J Supercomput (2008) 45: 341–364
DOI 10.1007/s11227-008-0178-0

An energy consumption characterization of on-chip interconnection networks for tiled CMP architectures

Antonio Flores · Juan L. Aragón · Manuel E. Acacio

Published online: 13 February 2008

Abstract Continuous improvements in integration scale have made possible the inclusion of several processor cores on the same chip. Such designs have been named chip-multiprocessors (or CMPs) and constitute a good alternative to traditional monolithic designs for several reasons, among others, better levels of performance, scalability, and performance/energy ratio. On the other hand, higher clock frequencies and increasing number of transistors available on a single chip have revealed energy consumption as a critical design issue in current and future microarchitectures. In these architectures, the design of the on-chip interconnection network has proven to have significant impact on overall system performance and energy consumption, and that the wires used in such interconnect can be designed with varying latency, bandwidth, and power characteristics.

In this work, we present a detailed characterization of the energy-efficiency of a CMP for parallel scientific applications using *Sim-PowerCMP*, a detailed architectural-level power-performance simulation tool for CMP architectures that integrates several well-known contemporary simulators (RSIM, Hot Leakage and Orion) into a single framework that allows precise analysis and optimization of power dissipation (both dynamic and static) taking into account performance. In this characterization, we pay special attention to the energy consumed on the interconnection network. Results for an 8- and 16-core CMP show that the most power consuming messages are the replies that carry data (almost 70% on average of the total energy consumed in the interconnect) although they represent 30% of the total number of

A. Flores (✉) · J.L. Aragón · M.E. Acacio
Departamento de Ingeniería y Tecnología de Compadores, Facultad de Informática,
Campus de Espinardo S/N, 30071 Murcia, Spain
e-mail: aflores@ditec.um.es

J.L. Aragón
e-mail: jlaragon@ditec.um.es

M.E. Acacio
e-mail: meacacio@ditec.um.es

messages. Furthermore, we show that using on-chip wires with varying latency, bandwidth, and energy characteristics can reduce the energy dissipated by the links of the interconnection network about 65% with an average impact of 10% in the execution time.

Keywords Chip-multiprocessor · Power dissipation model · Microarchitectural level simulator · Heterogeneus on-chip interconnection network · Parallel scientific applications

1 Introduction

Continuous improvements in integration scale have made major microprocessor vendors move to designs that integrate several processor cores on a single die, also known as chip-multiprocessors (CMPs). Chip-multiprocessors can provide higher throughput, more scalability, and greater energy-efficiency compared to wider-issue, single-core processors. Furthermore, energy-efficient architectures are currently one of the major goals pursued by designers in both high performance and embedded processor domains.

On the other hand, tiled architectures provide a scalable solution for supporting families of products with varying computational power, managing the design complexity, and effectively using the resources available in advanced VLSI technologies. Therefore, it is expected that future CMPs will be designed as arrays of replicated tiles connected over a switched direct network [23, 28]. In these architectures, the design of the on-chip interconnection network has proven to have a significant impact on overall system performance and energy consumption, since it is implemented using global wires that show long delays and high capacitance properties. Recently, Wang et al. [24] reported that the on-chip network of the Raw processor consumes 36% of the total chip power. Magen et al. [15] also attribute 50% of overall chip power to the interconnect.

In this paper, we present a detailed evaluation and a characterization of the energy-efficiency of an 8- and 16-core CMP while executing parallel scientific applications using *Sim-PowerCMP* [9], an architectural-level power-performance simulation tool that estimates both dynamic and leakage power for CMP architectures and is based on RSIM x86 [8] (a Linux port of RSIM [10]). Experimental results show that depending on the number of processor cores, the contribution of the interconnection network to the total power ranges from 15% (8-core tiled CMP) to 30% (16-core tiled CMP) on average, with some applications reaching up to 50%. Additionally, we found that most of this power is dissipated in the point-to-point links of the interconnect. Similar results have been reported in [19, 25].

Finally, we show that the most power consuming messages are the replies that carry data although they represent 30% of the total number of messages. By tuning wire's characteristics, it is possible to design wires with varying latency, bandwidth, and energy properties [3]. Using links that are comprised of wires with different physical properties, a heterogeneous on-chip interconnection network is obtained [7]. With such an interconnection network, we show that the energy dissipated by the links can be reduced about 65% with an average impact of 10% in the execution time.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 presents the architecture of the CMP power-performance simulator used in our experiments. Section 4 describes the validation process of the different power models implemented on *Sim-PowerCMP*. The characterization of the energy-efficiency of a CMP and a proposal for reducing the energy consumed by the interconnection links is presented in Sect. 5. Finally, Sect. 6 summarizes the main conclusions of the work.

2 Related work

A large body of research has been recently targeted at understanding the performance, energy, and thermal efficiency of different CMP organizations. Concerns about the increasing energy consumption and thermal constrains in modern high performance processors have resulted in the proposal and development of architectural-level simulation tools that provide power and energy measurements as well as thermal spatial distribution maps.

In [5], Brooks et al. introduce *Wattch*, a dynamic power-performance simulator based on *SimpleScalar* [1] and CACTI [20] that implements dynamic power models for the different structures in a superscalar processor. This simulator was validated with published power numbers for several commercial microprocessors and has been largely used by the research and academic community in the last years. *HotLeakage* [29] is another simulation tool that extends *Wattch* by adding leakage power models for some processor regular structures (caches and register files) allowing more detailed and complete power estimation. In [6], Chen et al. present *SimWattch*, a full system energy simulator based on Simics (a system level simulation tool) and *Wattch*. In [17], a characterization for multicore architectures is presented using a modified version of SESC simulator [18] that uses *Wattch* power models.

In [9], we present *Sim-PowerCMP*, a detailed architectural-level power-performance simulation tool that estimates both dynamic and leakage power for CMP architectures and is based on RSIM x86 [8] (a Linux port of RSIM [10]). We chose RSIM as performance simulator instead of a full-system simulator such as GEMS/Simics [16] or M5 [4] for several reasons. First, RSIM models the memory hierarchy and the interconnection network in more detail. Second, when scientific and multimedia workloads are executed for characterization purposes, the influence of the operating system is negligible and can be ignored. Furthermore, this would be desirable in some cases, where minimal operating system support is required. And third, full-system simulators are progressively slower as the number of processors in a CMP increases. The latter is important since the number of processor cores is expected to increase (for example, the Cell processor integrates 9 processor cores on-chip [11]).

Due to the difficulty of validating our own power models, *Sim-PowerCMP* incorporates already proposed and validated power models for both dynamic power (from *Wattch* [5], CACTI [20]) and leakage power (from *HotLeakage* [29]) of each processing core, as well as the interconnection network (from *Orion* [26]). However, those power models had to be adapted to the peculiarities of CMP architectures.

The on-chip interconnection network is a critical design element in a multi-core architecture and, consequently, it is the subject of several recent works. Among others, Kumar et al. [13] analyze several on-chip interconnection mechanisms and topologies, and quantify their area, power, and latency overheads. Their study concludes that the design choices for the interconnect have significant effect on the rest of the chip, potentially consuming a significant fraction of the real estate and power budget.

A reduced number of works have attempted to exploit the properties of a heterogeneous interconnection network at the microarchitectural level in order to reduce the interconnect energy share. Balasubramonian et al. make the first proposal of wire management at the microarchitectural level. They introduce the concept of a heterogeneous interconnect that is comprised of wires with varying area, latency, bandwidth, and energy characteristics, and they apply it to register communication within a clustered architecture. Finally, Cheng et al. [7] applied the heterogeneous network concept to the cache coherence traffic problem in CMPs. In particular, they propose an interconnection network composed of three sets of wires with varying latency, bandwidth, and energy characteristics, and map coherence messages to the appropriate set taking into account their latency and bandwidth needs. They report significant performance improvement and interconnect energy reduction when a two-level tree interconnect is used to connect the cores and the L2 cache. Unfortunately, insignificant performance improvements are reported for direct topologies (such as the ones employed in tiled CMPs).

3 The *Sim-PowerCMP* simulator

3.1 *Sim-PowerCMP* architecture overview

Sim-PowerCMP is a power-performance simulator derived from RSIM x86 [8] (a Linux port of RSIM [10]). It models a tiled CMP architecture consisting of arrays of replicated *tiles* connected over a switched direct network (Fig. 1). Each tile contains a processing core with primary caches (both instruction and data caches), a slice of the L2 cache, and a connection to the on-chip network. The L2 cache is shared among the different processing cores, but it is physically distributed between them.[1] Therefore, some accesses to the L2 cache will be sent to the local slice while the rest will be serviced by remote slices (L2 NUCA architecture [12]). In addition, the L2 cache stores (in the tags' part of the local L2 slice) the directory information needed to ensure coherence between the L1 caches. On a L1 cache miss, a request is sent down to the appropriate tile where further protocol actions are initiated based on that block's directory state, such as invalidation messages, intervention messages, data writeback, data block transfers, etc. In this paper, we assume a process technology of 65 nm, tile area of approximately 25 mm^2, and a die size in the order of 400 mm^2 [28, 30]. Note that this area is similar to the largest die in production today (Itanium 2 processor—around 432 mm^2 [14]). Note also that due to manufacturing costs and form factor limitations, it would be desirable to keep die size as low as possible [30].

[1] Alternatively, each L2 slice could have been treated as a *private* L2 cache for the local processor. In this case, cache coherence had to be ensured at the L2 cache level (instead of L1). In any case, our proposal would be equally applicable in such configuration.

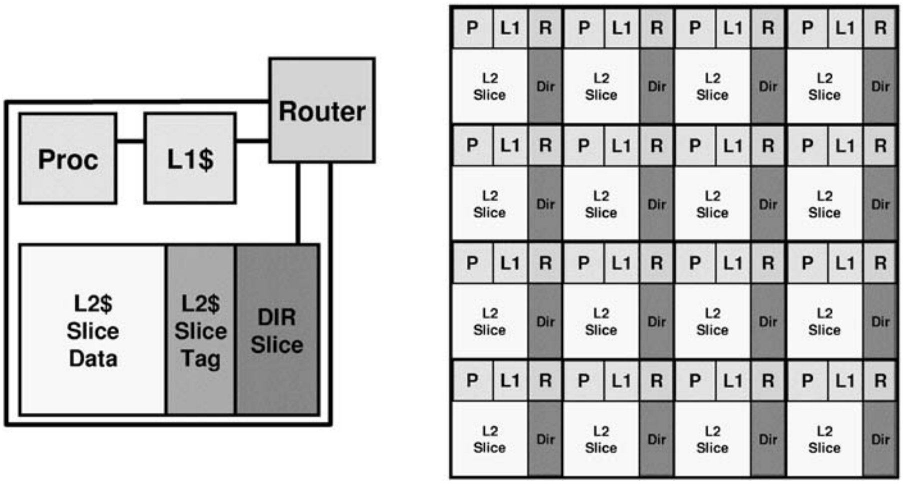


Fig. 1 Tiled CMP architecture overview

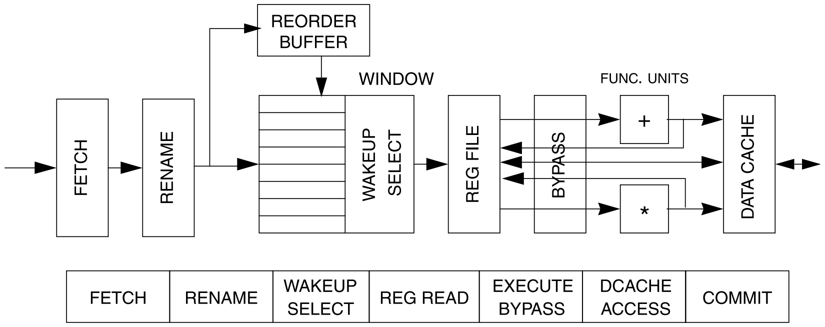


Fig. 2 Architecture overview of each core in *Sim-PowerCMP*

Each processing core is an out-of-order multiple issue processor (although in-order issue is also supported), modeled according to the pipeline organization shown in Fig. 2. The fetch unit fetches several instructions per cycle from the instruction cache. These instructions are decoded and register renaming is performed. In case of fetching a branch, its outcome and target address is predicted. Renamed instructions are placed in the instruction window (IW) while the reorder buffer (ROB) keeps track of program ordering. Once an instruction has all its inputs ready, it can be executed by the corresponding functional unit. The MIPS R10000 processor, in which RSIM simulator is based (remember that *Sim-PowerCMP* is derived from RSIM) and the Alpha 21264 processor are two examples of this architectural model.

On the other hand, *Wattch* and *HotLeakage* simulators that will be used for validating *Sim-PowerCMP* in Sect. 4, are based on the RUU architectural model proposed by Sohi [22]. The RUU is a big structure that unifies the instruction window and ROB at the same time it acts as a physical register file that temporary stores the results of noncommitted instructions.

There are two major differences between the two models. The first difference is that in the architectural model of Fig. 2 (followed by the MIPS R10000, RSIM, and *Sim-PowerCMP*), all computed values, speculative or not, are stored in the register file. However, in Sohi's model, the RUU is responsible of temporary storing the non-committed output values while a separate register file is responsible of storing the output values only when instructions have been committed and, therefore, they are nonspeculative values. The second major difference is that in *Sim-PowerCMP* architectural model, computed values are not sent to the instruction window, only the tags are sent for the tagmatch (or wake-up) process. Computed values are send to the register file. However, in Sohi's model (used by *HotLeakage*), all computed values are sent to the RUU and, therefore, dependent instructions get their inputs from the RUU. These two architectural differences must be taken into account when validating the proposed power model, as we will show in next section.

3.2 *Sim-PowerCMP* power model overview

We have incorporated into our simulator already proposed and validated power models for both dynamic power (from *Wattch* [5], CACTI [20]) and leakage power (from *HotLeakage* [29]) of each processing core, as well as the interconnection network (from Orion [26]). Adapting the power models of the main hardware structures of each core and the interconnection network to *SimPower-CMP* is not a trivial task. The power modeling infrastructure used in *Wattch* and *HotLeakage* is strongly coupled with the performance simulator code and, although they are mostly parametrized power models, considerable effort and deep understanding of the simulator implementation is needed in order to port the power model infrastructure to *SimPower-CMP*. One major hurdle was the extensive use of global variables to keep track of which units are accessed per cycle in order to account for the total energy consumed by an application. In a power-aware CMP simulator, these counters and statistics must be collected on a per-core basis, and the use of global activity counters is forbidden. In order to avoid rewriting the power model code used in these simulators, we choose to keep original global variables that are widely used across *Wattch* and *HotLeakage* and define per core basis counters and partial/global power results. These values are transferred to/from the global variables each time the power model functions are called. In this way, we were able to decouple the power model code borrowed from *Wattch* and *HotLeakage* simulators.

On the other hand, the interconnection network power model used in *Orion* is loosely coupled with the Liberty infrastructure in which the simulator is based, making its integration with another performance simulator easier. However, some additional changes were needed in order to make it fully interoperative with *SimPower-CMP*. Specifically, *Orion* uses the power models defined in *Wattch,* but most of the constant values used in *Wattch* have been converted to parameters that can be modified at execution time making *Orion* more flexible. Therefore, it was necessary to reconcile both approaches used to model the power dissipation. Our decision was to use the more flexible approach proposed by *Orion*, modifying the header files where those constant values were defined.

Finally, we either changed some power models or derive new ones to match several of the particularities of the CMP implemented in *SimPower-CMP*. For instance,

Table 1 Configuration of the CMP architecture used in the validation of the power models of *Sim-PowerCMP*

Core configuration		
Parameter	*HotLeakage*	*Sim-PowerCMP*
Fetch/issue/commit width	4	
Active list	–	64
Instr. window (RUU)	32	–
Register file	32	64
Functional units	2 IntALU, 2 FPALU	
	2 AddrGen, 2 mem ports	
LSQ entries	64	
L1 I/D-cache	32 K, 4-way	
L2 cache	256 K, 4-way, 10 + 20 cycles	
Memory	400 cycles	
Branch pred.	two-level, 4 K-entries	
BTB	4 K-entries	
CMP parameters		
Technology	65 nm	
Core size	25 mm^2	
Number of cores	8	
Interconnection network	2D mesh	
Network bandwidth	75 GB/s	
Router parameters		
Link length	5 mm	
Flit size	75 Bytes	
Buffer size	64 flits	

we needed to model the impact of the directory. In the CMP architecture modeled in *SimPower-CMP*, the L2 cache stores the directory information needed to ensure coherence between the L1 caches. So, we changed the power model of the L2 cache to account for both the extra storage bits and the extra accesses to the directory.

4 Validation of the power model of *Sim-PowerCMP*

Validating power models is a crucial task to obtain reasonably accurate simulation results. We have used a validation methodology based on checking our results against the ones obtained under the same configuration with other power simulators that have already been validated and are widely used by the research community: *HotLeakage* in the case of processor cores and *Orion* for the power of the internal interconnection network.

Table 1 shows the configuration used to validate the power models used in *Sim-PowerCMP*. It describes an 8-core CMP built in 65 nm technology. The tile area has been fixed to 25 mm^2, including a portion of the second-level cache [28]. With

Table 2 Dynamic power breakdown for the different structures in a processor core

	HotLeakage (W)		*Sim-PowerCMP* (W)	
Total dynamic power consumption:	**19.36**		**19.46**	
Branch predictor power consumption:	0.82	4.72%	0.82	4.69%
Rename logic power consumption:	0.08	0.49%	0.09	0.54%
Instruction decode power (W):	0.0040		0.0040	
RAT decode_power (W):	0.0316		0.0316	
RAT wordline_power (W):	0.0085		0.0097	
RAT bitline_power (W):	0.0386		0.0463	
DCL comparators (W):	0.0023		0.0023	
Instruction window power consumption:	0.52	3.01%	0.07	0.39%
tagdrive (W):	0.0354		0.0425	
tagmatch (W):	0.0169		0.0198	
selection logic (W):	0.0068		0.0067	
decode_power (W):	0.0316		0	
wordline_power (W):	0.0205		0	
bitline_power (W):	0.4123		0	
Load/store queue power consumption:	0.64	3.69%	0.64	3.67%
Arch. register file power consumption:	0.46	2.68%	0.82	4.68%
decode_power (W):	0.0316		0.0653	
wordline_power (W):	0.0205		0.0205	
bitline_power (W):	0.4123		0.7308	
Result bus power consumption:	0.77	4.44%	1.02	5.85%
Total clock power:	7.30	42.07%	7.24	41.49%
Int ALU power:	1.55	8.91%	1.55	8.87%
FP ALU power:	2.37	13.66%	2.37	13.58%
Instruction cache power consumption:	0.67	3.88%	0.67	3.86%
Itlb_power (W):	0.05	0.29%	0.05	0.28%
Data cache power consumption:	1.35	7.77%	1.35	7.72%
Dtlb_power (W):	0.17	0.97%	0.18	0.96%
Level 2 cache power consumption:	0.59	3.43%	0.59	3.41%
Ambient power consumption:	2.00	10.33%	2.00	10.28%

this configuration, links that interconnect routers configuring the 2D mesh topology would measure around 5 mm.

Table 2 shows an *a priori* comparison of the maximum dynamic power breakdown for a core of the CMP using *Sim-PowerCMP* and *HotLeakage*. Note that there are some differences for structures such as the rename logic, the register file, and mainly, the instruction window. These differences are due to the different superscalar architectures implemented in both simulators, as mentioned in the previous section. *HotLeakage* simulator, based on *SimpleScalar*, implements the RUU model that integrates the instruction window, ROB, and physical registers in the same hardware

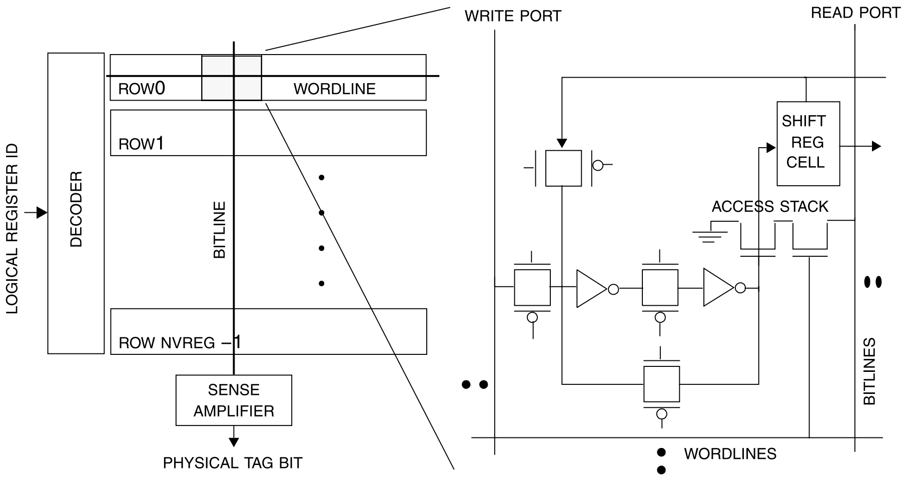


Fig. 3 Mapping table of the rename logic (*left*). Single cell of the mapping table (*right*)

Table 3 Static power dissipation for regular structures of a processor core

	HotLeakage	*Sim-PowerCMP*
Total static power consumption (W):	**0.23812**	**0.24665**
Arch. register file power consumption: (W)	0.00449	0.00898
Instruction cache power consumption: (W)	0.02397	0.02397
Data cache power consumption: (W)	0.02397	0.02397
Level 2 cache power consumption: (W)	0.18974	0.18974

structure. So, the power dissipation of the instruction window (RUU for *HotLeakage*) is quite high. It is important to note that the main contribution to instruction window power dissipation is due to the accesses to the physical registers (*bitline-power* in Table 2). In *SimPower-CMP* superscalar processor model, this power dissipation is zero. On the other hand, the model used in *SimPower-CMP* requires to duplicate the size of the register file because it keeps both speculative and nonspeculative result values (logical and physical registers). This explains the higher power dissipation in the register file (as shown in Table 2). The higher power dissipation in the rename logic is due to the fact that physical register tags have one additional bit in *Sim-PowerCMP* model because we double the size of the register file, as it can be observed in Fig. 3.

Table 3 shows the static power dissipation for the main regular hardware structures in a core. The only difference is found in the register because of the bigger size in *Sim-PowerCMP*.

After this *a priori* analysis of the maximum power dissipation for a core, the next step in the validation of the power model is to compare the dynamic power dissipation when real programs are simulated. However, it is important to note that both simulators use different instruction set architectures (SPARC and PISA ISAs for *Sim-PowerCMP* and *HotLeakage*, respectively) which complicates the comparison.

Table 4 Percentage of instructions committed in both test programs

	Test 1		Test 2	
	HotLeakage	*PowerCMP*	*HotLeakage*	*PowerCMP*
Arithmetic-logical	54.10%	54.17%	55.58%	61.92%
Data transfer	41.71%	41.67%	38.86%	33.31%
Unconditional jump	0%	0%	0%	0%
Conditional branch	4.18%	4.17%	4.56%	4.77%

Furthermore, the use of different compilers as well as slightly different optimization flags can appreciably change the instruction mix for a program. Therefore, we decided to use a two-step validation strategy, initially using very simple test programs in order to obtain a preliminary validation. Then we performed the final validation using some of the SPEC-2000 applications.

A preliminary validation using mini tests has two advantages: (1) the percentage of memory accesses, arithmetic-logic, and control instructions are very similar when we use these simple programs, even when the ISAs are different, allowing an easier comparison of dynamic power dissipation; (2) the generated code is simple enough to predict which core structures are the main contributors to dynamic power dissipation in order to explain the sources of potential discrepancies.

The test programs used for the preliminary power model validation were two mini tests written in C and compiled using PISA (*HotLeakage* simulator) and SPARC (*Sim-PowerCMP* simulator) versions of the gcc compiler. The optimization options activated in both cases were `-O3 -funroll-loops`. The first test performs a sequential access to an array of integers, accumulating all the values into a global variable, whereas the second one implements the multiplication of two matrices of doubles. Table 4 shows the percentage of instructions that are committed in both cases. Even with these simple codes and using the same compiler with the same optimization options, the obtained percentages are not exactly the same, but are very similar. To perform the comparison, we used the power model *cc3* defined in [5], which implements a clock-gating scheme that adjusts the dynamic power based on resource utilization.[2]

Table 5 shows the results obtained after completing the simulation for both mini tests under perfect cache assumption. It can be observed that results are almost identical, except for the register file and the instruction window. These differences are related with the particular microarchitecture modeled by each simulator. As explained before, the main contribution to instruction window power consumption is due to the accesses to the physical registers. In the model implemented in *SimPower-CMP*, these acesses are done to the register file that now keeps both speculative and nonspeculative result values (logical and physical registers). This explains the higher average

[2]If a multiported unit is used in a clock cycle, its maximum power dissipation (Table 2) is scaled linearly with port usage. Unused units in a clock cycle are supposed to still dissipate 10% of their maximum power, rather than drawing zero power.

Table 5 Dynamic power dissipation for a core after simulating both minitests

	Test 1				Test 2			
	Avg. access/c		Avg. power (W)		Avg. access/c		Avg. power (W)	
	HL	*SPcmp*	*HL*	*SPcmp*	*HL*	*SPcmp*	*HL*	*SPcmp*
Rename table	2.40	2.30	0.05	0.05	2.86	2.57	0.06	0.06
Branch prediction	0.10	0.10	0.11	0.11	0.16	0.14	0.13	0.13
Instruction window	9.19	4.70	0.52	0.04	10.59	5.57	0.55	0.05
LSQ	1.00	1.00	0.28	0.28	1.27	1.28	0.26	0.25
Register file	3.50	5.80	0.13	0.38	4.25	7.14	0.16	0.46
L1 I-cache	2.40	2.40	0.72	0.72	2.86	3.00	0.70	0.72
L1 D-cache	1.00	1.00	0.79	0.80	0.84	0.86	0.73	0.69
Int + FP ALU	2.40	2.40	1.17	1.17	2.85	2.99	1.73	1.72
Result bus	3.30	3.30	0.64	0.58	3.49	3.57	0.64	0.63
Clock			2.51	2.84			3.24	3.44
Fetch stage			0.84	0.84			0.86	0.85
Dispatch stage			0.05	0.05			0.06	0.06
Issue stage			3.47	2.94			3.93	3.39
Avg. power/cycle			7.24	7.08			8.25	8.21
Avg. power/instr.			3.40	3.42			3.49	3.31
Max power/cycle			9.63	9.26			12.12	11.49

accesses per cycle and power consumption in the register file for *SimPower-CMP* as well as the underutilization of the instruction window compared with the results obtained for the *HotLeakage* simulator.

The second step of our power model validation methodology consisted in comparing the results obtained after running a subset of the SPEC2000 applications. In these simulations, we still assume perfect L1 caches in order to avoid interferences due to the different implementations of the memory hierarchy in both simulators. Figure 4 shows the dynamic power dissipation as well as the IPC obtained for each application. In general, we obtain the same power distribution among the different hardware structures of a core, although there are some differences that are worth to explain.

First, we observe higher power dissipation in a core for *HotLeakage*, due to the fact that for the same applications the IPC obtained in this simulator is usually higher due to the different instruction set architectures used in both simulators. The higher the IPC, the higher the number of accesses per cycle to the different hardware structures that are modeled. This leads to an increase in the dynamic power of these structures. For the *mcf* application, where the IPC obtained in both simulators is similar, power dissipation is very close. Finally, if we analyze the power distribution for *Sim-PowerCMP*, we can appreciate a considerable drop in the power dissipated by the instruction window, partially compensated by the higher power dissipation in the register file. The reason to this global dynamic power drop is the different types of processor cores modeled inside each simulator, as cited previously.

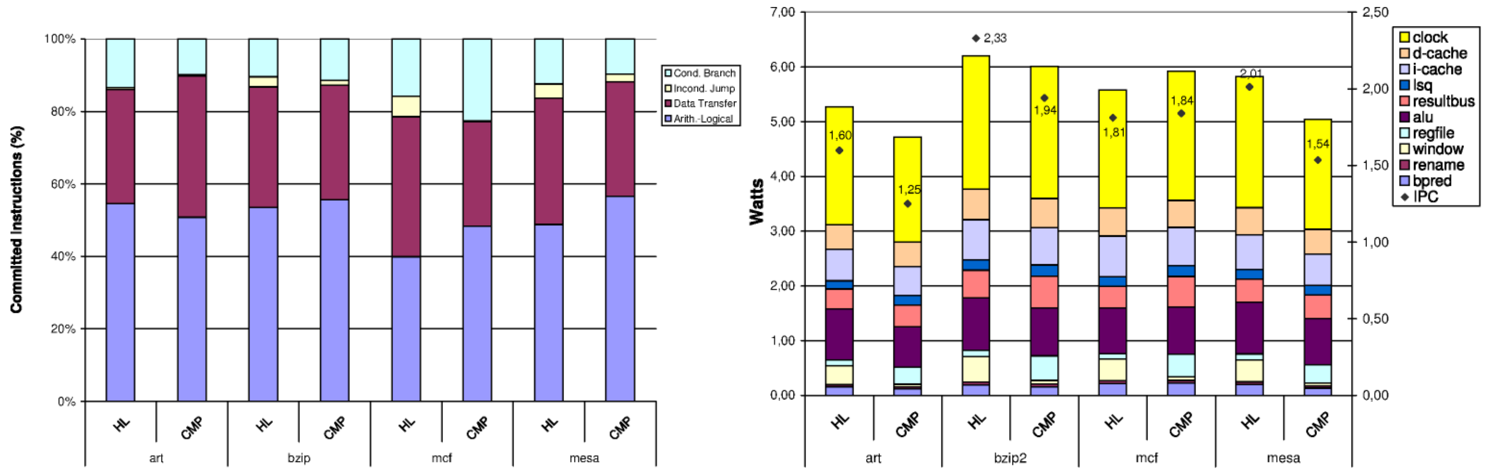


Fig. 4 Classification of committed instructions (*left*); and comparison of dynamic power for the SPEC2000 on a single core of the CMP (*right*)

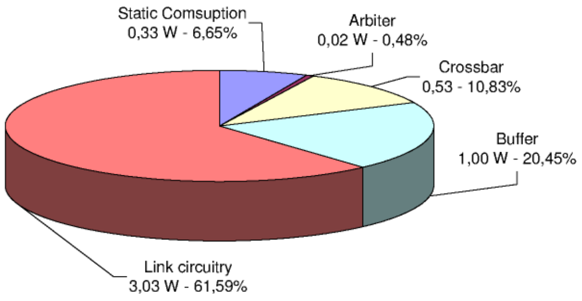


Fig. 5 Distribution of the power dissipation inside a router

Once we finished the validation of the power model associated with each core of the CMP, the next step was to validate the power model of the interconnection network. Figure 5 shows the distribution of the power dissipation for routers that implement the 2D-mesh. For our modeled routers, 62% of the total power comes from the link circuitry. This value is similar to the 60% dissipated by links in the Alpha 21364 routers and a little lower than the 82% cited in [19]. The power dissipated by the links strongly depends on the amount of buffer space assigned to the router compared with channel bandwidth. Our results also agree with the results reported in [25], with a maximum power dissipation of 2–3 W per router inside a CMP (excluding link circuitry). With this power data, the interconnection network takes about 20% of the total CMP power budget, as published in different works [19, 25].

As conclusion, we have compared the results obtained when several applications are simulated in both *SimPower-CMP* and *HotLeakage* in order to validate the power models implemented in our simulation framework. As expected, some differences were obtained mainly due to the different architecture organization modeled in both simulators as well as the different instruction set architectures. Moreover, the overall distribution of the power consumption among the main hardware structures presented in a tiled chip multiprocessor agree with the results reported by different researchers [17, 19, 25].

5 Evaluating the energy-efficiency of CMP architectures

5.1 Experimental framework

In this section, we present a characterization of the energy-efficiency of an 8- and 16-core tiled CMP executing parallel scientific applications. Table 6 (left) shows the architecture configuration used across this paper. Reply messages are 67-byte long since they carry control information (3-bytes) and a cache line (64 bytes). On the contrary, request, coherence, and coherence reply messages that do not contain data are at most 11-byte long (just 3-byte long for coherence replies).

Table 6 (right) shows the applications used in our experiments. *MP3D* is from the SPLASH benchmark suite [21], *Barnes-Hut*, *FFT*, *LU-cont*, *LU-noncont*, *Ocean-cont*, *Ocean-noncont*, *Radix*, *Raytrace,* and *Water-nsq* are from the SPLASH-2 benchmark suite [27], Berkeley *EM3D* simulates the propagation of electro-magnetic waves through objects in three dimensions, and *Unstructured* is a computational fluid

Table 6 Configuration of the baseline CMP architecture (*left*) and applications and problem sizes evaluated (*right*)

CMP configurations	
Parameter	
Process technology	65 nm
Tile area	25 mm^2
Number of tiles	8, 16
Cache line size	64 bytes
Core	4 GHz, in-order 2-way model
L1 I/D-Cache	32 KB, 4-way
L2 Cache (per core)	256 KB, 4-way, 10 + 20 cycles
Memory access time	400 cycles
Network configuration	2D mesh
Network bandwidth	75 GB/s
Link width	75 bytes
Link length	5 mm

Application	Problem size
Barnes-Hut	16 K bodies, 4 timesteps
EM3D	9600 nodes, 5% remote links, 4 timesteps
FFT	256 K complex doubles
LU-cont	256 × 256, B = 8
LU-noncont	256 × 256, B = 8
MP3D	50000 nodes, 2 timesteps
Ocean-cont	258 × 258 grid
Ocean-noncont	258 × 258 grid
Radix	2 M keys
Raytrace	car.env
Unstructured	mesh.2 K, 5 timesteps
Water-nsq	512 molecules, 4 timesteps
Water-spa	512 molecules, 4 timesteps

dynamics application that uses an unstructured mesh. Problem sizes have been chosen commensurate with the size of the L1 caches and the number of cores used in our simulations, following the recommendations given in [27]. All experimental results reported in this work are for the parallel phase of these applications. Data placement in our programs is either done explicitly by the programmer or by our simulator which uses a first-touch policy on a cache-line granularity. Thus, initial data-placement is quite effective in terms of reducing traffic in the interconnection network.

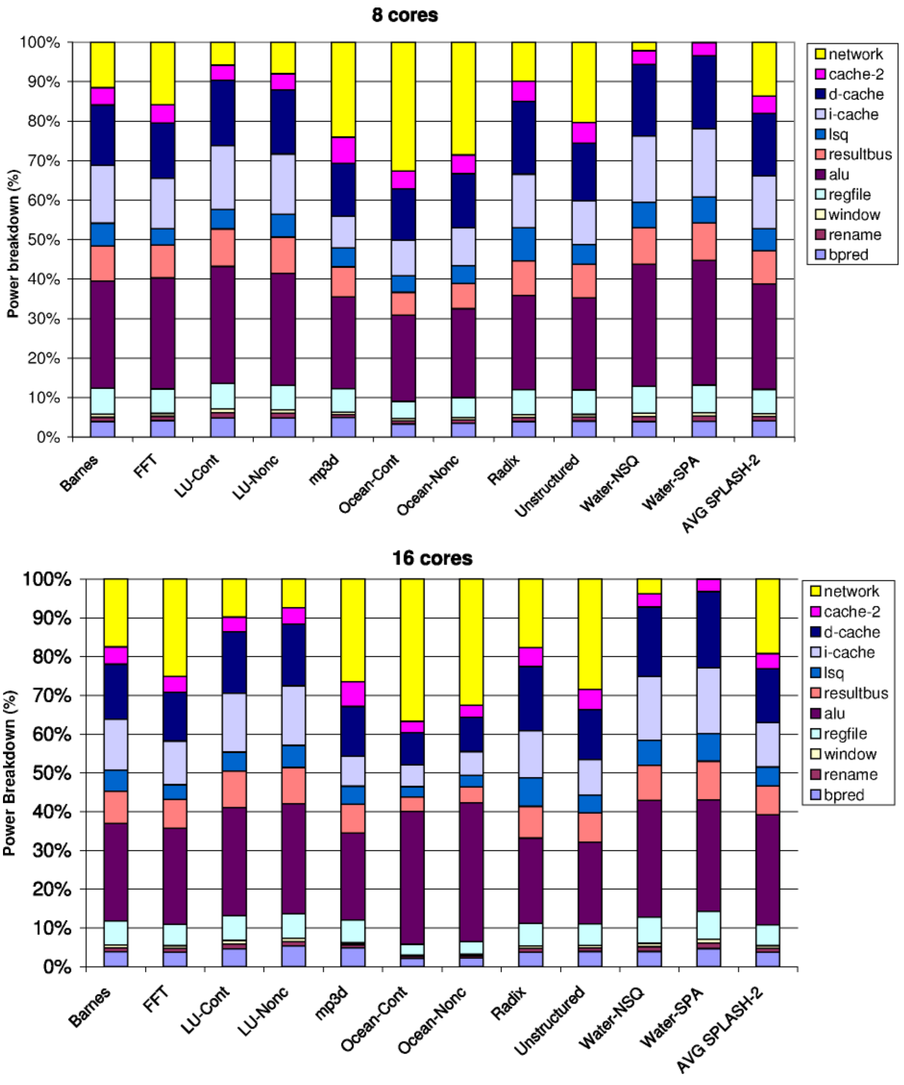


Fig. 6 Breakdown of the power dissipation in an 8-core (*top*) and 16-core CMP (*bottom*) for several parallel scientific applications (the contribution of the clock logic is not included for clarity)

5.2 Characterization of the energy-efficiency of a CMP executing parallel scientific applications

Figure 6 presents a breakdown of the power dissipated in an 8- and 16-core CMP. Total power dissipation is split among the most important structures of the CMP (for the sake of legibility, we have omitted the contribution of the clock). As expected, it can be observed that most of the power is dissipated in the processor cores of the CMP. In particular, the ALU reveals as one of the most consuming structures of the CMP, as reported in [17]. Regarding the caches (private L1 caches and the shared multibanked L2 cache), we can see that their fraction of the total power is quite significant. Additionally, we see that in this case, most of the power is dissipated in the first-level instruction and data caches. Figure 6 also shows that the contribution of the interconnection network to the total power is close to 20% on average, with several applications reaching up to 30%. In this case, we have observed that most of this power is dissipated in the point-to-point links used to configure the interconnect. In the literature, it can be found a plethora of techniques and proposals aimed at reducing the energy consumption of the main core structures, however, the impact that the interconnection network has on total energy consumption is very significant and techniques aimed at optimizing the delivery of messages through the interconnect should be the goal of current research efforts.

For these reasons, this paper focuses on the characterization of the energy-efficiency of the interconnection network in future CMP architectures. For this purpose, it would be helpful to analyze how this power distributes among the different types of messages that travel on the interconnect.

5.3 Characterization of the on-chip CMP interconnection network

There are a variety of message types traveling on the interconnect of a CMP, each one with properties that are clearly distinct. In general, we can classify messages into the following groups:

1. *Request messages*, that are generated by cache controllers in response to L1 cache misses. Requests are sent to the corresponding home L2 cache to demand privileges (read-only or read/write) over a memory line.
2. *Response messages*, that are sent in response to requests. These messages can be generated by the home L2 cache controller or, alternatively, by the remote L1 cache that has the single valid copy of the data, and they can carry the memory line or not. The latter is due to the presence of upgrade requests used to demand ownership for a line already kept in the local L1 cache.
3. *Coherence commands*, that are sent by the home L2 cache controller to the corresponding L1 caches to ensure coherence (for example, invalidation messages).
4. *Coherence responses*, sent by the L1 caches back to the corresponding home L2 in response to coherence commands.
5. *Replacement messages*, that the L1 caches generate in case of exclusive or modified lines being replaced.

Figure 7 plots the fraction of each message type on the total number of messages for an 8- and a 16-core CMP configuration for the applications used in our evaluation. As it can be seen, on average, more than 60% of the messages are related to memory accesses (a request and its corresponding reply), whereas the rest has to do with coherence enforcement (25%) and block replacement (15%). It is interesting to note that almost all replies imply the transfer of memory blocks (*Resp.* + *Data*).

Even more interesting is Fig. 8 which shows the power dissipated by each message type. Most of the power in the interconnect is associated to reply messages that carry a cache line (55%–65%). As previously commented, most of this power is dissipated in the point-to-point links and, therefore, message size plays a major role. In particular,

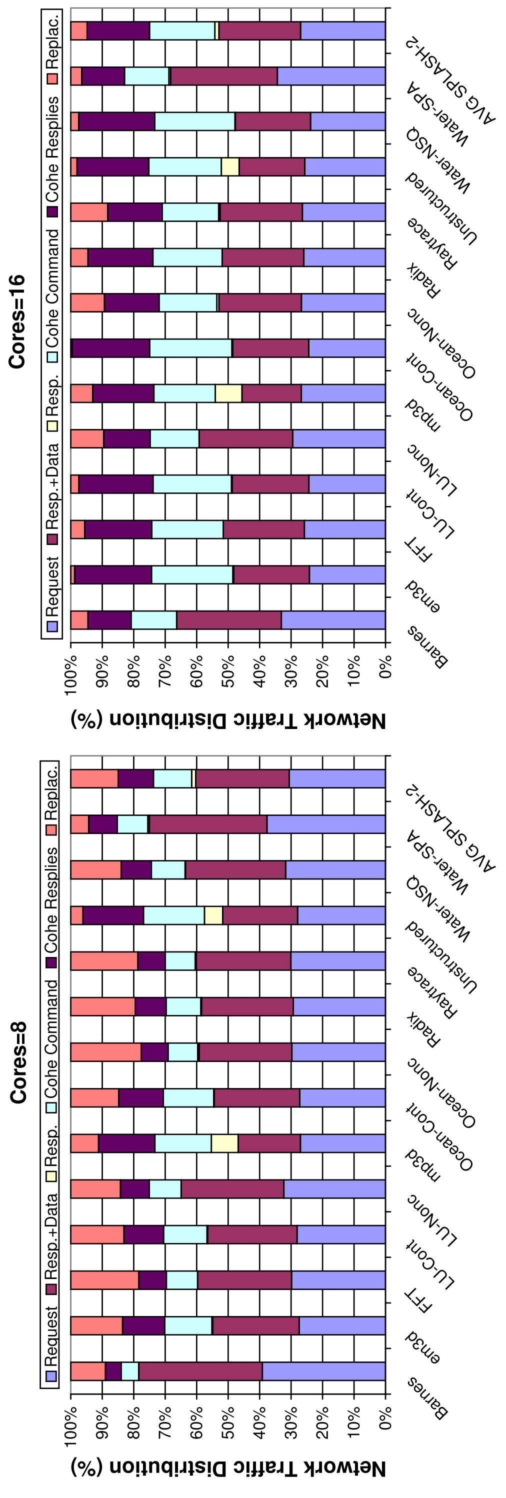
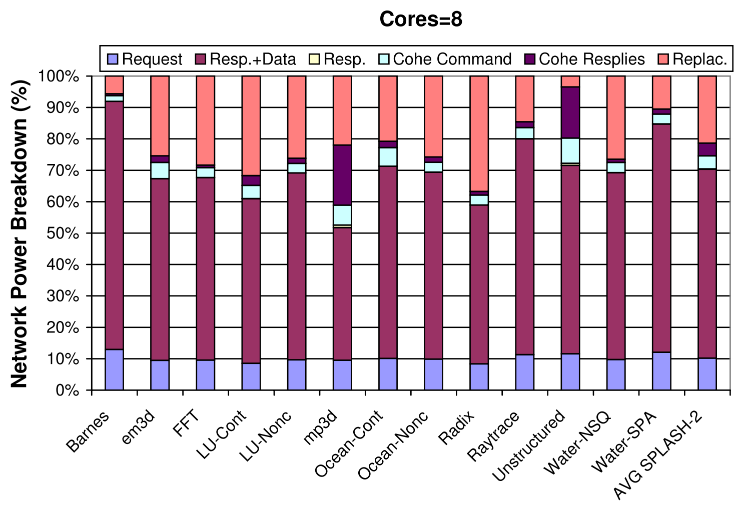


Fig. 7 Breakdown of the messages that travel on the interconnection network for an 8-core (*left*) and a 16-core CMP (*right*)



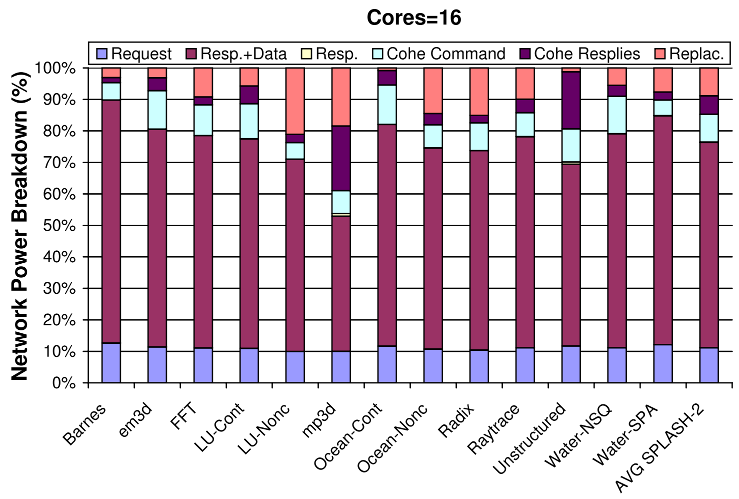


Fig. 8 Percentage of the power dissipated in the interconnection network by each type of message for an 8-core (*left*) and a 16-core CMP (*right*)

Table 7 Area, delay and power characteristics of wire implementations (extracted from [7])

Wire type	Relative latency	Relative area	Dynamic power (W/m) α = switching factor	Static power W/m
Baseline-Wire	1x	1x	2.65α	1.0246
L-Wire	0.5x	4x	1.46α	0.5670
PW-Wire	2x	0.5x	0.80α	0.2720

reply messages are 67-byte long since they carry control information (3-bytes) and a cache line (64 bytes). On the contrary, requests and coherence commands are 11-byte long since beside control information (3 bytes) they also carry address information. Finally, coherence replies are just 3-byte long. This result shows that optimizing the delivery of reply messages that carry data will be rewarding to reduce the energy consumed by the interconnection network in CMPs.

As a case of study, we analyze in this paper the use of a heterogeneous interconnect [2, 7] by replacing global inter-core wires with two new wire types: a low-latency *L-Wire* and power-efficient *PW-Wire*. *L-Wires* are designed by increasing the width and spacing of the repeaters, whereas *PW-Wires* are designed by decreasing the number and size of repeaters within minimum-width wires. This design is similar to that proposed by Cheng et al. in [7], although their proposal used a heterogeneous interconnect with three types of wires in two metal planes.

Table 7 shows the area, delay and power characteristics of *L-* and *PW-Wires* related to baseline wires as reported in [7]. *L-Wires* yield a two-fold latency improvement at a four-fold area cost, relative to baseline wires. *PW-Wires* are designed to reduce power dissipation with twice the delay of baseline wires. In order to match the metal area of the baseline configuration, each original 75 byte unidirectional link is designed to be made up of 88 *L-Wires* (11 bytes) and 248 *PW-Wires* (31 bytes). Request, coherence, and coherence reply messages that do not contain data are sent using the *L-Wires*, whereas reply messages that carry data and replacements will be send using the *PW-Wires*.

Using the described heterogeneous interconnect, Fig. 9 (left) depicts the normalized execution time with respect to that obtained for the baseline configuration. On average, we observe that performance is degraded 10%. The reason for this performance degradation is the increased latency of the reply messages that carry data, which are now sent through the slower *PW-Wires*. This degradation has high variability, ranging from almost negligible degradation for MP3D and Water-NSQ applications to almost 40% for Ocean-Cont application. This result is very interesting because it shows that some applications, such as MP3D or Water-NSQ, have enough parallelism to hide the increased latency imposed by the use of *PW-Wires*, whereas other applications, as Barnes or Ocean-Cont, are not able to hide these longer latencies and show poor performance. So, further investigation needs to be done in order to deal with that kind of applications.

Figure 9 (right) shows both normalized link energy and energy-delay product metric (EDP). Through the use of the heterogeneous interconnect presented in this work, we obtain an average reduction of 60%–65% in the energy dissipated by the intercore links. This reduction is quite similar for all the applications. The Energy-Delay metric

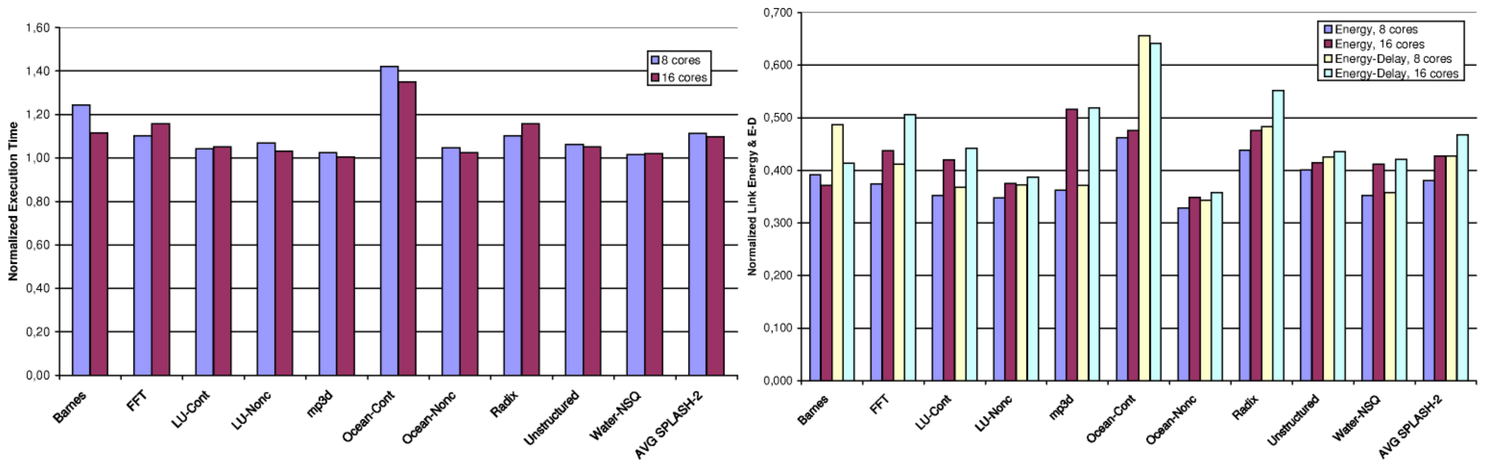


Fig. 9 Normalized execution times (*left*) and link energy and energy-delay (ED) for the network (*right*) when heterogeneous links are used

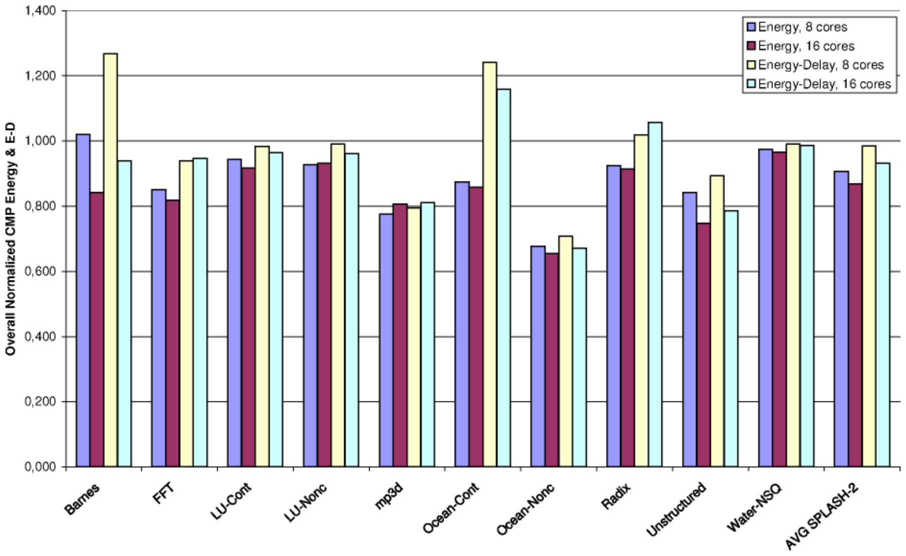


Fig. 10 Normalized energy and energy-delay product (*EDP*) for the full CMP when heterogeneous links are used

also shows good results with average savings close to 55%, although in this case the variability between applications is higher because in the EDP metric the execution time gains importance.

Finally, Fig. 10 presents both the normalized energy and energy-delay product metrics for the full CMP. As it can be observed, important energy savings are obtained for the whole CMP when the heterogeneous interconnect introduced in this paper is used. The extent of these savings depends on the total number of cores of the CMP, ranging from 10% for the 8-core configuration to 15% for the 16-core one. On the other hand, when the energy-delay metric is considered, we still find savings ranging from 2% for the 8-core CMP to 9% for the 16-core one. These results reveal that correctly organizing the interconnection network and properly managing the different types of messages through it will have significant impact on the energy consumed by future CMPs.

6 Conclusions and future work

In this work, we present a characterization of the energy-efficiency of a CMP executing several parallel scientific applications using *Sim-PowerCMP* [9], an architectural-level power-performance simulation tool that estimates both dynamic and leakage power for CMP architectures. Experimental results show that the contribution of the interconnection network to the total power ranges from 15% (8-core tiled CMP) to 30% (16-core tiled CMP) on average, with some applications reaching up to 50%. Then we perform a deeper analysis of the energy consumed in the interconnection network for each message type. Results for an 8- and 16-core CMP show that the most consuming messages are the replies that carry data (almost 70% on average of the total energy consumed in the interconnect) although they represent 30% of the total number of messages. Furthermore, we show that using on-chip wires with varying latency, bandwidth, and energy characteristics can reduce the power dissipated by the links of the interconnection network about 65% with an average impact of 10% in the execution time. Overall CMP energy savings range from 10% for the 8-core configuration to 15% for the 16-core one (from 2% to 9% if the energy-delay product metric is considered).

As part of our future work, we plan to develop new techniques aimed at reducing the energy consumed by reply messages. Our proposals are based on the observation that when a load or store misses at the L1 cache, not all the memory block is needed to allow the load or store to proceed, just the requested word. In this way, dynamically adjusting the size of the memory blocks would lead to reductions in energy consumption without degrading performance.

References

1. Austin T, Larson E, Ernst D (2002) SimpleScalar: an infrastructure for computer system modeling. Computer 35(2):59–67
2. Balasubramonian R, Muralimanohar N, Ramani K, Venkatachalapathy V (2005) Microarchitectural wire management for performance and power in partitioned architectures. In: Proc of the 11th int'l symp on high-performance computer architecture (HPCA-11), pp 28–39
3. Banerjee K, Mehrotra A (2002) A power-optimal repeater insertion methodology for global interconnects in nanometer designs. IEEE Trans Electron Devices 49(11):2001–2007
4. Binkert NL, Dreslinski RG, Hsu LR, Lim KT, Saidi AG, Reinhardt SK (2006) The M5 simulator: modeling networked systems. IEEE Micro 26(4):52–60
5. Brooks D, Tiwari V, Martonosi M (2000) Wattch: a framework for architectural-level power analysis and optimizations. In: Proc of the 27th int'l symp on computer architecture (ISCA-27), pp 83–94
6. Chen J, Dubois M, Stenström P (2003) Integrating complete-system and user-level performance/power simulators: the SimWattch approach. In: Proc of the 2003 IEEE int'l symp on performance analysis of systems and software
7. Cheng L, Muralimanohar N, Ramani K, Balasubramonian R, Carter J (2006) Interconnect-aware coherence protocols for chip multiprocessors. In: Proc of the 33rd int'l symp on computer architecture (ISCA-33), pp 339–351
8. Fernandez-Pascual R, Garcia JM (2005) RSIM x86: a cost effective performance simulator. In: Proc of the 19th European conf on modelling and simulation
9. Flores A, Aragón JL, Acacio ME (2007) Sim-PowerCMP: a detailed simulator for energy consumption analysis in future embedded CMP architectures. In: Proc of the 4th int'l symp on embedded computing (SEC-4)
10. Hughes CJ, Pai VS, Ranganathan P, Adve SV (2002) RSIM: simulating shared-memory multiprocessors with ILP processors. IEEE Comput 35(2):40–49
11. Kahle JA, Day MN, Hofstee HP, Johns CR, Maeurer TR, Shippy D (2005) Introduction to the cell multiprocessor. IBM J Res Dev 49(4/5):589–604
12. Kim C, Burger D, Keckler SW (2002) An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In: Proc of the 10th int'l conf on architectural support for programming languages and operating systems (ASPLOS-10), pp 211–222
13. Kumar R, Zyuban V, Tullsen DM (2005) Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling. In: Proc of the 32nd int'l symp on computer architecture (ISCA-32), pp 408–419
14. Liu C, Sivasubramaniam A, Kandemir M (2004) Organizing the last line of defense before hitting the memory wall for CMPs. In: 10th int'l symp on high performance computer architecture (HPCA-10), pp 176–185

15. Magen N, Kolodny A, Weiser U, Shamir N (2004) Interconnect-power dissipation in a microprocessor. In: Proc of the 2004 int'l workshop on system level interconnect prediction (SLIP'04), pp 7–13
16. Martin MMK, Sorin DJ, Beckmann BM, Marty MR, Xu M, Alameldeen AR, Moore KE, Hill MD, Wood DA (2005) Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Comput Archit News 33(4):92–99
17. Monchiero M, Canal R, Gonzalez A (2006) Design space exploration for multicore architectures: a power/performance/thermal view. In: ICS '06: proc of the 20th int'l conf on supercomputing, pp 177–186
18. Renau J SESC Website http://sourceforge.net/projects/sesc
19. Shang L, Peh L, Jha N (2003) Dynamic voltage scaling with links for power optimization of interconnection networks. In: Proc of the 9th int'l symp on high-performance computer architecture (HPCA-9), pp 91–102
20. Shivakumar P, Jouppi NP (2001) Cacti 3.0: an integrated cache timing, power and area model. Technical report, Western Research Lab (WRL)
21. Singh J, Weber W-D, Gupta A (1992) SPLASH: Stanford parallel applications for shared-memory. Comput Archit News 20(1):5–44
22. Sohi GS (1990) Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers. IEEE Trans Comput 39(3):349–359
23. Taylor MB, Kim J, Miller J, Wentzlaff D, Ghodrat F, Greenwald B, Hoffman H, Johnson P, Lee J-W, Lee W, Ma A, Saraf A, Seneski M, Shnidman N, Strumpen V, Frank M, Amarasinghe S, Agarwal A (2002) The Raw microprocessor: a computational fabric for software circuits and general-purpose programs. IEEE Micro 22(2):25–35
24. Wang H, Peh L-S, Malik S (2003) Power-driven design of router microarchitectures in on-chip networks. In: Proc of the 36th int'l symp on microarchitecture (MICRO-36), pp 105–111
25. Wang H-S, Peh L-S, Malik S (2003) A power model for routers: modeling alpha 21364 and Infiniband routers. IEEE Micro 23(1):26–35
26. Wang H-S, Zhu X, Peh L-S, Malik S (2002) Orion: a power-performance simulator for interconnection networks. In: Proc of the 35th int'l symp on microarchitecture (MICRO-35), pp 294–305
27. Woo SC, Ohara M, Torrie E, Singh JP, Gupta A (1995) The SPLASH-2 programs: characterization and methodological considerations. In: Proc of the 22nd int'l symp on computer architecture (ISCA-22), pp 24–36
28. Zhang M, Asanovic K (2005) Victim replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors. In: Proc of the 32nd int'l symp on computer architecture (ISCA-32), pp 336–345
29. Zhang Y, Parikh D, Sankaranarayanan K, Skadron K, Stan M (2003) HotLeakage: a temperature-aware model of subthreshold and gate leakage for architects. Technical report, University of Virginia
30. Zhao L, Iyer R, Makineni S, Moses J, Illikkal R, Newell D (2007) Performance, area and bandwidth implications on large-scale CMP cache design. In: Proc of the 1st workshop on chip multiprocessor memory systems and interconnects (CMP-MSI'07). In conjunction with HPCA-13

Antonio Flores received his M.S. degree in Computer Science in 1994 from the Univesidad de Murcia, Spain. He is now a Ph.D. student and Assistant Professor in the Computer Engineering Department at the Universidad de Murcia. His research interests include CMP architectures, processor microarchitecture, and poweraware cache-coherence protocol design.

Juan L. Aragón received his M.S. degree in Computer Science in 1996 and his Ph.D. degree in Computer Engineering in 2003, both from the Universidad de Murcia, Spain, followed by a 1-year postdoctoral stay as a Visiting Assistant Professor and Researcher in the Computer Science Department at the University of California, Irvine. In 1999 he joined the Computer Engineering Department at the Universidad de Murcia, where he currently is an Associate Professor. His research interests are focused on CMP architectures, processor microarchitecture, and energy-efficient and reliable systems. Dr. Aragón is a member of the IEEE Computer Society.

Manuel E. Acacio received the M.S. and Ph.D. degrees in computer science from the Universidad de Murcia, Spain, in 1998 and 2003, respectively. He joined the Computer Engineering Department at the Universidad de Murcia, in 1998, where he is currently an Associate Professor of computer architecture and technology. His research interests include prediction and speculation in multiprocessor memory systems, hardware transactional memory, multiprocessor-on-a-chip architectures, and power-aware and fault-tolerant cache-coherence protocol design.