Energy-Efficient Hardware Prefetching for CMPs using Heterogeneous Interconnects

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

***Abstract*—In the last years high performance processor designs have evolved toward Chip-Multiprocessor (CMP) architectures that implement multiple processing cores on a single die. As the number of cores inside a CMP increases, the on-chip interconnection network will have significant impact on both overall performance and power consumption as previous studies have shown. On the other hand, CMP designs are likely to be equipped with latency hiding techniques like hardware prefetching in order to reduce the negative impact on performance that, otherwise, high cache miss rates would lead to. Unfortunately, the extra number of network messages that prefetching entails can drastically increase the amount of power consumed in the interconnect. In this work, we show how to reduce the impact of prefetching techniques in terms of power (and energy) consumption in the context of tiled CMPs. Our proposal is based on the fact that the wires used in the on-chip interconnection network can be designed with varying latency, bandwidth and power characteristics. By using a heterogeneous interconnect, where low-power wires are used for dealing with prefetched lines, significant energy savings can be obtained. Detailed simulations of a 16-core CMP show that our proposal obtains improvements of up to 30% in the power consumed by the interconnect (15-23% on average) with almost negligible cost in terms of execution time (average degradation of 2%).**

***Keywords*-tiled chip-multiprocessor; energy-efficient architectures; prefetching; heterogeneous on-chip interconnection network; parallel scientific applications;**

I. Introduction

In the last years, processor designs have evolved toward architectures that implement multiple processing cores on a single die, commonly known as chip-multiprocessors or CMPs. Today, multi-core architectures are envisioned as the only way to ensure performance improvements after microprocessors designers have acknowledged that it is no longer efficient to rely on higher clock rates and/or exploiting greater levels of instruction-level parallelism (ILP). In this way, most of the industry would agree that multi-core is the way forward and that designs with tens of cores on the die will be a reality within this decade. As an example, Intel recently unveiled an 80-core research prototype called Polaris [1]. Additionally, future many-core CMPs with several tens (or even hundreds) of processor cores probably will be designed as arrays of replicated tiles connected over an on-chip switched direct network [2]. These tiled architectures have been claimed to provide a scalable solution for managing the design complexity, and effectively using the resources available in advanced VLSI technologies. Maybe, one of the best known examples of a tiled CMP architecture today is the already mentioned 80-core Intel's Polaris prototype [1].

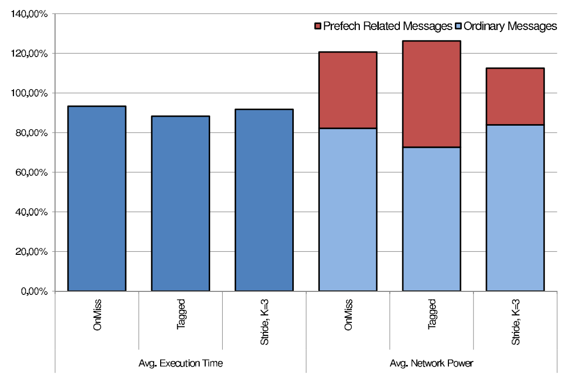

Figure 1. *Normalized execution time and network power consumption for a 16-core CMP when different prefetching techniques are considered.*

However, one of the greatest bottlenecks to provide high performance and energy efficiency in such tiled CMP architectures is the high cost of on-chip communication through global wires [3]. Wang *et al.* [4] reported that the on-chip network of the Raw processor consumes 36% of the total chip power. Magen *et al.* [5] also attribute 50% of overall chip power to the interconnect. Most of this power is consumed in the point-to-point links of the interconnect [4]. Thus, wires pose major performance and power consumption problems as technology shrinks and total die area increases. This trend will be exacerbated in future many-core CMP designs. Therefore, as communication emerges as a power and performance constraint, more important in some cases than computation itself, wire properties should be exposed to architects in order to enable them to find out novel ways to exploit these properties.

One way to tackle problems due to wire delay is to use latency hiding techniques like hardware prefetching, which eliminates some cache misses and/or overlaps the latencies of others. Unfortunately, hardware prefetching significantly increases on-chip communication since coherence between the L1 caches of the tiled CMP must be ensured, increasing the power consumption of the on-chip interconnect. As an example of the latter, Fig. 1 shows, for three hardware prefetching alternatives (see Section II-B for further details), the average improvement in terms of execution time and the increase in the on-chip network power consumption due to the extra communication that prefetching entails for the parallel scientific applications considered across this paper (see section IV-A for details about the evaluation methodology). As it can be observed, increases over 20% in the on-chip network power consumption are obtained for some of the prefetching techniques considered in this work.

Another approach to alleviate the negative effect of wire delays and the increasing interconnect power consumption is the use of heterogeneous on-chip interconnection networks [6], i.e., an interconnect with links 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 [7].

This paper explores such approach by proposing the use of a heterogeneous interconnect in the context of hardware prefetching schemes for tiled CMPs. In this way, we can improve the energy-efficiency of prefetching techniques by transmitting prefetched lines through low-power 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 prefetching scheme but at exploiting the non-critical nature of these messages by means of using the energy-efficient and slower wires of a heterogeneous interconnect. Detailed simulations of a 16-core CMP show that our proposal brings improvements of up to 30% in the power consumed by the interconnect (15-23% on average) with almost negligible cost in terms of increased execution time (average degradation of 2% with respect to the baseline configuration) when such a heterogeneous interconnection network is used in the context of a hardware prefetching scheme.

The rest of the paper is organized as follows. Section II reviews some related work and presents a background on hardware prefetching and techniques that enable different wire implementations for 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 III. Section IV describes the evaluation methodology and presents the results of the proposed mechanism. Finally, section V summarizes the main conclusions of the work and points out some future work.

II. Preliminaries

A. 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.* [8] 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 a significant effect on the rest of the chip, potentially consuming a significant fraction of the real estate and power budget.

Hardware prefetching has been proposed and explored by many researchers [9], [10], and is currently implemented in many existing systems [11], [12]. From mid-sixties, early studies [13] of cache design recognized the benefits of prefetching. Hardware prefetching of separate cache blocks was later implemented in the IBM 370/168 and Amdahl 470V [14]. Smith summarizes several of these early approaches in his survey of cache memories [15]. Jouppi [16] introduced stream buffers that trigger successive cache line prefetches on a miss. Chen and Baer [9] proposed variations of stride-based hardware prefetching to reduce the cache-to-memory latency. Dahlgren *et. al.* [17] proposed an adaptive sequential (unit-stride) prefetching scheme that adapts to the effectiveness of prefetching. Ki and Knowles [18] used extra cache bits to increase the accuracy of prefetching. Srinivasan *et. al.* [19], classified prefetches according to whether they reduce or increase misses or traffic.

On the other hand, a reduced number of works have attempted to exploit the properties of a heterogeneous interconnection network at the microarchitecture level in order to reduce the interconnect energy share. Beckmann and Wood [20] 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 [6], Balasubramonian *et al.* make the first proposal of wire management at the microarchitecture 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 [21] with techniques aimed at accelerating cache accesses in large L2/L3 split caches (L2/L3 NUCA architectures) by taking advantage of a lower-bandwidth, lower-latency network.

Recently, Cheng *et al.* [22] 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 [1]).

More recently, we have proposed in [23] *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 split 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

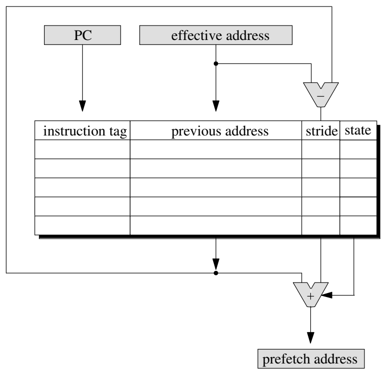

Figure 2. The organization of the reference prediction table (extracted from [24]).

to use 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 presented in the current paper is orthogonal to that and both can be used in conjunction.

B. Hardware Prefetching

As mentioned before, hardware prefetching has been proposed and explored by many researchers, and is currently implemented in many existing systems [11], [12]. In this section we present a general background on hardware prefetching focusing on the schemes that we have chosen to evaluate our proposal. A more exhaustive study about data prefetching can be found in [24].

The simplest hardware prefetching schemes are variations upon the one block lookahead (OBL) approach which initiates a prefetch for block $b+1$ when block b is accessed. Smith [15] summarizes several of these approaches of which the prefetch-on-miss ($OnMiss$) and tagged prefetch algorithms ($Tagged$) will be discussed here. The prefetch-on-miss algorithm simply initiates a prefetch for block $b+1$ whenever an access for block b results in a cache miss. The tagged prefetch algorithm associates a tag bit with every memory block. This bit is used to detect when a block is demand-fetched or a prefetched block is referenced for the first time. In either of these cases, the next sequential block is fetched. In order to avoid memory stalls suffered by processors, it is possible to increase the number of blocks prefetched after a demand fetch from one to K, where K is known as the *degree of prefetching*.

The main problem with the above techniques is that they are not able to detect memory access patterns. To solve this problem several techniques have been proposed which employ special logic to monitor the processor's address referencing pattern to detect memory access patterns originating from looping structures [9], [17], [18]. This is accomplished by comparing successive addresses used by load or store instructions. A reference prediction table (RPT) is used to keep this information for the most recently used memory instructions. The organization of the RPT is depicted in Fig. 2. Table entries contain the address of the memory instruction, the previous address accessed by this instruction, a stride value for those entries which follow a stride pattern and a state field which records the entry's current state.

C. Wire Implementation for Heterogeneous Interconnects

The delay of a wire can be modeled as a first-order RC circuit [3]. 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 wire's characteristics such as wire width or spacing between adjacent wires. 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.

The average leakage power of a repeater is given by

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

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.

On the other hand, the dynamic power consumption driving the wire segment with activity factor α 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.

Equations (2) and (3) show that the dissipated power can be reduced by employing smaller repeaters and by increasing their spacing. Banerjee *et al.* [7] 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 [22], 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 I shows the relative delay, area, and power characteristics of *L-* and *PW-Wires* compared to baseline wires (*B-Wires*), as reported in [22]. A 65 nm process technology is considered assuming 10 metal layers: 4

Table I
AREA, DELAY, AND POWER CHARACTERISTICS OF WIRE IMPLEMENTATIONS (EXTRACTED FROM [22]).

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

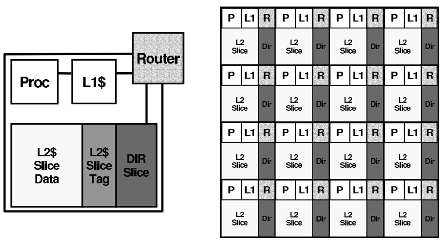

Figure 3. *Tiled CMP architecture overview.*

layers in 1X plane, and 2 layers in each 2X, 4X, and 8X planes [8]. 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 [8], it is assumed that 4X and 8X wires are routed over memory arrays.

III. A PROPOSAL FOR ENERGY-EFFICIENT PREFETCHING MANAGEMENT IN TILED CMPS

In this section we present our proposal for reducing the energy dissipated by prefetching techniques in tiled CMPs. 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.

A. Tiled CMP Architectures

A tiled CMP architecture consists of a number of replicated *tiles* connected over a switched direct network (Fig. 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. 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 [2], [25]. 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 [25]. Further details about the evaluation methodology and the simulated CMP configuration can be found in section IV.

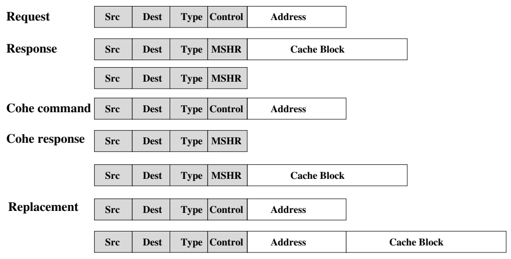

Figure 4. Classification of messages that travel on the interconnection network of a Tiled CMP Architecture.

B. 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 Fig. 4): *Request messages*, that are generated by cache controllers in response to L1 cache misses, or a likely future L1 cache miss when prefetching is considered, and 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 they 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 Fig. 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 expected, delaying a critical message will result in longer L1 cache miss latencies. On the other hand, slight slowdowns in the delivery of non-critical messages will not cause any performance degradation. Using this criterion, all messages related with prefeching are non-critical because they deal with data blocks that will be needed in the future. It is clear that energy is saved, theoretically without affecting performance, when this kind of messages travel on slower, power-efficient *PW-Wires*. They will be the focus of our proposal.

Fig. 5 (top) plots the fraction of each message type on the total number of messages for a 16-core CMP configuration when different prefetching mechanisms are considered and for the applications used in our evaluation

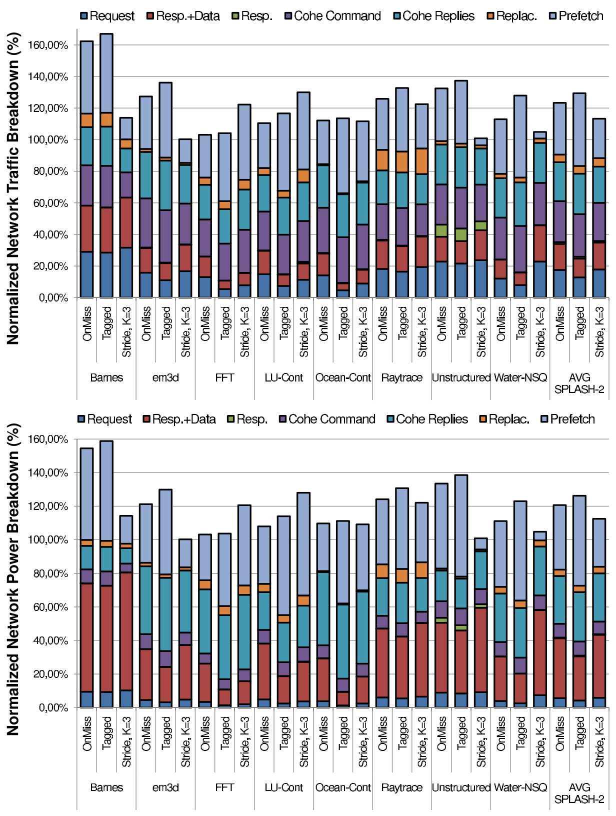

Figure 5. Breakdown of the messages that travel on the interconnection network for a 16-core CMP (top) and percentage of the power consumption in the interconnect by each type of message (bottom).

(see section IV-A for evaluation details). Results have been normalized with respect to a base configuration without prefetching. As pointed out before, hardware prefetching significantly increases on-chip communication. Average increases of about 20% in the network traffic are observed. And, on average, between 16% to 34% of the network traffic is due to prefetching (prefetch requests, its corresponding replies and all coherence traffic involved), whereas the rest has to do with ordinary messages.

Even more interesting is Fig. 5 (bottom) which shows a breakdown of the network power consumption for each message type. Again, results are normalized with respect to the network power consumption when no prefetching technique is used. The amount of power consumed in the interconnect associated with prefetching traffic ranges from 17-18% when the more complex stride prefetching is used, to 32-40% for the simplest schemes.

As previously commented, most of this power is consumed in the point-to-point links, and therefore, message size plays a major role. In particular, prefetch replies 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 (8 bytes). Finally, coherence replies are just 3-byte long. Therefore, optimizing the delivery of the prefetch replies that carry data will be rewarding to reduce the energy dissipated by the interconnection network in CMPs.

C. *Interconnect Design for Efficient Message Management*

As discussed in section II-C, *PW-Wires* have the same area cost than baseline wires while they are twice slower. Fixing the number of *PW-Wires* is not a naive task. They will be used for sending prefeching-related messages (in particular, prefetch replies with data), whereas the remaining area will be consumed by *B-Wires* employed for sending ordinary messages and short (11-byte and 3-byte long) prefeching-related messages. The proportion between *PW-* and *B-Wires* has a direct impact in both the execution time and the power consumption of the interconnect. Preliminary simulations with different proportions of *PW-* and *B-Wires* showed that the best option is to replace half of the original wires by *PW-Wires*.

In this work, we use the same main parameters for the interconnect as in [22]. 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 are just 3-byte long. Replacements for lines in modified state are 75-byte long since they carry both address (8 bytes) and a cache line (64 bytes) beside control information (3 bytes). Finally, ordinary/prefetch reply messages are 67-byte long since they carry control information (3-bytes) and a cache line (64 bytes).

In order to match the metal area of the baseline configuration, in our heterogeneous interconnect design, each original 75-byte unidirectional link is designed to be made up of 296 *B-Wires* (37 bytes) and 304 *PW-Wires* (38 bytes). For a discussion regarding the implementation complexity of heterogeneous interconnects we refer the interested reader to [22].

IV. Experimental Results

This section shows the results that are obtained for our proposal under different scenarios and compare them against those achieved with the configuration that employs just *B-Wires*, which is taken as baseline.

A. *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* [26], that estimates both dynamic and leakage power and is based on RSIM [27]. In particular, *Sim-PowerCMP* employs as performance simulator a modified version of RSIM that models the architecture of the tiled CMP presented in section III. *Sim-PowerCMP* also implements already proposed and validated power models for both dynamic power and leakage power of each processing core, as well as the interconnection network.

Table II (top) shows the architecture configuration used across this paper. It describes a 16-core CMP built in

Table II
CONFIGURATION OF THE EVALUATED BASELINE CMP ARCHITECTURE AND APPLICATIONS.

CMP Configuration	
Process technology Tile area	65 nm 25 mm^2
Number of tiles	16
Cache line size	64 bytes
Core L1 I/D-Cache L2 Cache (per core)	4GHz, in-order 2-way model 32KB, 4-way 256KB, 4-way, 6+2 cycles
Memory access time	400 cycles
Network configuration Network bandwidth Link width Link length	2D mesh 75 GB/s 75 bytes (*8X-B-Wires*) 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 × 256, B=8
Ocean-cont	258 × 258 grid
Raytrace	car.env
Unstructured	mesh.2K, 5 timesteps
Water-nsq	512 molecules, 4 timesteps

65 nm technology. The tile area has been fixed to 25 mm^2, including a portion of the second-level cache [2]. With this configuration, links that interconnect routers configuring the 2D mesh topology measure around 5 mm. Table II (bottom) shows the applications used in our experiments. *Barnes-Hut*, *FFT*, *LU-cont*, *Ocean-cont*, *Raytrace* and *Water-nsq* are from the SPLASH-2 benchmark suite; Berkeley *EM3D* simulates the propagation of electromagnetic 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. All experimental results reported in this work are for the parallel phase of these applications.

B. Simulation results and analysis

In this section we analyze the impact of our proposal on both the execution time and on the power consumption for the inter-core links. All results have been normalized with respect to the baseline non-prefetching configuration where only *B-Wire*, unidirectional 75-byte wide links are considered (with the exception of Fig. 8 where results are normalized with respect to a 16-core CMP with the same prefetching technique).

Fig. 6 shows the normalized execution time with respect to that obtained for the baseline configuration for a 16-core CMP without prefetching. Barlines show the normalized execution time for prefetch-on-miss ($OnMiss$), tagged prefetch ($Tagged$), and stride-based prefetch ($Stride$) techniques (see Section II-B) applied to the L1D private caches. The stride-based prefetching is based on the implementation incorporated in the IBM Power 4 [28]. Each prefetcher contains three separate filter tables: positive unit stride, negative unit stride, and non-unit stride. Once a filter table entry detects a miss stream, the prefetcher allocates a stream table entry and initiates the prefetch of K consecutive cache blocks (for the $OnMiss$ and $Tagged$ prefetching schemes the value of K is set to 1). For comparison purposes, the normalized execution time obtained when only *B-Wires* are employed is also shown. On average, we obtain improvements in execution time of around 10% for all the prefetching techniques evaluated, which demonstrates the convenience of using hardware prefetching in future many-core CMPs. This improvement has high variability, ranging from almost negligible or even a slight degradation for Barnes, Raytrace, Unstructured and Water to improvements of 20-35% for em3d and Ocean-Cont.

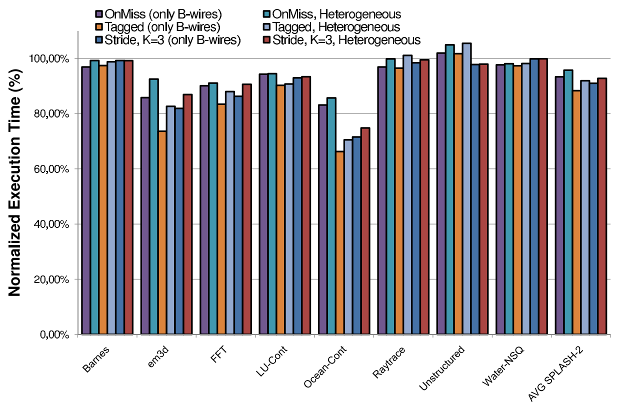

Figure 6. Normalized execution time for different prefetching schemes (with and without heterogeneous links) for a 16-core CMP.

This observed variability is due to the memory access patterns exhibited by the applications. Some applications, as FFT, LU-Cont, or Ocean-Cont, present regular memory access patterns that lead to high percentage of useful prefetches as we can see in Fig. 7 (top) where we present a classification of the prefetches. In this figure, prefetches are classified into: *useful* if the prefetched line is accessed before being replaced, *late* if other requests coalesce into the MSHR allocated for the prefetched line, *useless* if the prefetched line gets replaced before it is requested by the processor, *unnecesary* if the prefetch coalesces into a MSHR for an already-on-the-fly cache miss, and *invalidated* if the prefetched line gets invalidated before being requested by the processor. On the other hand, applications such as Barnes or Raytrace show a high percentage of late or useless prefetches that lead to negligible improvements in the execution time.

Going back to Fig. 6 again, when heterogeneous links are considered, an average slowdown of about 2% is observed with respect to the *B-Wire*-only configuration that also uses prefetching. In this case, similar degradations are obtain for all the applications. This degradation is explained by the additional delay of sending the prefetch replies through *PW-Wires*. Fig. 7 (bottom) shows that 5% of the previously useful prefetches are now classified as late prefetches, explaining the observed slowdown.

However, the benefits of using a heterogeneous interconnect in the context of hardware prefetching, as we

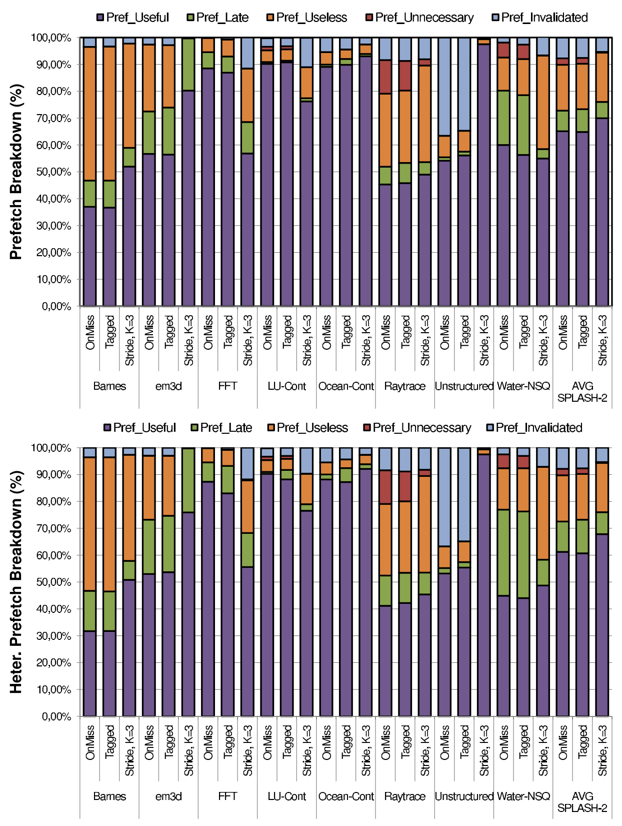

Figure 7. Classification of the different types of prefetches observed for the *B-Wire* only (top) and heterogeneous interconnect (bottom).

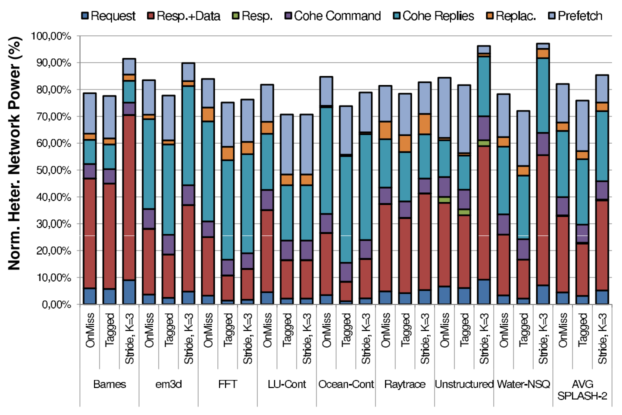

Figure 8. Normalized link power consumption for different prefetching schemes over heterogeneous links (baseline configuration: 16-core CMP with prefetching).

propose, can be noticed when considering the network power consumption. Fig. 8 plots the normalized link power consumption when the prefetch replies are sent through *PW-Wires*. The baseline configuration used for normalization is a 16-core CMP implementing the same prefetching technique but using only *B-Wire* links. Reductions of up to 30% are obtained (15-23% on average) and in this case the variability among applications is reduced. Better results are obtained, as expected, for the *OnMiss* and *Tagged* prefetching techniques due to the bigger emphasis on prefetching traffic that these techniques show (as seen in Fig. 5). This leads to more important reductions in the link power consumption when prefetched lines are sent through *PW-Wires*. These improvements translate into reductions in the normalized energy consumed for the full 16-core CMP of up to 10% for applications such as EM3D or Ocean-Cont, with average reductions of 4%.

V. Conclusions and Future Work

One way to tackle the problems due to wire delay is to use latency hiding techniques like hardware prefetching, which eliminates some cache misses and overlaps the latencies of others. Although, hardware prefetching can bring important improvements in terms of execution time, it significantly increases the number of messages in the on-chip network, therefore increasing its power consumption.

On the other hand, the use of heterogeneous on-chip interconnection networks has been previously demonstrated as an effective approach for alleviating the effects of wire delays and power consumption. An heterogeneous interconnection network is 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.

In this work we propose an energy-efficient prefetching proposal for tiled CMPs that consists in the use of a heterogeneous interconnect along with several hardware prefetching schemes. A heterogeneous interconnect comprised of only two different types of wires is proposed: low-power wires (*PW-Wires*) used for transmitting prefetched lines; and baseline wires (*B-Wires*) used for the transmission of the rest of messages. Again, we want to point out that this work is not aimed at proposing a particular prefetching scheme but at exploiting the non-critical nature of prefetching traffic by means of using a heterogeneous interconnect.

Results obtained through detailed simulations of a 16-core CMP show that the proposed on-chip message management mechanism can reduce the power consumed by the links of the interconnection network about 23% with a degradation in execution time of 2%. Finally, these reductions translate into overall CMP savings of up to 10% (4% on average) when the consumed energy is considered.

All 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 the on-chip interconnect, especially for next-generation dense CMP architectures.

As part of our future work, we plan to evaluate the interactions between the proposal presented in this work and *Reply Partitioning* [23] in order to determine if a positive interaction occurs when both techniques are used together. Moreover, it will be interesting to evaluate the impact of our proposal for the lookahead program counter (LA-PC) stride prefeching [9], which is is similar to the basic stride prefetching but triggered by LA-PC, which is several iterations ahead of the PC, improving timeliness.

We believe that the use of an appropriately adjusted LA-PC could reduce the small degradation in execution time experienced by our proposal.

Acknowledgment

This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2006-15516-C4-03", and also by the Fundación Séneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) under grant 05831/PI/07.

References

[1] S. Vangal, J. Howard, G. Ruhl, S. Dighe *et al.*, "An 80-tile 1.28tflops network-on-chip in 65nm cmos," Proc. of the Solid-State Circuits Conference, Feb. 2007, pp. 98–589.

[2] M. Zhang and K. Asanovic, "Victim replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors," Proc. of the 32nd Int'l Symp. on Computer Architecture, IEEE Press, Jun. 2005, pp. 336–345.

[3] R. Ho, K. Mai, and M. Horowitz, "The future of wires," Proceedings of the IEEE, vol. 89, Apr. 2001, pp. 490–504.

[4] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," Proc. of the 35th Int'l Symp. on Microarchitecture, IEEE Press, Nov. 2002, pp. 294–305.

[5] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," Proc. of the 6th Int'l Workshop on System Level Interconnect Prediction, ACM Press, Feb. 2004, pp. 7–13.

[6] R. Balasubramonian, N. Muralimanohar, K. Ramani, and V. Venkatachalapathy, "Microarchitectural wire management for performance and power in partitioned architectures," Proc. of the 11th Int'l Symp. on High-Performance Computer Architecture, IEEE Press, Feb. 2005, pp. 28–39.

[7] K. Banerjee and A. Mehrotra, "A power-optimal repeater insertion methodology for global interconnects in nanometer designs," IEEE Trans. on Electron Devices, vol. 49, Nov. 2002, pp. 2001–2007.

[8] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling," Proc. of the 32nd Int'l Symp. on Computer Architecture, Jun. 2005, IEEE Press, pp. 408–419.

[9] T.-F. Chen and J.-L. Baer, "Effective hardware-based data prefetching for high-performance processors," IEEE Trans. Comput., vol. 44, May 1995, pp. 609–623.

[10] A. Roth, A. Moshovos, and G. S. Sohi, "Dependence based prefetching for linked data structures," SIGPLAN Not., vol. 33, Nov. 1998, pp. 115–126.

[11] G. Hinton, D. Sager, M. Upton, D. Boggs *et al.*, "The microarchitecture of the pentium&reg; 4 processor," Intel Technology Journal, vol. 1, 2001.

[12] B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner, "Power5 system microarchitecture," IBM J. Res. Dev., vol. 49, Jul. 2005, pp. 505–521.

[13] W. Anacker and C. P. Wang, "Performance evaluation of computing systems with memory hierarchies," IEEE Trans. Comput., vol. 16, Dec. 1967, pp. 764–773.

[14] A. Smith, "Sequential program prefetching in memory hierarchies," Computer, vol. 11, Dec. 1978, pp. 7–21.

[15] A. J. Smith, "Cache memories," ACM Comput. Surv., vol. 14, Sep. 1982, pp. 473–530.

[16] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," SIGARCH Comput. Archit. News, vol. 18, Jun. 1990, pp. 364–373.

[17] F. Dahlgren, M. Dubois, and P. Stenström, "Sequential hardware prefetching in shared-memory multiprocessors," IEEE Trans. Parallel Distrib. Syst., vol. 6, Jul. 1995, pp. 733–746.

[18] A. Ki and A. E. Knowles, "Adaptive data prefetching using cache information," Proc. of the 11th Int'l Conf. on Supercomputing, ACM Press, Jul. 1997, pp. 204–212.

[19] V. Srinivasan, E. S. Davidson, and G. S. Tyson, "A prefetch taxonomy," IEEE Trans. Comput., vol. 53, pp. 126–140, 2004.

[20] B. M. Beckmann and D. A. Wood, "TLC: transmission line caches," Proc. of the 36th Int'l Symp. on Microarchitecture (MICRO 03), IEEE Press, Dec. 2003, pp. 43–54.

[21] N. Muralimanohar and R. Balasubramonian, "The effect of interconnect design on the performance of large L2 caches," Proc. of the 3rd IBM Watson Conf. on Interaction between Architecture, Circuits, and Compilers (P=ac2), Oct. 2006.

[22] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. Carter, "Interconnect-aware coherence protocols for chip multiprocessors," Proc. of the 33rd Int'l Symp. on Computer Architecture (ISCA 06), Jun. 2006, pp. 339–351.

[23] A. Flores, J. L. Aragón, and M. E. Acacio, "Efficient message management in tiled cmp architectures using a heterogeneous interconnection network," Proc. of the 14th Int'l Conf. on High Performance Computing, Dec. 2007, pp. 133–146.

[24] S. P. Vanderwiel and D. J. Lilja, "Data prefetch mechanisms," ACM Comput. Surv., vol. 32, Jun. 2000, pp. 174–199.

[25] L. Zhao, R. Iyer, S. Makineni, J. Moses *et al.*, "Performance, area and bandwidth implications on large-scale cmp cache design," Proc. of the 1st Workshop on Chip Multiprocessor Memory Systems and Interconnects, IEEE Press, Feb. 2007.

[26] 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, Sept. 2008, pp. 341 - 364.

[27] C. J. Hughes, V. S. Pai, P. Ranganathan, and S. V. Adve, "RSIM: simulating shared-memory multiprocessors with ILP processors," IEEE Computer, vol. 35, Feb. 2002, pp. 40–49.

[28] J. M. Tendler, J. S. Dodson, J. S. F. Jr., H. Le, and B. Sinharoy, "Power4 system microarchitecture," IBM Journal of Research and Development, vol. 46, Jul. 2002, pp. 5–26.