Enhancing Effective Throughput for Transmission Line-Based Bus [*]

Aaron Carpenter[†], Jianyun Hu, Ovunc Kocabas, Michael Huang, and Hui Wu
Dept. of Electrical and Computer Engineering
University of Rochester
{jianyun.hu, ovunc.kocabas, michael.huang, hui.wu}@rochester.edu
[†]Binghamton University
carpente@binghamton.edu

Abstract

Main-stream general-purpose microprocessors require a collection of high-performance interconnects to supply the necessary data movement. The trend of continued increase in core count has prompted designs of packet-switched network as a scalable solution for future-generation chips. However, the cost of scalability can be significant and especially hard to justify for smaller-scale chips. In contrast, a circuit-switched bus using transmission lines and corresponding circuits offers lower latencies and much lower energy costs for smaller-scale chips, making it a better choice than a full-blown network-on-chip (NoC) architecture. However, shared-medium designs are perceived as only a niche solution for small- to medium-scale chips.

In this paper, we show that there are many low-cost mechanisms to enhance the effective throughput of a bus architecture. When a handful of highly cost-effective techniques are applied, the performance advantage of even the most idealistically configured NoCs becomes vanishingly small. We find transmission line-based buses to be a more compelling interconnect even for large-scale chip-multiprocessors, and thus bring into doubt the centrality of packet switching in future on-chip interconnect.

1 Introduction

Main-stream general-purpose microprocessors already integrate a handful of high-performance cores and thus require a collection of high-performance interconnects to supply the necessary data movement. The trend of continued increase in core count has prompted the design of packet-switched networks (sometimes called network-on-chip or NoC) as a solution to provide scalable throughput[1] for future-generation chips. However, this scalability comes with costs. First of all, a packet-switched network is an energy-intensive solution as routing, switching, and relaying packets all incur non-trivial energy overheads that add up quickly. Secondly, packet switching does not address the fundamental issue of wires, but instead adds to the overall communication latency while taking up significant chip area. These overheads can be high and especially hard to justify for smaller-scale chips.

An alternative to the architecture-level packet switching for throughput boosting is to employ a more sophisticated signaling chain at the circuit level. Faster device speed enables practical design of energy-efficient, high-performance (analog) communication circuits that allows efficient use of an underlying communication channel. Properly engineered on-chip transmission lines can provide excellent communication channels, allowing speed-of-light signal propagation [15] without the technological hurdles of optical interconnects. Together, these elements create a different type of data links than conventional repeated digital wires. With these transmission line links, even a simple global bus architecture can deliver a significant amount of throughput, sufficient for a moderately sized system (*e.g.*, 16 cores) [11]. Without any packet switching or relay, such a bus keeps the latency and energy benefits of the links.

Despite all the advantages of transmission line based buses, the simple bus structure does impose a throughput limit and does not readily support a large-scale chip-multiprocessor (CMP). If we can increase bus throughput to support a larger number of (*e.g.*, 64) cores, the benefit is clear: for chips of that scale, no packet switching is needed; the communication substrate design will be greatly simplified; the bus design provides unique opportunities to simplify and optimize the shared-memory coherence substrate. Even if future chips grow much beyond that scale, they are likely to be used in virtualized server environments with logical partitions with the size of 32 to 64 cores. Having packet-switching-free partitions still offers tremendous benefits.

In this paper, we explore a number of approaches to improve the overall throughput of a transmission line based bus and analyze the cost of achieving higher throughput. The analysis shows a rather straightforward path to supporting a large number of cores with bus architectures. As such, packet switching is not a central or even necessary element of on-chip interconnect. Instead of only focusing

[*]This work is supported in part by NSF under the grants 0901701, 0829915, and 0747324, and by the NSFC under grant 61028004.

[1]To avoid ambiguity, we use *throughput* to refer to the data rate (measured in bytes/second) of links and networks, and *bandwidth* (measured in Hz) for the 3dB frequency response of the medium.

on further improving NoC, we should also explore packet-switching-free interconnect and designs of low-cost packet switching that serves only as an auxiliary mechanism.

The rest of the paper is organized as follows. Section 2 discusses background and related work. Section 3 briefly summarizes the baseline bus design. Section 4 discusses techniques for increasing effective throughput. Section 5 presents the experimental analysis of these techniques. Section 6 concludes.

2 Background and Related Work

With the integration of multiple cores on a single die, proposals of advanced interconnection have emerged. These proposals range from networks-on-chip (NoC) [5, 19, 24, 35, 38, 43] to optical interconnects [16, 23, 30, 31, 40, 42, 45] or RF interconnects [6, 12–14]. Even with the use of circuit- or device-level support for optics or RF circuitry, many designs still rely on packet-switching at the architecture level [6, 12–14, 16].

Recently, bus designs have started to gain more attention as a supplement or alternative to pure packet-switched networks. Conventional digital buses are being explored as part of the interconnect design [20, 41]. These designs still rely on packet-switching to connect multiple buses either explicitly through routers [20] or implicitly via hubs connecting multiple bus segments [41]. With only buses in the system, it is argued that the coherence substrate can switch to a snoopy protocol that helps reduce transaction hops and thus overall latency. Transmission lines are used with wideband communication circuits to provide a bus design with low latencies and high overall throughput, which in turn allows the bus to be the only fabric in a CMP and a purely circuit-switched fabric [11]. Such a design is intended for a small- to medium-sized CMP and becomes a bottleneck for demanding applications in a large-scale chip. The low propagation delay of transmission lines has also been exploited to speed up access to remote L2 banks [6, 7] or remote nodes in a mesh network [12, 13], or to build a fast barrier mechanism [36]. Finally, using transmission lines for communication is a well-established technique in mixed-signal and analog systems. There is no need to rely on future development for devices and technologies, as in on-chip optical interconnect.

In addition to leveraging transmission lines, packet latency can be reduced via various optimizations in a packet-switched interconnect. New topologies, such as flattened butterfly, use higher radix routers to reduce network diameter and thus the average number of hops [29]. Wiring between routers can also be optimized with customized sizing to trade off among latency, throughput density, and energy [33, 34].

3 Basic Transmission Line-Based Bus

We first establish a baseline design of the bus. Like a typical bus-based multiprocessor system with a narrower address bus and a wider data bus, we assume two separate logical buses. One for control messages, such as requests (which we call the meta bus), and the data bus for cache-line size data responses. A central pipelined arbiter is used. To minimize receiver power, only the intended receiver is woken up by the arbiter. This system allows point-to-point communication between a transmitter and a receiver node.

Circuit design: On the circuit level, we use differential analog transceiver circuitry and on-off keying (OOK) encoding. Our results show that such a setup (in 32nm technologies) can deliver over 25Gbps on a single pair of differential coplanar strip transmission lines. The transceiver circuits are designed and laid out at the transistor level and characterized using detailed circuit and EM simulations (see Section 5.1 for details). Impedance matching is accomplished using shunt resistance to terminate the transmission lines, minimizing reflection. The key specs of our baseline bus are summarized in Table 1. While more detailed analyses of the link design and implementation could be helpful, they are beyond the scope of this paper and have been dealt with in some prior works [26, 27, 32, 39] . We have also taped out a test chip (Figure 1) to further validate the circuit parameters obtained from our circuit simulations. In general, multidrop, global on-chip transmission line links could reliably operate at 10s of Gbps in current and future technologies.

Tile structure	$5mm \times 5mm$ tiles forming a $2cm \times 2cm$ chip
Cache coherence	Directory-based, MESI protocol
Transceiver circuit	Power per bit-line: 17.7mW Latency: 817ps; Encoding: OOK, 26.4Gbps Area per bit-line: $820\mu m^2$ Total area for all transceivers: $< 0.2\%$ of chip area
Transmission lines	Co-planar strips, 7.5mm meandering line $45\mu m$ pitch ($10\mu m$ wide, $10\mu m$ spacing) 56 lines, 440ps max propagation delay (29ps per hop)

Table 1: Key specs of the baseline transmission line bus circuitry. For more details, see [10, 11]

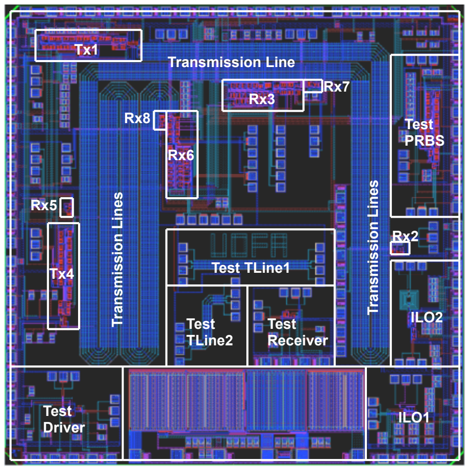


Figure 1: Test chip of transmission line links.

Central arbiter: In theory, any central arbiter has a scalability limit. In practice, we find that the central arbiter is not a concern before we reach the saturation of the bus itself. This is to a large extent due to the simplicity of design – it is essentially just a priority encoder for, say, 16 bits in a 16-node system. Larger, far more complex priority encoders are used in the timing-critical store-forwarding circuit inside the core. Furthermore, when we use techniques such as segmentation (discussed later) to improve the throughput of the bus, the scale of the arbiter actually decreases as each segment is smaller. We have measured a straightforward, unoptimized synthesis of a 16-node arbiter and compared it to the synthesized router used in a packet-switched interconnect [37]. The router's overall delay is 4.3x that of the arbiter (1.65ns vs 0.38ns). The router is also much larger (10x), consumes far more power (20x), and is used more frequently (per flit-hop).

The request and grant signals are transferred over transmission lines similar to those used to build the bus. Such transfers take additional latency (modeled faithfully in this study) that will only be exposed when the bus is lightly loaded.

Coherence protocol: Traditionally, a bus-based system uses a snoopy coherence protocol. However, such an association is not fundamental and is perhaps inappropriate for a transmission line based implementation: First, leveraging analog circuit and transmission lines, a bus can support a rather large number of processor cores. Fanning out snooping requests to a large number of cores incurs significant energy overhead in cache controllers and is undesirable. Second, while a conventional digital bus can support broadcast primitives in a straightforward (but costly) way, broadcast operations are more demanding on analog transmission line designs, especially if the fan-out is large.

Of course, both issues can be addressed. Snooping overhead can be mitigated by incorporating elements similar to a directory-based protocol that filters out nodes known to be unrelated to the current transaction. Special broadcast-capable bus can be demanded from the circuit layer. It is unclear whether these fixes are more cost-effective than avoiding broadcast with a directory-based protocol. In this study, we opt to assume a directory-based protocol in the baseline design.

4 Increasing Effective Bus Throughput

Given a basic design, we can increase the throughput of the bus via a number of simple techniques at the circuit or architecture level, or with a combination of both. It is worth noting that some optimizations are a unique result of the shared-memory chip multiprocessor environment, including its traffic properties, and are not necessarily applicable to bus topology in general. The proposed techniques can be broadly categorized into three groups (a) increasing the underlying supply of raw throughput, (b) improving the utilization efficiency, and (c) co-optimization with protocol layer to reduce traffic demand. Of course, sometimes a particular mechanism defies exact categorization and can fall into more than one group.

4.1 Increasing Raw Link Throughput

Perhaps the first thought that comes to mind about increasing the throughput of a bus is to increase the raw throughput of each individual link. Intuitively, these approaches are more or less brute force approaches that are less efficient than others in achieving the goal. Nevertheless, we analyze some options and evaluate their efficiency later in Section 5.

The potential of link throughput is high, thanks to the well-controlled on-chip environment and the relatively short distances of on-chip transmission lines, the inherent channel bandwidth of the transmission line is quite high. Figure 2 illustrates an experiment to determine the aggregate bandwidth and potential throughput provided by an array of transmission lines.

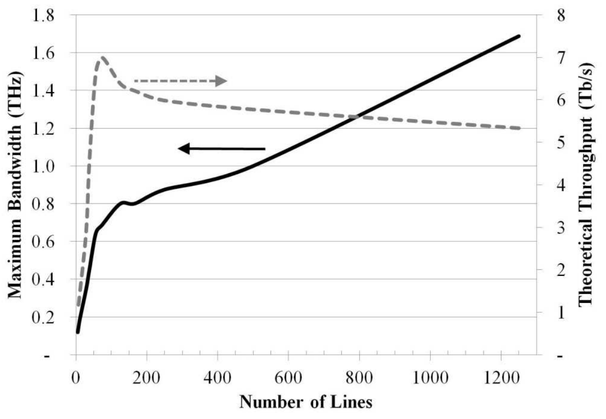


Figure 2: Aggregate baseband bandwidth and theoretical capacity for transmission lines (coplanar strips).

In this experiment, we limit the total pitch of the transmission lines to 2.5mm but vary the width, gap (between the pair of differential lines), and spacing (between two neighboring pairs) of the transmission lines. The length of the lines are set to 7.5cm, assuming a meandering shape going through the centers of sixteen $5mm \times 5mm$ tiles forming a $2cm \times 2cm$ chip. We then use EM and circuit simulators (see Sec. 5.1 for details about the tools) to estimate the 3dB bandwidth of the transmission lines and aggregate the bandwidth for the entire array. We also model noise coupled from neighboring aggressor lines and power supply noise in transceiver circuitry and estimate overall signal-to-noise ratio (SNR). This can give us the theoretical limit on the channel capacity.

Note that this experiment provides an approximate understanding of the potentials and cannot be used as a bound: in practical operating environments, the presence of thermal noise creates the noise floor that is not captured by the analysis of SNR. On the other hand, the bandwidth limit

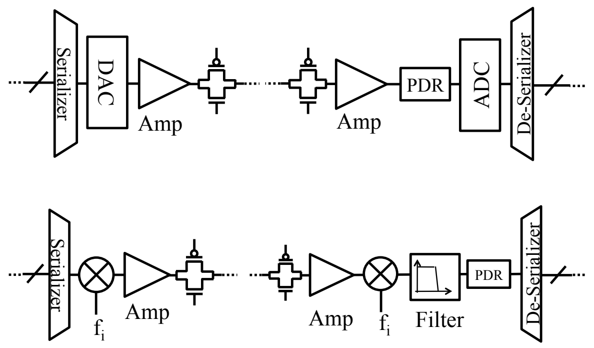


Figure 3: Block diagram of links using 4-PAM and FDM.

can be overcome, to a certain extent, using various circuit techniques. The bandwidth can also be expanded using multi-band designs. Nevertheless, this analysis shows that on-chip transmission lines do possess relatively abundant bandwidth and can support a substantial amount of throughput.

There are many coding strategies to increase the raw throughput. For on-chip communication, however, we are likely limited to simpler variations. We first turn to 4-PAM (Pulse Amplitude Modulation) which doubles the data rate compared to OOK. The additional circuit includes a digital-analog converter (DAC) for the transmitter and an analog-digital converter (ADC) for the receiver (Figure 3). These elements not only increase energy but also add latency on the packet transmission path. In order to minimize the latency impact, we use it only for data packet buses.[2]

Second, we investigate Frequency Division Multiplexing (FDM). FDM allows us to use higher frequency bands on the same physical media. The attenuation in these bands can be high and it increases with frequency. When used as global buses, the higher-frequency bands quickly become too lossy and thus inoperable. A simple calculation can illustrate the problem. Assume we have a 10GHz channel spacing and use 6 such channels [13], Figure 4 shows the frequency response of our transmission lines in the needed spectrum (between dc and 50GHz). At 50GHz, the attenuation is around 9dB. Furthermore, mixers introduce non-trivial *noise figure* (or degradation of SNR), especially for high-frequency operations. Even with bipolar designs, the noise figure can be around 10dB per mixer [25]. The combined effect of two mixers and the transmission line itself can amount to 29dB (800x), not to mention the filter's loss. A rough interpretation is that in the 50GHz channel, the power of the transmitter and the sensitivity of the receiver need to increase a combined 800 times to achieve the same SNR as when using the baseband without mixers, which takes about 30 times more power on each side. Clearly, the higher frequency channels are exceedingly expensive to use

[2]One can even use global intelligence about traffic conditions to bypass 4-PAM when traffic demand is low in order to further minimize latency and energy overhead. This part of the design space is not explored.

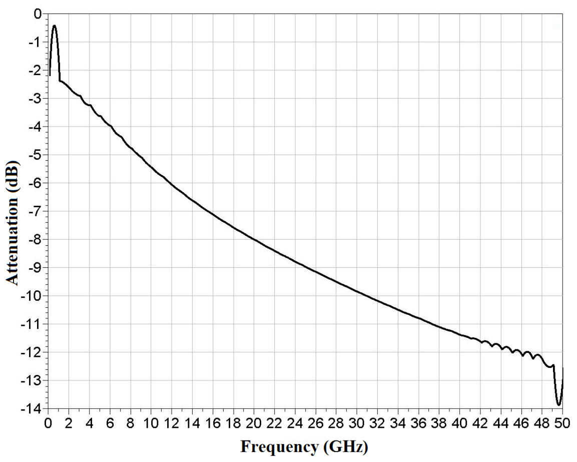


Figure 4: Transmission line frequency response. As the frequency approaches the boundaries of the spectrum, there are noticeable noises due to artifacts of the simulation setup.

in long on-chip transmission lines. They are intended for much shorter communications [13].

For this study, we use only two bands. The circuit support includes mixers for both the transmitter and the receiver side and a filter for the receiver side (Figure 3). Accurately estimating the power costs of the supporting circuitry is challenging. These non-trivial analog components need to be designed, tested, and fine-tuned to work at the required specifications. For this study, we use a simplifying analysis to estimate the *minimum* power cost to support frequency-division multi-band transmission. We use the design similar to [13] but adapted to the baseline system design.

4.2 Increasing the Utilization Efficiency

While the underlying global transmission lines support a very high data rate, using them to shuttle around short packets found in a shared-memory system can cause significant under-utilization. First, the relatively long line means that a packet can take a long time to "drain" from the transmission line (the worst case propagation delay in our bus is 440 ps). A simple arbitration scheme that waits for the bus to drain is one source of under-utilization. Second, packets destined for a near neighbor are a poor match to the global line structure. A number of techniques can address these issues.

Partitioning: A most straightforward option is to partition the same number of underlying links into more, narrower buses. In a narrower bus, longer serialization reduces the waste due to draining. To minimize extra transmission delays due to packet serialization, we can limit serialization to the data bus, and use critical-word-first data marshaling.

An interesting side effect of partitioning the wide data bus into narrower buses is that the finer granularity allows us to better balance the load of the two types of buses. Instead of using 1-flit-wide meta bus and a 4-flit-wide data

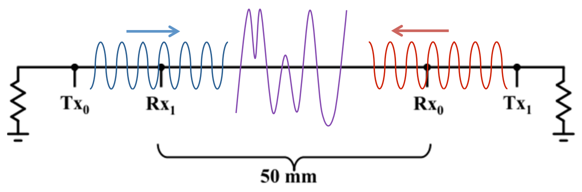


Figure 5: Two independent waves transmitting simultaneously.

bus, we can partition the five 1-flit-wide buses into any combinations of meta buses and data buses. Such partitioning can even be done at runtime, based on individual applications' traffic pattern. In this paper, we use a fixed configuration that achieves the best average performance. It has two meta and three data buses, which best matches the *average* traffic pattern.[3]

Wave-based arbitration: Another mechanism to reduce the impact of draining latency is to allow waves to coexist on the transmission lines. When waves meet, they travel on without impacting each other, only creating a superposition where they meet. In the example shown in Figure 5, two far apart nodes send each other a pulse train. The two trains cross each other over inactive nodes and do not interfere with each other when they reach their respective receiver.

In theory, we can send multiple pulse trains on the links so long as no two trains cross over at an active receiver or transmitter. In practice, we send at most two such trains and use a simple rule to pick a second pair of transmitter and receiver (Tx_1 and Rx_1) that do not interfere with the already-selected first pair (Tx_0 and Rx_0). In this case, the distance between Tx_0 and Tx_1 and between Rx_0 and Rx_1 need to be larger than half the total length of the bus. We tested the design of such an arbiter and found that it does not affect cycle-level performance.

Segmentation: In addition to increasing the temporal efficiency of the bus, we can improve its spatial utilization. One benefit of packet-switched interconnect is that multiple transmissions can happen simultaneously at different parts of the fabric. A similar effect can be achieved if we divide the transmission line into a few segments. When a node is communicating with another node within the same segment, it only needs to arbitrate for the segment of the bus, leaving other segments free for other independent transmissions. When the communication crosses multiple segments, the transmitter needs to obtain permissions for all segments and the segments will be connected to act as a single transmission line.

Note that such electrical segmentation is fundamentally different from buffered buses which are essentially ring interconnects. Our segmentation does not change the architectural model of a global bus: delivery of a packet about an address does not overlap with that of another packet on the same address. Those packets are globally serialized. Maintaining such feature allows significant simplification of the coherence protocol [22] and other optimizations.

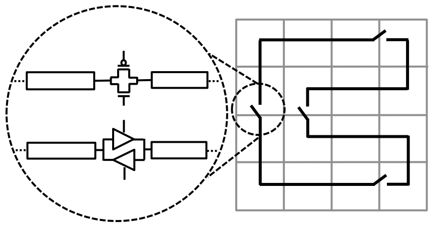


Figure 6: The chip layout with 4 segments. Each segment is connected by either a pass-gate switch or an amplifier pair.

Electrically, the segments can be connected in two ways, as in Figure 6. First, a pass-gate can be used to form a passive, bi-directional connection. In this case, the pass-gate adds a little bit of attenuation and signal distortion. We find the impact to be acceptable when the number of segments is low.

Second, two separate uni-directional amplifiers can be used to connect neighboring segments. The cost of this approach is the power consumption for the amplifiers. However, with these amplifiers, the source transmitter power can be lowered somewhat since the signal travels at most the length of one segment and is essentially repeated at the segment boundary.

For arbitration, we use one local arbiter for each segment. Each arbiter has request and grant ports to all local nodes as well as to other arbiters. Intra-segment arbitration is completely local to the segment arbiter. Inter-segment communication requires two-stage arbitration, where the sender's local arbiter request for the use of other segments.

Local links: In shared-memory programs, there are intrinsic reasons behind near-neighbor communications that result in local packets. An extreme form of that locality is nearest-neighbor communication. A globally-shared bus topology delivers 100% of it total bandwidth as its bisection throughput.[4] This allows the bus to have a much lower total throughput compared to alternatives and yet still satisfy real workloads competently. But a global bus is a poor match for nearest-neighbor communication patterns. Adding dedicated links just for neighbor communication is one way to mitigate the problem.

We avoid any packet switching and the associated complexities on these local links. Furthermore, since these links are not intended to suit all traffic patterns, we simply use a ring. Such links can be built with just digital links since the distance is relatively small. If transmission lines are used for local links, the pitch needed is much smaller than that of the global lines.

[3]In our setup, about 41% of packets are data packets. With narrow buses, the longer data packets utilize raw bus throughput more efficiently (4 flits in 6 cycles compared to 1 flit in 3 cycles for meta packets). Thus, the total effective throughput demand for data packets is about 58%.

[4]Compared to 1/6, 1/4, and 1/3 for (2D) mesh, torus, and flattened butterfly topologies in a 4×4 network.

4.3 Optimizations on the Use of Buses

Unlike its off-chip counterpart, an on-chip interconnect is not subject to certain interface requirements such as those dictated by the pins of the chip. Evaluating a bus only as a backward-compatible, drop-in replacement for a packet-switched interconnect would underestimate its potential to help optimize the entire stack. Given the bus's unique properties, we can convey certain information much more efficiently.

Invalidation acknowledgement omission: A first example is the opportunity to omit invalidation acknowledgements. In a coherent shared-memory system, the knowledge of store completion is needed in implementing memory barriers or write atomicity (our system supports Alpha consistency model with write atomicity). With a packet-switched network, protocols rely on explicit invalidation acknowledgements to provide the knowledge of completion. If the interconnect offers certain capability to help *infer* the delivery, an explicit acknowledgement can be avoided [22]. A traditional bus is one such case. Protocols rely on the *commitment* of carrying out received invalidation requests instead of acknowledgement [18]. In other words, the nodes ensure that the invalidation will be logically ordered before any out-going transactions and this commitment effectively serves as an implicit, instantaneous acknowledgement.

Note that interconnects such as the Element Interconnect Bus for IBM Cell processors [4] are essentially rings, despite the name. These "buses" relay packets and cannot omit invalidation acknowledgements. Our system always delivers packets end-to-end in an atomic bus transaction. This is true even with the segmentation discussed earlier, since the segments are electrically joined into a single bus before the packet is transmitted in an atomic transaction.

Limited multicasting: While transmission lines are most often used for point-to-point communications, they can be designed to allow multicast operations. In our system, supporting a small number of simultaneously operating receivers is relatively easy. Our circuit simulation shows that if two receivers are turned on, there is a tolerable 5% additional attenuation for the signal at the more distant receiver. Multicasting finds natural usage in sending out invalidations to multiple nodes. We find that on average, 40% of invalidations are directed at multiple nodes. We choose to support 2-way multicasting only. While the traffic reduction due to 2-way multicasting may not be dramatic, it drastically cuts the latency and queuing delays during traffic bursts resulting from invalidations of widely held data.

Fuzzy invalidation: We can send shortened messages to convey invalidation addresses. One approach is to use lossy compression that reduces the precision about the lines to invalidate. Taken to an extreme, our design uses 1 bit to represent a group of special lines. We find that a sizable fraction of cache lines are used only immediately after they are brought in and never again until eviction or invalidation. If a line shows this behavior consistently, it is a candidate for such imprecise or fuzzy invalidation (FI or ϕ) as the risk of invalidating the line prematurely is low. We model a simple implementation that only identifies lines not accessed again after being brought in. When such a line is evicted, with a certain probability (25% in our case) the cache will notify the directory about its ϕ-readiness. When a ϕ-line is fetched to L1 cache, the line's ϕ-bit will be set. The cache controller uses this bit to flash-invalidate all ϕ-lines upon receiving a fuzzy invalidation command. When the directory serves a write request to a ϕ-line, it send the ϕ-command by sending a pulse over a special broadcast transmission line.

Boolean bus: Similar to fuzzy invalidation, we can build a narrow specialized bus to support transfers of boolean values (0s and 1s), which are commonly used in synchronizations. To simplify the design, the boolean bus is only used to send a data reply when the line is boolean, *i.e.*, all but the least significant bit are zero. Our software library that implements locks and barriers spaces the synchronization variables into single-word lines and uses load-link instructions to suggest the hardware to send special boolean requests. When serving such a request, a simple 0-test is performed to decide whether the boolean bus is used to send the reply.

4.4 Interactions Between Techniques

The aforementioned three groups of techniques are largely orthogonal as they target different sources of performance gain: increasing the raw supply of throughput, reducing the throughput that has gone to waste, and reducing the number of messages. Within each group, there is a varying degree of overlap between techniques.

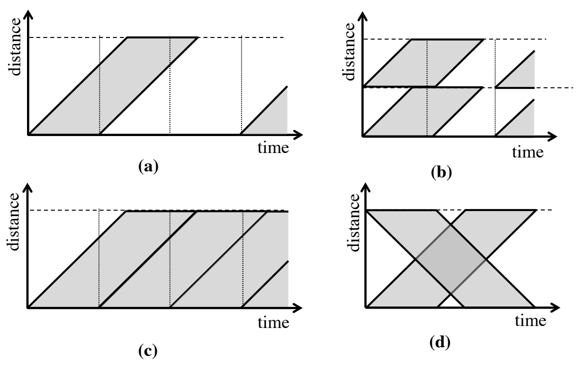


Figure 7: Illustration of throughput utilization of different configurations. The shaded area shows the pulse train propagates along the bus in time. The baseline configuration (a) leads to under-utilization. Segmentation (b), partitioning (c), which serializes a data packet into a multicycle pulse train, and wave-based arbitration (d) reduce idle time in different ways.

Let us first look at the group of techniques that increase utilization efficiency. When we send a pulse train on the bus, we wait until it propagates beyond the ends (absorbed into the impedance-matched terminators) before allowing another pulse train on the bus. Because the propagation delay is significant compared to the length of the pulse train, any point on the bus is actually "silent" most of the time. In other words, the duty cycle of the bus is rather low, as illustrated in Figure 7-a. Techniques in Section 4.2 improve the duty cycle in different ways (Figure 7). In general, implementing one technique reduces the potential of another.

The group of techniques which optimize the usage of the buses (Section 4.3), on the other hand, have less overlap in their effects as they tend to reduce different types of traffic: some reduce packets for invalidations and others packets for acknowledgements. Of course, in the end, when multiple techniques are applied (regardless of their target), we can reach diminishing returns.

5 Experimental Analysis

5.1 Experimental Setup

Sonnet [1] and Advanced Design System (ADS, from Agilent Technologies) were used to perform circuit and physical simulations of the transmission line links. Sonnet is an 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, 112)
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 cycles
L2 cache (shared)	128KB slice/core, 8-way, 64B line, 15 cycles, 2 ports
Intra-node fabric delay	3 cycles
Main memory	at least 250 cycles, 8 memory controllers
Network packets	Flit size: 72-bits data packet: 4 flits, meta packet: 1 flit
NoC interconnect	4 VCs; 2-cycle router; buffer: 5x12 flits wire delay: 1 cycles per hop [33]
Transmission line link (each node)	
Bit Rate	26.4 Gb/s, 8 bits per CPU cycle
Transmission latency	2 cycles (worst-case)
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
Arbiter	Arbitration latency: 1 cycle Request/grant propagation delay: 120ps (max)

Table 2: System configuration.

Architectural simulations of the proposed design were performed using an extensively modified version of SimpleScalar [9]. 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 are listed in Table 2.

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

We use a set of diverse multi-threaded 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 3 lists the applications used. Abbreviations are used in the data figures and listed in the table. Inputs for each application are listed along with a brief description of the application. Each application is fast-forwarded past the initialization. To mimic data placement optimizations such as placing private pages at the local node, cache misses are sampled randomly at the rate of 0.1% offline. Data pages with a dominant accessing node are mapped local to that node. Other pages are allocated round-robin.

Splash-2 [44]	
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)
radix (rx)	integer sort algorithm (1M integers)
raytrace (ry)	3-D rendering (car.env)
water-sp (ws)	molecular dynamics (512 molecules)
Parsec [8]	
blackscholes (bl)	financial analysis/calculation (16K options)
fluidanimate (fl)	animation (5 frames, 35K)
Other Benchmarks [17, 21]	
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)

Table 3: Benchmarks used.

5.2 Application Characteristics

We first analyze the characteristics of the benchmarks. In a first testbed, the 64-cores are clustered into 16 nodes, 4 cores each. The cores in the same cluster share the interconnect circuitry. It is worth noting that among our benchmarks, some are already performing well on the baseline TLL bus without any throughput enhancing techniques. Since their performances already come close to running on an idealized interconnect, there is little room for further improvement. To more clearly understand the impact of the techniques discussed so far, we have divided the benchmarks into 3 groups (G1-G3) with increasing potential performance benefits of throughput enhancement. This grouping is done by comparing the performance of applications under three different types of interconnects: a baseline TLL bus, a (concentrated) mesh NoC (with both a 2-cycle router and an idealized 1-cycle router), and an ideal interconnect modeling only the latency of aggressively configured metal wires [33]. Figure 8 illustrates this classification.

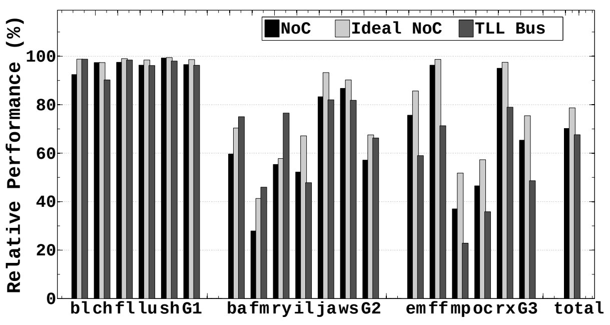


Figure 8: Performance of baseline TLL bus and NoC (both 2-cycle and ideal 1-cycle routers) normalized to wire-delay-only ideal interconnect. The 3 groups represent, from left to right, the benchmarks with increasing room for performance improvement for the TLL bus.

In G1, the benchmarks have low throughput demand that is well met by the baseline bus and the performance comes in at least 90% that of ideal interconnect. These applications will see little, if any, performance improvement from optimizing just the interconnect.

In G2, even though there is a significant performance gap between baseline bus and ideal, the bus still performs better or within 10% of the NoC. In fact, the bus outperforms the NoC on average. Only when we use the 1-cycle ideal router do we see the NoC slightly outperforming the bus. Clearly, the latency advantage of the bus is important. For these applications, improving throughput will not help if it comes at a significant cost of latency.

Finally, in G3, NoC clearly outperforms the bus, suggesting ample room for improvement when the bus throughput increases.

5.3 Performance Benefits

We have described many different ways of improving effective throughput. Which ones ought to be pursued in a practical design depends on many factors, some of which hard to quantify. Below, we will first show these techniques' impact on execution speed and on on traffic and effective throughput when applied in isolation.

In Figure 9, we sort the techniques by decreasing mean performance improvement. For brevity, we only show the (geometric) mean and the range of relative performance. As a frame of reference, we also include the result from the ideal interconnect, which clearly shows the ample performance headroom as well as significant variability among applications. Two general observations can be made from this summary figure, which we elaborate below: 1. raw throughput is not as critical as intuitively expected; and 2. even simple coherence optimizations can be fairly effective.

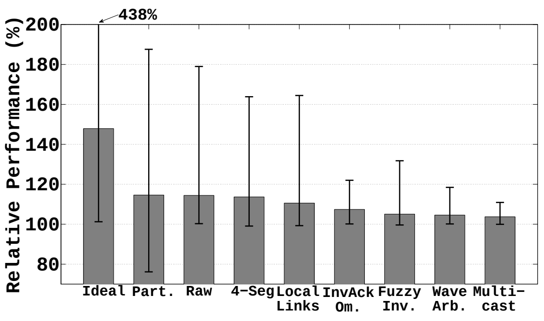


Figure 9: The performance impact of the techniques discussed. The bars show the geometric mean of relative performance over a baseline bus on all application, while the I-beams show the range from all applications. Note that the y-axis does not start at 0.

Importance of throughput: Throughput is a metric used routinely to characterize a network's performance. This can be a misleading oversimplification in the context of on-chip interconnect for CMPs. In a CMP, traffic is a direct result of cache misses. Various data dependencies and limits on buffers or outstanding requests constrain the packet-level parallelism. This is different from, say, file transfers where more throughput can always be readily leveraged. Once over a certain threshold, throughput only has a secondary impact as it affects latency indirectly through queuing and serialization delays. As a result, more throughput supply is only beneficial when the latency impact of obtaining more throughput is small.

We can see this effect in Figure 9. Increasing the raw throughput (in this case doubling it via either 4-PAM encoding or 2-band FDM) provides similar benefits as the techniques that merely try to incrementally improve the utilization efficiency. These techniques (*e.g.*, partitioning) carry little latency and energy costs. In contrast, a NoC achieves high throughput at a more significant latency cost due to repeated packet relays. The high latency is then being mitigated with complex, speculative router designs that further drive up energy cost. Eliminating or at least reducing the reliance on packet switching in on-chip interconnect design is a direction that deserves more attention.

Effectiveness of simple coherence optimizations: Another set of techniques reduces traffic demand by leveraging the properties of a bus. These include invalidation acknowledgement omission (IAO), fuzzy invalidation, and multicasting. These techniques can make a non-trivial performance impact (*e.g.*, up to 1.3x for fuzzy invalidation), although they do not directly increase the nominal throughput of the interconnect. Note that in some cases, the benefits will increase when programs start to use these underlying mechanisms (*e.g.*, boolean bus) for more purposes.

Interaction between techniques: Table 4 shows the techniques' impact on reducing traffic and improving bus utilization. We define bus utilization as the ratio of actual throughput over theoretical throughput under continuous traffic. In other word, we exclude cycles when there is no traffic demand. Under this definition, the baseline bus has an average of 51% for the meta bus and 58% for the data bus.

	Utilization Improvement		Traffic Reduction	
Technique	Average	Maximum	Average	Maximum
Partition	1.08x (M) 1.51x (D)	1.16x (M) 1.70x (D)	-	-
4 Segments	1.24x (M) 1.29x (D)	1.30x (M) 1.56x (D)	-	-
Local Links	1.07x (D)	1.37x (D)	32% (D)	89% (D)
IAO	-	-	12% (M)	25% (M)
Fuzzy Inv.	-	-	4% (M)	22% (M)
Multicast	-	-	2% (M)	5% (M)

Table 4: The utilization improvement and traffic reduction for meta buses (M) and data buses (D) for each technique.

To a first degree of approximation, performance improvement is correlated with the degree of traffic reduction or utilization improvement. Also, the overlap between different techniques is not significant. For instance, partitioning and segmentation improve data bus utilization by 1.51x and 1.29x respectively. When combined, the utilization improves by 1.65x to 96%. Similarly, when the traffic-reduction techniques are applied together, the effect is close to additive, though some techniques (such as multicasting) have small contributions.

5.4 Costs

The costs of these techniques include extra circuit support and runtime energy expenditure. The techniques can be grouped based on these costs:

- Little to no cost: Partitioning and IAO require only a different way of organizing resources and need no new circuits.
- Some circuit cost: Multicasting, fuzzy invalidation, and wave-based arbitration require some support from the circuit, but there is no run-time energy cost.
- With circuit and energy costs: The remaining techniques incur some circuit costs and energy costs.

  In segmentation, the energy cost depends on the implementation of the bridges: a bridge with a pass gate incurs very little energy overhead itself but attenuates the signal a little bit. We conservatively assume a more costly amplifier-based bridge, each consuming about 90% of the transmitter's power. On average, we observe about 40% of the packets cross the bridge in 2 segments, and about 65% cross one or two bridges in 4 segments.

  Local links implemented with transmission lines do not add any energy overhead and in fact use less powerful drivers. They do require slightly more area to be devoted to transmission lines even though each link uses narrower transmission lines. Conservatively, we assume local links using digital wires that do not take metal area for the global transmission lines. Our synthesis results show that factoring in the controller, transmitting over local digital links costs about 4 times the energy as that over the global buses. On average, about 35% of data traffic is off-loaded to the local links.

  Finally, providing raw throughput, especially through FDM, is a more energy intensive option. We estimate the PAM design to double the energy per bit of an OOK link. FDM requires an increase in the transmitter and receiver power in order to compensate for the increased attenuation on the higher frequency band and the noise figure, introduced by mixers. We assume a noise figure of 5dB per mixer, a 6dB increase in attenuation, and thus a compensation of 8dB on both transmitter and the receiver side.

5.5 Comparison of Cost Effectiveness

We now summarize the high-level cost benefit analysis of each individual mechanism. This analysis does not tease out the synergy or overlap between multiple techniques when deployed together. But the analysis still gives a reasonable picture of what some of the first steps we should take to increase effective throughput.

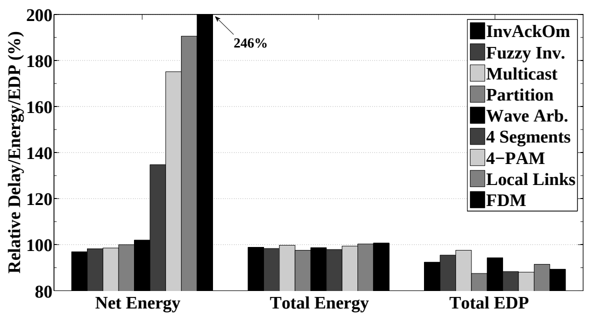
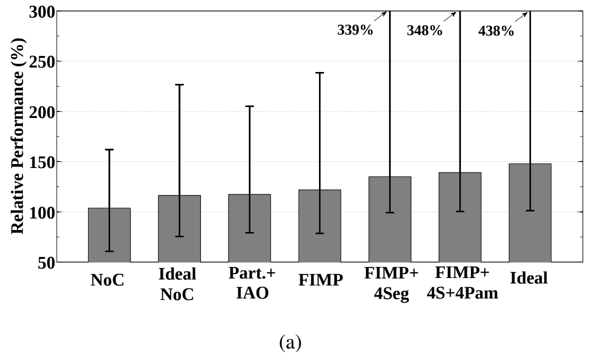


Figure 10: The relative network energy, chip energy, and energy-delay product of applying each technique discussed in a 64-core, 16-node system.

Figure 10 shows energy-related metrics for each individual technique applied in isolation, all normalized to baseline bus. From left to right, the bars are ordered by increasing network energy. The first five techniques have very little energy overhead and in fact some save network energy by sending fewer packets. Starting with (4-way) segmentation, the last four techniques have noticeable energy increases in the network. But the performance benefit reduces energy spending elsewhere in the system (*e.g.*, clock distribution), so the chip wide energy is actually reduced.

5.6 Example Design Points

Given this array of the techniques, a chip designer can put together a set of them to suit the needs of the chip. In Figure 11, we show four such configurations. The first configuration combines partitioning and IAO. These two techniques not only have little costs, but have significant performance impacts. The next configuration adds multicasting and fuzzy invalidation. At this point, there is no change in the nominal throughput of the bus and no increase in energy of the network. The performance is already 1.22x that of the baseline bus, 1.17x faster than a mesh NoC, and higher than a mesh NoC with idealized 1-cycle routers.



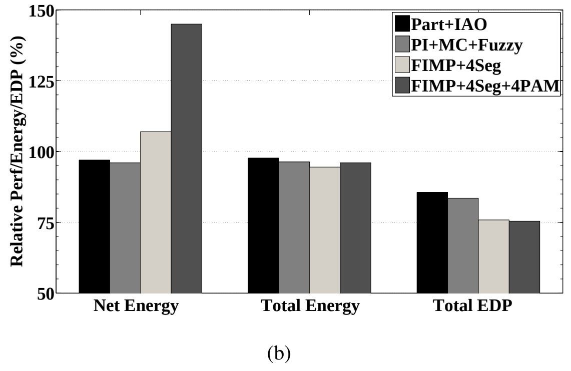


Figure 11: The effect of a few configurations. FIMP is short for the combination of fuzzy invalidation, IAO, multicasting, and partitioning. (a) The relative performance with geometric mean and range from all applications. An ideal interconnect is shown as a reference. (b) The relative network energy, total chip energy, and energy-delay product.

In the next two configurations, we progressively add segmentation and 4-PAM. The system performance improves to 1.39x. An ideal interconnect is only 1.06x faster. While the network energy is much higher, the faster speed compensates partly and the chip energy is still comparable to the baseline bus. Note that these results are the average of all applications. Within G3 applications, the benefits are much more obvious: 1.89x speedup at an average of 7% less energy.

Clearly, the effectiveness of these techniques is highly dependent on the application behavior. As we already saw, G1 applications will not see much performance gain even if the interconnect is ideal. Therefore, it would be helpful to have some dynamic adjustment to turn on power-hungry communication mechanisms only when there is significant performance benefit to be gained.

Comparison with NoC: As we have shown, with some enhancements, the effective throughput can be increased with low energy costs. Compared to the bus, a NoC solution starts from a high-throughput design point. But the high throughput comes at the expense of energy intensity (NoC's network energy is 15x that of baseline bus) and higher latency and may not necessarily translate to high application performance. Figure 12 shows the experiment that uses injected traffic (uniform random) to measure packet latency in different configurations. We can see that techniques discussed significantly extend the saturation point of the bus without increasing packet latency at low load. While NoC has a higher saturation point, the common-case latency is worse.

Note that the uniform random traffic patterns show the best cases for the NoC configurations, whereas a bus architecture is much less sensitive to the traffic pattern. When we use execution-driven simulations, the benefits becomes much more obvious. In G3 applications, where the baseline bus lags significantly behind NoC in performance (0.75x), the improved bus now is 1.4x times faster than NoC.

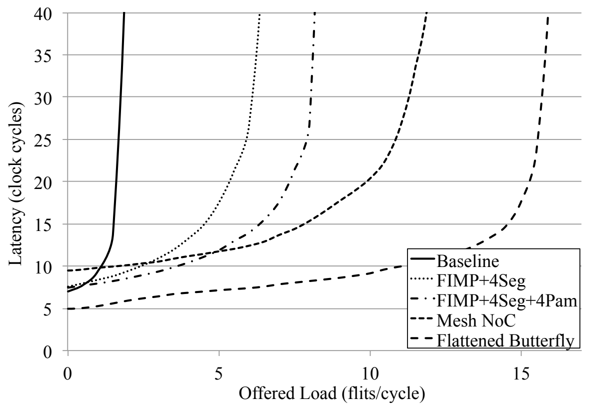


Figure 12: Packet latency vs offered traffic of various interconnect configurations. Note that the model of flattened butterfly assumes no increase in router delay.

Of course, there are optimizations to improve the latency of the router and to minimize network diameter. But these come at even higher energy costs and can have limited effectiveness. For example, when we idealize the router delay to 1 cycle, the improved bus still shows a speedup of 1.19x over all applications (Figure 11-b). It is only when we use a flattened butterfly topology with the idealized 1-cycle router, that the NoC is outperforming the improved bus by 1.04x.

In practice, these speculative, look-ahead routers can only achieve 1-cycle routing delay in the best case. And using higher radix routers (to enable topologies like flattened butterfly) do not fundamentally change the total routing delays, but only reduces hop at the expense of increasing per-hop router delay [28]. Our models of NoC, especially with flattened butterfly topology, are only capturing the benefits not the performance costs – nor any energy costs. These models are providing an increasingly loose upper-bound for their performance potentials.

5.7 Scaling Up

It is a little tricky to study the impact of these techniques in an even larger system. The parallelism of the applications, the simulation environment, and the data set all start to reach or pass their fidelity limit and will contribute significant noise towards the measurements. So instead of trying to simulate more cores and threads, we do the following two things to escalate only the traffic on the network. First, we turn off the data placement optimization. Second, we make each core a standalone node. We hope to use this environment only to shed some light on the techniques' impact in a larger scale environment, not to predict precise performance or energy gains.

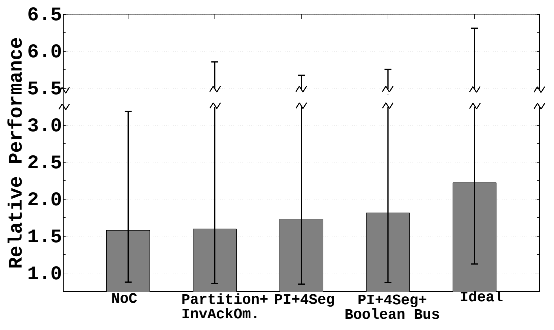


Figure 13: The performance of various configurations under escalated traffic environment, all normalized to baseline bus.

Figure 13 shows the comparison of the NoC, bus, and a few configurations of improved bus. All results are shown as normalized to baseline bus. In such an environment, the baseline bus is under far more pressure and, at less than half of the performance of ideal interconnect, it is significantly slower than NoC on average. The several improved bus designs all outperform the NoC, while using 9-15x less energy in the network.

With these analyses, including the limited scaling study, some insights can be obtained:

1. A bus architecture can be augmented with various techniques to be a viable solution even for large-scale CMPs.
2. Sometimes, these techniques come at a non-trivial cost in interconnect energy efficiency. Nevertheless, compared to the NoC approach, the energy cost is still much smaller.
3. Applications demonstrate a diverse set of behaviors that call for an adaptive control mechanism that can increase throughput on demand at the cost of extra energy overhead. NoC, to the opposite, operates at a point that provides high throughput at a significant cost of energy and latency.

6 Conclusions

In this paper, we have discussed an array of techniques to enhance throughput of transmission line buses via increasing the utilization efficiency, leveraging the bus properties and transmission line capabilities to reduce traffic demand, and to directly increase the raw link throughput. Among these techniques, those that increase the raw throughput often carry a higher energy cost for the same performance benefit. Even so, the energy cost is still far lower than that of using a NoC.

In a 64-core, 16-node system, when a number of techniques are applied, the performance of the system is improved by 1.39x and is 1.34x faster than the same system using an mesh NoC. This performance improvement is achieved with a 2x increase in interconnect energy, but a decrease of 5% of chip energy compared to the baseline bus thanks to faster execution. Compared to the mesh NoC, the network energy is still 8x lower, while the chip energy is reduced by 7%. In a limited scaling study where interconnect traffic is escalated, the throughput-augmented bus continues to outperform NoC almost consistently across all benchmarks.

In summary, transmission line-based links are a readily available mechanism to provide high-speed low-power communication. Using these links to build bus structures is a compelling alternative to NoC and other technologies such as on-chip photonics, which are far from being practical in the near term. The effective throughput of bus can be significantly increased with simple, practical designs. As a result, a bus architecture can support chip-multiprocessors at the scale of 64 cores competently with a much better energy profile than NoC. These findings bring into doubt the necessity of heavy-duty packet switching for on-chip interconnect in the foreseeable future.

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

[5] 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.

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

[7] 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.

[8] 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.

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

[10] A. Carpenter, J. Hu, M. Huang, H. Wu, and P. Liu. A Design Space Exploration for of Transmission-Line Links for On-Chip Interconnect. In *Proc. Int'l Symp. on Low-Power Electronics and Design*, pages 265–270, August 2011.

[11] A. Carpenter, J. Hu, J. Xu, M. Huang, and H. Wu. A Case for Globally Shared-Medium On-Chip Interconnect. In *Proc. Int'l Symp. on Comp. Arch.*, June 2011.

[12] 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.

[13] 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.

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

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

[16] M. Cianchetti, J. Kerekes, and D. Albonesi. Phastlane: A rapid transit optical routing network. In *Proc. Int'l Symp. on Comp. Arch.*, pages 441–450, June 2009.

[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] D. Culler and J. Singh. *Parallel Computer Architecture: a Hardware/Software Approach.* Morgan Kaufmann, 1999.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] V. Issakov, H. Knapp, M. Tiebout, A. Thiede, W. Simburger, and L. Maurer. Comparison of 24 GHz low-noise mixers in CMOS and SiGe:C Technologies. In *European Microwave Integrated Circuits Conference*, pages 184–187, September 2009.

[26] H. Ito, J. Inoue, S. Gomi, H. Sugita, K. Okada, and K. Masu. On-chip Transmission Line for Long Global Interconnects. In *IEEE International Electron Devices Meeting. IEDM Technical Digest*, pages 677–680, December 2004.

[27] H. Ito, M. Kimura, K. Miyashita, T. Ishii, K. Okada, and K. Masu. A Bidirectional- and Multi-Drop-Transmission-Line Interconnect for Multipoint-to-Multipoint On-Chip Communications. *IEEE Journal of Solid-State Circuits*, 43(4):1020–1029, April 2008.

[28] J. Kim, W. Dally, B. Towles, and A. Gupta. Microarchitecture of a High-Radix Router. In *Proc. Int'l Symp. on Comp. Arch.*, pages 420–431, June 2005.

[29] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das. A Novel Dimensionally-decomposed Router for On-chip Communication in 3D Architectures. In *Proc. Int'l Symp. on Comp. Arch.*, pages 138–149, June 2007.

[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] K. Miyashita, T. Ishii, H. Ito, N. Ishihara, and K. Masu. An Over-12-Gbps On-Chip Transmission Line Interconnect with a Pre-Emphasis Technique in 90nm CMOS. In *Electrical Performance of Electronic Packaging, 2008 IEEE-EPEP*, pages 303–306, October 2008.

[33] 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.

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

[35] 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.

[36] 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.

[37] L. Peh and W. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proc. Int'l Symp. on High-Perf. Comp. Arch.*, pages 255–266, 2001.

[38] 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.

[39] J. Seita, H. Ito, K. Okada, T. Sato, and K. Masu. A Multi-Drop Transmission-Line Interconnect in Si LSI. In *Asia and South Pacific Design Automation Conference*, pages 118–119, January 2007.

[40] A. Shacham, K. Bergman, and L. Carloni. On the Design of a Photonic Network-on-Chip. In *First Proc. Int'l Symp. on Networks-on-Chip*, pages 53–64, May 2007.

[41] 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.

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

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

[44] 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.

[45] 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.