Leveraging Wire Properties at the Microarchitecture Level

In future microprocessors, communication will emerge as a major bottleneck. The authors advocate composing future interconnects of some wires that minimize latency, some that maximize bandwidth, and some that minimize power. A microarchitecture aware of these wire characteristics can steer on-chip data transfers to the most appropriate wires, thus improving performance and saving energy.

Rajeev Balasubramonian
Naveen Muralimanohar
Karthik Ramani
Liqun Cheng
John B. Carter
University of Utah

As device dimensions shrink, conventional global on-chip wires are not scaling well.[1] Reducing a wire's cross-sectional area and packing wires closely together can have a detrimental effect on wire delay. Whereas microprocessor chips in the 1990s sent a signal from one end of the chip to the other in a single clock cycle, analysts project that this delay will grow to 30 clock cycles by 2014. Researchers also report that interconnect power accounts for 50 percent of total chip dynamic power in some Intel processors.[2] Thus, any form of distant on-chip communication, which cost almost nothing in the past, will be very expensive in the future.

There are three types of distant communication on chips:

- A single thread executes instructions on functional units that are not in proximity, introducing wire delays between the execution of dependent instructions.
- A parallel application executing on a multicore chip frequently transfers data between cores that are millimeters apart.
- Address and data signals must navigate long wires when accessing a bank within a multiple-megabyte on-chip L2 cache.

In this article, we apply our interconnect design ideas primarily to the first two problem domains.

Our shift into an era of communication-bound microprocessor chips has led to a flurry of activity in the fields of VLSI, materials science, and optics—all attempting to improve the speed of communication fabrics. Recent innovations in on-chip optical interconnects[3] and transmission line technology[4] have raised hopes of circumventing the communication problem. These technologies use the medium as optical or electromagnetic waveguides, and each achieves speeds equivalent to the speed of light in that medium. Architects have a reasonable idea of the performance improvement possible if these low-latency interconnects could make on-chip communication free of cost again—they need only consult the literature of the 1990s, which assumed communication cost to be nearly zero. But that is now an unrealistic scenario. The costs of low-latency interconnects are so high that future chips will have to use them frugally.

Consider the costs associated with a transmission line: Each line must have great width, thickness, horizontal spacing, and vertical spacing. Each line also requires signal modu-

Published by the IEEE Computer Society

lation and sensing circuits, reference planes above and below the metal layer, and adjacent shielding power and ground lines.[4,5] According to a recent study, the delay and power cost of optical modulators (transmitters), even with optical interconnects, are so high that they cannot compete with regular resistance-capacitance (RC)-based wires, although these costs will decrease as device dimensions scale down.[6] (It is also possible to make RC-based wires operate at low latency, at a significant metal area cost.)

Given the costs, it is unlikely that a single metal layer can accommodate, say, 64 transmission lines to transmit a word between multiple cores. In a more realistic scenario, a chip will include a few low-latency wires, and it will be up to microarchitects to find creative ways to put these precious resources to good use.

To the best of our knowledge, only two other efforts have attempted to exploit exotic interconnects at the microarchitecture level. Beckmann and Wood propose speeding up access to large L2 caches by introducing transmission lines between the cache controller and individual banks.[5] Nelson et al. propose using optical interconnects to reduce intercluster latencies in a clustered architecture in which clusters are widely spaced to alleviate power density.[7]

We propose designing a heterogeneous interconnect layer composed of wires with differing characteristics. For an on-chip data transfer, we have the option of using a few low-latency wires or many higher-latency wires. In addition to this basic latency-bandwidth trade-off, we can leverage the latency-power trade-off by introducing the option of transferring data on slower, power-efficient wires. To fully exploit the heterogeneous interconnect, we must design the microarchitecture to be aware of not only the different wire properties but also the needs of different data transfers. Thus, the microarchitecture can map data transfers to sets of wires in a manner that optimizes performance and power.

Assuming that the costs of low-latency interconnects remain high, what is the potential for performance and power improvement with a limited number of such interconnects? Currently missing from the architecture literature are any quantitative results to answer this question. The results we present here take only a first stab at an answer. We will continue to need creative ideas to leverage low-latency interconnects.

Wire characteristics

The ultimate goal of this study is to understand the potential benefits of exotic interconnects such as transmission lines and optical interconnects, but projecting parameters for this nascent technology is difficult. Therefore, for our initial detailed study, we rely on well-understood RC-based wires to derive estimates for the latency-bandwidth and latency-power trade-offs. We also examine the effect of aggressive latency assumptions that might reflect the behavior of future transmission lines.

Delay equations

Consider traditional wires in CMOS metal layers, where a wire's delay is governed by its *RC* time constant. The following equation expresses the wire's resistance per unit length:[1]

$$R_{wire} = \frac{p}{(thickness - barrier)(width - 2barrier)} \quad (1)$$

Thickness and *width* represent the geometrical dimensions of the wire cross section, *barrier* represents the thin barrier layer around the wire that prevents copper from diffusing into surrounding oxide, and p is material resistivity.

We model capacitance per unit length as four parallel-plate capacitors for each side of the wire and a constant for fringing capacitance:[1]

$$C_{\text{wire}} = \varepsilon_0 \left(\begin{array}{l} 2K\varepsilon_{\text{horiz}} \dfrac{thickness}{spacing} + \\ 2\varepsilon_{\text{vert}} \dfrac{width}{layer\ spacing} \end{array} \right) + fringe(\varepsilon_{horiz}, \varepsilon_{vert}) \quad (2)$$

Here, $\varepsilon_{\text{horiz}}$ and $\varepsilon_{\text{vert}}$ represent the potentially different relative dielectrics for the vertical and horizontal capacitors, K accounts for Miller-effect coupling capacitances, *spacing* represents the gap between adjacent wires on the same metal layer, and *layer spacing* represents the gap between adjacent metal layers. Next we examine techniques that enable wires with varying properties.

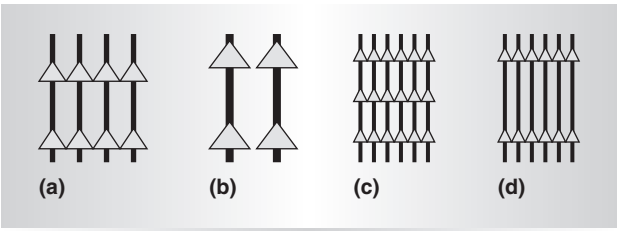

Figure 1. Wire types: B wires (a), L wires (b), W wires (c), and PW wires (d). B, L, and W wires differ in width and spacing. PW wires are W wires with a reduced size and number of repeaters.

Wire width and spacing

As Equation 1 shows, increasing the wire width can significantly decrease resistivity, while also resulting in a modest increase in capacitance per unit length (Equation 2). Similarly, increasing the spacing between adjacent wires results in a drop in C_{wire}. The overall effect of allocating more metal area per wire and increasing the wire width and spacing is that the product of R_{wire} and C_{wire} decreases, resulting in shorter wire delays. By modeling such "fat" wires, we are introducing a latency-bandwidth trade-off very similar to that of transmission lines, which require a large area per wire while yielding low latencies.

In a comparison of different interconnect strategies, Chang et al. confirm that the fat-wire model is a reasonable representation of the potential delay benefits of other exotic interconnects.[4] Delay is only one of the metrics these researchers use to compare interconnects—they also consider power, signal integrity, manufacturability, and other metrics. They show that a minimum-width global wire in 0.18-micron technology takes 1,400 ps to traverse 20 mm. A transmission line covers the same distance in 300 ps, an optical interconnect in 500 ps, and a fat wire with the same area as the transmission line covers the distance in 400 ps. Such quantitative data constantly changes with innovations, so this is not a conclusive statement on each technology's relative merits. But this data gives us confidence that fat-wire delay, which we can reliably model, is an approximate indication of the potential delay benefits of other interconnects.

Clearly, wide wires are more suitable for low-bandwidth traffic such as clock and power distribution. If global communication involves transfer of 64-bit data between cores, employing 64 wide wires can have an enormous area overhead. For a given metal area, the wider the wire, the fewer wires the area can accommodate, which leads us to the latency-bandwidth trade-off.

Figure 1 depicts various wire widths. Wires that make up a baseline interconnect with minimum width and spacing are called B wires, and those that make up an interconnect with wide width and spacing are called L wires. W wires make up an interconnect in a lower (thinner) metal layer that can accommodate wires with even smaller width and spacing than B wires.

Repeater size and spacing

Now let's examine the latency-power trade-off in wire design. A wire's resistance and capacitance are both linear functions of its length. Hence, wire delay, which depends on the product of wire resistance and capacitance, is a quadratic function of wire length. A simple technique for overcoming this quadratic dependence is to break the wire into multiple smaller segments and connect them with repeaters. As a result, wire delay becomes a linear function of wire length and depends on the number of segments, the wire delay across each segment, and the logic delay across each repeater. Designers can minimize overall wire delay by selecting optimal repeater sizes and spaces between repeaters, a technique commonly used in today's processors.

However, the repeaters incur high power overheads. Banerjee and Mehrotra report that sub-100-nm designs will include more than a million repeaters, and that optimal-size repeaters are approximately 450 times the minimum-size inverter at that technology point.[8] Thus, long wires will impose not only delay penalties but also significant power overheads.

Designers can reduce interconnect energy by employing repeaters smaller than the optimal size and increasing the spacing between them. Of course, deviating from the delay-optimal repeater configuration increases overall wire delay. Banerjee and Mehrotra developed a methodology for estimating the repeater size and spacing that minimizes power consumption for a fixed wire delay.[8] They show that in 50-nm technology, it is possible to design a repeater configuration in which the wire has twice the delay and one-

fifth the energy of a delay-optimal wire. In Figure 1, the first three sets of wires have repeaters sized and spaced to optimize delay. To exploit the latency-power trade-off, the PW wires use small, widely spaced repeaters. This design has been applied to wires with dimensions similar to W wires, but it can be applied to any interconnect.

Heterogeneous interconnects

Clearly, many different wire implementations are possible. Traditionally, if we were attempting low-latency on-chip communication for a 64-bit word, we would implement a homogeneous interconnect composed of 64 B wires. We propose a heterogeneous interconnect composed, for example, of 32 B wires and 8 L wires. Assuming that each L wire has four times the width and spacing of a B wire, such a layout has the same area cost as the homogeneous design. It is possible that the heterogeneous interconnect will yield higher performance because of its ability to transmit eight (carefully selected) bits at low latency.

For our evaluations, we will assume a network in which, on any link, the microarchitecture has the option to transmit data on either B, L, or PW wires. L wires represent the low-latency, low-bandwidth option, and PW wires represent the slow, high-bandwidth, low-power option. Once assigned to a set of wires, the message cannot switch to a different set of wires at intermediate routers between the sender and receiver. Additional multiplexers and demultiplexers at the sender and receiver provide the capability of routing an entity over multiple sets of wires in a heterogeneous interconnect. These logic blocks impose a minor delay and power overhead.

Wire management for clustered architectures

Heterogeneous interconnects are applicable to a variety of problem domains. We first examine their application to clustered architectures, which boost instruction-level parallelism for a single thread without compromising clock speed. The biggest bottleneck to high performance in a clustered architecture is the cost of wire delays between clusters.

Baseline clustered architecture

A partitioned or clustered architecture consists of many small, fast computational units connected by an interconnect fabric. Each computational unit, or cluster, typically consists of a limited number of ALUs, local register storage, and an instruction issue buffer. Because a cluster has limited resources and functionality, it enables fast clocks, low power, and low design effort. Abundant transistor budgets allow the incorporation of many clusters on a chip. A single program's instructions are distributed across the clusters, thereby enabling high parallelism. It is impossible to localize all dependent instructions to a single cluster, so data is frequently communicated between clusters over the intercluster interconnect fabric.

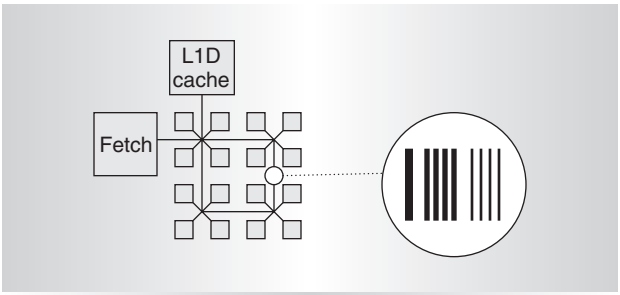

Figure 2. Baseline 16-cluster system with an example of a heterogeneous link (inset) composed of different wire types.

For our experiments, we assume processor models with either 4 or 16 clusters. Figure 2 shows a baseline 16-cluster system. In the decode and rename stage, the centralized fetch unit brings in instructions and assigns them to one of many clusters. A state-of-the-art instruction-steering heuristic attempts to distribute instructions in a manner that balances load and minimizes intercluster communication.[9] Results produced within a cluster are bypassed to consumers in that cluster in the same cycle; communicating the result to consumers in other clusters takes additional cycles. To transfer data between clusters, the instruction decode and rename stage inserts a copy instruction in the producing cluster, which places the value on the intercluster network as soon as the value is available.

Loads and stores that execute in the clusters must communicate addresses and data to and from a centralized load/store queue and data cache over the intercluster network. A centralized LSQ and cache organization entails far less complexity than a decentralized organization

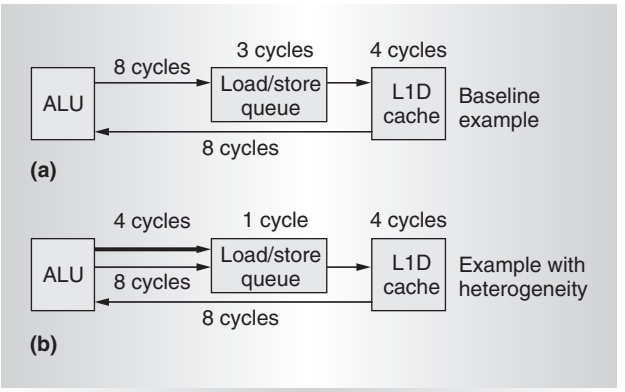

Figure 3. How heterogeneous interconnects reduce L1 cache access latencies: baseline processor without heterogeneity (a) and processor with heterogeneous interconnect (b).

and achieves similar performance. The intercluster network has a hierarchical topology—four-cluster sets connect through a crossbar, and a ring connects the four crossbars. As Figure 2 shows, each link on the network is heterogeneous, consisting of some combination of B, L, and PW wires. For this discussion, we assume 72 B wires, 18 L wires, and 144 PW wires occupy equivalent metal areas.

Accelerating cache access

Now let's examine how to improve performance by exploiting low-latency, low-bandwidth L wires. L wires are designed as either fat RC-based wires, transmission lines, or optical waveguides.

Consider the behavior of the cache pipeline in the baseline processor. Figure 3a shows an example. When a cluster executes a load instruction, it computes the effective address and communicates it to the centralized LSQ and cache. In the baseline example, this transfer takes eight cycles on conventional B wires. The LSQ waits until it receives addresses of stores before the load in program order, guarantees that there are no memory dependencies, and then initiates the cache access (this takes three cycles in the example). The communication to the cache influences load latency in two ways: It delays the arrival of load addresses at the LSQ, and it delays the arrival of store addresses at the LSQ, thereby delaying the resolution of memory dependencies. After the load accesses the L1D cache (four cycles), data goes to the requesting cluster (eight cycles). The entire operation consumes 23 cycles in the Figure 3a example.

To accelerate cache access, we propose a new technique: As Figure 3b shows, we transmit a subset of the address bits on low-latency L wires to prefetch data from the L1D cache and hide the high communication cost of transmitting the entire address. After the cluster computes the effective address, the cluster transmits the least-significant (LS) bits of the address on L wires and the most-significant (MS) bits on B wires. In the example in Figure 3b, the former transfer consumes four cycles, and the latter, eight cycles. The same happens for store addresses. Thus, the LSQ quickly receives the LS bits for loads and stores. The early arrival of partial addresses allows the following optimizations.

The LSQ can effect a partial comparison of load and store addresses with the available LS bits. If the load's LS bits don't match any earlier store's LS bits, the load is guaranteed not to have any memory dependence conflicts and it can begin cache access. In the example, cache access begins one cycle after the arrival of the load's LS bits. If the load's LS bits match an earlier store's LS bits, the load must wait for the MS bits to arrive before the LSQ determines whether there is a true dependence.

For an L1 data cache access, the cache uses the LS bits of the effective address to index into the data, tag RAM arrays, and read out a relevant set of cache blocks. The MS bits of the effective address index into the translation look-aside buffer (TLB), and the comparator logic then compares the resulting translation with the tags to select the appropriate data block and forward it to the cluster. Because accesses to the cache RAM arrays don't require the MS bits, the accesses can start as soon as the LS bits of the address arrive on L wires (provided the L wires transmit enough bits to determine the set index).

Similarly, the transfer on the L wires can include a few bits of the virtual page number. This allows TLB access to proceed in parallel with RAM array lookup. When the rest of the effective address arrives, the tag comparison selects the correct translation from a small subset of candidate translations. A highly associative TLB design might be more amenable to this modified pipeline than a fully associative one. Eighteen L wires are enough to accommodate 8 bits of cache

index, 4 bits of TLB index, and 6 bits to identify the entry in the LSQ.

Thus, transferring partial address bits on L wires enables the cluster to prefetch data from L1 cache and TLB banks and hides RAM access latency, which is the biggest component of cache access time. If the cache RAM access has completed by the time the entire address arrives, the processor spends only an additional cycle to detect the correct TLB translation and make the tag comparison before returning data to the cluster. This overlap of effective address transfer with cache RAM and TLB access can result in a reduction in effective load latency if the latency difference between L and B wires is significant. In the example with heterogeneity (Figure 3b), L1D cache access begins five cycles after the ALU computes the effective address. By the time blocks are fetched from the L1D, the entire address has arrived on slower wires. One cycle later (after the correct block has been selected), data goes back to the requesting ALU. The entire process takes 18 cycles (4 + 1 + 4 + 1 + 8), five fewer than the baseline example.

Narrow-bit-width operands

An interconnect composed of L wires can also transfer results that can be encoded in a few bits. Eighteen L wires can accommodate 8 bits of register tag and 10 bits of data. We use the simplest form of data compaction here—integer results between 0 and 1,023 are eligible for transfer on L wires. The hardware required to detect narrow-bit-width data is easy to implement—the PowerPC 603, for example, has hardware for detecting the number of leading zeros; it then uses this value to determine latency for integer multiply. A special case of transferring narrow-bit-width data is the communication of a branch misprediction back to the front end. This involves only the branch ID, which L wires easily accommodate, thereby reducing the branch misprediction penalty.

Exploiting PW wires

PW wires can reduce not only contention in other wires but also energy consumption. Our objectives are to identify data transfers that can tolerate the higher latency of PW wires, and situations in which the cost of contention on B wires offsets their wire latency advantage. If a data transfer has the option of using either B or PW wires, three criteria dictate when it should use high-bandwidth, low-energy, high-latency PW wires:

- We always assign store data to PW wires because stores are usually off the program's critical path.
- If an instruction's input operands are ready in a remote cluster's register file at the time the instruction is dispatched, we transfer the operands to the instruction's cluster on PW wires. The rationale is that there is usually a long gap between instruction dispatch and issue, and the long communication latency for the ready input operand can be tolerated.
- We keep track of the amount of traffic injected into both interconnects (B wires and PW wires) in the past N cycles (N = 5 in our simulations). If the difference between the traffic in the two interconnects exceeds a certain prespecified threshold (10 in our simulations), we steer subsequent data transfers to the less congested interconnect.

Thus, by steering noncritical data to the high-bandwidth, energy-efficient interconnect, we are likely to see little performance degradation, and by steering data away from the congested interconnect, we can potentially see performance improvement. Our simulations revealed that congestion control had a negligible effect on performance and power. Its implementation cost, on the other hand, is nontrivial. We list this optimization here because it might be applicable to other domains.

Applications to CMP cache coherence

The second relevant problem area to which we can apply heterogeneous interconnects is multicore chips, or chip multiprocessors. Most major chip manufacturers have announced CMPs; Sun's is Niagara, and IBM's are Power5 and Cell. In CMPs, an intercore network maintains coherence between each core's L1 data cache, an important requirement for executing multithreaded or parallel applications. Coherence traffic has varying latency and bandwidth needs, and a heterogeneous interconnect can meet the specific needs of each individual message. For

Table 1. Area, delay, and power characteristics of different wire implementations used in CMP architecture evaluation.

Wire type	Relative latency	Relative area (wire width + spacing)	Dynamic power (W/m)*	Static power (W/m)
B (8X plane)	1	1	2.65α	1.0246
W (4X plane)	1.6	0.5	2.9α	1.1578
L (8X plane)	0.5	4	1.46α	0.5670
PW (4X plane)	3.2	0.5	0.87α	0.3074

* α = switching factor

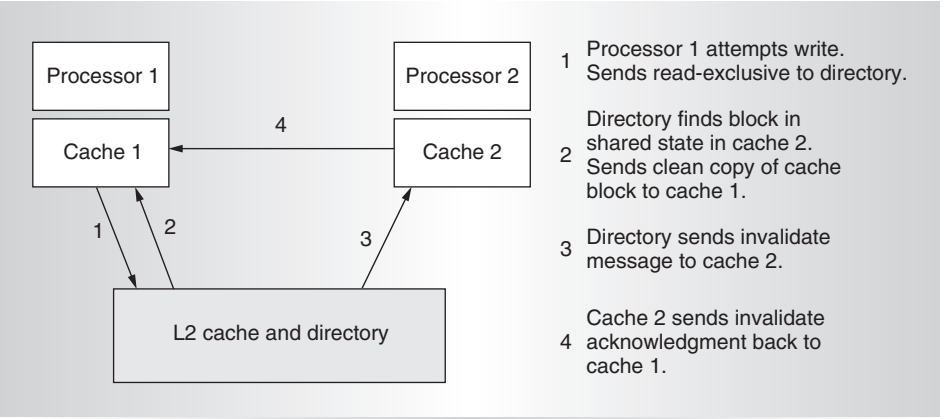

Figure 4. Example of hop imbalance in a directory-based cache coherence protocol.

this discussion, we assume that each processing core has a private L1 data cache, and that all the cores share the L2 data cache. We assume that, as in industrial implementations, the CMP maintains coherence among L1s through a directory-based protocol, with directory state maintained at the L2 cache.

The following novel mechanisms map coherence traffic to sets of wires for improved performance and power. Many other applications of heterogeneous interconnects within a cache coherence protocol (including snooping-based protocols) are possible.[10]

Hop imbalance

Figure 4 shows an example of hop imbalance in a directory-based cache coherence protocol. When a processor's write request goes to the directory, the directory takes the following steps, depending on the state of the block in other caches: Typically, the directory sends a speculative data block back to the requesting processor and contacts other caches that share or own that data block to invalidate their copies. These caches must then send acknowledgments or the latest cache copy to the requesting processor. Thus, part of the directory's response (the data block) reaches the requesting processor in a single hop, whereas the remaining responses take two hops because they must go through other sharers. The requesting processor must wait for all acknowledgments before it can proceed.

Because of the hop imbalance, the data block's one-hop transfer is usually not on the critical path, while the two-hop transfers usually are. This provides the opportunity to transmit one-hop messages on slower on-chip wires optimized for power and bandwidth (PW wires), without degrading system performance.

Narrow-width messages

Many messages within a coherence protocol don't involve address or data transfers. For example, if the directory is busy, it sends back a negative acknowledgment (NACK) message to the requestor and identifies the relevant cache block by the entry within the requestor's miss-status-holding registers. Similarly, messages such as acknowledgments or replies for

an upgrade request don't contain addresses or data and therefore consist of fewer than 20 bits. All these messages can be transmitted on low-bandwidth wires optimized for low latency (L wires), potentially yielding performance improvements.

Writeback messages

Writebacks arising from cache evictions are almost always off the critical path and can tolerate longer latencies. Such messages are ideal candidates for transfer on bandwidth- and power-optimized PW wires.

Evaluation

We evaluated the proposed heterogeneous interconnect in a clustered architecture and a CMP. Detailed methodologies for the two evaluations appear elsewhere.[9,10] Here, we describe only the salient properties of our simulation platform.

Simulation methodology

We computed the relative characteristics of B, L, and PW wires using International Technology Roadmap for Semiconductors projections and analytical equations from other researchers.[1,8] We assumed that B wires were minimum-width wires on the 8X metal plane, a commonly used global-wire configuration, and W wires were minimum-width wires on the 4X metal plane. For L wires, we quadrupled the metal area of B wires. For PW wires, we adopted a repeater configuration for minimum-width 4X wires (W wires) that minimizes power for a given delay penalty.[8] (The delay penalties for the intercluster and cache coherence networks are 20 percent and 100 percent, respectively.) Table 1 shows the set of parameters for 65-nm technology that we used in evaluating the CMP architecture.

The clustered architecture has the same network topology as that in Figure 2. For the CMP evaluation, we connected the L1 caches and the L2 cache controller with a crossbar-based hierarchical interconnect similar to SGI's NUMAlink4 interconnect. Every network link is fully pipelined. Our interconnect power models take into account the effect of additional latches for links with long latencies. We also modeled power consumption within routers, including separate buffers for each set of wires.[11] For a network using virtual-channel flow control, we treat each set of wires in the heterogeneous network link as a separate physical channel and maintain the same number of virtual channels per physical channel, as in the baseline. Therefore, the heterogeneous network has a larger total number of virtual channels, and the routers require more state fields to keep track of these additional virtual channels than a baseline homogeneous network. Data transfers in the clustered architecture are not wider than a (64-bit) word and are not broken into packets. Data transfers on the coherence network are packet switched.

Uncontended latency for intercluster communication ranges from two to six cycles on B wires, one to three cycles on L wires, and three to nine cycles on PW wires. Uncontended latency for each hop on the coherence network is four cycles on B wires, two cycles on L wires, and 13 cycles on PW wires.

We modeled the clustered architecture in SimpleScalar and evaluated it on SPEC-cpu2000 benchmarks. We simulated the 16-core CMP on the functional simulator Simics, augmented with the General Execution-Driven Multiprocessor Simulator (GEMS) timing model.[12] We modeled the CMP cores as in-order processors and used a one-level MOESI (modified, owned exclusive, shared, invalid) directory-based protocol to maintain coherence among the L1s. We report simulation results for the parallel phases of all programs in the Splash-2 suite.

Clustered architecture results

Let's look first at how L wires enable the optimizations we described earlier. Figure 5 shows throughput in instructions per cycle (IPC) for SPEC2000 programs for two 16-cluster systems. The first is our baseline organization, which has only one interconnect layer composed entirely of B wires. Each link can transfer 64 data bits and 8 tag bits in each direction. In the second 16-cluster system, the baseline interconnect is augmented with another metal layer composed entirely of L wires. Each link on this layer can transfer 18 data bits in each direction.

For this experiment, we aggressively assumed that the latency for each hop on an L wire was only one cycle. The processor used L wires to send the LS bits of a load or store effective

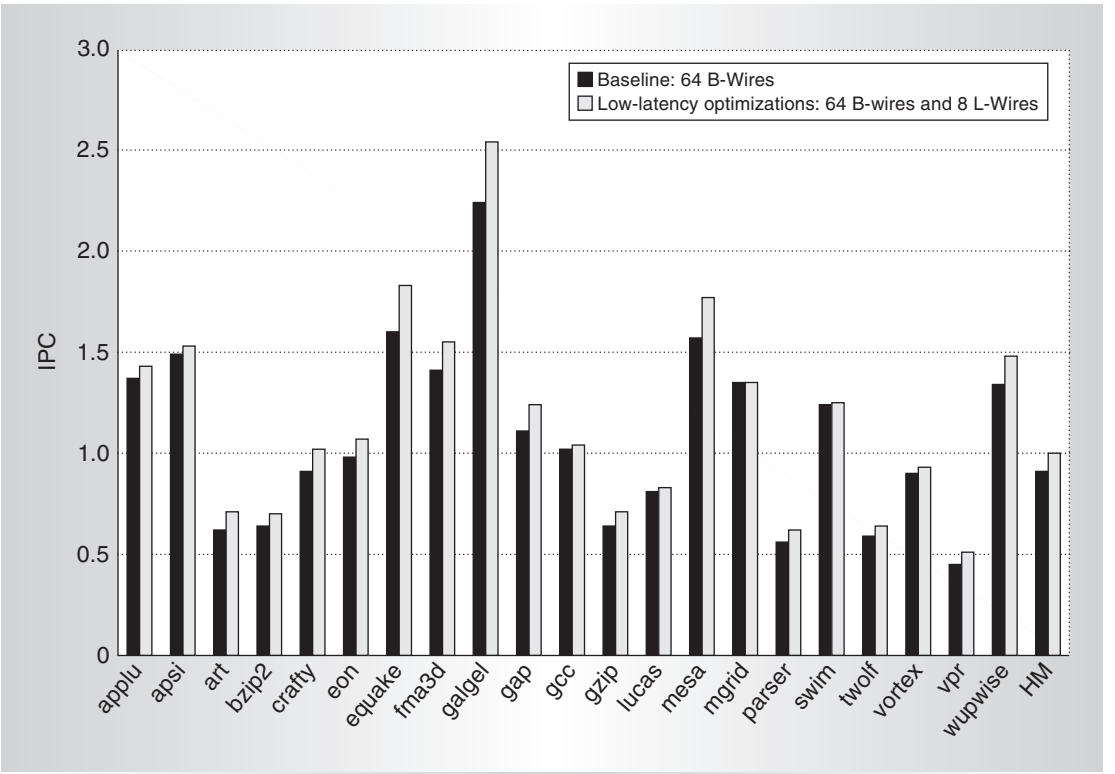

Figure 5. Instructions per cycle (IPC) throughput for baseline 16-cluster partitioned architecture using one layer of B wires and for partitioned architecture using one layer of B wires and one layer of L wires. The L wires transmit narrow-bit-width data, branch mispredict signals, and LS bits of load/store addresses.

address and to transfer narrow-bit-width data. We found that overall performance improved by 10 percent, when averaging all of the programs. For each component modeled individually, the performance improvements were

- 5 percent for the novel cache pipeline,
- 4 percent for narrow-width operands, and
- less than 1 percent for branch mispredict signals.

Of the 21 programs we studied, 12 yielded performance improvements of 10 to 15 percent. This is not surprising considering that access to the data cache is a key operation in most programs and has a significant impact on overall performance. Instruction set architectures such as the x86, which have fewer registers and more loads and stores, are likely to exhibit higher overall performance improvements from speeding up the cache pipeline (our benchmark executables are in the Alpha AXP ISA). Improvements will be greater in future technology levels that are even more wire constrained. Furthermore, we have barely scratched the surface in terms of creative techniques for leveraging L wires.

For the rest of our analysis, we assume L wires implemented as fat wires, with each hop on the ring network consuming two cycles. This allows us to make credible quantitative assessments of IPC and power with today's mature technology. Doubling the latency on the ring network reduces the benefit of L wires to 8 percent. The improvement decreases to 4 percent for a four-cluster system. Wire latencies are less a bottleneck in four-cluster systems, and in keeping with Amdahl's law, interconnect optimizations yield lower improvements for such systems.

The preceding evaluation shows performance improvements from the addition of a metal layer composed entirely of L wires. Although this helps gauge L wires' potential to reduce the cost of long wire latencies, it is not a fair comparison because of the systems' difference in number of metal layers. Let's turn

Table 2. Heterogeneous interconnect energy and performance for a 16-cluster system.

Model	Link description	Relative metal area	IPC	Relative dynamic energy*	Relative leakage energy*	Relative energy-delay*	Comments
1	144 B wires	1.0	0.91	100	100	100	High performance
2	288 PW wires	1.0	0.83	39	81	89	
3	144 PW wires, 36 L wires	1.5	0.90	36	47	79	Low energy-delay
4	288 B wires	2.0	0.96	99	190	102	
5	144 B wires, 288 PW wires	2.0	0.93	84	169	98	
6	288 PW wires, 36 L wires	2.0	0.93	36	81	79	Low energy-delay
7	144 B wires, 36 L wires	2.0	1.00	89	99	88	High performance
8	432 B wires	3.0	0.99	98	275	105	
9	288 B wires, 36 L wires	3.0	1.02	88	187	93	High performance
10	144 B wires, 288 PW wires, 36 L wires	3.0	0.98	61	170	88	Low energy-delay

* All energy and energy-delay values are normalized with respect to Model 1. Energy-delay is computed by multiplying total processor energy by number of executed cycles.

now to comparisons that help determine the best use of available metal area.

We'll start by evaluating a processor model (similar to the baseline in Figure 5) that has only enough metal area per link to accommodate either 144 B wires (72 in each direction), 288 PW wires, or 36 L wires. Then we'll examine processors that have twice and thrice as much metal area, allowing more interesting combinations of heterogeneous wires. In some cases, high-bandwidth interconnects enable the transfer of multiple words in a single cycle.

Table 2 summarizes performance and energy characteristics of interesting heterogeneous interconnect organizations for the 16-cluster system. All energy and energy-delay values are normalized with respect to the values for Model 1, which contains 144 B wires per link. We compute energy-delay by multiplying total processor energy with the number of cycles used to execute 100 million instructions. Total processor energy assumes that interconnect energy accounts for 36 percent of total chip energy in Model 1 (similar to estimates derived for the RAW partitioned architecture[11]) and that leakage and dynamic energy are in the ratio 3:10 for Model 1. Additional data—including metrics such as energy × delay^2, different assumptions on the contribution of interconnect energy, and results for four-cluster systems—is available elsewhere.[9]

Our results indicate that for various metal area budgets, heterogeneous wires potentially can significantly improve performance and energy characteristics, compared with a baseline approach that employs homogeneous wires. For Model 10, for example, the percentage of messages on B, L, and PW wires are 33, 29, and 38, respectively. We found overall processor energy-delay reductions of 14 to 21 percent from employing energy-efficient and low-latency wires. We have also seen similar improvements for other metrics such as energy × delay^2.

CMP results

Figure 6a shows performance speedups for the CMP model with a heterogeneous interconnect. Each unidirectional link in the baseline model is 75 bytes wide (64 data bytes, 8 address bytes, and 3 bytes of control information) and composed entirely of B wires on the 8X metal layer. The heterogeneous interconnect has an identical metal area cost and consists of 24 L wires, 256 B wires, and 512 PW wires. The assignment of messages to each set of wires follows the criteria given earlier. We observed an average performance improvement of 11 percent. Most of this improvement is attributable to the transfer of control messages on L wires. Approximately one-third of these messages are ACK and NACK messages, and two-thirds are messages that place a block in and out of transient states in the directory. Hop imbalances in coherence transactions (caused by read-exclusive requests for a block in shared state) are uncommon in

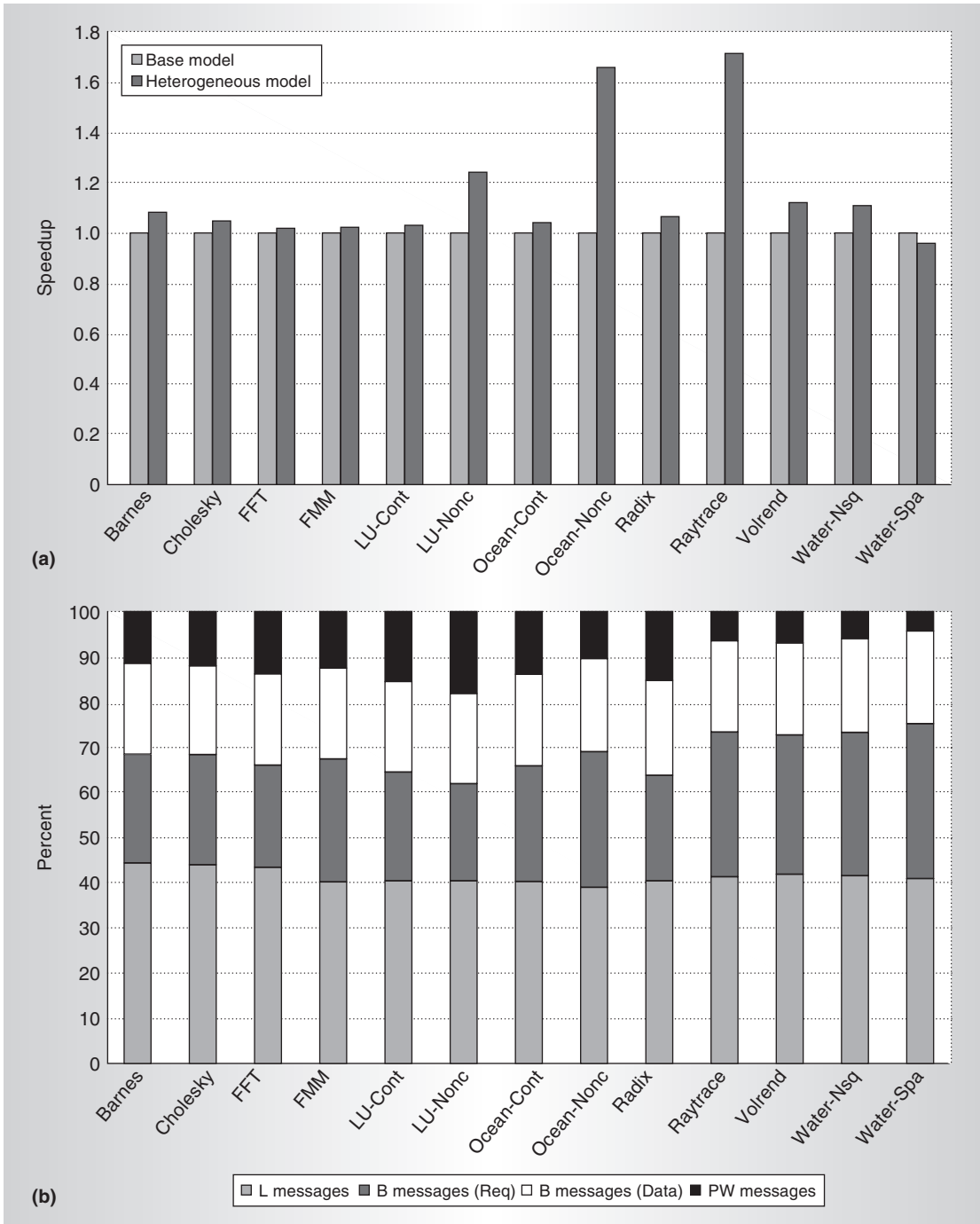

Figure 6. Behavior of heterogeneous interconnect in the CMP architecture: speedup of heterogeneous interconnect (a); message distribution on the heterogeneous network (b). B wire transfers are classified as Request and Data.

Splash-2 programs and contribute negligibly to the performance improvement.

Figure 6b shows the percentage of messages sent on each set of wires. Message transfers on L wires and PW wires are more power-efficient, leading to a 22 percent savings in network energy.

Many efforts are being made to improve the speed of on-chip interconnects.

While high speed appears to be a tangible goal in the future, it is unlikely that the cost will be low. Evaluations of our heterogeneous interconnect indicate performance improvements on the order of 10 percent and network energy savings up to 60 percent. Interconnects also form a significant part of the L2 cache, especially recently proposed nonuniform cache architectures (NUCAs). In future work, we will consider the application of heterogeneous interconnects to NUCA organizations. L wires can speed the search of data blocks in a NUCA cache, and PW wires can lower the power cost of noncritical operations, such as data block writeback and migration. For future work, we will also derive cache access models that parameterize latency and power as a function of interconnect choices. Such tools can abstract interconnect properties in a manner that enables architects to derive performance- or power-optimal cache organizations. MICRO

References

1. R. Ho, K.W. Mai, and M.A. Horowitz, "The Future of Wires," *Proc. IEEE*, vol. 89, no. 4, Apr. 2001, pp. 490-504.
2. N. Magen et al., "Interconnect-Power Dissipation in a Microprocessor," *Proc. Int'l Workshop System-Level Interconnect Prediction* (SLIP 04), ACM Press, 2004, pp. 7-13.
3. Q. Xu et al., "Micrometre-Scale Silicon Electro-Optic Modulator," *Nature*, vol. 435, no. 7040, 19 May 2005, pp. 325-327.
4. R. Chang et al., "Near Speed-of-Light Signaling Over On-Chip Electrical Interconnects," *IEEE J. Solid-State Circuits*, vol. 38, no. 5, May 2003, pp. 834-838.
5. B.M. Beckmann and D.A. Wood, "Managing Wire Delay in Large Chip-Multiprocessor Caches," *Proc. 37th IEEE/ACM Int'l Symp. Microarchitecture* (Micro 37), IEEE CS Press, 2004, pp. 319-330.
6. M. Haurylau et al., "On-Chip Optical Interconnect Roadmap: Challenges and Critical Directions," *Proc. 2nd IEEE Int'l Conf. Group IV Photonics*, IEEE Press, 2005, pp. 17-19.
7. N. Nelson et al., "Alleviating Thermal Constraints while Maintaining Performance via Silicon-Based On-Chip Optical Interconnects," *Proc. Workshop on Unique Chips and Systems* (UCAS 1), 2005, pp. 45-52.
8. K. Banerjee and A. Mehrotra, "A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs, *IEEE Trans. Electron Devices*, vol. 49, no. 11, Nov. 2002, pp. 2001-2007.
9. R. Balasubramonian et al., "Microarchitectural Wire Management for Performance and Power in Partitioned Architectures," *Proc. 11th Int'l Symp. High-Performance Computer Architecture* (HPCA 05), IEEE CS Press, 2005, pp. 28-39.
10. L. Cheng et al., "Interconnect-Aware Coherence Protocols for Chip Multiprocessors," *Proc. 33rd Int'l Symp. Computer Architecture* (ISCA 06), IEEE CS Press, 2006, pp. 339-351.
11. H.-S. Wang, L.-S. Peh, and S. Malik, "Power-Driven Design of Router Microarchitectures in On-Chip Networks," *Proc. 36th IEEE/ACM Int'l Symp. Microarchitecture* (Micro 36), IEEE CS Press, 2003, pp. 105-116.
12. M. Martin et al., "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 4, Nov. 2005, pp. 92-99.

Rajeev Balasubramonian is an assistant professor in the School of Computing at the University of Utah. His research interests include the design of high-performance microprocessors that can efficiently tolerate long on-chip wire delays, high power densities, and frequent soft errors. Balasubramonian has a BTech in computer science and engineering from the Indian Institute of Technology, Bombay, and an MS and a PhD, both in computer science, from the University of Rochester. He is a member of the IEEE.

Naveen Muralimanohar is pursuing a PhD in computer science at the University of Utah. His research interests include interconnect design for future communication-bound processors and microarchitectural techniques to exploit special wires in an on-chip network. Muralimanohar has a bachelor's degree in electrical and electronics engineering from the University of Madras, India.

Karthik Ramani is pursuing a PhD in the School of Computing at the University of Utah. His research interests include workload

characterization of computationally intensive applications, compilation techniques for embedded architectures, and temperature-aware interconnect architectures. Ramani has a bachelor's degree in electronics and communication engineering from the University of Madras, India, and an MS in electrical engineering from the University of Utah. He is a student member of the IEEE and the ACM.

Liqun Cheng is pursuing a PhD in the School of Computing at the University of Utah. His research interests include computer architecture, parallel and distributed computing, system simulators, performance analysis, and advanced memory systems. Cheng has a BS in computer science from Shanghai Jiao Tong University. He is a student member of the IEEE, the IEEE Computer Society, and the ACM.

John B. Carter is an associate professor in the School of Computing at the University of Utah. His research interests include computer architecture, operating systems, distributed systems, networking, and parallel computing, with particular emphasis on memory and storage systems. Carter has a BS in electrical and computer engineering and an MS and a PhD, both in computer science, all from Rice University.

Direct questions and comments about this article to Rajeev Balasubramonian, 50 S. Central Campus Dr., Rm. 3190, Salt Lake City, UT 84112; rajeev@cs.utah.edu.