Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

HIGH LATENCY AND CONTENTION ON SHARED L2-CACHE FOR MANY-CORE ARCHITECTURES

MARCO A. Z. ALVES[‡], HENRIQUE C. FREITAS[†] and PHILIPPE O. A. NAVAUX[‡]

[‡]*Institute of Informatics, Universidade Federal do Rio Grande do Sul (UFRGS), Av. Bento Gonçalves, 9500, Porto Alegre, Rio Grande do Sul, 91501-970, Brazil.*

[†]*Institute of Informatics, Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Av. Dom José Gaspar, 500, Belo Horizonte, Minas Gerais, 30535-901, Brazil.*

Received October 2009
Revised April 2010
Communicated by J. Dongarra

ABSTRACT

Several studies point out the benefits of a shared L2 cache, but some other properties of shared caches must be considered to lead to a thorough understanding of all chip multiprocessor (CMP) bottlenecks. Our paper evaluates and explains shared cache bottlenecks, which are very important considering the rise of many-core processors. The results of our simulations with 32 cores show low performance when L2 cache memory is shared between 2 or 4 cores. In these two cases, the increase of L2 cache latency and contention are the main causes responsible for the increase of execution time.

Keywords: Shared cache memory, Many-core processor, Performance evaluation

1. Introduction and Motivation

Parallel processing in current multi-core processors uses the cache memory in order to achieve high performance. Although shared cache memory is a good approach for intra-chip communication (shared variables), for many-core processors there are another alternative based on networks-on-chip (message passing). Comparing these two types of communication, the major issue is the performance of each approach. In this paper, the main questions are the following: When is the shared cache memory a bottleneck? What is the performance impact when more than one core shares the same cache memory?

Commercial multi-core processor design points out to shared L2 and L3 caches [18], [11]. In addition, these processors present the on-chip interconnection as an important mechanism to access the cache. However, these current processors have a small number of processing cores, i.e., two, four and eight cores.

In accordance with the advantages to program a multi-core architecture based on shared memory [5], and the doubts about the performance of the intra-chip

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

message-passing model through a Network-on-Chip (NoC) [6], several researchers suggest the use of shared cache memory as the alternative for a large number of cores, since all cores need to access the same physical memory [14], [16]. Moreover, the NoC [4][7] for a large number of cores is still under evaluation for the next generation of processors.

Even with the related work as presented in the next section, and other studies that describe the importance of a good organization of cache memory to achieve high performance, there is no consensus. However, it is important to notice that the related studies does neither address the high latency of shared cache memory for a large number of cores, nor models the contention due to the restricted number of ports on the shared cache; both have a high impact on the final performance. Hence, the main motivation of this paper is to evaluate shared cache memory by taking into account the latency and contention.

Therefore, the main goal of this paper is to evaluate different memory organizations, varying the number of cores sharing the cache, for a large number of processing cores, in order to identify bottlenecks and to measure the impact on the performance. In addition, there are contributions to the methodology and to the simulation based on cache features that have not been widely discussed in related work: the memory latency, which depends on the memory size and the number of ports; and the contention due to the restricted number of cache access ports.

This paper is organized into the following sections: Related Work, Workload, Contention Modeling, Simulation Setup Evaluation, Simulation Proposal, Experimental Results, Conclusions and Future Work, and Acknowledgement.

2. Related Work

Related studies [16], [15], [14], [19] evaluated the cache performance of systems with a large number of processing cores by executing multiple threads at the same time. Marino [14] used a system consisting of 32 cores, varying the number of cores sharing the L2 cache between one, two and four. The results showed better performance for all benchmarks when increasing the number of cores sharing the same L2 cache. On the other hand, the work of Zahran [19] compared private and shared L1 and L2 cache memories and came to the conclusion that private caches result in faster performance.

Our previous work [3] compared a private L2 cache with a L2 cache shared between two cores, taking into account that the shared cache was twice as big as the private cache, resulting in an increased cache latency. The results show that irregular data access applications perform faster with the shared cache, even with the increase of the cache latency. Moreover, the results showed that it is possible to share the cache without performance losses, by increasing the L2 cache line size. However, the weakness of this previous work is that memory port contention was not modeled and just one and two cores per cache were evaluated.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

3. Workload

In order to evaluate the cache of our simulated many-core processor, we used a workload based on a subset of NAS (Numerical Aerodynamic Simulation) Parallel Benchmark (NPB), version 3.3 [10], using OpenMP implementations to distribute the workload across 32 threads. A small input data set (the *W* size) of the workload was compiled with the Sun Studio C and Fortran compilers with the *-fast* optimization. For the simulation, the following benchmarks from the NPB suite were used:

- BT.W - To solve 3D compressible Navier-Stokes equations with an implicit algorithm. Based on Alternating Direction Implicit (ADI) finite differences solver where the resulting system is Block-Tridiagonal, which is solved sequentially along each dimension. 2.7 MB
- CG.W - Conjugate Gradient method used to compute the smallest eigenvalue of a large, sparse, unstructured matrix. Exercising unstructured grid computations and communications. 13.7 MB
- MG.W - Multigrid V-cycle method used to solve the 3D scalar Poisson equation. The algorithm works between coarse and fine grids. It exercises both short and long distance data movement. 55.7 MB
- EP.W - Embarrassingly Parallel benchmark, which generates pairs of Gaussian random. Aiming to establish the reference point for peak performance of a given platform. 1.3 MB
- SP.W - Computational Fluid Dynamics (CFD) application similar to BT. The problem is based on a Beam-Warming approximate factorization that decouples in 3D. The resulting Scalar Pentadiagonal system is solved sequentially along each dimension. 8.7 MB
- LU.W - Simulated CFD application that uses symmetric successive over-relaxation (SSOR) method based on finite-difference discretization of Navier-Stokes equations in 3D by splitting into block Lower and Upper triangular systems. 6.6 MB
- IS.W - Test Integer Sort operation that is important in particle method codes. This code exercises integer and communication performance. 3.4 MB
- FT.W - Computational kernel of a 3D Fast Fourier Transform (FFT) method. FT performs three 1D FFT, one for each dimension. 20 MB
- UA.W - Unstructured Adaptive benchmark, which exercises irregular and continually memory accesses measuring its effect. 16.3 MB

4. Contention Modeling

Even with the ability of full-system simulation and detailed parameterization, the Simics simulator does not model the contention caused by a restricted number of access ports in any memory level. Thus, the following analytical equations were used to model the port contention of the L2 cache in order to evaluate the experiments.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

In the following formulas, j is the number of available L2 caches, k is the number of L1 caches connected to each L2 cache, p is the number of ports. The latency of the crossbar, which connects the L1 to the L2 cache is L_{Xbar1}, the latency of the crossbar which connects the L2 to the main memory is L_{Xbar2}. The latency to access the L1 cache, L2 cache and main memory are L_{L1}, L_{L2} and L_{RAM}, respectively.

The basic formula for modeling the total execution time (T_{Total}) is given in Equation 1.

$$T_{Total} = T_{Max\_Inst} + T_{L2\_Write\_Back} + T_{L2\_Misses} + T_{L1\_Hits} + T^{p}_{L1\_Misses} \quad (1)$$

The overall time spent to execute the instructions (T_{Max_Inst}) is defined by the processor which executed the biggest number of instructions, as described in Equation 2.

$$T_{Max\_Inst} = Max\left\{\bigcup_{i=1}^{32} P^{i}_{Instructions}\right\} \quad (2)$$

The time required to execute the write-back operation for all the L2 caches ($T_{L2\_Write\_Back}$) appears in Equation 3.

$$T_{L2\_Write\_Back} = \sum_{i=1}^{j} L2^{i}_{Write\_Back} \cdot (L_{Xbar2} + L_{RAM}) \quad (3)$$

The time spent to access the main memory during L2 cache misses (T_{L2_Misses}) is described in Equation 4.

$$T_{L2\_Misses} = \sum_{i=1}^{j} L2^{i}_{Misses} \cdot (L_{Xbar2} + L_{RAM}) \quad (4)$$

The time consumed by the L1 cache hits (T_{L1_Hits}) is presented in Equation 5.

$$T_{L1\_Hits} = Max\left\{\bigcup_{i=1}^{32} L1^{i}_{Hits}\right\} \cdot (L_{L1}) \quad (5)$$

To calculate the total time spent during the L1 cache misses ($T^{p}_{L1\_Misses}$), it is necessary to consider the amount of ports on L2 cache. Thus, three formulas were developed, with 1 port ($T^{1}_{L1\_Misses}$), 2 ports ($T^{2}_{L1\_Misses}$) and 4 ports ($T^{4}_{L1\_Misses}$) in order to model the organizations evaluated in the experiments.

The Equation 6 calculates the time spent during L1 cache misses ($T^{1}_{L1\_Misses}$) for the L2 cache with 1 port.

$$T^{1}_{L1\_Misses} = Max\left\{\sum_{i=1}^{k} L1^{i}_{Misses}; \sum_{i=k+1}^{2k} L1^{i}_{Misses}; \ldots; \sum_{i=32-k}^{32} L1^{i}_{Misses}\right\} \cdot (L_{L1} + L_{XBar1} + L_{L2}) \quad (6)$$

The Equation 7 represents the time spent during L1 cache misses, considering each L2 cache with 2 ports ($T^{2}_{L1\_Misses}$). The equation uses the function $F^{n}(L1^{Begin}, L1^{End})$ that returns the n-th higher value of data misses among the L1 caches ($L1^{Begin}$) and ($L1^{End}$). For example, the function $F^{2}_{Misses}(L1^{1}, L1^{4})$ must return the second higher number of misses among the first four L1 caches.

Considering a L2 cache with 2 access ports, it would not be fair to consider the 2 highest numbers of L1 cache misses on the same port. Thus, for each L2 cache,

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

we assigned the L1 caches with the highest and lowest number of L1 cache misses to one port, and the other two L1 caches to the other port. Then, after obtaining the number of accesses to each port of the L2 cache, we calculate the time spent during L1 cache misses using the port with the highest number of cache misses.

$$T^2_{L1\_Misses} = Max \left\{ \begin{array}{c} F^1(\bigcup_{i=1}^{k} L1^i_{Misses}) + F^4(\bigcup_{i=1}^{k} L1^i_{Misses}); \\ F^2(\bigcup_{i=1}^{k} L1^i_{Misses}) + F^3(\bigcup_{i=1}^{k} L1^i_{Misses}); \\ F^1(\bigcup_{i=k+1}^{2k} L1^i_{Misses}) + F^4(\bigcup_{i=k+1}^{2k} L1^i_{Misses}); \\ F^2(\bigcup_{i=k+1}^{2k} L1^i_{Misses}) + F^3(\bigcup_{i=k+1}^{2k} L1^i_{Misses}); \\ \vdots \\ F^1(\bigcup_{i=32-k}^{32} L1^i_{Misses}) + F^4(\bigcup_{i=32-k}^{32} L1^i_{Misses}); \\ F^2(\bigcup_{i=32-k}^{32} L1^i_{Misses}) + F^3(\bigcup_{i=32-k}^{32} L1^i_{Misses}); \end{array} \right\} \cdot (L_{L1} + L_{XBar1} + L_{L2}) \quad (7)$$

The Equation 8 calculates the time spent during L1 cache misses ($T^4_{L1\_Misses}$) for the L2 cache with 4 ports.

$$T^4_{L1\_Misses} = Max \left\{ \bigcup_{i=1}^{32} L1^i_{Misses} \right\} \cdot (L_{L1} + L_{XBar1} + L_{L2}) \quad (8)$$

5. Simulation Setup Evaluation

According to Gibson, et al. [8] a simulation model needs to be validated against real machines in order to ensure reliable results for hypothetical architectures. Although Simics is widely used by the community, this section shows a comparison between a real machine and simulation results validating the simulation setup for the next sections.

The rest of the paper, simulates next generations of many-core processors with 32 cores which prototypes are not available for evaluation. Thus, in this section a simulation modeling an 8 core processor is compared against a real machine with 2 processors Intel Harpertown, which is a quad-core processor. The general cache parameters were chosen based on the this Intel Harpertown, which also represents the cache configuration of a wide range of processors. A list of fixed parameters of the real machine and the values modeled in the experiments is shown in Table 1.

Comparing the simulation parameters and the real machine configuration, besides the not available information about the real processor, it is notable that some simulation parameters are different from the real machine, as the pipeline, out-of-order execution, instruction set architecture and others parameters varies. But in this paper, we do not intent to validate the core microarchitecture, just to demonstrate that our simulation results are comparable to a real machine execution, meaning that our simulation results fits with the reality and correspond to the same behavior of a real machine.

For our comparisons the NAS parallel benchmark parallelized using OpenMP were executed on both, real machine and simulation. For all the 9 applications of the NPB suite, 5 executions were made varying between 1, 4, and 8 threads. The problem size and compilation parameters for the workload are the same used on the rest of the paper. In order to compare the impact of our formulas presented on section 4, the results present the simple simulation results (without analytical

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

Table 1. Real machine and simulation parameters.

Element	Parameter	Real Machine	Simulation Model
Processing	Non-memory IPC	Varies	1.0
Core	Pipeline	Yes	Not modeled
	Out-Of-Order	Yes	Not modeled
	Processors	2x Xeon 5405	1x UltraSparc III+
	Cores	2x 4 cores	8 cores
	Instruction set	Intel x86_64	Sparc V9
	Frequency	2 GHz	2 GHz
	Integration technology	45 nm	45 nm
Interconnection	Type	N/A	Cross-Bar
	Latency	N/A	2 Cycles
L1 Cache	Replace and write policy	N/A	LRU with write-through
Memory	Integration technology	45 nm	45 nm
	Memory size (Instruction)	32 KB	32 KB
	Memory size (Data)	32 KB	32 KB
	R/W data latency	N/A	1.29 ns = 3 cycles
	Associativity	8-way set associative	8-way set associative
	Line size	64 Bytes	64 Bytes
L2 Cache	Organization	1 L2 shared by 2 cores	1 L2 shared by 2 cores
Memory	Replace and write policy	N/A	LRU with write-back
	Integration technology	45 nm	45 nm
	Memory size	6 MB	6 MB
	R/W latency 1 Port	N/A	5.43 ns = 11 cycles
	R/W latency 2 Ports	N/A	8.47 ns = 17 cycles
	Associativity	24-way set associative	24-way set associative
	Line size	64 Bytes	64 Bytes
	Data coherency protocol	N/A	MESI based on snooping
Main	Size	8 GBytes	1 GBytes
Memory	R/W latency	N/A	38 ns = 78 cycles
	Integration technology	65 nm	65 nm

formulas), and the estimates considering the L2 cache with 1 and 2 R/W access ports (1P and 2P respectively).

The first results comparing the real machine to the system simulated with an 8 core processor is presented in the Figure 1 showing the execution time for real and simulation executions for all the application running sequentially. Figure 2 presents the execution time running 4 threads. The execution time for 8 threads is presented in Figure 3.

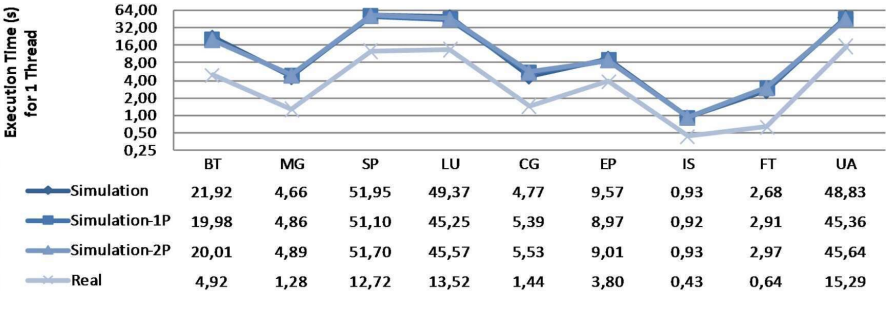
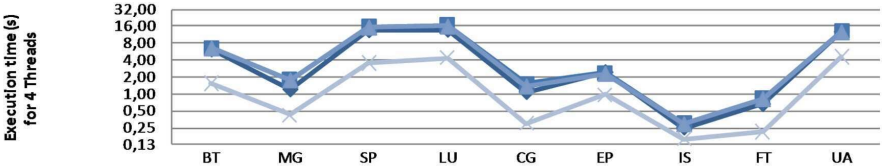


Fig. 1. Execution time for 1 thread running on real machine and simulated.

Electronic version of an article published in Parallel Processing Letters

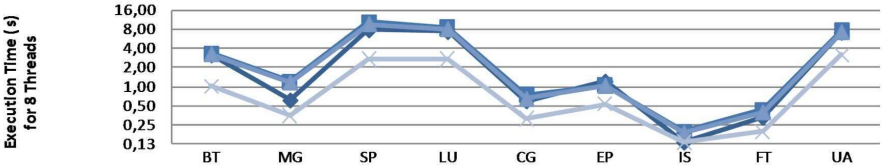
v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096



	BT	MG	SP	LU	CG	EP	IS	FT	UA
Simulation	6,14	1,19	13,49	13,78	1,07	2,41	0,25	0,65	12,70
Simulation-1P	6,64	1,80	15,50	16,48	1,51	2,34	0,30	0,83	13,01
Simulation-2P	6,58	1,78	15,46	16,08	1,39	2,33	0,30	0,83	12,69
Real	1,55	0,42	3,56	4,39	0,30	0,97	0,16	0,21	4,55

Fig. 2. Execution time for 4 threads running on real machine and simulated.



	BT	MG	SP	LU	CG	EP	IS	FT	UA
Simulation	3,17	0,62	7,89	7,41	0,60	1,22	0,13	0,33	7,20
Simulation-1P	3,36	1,20	10,36	8,53	0,75	1,06	0,20	0,44	7,51
Simulation-2P	3,29	1,18	9,73	8,12	0,65	1,05	0,19	0,40	7,29
Real	1,02	0,36	2,74	2,74	0,32	0,54	0,14	0,20	3,14

Fig. 3. Execution time for 8 threads running on real machine and simulated.

Results show that simulation overestimates the execution time from 2.5 to 3.6 times more than the real machine execution. But, as the simulations with 1P and 2P keep the factor of overestimation, about 4.5 times slower reducing the variation in 30%, for all the applications, we can conclude that our formulas improved the results, making the applications results perform with similar trends.

The difference of the execution time can be explained by the differences between the real machine configuration and simulation parameters. The processor simple model simulated on Simics, as cited on the beginning, does not simulate pipeline, superscalarity and out-of-order execution which would improve the performance results. Furthermore, the real machine uses the Intel x86_64 ISA while the simulator model the Sun Sparc V9 ISA, which shall lead to different results from the real machine. This ISA difference can lead to an important question about the choice of the Sun Sparc processor on our simulations, an the answer is that Simics provides more flexibility using Sun Sparc architecture, supporting a larger number of cores on chip-multiprocessor models with this ISA.

This simulation setup evaluation is important to check the simulation behavior results [8] [12], even with this simplified processor model simulated, the execution time comparison between the simulation and the real machine shows that our simulation results correspond to the same real behavior, which indicates that the relevant

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

performance effects on the simulation are the same as those on the real machine, what is easily seen on the results plot. Moreover, the execution time results can be improved with the contention model enhancing the execution time trends.

6. Simulation Proposal

The evaluation of computer systems [9] can be done with three different techniques: analytical modeling, simulation, or measurement. Measurements can only be done if a similar system is implemented. As prototypes based on 32 cores processor with shared cache memories are not available for evaluation, just the analytical model and the simulation are alternatives. Thus, in this paper simulation and analytical models were used to evaluate and compare characteristics of cache organization.

Virtutech Simics [13] is a full-system simulator at the instruction set level which measures execution time based on the number of instructions executed and the number of stall cycles caused by the latency of all components.

Table 2. Modeled simulation parameters.

Processing Core	Non-memory IPC	1.0
	Pipeline	Not modeled
	Cores	32 cores - UltraSparc III+
	Instruction set	Sparc V9
	Frequency	2 GHz
	Integration technology	45 nm
Interconnection	Type	Cross-Bar
	Latency	2 Cycles
L1 Cache Memory	Replace and write policy	LRU with write-through
	Integration technology	45 nm
	Memory size	32 KB (instruction) + 32 KB (data)
	R/W instruction latency	0 ns = 0 cycles
	R/W data latency	0.74 ns = 2 cycles
	Associativity	2-way set associative
	Line size	32 Bytes
	Dynamic energy	0.038 nJ
	Static power	0.024 W
	Occupied area	$0.228mm^2$
L2 Cache Memory	Data mapping mode	Set associative
	Replace and write policy	LRU with write-back
	Integration technology	45 nm
	R/W Latency	Modeled using Cacti
	Data coherency protocol	MESI based on snooping
Main Memory	Size	1 GByte
	R/W latency	38 ns = 78 cycles
	Integration technology	65 nm
	Dynamic energy	21.125 nJ
	Static power	0.091 W
	Occupied area	$739.540mm^2$

A 32-core processor was modeled based on the UltraSparc-III+ architecture, with the Solaris 10 operating system installed. Table 2 summarizes the modeled components and their parameters used on the simulations. The clock frequency chosen is not the main issue to be investigated, but it must be defined to model the penalty cycles for all the components throughout the memory hierarchy.

Electronic version of an article published in Parallel Processing Letters
v. 21, issue 1, 2011, p. 85-106
http://dx.doi.org/10.1142/S0129626411000096

Since different memory organizations were simulated, the parameters of the L2 cache memory modeled for each specific experiment are shown in the beginning of each result section. These parameters were used to calculate the memory latency for each experiment by using the CACTI [17] memory modeling tool.

7. Experimental Results

This section presents four experiments, varying the cache sharing, cache size, associativity and line size as described bellow.

- Experiment 1 - Cache sharing: Evaluating the different cache organizations while keeping the total amount of cache constant.
- Experiment 2 - Cache size: Evaluating the different cache organizations with different total amounts of cache.
- Experiment 3 - Cache associativity: Evaluating the different cache organizations with twice L2 cache associativity.
- Experiment 4 - Cache line size: Evaluating the different cache organizations with twice L2 cache line size.

For the proposed experiments, some different cache organizations were evaluated. The naming scheme adopted for the results is presented in Figure 4. It consists of the number of cores per cache and the number of communication ports of each cache. For example, the name of the organization with 4 cores per L2 cache and 2 ports is 4Cores/L2-2P.

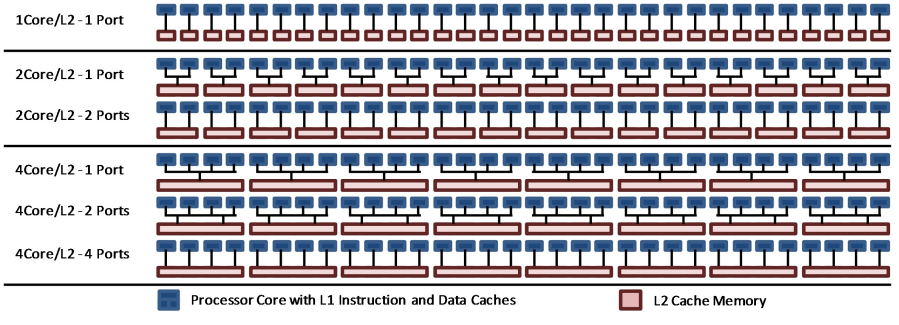


Fig. 4. Processors and cache layout diagram.

Considering that Simics is a deterministic simulator, where all executions of the same application launched from a checkpoint will return the same results, one can conclude that only one measurement leads to a plausible result. However, it is necessary to consider that the workload is not deterministic, i.e. small variations in the initial state of the execution may lead to different execution paths. Thus, a single execution of the experiment can lead to erroneous conclusions [1].

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

Each application was executed 5 times sequentially to insert non-determinism caused by interrupts and other operating system interferences. This number of repetitions was determined by following the procedure outlined in [12]. Thus, after doing a preliminary statistical analysis of the system behavior, 5 repetitions were sufficient to keep our chosen confidence interval. Moreover, before measurements were made, the workload was executed once to warm up the cache and therefore, reduce transient effects [2].

In order to show that the results are reliable, the plots show the standard deviation, calculated using a confidence interval equal to 81.30% on the Student's t-distribution. In all experiments, the workload was executed by 32 threads running in parallel.

7.1. *Experiment 1 - Cache sharing*

This first experiment aims to evaluate the influence of shared cache between multiple cores. With the increase in the shared cache memory, the number of cache misses tends to decrease, therefore, the parallel applications tend to share data faster. On the other hand, the cache organizations in this experiment keep the total cache size of the system constant. As a consequence, it increases the data access latency, power consumption and physical area occupation. Table 3 shows the values modeled for the cache memories on this first experiment.

Table 3. L2 Cache memories description for the first experiment.

Experiment 1 - Cache memory sharing						
Sharing	1Core/L2	2Cores/L2	2Cores/L2	4Cores/L2	4Cores/L2	4Cores/L2
R/W Ports	1	1	2	1	2	4
Cache Size	1 MB	2 MB	2 MB	4 MB	4 MB	4 MB
Line Size	64 B	64 B	64 B	64 B	64 B	64 B
Associativity	8-way	8-way	8-way	8-way	8-way	8-way
R/W Time	1.22 ns	1.70 ns	2.41 ns	2.17 ns	3.17 ns	5.17 ns
R/W Latency	3 Cycles	4 Cycles	5 Cycles	5 Cycles	7 Cycles	11 Cycles
Dyn. Energy	0.63 nJ	1.03 nJ	1.75 nJ	1.53 nJ	2.41 nJ	3.75 nJ
Static Power	1.50 W	3.03 W	4.15 W	6.07 W	8.39 W	14.44 W
Area	$9.87mm^2$	$20.85mm^2$	$63.78mm^2$	$42.58mm^2$	$129.85mm^2$	$386.63mm^2$

The first result is shown in Figure 5, which displays the graphic with values of execution cycles, lost cycles caused by data misses stalls in L1 cache, speedup and percentage of lost cycles due to faults in the L1 cache considering the total time for the experiment execution. Regarding this first result, we can see that the organization 1Core/L2-1P has the best performance, and also presents the least amount of lost cycles waiting for data from the L2 cache, which represent slightly more than 15% of the workload runtime.

However, comparing all the organizations evaluated, we can see that the reduced number of ports created more lost cycles than the increase of memory access latency.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

Therefore, it becomes clear that memory port contention will have a big impact on the systems' final performance.

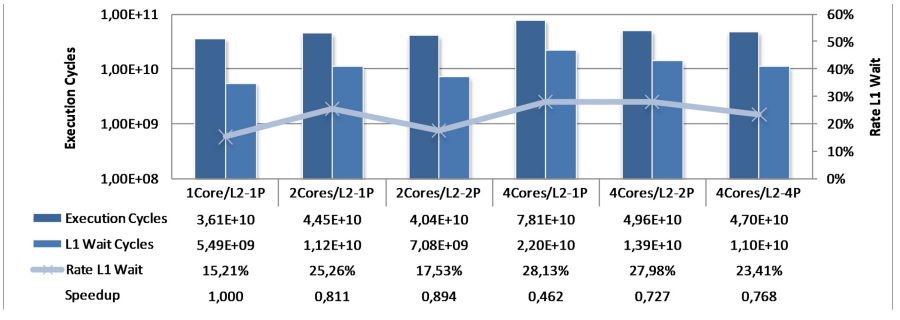


Fig. 5. Execution results for the first experiment.

The graphic presented in Figure 6 shows estimated values for energy and total power consumption of the memory system. It is possible to see that the 1Core/L2-1P, 4cores/L2-2P and 4Cores/L2-1P organizations show the lowest energy consumption. The energy consumption of the 2Cores/L2-2P and 4cores/L2-4P organizations, both having 32 ports, are the highest due to the high number of ports inside each cache bank, 2 and 4 respectively.

In terms of energy consumption, having one cache per core, or increasing the shared memory while reducing the total number of ports of the system is best.

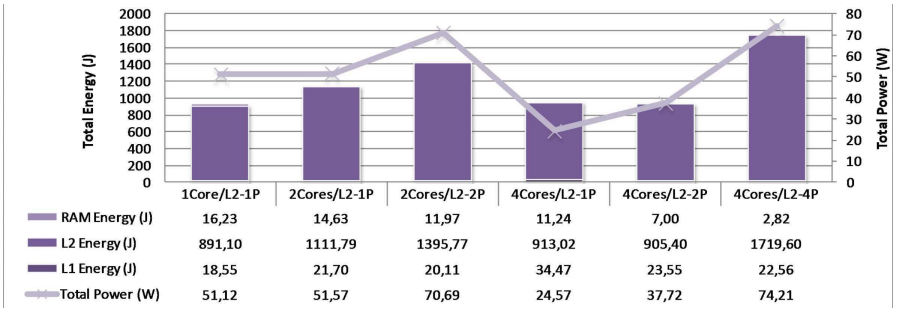


Fig. 6. Memory subsystem total energy and power consumption for the first experiment.

The area occupied by the L2 cache for the different organizations and number of ports is shown in Figure 7. We note that only organizations 1Core/L2-1P, 2Cores/L2-1P and 4Cores/L2-1P maintain a reasonable area occupation. In the other cases, the occupied area is up to three times bigger due to the increase in the number of ports per cache bank.

In terms of performance, the best organizations are 1Core/L2-1P, 2Cores/L2-2P and 2Cores/L2-1P. In terms of energy consumption, the best organizations are

Electronic version of an article published in Parallel Processing Letters
v. 21, issue 1, 2011, p. 85-106
http://dx.doi.org/10.1142/S0129626411000096

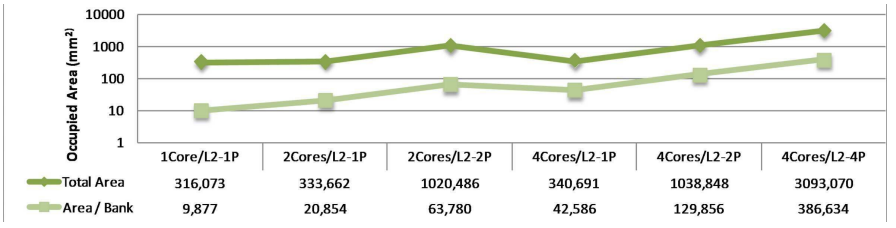


Fig. 7. Occupied area for L2 cache memory on the first experiment.

1Core/L2-1P, 4Cores/L2-2P and 4Cores/L2-1P. In terms of the occupied area, the best organizations are 1Core/L2-1P, 2Cores/L2-1P and 4Cores/L2-1P. Combining these results, the organization with the most advantages is the 1Core/L2-1P.

7.2. *Experiment 2 - Cache size*

The second experiment aims to evaluate the influence of the shared L2 cache size among multiple cores. For this experiment three cache sizes were chosen: 1, 2 and 4 MBytes. Instead of keeping the total size of the L2 cache in the system constant like in the first experiment, each L2 cache has the same size but the number of L2 caches varies with each organization. Thus, the total amount of L2 cache memory varies with each organization.

This experiment evaluates two parameters on the performance: One, increasing the number of cores sharing the L2 cache while reducing the total amount of the L2 cache; Two, increasing the total amount of L2 cache. In the first case, the system can share data faster between the cores, but the number of cache misses tends to increase. The cache latency will stay constant, except when changing the number of ports. In the second case, the number of cache misses tends to decrease while the cache latency increases.

Considering the large number of results of this experiment, this section is divided into three subsections, one for each cache size evaluated (1, 2 and 4 MBytes).

7.2.1. *L2 cache size 1 MByte*

In this section, the L2 cache was set to 1 MByte. Based on the organizations to be evaluated, Table 4 shows the values modeled for the L2 cache.

The graphic in Figure 8 presents the results for 1 MB per L2 cache. For comparison, the graphic shows the result from the first experiment (organization 1Core/L2-1P-1Exp) as well. The best organization in terms of performance is 1Core/L2-1P. 2Cores/L2-2P and 2Cores/L2-1P are the second and third, respectively.

In this experiment, the total number of stall cycles due to L1 cache misses increases as the number of cores sharing the L2 cache increases. Furthermore, increasing the contention (4Cores/L2-1P) performs worse than increasing the latency combined with increasing the number of ports (4Cores/L2-4P).

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

Table 4. L2 Cache memories description for the second experiment (1 MB per cache).

Experiment 2 - cache size 1 MB						
Sharing	1Core/L2	2Cores/L2	2Cores/L2	4Cores/L2	4Cores/L2	4Cores/L2
R/W Ports	1	1	2	1	2	4
Cache Size	1 MB	1 MB	1 MB	1 MB	1 MB	1 MB
Line Size	64 B	64 B	64 B	64 B	64 B	64 B
Associativity	8-way	8-way	8-way	8-way	8-way	8-way
R/W Time	1.22 ns	1.22 ns	1.83 ns	1.22 ns	1.83 ns	2.78 ns
R/W Latency	3 Cycles	3 Cycles	4 Cycles	3 Cycles	4 Cycles	6 Cycles
Dyn. Energy	0.63 nJ	0.63 nJ	1.01 nJ	0.63 nJ	1.01 nJ	1.84 nJ
Static Power	1.50 W	1.50 W	2.28 W	1.50 W	2.28 W	3.99 W
Area	$9.87mm^2$	$9.87mm^2$	$31.51mm^2$	$9.87mm^2$	$31.51mm^2$	$109.14mm^2$

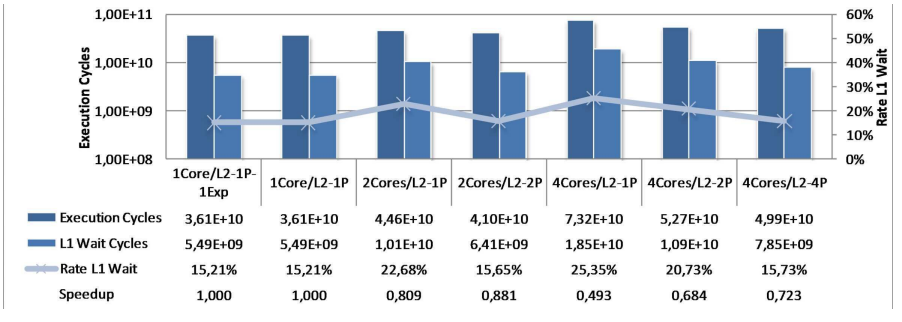


Fig. 8. Results for the second experiment running with a cache size of 1 MByte.

Figure 9 presents energy and power consumption results of the memory system modeled. The consumption depends on two factors: the total L2 cache size and the number of L2 cache ports. The lowest energy consumption is achieved with the organizations 4Cores/L2-1P, 2Cores/L2-1P and 4Cores/L2-2P. These results show that to balance the increase in energy consumption due to the increase on the number of ports, the total amount of cache memory should be reduced.

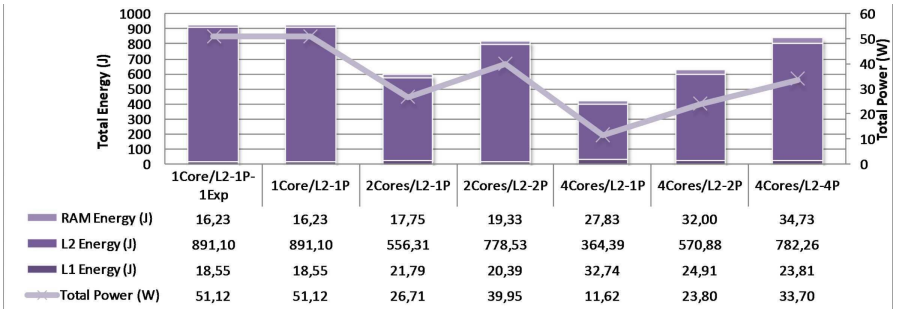


Fig. 9. Memory subsystem total energy and power consumption for the second experiment with 1 MB cache size.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

Figure 10 presents results for the L2 cache area occupation. As the power consumption results already indicated, the organizations with the smallest area occupation are 4Cores/L2-1P, 2Cores/L2-1P and 4Cores/L2-2P, emphasizing the physical impact of the amount of cache ports on the system. Thus, it is good to reduce the cache sharing from 4Cores/L2-1P to 2Cores/L2-1P, which has twice as much cache memory.

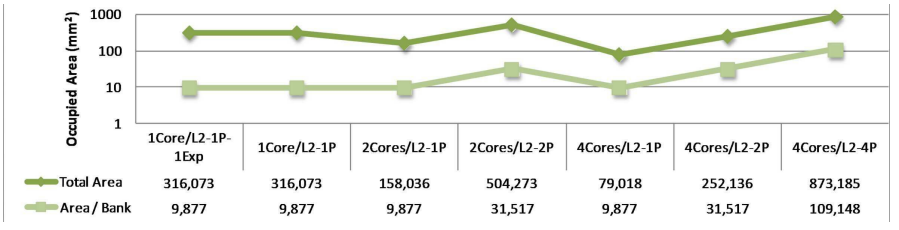


Fig. 10. Occupied area for L2 cache memory with 1 MB cache size.

7.2.2. *L2 cache size 2 MBytes*

In this section, the L2 cache was set to 2 MBytes. Based on the organizations to be evaluated, Table 5 shows the values modeled for the L2 cache.

Table 5. L2 Cache memories description for the second experiment (2 MB per cache).

Experiment 2 - Cache memory size - 2 MB						
Sharing	1Core/L2	2Cores/L2	2Cores/L2	4Cores/L2	4Cores/L2	4Cores/L2
R/W Ports	1	1	2	1	2	4
Cache Size	2 MB	2 MB	2 MB	2 MB	2 MB	2MB
Line Size	64 B	64 B	64 B	64 B	64 B	64 B
Associativity	8-way	8-way	8-way	8-way	8-way	8-way
R/W Time	1.70 ns	1.70 ns	2.41 ns	1.70 ns	2.41 ns	3.96 ns
R/W Latency	4 Cycles	4 Cycles	5 Cycles	4 Cycles	5 Cycles	8 Cycles
Dyn. Energy	1.03 nJ	1.03 nJ	1.75 nJ	1.03 nJ	1.75 nJ	2.47 nJ
Static Power	3.03 W	3.03 W	4.15 W	3.03 W	4.15 W	7.16 W
Area	$20.85mm^2$	$20.85mm^2$	$63.78mm^2$	$20.85mm^2$	$63.78mm^2$	$192.99mm^2$

The graphic in Figure 11 presents the results for 2 MB per L2 cache. The best organization in terms of performance is 1Core/L2-1P as a consequence of a balance between reducing cache misses in the L2 cache and increased latency. Note that just increasing the total cache size, without changing the cache sharing, led to an increase of the performance.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

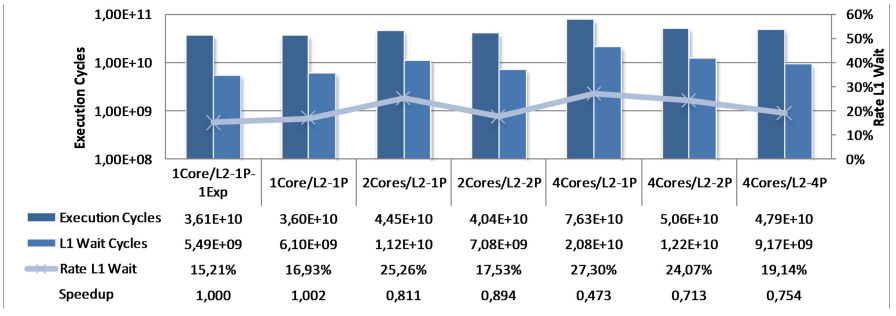


Fig. 11. Results for the second experiment running with a cache size of 2 MBytes.

The graphic in Figure 12 shows the values of energy and power consumption for this second experiment. We can note that there is an increase of energy consumption in the same organization that had a performance improvement, since the system has doubled the memory size the power consumption also is doubled, as observed by comparing the organization 1Core/L2-1P-1Exp of the first experiment.

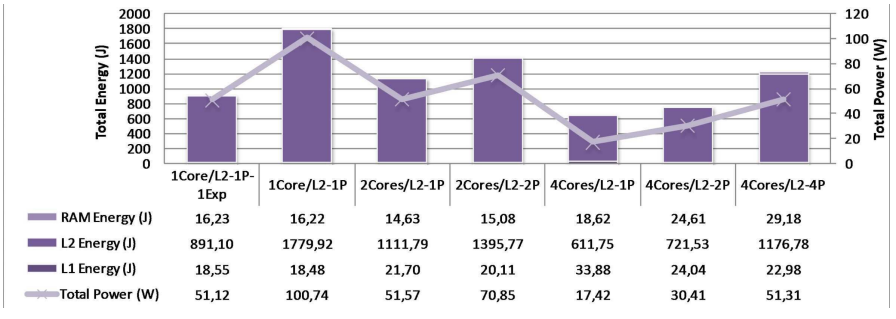


Fig. 12. Memory subsystem total energy and power consumption for the second experiment with 2 MB cache size.

The area occupation for the L2 cache memory system is shown in Figure 13. Predictably, the physical size of the caches also doubled in comparison to the first experiment.

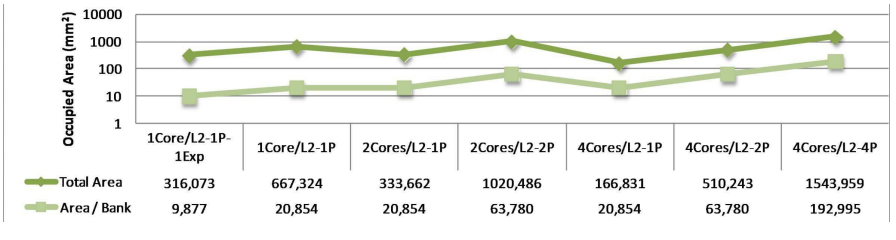


Fig. 13. Occupied area for L2 cache memory with 2 MB cache size.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

7.2.3. *L2 cache size 4 MBytes*

In this section, the L2 cache was set to 4 MBytes. Based on the organizations to be evaluated, Table 6 shows the values modeled for the L2 cache.

Table 6. L2 Cache memories description for the second experiment (4 MB per cache).

Experiment 2 - Cache memory size 4 MB						
Sharing	1Core/L2	2Cores/L2	2Cores/L2	4Cores/L2	4Cores/L2	4Cores/L2
R/W Ports	1	1	2	1	2	4
Cache Size	4 MB	4 MB	4 MB	4 MB	4 MB	4 MB
Line Size	64 B	64 B	64 B	64 B	64 B	64 B
Associativity	8-way	8-way	8-way	8-way	8-way	8-way
R/W Time	2.17 ns	2.17 ns	3.17 ns	2.17 ns	3.17 ns	5.17 ns
R/W Latency	5 Cycles	5 Cycles	7 Cycles	5 Cycles	7 Cycles	11 Cycles
Dyn. Energy	1.53 nJ	1.53 nJ	2.41 nJ	1.53 nJ	2.41 nJ	3.75 nJ
Static Power	6.07 W	6.07 W	8.39 W	6.07 W	8.39 W	14.44 W
Area	$42.58mm^2$	$42.58mm^2$	$129.85mm^2$	$42.58mm^2$	$129.85mm^2$	$386.63mm^2$

The results in Figure 14 show the balance between the cache size and the access latency. For 1Core/L2-1P, the system has 4 times more L2 cache memory in comparison to the 1Core/L2-1P-1Exp from the first experiment. However, the performance is equivalent, although there was a reduction in the rate of data misses, there was also an increase in the access latency, so performance did not increase.

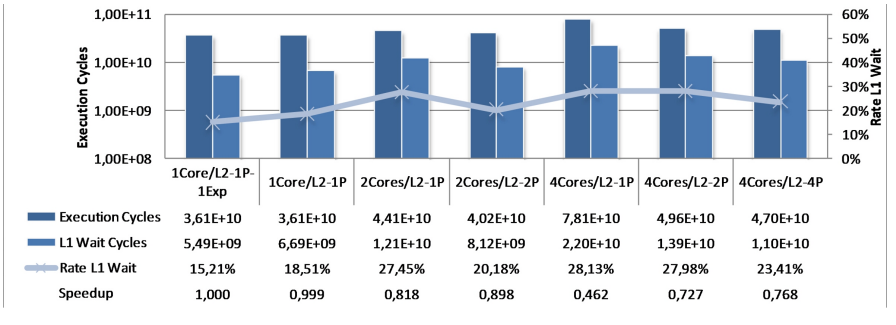


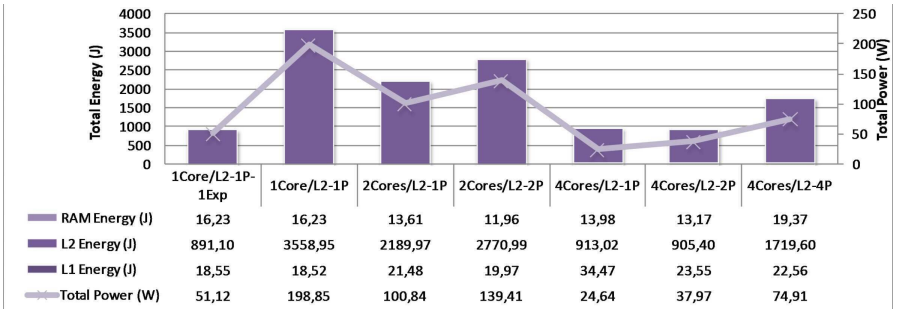
Fig. 14. Results for the second experiment running with a cache size of 4 MBytes.

The energy and power consumption is shown in Figure 15. We can see the extra cost of increasing the cache size by four, the energy consumption of the organization 1Core/L2-1P was 4 times as high as of the same organization in the first experiment.

The L2 cache area occupation illustrated in Figure 16 shows a high variation as we change the amount of L2 cache memory and the number of ports as in the two previous experiments.

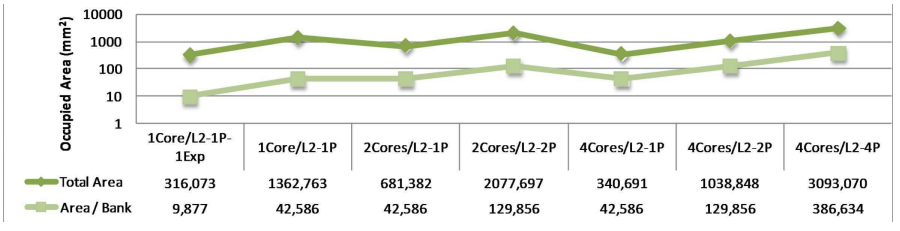
With the evaluations of this second experiment with 1, 2 and 4 MB per L2 cache, we can conclude that, the increase of memory size can increase the performance.

Electronic version of an article published in Parallel Processing Letters
v. 21, issue 1, 2011, p. 85-106
http://dx.doi.org/10.1142/S0129626411000096



	1Core/L2-1P-1Exp	1Core/L2-1P	2Cores/L2-1P	2Cores/L2-2P	4Cores/L2-1P	4Cores/L2-2P	4Cores/L2-4P
RAM Energy (J)	16,23	16,23	13,61	11,96	13,98	13,17	19,37
L2 Energy (J)	891,10	3558,95	2189,97	2770,99	913,02	905,40	1719,60
L1 Energy (J)	18,55	18,52	21,48	19,97	34,47	23,55	22,56
Total Power (W)	51,12	198,85	100,84	139,41	24,64	37,97	74,91

Fig. 15. Memory subsystem total energy and power consumption for the second experiment with 4 MB cache size.



	1Core/L2-1P-1Exp	1Core/L2-1P	2Cores/L2-1P	2Cores/L2-2P	4Cores/L2-1P	4Cores/L2-2P	4Cores/L2-4P
Total Area	316,073	1362,763	681,382	2077,697	340,691	1038,848	3093,070
Area / Bank	9,877	42,586	42,586	129,856	42,586	129,856	386,634

Fig. 16. Occupied area for L2 cache memory with 4 MB cache size.

However, this increase of memory size should be done with caution, since the increase of cache size by four times does not result in a performance gain, but it always results in a higher energy consumption and higher total area occupation.

Thus increasing the cache size may lead to better performance in some cases but factors as energy consumption, occupied area and access latency must be considered as well.

7.3. *Experiment 3 - Cache associativity*

The increase on L2 cache associativity in this third experiment aims to evaluate the influence of temporal locality on cache memory, besides the reduction of the cache misses due to address conflicts, which are inherent in increasing the associativity.

This experiment, like the first one, keeps the total cache size constant at 32 MBytes for all organizations evaluated, only changing the associativity from 8-way to 16-way. The parameters modeled for the L2 cache in this third experiment are described in Table 7.

The results of the third experiment are shown in Figure 17. In this graph, the performance of the organization 1Core/L2-1P is similar to the same organization from the first experiment. The second best organization is 2Cores/L2-2P, which is 2% faster than the same organization in the first experiment.

Figure 18 presents the values for energy and power consumption. The 4Cores/L2-2P and 1Core/L2-1P organizations have the lowest power consumption, although

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

Table 7. L2 Cache memories description for the third experiment.

Experiment 3 - Cache memory associativity						
Sharing	1Core/L2	2Cores/L2	2Cores/L2	4Cores/L2	4Cores/L2	4Cores/L2
R/W Ports	1	1	2	1	2	4
Bank Size	1 MB	2 MB	2 MB	4 MB	4 MB	4 MB
Line Size	64 B	64 B	64 B	64 B	64 B	64 B
Associativity	16-way	16-way	16-way	16-way	16-way	16-way
R/W time	1.49 ns	1.66 ns	2.40 ns	2.25 ns	3.15 ns	5.01 ns
R/W Latency	3 Cycles	4 Cycles	5 Cycles	5 Cycles	7 Cycles	11 Cycles
Dyn. Energy	0.66 nJ	0.98 nJ	1.41 nJ	1.60 nJ	2.13 nJ	3.97 nJ
Static Power	1.46 W	3.02 W	4.08 W	6.62 W	8.17 W	14.19 W
Area	$11.53mm^2$	$18.80mm^2$	$53.89mm^2$	$40.93mm^2$	$108.12mm^2$	$366.29mm^2$

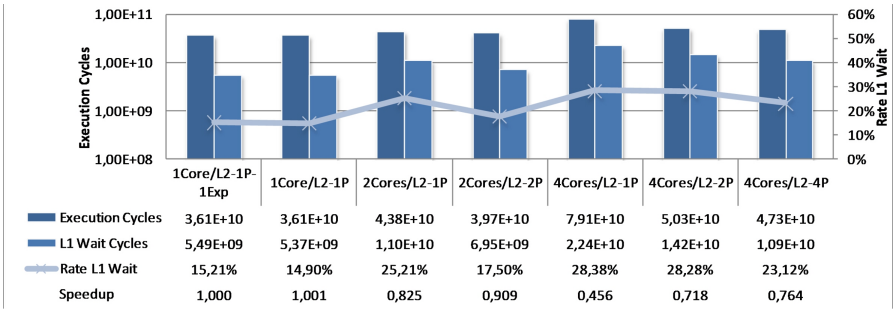


Fig. 17. Execution results for the third experiment.

the 4Cores/L2-2P organization takes more time to run the workload. This behavior is due to the reduction of the number of L2 cache ports and the L2 cache misses.

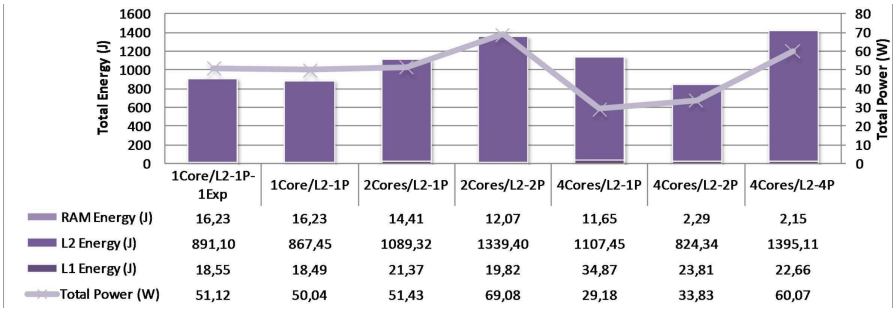


Fig. 18. Memory subsystem total energy and power consumption for the third experiment.

Figure 19 shows the area occupation for the organizations evaluated in this subsection. Even with the lowest power consumption in this experiment, the 4Cores/L2-2P organization occupies more area compared with the first organization.

We can conclude that increasing the associativity, it is possible to increase the L2 cache sharing with lower performance losses than those presented in the first

Electronic version of an article published in Parallel Processing Letters
v. 21, issue 1, 2011, p. 85-106
http://dx.doi.org/10.1142/S0129626411000096

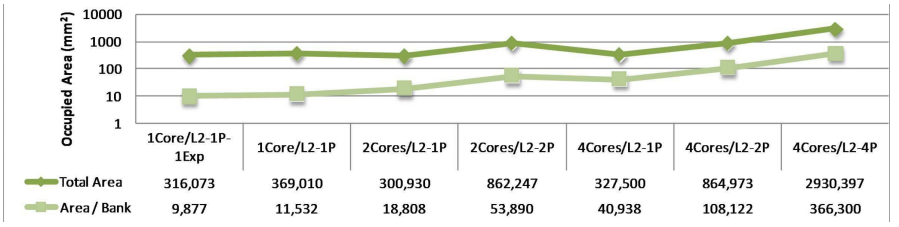


Fig. 19. Occupied area for L2 cache memory on the third experiment.

experiment. However, the organization 2Cores/L2-2P, which is the second fastest, presents unfavorable results in terms of power consumption and area occupation.

7.4. *Experiment 4 - Cache line size*

The increase on the line size of the L2 cache proposed in this fourth experiment, increases the spatial locality of data by getting bigger blocks from the main memory. Thus, in this fourth experiment the line length was increased from 64 to 128 bytes, as shown in Table 8.

Table 8. L2 Cache memories description for the fourth experiment.

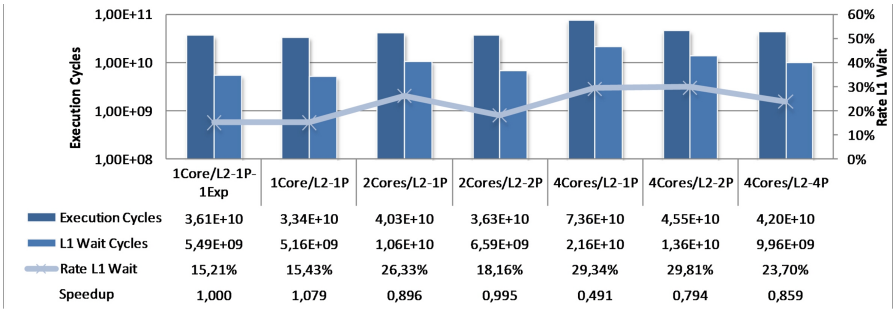
Experiment 4 - Cache memory line size						
Sharing	1Core/L2	2Cores/L2	2Cores/L2	4Cores/L2	4Cores/L2	4Cores/L2
R/W Ports	1	1	2	1	2	4
Bank Size	1 MB	2 MB	2 MB	4 MB	4 MB	4 MB
Line Size	64 B	64 B	64 B	64 B	64 B	64 B
Associativity	8-way	8-way	8-way	8-way	8-way	8-way
R/W Time	1.43 ns	1.59 ns	2.42 ns	2.18 ns	3.02 ns	4.99 ns
R/W Latency	3 Cycles	4 Cycles	5 Cycles	5 Cycles	7 Cycles	10 Cycles
Dyn. Energy	1.48 nJ	2.10 nJ	3.69 nJ	3.88 nJ	5.89 nJ	10.71 nJ
Static Power	1.47 W	2.89 W	4.05 W	5.90 W	7.79 W	14.09 W
Area	$10.18mm^2$	$18.10mm^2$	$58.29mm^2$	$39.88mm^2$	$103.61mm^2$	$382.56mm^2$

The results in Figure 20 show good performance for 1Core/L2-1P and 2Cores/L2-2P organizations, where the first organization achieves 7.9% speedup over the first experiment (1Core/L2-1P-1Exp).

The Figure 21 shows that the power consumption of the L2 cache is very high as we increase the number of ports and the cache sharing. This increase is caused by the static power consumption, since the increase of the runtime of the workload leads to more static power consumption. Only the 1Core/L2-1P, 2Cores/L2-1P and 2Cores/L2-2P organizations have a reasonable energy use.

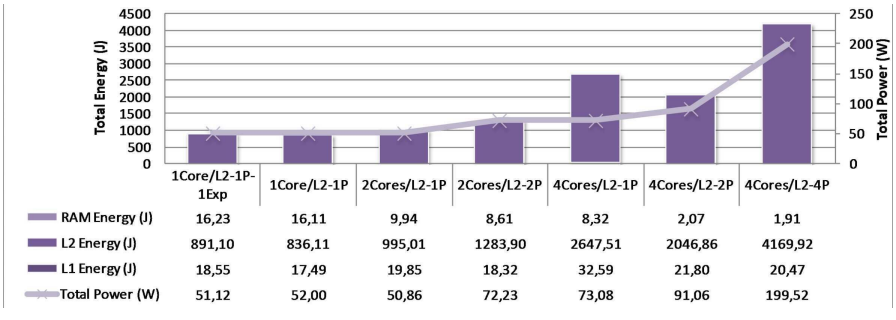
The L2 cache area occupation is shown in Figure 22.As the number of cache ports increases, the occupied area increases as well. Thus, only the organizations with just one data access port present a reasonable area occupation.

Electronic version of an article published in Parallel Processing Letters
v. 21, issue 1, 2011, p. 85-106
http://dx.doi.org/10.1142/S0129626411000096



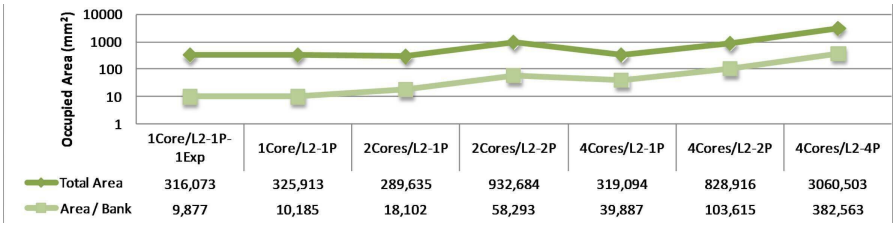
	1Core/L2-1P-1Exp	1Core/L2-1P	2Cores/L2-1P	2Cores/L2-2P	4Cores/L2-1P	4Cores/L2-2P	4Cores/L2-4P
Execution Cycles	3,61E+10	3,34E+10	4,03E+10	3,63E+10	7,36E+10	4,55E+10	4,20E+10
L1 Wait Cycles	5,49E+09	5,16E+09	1,06E+10	6,59E+09	2,16E+10	1,36E+10	9,96E+09
Rate L1 Wait	15,21%	15,43%	26,33%	18,16%	29,34%	29,81%	23,70%
Speedup	1,000	1,079	0,896	0,995	0,491	0,794	0,859

Fig. 20. Execution results for the fourth experiment.



	1Core/L2-1P-1Exp	1Core/L2-1P	2Cores/L2-1P	2Cores/L2-2P	4Cores/L2-1P	4Cores/L2-2P	4Cores/L2-4P
RAM Energy (J)	16,23	16,11	9,94	8,61	8,32	2,07	1,91
L2 Energy (J)	891,10	836,11	995,01	1283,90	2647,51	2046,86	4169,92
L1 Energy (J)	18,55	17,49	19,85	18,32	32,59	21,80	20,47
Total Power (W)	51,12	52,00	50,86	72,23	73,08	91,06	199,52

Fig. 21. Memory subsystem total energy and power consumption for the fourth experiment.



	1Core/L2-1P-1Exp	1Core/L2-1P	2Cores/L2-1P	2Cores/L2-2P	4Cores/L2-1P	4Cores/L2-2P	4Cores/L2-4P
Total Area	316,073	325,913	289,635	932,684	319,094	828,916	3060,503
Area / Bank	9,877	10,185	18,102	58,293	39,887	103,615	382,563

Fig. 22. Occupied area for L2 cache memory on the fourth experiment.

The fourth experiment shows good results in terms of performance, power consumption and occupied area, although the organization 1Core/L2-1P shows the best results among all experiments evaluated. These results are caused by combining the reduction of the cache miss rate with the low latency and low data contention.

Summing up the experimental results, we can see that the traditional techniques (increasing the cache size, associativity, and line size) still have some performance gains. However, all the shared cache organizations we tested performed worse than the private caches, even when the traditional techniques were used. On the other hand, systems with the requirement of low power consumption and occupied area can benefit from shared caches.

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

8. Conclusions and Future Work

Analyzing the current multi-core processors and the research about the shared cache memory, it is impossible to identify a trend about cache sharing among the cores. In this context, this article has a focus on the evaluation of shared L2 cache memory, by simulating a 32 core processor. Thus, we evaluated the influence of the cache size, the associativity, the block size in order to study the influence of these factors on the shared L2 cache and consequently, on the performance of many-core processors.

Concerning the goal, this paper presented the gap missing in related work, by modeling the latencies of cache memories, estimated by the Cacti tool, generating data about the access latencies, power consumption and occupied area similar to those in real systems. Furthermore, this work evaluates cache contention with a different number of ports, which is not extensively discussed in other studies, thus forming a more solid foundation for comparisons.

According to the results of the experiments, shared cache imposes several restrictions on access latency, power and area occupation to increase the number of ports on the system. On the other hand, when using fewer ports, the experiments show a bottleneck to access the cache memory. A performance gain occurs only with the increase of the cache size (+0.20%) and increase of the cache line size (+7.90%). Moreover, in the worst case of port reduction 4Core/L2-1P, generating high contention on cache memory, the best technique evaluated was the increase on line size, which increases performance and reduces power consumption and area occupation.

The experiments show the importance of integrating the cache memory organization and the physical design in order to obtain the best trade-off between performance, power consumption and occupied area.

Considering the metrics evaluated, it became clear that well-known techniques that used to increase the system performance through enlargement of the cache size or associativity will not achieve good performance results for the newer integration technologies. Thus innovation on the cache architecture is necessary in order to ensure high performance for the next generation of many-core processors.

As future work we consider to extend the work to more levels in the cache in order to get specific conclusions about the addition of a level in a memory hierarchy. In addition, studies evaluating the next generation of cache memory architecture are very important, e.g. studies about non-uniform cache architecture are promising, which is also a focus for future work.

Acknowledgement

We would like to thank Matthias Diener, Felipe L. Madruga and the anonymous reviewers for their comments and suggestions. This work was supported in part by CNPq (Brazilian Government).

Electronic version of an article published in Parallel Processing Letters

v. 21, issue 1, 2011, p. 85-106

http://dx.doi.org/10.1142/S0129626411000096

References

[1] Alameldeen, A. R., Mauer, C. J., Xu, M., Harper, P. J., Martin, M. M., Sorin, D. J., Hill, M. D., and Wood, D. A., Evaluating non-deterministic multi-threaded commercial workloads, *Computer Architecture Evaluation using Comercial Workloads* (2002).

[2] Alameldeen, A. R. and Wood, D. A., Variability in architectural simulations of multi-threaded workloads, in *Proceedings HPCA: Int. Symp. on High-Performance Computer Architecture* (2003), pp. 7–18.

[3] Alves, M. A. Z., Freitas, H. C., and Navaux, P. O. A., Investigation of shared l2 cache on many-core processors, in *Proceedings Workshop on Many-Core* (VDE Verlag GMBH, Berlin, 2009), ISBN 978-3-8007-3133-6, pp. 21–30.

[4] Benini, L. and Bertozzi, D., Network-on-chip architectures and design methods, *Computers and Digital Techniques* **152** (2005) 261–272.

[5] Chapman, B., The challenge of providing a high-level programming model for high-performance computing, in *High-Performance Computing: paradigm and infrastructure*, eds. Yang, L. T. and Guo, M., chapter 2 (Wiley Press, 2005), pp. 21–50.

[6] De Micheli, G. and Benini, L., *Networks on Chips: Technology and Tools* (Morgan Kaufmann, 2006).

[7] Freitas, H. C., Santos, T. G. S., and Navaux, P. O. A., Design of programmable noc router architecture on fpga for multi-cluster nocs, *Electronics Letters* **44** (2008) 969–971.

[8] Gibson, J., Kunz, R., Ofelt, D., Horowitz, M., Hennessy, J., and Heinrich, M., FLASH vs.(simulated) FLASH: Closing the simulation loop, *ACM SIGPLAN Notices* **35** (2000) 49–58.

[9] Jain, R., *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling* (J. Wiley, New York, USA, 1991).

[10] Jin, H., Frumkin, M., and Yan, J., The openmp implementation of nas parallel benchmarks and its performance, in *Technical Report: NAS-99-011* (1999).

[11] Kongetira, P., Aingaran, K., and Olukotun, K., Niagara: a 32-way multithreaded sparc processor, *IEEE Micro* **25** (2005) 21–29.

[12] Lilja, D. J., *Measuring Computer Performance* (Cambridge University Press, Cambridge, 2004).

[13] Magnusson, P. *et al.*, Simics: A full system simulation platform, *IEEE Computer Micro* **35** (2002) 50–58.

[14] Marino, M. D., 32-core cmp with multi-sliced l2: 2 and 4 cores sharing a l2 slice, in *Proceedings SBAC-PAD: Int. Symp. on Computer Architecture and High Performance Computing* (IEEE, 2006), pp. 141–150.

[15] Marino, M. D., L2-cache hierarchical organizations for multi-core architectures, in *Proceedings ISPA: Int. Symp. on Parallel and Distributed Processing and Applications* (IEEE, 2006), pp. 74–83.

[16] Nayfeh, B. A., Olukotun, K., and Singht, J. P., The impact of shared-cache clustering in small-scale shared-memory multiprocessors, in *Proceedings HPCA: Second Int. Symp. on High-Performance Computer Architecture* (IEEE, 1996), ISBN 0-8186-7237-4, pp. 74–84.

[17] Shyamkumar Thoziyoor and Naveen Muralimanohar and Norman P. Jouppi, Cacti 5.0, *HP Laboratories* (2007).

[18] Sinharoy, B., Kalla, R. N., Tendler, J. M., Eickemeyer, R. J., and Joyner, J. B., Power5 system microarchitecture, *IBM Journal of Research and Development* **49** (2005).

[19] Zahran, M. M., On cache memory hierarchy for chip-multiprocessor, *SIGARCH Computer Architecture News* **31** (2003) 39–48.