefore, trade-offs analysis of LLC on-chip architectures is a significant topic for the computer architecture community.

The main contributions of this paper are:

- Evaluation of the effects for committing to a given LLC architecture targeting homogeneous multicore systems. The first metric analyzed is the application time achieved through the execution of eight application from the PARSEC benchmark [6].
- An LLC architecture analysis related to dynamic and static power dissipation and area consumption. These two metrics are achieved through the 32 nm technology characterization of the McPAT framework [7]. Although recently made 28nm chips have been able to reduce its static power dissipation using newer fabrication processes [8], LLC still can produce significant static power due to its memory footprint [9]. Besides, advances on minimizing power dissipation are reaching the believed theoretical minimal possible [10].

The experimental results enable to get several interesting conclusions. Applications that share even less than 20% of data can still benefit significantly from a shared LLC [11]. However, shared designs have a single large footprint that requires more power dissipation for every access to its content. Therefore, we observed the opposite scenario for both area consumption and power dissipation. Privately accessed designs were up to twenty-two times and twenty-five times more efficient than a traditional shared LLC design, respectively.

This paper is organized as follows. Section II discusses some related works. Section III describes the system and LLC architecture. Section IV shows our methodology for evaluating architecture designs. Section V presents and Section VI discusses the experimental results. Finally, Section VII presents our conclusions and future works.

II. RELATED WORK

There have been some studies of the impact of LLC on multiple core systems over the years. There is not a single LLC architecture that is perfect for all workloads. Therefore, it is imperative to state the intended group of requirements that the LLC must attain.

Asaduzzaman et al. [1] analyze the impact of sharing LLC space on the performance and power of homogeneous multicore embedded systems. They compare shared and distributed cache organizations over a single and an 8-core system. Experimental results have shown that the FFT application mean delay is reduced by 64% for distributed LLC and 18% for shared LLC when an 8-core is compared to a single-core system. For the three applications analyzed (FFT, MI, and DFT), the distributed LLC always outperforms the shared LLC. Furthermore, the system consumed less power

978-1-5090-2736-1/16/$31.00 ©2016 IEEE

when employing the distributed organization. The authors do not consider the impact of the system interconnect (it is assumed to be negligible) and the small set of applications is restricted to fit in the L1 cache. Therefore, they do not benefit instruction-wise from sharing LLC space.

Sabry et al. [4] investigated the best L2 cache architecture for embedded MPSoCs. They propose a framework to study four L2 cache architectures: private, multi-port shared L2, single-port multiplexed L2 and a hybrid architecture that uses a small cache for shared data and private caches for private data. Their work uses a fixed shared L3 cache for the LLC architecture. The shared designs of L2 cache dissipated the most power (as they consume more power per access) and had the worst execution time. The multi-port shared L2 had a better performance since it provides multiple entries and avoids congestion. This cache organization occupied a large area due to its wiring scheme. For applications with shared data, the hybrid cache had the best performance and the lowest energy consumption of all designs analyzed. Unfortunately, the framework proposed and employed in this work is not readily available for other researchers.

Yun et al. [5] evaluate the effectiveness of employing a page-coloring technique for cache partitioning on multicore platforms. This technique is intended to isolate LLC space according to the system's need – in other words, the LLC architecture is controlled by the operating system and can be shared, private or a hybrid between the two. Unfortunately, as the cache hardware is not entirely available for software parametrization, negative side effects can occur. The authors observed up to 14 times slowdown in out-of-order cores due to interference in accessing the LLC – precisely the opposite of the desired effect. The cause is attributed to the contention in the Miss Status Holding Registers (MSHR) of the LLC.

Cheng et al. [9] conduct a study for energy-efficient LLCs in Chip Multiprocessors. They propose a runtime strategy to disable portions of a shared LLC for energy reduction through hardware monitors. Using similar tools (Gem5, PARSEC, and McPAT) the authors analyze the power-saving and performance impact with different power management policies. They observed power reduction with less than 2% of performance degradation and less than 3% of area overhead.

III. The Proposed Architecture

This section defines the baseline architecture for evaluating LLC designs. Next, we propose four additional LLC designs to evaluate against the baseline.

A. Baseline Architecture

The baseline architecture comprises eight homogeneous 1GHz ARM-v7a cores capable of executing out-of-order instructions on a seven-stage pipeline. All cores have private access to L1 cache of data and instructions, each supporting 32KiB of data. Both L1 caches use a 2-way associativity and an MSHR queue of six slots. The L1 instruction cache and data cache have a 1ns latency (one core cycle) and a 2ns latency (two core cycles), respectively. The baseline LLC is an L2 cache shared by all cores. Coherence is maintained by a cache-coherent interconnect based on the ARM CCI-400 [12] and the MOESI coherence protocol.

The Wide I/O version 2 is employed as the main memory here. Wide I/O is a standard memory interface that increases the memory bandwidth at a lowest possible power dissipation. The key is to stack multiple memory channels on top of the system and interconnect them through TSV (Through Silicon Via). Recently, JEDEC[1] published the second standard of Wide I/O that presents significant improvements. We use eight memory channels clocking at 266MHz (real clock).

Figure 1 depicts a schematic representation of the baseline architecture. In this figure, only four cores are shown instead of the eight actually present.

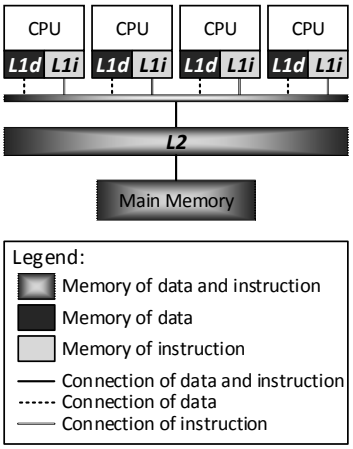


Figure 1. Schematic representation of the baseline architecture.

B. LLC Design Exploration

The LLC design exploration will be achieved using five cache designs, which three of them are shared by the cores, and two of them are privately accessed. Figure 2 represents the four additional LLC designs to the baseline. Their definition is as follows.

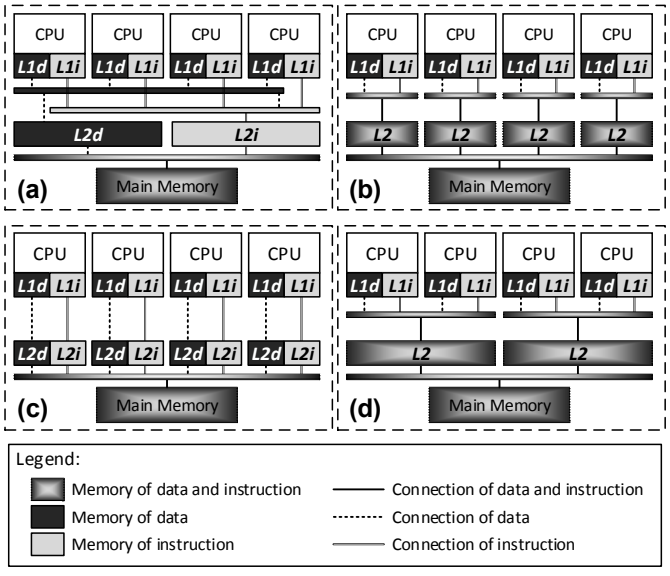


Figure 2. The LLC design exploration.

[1] JEDEC is a global leader in developing open standards for the microelectronics industry, with more than 3,000 volunteers representing nearly 300 member companies (source: www.jedec.org/about-jedec).

(i) **Shared LLC** (Figure 1) is the baseline cache design. There is no distinction between instructions and data on L2 cache; (ii) **Shared LLC I+D** (Figure 2(a)) has the same shared design as the previous cache, yet it separates L2 in instructions and data cache; (iii) **Paired Shared LLC** (Figure 2(d)) is based on the LLC proposed by SPARC M7 [13]. It shares an LLC by a subset of the total core count. In our evaluation, there is a shared LLC for every two cores; (iv) **Private LLC (**Figure 2(b)) has one cache for each core in the system. For our evaluation, this results in eight LLCs. Besides, it shares data and instructions indiscriminately; and (v) **Private LLC I+D** (Figure 2(c)) has two caches for each core in the system.

Table 1 presents the configuration parameters for all LLC designs. We use the following principle for size configuration: all caches must have the same overall size. Thus, a single LLC (Shared L2) have a total of 1MiB of unified data. The Shared L2 I+D has two LLCs for the system – consequently, each one of them has 512KiB of data, and so on.

Table 1. Cache parameters for five LLC designs.

	Shared L2	*Shared L2 I+D*	*Private L2*	*Private L2 I+D*	*Paired shared L2*
Size	1 MiB	512 KiB + 512 KiB	128 KiB	64 KiB + 64 KiB	256 KiB
Hit latency	12 ns	10 ns + 10 ns	8 ns	4 ns + 4 ns	10 ns
Associativity	16-way	16-way + 16-way	16-way	16-way + 16-way	16-way
MSHR queue size	16	14 + 14	12	8 + 8	14

The hit latency and MSHR queue size are adjusted according to their respective cache size. We use conservative values based on previous work using McPAT [14]. Bigger data space has higher access latency because we employ uniform memory access. Associativity is fixed at 16-way as this is a value used by the ARM family on L2 caches [15].

IV. Methodology

We used two frameworks to achieve the experimental results. Firstly, we employed the Gem5 full system simulator, a widely applied framework for architecture exploration [16]. Secondly, we used the McPAT framework, as it is one of the most accurate tool for this purpose [4], to evaluate the power dissipation and area consumption of every LLC design.

We used the out-of-order CPU model, which is the most detailed mode of execution on Gem5, to provide an accurate characterization of the application set.

Our application set is the PARSEC benchmark suite due to its broad range of application domains, parallel techniques and emerging workloads [17]. Table 2 highlights the subset of PARSEC used here. This subset gives us a diverse application domain and all combinations of parallelization model, working set size and data usage. We simulated the entire application time using the largest input intended for simulators available (*simlarge*). All applications were compiled with the GNU Compiler Collection using the O3 optimization flag. Moreover, every application was executed three times, and their average results were normalized.

The Linux Kernel 3.03-rc3 was compiled, and a suitable filesystem was built to execute PARSEC. The effects of mapping techniques are outside of this work. They are complementary for our evaluation. As such, the standard Linux Kernel scheduler, Completely Fair Scheduler, was used. Finally, we targeted our system for 32 nm processing technology and SRAM technology for LLCs.

Table 2. The eight applications of PARSEC benchmark suite employed in this work and its characteristics [17].

Program	Application domain	Parallelization		Working set	Data usage	
		Model	Granularity		Sharing	Exchange
Blackscholes	Financial analysis	Data-parallel	Coarse	Small	Low	Low
Bodytrack	Computer vision	Data-parallel	Medium	Medium	High	Medium
Canneal	Engineering	Unstructured	Fine	Unbounded	High	High
Dedup	Enterprise storage	Pipeline	Medium	Unbounded	High	High
Fluidanimate	Animation	Data-parallel	Fine	Large	Low	Medium
Swaptions	Financial analysis	Data-parallel	Coarse	Medium	Low	Low
Vips	Media processing	Data-parallel	Coarse	Medium	Low	Medium
X264	Media processing	Pipeline	Coarse	Medium	High	High

V. Experimental Results

We first analyze the performance of the five cache designs using the execution time of the *simlarge* input set for eight applications of PARSEC. Then, we analyze the area and power consumption for the same cache designs.

A. Execution time for eight PARSEC benchmarks

Figure 3 summarizes the results obtained. All values are relatively normalized according to the baseline LLC; i.e., for all applications, the execution time of the baseline cache is 0% and the remaining values are perceptual deviations of this reference. The first significant observation from these results is the fact that no other cache organization executes substantially faster than the baseline. Blackscholes, Fluidanimate and Swaptions are the cases where some cache organization executes faster; however, for a limited amount (less than 3%). For the remaining five applications, the new organizations are up to 90% slower, as in the case of Bodytrack.

To clarify how these cache organizations are affecting the performance of the applications, we discuss the LLC miss rate for three applications: Blackscholes, Bodytrack, and x264, as illustrated in Table 3.

Table 3. LLC miss rate for Blackscholes, Bodytrack, and x264.

	Core 0	*Core 1*	*Core 2*	*Core 3*	*Core 4*	*Core 5*	*Core 6*	*Core 7*		
Blackscholes	**4.0%**									**Baseline**
	0.2%								**I**	**Shared L2**
	10.9%								**D**	
	16.2%	**14.8%**	**16.5%**	**14.0%**	**3.1%**	**12.1%**	**11.1%**	**12.4%**		**Private L2**
	50.8%	**45.1%**	**47.7%**	**47.3%**	**0.3%**	**34.4%**	**33.2%**	**50.5%**	**I**	**Private L2**
	20.1%	**19.5%**	**20.5%**	**18.1%**	**20.5%**	**17.6%**	**16.7%**	**18.0%**	**D**	
		12.5%		**11.5%**		**3.27%**		**8.78%**		**Paired Shared L2**
Bodytrack	**1.0%**									**Baseline**
	0.2%								**I**	**Shared L2**
	1.4%								**D**	
	31.0%	**30.9%**	**30.7%**	**31.6%**	**31.5%**	**31.0%**	**30.7%**	**30.9%**		**Private L2**
	2.2%	**1.5%**	**2.2%**	**1.9%**	**3.5%**	**2.0%**	**1.4%**	**1.1%**	**I**	**Private L2**
	61.1%	**61.0%**	**61.0%**	**61.0%**	**59.6%**	**60.1%**	**60.9%**	**61.2%**	**D**	
		3.9%		**3.3%**		**6.0%**		**3.2%**		**Paired Shared L2**
x264	**12.8%**									**Baseline**
	0.1%								**I**	**Shared L2**
	25.5%								**D**	
	33.4%	**34.1%**	**31.6%**	**33.2%**	**33.1%**	**31.7%**	**33.8%**	**33.2%**		**Private L2**
	25.2%	**24.9%**	**25.1%**	**26.0%**	**25.3%**	**23.8%**	**25.0%**	**25.1%**	**I**	**Private L2**
	37.5%	**38.6%**	**38.0%**	**39.4%**	**36.2%**	**37.1%**	**37.2%**	**37.7%**	**D**	
		22.2%		**21.7%**		**22.2%**		**21.2%**		**Paired Shared L2**

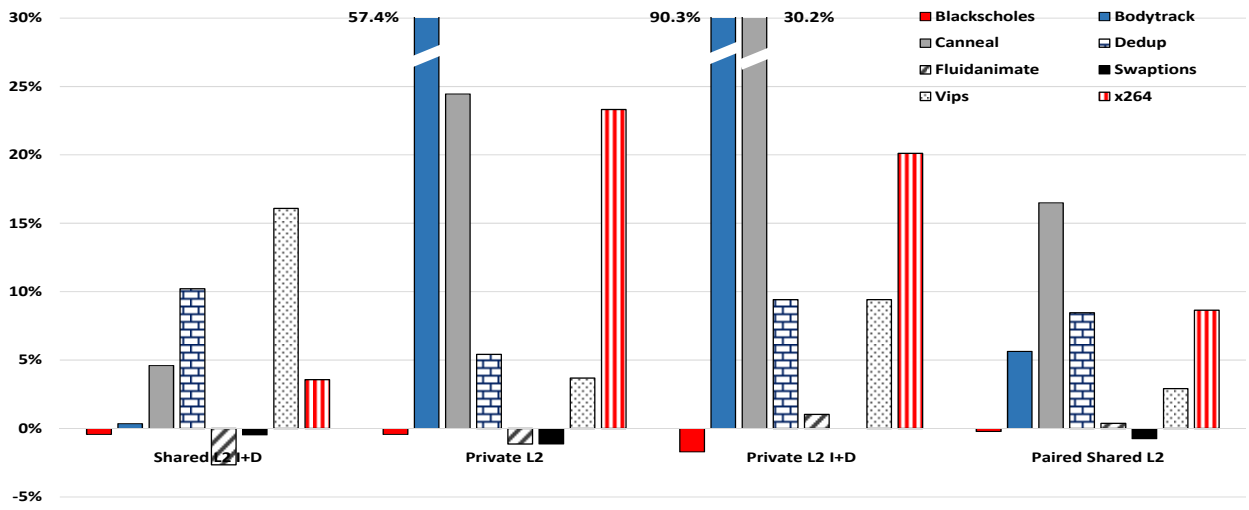


Figure 3. Experimental results for the execution time of Parsec applications targeting four cache architectures.

Using these three applications, we have a representative set to show the effects of cache organization. They represent, respectively: a simple application with scarce communication; an application with huge synchronization barriers that limits its achievable speedup and; high-parallel application.

For the Blackscholes application, the baseline has a 4.04% miss rate while the miss rate of the shared LLC I+D is 0.18% and 10.87% for instructions and data, respectively. These miss rates imply that the instruction cache size is overestimated for this scenario. Both private LLC and paired shared LLC have an increase of miss rates for the majority of cores.

From the configuration values and the parallel nature of the application set, it is expected that LLCs that share data will have a distinct benefit. Bienia et al. [17] have shown that PARSEC applications have at least 10% of share data and can reach up to 50% on a 4MiB LLC (including false sharing).

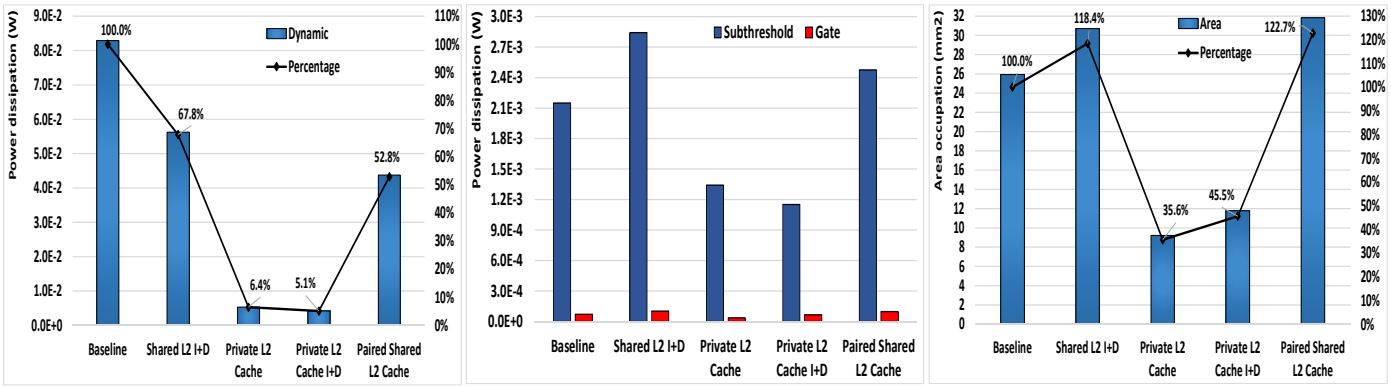
The private LLC I+D cache organization shows a remarkable phenomenon that affects all applications analyzed in this work, since instructions miss surpasses the 40% rate. Even though we set PARSEC parameters the intent of using an equal number of threads and cores, it internally uses additional threads [17]. In this case, Blackscholes employ one additional thread, which cannot be pinned to a single core, and must hop around the available cores thrashing the cache data with its working set. This thrashing affects the performance of private LLC for parallel applications negatively. In the particular case of Blackscholes, the execution time is not affected even with exorbitant L2 miss rates because Blackscholes is the simplest applications of the PARSEC suite.

Bodytrack LLC miss rates are similar to Blackscholes. Once again, the baseline cache has a low LLC miss rate (i.e., 1.03%). The shared L2 data cache has approximately the same miss rate as the baseline while the instruction cache has near zero miss rate (0.15%). Conversely, the private LLC I+D has thrashing of data information instead of instructions, as was the case of Blackscholes.

Bodytrack has a significant amount of data sharing. Luo et al. [11] showed that Bodytrack can share up to 10% of the cache size across all cores – in the case studied by them, this means that 10% of data is shared by eight cores, which is the highest sharing encountered when seven applications of PARSEC were compared. For privately accessed caches, this phenomenon can degrade performance considerably. The use of an MSI-like coherence protocol (as it is done in our system) can result in two effects: copy & invalidate operation on a store instruction (i.e., one cache has its line invalidated and another gains the exclusive/owned access to that line) and a "ping-pong" effect of two caches writing to the same cache line. Both events increase latency to exchange cache lines. An additional execution with increased data size (64KiB to 128KiB) and decreased instruction size (64KiB to 16KiB) was done to assert the effect of data sharing in this application. The average data miss was impacted significantly – falling from 61% to 28%.

The Bodytrack results from Figure 3 depict a different scenario than the one presented in the Blackscholes results. The L2 miss rate increases and the data sharing discussed earlier rises the execution time significantly. The private LLC organization showed 58% and 90% raise for the unified and I+D organizations, respectively. The shared LLC I+D shows approximately the same execution time as the baseline cache – corroborated by similar miss rates. The paired shared LLC organization increases 0.5% the execution time and 2 percent points for miss rate.

The x264 application has an interesting characteristic: it creates a vast number of threads – for the *simlarge* input, 257 threads are created during the entire execution. The baseline cache has 12.75% of miss rate. The Shared LLC I+D has the lowest instruction miss rate of all applications (0.03%) but increases the data miss rate to 25.54%. As in the case of the Blackscholes, yielding up the instruction size for data size would benefit the execution time. The private LLC I+D organization again shows high miss rate for both instruction and data, which are in average 25.04% and 37.69%, respectively. The private LLC organization presents a miss rate between the instruction and data caches of the private LLC I+D organization. Finally, the paired shared LLC shows a better miss rate than the shared L2 D (i.e., 21.83%, in average).



(a) (b) (c)

Figure 4. Power dissipation and area consumption results of five cache architectures.

The shared LLC I+D and paired shared LLC have an increase of 4% and 9% of execution time from the baseline cache, respectively. Private LLC I+D and private LLC have an increase of 23% and 20% of execution time from the baseline cache, respectively. These results match the findings of increased LLC miss rate.

The remaining applications of Figure 3 had the following results. Canneal and Vips had more than 10% of performance variance. For Canneal, private LLC organization had the worst execution time; i.e., 24% and 30% of increase when comparing the unified and I+D schemes with the baseline, respectively. The paired shared LLC organization also surpass the 10% mark, increasing performance by 16% compared to the baseline. For the Vips application, only the shared LLC I+D exceeded the 10% mark – also increasing performance by 16% compared to the baseline. The variance of all other cache organizations is under 10%. Three cache organizations are approximately at the 10% mark for Dedup application – shared LLC I+D (10%), LLC private I+D (9%), and paired shared LLC (8%). The L2 private designs showed the best execution time of the new cache designs, albeit it is still worse compared to the baseline.

B. Power Dissipation and Area Consumption

This section shows the dynamic and static power dissipation, and area consumption results regarding a 32 nm CMOS technology, which is characterized by the McPAT framework. The dynamic power dissipation covers the power dissipated for readings and writings accesses; whereas the static power dissipation comprises two types of static power dissipated due to circuit leakage – subthreshold and gate.

Figure 4(a) and (b) illustrate the dynamic and static power dissipation of five cache organizations using the parameters established in Table 1, respectively. The comparison is made summarizing all caches available. For instance, for the private LLC, the subthreshold and gate leakage of eight caches are summed up. Note that all caches have the same 1MiB data size restriction when summed up.

Figure 4(a) depicts that the baseline dissipates more dynamic power than all other cache organizations; e.g., the privative cache organizations dissipate almost 20 times less dynamic power than the baseline. This low power dissipation is an exciting aspect in contraposition of the notorious reduction of performance when using the baseline cache organization. Besides, Figure 4(b) shows that only the private cache organization dissipates less static power than the baseline cache architecture. The baseline cache dissipates 60% and 100% more subthreshold and gate leakage power than the private LLC organization, respectively. However, this changes to 87% more and 5% less subthreshold and gate leakage power than the private LLC I+D organization, respectively. Additionally, the remaining two cache organizations shared LLC I+D and paired shared LLC, dissipates more static power than the baseline for both cases.

Figure 4(c) depicts the area consumption of the entire LLC system. The attained values for individual caches are 25.92 mm² for the baseline cache design; 15.35 mm² for one shared LLC I+D; 1.15 mm² for one private LLC; 0.73 mm² for one private LLC I+D; and 7.95 mm² for one paired shared cache. These values are within the expected range because, considering a rough assumption of halving the cache size results in halving the area consumption, the results have the following disparity: 2.38 mm² (+18%), -2.08mm² (-65%), -0.88 mm² (-55%), 1.47mm² (22%). Clearly, many extra factors are influencing the attained area, such as floor planning design, MSHR queue size, write-back buffer size, and so on.

The experiments show that only the private cache organization diminish the area consumption when compared to the baseline. Both private caches consume under 50% of the baseline area. The fact that shared LLC I+D consumes more area is not surprising since it essentially doubles the amount of logic required to control the same cache size. The paired shared design needs four caches, and each one has additional area requirements described earlier. The result is that this cache organization consumes 22% more area than the baseline.

VI. EXPERIMENTAL FINDINGS

In general, the baseline LLC architecture produces the smaller execution times for the evaluated benchmark, as the workload comprises parallel applications intended to share data across tasks. The private cache organizations reach higher execution time due to the small cache sizes and the increase of coherence overhead of multiple LLCs. The additional shared designs is an intermediate solution increasing the execution time up to 16%.

Although the private cache designs degrade the performance of the system, they dissipate less static power than all shared designs. Hence, private caches are much smaller and dissipate less static power. Of all shared designs, the traditional shared LLC showed the minimum static power dissipation.

The baseline cache organization is the architecture that far more dissipates dynamic power which was fifteen and nineteen times the dynamic power dissipation of the set of private LLC and private LLC I+D organizations, respectively. A similar trend occurs with the area occupation where they have decreased twenty-two and thirty-five times compared to the baseline for the same cache set. When the shared organizations were compared to the baseline, they showed a drawback (an increase of area occupation) and a benefit (decrease of power dissipation). The set of shared LLC I+D dissipates two-thirds of the dynamic power and occupy six-fifths of the area compared to the baseline, approximately. The set of paired shared L2 has half of the power dissipation and six-fifths of the area occupation compared to the baseline, approximately.

The paired shared LLC did not achieve the middle ground we expected between performance and power dissipation. Although it had lower dynamic power dissipation, its performance was on par with the shared LLC I+D, and we expected better than this. However, it is important to note that refined mapping techniques can change this scenario.

VII. CONCLUSIONS

This work presents an analysis of execution time, power dissipation and area consumption of five cache organizations on real scenarios. We have shown that the cache organizations provide compelling tradeoffs for these requirements allowing the designer to employ a design that suits his needs. For the lowest energy consumption, we recommend employing privately small LLC. In the exact opposite of this scenario; i.e., highest performance, we recommend applying large shared LLCs. The additional shared L2 organization explored in this work are the middle ground of these scenarios and can be refined further by cache-aware mapping techniques. The cache organizations employed in this work are available as Gem5 patches on this link: *http://reviews.gem5.org/r/3506/*.

For future work, we intend to expand this study for Non-Uniform Cache Architectures (NUCA) LLC and some cache-coherence protocol schemes.

ACKNOWLEDGMENT

This work is partially funded by CNPq-Brasil (process number 132778/2014-9).

REFERENCES

[1] A. Asaduzzaman; F. Sibai; M. Rani. **Impact of Level-2 Cache Sharing on the Performance and Power Requirements of Homogeneous Multicore Embedded Systems**. *Microprocessors and Microsystems*, vol. 33, Issues 5-6, pp. 388-397, Aug. 2009.

[2] H. Esmaeilzadeh et al. **Dark Silicon and the End of Multicore Scaling**. *International Symposium on Computer Architecture (ISCA)*, pp. 365-376, 2011.

[3] M. Hajkazemi; M. Tavana; H. Homayoun. **Wide I/O or LPDDR? Exploration and Analysis of Performance, Power and Temperature Trade-offs of Emerging DRAM Technologies in Embedded MPSoCs**. *International Conference on Computer Design (ICCD)*, pp. 62-69, 2015.

[4] M. Sabry; M. Ruggiero; P. Valle. **Performance and Energy Trade-Offs Analysis of L2 on-chip Cache Architectures for Embedded MPSoCs**. *Great Lake Symposium on VLSI (GLVLSI)*, pp. 305-310, 2010,

[5] H. Yun; P. Valsan. **Evaluating the Isolation Effect of Cache Partitioning on COTS Multicore Platforms**. *Workshop on Operating Systems Platforms for Embedded Real-time Applications (OSPERT)*, pp. 45-50, 2015.

[6] PARSEC team. **The PARSEC Benchmark suite**. parsec.cs.princeton.edu/.

[7] McPAT research team. **McPAT**. www.hpl.hp.com/research/mcpat/.

[8] Altera Corporation. **Meeting the Low Power Imperative at 28 nm**. *Whitepaper*, pp. 1-12, Sep. 2012.

[9] H.-Y. Cheng et al. **EECache: A Comprehensive Study on the Architectural Design for Energy-Efficient Last-Level Caches in Chip Multiprocessors**. *ACM Transactions on Architecture and Code Optimization (TACO),* vol. 12, Issue 2, pp. 1-22, Jul. 2015.

[10] SureCore Technology. **Technology Whitepaper**. *White Papers Repository*, pp 1-6, 2013.

[11] H. Luo; P. Li; C. Ding. **Parallel Data Sharing in Cache: Theory, Measurement, and Analysis**. *Technical Report TR-994*, pp. 1-25, Mar. 2015.

[12] U. Wiener. **Modeling and Analysis of a Cache Coherent Interconnect**. *Thesis Report*, *Eindhoven University of Technology*, pp. 1-83, Aug. 2012.

[13] R. Sivaramakrishnan; S. Jairath. **Next Generation SPARC Processor Cache Hierarchy**. *Presentation at Hot Chips (HC)*, pp. 1-28, 2014.

[14] D. Woo; N. Seong; D. Lewis; H.-H. Lee. **An Optimized 3D-Stacked Memory Architecture by Exploiting Excessive, High-Density TSV Bandwidth**. *International Symposium on High Performance Computer Architecture (HPCA)*, pp 1-12, 2010.

[15] ARM. **Cortex A-15**. *Technical Reference Manual*, pp. 1-364, 2011.

[16] N. Binkert et al. **The gem5 Simulator**. *ACM SIGARCH Computer Architecture News*, vol. 39, Issue 2, pp. 1-7, May 2011.

[17] C. Bienia et al. **The PARSEC Benchmark Suite: Characterization and Architectural Implications**. *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72-81, 2008.

[18] G. Southern; J. Renau. **Deconstructing PARSEC Scalability**. *Annual Workshop on Duplicating, Deconstructing and Debunking of International Symposium on Computer Architecture (ISCAWDDD)*, pp. 1-10, 2015.