# Wire Management for Coherence Traffic in Chip Multiprocessors

Liqun Cheng, Naveen Muralimanohar, Karthik Ramani, Rajeev Balasubramonian, John Carter
School of Computing, University of Utah *

## Abstract

*Improvements in semiconductor technology have made it possible to include multiple processor cores on a single die. Chip Multi-Processors (CMP) are an attractive choice for future billion transistor architectures due to their low design complexity, high clock frequency, and high throughput. In a typical CMP architecture, the L2 cache is shared by multiple cores and data coherence is maintained among private L1s. Coherence operations entail frequent communication over global on-chip wires. In future technologies, communication between different L1s will have a significant impact on overall processor performance and power consumption.*

*On-chip wires can be designed to have different latency, bandwidth, and energy properties. Likewise, coherence protocol messages have different latency and bandwidth needs. We propose an interconnect comprised of wires with varying latency, bandwidth, and energy characteristics, and advocate intelligently mapping coherence operations to the appropriate wires. In this paper, we present a comprehensive list of techniques that allow coherence protocols to exploit a heterogeneous interconnect and present preliminary data that indicates the potential of these techniques to significantly improve performance and reduce power consumption. We further demonstrate that most of these techniques can be implemented at a minimum complexity overhead.*

## 1. Introduction

Advances in process technology have led to the emergence of new bottlenecks in future microprocessors. One of the chief bottlenecks to performance is the high cost of on-chip communication through global wires [19]. Power consumption has also emerged as a first order design metric and wires contribute up to 50% of total chip power in some processors [28]. Future microprocessors are likely to exploit huge transistor budgets by employing a chip multi-processor (CMP) architecture [30, 32]. Multi-threaded workloads that execute on such processors will experience high on-chip communication latencies and will dissipate significant power in interconnects. In the past, the design of interconnects was primarily left up to VLSI and circuit designers. However, with *communication* emerging as a larger power and performance constraint than *computation*, architects may wish to consider different wire implementations and identify creative ways to exploit them [6]. This paper presents a number of creative ways in which coherence communication in a CMP can be mapped to different wire implementations with minor increases in complexity. We present preliminary results that demonstrate that such an approach can both improve performance and reduce power dissipation.

In a typical CMP, the L2 cache and lower levels of the memory hierarchy are shared by multiple cores [22, 32]. Sharing the L2 cache allows high cache utilization and avoids duplicating cache hardware resources. L1 caches are typically not shared as such an organization entails high communication latencies for every load and store. Maintaining coherence between the individual L1s is a challenge in CMP systems. There are two major mechanisms used to ensure coherence among L1s in a chip multiprocessor. The first option employs a bus connecting all of the L1s and a snoopy bus-based coherence protocol. In this design, every L1 cache miss results in a coherence message being broadcast on the global coherence bus. Individual L1 caches perform coherence operations on their local data in accordance with these coherence

*This work was supported in part by NSF grant CCF-0430063 and by Silicon Graphics Inc.

messages. The second approach employs a centralized directory in the L2 cache that tracks sharing information for all cache lines in the L2 and implements a directory-based coherence protocol. In this design, every L1 cache miss is sent to the L2 cache, where further actions are taken based on directory state. Numerous studies [1, 10, 20, 23, 27] have characterized the high frequency of cache misses in parallel workloads and the high impact these misses have on total execution time. On a cache miss, a variety of protocol actions are initiated, such as request messages, invalidation messages, intervention messages, data block writebacks, data block transfers, etc. Each of these messages involves on-chip communication with latencies that are projected to grow to tens of cycles in future billion transistor architectures [2].

VLSI techniques enable a variety of different wire implementations that are typically not exploited at the microarchitecture level. For example, by tuning wire width and spacing, we can design wires with varying latency and bandwidth properties. Similarly, by tuning repeater size and spacing, we can design wires with varying latency and energy properties. To take advantage of VLSI techniques and better match the interconnect design to communication requirements, heterogeneous interconnects have been proposed [6], where every link consists of wires that are optimized for either latency, energy, or bandwidth. In this study, we explore optimizations that are enabled when such a heterogeneous interconnect is employed for coherence traffic. For example, on a cache write miss, the requesting processor may have to wait for data from the home node (a two hop transaction) and for acknowledgments from other sharers of the block (a three hop transaction). Since the acknowledgments are on the critical path and have low bandwidth needs, they can be mapped to wires optimized for delay, while the data block transfer is not on the critical path and can be mapped to wires that are optimized for low power.

The paper is organized as follows. Section 2 reviews techniques that enable different wire implementations and the design of a heterogeneous interconnect. Section 3 describes the proposed innovations that map coherence messages to different on-chip wires. Section 4 presents preliminary results that indicate the potential of our proposed techniques. Section 5 discusses related work and we conclude in Section 6.

## 2. Wire Implementations

We begin with a quick review of factors that influence wire properties. It is well-known that the delay of a wire is a function of its RC time constant (R is resistance and C is capacitance). Resistance per unit length is (approximately) inversely proportional to the width of the wire [19]. Likewise, a fraction of the capacitance per unit length is inversely proportional to the spacing between wires, and a fraction is directly proportional to wire width. These wire properties provide an opportunity to design wires that trade off bandwidth and latency. By allocating more metal area per wire and increasing wire width and spacing, the net effect is a reduction in the RC time constant. This leads to a wire design that has favorable latency properties, but poor bandwidth properties (as fewer wires can be accommodated in a fixed metal area). Our analysis [6] shows that in certain cases, nearly a three-fold reduction in wire latency can be achieved, at the expense of a four-fold reduction in bandwidth. Further, researchers are actively pursuing transmission line implementations that enable extremely low communication latencies [12, 16]. However, transmission lines also entail significant metal area overheads in addition to logic overheads for sending and receiving [8, 12]. If transmission line implementations become cost-effective at future technologies, they represent another attractive wire design point that can trade off bandwidth for low latency.

Similar trade-offs can be made between latency and power consumed by wires. Global wires are usually composed of multiple smaller segments that are connected with repeaters [5]. The size and spacing of repeaters influences wire delay and power consumed by the wire. When smaller and fewer repeaters are employed, wire delay increases, but power consumption is reduced. The repeater configuration that minimizes delay is typically very different from the repeater configuration that minimizes power consumption. Banerjee *et al.* [7] show that at 50nm technology, a five-fold reduction in power can be achieved at the expense of a two-fold increase in latency.

Thus, by varying properties such as wire width/spacing and repeater size/spacing, we can implement wires with different latency, bandwidth, and power properties. If a data packet has 64 bits, global interconnects are typically designed to min-

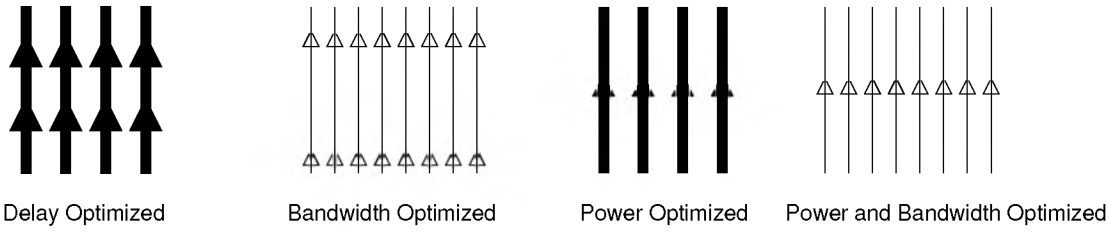


**Figure 1. Examples of different wire implementations. Power optimized wires have fewer and smaller repeaters, while bandwidth optimized wires have narrow widths and spacing.**

imize delay for the transfer of 64-bit data, while not exceeding the allocated metal area. We refer to these wires as *B-Wires*. In addition to this base 64-bit interconnect, there are at least three other wire implementations that are potentially beneficial:

- *P-Wires*: Wires that are power-optimal. The wires have longer delays as they employ small repeater size and wide repeater spacing.
- *W-Wires*: Wires that are bandwidth-optimal. The wires have minimum width and spacing and have longer delays.
- *L-Wires*: Wires that are latency-optimal. These wires employ very wide wires and have low bandwidth.

To limit the range of possibilities, *P-Wires* and *W-Wires* can be combined to form a single wire implementation *PW-Wires*, that have poor delay characteristics, but allow low power and high bandwidth. While a traditional architecture would employ the entire available metal area for B-Wires, we propose the design of a heterogeneous interconnect, where part of the available metal area is employed for B-Wires, part for L-Wires, and part for PW-Wires. Thus, any data transfer has the option of using one of three sets of wires to effect the communication. Figure 1 demonstrates the differences between the wire implementations. In the next section, we will demonstrate how these options can be exploited to improve performance and reduce power consumption. If the mapping of data to a set of wires is straightforward, the logic overhead for the decision process is likely to be minimal. This issue will be treated in more detail in subsequent sections.

## 3. Optimizing Coherence Traffic

The previous section outlines wire implementation options available to an architect. For each cache coherence protocol, there exist myriad coherence operations with varying bandwidth and latency needs. Because of this diversity, there are numerous opportunities to improve performance and power characteristics by employing a heterogeneous interconnect. The goal of this section is to present a comprehensive listing of such opportunities. In Section 3.1 we focus on protocol-specific optimizations. We then discuss a variety of protocol-independent techniques in Section 3.2. Finally, we discuss the implementation complexity of the various techniques in Section 3.3.

### 3.1. Protocol-dependent Techniques

We begin by examining the characteristics of coherence operations in both directory-based and snooping bus-based coherence protocols. We then describe how these coherence operations can be mapped to the appropriate set of wires. In a bus-based design, the ability of a cache to directly respond to another cache's request leads to low L1 cache-to-cache miss latencies. L2 cache latencies are relatively higher as a processor core has to acquire the bus before sending the request to L2. It is difficult to support a large number of processor cores with a single bus due to the bandwidth and electrical limits of a centralized bus [11]. In a directory-based design [14, 25], each L1 connects to the L2 cache through a point-to-point link. This design has low L2 hit latency and scales better. However, each L1 cache-to-cache miss must be forwarded by the L2 cache, which implies high L1 cache-to-cache latencies. The performance comparison between these two design choices depends on the cache size, miss rate,

number of outstanding memory requests, working-set size, sharing behavior of the targeted benchmarks, etc. Since either option may be attractive to chip manufacturers, we will consider both forms of coherence protocols in our study.

**Write-Invalidate Directory-based Protocol**

Write-invalidate directory-based protocols have been implemented in existing dual-core CMPs [32] and will likely be used in larger scale CMPs as well. In a directory-based protocol, every cache line has a directory where the states of the block in all L1s are stored. Whenever a request misses in an L1 cache, a coherence message is sent to the directory at the L2 to check the cache line's global state. If there is a clean copy in the L2 and the request is a READ, it is served by the L2 cache. Otherwise, another L1 must hold an exclusive copy and the READ request is forwarded to the exclusive owner, which supplies the data. For a WRITE request, if any other L1 caches hold a copy of the cache line, coherence messages are sent to each of them requesting that they invalidate their copies. When each of these invalidation requests is acknowledged, the L2 cache can supply an exclusive copy of the cache line to the requesting L1 cache.

Hop imbalance is quite common in a directory-based protocol. To exploit this imbalance, we can send critical messages on fast wires to increase performance and send non-critical messages on slow wires to save power. For the sake of this discussion, we assume that the hop latencies of different wires are in the following ratio: L-wire : B-wire : PW-wire :: 1 : 2 : 3

**Proposal I:** *Read exclusive request for block in shared state*

In this case, the L2 cache's copy is clean, so it provides the data to the requesting L1 and invalidates all shared copies. When the requesting L1 receives the reply message from the L2, it collects invalidation acknowledgment messages from the other L1s before returning the data to the processor core[1]. Figure 2 depicts all generated messages.

The reply message from the L2 takes only one hop, while the invalidation acknowledgment messages take two hops – an example of hop imbalance. Since there is no benefit to receiving the cache line early, latencies for each hop can be chosen that equalize communication latency for the cache line and the acknowledgment messages. Acknowledgment messages include identifiers so they can be matched against the outstanding request in the L1's MSHR. Since there are only a few outstanding requests in the system, the identifier requires few bits, allowing the acknowledgment to be transferred on low-bandwidth low-latency L-Wires. Simultaneously, the data block transmission from the L2 can happen on low-power PW-Wires and still finish before the arrival of the acknowledgments. This strategy improves performance (because acknowledgments are often on the critical path) and reduces power consumption (because the data block is now transferred on power-efficient wires). While circuit designers have frequently employed different types of wires within a circuit to reduce power dissipation without extending the critical path, the proposals in this paper represent some of the first attempts to exploit wire properties at the architectural level.

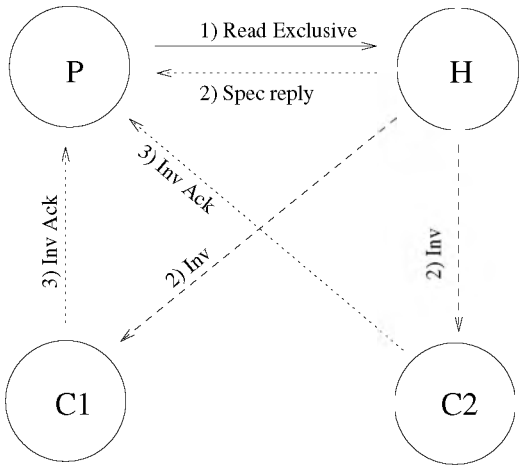


**Figure 2. Read exclusive request for a block in shared state**

**Proposal II:** *Read request for block in exclusive state*

In this case, the value in the L2 is likely to be stale and the following protocol actions are taken. The L2 cache sends a speculative data reply to the requesting L1 and forwards the read request as an intervention message to the exclusive owner. If the cache copy in the exclusive owner is clean, an acknowledgment message is sent to the requesting L1, indicating that the speculative data reply from the L2 is valid. If the cache copy is dirty, a response message with the latest data is sent to the

[1] Some coherence protocols may not impose all of these constraints, thereby deviating from a sequentially consistent memory model.

requesting L1 and a write-back message is sent to the L2. Since the requesting L1 cannot proceed until it receives a message from the exclusive owner, the speculative data reply from the L2 (a single hop transfer) can be sent on slower PW-Wires. The forwarded request to the exclusive owner is on the critical path, but includes the block address. It is therefore not eligible for transfer on L-Wires. If the owner's copy is in the exclusive clean state, a low-bandwidth acknowledgment to the requestor can be sent on L-Wires. If the owner's copy is dirty, the cache block can be sent over B-Wires, while the low priority writeback to the L2 can happen on PW-Wires. With the above mapping, we accelerate the critical path by using faster L-Wires, while also lowering power consumption by sending non-critical data on PW-Wires. The above protocol actions apply even in the case when a read-exclusive request is made for a block in the exclusive state.

**Proposal III:** *NACK messages*
When the directory state is busy, incoming requests are often NACKed by the home directory, i.e., a negative acknowledgment is sent to the requester rather than buffering the request. Typically the requesting cache controller reissues the request and the request is serialized in the order in which it is actually accepted by the directory. A NACK message can be matched by comparing the request id (MSHR index) rather than the full address, so a NACK is eligible for transfer on low-bandwidth L-Wires. When network contention is low, the home node should be able to serve the request when it arrives again, in which case sending the NACK on fast L-Wires can improve performance. In contrast, when network contention is high, frequent backoff-and-retry cycles are experienced. In this case, fast NACKs only increase traffic levels without providing any performance benefit. In order to save power, NACKs can be sent on PW-Wires.

**Write-Invalidate Bus-Based Protocol**
We next examine techniques that apply to bus-based snooping protocols. The role of the L1s and the L2 in a bus-based CMP system are very similar to that of the L2s and memory in a bus-based SMP (symmetric multiprocessor) system.

**Proposal IV:** *Signal wires*
Three wired-OR signals are typically used to avoid involving the lower/slower memory hierarchy [15]. Two of these signals are responsible for reporting the state of snoop results and the third indicates that the snoop result is valid. The first signal is asserted when any L1 cache, besides the requester, has a copy of the block. The second signal is asserted if any cache has the block in the exclusive state. The third signal is an inhibit signal, asserted until all caches have completed their snoop operations. When the third signal is asserted, the requesting L1 and the L2 can safely examine the other two signals. Since all of these signals are on the critical path, implementing them using low-latency L-Wires can improve performance.

**Proposal V:** *Voting wires*
Another design choice is whether to use cache-to-cache transfers if the data is in the shared state in a cache. The Silicon Graphics Challenge [17] and the Sun Enterprise use cache-to-cache transfers only for data in the modified state, in which case there is a single supplier. On the other hand, in the full Illinois MESI protocol, a block can be preferentially retrieved from another cache rather than from memory. However, when multiple caches share a copy, a "voting" mechanism is required to decide which cache will supply the data, and this voting mechanism can benefit from the use of low latency wires.

### 3.2. Protocol-independent Techniques

**Proposal VI:** *Narrow Bit-Width Operands for Synchronization Variables*
Synchronization is one of the most important factors in the performance of a parallel application. Synchronization is not only often on the critical path, but it also contributes a large percentage (up to 40%) of coherence misses [27]. Locks and barriers are the two most widely used synchronization constructs. Both of them use small integers to implement mutual exclusion. Locks often toggle the synchronization variable between zero and one, while barriers often linearly increase a barrier variable from zero to the number of processors taking part in the barrier operation. Such data transfers have limited bandwidth needs and can benefit from using L-Wires.

This optimization can be further extended by examining the general problem of cache line compaction. For example, if a cache line is comprised mostly of 0 bits, trivial data compaction algorithms may reduce the bandwidth needs of the cache line, allowing it to be transferred on L-Wires instead of B-Wires. If the wire latency difference between the two wire implementations is greater than the delay of the compaction/de-

compaction algorithm, performance improvements are possible.

**Proposal VII:** *Assigning Writeback Data to PW-Wires*
Writeback data transfers result from cache replacements or external request/intervention messages. Since writeback messages are rarely on the critical path, assigning them to PW-Wires can save power without incurring significant performance penalties.

**Proposal VIII:** *Assigning Narrow Messages to L-Wires*
Coherence messages that include the data block address or the data block itself are many bytes wide. However, many other messages, such as acknowledgments and NACKs, do not include the address or data block and only contain control information (source/destination, message type, MSHR id, etc.). Such narrow messages can be assigned to low latency L-Wires.

### 3.3. Implementation Complexity

In a conventional multiprocessor interconnect, a subset of wires are employed for addresses, a subset for data, and a subset for control signals. Every bit of communication is mapped to a unique wire. When employing a heterogeneous interconnect, a communication bit can map to multiple wires. For example, data returned by the L2 in respose to a read-exclusive request may map to B-Wires or PW-Wires depending on whether there are other sharers for that block (**Proposal I**). Thus, every wire must be associated with a multiplexor and de-multiplexor.

The decision process in selecting the right set of wires is minimal. For example, in **Proposal I**, an OR function on the directory state for that block is enough to select either B- or PW-Wires. In **Proposal II**, the decision process involves a check to determine if the block is in the exclusive state. To support **Proposal III**, we need a mechanism that tracks the level of congestion in the network (for example, the number of buffered outstanding messages). There is no decision process involved for **Proposals IV**, **V**, and **VII**. **Proposals VI** and **VIII** require logic to compute the width of an operand, similar to logic used in the PowerPC 603 [18] to determine the latency of integer multiply.

Cache coherence protocols are already designed to be robust in the face of variable delays for different messages. In all proposed innovations, a data packet is not distributed across different sets of wires. Therefore, different components of an entity do not arrive at different periods of time, thereby eliminating any timing problems. It may be worth considering sending the critical word of a cache line on L-Wires and the rest of the cache line on PW-Wires. Such a proposal may entail non-trivial complexity to handle corner cases and is not discussed further in this paper.

In a snooping bus-based coherence protocol, transactions are serialized by the order in which addresses appear on the bus. None of our proposed innovations for snooping protocols affect the transmission of address bits (address bits are always transmitted on B-Wires), so the transaction serialization model is preserved.

The use of heterogeneous interconnects does not imply an increase in metal area. Rather, we advocate that the available metal area be partitioned among different wire implementations.

## 4. Results

### 4.1. Methodology

The evaluation is performed with a detailed CMP architecture simulator based on UVSIM [34], a cycle-accurate execution-driven simulator. The CMP cores are out-of-order superscalar processors with private L1 caches, shared L2 cache and all lower level memory hierarchy components. Contention for memory hierarchy resources (ports, banks, buffers, etc.) are modeled in detail. In order to model an aggressive future generation CMP, we assume 16 processor cores, connected by a two-level tree interconnect. The simulated on-chip interconnect is based on SGI's NUMALink-4. A set of four processor cores is connected through a crossbar router, allowing low-latency communication to neighboring cores. As shown in Figure 3, the crossbar routers are connected to the root router, where the centralized L2 lies. We do not model contention within the routers, but do model port contention on the network interfaces. Each cache line in the L2 cache has a directory which saves sharing information for the processor cores. Every L1 cache miss is sent to the L2 cache, where further actions are taken based on the directory state. We model the directory-based cache coherence protocol that is employed in the SGI Origin 3000 [31]. It is an MESI write-invalidate protocol with

| Parameter | Value |
|---|---|
| Processor | 4-issue, 48-entry active list, 2GHz |
| L1 I-cache | 2-way, 64KB, 64B lines, 1-cycle lat. |
| L1 D-cache | 2-way, 64KB, 64B lines, 2-cycle lat. |
| On-chip Network | 10 processor cycles per hop |
| L2 cache | 4-way, 8MB, 64B lines, 4-cycle bank-lat. |
| MSHR per CPU | 16 |
| DRAM | 16 16-bit-data DDR channels |

**Table 1. System configuration.**

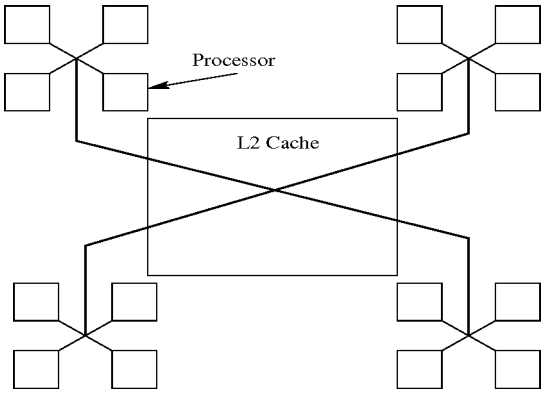


**Figure 3. Interconnect Topology**

migratory optimization. The migratory optimization causes a read (shared) request to return exclusive ownership if the requested cache line is in the UNOWN state. Important simulation parameters are listed in Table 1.

To test our ideas, we employ a workload consisting of all programs from the SPLASH-2 [33] and NAS parallel benchmark [4] suites that were compatible with our simulator. The programs were run to completion, but all experimental results reported in this paper are for the parallel phase of these applications and employ the default input sets for SPLASH-2 and the S-class for NAS.

### 4.2. Preliminary Results

While Section 3 provides a comprehensive list of potential optimizations, our preliminary study only examines the effect of one class of optimizations on a directory-based cache coherence protocol. Our base model assumes a fat-tree structure with four children on each non-leaf node. Processor cores are on the leaf node, and four neighboring cores are grouped into a sub-tree. We define the hop number as the number of routers used to transfer a network message between two processor cores. As illustrated in Figure 3, it takes one network hop to transfer a message between two processors in the same domain (sub-tree), and three network hops to transfer a message between any two processors which belong to different domains (sub-trees). The latencies on the interconnects would depend greatly on the technology, processor layout, and available metal area. The estimation of some of these parameters is beyond the scope of this study. For the base case, we assume the metal layer is comprised entirely of B-Wires, and one hop latency is 10 processor cycles. This assumption is based on projections [2, 3] that claim on-chip wire delays of the order of tens of cycles.

In the base case, each processor can issue a single 64-bit packet every cycle that is transmitted on B-Wires. In our proposed heterogeneous interconnect, we again assume that each processor can issue a single packet every cycle, but that packet can be transmitted on one of three possible sets of wires. The transmission can either happen on a set of 64 B-Wires, a set of 64 PW-Wires, or a set of 16 L-Wires. While we have kept the packet throughput per processor constant in the base and proposed cases, the metal area cost of the heterogeneous interconnect is higher than that of the base case. The comparison of designs that consume equal metal area is part of future work. The experiments in this paper are intended to provide preliminary best-case estimates of the potential of a heterogeneous interconnect. Relative energy and delay estimates of wires have been derived in [6]. L-Wires (latency-optimal) have a latency of five cycles for each network hop and consume about 45% more dynamic energy than B-Wires. PW-Wires (low power and high bandwidth) have a latency of 15 cycles per hop and consume about half the dynamic energy of B-Wires. We assume that every interconnect is perfectly pipelined. It must be noted that L-Wires can yield lower latency than B-Wires even when sending messages larger than 16 bits (and lower than 112 bits). However, sending large messages through L-Wires will increase wait time for other messages, resulting in overall lower performance.

We will consider only the simplest subset of techniques proposed in Section 3, that entail the least complexity in mapping critical data transfers to L-Wires and non-critical data transfers to PW-Wires. Firstly, not all critical messages are eligible for transfer on L-

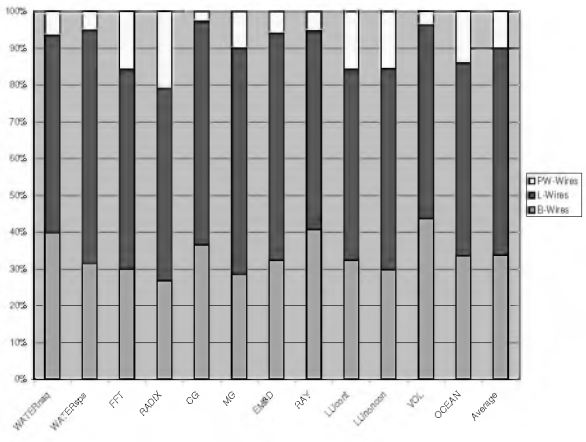


**Figure 4. Percentage of critical and non-critical messages**

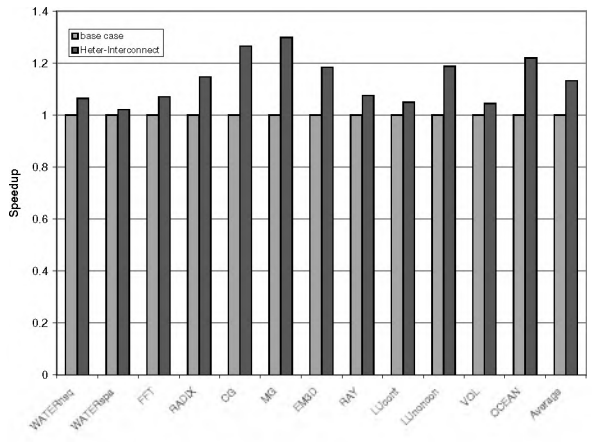


**Figure 5. Performance improvements**

Wires. Since we assume 16 L-Wires and 64-bit addresses, a message that includes the full address is always sent on B-Wires or PW-Wires. To identify what messages can be sent through B-Wires and PW-Wires, we classify all coherence messages into six categories: REQUEST, WRITE, PROBE, REPLY2MD, RESPONSE, and REPLY2PI.

Every memory transaction that misses in the local L1 cache will send out a REQUEST message to the L2 cache. A REQUEST message includes request type (READ, RDEXC, UPGRADE, etc.), source id, MSHR id, and data address. Although REQUEST messages are mostly on the critical path, they are too wide to benefit from L-Wires. Therefore, REQUEST messages are always transmitted on B-Wires.

WRITE messages result from L1 cache replacement. WRITE messages are often not on the critical path and they can be always sent on PW-Wires without degrading performance.

PROBE messages happen in cache-to-cache misses and can be further classified as INTERVENTION messages and INVALIDATE messages. An INTERVENTION request retrieves the most recent data for a line from an exclusive copy that may exist within a remote cache. An INVALIDATE request removes copies of a cache line that may exist in remote caches. Both INTERVENTION and INVALIDATE messages include request type, source id, address, and the MSHR id of the REQUEST message that generated the PROBE message. PROBE messages are usually critical, but can only be sent through B-Wires due to the bandwidth limitation of L-Wires.

When a processor receives a PROBE message, it sends a REPLY2MD message to the home directory (if it has a dirty copy of the cache line) and a RESPONSE message to the processor that generated the REQUEST message. Response messages can have data, in which case, they must be sent on B-Wires. In most cases, the message is simply an acknowledgment that only needs reply type (4 bits), source id (4 bits in 16-core CMP), and MSHR id (4 bits), and can be sent on L-Wires. REPLY2MD is often off the critical path and can be sent on PW-Wires.

REPLY2PI messages are the messages sent from the home directory to the requester. Some REPLY2PI messages include data, while others only include control bits (such as NACK or the reply for an UPGRADE request). REPLY2PI messages not only have variable bandwidth needs, but also have variable criticality, as discussed in Section 3. For example, in cache-to-cache misses, REPLY2PI messages tend to arrive at the requester faster than the PROBE/RESPONSE messages and are therefore off the critical path.

In order to simplify the decision process in mapping data to wires, we adopt the following policy: (i) WRITE and REPLY2MD messages are always sent on PW-Wires, (ii) all other messages that are narrower than 16 bits are sent on L-Wires, and (iii) all other messages that are wider than 16 bits are sent on B-Wires. Thus, we are only incorporating **Proposal VII**, **Proposal VIII**, and parts of **Proposal III** in our simulation model. Detailed evaluations of other proposals are left for future work.

Figure 4 shows the percentage of messages sent

through different sets of wires, while assuming the allocation policy described above. It must be noted that a significant fraction of all messages are narrow enough that they can be sent on L-Wires. Messages sent on B- and PW-Wires are wider than messages sent on L-Wires. As a result, the fraction of bits transmitted on L-Wires is lower than that indicated in Figure 4. If we assume that dynamic energy consumed by transmissions on B, L, and PW-Wires are in the ratio 1: 1.45: 0.52 (as estimated in a prior study [6]), interconnect dynamic energy in the proposed design is reduced by 40%. Further, this reduction in interconnect energy is accompanied by improvements in performance. Figure 5 shows the performance speedup achieved by the transmission of some signals on low-latency L-Wires. The overall average improvement across the benchmark set is 13.3%.

## 5. Related Work

Beckmann *et al.* [9] address the problem of long L2 cache access times in a chip multiprocessor by employing low latency, low bandwidth transmission lines. They utilize transmission lines to send data from the center of the L2 cache to different banks. Kim *et al.* [21] proposed a dynamic non-uniform cache access (DNUCA) mechanism to accelerate cache access. Our proposal is orthogonal to the above technique and can be combined with non-uniform cache access mechanisms to improve performance.

A recent study by Citron *et al.* [13] examines entropy within data being transmitted on wires and identifies opportunities for compression. Unlike the proposed technique, they employ a single interconnect to transfer all data. Balasubramonian *et al.* [6] utilize heterogeneous interconnects for the transmission of register and load/store values to improve energy-delay characteristics in a partitioned microarchitecture.

Recent studies [20, 24, 26, 29] have proposed several protocol optimizations that can benefit from heterogeneous interconnects. For example, in the *Dynamic Self Invalidation* scheme proposed by Lebeck *et al.* [26], the self-invalidate [24, 26] messages can be effected through power-efficient PW-Wires. In a processor model implementing token coherence, the low-bandwidth token messages [29] are often on the critical path and thus, can be effected on L-Wires. A recent study by Huh *et al.* [20] reduces the frequency of false sharing by employing incoherent data. For cache lines suffering from false sharing, only the sharing states need to be propagated and such messages are a good match for low-bandwidth L-Wires.

## 6. Conclusions and Future Work

Coherence traffic in a chip multiprocessor has diverse needs. Some messages can tolerate long latencies, while others are on the program critical path. Further, messages have varied bandwidth demands. On-chip global wires can be designed to optimize latency, bandwidth, or power. We advocate partitioning available metal area across different wire implementations and intelligently mapping data to the set of wires best suited for its communication. This paper presents numerous novel techniques that can exploit a heterogeneous interconnect to simultaneously improve performance and reduce power consumption.

Our preliminary evaluation of a subset of the proposed techniques shows that a large fraction of messages have low bandwidth needs and can be transmitted on low latency wires, thereby yielding a performance improvement of 13%. At the same time, a 40% reduction in interconnect dynamic energy is observed by transmitting non-critical data on power-efficient wires. These improvements are achieved at a marginal complexity cost as the mapping of messages to wires is extremely straightforward.

For future work, we plan to strengthen our evaluations by comparing processor models with equal metal area. We will carry out a sensitivity analysis with respect to important processor parameters such as latencies, interconnect topologies, etc. We will also evaluate the potential of other techniques listed in this paper.

## References


[1] M. E. Acacio, J. Gonzalez, J. M. Garcia, and J. Duato. The Use of Prediction for Accelerating Upgrade Misses in CC-NUMA Multiprocessors. In *Proceedings of PACT-11*, 2002.

[2] V. Agarwal, M. Hrishikesh, S. Keckler, and D. Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. In *Proceedings of ISCA-27*, pages 248–259, June 2000.

[3] S. I. Association. International Technology Roadmap for Semiconductors 2003. http://public.itrs.net/Files/2003ITRS/Home2003.htm.

[4] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, Fall 1994.

[5] H. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.

[6] R. Balasubramonian, N. Muralimanohar, K. Ramani, and V. Venkatachalapathy. Microarchitectural Wire Management for Performance and Power in Partitioned Architectures. In *Proceedings of HPCA-11*, February 2005.

[7] K. Banerjee and A. Mehrotra. A Power-optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs. *IEEE Transactions on Electron Devices*, 49(11):2001–2007, November 2002.

[8] B. Beckmann and D. Wood. TLC: Transmission Line Caches. In *Proceedings of MICRO-36*, December 2003.

[9] B. Beckmann and D. Wood. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proceedings of MICRO-37*, December 2004.

[10] E. E. Bilir, R. M. Dickson, Y. Hu, M. Plakal, D. J. Sorin, M. D. Hill, and D. A. Wood. Multicast Snooping: A New Coherence Method using a Multicast Address Network. *SIGARCH Comput. Archit. News*, pages 294–304, 1999.

[11] F. A. Briggs, M. Cekleov, K. Creta, M. Khare, S. Kulick, A. Kumar, L. P. Looi, C. Natarajan, S. Radhakrishnan, and L. Rankin. Intel 870: A building block for cost-effective, scalable servers. *IEEE Micro*, 22(2):36–47, 2002.

[12] R. Chang, N. Talwalkar, C. Yue, and S. Wong. Near Speed-of-Light Signaling Over On-Chip Electrical Interconnects. *IEEE Journal of Solid-State Circuits*, 38(5):834–838, May 2003.

[13] D. Citron. Exploiting Low Entropy to Reduce Wire Delay. *IEEE Computer Architecture Letters*, vol.2, January 2004.

[14] Corporate Institute of Electrical and Electronics Engineers, Inc. Staff. *IEEE Standard for Scalable Coherent Interface, Science: IEEE Std. 1596-1992*. 1993.

[15] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: a Hardware/software Approach*. Morgan Kaufmann Publishers, Inc, 1999.

[16] W. Dally and J. Poulton. *Digital System Engineering*. Cambridge University Press, Cambridge, UK, 1998.

[17] M. Galles and E. Williams. Performance Optimizations, Implementation, and Verifi cation of the SGI Challenge Multiprocessor. In *HICSS (1)*, pages 134–143, 1994.

[18] G. Gerosa and et al. A 2.2 W, 80 MHz Superscalar RISC Microprocessor. *IEEE Journal of Solid-State Circuits*, 29(12):1440–1454, December 1994.

[19] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. *Proceedings of the IEEE*, Vol.89, No.4, April 2001.

[20] J. Huh, J. Chang, D. Burger, and G. S. Sohi. Coherence Decoupling: Making Use of Incoherence. In *Proceedings of ASPLOS-XI*, pages 97–106, 2004.

[21] J. Kim, M. Taylor, J. Miller, and D. Wentzlaff. Energy Characterization of a Tiled Architecture Processor with On-Chip Networks. In *Proceedings of ISLPED*, pages 424–427, 2003.

[22] K. Krewell. UltraSPARC IV Mirrors Predecessor: Sun Builds Dualcore Chip in 130nm. *Microprocessor Report*, pages 1,5–6, Nov. 2003.

[23] A.-C. Lai and B. Falsafi . Memory Sharing Predictor: The Key to a Speculative Coherent DSM. In *Proceedings of ISCA-26*, 1999.

[24] A.-C. Lai and B. Falsafi . Selective, Accurate, and Timely Self-Invalidation Using Last-Touch Prediction. In *Proceedings of ISCA-27*, pages 139–148, 2000.

[25] J. Laudon and D. Lenoski. The SGI Origin: A cc-NUMA Highly Scalable Server. In *Proceedings of ISCA-24*, pages 241–251, June 1997.

[26] A. R. Lebeck and D. A. Wood. Dynamic Self-Invalidation: Reducing Coherence Overhead in Shared-Memory Multiprocessors. In *Proceedings of ISCA-22*, pages 48–59, 1995.

[27] K. M. Lepak and M. H. Lipasti. Temporally Silent Stores. In *Proceedings of ASPLOS-X*, pages 30–41, 2002.

[28] N. Magen, A. Kolodny, U. Weiser, and N. Shamir. Interconnect Power Dissipation in a Microprocessor. In *Proceedings of System Level Interconnect Prediction*, February 2004.

[29] M. M. K. Martin, M. D. Hill, and D. A. Wood. Token Coherence: Decoupling Performance and Correctness. In *Proceedings of ISCA-30*, 2003.

[30] K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K.-Y. Chang. The Case for a Single-Chip Multiprocessor. In *Proceedings of ASPLOS-VII*, October 1996.

[31] Silicon Graphics, Inc. *SGI™Origin™3000 Series Technical Report*, Jan 2001.

[32] J. Tendler, S. Dodson, S. Fields, H. Le, and B. Sinharoy. POWER4 System Microarchitecture. Technical report, IBM Server Group Whitepaper, October 2001.

[33] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of ISCA-22*, pages 24–36, June 1995.

[34] L. Zhang. UVSIM Reference Manual. Technical Report UUCS-03-011, University of Utah, May 2003.