A Case for Globally Shared-Medium On-Chip Interconnect*

Aaron Carpenter, Jianyun Hu, Jie Xu, Michael Huang, and Hui Wu
Dept. of Electrical & Computer Engineering, University of Rochester
Rochester, New York 14627
{aaron.carpenter, jianyun.hu, jie.xu, michael.huang, hui.wu}@rochester.edu

ABSTRACT

As microprocessor chips integrate a growing number of cores, the issue of interconnection becomes more important for overall system performance and efficiency. Compared to traditional distributed shared-memory architecture, chip-multiprocessors offer a different set of design constraints and opportunities. As a result, a conventional packet-relay multiprocessor interconnect architecture is a valid, but not necessarily optimal, design point. For example, the advantage of off-the-shelf interconnect and the *in-field* scalability of the interconnect are less important in a chip-multiprocessor. On the other hand, even with worsening wire delays, packet switching represents a non-trivial component of overall latency.

In this paper, we show that with straightforward optimizations, the traffic between different cores can be kept relatively low. This in turn allows simple shared-medium interconnects to be built using communication circuits driving transmission lines. This architecture offers extremely low latencies and can support a large number of cores without the need for packet switching, eliminating costly routers.

Categories and Subject Descriptors: B.4.3 [Interconnections (subsystems)]: Topology C.1.4 [Parallel Architectures] C.2.1 [Network Architecture and Design] : Network Topology

General Terms: Design, Performance

Keywords: Transmission Line, On-chip Interconnect

1. INTRODUCTION

Mainstream microprocessors already include a handful of high-performance cores in each chip. As the scale increases, a natural component in these chip-multiprocessors is a high-performance on-chip interconnect. While conventional designs used in multiprocessors are obvious candidates, the on-chip environment offers a different set of constraints and opportunities for new designs and optimizations.

Because of the relatively limited market, traditional parallel machines often use commercial, off-the-shelf components such as microprocessors, chip-sets, and routers. They also use scalable designs that can be configured (in the field) for different sizes [4,33]. Packet-switched networks fit the bill for the interconnect needs: existing routers can be directly used in multiple system designs and configurable routing tables allow easy customization for a scalable interconnect.

With chip-multiprocessors, each implementation only needs to deal with a fixed configuration. Thus, there is room for niche designs that do not scale to large configurations. Furthermore, future general-purpose chips are not necessarily destined to all be "many-core" designs. Increasing core count can lead to diminishing returns of utility. Extra transistor budget can find fruitful deployment in storage, specialized accelerators, and continued integration of traditionally discrete system components. Even given a large number of cores built in a chip, the actual communication demand is not automatically high. Consider a fairly common use of a multi-core chip: as a throughput engine processing largely independent tasks. There is little inherent need for inter-core communication.

In short, not all general-purpose chips need high, scalable on-chip communication bandwidths. And providing scalable bandwidths is not for free. Recent studies have warned about the potential costs of powerful interconnects [32] and argued for the continued use of (optimized) digital buses with extensions of limited packet-switching for better scalability [45].

Considering these factors, there may be candidate designs for on-chip interconnect that trade off high bandwidth and scalability (and the associated cost) for improvements in other metrics of interest, such as latency and energy efficiency. As the on-chip interconnect offers a large design space and is related to many different aspects of chip-multiprocessor design, navigating the design space is an inherently imprecise process that requires iterative, community efforts. In this paper, we attempt to make a case for an interconnect that is free of packet switching/relay, explicit or implicit. The absence of packet switching allows

*This work is supported in part by NSF under the grants 0901701, 0829915, and 0747324.

ISCA'11, June 4–8, 2011, San Jose, California, USA.

a shared medium to be used as a communication channel. In particular, this allows us to take advantage of communication circuit technologies driving on-chip transmission lines. We can achieve tens of Gb/s of signaling rate and a propagation velocity approaching the speed of light, all at an energy cost that is a fraction of a conventional packet-switching interconnect. We believe that such a design would be a serious option for general-purpose chips.

The rest of the paper is organized as follows. Section 2 discusses background and related work of on-chip interconnect. Section 3 describes a shared-medium, relay-free interconnect design. Section 4 presents experimental analysis of the design. Section 5 concludes.

2. BACKGROUND AND RELATED WORK

Early multiprocessors used buses for on-chip communication. As the speed of the processor increased, the bus's inability to provide high throughput became a bottleneck. Larger-scale multiprocessors (*e.g.*, SGI Orion [33]) and other parallel systems (*e.g.*, IBM SP2 [4]) use packet-switching routers that can provide different connection topologies and configurations depending on the system's size. While earlier systems used I/O buses to connect processors with routers, further integration allows the chip to contain both processor core and router for interconnection [35].

With the integration of multiple cores on a single die, proposals of advanced interconnection have emerged. These proposals range from networks-on-chip (NoC) [7, 23, 38, 43, 50] to optical interconnects [22, 30, 31, 47, 53] or RF interconnects [8, 13–15, 41]. Transmission line mechanisms have been explored by Beckmann and Wood to enhance the communication in L2 caches [8,9]. Recent proposals use RF circuitry as an acceleration mechanism to supplement the conventional mesh network [13, 14]. To supply sufficient bandwidth, the transmission line is used as a multi-band medium, at the cost of complexity in transceiver design. Finally, transmission lines are also used in building fast synchronization mechanisms [39].

In the commercial chip space, most designs with more than a handful of cores still use conventional electrical interconnect, such as Intel's 80-core ring topology [46] or IBM's Cell Broadband Engine's Element Interconnect Bus [5].

In the domain of packet-switched on-chip interconnect, there are various proposals to optimize the canonical design. Muralimanohar et al. use varied wires for different purposes to reduce the long-latency, high-power communication [36, 37]. Sanchez et al. analyze different packet-switched topologies in order to find the optimum design [43]. Kim changes the router design to reduce area and power overheads associated with the on-chip routing [29]. Hierarchical structures are being proposed to avoid relying on full-fledged packet-switching architecture [19, 45]. It is worth noting that such a design does not necessarily eliminate packet switching. For instance, the hubs connecting the bus segments need to buffer packets and arbitrate for the next segment, essentially serving as a router, albeit with fewer input and output ports [45].

While not directly related to interconnect architecture, plenty of work has been done on data placement optimizations such as page coloring [6, 16, 27], data migration [28, 54], and cache clustering [38]. The result of these innovations is a reduction of interconnect traffic, and hence, a decrease in the demand for more raw throughput of the underlying interconnect hardware.

Insightful arguments have been made to show that the interconnect needs to be co-designed with other parts of the system, such as the coherence and on-chip caching strategies, and underlying circuit techniques [19, 32, 43, 45]. It is inherently difficult to obtain global optimality in a large co-design space. A primary challenge lies in accurate modeling of multiple metrics (*e.g.*, performance and energy) of interest. The accuracy depends heavily on that of the parameters and other assumptions. Furthermore, some pros and cons involved in different design options are qualitative and can only be analyzed subjectively. Therefore, we believe it will take iterative, collective efforts of the community to develop insights and approximations to help hardware designers successfully navigate the design space. In this paper, we attempt to make a narrow point that with reasonable effort of planning what traffic uses the on-chip interconnect, and considering the practical scale of general-purpose chips, the bandwidth demand can be provided by a shared-medium fabric using high-performance communication circuitry without any packet switching or relay. The resulting design can have better latencies and a much better energy efficiency.

3. GLOBALLY SHARED-MEDIUM ON-CHIP INTERCONNECT

As microarchitecture becomes more complex, data communication occurs more and more frequently and is becoming increasingly explicit. Earlier systems used wires to carry data from one logical unit to another without drawing designers' attention. Gradually, wire delays were accounted for and pipeline stages were added explicitly to carry data. The difficulties in routing wires have long prompted researchers to call for a more general-purpose communication substrate to carry standardized, containerized payloads (packets/flits) as opposed to provisioning ad-hoc data passages. This argument is aptly summarized into the slogan "route packets, not wires" [18, 44]. As chips integrate more and more cores, packet-switched interconnect seems to be accepted by many as the default solution for inter-core fabric.

This model has many appealing aspects. A common fabric provides an economical way of supporting different pathways between connected entities. Different types of payloads reuse the same passageway. However, there are limitations to packet-switching as well. Every stop the payload goes through has non-trivial handling that adds latency and energy overhead and demands significant hardware router infrastructure. Packetization also adds overhead at the source and destination that only becomes negligible when the distance traveled is sufficiently

long. Indeed, in existing chip-multiprocessor designs, on-chip packet-switched fabric only serves as the backbone network connecting multiple "nodes". Within a node, a variety of fabrics, such as crossbars and point-to-point links, are used to connect components. As such, how many nodes are being connected and how much traffic is there between these nodes are important factors to decide interconnect design. For certain chips, a more suitable implementation of the backbone may be a shared-medium system that delivers sufficient bandwidth without the delay and energy overhead of packet relays. In particular, transmission lines and appropriate transceiver circuitry provide the opportunity to build high-bandwidth interconnect without packet switching or relay.

In this section, we first discuss the capabilities of a transmission-line link under current VLSI technology (Sec. 3.1); then discuss some factors that affect the actual backbone traffic demand (Sec. 3.2); and describe a bus architecture and circuit design (Sec. 3.3 and 3.4).

3.1 Transmission Line Link Design Space

Properly designed transmission lines and transceiver circuitry allow low-latency propagation of signals over long distances and can potentially support intra-chip network traffic. Finding an optimal solution, however, requires comprehensive exploration of the design space that is beyond the scope of this paper. In the following, we will briefly discuss this design space and focus on a design point that has relatively low costs compared to other options that offer even higher performance.

There are two interrelated elements in designing a transmission-line-based communication link: the transmission line medium and the transceiver circuitry. Figure 1 shows a few popular layouts of transmission lines. Other than their dimensions, microstrips are similar to conventional digital lines and indeed can be driven by purely digital drivers. This option is perhaps the simplest and in isolation, each line can support a rather high data rate ($> 20Gb/s$) thanks partly to the ever increasing speed of transistors in CMOS technology. However, crosstalk from neighboring lines either requires very large spacing for mitigation or can significantly limit the actual achievable data rate.

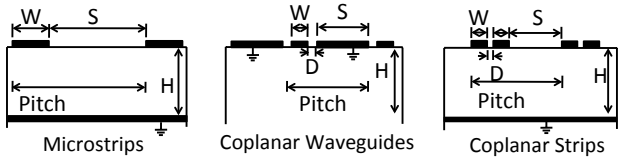


Figure 1: Popular layouts of transmission lines.

One option is to use a grounded strip in between the signal lines (forming coplanar waveguides) to mitigate the crosstalk. However, significant spacing between signal lines and thick dielectric heights may still be required. Another option is to use the more noise-tolerant differential signaling on a pair of lines, commonly called coplanar strips.

Depending on the characteristics of transmission lines and the desired signaling speed, one can use just digital transmitters and receivers, or something more complex: An analog receiver allows more attenuation and thus higher rates than digital systems. For even higher data rates, analog transmitters can be used together with more sophisticated encoding schemes, such as phase-shift-keying.

Due to the frequency-dependent loss of the transmission line, the bandwidth is limited and the transmitted data suffers from inter-symbol interference (ISI), which then limits the transmission speed and the length of the transmission line. Equalization techniques can be applied to the transmitter and receiver to resolve this issue [34, 48]. Equalization increases the data rate, but at the (potentially significant) cost of more power consumption, larger chip area, longer latency, and higher system complexity. Another option is to insert repeaters, similar to conventional wire designs, but with fewer repeaters and faster propagation. However, this would require either multiple unidirectional buses or a ring topology, as well as higher power consumption due to the extra circuitry per hop.

Figure 2 qualitatively illustrates the design space. For a rough sense of the data rate achievable in these different setups, the figure shows data rates achieved in a few sample design points. Note that these designs are not fine-tuned and only serve to give an approximate scale.

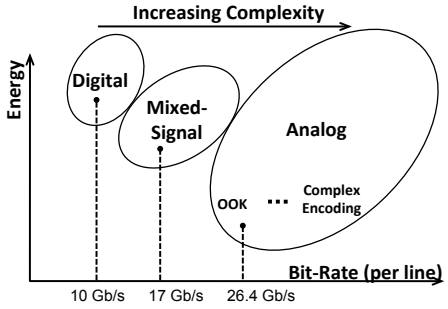


Figure 2: An illustration of the transmission line link design space. The bit-rates indicate validated design points in this space. They do not represent the maximum speed achievable.

In this paper, we use coplanar strips, as they utilize the space of the top metal layer more efficiently than the microstrips or coplanar waveguides. We use basic differential transmitters and receivers, scaled inversely, without any equalization [42]. To see if such a setup is sufficient for building a backbone interconnect, we make some simple assumptions about dimensions. Assuming a 2cm×2cm chip divided into 16 nodes, the longest distance between any 2 nodes on a single bus meandering through all nodes would be about 7.5 cm long.

Figure 3 shows a schematic of the differential-pair design point of a transmission-line link (TLL). Simulations show that a data rate of $26.4Gb/s$ can be achieved (Sec. 4.2) for a pair of transmission lines with a total ptich (including spacing) of $45\mu m$. Within 2.5mm of space, this pitch allows 55 pairs to be laid out, allowing $1.45Tb/s$ of total

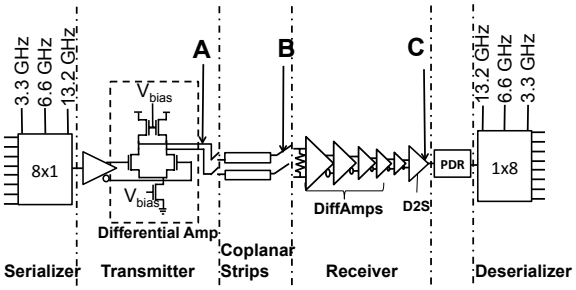


Figure 3: Link diagram with differential transmitter and receiver. D2S is a differential-to-single-ended circuit, converting the differential signals to a full swing digital signal.

bandwidth. Depending on various factors impacting traffic demand, a straightforward backbone interconnect based on tranmisison line links can be a good design option for general-purpose chip multiprocessors.

3.2 Traffic Demand

Node structure: In conventional multiprocessors, the packaging of the microprocessor dictates the boundary of nodes to be interconnected. A processor core is packaged with its local caches into a chip. Therefore any traffic between the core and the on-chip caches does not go through the interconnect. Sometimes, multiple chips are part of one node and share the same router.

With chip-multiprocessors, there is more flexibility to determine what on-chip communication uses the packetized interconnect. A baseline assumption often made in literature is that a chip consists of tiles, each with a core, an L1 cache, and a slice of a globally shared L2 (last-level) cache. In such a system, if an L1 miss occurs, the access will result in a packet injected into the interconnect if the address maps to a remote node. Otherwise, the L1 miss is served by the local L2 bank. Intuitively, it pays to have a direct interface between the L1 and its local L2 slice, rather than forcing the local request to also go through the router. This avoids adding a whole sequence of unnecessary overheads – packetizing/depacketizing and router pipeline delay – for local traffic. And in certain cases (*e.g.*, single-threaded applications when the data is mapped locally to the L2 bank) almost all traffic is local. In short, it is worthwhile to add an ad hoc channel between L1 and its local L2 slice.

In a similar vein, there may be a benefit of clustering a small number of cores and L2 slices into a node (and concentrate interconnect demand). In such a system, the backbone network only makes a stop at every node. This organization of cores requires an intra-node fabric (*e.g.*, crossbar) that connects multiple L1 caches and the L2 cache banks in the node.

In terms of performance, clustering adds extra latency for accesses from an L1 cache to the nearest L2 bank (*e.g.*, Figure 4-b Core0 to $L2_0$) that would otherwise have been connected with a smaller intra-node fabric (Figure 4-a). However, it makes accessing neighboring cache banks within the node (*e.g.*, Figure 4-b Core1 to $L2_0$) somewhat faster than without clustering, as packetization is avoided.

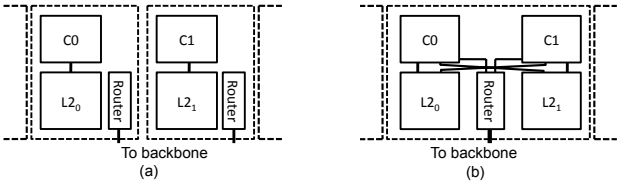


Figure 4: Node structures: (a) single-core nodes; and (b) clustering two cores into a node on the interconnect.

Moreover, it reduces the number of hubs a long-distance packet needs to traverse through and reduces the amount of traffic on the inter-node fabric. The extra cost of a larger intra-node fabric offsets the savings due to a lower number (and thus cost) of hubs for inter-node fabric. As Kumar et al. pointed out, sharing more than a few ways with crossbars quickly becomes prohibitive in cost [32]. Intuitively, the sweet spot would be a small number of cores per node (*e.g.*, 1-4). Even a small degree of clustering will reduce the number of nodes and the total traffic on the backbone.

Minimizing horizontal traffic: To sustain high-speed processing, each core demands sufficient "vertical" bandwidth to fetch data from lower levels in the memory hierarchy all the way up to the core. Ideally, this vertical bandwidth is being provided by dedicated links between different levels of caches in the core's node. However, depending on the address mapping, the data may be physically located on a cache in a remote node, incurring demand for "horizontal" bandwidth. Much research has been done to optimize the location of data to avoid unnecessary horizontal traffic. For instance, data can be mapped either statically or dynamically to the node where it is most often accessed or migrated there at runtime [6, 16, 27]. Such optimizations are important in their own right and will, as a side effect, significantly reduce the demand on the backbone, further strengthening the appeal of shared-medium, relay-free solutions.

On-chip accelerators: As the transistor budget keeps increasing while the power budget continues to be tight, on-chip accelerators are increasingly popular [26]. Because of their custom-designed nature, accelerators are typically far more energy-efficient than general purpose cores. At the same time, they can have much lower duty cycles due to the special-purpose nature. More prevalent use of accelerators potentially leads to lower pressure on the interconnect scalability. Furthermore, accelerators use more hardwired logic and are more likely to exhibit simpler, more predictable access patterns such as streaming patterns [24]. Shared-medium structures are a good match for streaming larger segments of data.

In summary, communication in a chip-multiprocessor is carried out on a collection of fabrics; many architectural factors impact how much traffic depends on the backbone. Hence, sacrificing scalability of the backbone to achieve better energy efficiency and latency can be a viable alternative.

3.3 Bus Architecture

As discussed earlier, even with simple transmitter and receiver circuitry, transmission line links (TLLs) can offer high data rates. We now discuss architectural design issues in building a bus backbone. Figure 5 shows an overview of the interconnect sub-system. Each node uses a high-speed communication circuit to deliver packets over shared transmission lines connecting all nodes. Note that unlike the conventional notion of a bus that often implies broadcast capability, our bus is merely a shared medium that allows point-to-point communication. Prior to the transfer of payload data on the bus, we perform two setup operations.

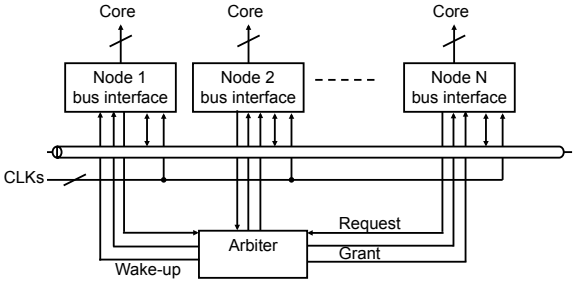


Figure 5: Overview of the bus-based communication subsystem.

Arbitration: The use of a shared-medium bus structure requires an arbitration mechanism. We can either use explicit permission granting or use carrier sensing for collision detection. We use the former. The two approaches offer almost opposite tradeoffs. Collision detection offers good latency in a contention-free environment but utilizes the bandwidth poorly: a synchronized/slotted system can not provide more than 36% of the channel capacity [40]. Granting explicit permission allows high utilization of the bandwidth at the expense of possible extra latencies, which better suits our bus. While any implementation of a permission granting system works, we use a centralized system which can be thought of a centralized token ring. Because the ring is centralized, the "token" can quickly pass to the next requester.

Receiver wakeup: For energy efficiency, the receivers operate in two modes. When the message is intended for a node, its receiver transfers energy from the transmission line to the detector. On the other hand, when the message is intended for another node, the node is set to cause minimum loss for the through signal. For this reason, a setup step is performed immediately before payload data transmission to "wake up" the intended receiver, while other receivers remain in the off (and high isolation) mode. This setup is done in a pipelined fashion and the timing is shown in Figure 6. The circuitry to achieve the modes is discussed in Section 3.4.

Turn-around time and bundling: After the transmission of the payload, the bus will be idle for a period of time to allow the signal to "drain" from the links. Even in the short distance of on-chip transmission lines, the wave's propagation delay is not negligible. The amount of time needed to wait before another node can start to use the bus to transmit depends on the distance between the current transmitting node and the next scheduled to transmit. In most cases, a full cycle of turn-around time is enough. In the extreme case, a two-cycle turn-around delay is needed.

Note that in the special case of the same node transmitting another packet there is no need for such a turn-around period. Thus for better utilization of the bus bandwidth, we use a policy that allows *bundling*: sending multiple packets for each bus arbitration. We quantify the impact of bundling in Section 4.5.

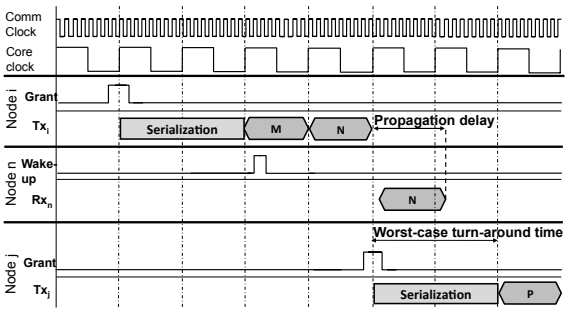


Figure 6: Illustration of bus operation timing. Three nodes (i, n, and j) are shown. Tx and Rx indicate transmitting and receiving ports respectively. Note that pulses are sent using a much higher clock rate and thus each logical communication cycle can transmit multiple bits per TLL.

To summarize the timing of the bus's operation, Figure 6 shows an example of a few packets transmitted over the bus. In this figure, node i sent two packets, one each to node m and n. In the background, the arbiter passes on the grant to node j after accounting for the total bus occupancy of node i, which includes the time for the draining of the signal (2 cycles in this case).

Partitioning the bus: A simple way to get high throughput out of the bus structure is to use a wide bus that minimizes serialization latency. For example, a 32-byte cache line payload can be sent in one processor cycle over a bus with 32 data links operating at a data rate 8 times the computing clock speed. Clearly, a wide bus is wasteful for smaller payloads such as requests. In a shared-memory architecture, meta packets are common (about 60% in our suite of applications). Having another, smaller bus for meta packets is a clear option. In fact, with relatively small costs, we can have multiple buses for meta packets. They can be used to increase throughput, or to support different types of requests such as in Alpha GS320 [21] (which prevents fetch deadlocks and eliminates the need to use NACK in their protocol). For simplicity, in this paper, we assume a single bus for meta packets and another one for data packets.

3.4 Interface Circuit Design

Other than the transmission lines, a TLL includes a few circuit elements as shown in Figure 7: a transmitter, a receiver, a serializer (SER), a deserializer (DES), and a phase

and data recovery circuit (PDR). Each node is connected to other nodes through the shared on-chip transmission lines.

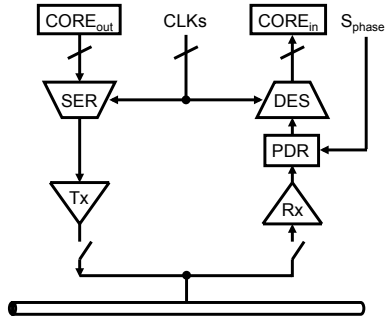


Figure 7: Circuit elements of a TLL.

Transceiver and SERDES: To demonstrate the feasibility and performance of the proposed interconnect, we design and simulate a transceiver circuit at each node using a generic 32-nm CMOS technology [3]. Thanks to the low loss and large bandwidth of the transmission line, no special analog signal processing, such as pre-emphasis or equalization, is required. Therefore, the transmitter (Tx) and receiver (Rx) are both implemented in standard CMOS technology without any special RF devices such as inductors. At $26.4Gb/s$, the communication speed is 8 times a typical core clock speed. Therefore three stages of standard 2:1 MUX/DEMUX are used for the serializer (SER) and deserializer (DES), which are implemented as high-speed digital circuits.

Phase and data recovery: To compensate for the transmission delay, phase recovery is required to achieve optimal data detection. The same reference clock is used for timing in both the transmitter and receiver. Therefore, instead of the more complex conventional clock and data recovery (CDR) circuit, where both frequency and phase information are recovered, a simpler phase and data recovery (PDR) circuit is used [12]. It includes a phase tuning block and a latch block. To recover the phase, a synchronization between the received data and the local clock is needed. The synchronization can be carried out at the start-up of the system, and it is a one-time calibration.

Isolation switches: A switch is necessary to isolate the receiver from the transmission line medium if the data is not meant for that particular node. A common design is to use a switch based on series-shunt topology [25]. However, our tests show that a simple pass-gate (at 32nm) is also a feasible option, allowing sufficiently high isolation and low insertion loss.

4. EXPERIMENTAL ANALYSIS

4.1 Experimental Setup

Sonnet [1] and Advanced Design System (ADS, from Agilent Technologies) were used to perform circuit and physical simulations of the tranmission line links. Sonnet is a first-principle EM simulator which we used to characterize transmission lines at the physical level, taking material and dimensions into account. It provides S-parameters for the transmission lines that feed into ADS simulations. ADS is a widely used electronic design automation software for RF, microwave, and high-speed digital applications. Using schematic designs, ADS has time-domain and frequency-domain analysis to evaluate such digital circuits using both global components and technology specific models, like the ones used for transistors. In this case, a 32-nm predictive technology model (PTM) is used for the transistor modeling [3].

Processor core	
Fetch/Decode/Commit	8 / 5 / 5
ROB	128
Issue Q/Reg. (int,fp)	(32, 32) / (112, 12)
LSQ(LQ,SQ)	64 (32,32) 2 search ports
Branch predictor	Bimodal + Gshare
- Gshare	8K entries, 13 bit history
- Bimodal/Meta/BTB	4K/8K/4K (4-way) entries
Br. mispred. penalty	at least 7 cycles
Process spec.	Feature size: 32nm, Freq: 3.3 GHz, V_{dd}: 1 V
Memory hierarchy	
L1 D cache (private)	16KB, 2-way, 32B line, 2 cycles, 2 ports
L1 I cache (private)	32KB, 2-way, 64B line, 2 cycle
L2 cache (per-core, shared)	128KB slice/node, 8-way, 64B line, 15 cycles, 2 ports
Intra-node fabric delay	2-core node: 1-cycle, 4-core node: 3-cycle
Memory latency	250 cycles
Network packets	Flit size: 72-bits data packet: 4 flits, meta packet: 1 flit
Mesh interconnect	4 VCs; 3-cycle router; buffer: 5x12 flits wire delay: 2 cycles per hop [36]
Transmission line link (each node)	
Bit Rate	26.4 Gb/s, 8 bits per CPU cycle
Transmission latency	2 cycles (worst-case)
Data link	36 links for data, 9 for meta
Outgoing queue	12 packets
Overhead	2 cycles each for (de)serialization, 30ps propagation delay per hop, 1 cycle for token request, 1 cycle for token grant/wake-up

Table 1: System configuration.

Architectural simulations of the proposed design were performed using an extensively modified version of SimpleScalar [11]. PopNet [2] is used to model the packet-switched network, while extra support was added to model the TLL bus. The details of the setup for both the 16-core and 64-core systems are listed in Table 1.

The cache coherence substrate for the architectural simulations is a directory-based MESI protocol with transients faithfully modeled both at the L1 and at the directory controller. The two state machines combined handle a total of 13 transient states and 57 legal transistions (excluding deferred handling) [52].

We use a set of diverse multithreaded applications to test the designs. These applications are compiled using a cross-compiler to generate Alpha binaries. The limitation of the cross-compiler prevents us from running certain applications. Table 2 lists the applications used. Abbreviations are used in the data figures, and the corresponding abbreviation is in parentheses next to the application names. Inputs for each application are listed along with a brief description of the application. Each application is fast-forwarded past the initialization. An offline profile is used to determine data page mapping.

Splash-2 [51]	
barnes (ba)	n-body simulation (16K particles)
cholesky (ch)	sparse matrix factorization (tk15.O)
fft (ff)	complex 1-D fft computation (64K points)
fmm (fm)	fast n-body simulation (16K particles)
lu (lu)	matrix factorization (512x512 matrix, 16x16 blocks)
ocean (oc)	simulation of ocean currents (256x256 matrix)
radiosity (rs)	graphics (large room)
radix (rx)	integer sort algorithm (1M integers)
raytrace (ry)	3-D rendering (car.env)
water-sp (ws)	molecular dynamics (512 molecules)
Parsec [10]	
blackscholes (bl)	financial analysis/calculation (16K options)
fluidanimate (fl)	animation (5 frames, 35K)
Other Benchmarks [17, 20]	
em3d (em)	electro-magnetic forces (1280 edges)
ilink (il)	genetic analysis (40 alleles)
jacobi (ja)	differential equation solver (512x512 matrix, 10 iterations)
mp3d (mp)	n-body simulation (40K molecules)
shallow (sh)	shallow water dynamics (512x512 matrix, 20 phases)
tsp (ts)	traveling salesman problem (18 city map)

Table 2: Benchmarks used.

4.2 Transmission Line Links

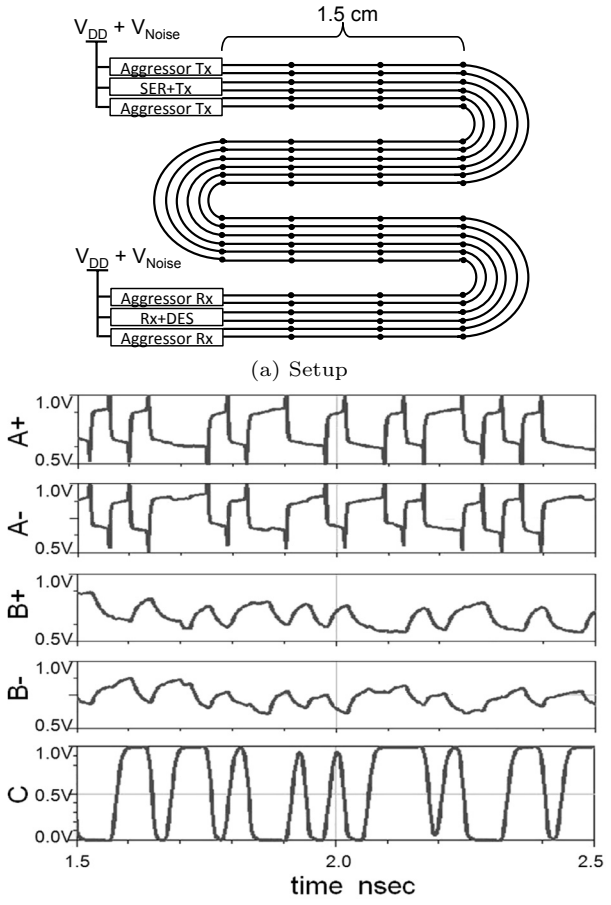
To determine the dimension of the transmission lines for good noise tolerance, we use S-parameter simulations sweeping through a frequency spectrum. After testing a few options, we choose a total pitch of $45\mu m$ and a line width of $10\mu m$. With this configuration, the crosstalk is about -30dB at 26GHz. The transmission lines are of a serpentine shape and measure about 7.5cm in total length. To measure the data rate of the transmission line system under realistic environment, we set up a simulated transmission system shown in Figure 8.

In this setup, the signal-line pair being tested is surrounded by other lines as crosstalk noise sources. A voltage noise source is also introduced to the differential amplifier's supply voltage. Figure 8 also shows the waveform of various points in the system. Simulation results show that the system can carry data reliably at $26.4GHz$, 8 times the computing clocking speed. At this speed, the characteristics of various circuit components are listed in Table 3. Other transceiver design points are also included for comparison. The total area of active circuitry associated with each pair of transmission lines at each node is about $1{,}200\mu m^2$. In a 16-node chip-multiprocessor, the total combined area for active circuitry is less than $1mm^2$, or about 0.2% of total chip area.

4.3 Traffic and Performance Analysis

We focus on multithreaded applications where there is a fundamental demand for horizontal communication. The applications can serve as stress-test workloads for our system which is more bandwidth constrained. The L1 miss rate of these applications ranges up to 61 misses per thousand instructions (MPKI).

Traffic impact of page placement: A significant body of research exists to reduce unnecessary remote accesses by trying to map data close to the threads that frequently access the data. The solutions range from simple heuristics to map pages (*e.g.*, first-touch) to sophisticated algorithms



(a) Setup

(b) Waveforms

Figure 8: Measurement setup and waveforms of transmission line circuit simulations. A, B, and C (as labeled in the design diagram shown in Figure 3) correspond to transmitter output, receiver amplifier input, and final latch output, respectively.

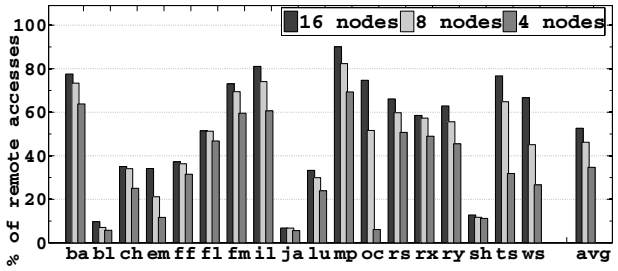
that migrate data on the fly. Such optimizations not only improve performance on their own by reducing average latencies, but also serve to reduce horizontal traffic. In this paper, we use a simple model as a proxy of a "middle-of-the-road" solution to localize data. Specifically, the last-level cache is shared and page interleaved. Off-line profiling assigns pages the color that matches the color of the node where the pages are accessed most frequently.

Figure 9-(a) shows that simple techniques can already cut down on unnecessary horizontal traffic. Without data mapping optimizations, using round-robin data distribution in an n-node system, each L1 miss has a 1 in n chance of being served locally. Hence, we would expect remote traffic to be roughly 94%, 88%, and 75% respectively for 16, 8, and 4 node systems. With even a simple profiling technique, the percentage of remote accesses drops to 53%, 46%, and 35%, respectively.

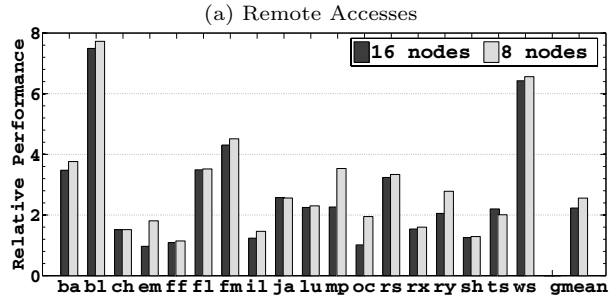
The performance impact of such data mapping on a canonical mesh interconnect is shown in Figure 9-(b). Note that the 16-node organization has 1 core linked to its own L2 slice. The 8-node organization clusters 2 cores into a

Component	Data Rate (Gb/s per line)	Transmitter Side Power (mW)	Latency (ps)	Area (μm^2)	Receiver Side Power (mW)	Latency (ps)	Area (μm^2)	Energy/bit (pJ)
Digital	10	5	30	150	1.5	30	50	0.65-10.4
Analog Receiver	17	20	30	250	8.0	35	60	1.65
Differential	26.4	3.1	22	200	6.4	45	550	0.36
SERDES (8:1)	-	1.24	570	220	1.65	460	165	0.11
PDR	-	-	-	-	0.4	150	60	0.02

Table 3: Transmission line circuit simulation results for three different design points. The differential signaling option is used for system-level simulations.



(a) Remote Accesses



(b) Cluster Performance

Figure 9: (a) Percentage of L2 accesses that are remote. The 3 configurations are 1, 2, and 4 cores per node. **(b)** Speedup due to profiling and clustering. The bar on the left is for 1 core per node, the right bar is for 2 cores per node. The baseline in this case is a 16-core mesh with round-robin data distribution.

single node. The result is a longer latency for using the intra-node fabric to access the cache slices local to the node, but a decrease in the number of remote accesses that use the backbone interconnect. The decrease in horizontal traffic and increased locality result in a speedup of more than 2x over a baseline with round-robin page allocation. Clearly, better data placement is an important optimization in its own right, and the sophistication and effect of the technique will only increase over time. The important side effect of traffic reduction alleviates the problem of the simpler shared-medium relay-free interconnect, such as our design.

Performance comparison: While our TLL bus has a more limited aggregate bandwidth, it offers a better latency in general and in particular for packets between far apart nodes. Figure 10 compares the execution speed of this interconnect (with a bundling factor of 3) with a mesh. In this experiment, the chip-multiprocessor has 16 cores and is organized into 16 or 8 nodes. At this scale, the limit in bandwidth is seldom a problem for any application and, in general, more than compensated for by the superior latency. Even the more bandwidth demanding applications, such as *em3d*, *mp3d*, and *ocean*, perform comparably to mesh, especially in an 8-node configurations. On average, applications run faster on the TLL bus than on the mesh by 1.15x in the 16-node and 1.17x in the 8-node configurations respectively.

We have also modeled an idealized interconnect system and verified that the TLL bus performs close to this upper-bound (more later). For instance, the 8-node system can achieve 91% performance of the ideal system.

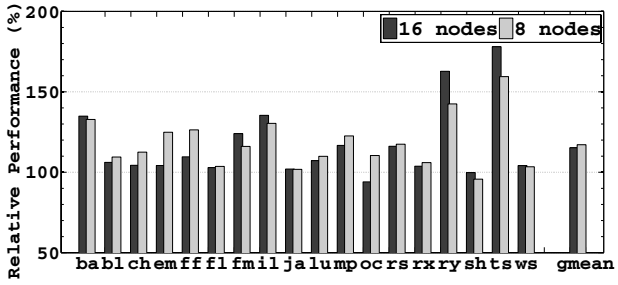


Figure 10: Speedup of TLL bus system over the respective (16- or 8-node) mesh-based system. The left bar in each group represent 16 node configuration and the right bar, 8 node. Note the y-axis does not start at 0.

As we saw in Figure 9-b, even though the intra-node fabric becomes slower as the node size increases, the benefit of having a smaller network in general outweighs the cost of slower intra-node accesses. In a mesh-based system, clustering helps improve performance by 4%. Just as with the case of better data placement, these optimizations reduce the demand on the backbone interconnect and has a slightly more significant benefit (6%) in the TLL bus system.

To summarize, even though bus architectures face bandwidth scalability challenges, in modest-scale chip-multiprocessors and when natural steps are taken to improve performance, the disadvantages of TLL bus are much mitigated and the benefit becomes more pronounced.

4.4 Power Savings

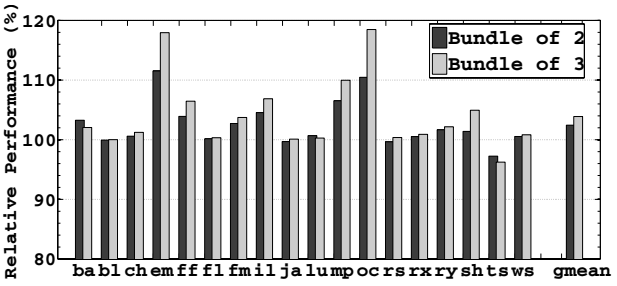
One of the main disadvantages of canonical mesh networks is the high power and energy consumption [19, 29, 36, 45]. On average, the network power accounts for around 20% of the total system's power. In contrast, the TLL bus uses no relay or energy-intensive routing. The power consumption of TLL bus is low in both absolute and relative terms. An entire link consumes 12.7mW while active (Table 3 shows power of individual components). Even when all lines are working all the time, the total power is around 600mW. When idling, the power consumption is even lower. We

estimate leakage in the communication circuit to be around $10\mu W$ per node [3], essentially negligible.

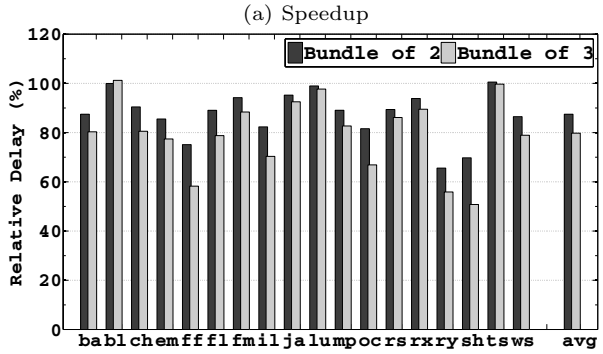
Comparing the energy consumed by the TLL bus to the power statistics from the network power model, Orion [49], we see a reduction in network energy of about 26x. With this reduction, the energy spent in the interconnect is less than 1% of the total energy consumption.

4.5 The Impact of Bundling

As discussed in Section 3.3, the turn-around time also wastes bus bandwidth and can be mitigated with bundling. So far, we have used a bundling factor of 3, *i.e.*, each node can send up to 3 packets before yielding the bus. Figure 11 shows the impact of varying the bundling factor from 1 (no bundling) to 3. As we can see, the performance generally increases when the bundling factor increases. Without bundling, much bandwidth is wasted due to turn-around, and so there is a noticeable performance increase with a bundling of 2. However, too much bundling can be detrimental to performance as well (*e.g.*, in the case of *tsp*). Figure 11-(b) shows the average overall packet latency for a bundling of 2 and 3 compared to no bundling. On average, bundling of 2 and 3 saves 13% and 20% respectively of the latency and improves performance by 2.0% and 3.4% respectively.



(a) Speedup



(b) Overall latency

Figure 11: (a) Speedup of the 16-node system with bundling of 2 and 3, over the system without bundling. Note the y-axis does not start at 0. **(b)** Overall packet latency relative to a non-bundled system. The left and right bar correspond to a bundling of 2 and 3 respectively.

4.6 Scaling Up

While many-core chips will fill a certain market niche, a significant fraction of general-purpose chip-multiprocessors may have only a relatively modest number of cores. The proposed design works well in such an environment. As the number of cores increases beyond a threshold, the viability of our current design will decrease. We conduct a limited scalability test with a 64-core system organized into 2- or 4-core nodes (32 nodes, 2 cores each; and 16 nodes, 4 cores each), using the exact same bus design as before. Figure 12 summarizes the performance result compared to the (scaled-up) mesh-based design with the same clustering.

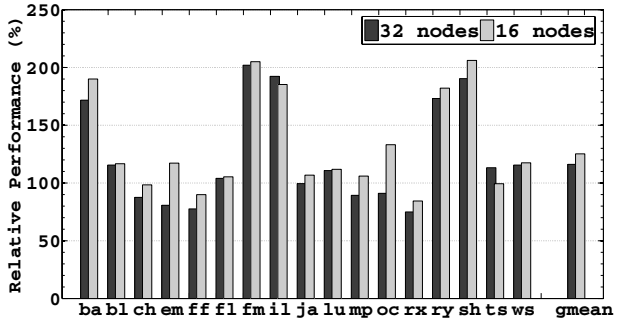


Figure 12: Relative performance of a 64-core system. For the TLL bus configurations, a bundle of 3 is used.

As the system grows in size, the probability of the bus becoming a bottleneck increases. In a few cases (*e.g.*, *fft* and *radix*), the performance of the TLL bus is significantly worse than the conventional mesh interconnect (Figure 12). On the other hand, when the bandwidth is not a bottleneck resource, the latency advantage over mesh becomes even more pronounced. As a result, the performance gap between the bus-based and mesh-based systems widens for many applications (*e.g.*, *fmm* and *shallow*). On average, the TLL bus performs 16% and 25% better than mesh for a 32- and 16-node system, respectively. Clearly, simply having better aggregate bandwidth scalability is not enough. A packet-switched interconnect (including segmented bus) segments wires to allow simultaneous traffic, improving overall bandwidth at the expense of latency. The result can also be a serious performance issue for chip-multiprocessors.

In other words, a bus architecture should not be written off as a possible solution for on-chip interconnect. After all, no design is truly scalable in all respects. The sacrifice in latency in some packet-switched interconnects can be an even more serious performance problem, not to mention the significantly higher energy cost.

To better understand the limitation of bus-based system, we also compare it to an idealized interconnect system using conventional digital wires. In this system, no bandwidth limitation or contention is modeled for the interconnect. A packet's delay is calculated as 0.03mm/ps based on the latency-optimized wires in [36].

Figure 13 shows the performance of the TLL bus in 32-node and 16-node configurations (both have 64 cores) normalized to that of the ideal interconnect. As we can see, while 7 out of 18 benchmarks perform within 10% of the idealized case, the limited bandwidth shows significant limitation in a number of applications where performance can be improved several folds. Nevertheless, the bus system

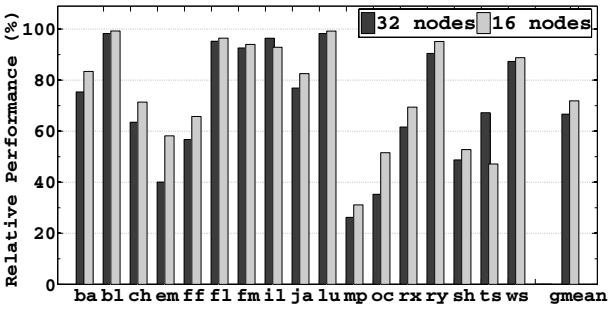


Figure 13: Performance of TLL bus relative to idealized contention-free, low-latency interconnect.

achieves 67% and 72% of the idealized performance, for 32- and 16-nodes respectively, showing a somewhat graceful degradation beyond its intended usage range. Recall, in a 16-core 8-node system, the bus can achieve 91% of the ideal's performance.

5. CONCLUSIONS

General-purpose microprocessors currently have a few cores and are expected to gain more cores as technology continues to scale. Packet-switched interconnect is accepted by many as the default solution for on-chip communication for future chips. While the superior scalability certainly carries significant advantages, there are, nonetheless, non-trivial issues such as the area cost of the router, the latency impact, and power overhead of repeated packet relays. While continued research will undoubtedly mitigate some of the issues, we should also investigate alternative solutions.

A number of factors suggest that main-stream chip-multiprocessors are unlikely to require an extreme amount of bandwidth for on-chip backbone communication. First, while aggressively scaled many-core designs will find their market, we believe main-stream chips will experience a diminished return of benefit after incorporating a modest number of cores. Second, on-chip node organization and a variety of optimizations to place data closer to the computation suggest that only a small number of nodes will be connected by packet-based backbone interconnect and the traffic on this fabric can be rather limited. Similar observations have prompted proposals of reduced-strength packet switching networks such as local buses connected through hubs that can intelligently perform traffic filtering.

In this paper, we echo the principle behind this group of solutions, but make a case for a different type of design. In our design, advanced wide-band communication circuit is used to treat on-chip metal wires explicitly as transmission lines. The resulting communication channels offer a much higher throughput and propagating speed. With this underlying capability, a truly packet-switching-free interconnect is both easy to build and quite competent to support the traffic demand for modestly sized chip-multiprocessors. Experimental analyses have shown that in a medium-scale 16-core system, this design achieves 91% of that in an *idealized* wire-based interconnect. The performance degrades rather gracefully, still achieving 72% performance of the ideal configuration in a 64-core system.

Contrast with a canonical mesh interconnect reveals that despite having a more limited overall throughput, the transmission line link (TLL) bus often shows advantages when latency is more critical and achieves better performance on average (1.17x in an 8-node 16-core system). This average performance advantage is slightly improved (1.16-1.25x) in a 64-core system, albeit with considerable application-to-application variations. This indicates that simply relying on packet switching to scale overall bandwidth is insufficient. A much more sophisticated architecture to keep latency low is crucial for a packet-switched interconnect.

Another important benefit of avoiding packet switching and relaying is the inherent energy efficiency of the communication system. The energy reduction in the backbone network is more than an order of magnitude compared to a mesh. This energy advantage of the TLL bus is important in itself and also provides capital for future optimizations that compensate for the bandwidth limitation.

REFERENCES

[1] `http://www.sonnetsoftware.com/`.

[2] PoPNet. `http://www.princeton.edu/~peh/orion.html`.

[3] Predictive Technology Modeling. `http://ptm.asu.edu/`.

[4] T. Agerwala, J. Martin, J. Mirza, D. Sadler, and D. Dias. SP2 System Architecture. *IBM Systems Journal*, 34(2):152–184, 1995.

[5] T. Ainsworth and T. Pinkston. Characterizing the Cell EIB On-Chip Network. *IEEE Micro*, 27(5):6–14, 2007.

[6] M. Awashti, K. Sudan, R. Balasubramonian, and J. Carter. Dynamic Hardware-Assisted Software-Controlled Page Placement to Manage Capacity Allocation and Sharing within Large Caches. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, pages 250–261, February 2009.

[7] J. Balfour and W. J. Dally. Design Tradeoffs for Tiled CMP On-Chip Networks. In *Proc. Int'l Conf. on Supercomputing*, pages 187–198, June 2006.

[8] B. Beckmann and D. Wood. TLC: Transmission Line Caches. In *Proc. Int'l Symp. on Microarch.*, pages 43–54, December 2003.

[9] B. Beckmann and D. Wood. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proc. Int'l Symp. on Microarch.*, pages 319–330, November 2004.

[10] C. Bienia, S. Kumar, J. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proc. Int'l Conf. on Parallel Arch. and Compilation Techniques*, September 2008.

[11] D. Burger and T. Austin. The SimpleScalar Tool Set, Version 2.0. Technical report 1342, Computer Sciences Department, University of Wisconsin-Madison, June 1997.

[12] B. Casper, J. Jaussi, F. O'Mahony, M. Mansuri, K. Canagasaby, J. Kennedy, E. Yeung, and R. Mooney. A 20Gb/s Forwarded Clock Transceiver in 90nm CMOS. In *IEEE Int'l Solid-State Circuits Conf.*, pages 263–272, 2006.

[13] M. Chang, J. Cong, A. Kaplan, C. Liu, M. Naik, J. Premkumar, G. Reinman, E. Socher, and S. Tam. Power Reduction of CMP Communication Networks via RF-Interconnects. In *Proc. Int'l Symp. on Microarch.*, pages 376–387, November 2008.

[14] M. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and R. Tam. CMP Network-on-Chip Overlaid With Multi-Band RF-Interconnect. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, pages 191–202, February 2008.

[15] M. Chang, E. Socher, S. Tam, J. Cong, and G. Reinman. RF Interconnects for Communications On-chip. In *Proc. Int'l Symp. on Physcial Design*, pages 78–83, 2008.

[16] S. Cho and L. Jin. Managing Distributed, Shared L2 Caches through OS-Level Page Allocation. In *Proc. Int'l Symp. on Microarch.*, pages 455–468, December 2006.

[17] D. Culler, A. Dusseau, S. Goldstein, A. Krishnamurthy, S. Lumetta, T. Eicken, and K. Yelick. Parallel Programming in Split-C. In *Proc. Supercomputing*, November 1993.

[18] W. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proc. Design Automation Conf.*, pages 684–689, June 2001.

[19] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das. Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, February 2009.

[20] S. Dwarkadas, A. Schaffer, R. Cottingham, A. Cox, P. Keleher, and W. Zwaenepoel. Parallelization of General Linkage Analysis Problems. *Human Heredity*, 44:127–141, 1994.

[21] K. Gharachorloo, M. Sharma, S. Steely, and S. Van Doren. Architecture and design of AlphaServer GS320. In *Proc. Int'l Conf. on Arch. Support for Prog. Lang. and Operating Systems*, pages 13–24, November 2000.

[22] G. Hendry, J. Chan, S. Kamil, L. Olifer, J. Shalf, L. Carloni, and K. Bergman. Silicon Nanophotonic Network-On-Chip Using TDM Arbitration. In *Hot Interconnect*, pages 88–95, August 2010.

[23] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. A 5-GHz Mesh Interconnect for a Teraflops Processor. *IEEE Micro*, 27(5):51–61, 2007.

[24] R. Hou, L. Zhang, M. Huang, K. Wang, H. Franke, Y. Ge, and X. Chang. Efficient Data Streaming with On-chip Accelerators: Opportunities and Challenges. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, February 2011.

[25] Y. Jin and C. Nguyen. Ultra-Compact High-Linearity High-Power Fully Integrated DC-20-GHz 0.18-um CMOS T/R Switch. *IEEE Transactions on Microwave Theory and Techniques*, 55(1):30–36, Jan. 2007.

[26] C. Johnson, D. Allen, J. Brown, S. Vanderwiel, R. Hoover, H. Achilles, C. Cher, G. May, H. Franke, J. Xenedis, and C. Basso. A Wire-Speed Power Processor: 2.3GHz 45nm SOI with 16 Cores and 64 Threads. In *IEEE Int'l Solid-State Circuits Conf.*, pages 104–106, 2010.

[27] R. Kessler and M. Hill. Page Placement Algorithms for Large Real-Indexed Caches. *ACM Transactions on Computer Systems*, 10(4):338–359, 1992.

[28] C. Kim, D. Burger, and S. Keckler. Non-Uniform Cache Architectures for Wire-Delay Dominated On-Chip Caches. *IEEE Micro*, 23(6):99–107, 2003.

[29] J. Kim. Low-Cost Router Microarchitecture for On-Chip Networks. In *Proc. Int'l Symp. on Microarch.*, pages 255–266, December 2009.

[30] N. Kirman, M. Kirman, R. Dokania, J. Martinez, A. Apsel, M. Watkins, and D. Albonesi. Leveraging Optical Technology in Future Bus-based Chip Multiprocessors. In *Proc. Int'l Symp. on Microarch.*, pages 492–503, December 2006.

[31] N. Kirman and J. Martinez. A Power-Efficient All-Optical On-Chip Interconnect Using Wavelength-Based Oblivious Routing. In *Proc. Int'l Conf. on Arch. Support for Prog. Lang. and Operating Systems*, pages 15–28, March 2010.

[32] R. Kumar, V. Zyuban, and D. Tullsen. Interconnections in Multi-core Architectures: Understanding Mechanisms, Overheads, and Scaling. In *Proc. Int'l Symp. on Comp. Arch.*, pages 408–419, June 2005.

[33] J. Laudon and D. Lenoski. The SGI Origin: A ccnuma Highly Scalable Server. In *Proc. Int'l Symp. on Comp. Arch.*, pages 241–251, June 1997.

[34] A. Momtaz and M.M. Green. An 80 mW 40 Gb/s 7-Tap T/2-Spaced Feed-Forward Equalizer in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 45(3):629–639, Mar. 2010.

[35] S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D Webb. The Alpha 21364 Network Architecture. *IEEE Micro*, 22(1):26–35, January/February 2002.

[36] N. Muralimanohar and R. Balasubramonian. Interconnect Design Considerations for Large NUCA Caches. In *Proc. Int'l Symp. on Comp. Arch.*, pages 369–380, June 2007.

[37] N. Jouppi N. Muralimanohar, R. Balasubramonian. Optimizing NUCA Organizations andWiring Alternatives for Large Caches With CACTI 6.0. In *Proc. Int'l Symp. on Microarch.*, pages 3–14, December 2007.

[38] B. Nayfeh, K. Olukotun, and J. Singh. The Impact of Shared-Cache Clustering in Small-Scale Shared-Memory Multiprocessors. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, pages 74–84, February 1996.

[39] J. Oh, M. Prvulovic, and A. Zajic. TLSync: Support for Multiple Fast Barriers Using On-Chip Transmission Lines. In *Proc. Int'l Symp. on Comp. Arch.*, June 2011.

[40] L. Roberts. ALOHA Packet System With and Without Slots and Capture. *ACM SIGCOMM Computer Communication Review*, 5(2):28–42, April 1975.

[41] A. Roy and M. Chowdhury. RS/Wireless Interconnects in Future On-Chip and Board-Level Clock Distribution Network. In *IEEE Int'l Conf. on Electro/Information Technology*, pages 542–545, May 2007.

[42] E. Sackinger and W. Fischer. A 3-GHz 32-dB CMOS Limiting Amplifier for SONET OC-48 Receivers. *IEEE Journal of Solid-State Circuits*, 35(12):1884–188, December 2000.

[43] D. Sanchez, G. Michelgeannakis, and C. Kozyrakis. An Analysis of On-Chip Interconnection Networks for Large-Scale Chip Multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 7(1), 2010.

[44] C. Seitz. Let's Route Packets Instead of Wires. In *Proceedings of the Sixth MIT Conference on Advanced Research in VLSI*, pages 133–138, 1990.

[45] A. Udipi, N. Muralimanohar, and R. Balasubramonian. Towards Scalable, Energy-Efficient, Bus-Based On-chip Networks. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, pages 1–12, January 2010.

[46] S. Vangal et al. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In *IEEE Int'l Solid-State Circuits Conf.*, pages 98–100, February 2007.

[47] D. Vantrease et al. Corona: System Implications of Emerging Nanophotonic Technology. In *Proc. Int'l Symp. on Comp. Arch.*, June 2008.

[48] H. Wang and J. Lee. A 21-Gb/s 87-mW Transceiver With FFE/DFE/Analog Equalizer in 65-nm CMOS Technology. *IEEE Journal of Solid-State Circuits*, 45(4):909–920, Apr. 2010.

[49] H. Wang, X. Zhu, L. S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proc. Int'l Symp. on Microarch.*, pages 294–305, November 2002.

[50] D. Wentzlaff et al. On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, 27(5):15–31, 2007.

[51] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proc. Int'l Symp. on Comp. Arch.*, pages 24–36, June 1995.

[52] J. Xue et al. An Intra-Chip Free-Space Optical Interconnect: Extended Technical Report. Technical report, Dept. Electrical & Computer Engineering, Univ. of Rochester, April 2010. `http://www.ece.rocehster.edu/~mihuang/`.

[53] J. Xue, A. Garg, B. Ciftcioglu, J. Hu, S. Wang, I. Savidis, M. Jain, R. Berman, P. Liu, M. Huang, H. Wu, E. Friedman, G. Wicks, and D. Moore. An Intra-Chip Free-Space Optical Interconnect. In *Proc. Int'l Symp. on Comp. Arch.*, pages 94–105, June 2010.

[54] M. Zhang and K. Asanovic. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In *Proc. Int'l Symp. on Comp. Arch.*, pages 336–345, June 2005.