# Address Compression and Heterogeneous Interconnects for Energy-Efficient High-Performance in Tiled CMPs

Antonio Flores, Manuel E. Acacio and Juan L. Aragón
Departamento de Ingeniería y Tecnología de Computadores
University of Murcia, 30100 Murcia, Spain
{aflores, meacacio, jlaragon}@ditec.um.es

## Abstract

*Previous studies have shown that the interconnection network of a Chip-Multiprocessor (CMP) has significant impact on both overall performance and energy consumption. Moreover, wires used in such interconnect can be designed with varying latency, bandwidth and power characteristics. In this work, we present a proposal for performance- and energy-efficient message management in tiled CMPs that combines both address compression with a heterogeneous interconnect. Our proposal consists of applying an address compression scheme that dynamically compresses the addresses within coherence messages allowing for a significant area slack. The arising area can be exploited for wire latency improvement by using a heterogeneous interconnection network comprised of a small set of very-low-latency wires for critical short-messages in addition to baseline wires. Detailed simulations of a 16-core CMP show that our proposal obtains average improvements of 10% in execution time and 38% in the Energy-Delay$^2$ Product of the interconnect.*

## 1 Introduction

High performance processor designs have evolved toward architectures that implement multiple processing cores on a single die (CMPs) as implementation technology scales down. 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 an on-chip switched direct network [24].

In CMP architectures, the design of the on-chip interconnection network has been shown to have 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.* [22] reported that the on-chip network of the Raw processor consumes 36% of the total chip power. Magen *et al.* [17] also attribute 50% of overall chip power to the interconnect. Most of this power is dissipated in the point-to-point links of the interconnect [22]. Thus, wires pose major performance and power dissipation problems as technology shrinks and total die area increases. This trend will be exacerbated in future dense CMP designs.

One way to contain problems due to wire delay and power dissipation is the use of links that are comprised of wires with varying physical properties. By tuning wire width and spacing, it is possible to design wires with varying latency and bandwidth properties. Similarly, by tuning repeater size and spacing, it is possible to design wires with varying latency and energy properties [2]. This results in what has been called *heterogeneous on-chip interconnection networks* [1].

Another approach to alleviate wire delays and power is to *transcode* the transferred information in order to better exploit the interconnect bandwidth. The area slack created due to compression can be exploited for further improving the latency of some particular wires. This paper explores such an approach by proposing the use of an address compression scheme in the context of a heterogeneous interconnect that allows most of the critical messages, used to ensure coherence between the L1 caches of a CMP, to be compressed in a few bytes and transmitted using very low latency wires meanwhile the rest of messages are transmitted using baseline wires. It is important to note that this work is not aimed at proposing a particular compression scheme but at exploiting the area slack due to compression for improving critical messages latency by means of using a heterogeneous interconnect. Detailed simulations of a 16-core CMP show average improvements of 10% in execution time and 38% in the Energy-Delay$^2$ Product of the interconnect (26% of the full CMP) when such a heterogeneous interconnection network is used in conjunction with an address compression scheme.

0190-3918/08 $25.00 © 2008 IEEE
DOI 10.1109/ICPP.2008.33

The rest of this paper is organized as follows. We review related work in Section 2. Section 3 presents a brief background of address compression mechanisms and reviews some techniques that enable different wire implementations and the design of a heterogeneous interconnect. Our proposal for optimizing the on-chip interconnection network energy and performance in tiled CMPs is presented in Section 4. Section 5 describes the evaluation methodology and presents the results of the proposed mechanism. Finally, Section 6 summarizes the main conclusions of the work.

## 2 Related Work

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.* [14] analyze several on-chip interconnection mechanisms and topologies, and quantify their area, power, and latency overheads. The study concludes that the design choices for the interconnect have a significant effect on the rest of the chip, potentially consuming a significant fraction of the real estate and power budget.

Several address compression schemes have been proposed to reduce their energy and/or area cost. Address compression schemes based on using a small compression cache were first proposed in [8] and were used to reduce off-chip bus widths and pin counts. Recent attempts at using compression to reduce on-chip wire delay and/or energy consumption have been presented in the last years. In [7], Cripton proposes to exploit low entropy in order to reduce wire delay. However, in that work, performance impact results are based on estimations instead of using simulations. Liu *et al.* [16] propose an evolution of the mechanism described in [8]. They describe partial match compression (PMC) scheme for the address bus between the L1 and L2 caches in a single-core processor executing sequential applications in order to reduce wire delay by increasing inter-wire spacing. Performance improvement of up to 16% are obtained for some configurations, however, they don't estimate the impact of their proposal in the energy consumption of the processor.

Basu *et al.* [3] propose placing a value cache at both ends of a communication channel. Upon a *hit*, the system sends the index to the cache entry instead of the whole word, to reduce bit transitions. Parcerisa and González [19] applied value prediction to inter-cluster communication on clustered microarchitectures in order to reduce long wire delays. In [15], the authors propose a communication value cache (CVC) to reduce the number of bit transfers between processors inside a bus-based CMP microarchitecture. Differently from previous works, we show how address compression could be used in conjunction with a heterogeneous network to improve performance and reduce energy consumption in tiled CMP architectures.

On the other hand, 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. Beckmann and Wood [4] propose the use of transmission lines to access large L2 on-chip caches in order to reduce the required cache area and the dynamic power consumption of the interconnection network. In [1], 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. In particular, cache accesses are accelerated by sending a subset of the address bits on low-latency wires to prefetch data out of the L1 D-cache, while non-critical register values are transmitted on low-power wires. They extend this proposal in [18] with techniques aimed at accelerating cache accesses in large L2/L3 split caches (L2/L3 NUCA architectures [13]) by taking advantage of a lower-bandwidth, lower-latency network.

Recently, Cheng *et al.* [6] 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 2D mesh typically employed in tiled CMPs).

More recently, we have proposed in [9] *Reply Partitioning*, a technique that allows all coherence messages to be classified into two groups: critical and short, and non-critical and long. In particular, *Reply Partitioning* focuses on replies that carry data and splits them into a critical and short *Partial Reply* message that carries the word requested by the processor, in addition to a non-critical *Ordinary Reply* with the full cache block. *Reply Partitioning* aims at using a heterogeneous interconnection network comprised of low-latency wires for critical messages and low-energy wires for non-critical ones, which also allows for a more balanced workload. Note that the proposal here presented is orthogonal to that, and could be used to accelerate even more the low-latency wires.

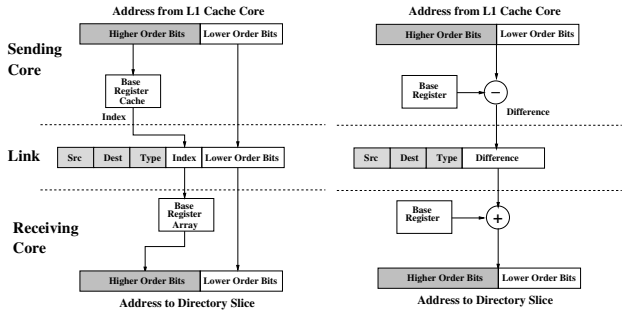


**Figure 1. Organization of the DBRC (left) and Stride (right) address compression schemes for tiled CMP architectures**

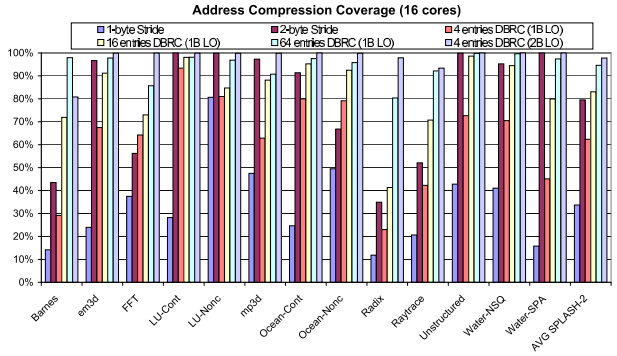


**Figure 2. Address compression coverage for a 16-core tiled CMP**

# 3 Preliminaries

## 3.1 Address Compression Schemes

Address buses have been studied widely in previous works and several strategies have been adopted in order to eliminate redundant and/or useless information. Dynamic compression schemes were first proposed by Farrens *et al.* [8]. Their dynamic base register caching (DBRC) scheme consists of a small compression cache at the sending end, and register files at the receiving end of a processor-memory address bus. When a *hit* happens in the sender compression cache, the system sends the index to the cache entry instead of the whole address, reducing the number of wires needed to implement the address bus. Figure 1 (left) shows how to adapt the DBRC scheme for a tiled CMP architecture in which the global bus is replaced by a switched direct network.

An alternative compression scheme is shown in Figure 1 (right). In this case, the compression cache at the sending end is replaced with a base register that stores the last non-compressed address sent by the source core to that destination. At the receiving end, a similar base register also stores that address. When the *difference* between the address stored at the sending end and a subsequent address can be represented using less than a fixed number of bits, the system sends just the difference instead of the whole address, updating the information stored in the base register in both the sending and the receiving cores. This simple scheme is based in the fact that many memory references, beyond affine access functions, can be easily predictable as they follow a stride pattern [20].

In order to evaluate the effectiveness of both compression schemes in the context of a tiled CMP running parallel applications, Figure 2 shows the fraction of compressed addresses using different configurations for both schemes (see Section 5 for further details about the 16-core CMP configuration and working sets evaluated). Some interesting results can be pointed out. First, if we try to keep address compression size to one byte, a low compression coverage is obtained for both the Stride compression scheme, and the DBRC scheme when a small compression cache is considered (1-byte Stride and 4 entries DBRC (1B LO) bars). In order to obtain acceptable compression coverages (over 80%), we need either to implement a compression cache with, at less, 16 entries (16 entries DBRC (1B LO) bar); or to use an address compression size of two bytes (2-byte Stride or 4 entries DBRC (2B LO) bars). In the last case, the Stride compression scheme obtains similar results to an equivalent DBRC configuration eliminating the need of using an adder. These results show that, in this case, compression schemes based on the use of stride patterns perform badly in comparison with other proposals. Finally, DBRC with a compression size of 2 bytes shows average address compression coverages of around 98%. These results show that traditional compression schemes provide significant coverage that can be exploited.

It is important to note that this work is not aimed at proposing a particular compression scheme but at exploiting the area slack due to compression for improving critical messages latency and ensuring efficient message management in CMPs by means of using a heterogeneous interconnect.

In order to evaluate the hardware cost of the address compression schemes for a tiled CMP architecture, Table 1 shows the area and power characteristics of different configurations of these two address compression schemes along with the percentage relative to one of the cores of the CMP. The measurements have been carried out using CACTI v4.1 [21] for a 65 $nm$ process technology. In each core, we need to implement one sending structure and as many receiving structures as the number of cores. Moreover, requests and coherence commands use their own hardware

**Table 1. Area and power characteristics of different address compression schemes for a 16-core tiled CMP**

| Compression Scheme | Size (Bytes) | Area ($mm^2$) | Max. Dyn. Power (W) | Static Power (mW) |
|---|---|---|---|---|
| 4-entry DBRC | 1088 | 0.0723 (0.29%) | 0.1065 (0.48%) | 10.78 (0.29%) |
| 16-entry DBRC | 4352 | 0.2678 (1.07%) | 0.3848 (1.72%) | 43.03 (1.21%) |
| 64-entry DBRC | 17408 | 0.8240 (3.30%) | 0.7078 (3.16%) | 133.42 (3.76%) |
| 2-byte Stride | 272 | 0.0257 (0.1%) | 0.0561 (0.25%) | 5.14 (0.15%) |

structures to avoid destructive interferences between both address streams. It can be seen that the relative cost of implementing these compression schemes with respect to a core ranges from 0.1-0.25% for the 2-byte Stride scheme to 3.3-3.7% for the 64-entry DBRC one.

### 3.2 Wire Implementation

The delay of a wire can be modeled as a first-order RC circuit [11]. In that model, a CMOS driver is seen as a simple resistor, $R_{gate}$, with a parasitic load, $C_{diff}$ as shown in Equation 1. The CMOS receiver at the other end of the wire presents a capacitive load $C_{gate}$. $C_{wire}$ and $R_{wire}$ are the wire resistance and capacitance, respectively.

$$Delay \propto R_{gate}(C_{diff} + C_{wire} + C_{gate}) + R_{wire}(\frac{1}{2}C_{wire} + C_{gate}) \quad (1)$$

The resistance per unit length of the wire, $R_{wire}$, depends on the geometrical dimensions of the wire cross-section. Increasing the width of the wire can significantly decrease its resistance although a modest increase in $C_{wire}$ is produced. Similarly, increasing the spacing between adjacent wires results in a $C_{wire}$ drop. Combining both factors, we can design wires with lower delays.

Furthermore, the delay of an uninterrupted wire grows quadratically with its length. Therefore, for long wires, designers must insert repeaters periodically along the wire to break this quadratic dependence of wire delay on the wire length. As repeaters divide a long wire into multiple shorter segments of length $l$, the total wire delay is the number of segments multiplied by the individual segment delay. Each segment's delay is still quadratically dependent on its segment length, but the total wire delay is now linear with total length. Overall wire delay can be minimized by selecting optimal repeater sizes and spacing between repeaters, being a commonly employed technique nowadays.

For a global interconnect of length $L$, the total power dissipation is

$$P_{line} = nP_{repeater} = n(P_{switching} + P_{leakage}) \quad (2)$$

where $n = L/l$ is the number of repeaters for that line.

**Table 2. Area, delay, and power characteristics of wire implementations (from [6])**

| Wire Type | Relative Latency | Relative Area | Dynamic Power (W/m) $\alpha$=Switching Factor | Static Power W/m |
|---|---|---|---|---|
| B-Wire (8X plane) | $1x$ | $1x$ | $2.65\alpha$ | 1.0246 |
| B-Wire (4X plane) | $1.6x$ | $0.5x$ | $2.9\alpha$ | 1.1578 |
| L-Wire (8X plane) | $0.5x$ | $4x$ | $1.46\alpha$ | 0.5670 |
| PW-Wire (4X plane) | $3.2x$ | $0.5x$ | $0.87\alpha$ | 0.3074 |

The dynamic power dissipated driving the wire segment with activity factor $\alpha$ is

$$P_{switching} = \alpha(s(C_{gate} + C_{diff}) + lC_{wire})fV_{DD}^2 \quad (3)$$

where $V_{DD}$ is the power supply voltage; $f$ is the clock frequency and $s$ is the size of the repeaters.

The average leakage power of a repeater is given by

$$P_{leakage} = V_{DD}I_{leakage} = V_{DD}\frac{1}{2}(I_{off_N}W_{N_{min}} + I_{off_P}W_{P_{min}})s \quad (4)$$

where $I_{off_N}$ ($I_{off_P}$) is the leakage current per unit NMOS (PMOS) transistor width and $W_{N_{min}}$ ($W_{N_{min}}$) is the width of the NMOS (PMOS) transistor in minimum size inverter.

Equations (3) and (4) show that the dissipated power can be reduced by employing smaller repeaters and by increasing their spacing. Banerjee *et al.* [2] developed a methodology to estimate repeater size and spacing that minimizes power consumption for a fixed wire delay.

In summary, by varying some physical properties such as wire width/spacing and repeater size/spacing, we can implement wires with different latency, bandwidth and power properties. As previously mentioned, in [6], the authors apply this observation to develop a heterogeneous interconnect. They propose to use two wire implementations apart from baseline wires (*B-Wires*): power optimized wires (*PW-Wires*) that have fewer and smaller repeaters, and bandwidth optimized wires (*L-Wires*) with higher widths and spacing. Then, coherence messages are mapped to the appropriate set of wires taking into account, among others, their latency and bandwidth requirements.

Table 2 shows the relative delay, area, and power characteristics of *L-* and *PW-Wires* compared to baseline wires (*B-Wires*), as reported in [6]. A 65 $nm$ process technology is considered assuming 10 metal layers: 4 layers in 1X plane, and 2 layers in each 2X, 4X, and 8X planes [14]. 4X and 8X metal planes are used for global inter-core wires. It can be seen that *L-Wires* yield a two-fold latency improvement at a four-fold area cost. On the other hand, *PW-Wires* are designed to reduce power consumption with twice the delay of baseline wires (and the same area cost). As in [14], it is assumed that 4X and 8X wires are routed over memory arrays.

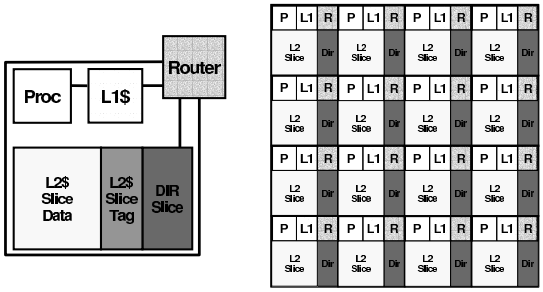


**Figure 3. Tiled CMP architecture overview**

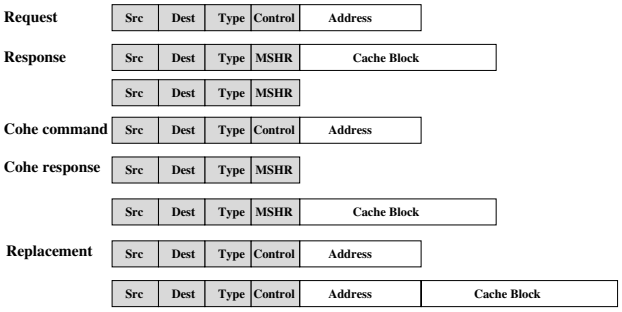


**Figure 4. Classification of messages that travel on the interconnection network of a tiled CMP architecture**

# 4 A Proposal for Efficient Message Management in Tiled CMPs

In this section we present our proposal for reduced energy consumption in tiled CMPs. As introduced before, there are two main components in our proposal. The first is the use of an address compression scheme that allows for a significant area slack. The second is the use of a heterogeneous interconnect that exploits that area slack for wire latency improvement by using two different set of wires. Messages are sent through the appropriate set, according to their latency and bandwidth requirements. This section starts with a description of the tiled CMP architecture assumed in this paper, followed by a classification of the messages in terms of both their criticality and size and, finally, the description of the proposed mechanism.

## 4.1 Tiled CMP Architectures

A tiled CMP architecture consists of a number of replicated *tiles* connected over a switched direct network (Figure 3). 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. 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 [13]). 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 $nm$, a tile area of approximately 25 $mm^2$, and a die size in the order of 400 mm$^2$ [24, 26]. Note that this area is similar to the largest die in production today (Itanium 2 processor – around 432 mm$^2$). Note also that, due to manufacturing costs and form factor limitations, it would be desirable to keep die size as low as possible [26]. Further details about the evaluation methodology and the simulated CMP configuration can be found in Section 5.

## 4.2 Classification of messages in Tiled CMP Architectures

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 (see Figure 4): *Request messages*, that are generated by cache controllers in response to L1 cache misses and are sent to the corresponding home L2 cache to demand privileges over a memory line. *Response messages* to these requests, generated by the home L2 cache controller or, alternatively, by the remote L1 cache that has the single valid copy of the data, and that can carry the memory line or not. *Coherence commands*, that are sent by the home L2 cache controller to the corresponding L1 caches to ensure coherence. *Coherence responses*, sent by the L1 caches back to the corresponding home L2 in response to coherence commands. *Replacement messages*, that the L1 caches generate in case of exclusive or modified lines being replaced (replacement hints are not sent for lines in shared state).

Messages involved in the L1 cache coherence protocol shown in Figure 4 can be classified according to their criticality into critical and non-critical messages. We say that a message is critical when it is in the critical path of the L1 cache miss. In other case, we call the message as non-critical. As an example, the coherence actions involved in a L1 read miss for a line in modified state in other tile consist of: (1) a request message that is sent down to the appropriate directory tile (the home L2 cache); (2) an intervention

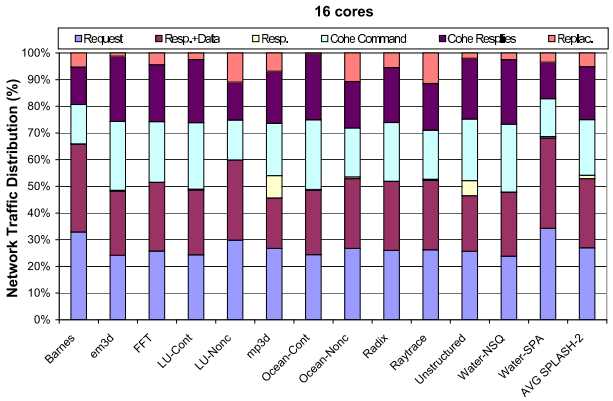


**Figure 5. Breakdown of the messages that travel on the interconnection network for a 16-core CMP**

message sent to the owner tile that (3a) sends back the line to the requestor and (3b) to the directory tile. Whereas messages (1), (2) and (3a) are critical because they belong to the critical path between the processor request and the memory system response, (3b) is non-critical. Using this criterion, we can observe that all message types but replacement messages and some coherence replies (such as revision messages) are critical. It is clear that performance is increased if critical messages are sent through low-latency *L-Wires* (assuming that there are enough wires).

On the other hand, coherence messages can also be classified according to their size into short and long messages. Coherence responses do not include the address or the data block and just contain control information (source/destination, message type, MSHR id, etc). Therefore, they are classified as short messages. Other message types, in particular requests, responses without data and coherence commands, also contain address block information but they are still narrow enough to be classified as short messages. They will be the focus of our proposal. Finally, replacements with data and data block transfers also carry a cache line and, therefore, they are classified as long messages.

Figure 5 plots the fraction of each message type on the total number of messages for a 16-core CMP configuration for the applications used in our evaluation (see Section 5.1 for evaluation details). 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 more than 50% of the messages are short messages containing address block information that can be compressed.

Previous work [6], [9] has shown that the use of a heterogeneous interconnect comprised of low-latency *L-Wires* and power-efficient *PW-Wires* allows for more energy-efficient interconnect utilization. However, since the number of *L-Wires* is small because of their four-fold area cost (relative to baseline wires) only short messages can take full advantage of them.

**Table 3. Relative delay, area, and power characteristics of VL-Wires (8X plane) related to baseline wires for different widths**

| Wire Width | Relative Latency | Relative Area | Dynamic Power (W/m) $\alpha$=Switching Factor | Static Power W/m |
|---|---|---|---|---|
| 3 Bytes | $0.27x$ | $14x$ | $0.87\alpha$ | 0.3065 |
| 4 Bytes | $0.31x$ | $10x$ | $1.00\alpha$ | 0.3910 |
| 5 Bytes | $0.35x$ | $8x$ | $1.13\alpha$ | 0.4395 |

### 4.3 Interconnect Design for Efficient Message Management

In this work, we use the same main parameters for the interconnect as in [6, 9]. In particular, message sizes and the width of the original links of the interconnect are the same. Short messages can take up to 11 bytes. Requests, coherence commands are 11-byte long since beside control information (3 bytes) they also carry address information. On the other hand, coherence replies and replacements without data are just 3-byte long. Finally, ordinary reply messages are 67-byte long since they carry control information (3 bytes) and a cache line (64 bytes).

As discussed in the previous section, *L-Wires* have a four-fold area cost compared to baseline wires and, therefore, the number of *L-Wires* is quite limited. Considering that they will be used for sending short, critical messages, previous proposals have fixed their amount according to the typical size of short messages (e.g., 11 bytes). The remaining area have been employed for sending long messages. By using an address compression scheme the amount of *L-Wires* can be reduced dramatically, from 11 bytes to 4-5 bytes depending on the size of the uncompressed low order bits used by the underlying compression scheme, arising an area slack than can be exploited for further improving wire latency. We denote this new kind of wires as *VL-Wires*.

Table 3 shows the relative delay, area, and power characteristics of *VL-Wires* for different wire widths compared to baseline wires (*B-Wires*) when a single metal plane is considered (8X plane). In order to match the metal area of the baseline configuration, in our proposal, each original 75-byte unidirectional link is designed to be made up of 24 to 40 *VL-Wires* (3 to 5 bytes) with different relative latencies, as shown in Table 3, and 272 *B-Wires* (34 bytes). *VL-Wires* will be used for sending already short, critical messages (e.g., coherence replies) as well as *compressed* requests and *compressed* coherence commands. Uncompressed and long

**Table 4. Configuration of the evaluated baseline CMP architecture and applications**

| CMP Configuration | |
|---|---|
| *Parameter* | |
| Process technology | 65 $nm$ |
| Tile area | 25 $mm^2$ |
| Number of tiles | 16 |
| Cache line size | 64 bytes |
| Core | 4GHz, in-order 2-way model |
| L1 I/D-Cache | 32KB, 4-way |
| L2 Cache (per core) | 256KB, 4-way, 6+2 cycles |
| Memory access time | 400 cycles |
| Network configuration | 2D mesh |
| Network bandwidth | 75 GB/s |
| Link width | 75 bytes (*8X-B-Wires*) |
| Link length | 5 $mm$ |

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

messages are sent using the original *B-Wires*. For a discussion regarding the implementation complexity of heterogeneous interconnects refer to [6].

# 5 Experimental Results

This section shows the results that are obtained for our proposal under different scenarios and compares them against those achieved with the configuration that employs just *B-Wires* which is taken as baseline. In order to obtain a better understanding, we also show the behavior under the perfect address compression assumption that is represented in the figures using lines in instead of barlines.

## 5.1 Evaluation Methodology

The results presented in this work have been obtained through detailed simulations of a full CMP. We have employed a cycle-accurate CMP power-performance simulation tool, *Sim-PowerCMP* [10], that estimates both dynamic and leakage power and is based on RSIM [12]. In particular, *Sim-PowerCMP* employs as performance simulator a modified version of RSIM that models the architecture of the tiled CMP presented in Section 4. *Sim-PowerCMP* also implements already proposed and validated power models for both dynamic power (from Wattch [5], CACTI [21]) and leakage power (from HotLeakage [25]) of each processing core, as well as the interconnection network (from Orion [22]).

Table 4 (top) shows the architecture configuration used across this paper. It describes a 16-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 [24]. With this configuration, links that interconnect routers configuring the 2D mesh topology measure around 5 $mm$. Reply messages are 67-byte long since they carry both 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 (3 bytes for header, 8 bytes for the memory address), just 3-byte long for coherence replies. Table 4 (bottom) shows the applications used in our experiments. *MP3D* is from the SPLASH benchmark suite; *Barnes-Hut*, *FFT*, *LU-cont*, *LU-noncont*, *Ocean-cont*, *Ocean-noncont*, *Radix*, *Raytrace* and *Water-nsq* are from the SPLASH-2 benchmark suite; Berkeley *EM3D* simulates the propagation of electro-magnetic waves through objects in three dimensions; and *Unstructured* is a computational fluid 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 [23]. All experimental results reported in this work are for the parallel phase of these applications.

## 5.2 Simulation results and analysis

In this section, we analyze, first, the impact of our proposal on the execution time and on the energy-delay$^2$ product metric for the inter-core links. Then, we evaluate the energy-delay$^2$ product metric for the full CMP. All results have been normalized with respect to the baseline configuration where only *B-Wire*, unidirectional 75-byte wide links are considered and they include the additional cost due to the extra hardware structures needed to implement the different address compression schemes evaluated.

Figure 6 (top) depicts the normalized execution time with respect to that obtained for the baseline configuration for a 16-core CMP. Barlines show the normalized execution time for several Stride and DBRC address compression schemes, in particular those with a compression coverage over 80% as reported in Figure 2. The number of bytes used to send the low order bits (1 or 2 bytes) determines the number of *VL-Wires* in the heterogeneous network (4 or 5 bytes). For comparison purposes and in order to show the potential improvement in the execution time, three additional solid lines have been added to show the execution time for the different heterogeneous network configurations when a perfect compression coverage is considered. It can

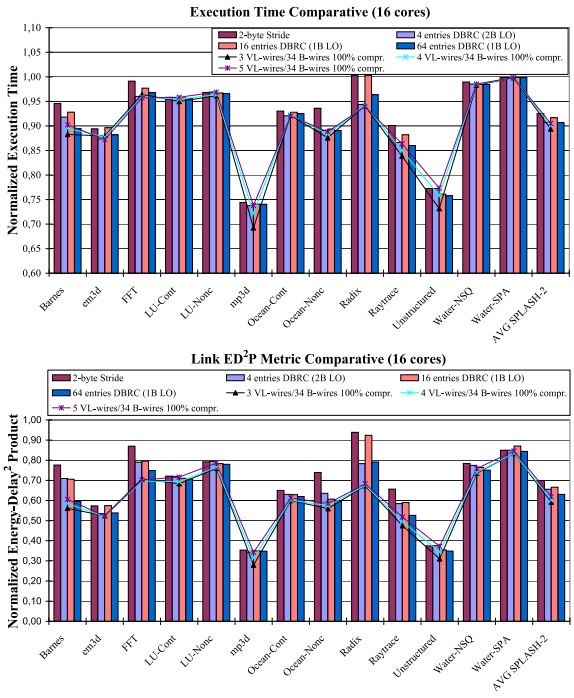


**Figure 6. Normalized execution time (top) and link $ED^2P$ metric (bottom) for different address compression schemes over heterogeneous links (VL-Wires width matches address compression size)**

be observed that a 4-entry DBRC compression scheme with 2 low-order bytes is enough to achieve an average performance improvement of 8% (close to 10% of maximum potential performance improvement). This improvement has high variability, ranging from almost 1-2% for Water and LU to 22-25% for mp3d and Unstructured. This variability is due to two factors. First, some applications, as Water or LU, present low intercore data sharing patterns [23]. In these cases, the coherence traffic is small and our proposal has little impact in the execution time. Other applications, such as mp3d or Unstructured, present better traffic patterns and can take advantage of a faster interconnect. The second factor that explains this variability is related with the address compression scheme coverage. As it was shown in Figure 2, applications such as Barnes or Radix exhibit low address compression coverage for most of the configurations proposed. For these applications, the reduction in the execution time does not match the maximum potential even when a 64-entry configuration is used.

Figure 6 (bottom) plots the normalized energy-delay$^2$ product ($ED^2P$) metric. Average reductions close to 30% are obtained, although again, a high variability among applications is observed. Some applications, such as Water and LU, show reductions of 20% due mainly to the lower power dissipation of the proposed heterogeneous interconnection network; others, such as mp3d and Unstructured, present a reduction of 65% in the $ED^2P$ metric due to bigger emphasis on the execution time that the $ED^2P$ metric does.

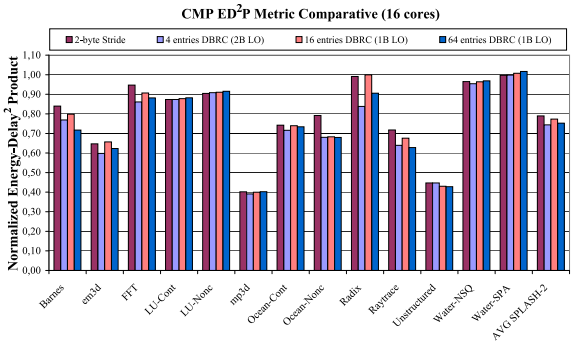


**Figure 7. Normalized energy-delay$^2$ product ($ED^2P$) for the full CMP**

Finally, Figure 7 presents the normalized $ED^2P$ metric for the full CMP. Average improvements range from 21% for the 2-byte Stride configuration to 26% for the 4-entry DBRC one. As it can be observed, when the number of entries of the DBRC compression scheme is increased, worse $ED^2P$ metrics are obtained for the full CMP. This is due to that the bigger impact that the extra hardware structures have is not compensated with a significant reduction in the execution time.

## 6 Conclusions

In this work we propose an energy and performance aware message management mechanism for tiled CMPs that consists of two components. The first one is the use of an address compression scheme that allows for a significant area slack. The second approach is a heterogeneous interconnect network that exploits the arising area slack for improving wire latency and that is comprised of only two different types of wires: *VL-Wires* for critical short messages and baseline wires for the rest of messages.

Results obtained through detailed simulations of a 16-core CMP show that the proposed on-chip message management mechanism can reduce the $ED^2P$ metric for the links of the interconnection network about 38% with an additional reduction in execution time of 10%. Finally, these reductions translate into overall CMP savings of 26% when the $ED^2P$ metric is considered. These results reveal that correctly organizing the interconnection network and properly managing the different types of messages through it

have significant impact on the energy consumed by CMPs, especially for next-generation dense CMP architectures.

## Acknowledgments

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2006-15516-C4-03".

## References

[1] R. Balasubramonian, N. Muralimanohar, et al. Microarchitectural Wire Management for Performance and Power in Partitioned Architectures. In *Proc. of the 11th Int'l Symp. on High-Performance Comp. Arch.*, 2005.

[2] K. Banerjee and A. Mehrotra. A power-optimal repeater insertion methodology for global interconnects in nanometer designs. *IEEE Trans. on Electron Devices*, 49(11):2001–2007, 2002.

[3] K. Basu, A. N. Choudhary, et al. Power protocol: reducing power dissipation on off-chip data buses. In *Proceedings of the 35th Int'l Symp. on Microarchitecture*, 2002.

[4] B. M. Beckmann and D. A. Wood. TLC: Transmission Line Caches. In *Proc. of the 36th Int'l Symp. on Microarchitecture* , 2003.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proc. of the 27th Int'l Symp. on Comp. Arch.* , 2000.

[6] L. Cheng, N. Muralimanohar, et al. Interconnect-Aware Coherence Protocols for Chip Multiprocessors. In *Proc. of the 33rd Int'l Symp. on Comp. Arch.*, 2006.

[7] D. Citron. Exploiting low entropy to reduce wire delay. *Computer Architecture Letters*, 3, 2004.

[8] M. Farrens and A. Park. Dynamic base register caching: A technique for reducing address bus width. In *Proceedings of the 18th Int'l Symp. on Comp. Arch.*, 1991.

[9] A. Flores, J. L. Aragón, and M. E. Acacio. Efficient message management in tiled cmp architectures using a heterogeneous interconnection network. In *Proc. of the 14th Int'l Conf. on High Performance Computing* , volume 4873 of *Lecture Notes in Computer Science*, pages 133–146. Springer, 2007.

[10] A. Flores, J. L. Aragón, and M. E. Acacio. An energy consumption characterization of on-chip interconnection networks for tiled cmp architectures. *The Journal of Supercomputing*, 2008.

[11] R. Ho, K.W. Mai, and M.A. Horowitz The Future Of Wires, *Proceedings of IEEE*, 89(4):490–504, 2001.

[12] C. J. Hughes, V. S. Pai, et al. RSIM: Simulating Shared-Memory Multiprocessors with ILP Processors. *IEEE Computer*, 35(2):40–49, 2002.

[13] C. Kim, D. Burger, and S. W. Keckler. 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* , 2002.

[14] R. Kumar, V. Zyuban, and D. M. Tullsen. Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling. In *Proc. of the 32nd Int'l Symp. on Comp. Arch.*, 2005.

[15] C. Liu, A. Sivasubramaniam, and M. Kandemir. Optimizing bus energy consumption of on-chip multiprocessors using frequent values. *J. Syst. Archit.*, 52(2):129–142, 2006.

[16] J. Liu, K. Sundaresan, and N. Mahapatra. Fast, performance-optimized partial match address compression for low-latency on-chip address buses. In *24th Int'l Conf. on Computer Design*, 2006.

[17] N. Magen, A. Kolodny, et al. Interconnect-power dissipation in a microprocessor. In *Proc. of the 2004 Int'l workshop on System Level Interconnect Prediction*, 2004.

[18] N. Muralimanohar and R. Balasubramonian. The Effect of Interconnect Design on the Performance of Large L2 Caches. In *3rd IBM Watson Conf. on Interaction between Architecture, Circuits, and Compilers*, 2006.

[19] J.-M. Parcerisa and A. Gonzalez. Reducing wire delay penalty through value prediction. In *Proc. of the 33rd Int'l Symp. on Microarchitecture*, 2000.

[20] Y. Sazeides and J. E. Smith. The predictability of data values. In *Proc. of the 30th Int'l Symp. on Microarchitecture*, 1997.

[21] D. Tarjan, S. Thoziyoor, and N. P. Jouppi. Cacti 4.0. Technical report, HP Laboratories Palo Alto, 2006.

[22] H.-S. Wang, X. Zhu, et al. Orion: a power-performance simulator for interconnection networks. In *Proc. of the 35th Int'l Symp. on Microarchitecture* , 2002.

[23] S. C. Woo, M. Ohara, et al. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proc. of the 22nd Int'l Symp. on Com. Arch.*, 1995.

[24] M. Zhang and K. Asanovic. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In *Proc. of the 32nd Int'l Symp. on Comp. Arch.*, 2005.

[25] Y. Zhang, D. Parikh, et al. HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects. Technical report, University of Virginia, 2003.

[26] L. Zhao, R. Iyer, et al. Performance, Area and Bandwidth Implications on Large-Scale CMP Cache Design. In *Proc. of the 1st Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2007.