Appears in the Proceedings of the
22nd International Conference on Parallel Architectures and Compilation Techniques (PACT), Pages 309-318, September 2013.

# Traffic Steering Between a Low-Latency Unswitched TL Ring and a High-Throughput Switched On-chip Interconnect

Jungju Oh
School of Computer Science
Georgia Institute of Technology
Atlanta, Georgia
jungju@gatech.edu

Alenka Zajic
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia
alenka.zajic@ece.gatech.edu

Milos Prvulovic
School of Computer Science
Georgia Institute of Technology
Atlanta, Georgia
milos@cc.gatech.edu

***Abstract*—Growth in core count creates an increasing demand for interconnect bandwidth, driving a change from shared buses to packet-switched on-chip interconnects. However, this increases the latency between cores separated by many links and switches. In this paper, we show that a low-latency unswitched interconnect built with transmission lines can be synergistically used with a high-throughput switched interconnect. First, we design a broadcast ring as a chain of unidirectional transmission line structures with very low latency but limited throughput. Then, we create a new adaptive packet steering policy that judiciously uses the limited throughput of this ring by balancing expected latency benefit and ring utilization. Although the ring uses 1.3% of the on-chip metal area, our experimental results show that, in combination with our steering, it provides an execution time reduction of 12.4% over a mesh-only baseline.**

***Keywords*—*traffic steering, network-on-chip, transmission line***

## I. Introduction

The on-chip core count has already reached the point where a shared bus no longer provides sufficient bandwidth for the coherence traffic. Further, bus latency and energy-per-bit are not scaling favorably. Consequently, a many-core chip must use a packet-switched network-on-chip (NoC), e.g. a mesh or torus [1]. In such NoCs, each link is short and its length scales down with technology, so latency and energy consumption scale well if most traffic is local. Further, the aggregate bandwidth of all the links scales with the number of cores. However, switched networks come at a significant latency cost for traffic between far-apart cores—these messages traverse more switches when the core count grows, in addition to longer wire delays due to technology scaling.

One approach for reducing NoC latency is to use transmission lines (TLs), where signals propagate close to the speed of light. However, TLs consume far more metal area than traditional wires, so a same-area TL-based NoC typically either provides far less throughput or requires very sophisticated signaling to improve this throughput.

The key insight we use in this paper is that the latency advantage of TLs over wires is the greatest for long-distance packets, while local traffic (which often represents the majority of the traffic) sees a relatively low latency even in a traditional mesh based NoC and it has little benefit from going over (expensive) TLs. Guided by this insight, we design a low-cost unidirectional TL (UTL) ring interconnect that provides very low maximum propagation latency (e.g. 2 ns for a 64-core chip), uses simple signaling and has an efficient yet simple arbitration mechanism. However, this ring has limited throughput, so we use it *together* with a traditional switched NoC, by judiciously steering each packet to the ring or the traditional packet-switched interconnect. In particular, this paper makes the following contributions:

- We design a novel unidirectional transmission line (UTL) structure that has low signal propagation latency (about 7.5 ps/mm) at relatively high bit rates ($>$16 Gbit/s) and much better multi-drop (multiple connections along the TL) and signal integrity properties than traditional TLs.
- We construct a novel ring out of these UTL structures. The ring behaves as a contiguous TL propagation medium, but uses electrical switching (CMOS pass-gates) to achieve efficient arbitration and loop-around prevention. Our EM and circuit-level models for this ring show a $<$2 ns total propagation latency for 64 cores ($<$3 ns with 256 cores) on a 16 mm$\times$16 mm chip.
- We devise an efficient and adaptive traffic steering mechanism, which manages the available bandwidth of the ring by allocating it only to packets that are expected to benefit the most from its reduced latency.

Our experimental evaluation shows that (1) combined ring+mesh, with our traffic steering, improves execution time by 12.4% on average compared to the mesh alone, (2) careful packet steering plays an important role in using this combined interconnect, and finally (3) our contributions continue to provide a performance benefit even when combined with more sophisticated (lower-latency) switched NoC topologies.

## II. Background and Related Work

### A. On-chip Network Topologies and Latencies

Generally, on-chip networks can be classified into three categories: unswitched (typically bus-based), packet-switched and hybrid interconnects that exploit both. Unswitched interconnects broadcast all traffic on a shared medium, allowing simpler design than switched interconnects. However, as the number of cores and connections increases, the resulting contention becomes a major performance issue [2], [3]. Traditional buses also suffer from unfavorable wire delay and power trend as technology scales [4]–[7].

In contrast, a packet-switched network typically uses short links between adjacent cores, which is efficient and has low-latency for local traffic. However, packets traveling to far-away destinations have to go through many links and on-chip routers. These routers are typically pipelined, which improves throughput but further increases per-hop latency. Studies show that this per-hop latency can be alleviated by techniques such as lookahead routing [8] where routing calculation is performed one hop ahead to shorten the router's pipeline, or aggressive speculation [9] which also reduces pipeline length by speculatively performing switch allocation early. However, even with these advanced router microarchitectures, each traversed node still adds several cycles to the latency of a packet.

Hop count can be reduced by more advanced network topology. In a concentrated mesh [10], several nodes share each router, so fewer routing hops are needed than in a traditional mesh. In a flattened butterfly [11], hop count is reduced by providing richer physical connectivity to non-adjacent nodes. Express virtual channel [12] uses dedicated virtual channels to shorten the latency for multiple-hop traverse and multi-drop express channel [13] achieves high connectivity in an efficient manner. However, even with these efforts, growing core counts still result in increasing hop counts and wire latency, leading to latencies of tens of cycles between far-apart cores.

The per-hop routing delays can be eliminated either by using a separate link for each pair of cores, which is extremely expensive for a large number of cores, or by using an unswitched interconnect. Unfortunately, as noted earlier, traditional unswitched interconnects (buses) suffer from both contention and poor scaling trends.

The latency and energy problems of wires in unswitched interconnects can be addressed with transmission lines (TLs), where signals propagate as electromagnetic waves (without charging the entire capacitance of the link) close to the speed of light in the conductor material [14]. For the specific TL-based design we use (Section III-A), electromagnetic simulations show 7.5 ps/mm propagation speed, compared to ITRS projections [5] of 140 ps/mm for optimally repeated wires in 22 nm, and 150 ps/mm in 10 nm technology. However, TLs do not address the throughput problem of unswitched interconnects—in fact, they make it worse because the metal area occupied by a single TL is equivalent to tens of wires. Thus, a TL-based interconnect tends to have less throughput than the same-cost wire-based one.

## *B. Related Work*

In contrast to our work, where both networks are fully capable of delivering any message from its source to its destination, most prior work with multiple networks uses them hierarchically. Bourdaus [15] proposes hierarchical rings connecting local meshes to reduce global hop count. Local buses with global meshes are another hierarchical interconnect [16], [17] that can improve latency and power efficiency through communication locality. However, these hybrid proposals still incur significant routing delays for cross-chip packets in many-core processors, and growing wire delays are still a problem. In contrast, our approach combines a (traditional or advanced) switched NoC with a low-cost UTL ring that provides extremely low latency for selected (e.g. cross-chip) packets.

Other approaches to improve latency of many core interconnect include attempts to hierarchically split chip-spanning bus into multiple local buses connected with point-to-point links [18]. Filtering broadcasts between segmented global and local buses further improves bus throughput [19] and the use of hierarchical (local and global) rings show performance close to the packet-switched interconnect with improved scalability [4]. However, these interconnects still show strain in high-traffic applications, so they are likely to only delay the transition to more scalable switched interconnects. In contrast, we embrace the transition to more scalable NoCs, but add TL ring to provide extremely low latency when needed. Furthermore, advances in scalable switched interconnects in terms of latency [12], [20] and power efficiency [21] may still result in significant latency differences depending on hop count and such NoCs can still benefit from our approach.

Ring interconnects have been widely used due to their simplicity, with trivial routers and manageable latency with short point-to-point wires. IBM Power5 [22] and IBM Cell Architecture [23] use a ring to connect multiple on-chip computing units. Our approach also benefits from the simplicity of the ring, but improves the ring's latency by using novel TL-based couplers and by allowing seamless signal propagation with pass-gates. Thus our ring can form a fast chain that starts and ends at the sending node. The coupler works similarly to a ring resonator in nanophotonic interconnects [24], but allows a node to both receive the signal and pass it on.

Steering messages in heterogeneous networks was explored in [25]. The heterogeneous network implements multiple interconnects with different characteristics in latency, bandwidth and power-efficiency by adjusting wire and repeater spacing. Then, coherence traffic is optimized to steer messages into different networks according to their properties (criticality, size, etc.). In contrast to steering between NoCs that are similar in nature, our approach steers between NoCs with very different latency, throughput and latency/hop characteristics.

Transmission lines (TLs) have been used for fast communications, e.g. low-latency barriers [26], [27], long-distance "express" links [28], [29] and connections between distant banks in large caches [30]. Shared-medium broadcast TL interconnects in [31] use optimizations to reduce on-chip traffic. While these optimizations defer the scalability problem, growth in core counts still inevitably leads to saturation of broadcast-type interconnects. In contrast, we rely on a switched network to provide throughput scaling, and use our TL ring only for packets that benefit the most from it.

Alternative emerging approaches for low-latency interconnection include nanophotonics [24], [32]–[34] and on-chip wireless communications [35]. Nanophotonic NoCs are able to address the power and latency problem of the traditional NoC, though they require further investigation in their suitability in replacing existing metal-based NoC architecture [36]. Our application of TL in augmenting the existing NoCs differ with

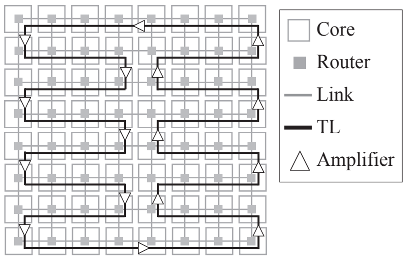

**Fig. 1:** TL-based ring with packet-switched network.

their approach in that we exploit already mature technologies and can ease the burden of installation. Hybrid wireless NoC expedites long-range communication but requires routing scheme and communication protocol that consume power and area overhead. Also, it has to overcome reliability and integration challenges.

Finally, a recent proposal describes a NoC composed of TL buses, with various optimizations to increase throughput [37]. However, such all-TL approaches increase throughput by using more TLs, complex encoding/decoding to embed more bits in the signal, and/or signaling at extremely high frequencies. Our work is based on the insight that TLs provide little benefit for a large portion of the traffic (e.g. local traffic). That leads us to a low-cost approach of minimizing total TL length and using binary pulse signaling at frequencies for which the required circuitry has already been demonstrated in existing CMOS technology [38], [39]. We then use the UTL ring to provide low latency for packets that can benefit from it while relying on a traditional scalable NoC for throughput requirement.

## III. Transmission Line Ring

This section presents a detailed design of a TL-based low-latency ring. Because TLs are expensive and have extremely fast propagation, we chose a topology that minimizes overall TL length rather than the longest path—a ring (Figure 1) rather than a double tree [27] or a grid [26]. Our models (Section III-A) show that the entire ring can still be traversed in 1.6 ns (64-core chip in 22 nm technology) to 2.6 ns (256-core chip, 10 nm technology). The latency gain from a lower-latency topology would be only 1–2 cycles, but the cost would be several times higher. Furthermore, a ring design allows a fast arbitration approach (Section III-B).

### A. *Unidirectional Transmission Line Ring*

We implement our TL ring by chaining unidirectional transmission line (UTL) elements and periodically using amplifiers to make up for losses along the ring. Following the discussion of the UTL elements and how they are connected into a ring (Section III-A1), we discuss how we prevent the signal from infinitely looping around the ring (Section III-A2) and how data bits are transmitted and received on this ring (Section III-A3).

*1) UTL-Based Ring Design:* The goal of our TL design is to manage two main problems that traditional TL designs with many connections (for transmitters and receivers) face—attenuation of signal power due to split at each connection and signal reflections/reverberations due to connections and other discontinuities (e.g. slight impedance mismatches along the TL) [27]. With naïve connections like T-junctions, significant reflection cannot be avoided. Worse, the optimal design from reflection standpoint is to split the remaining signal power half-half, but it weakens the forward-propagating signal quickly as the signal goes through multiple connections. Soon, the "noise" caused by the sum of the many reflections of reverberating signal (with various delays and attenuations) between the connections will dominate the weakened signal. As a result, the number of connections that are feasible for one TL segment is low, especially for high frequencies: 6–8 connections (two at the ends of the TL and 4–6 along the way) up to 10 GHz frequency and even fewer at higher frequencies (but they allow high data rates).

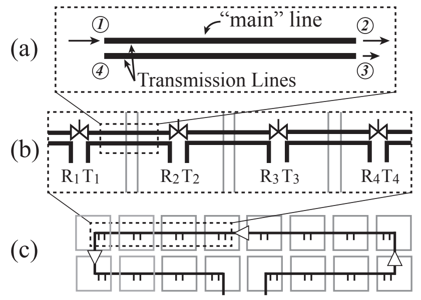

**Fig. 2:** Connecting UTL couplers into a ring.

Because of these considerations, we design a new TL structure that provides unidirectional propagation. This way, reflections cannot reverberate through the TL medium and more signal power can continue to propagate along the ring at each connection. The new structure is based on a four-port coupler, which is traditionally used in microwave designs to reduce reflection issues in signal split. We base our coupler on a recent design [40] that has excellent directivity and frequency characteristics. A simple coupler consists of two TL structures running in parallel close to each other, as shown in Figure 2a. If the signal is injected into one end of one of the lines (e.g. ① in Figure 2a), it propagates to the other end, but some part of the signal's power is transferred to the other line through EM coupling. Interestingly, with careful design, it is possible to achieve excellent *directivity* for such a coupler. In our example, almost all of the transferred power goes to port ③ on the bottom line while nearly nothing goes to port ④. Note that the coupler is symmetric and any of the four connections can be used as "input". For example, if the signal is injected into the port ④ instead of the port ① in Figure 2a, it still splits between the right ends of the two lines (port ② and ③).

This allows us to chain couplers together using one of the lines as the "main" TL and the other line to connect transmitters and receivers (Figure 2b). Traditional couplers are designed to split the signal equally between the outputs and the signal on the "main" line loses half of its power at each coupler. This would require amplifiers to be placed frequently along the ring. We modify the coupler design so that most of the signal's power continues down to the "main" TL and only 25% of the power goes to the receiver. This allows us to save half of the amplifiers.

To achieve improved frequency, directionality and power transfer properties, in our coupler, each of the two "lines" consists of two conductors that are connected at their endpoints

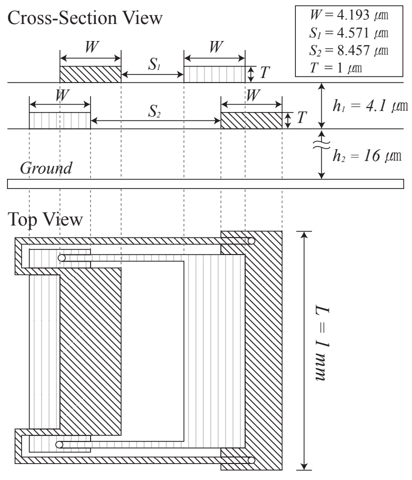

**Fig. 3:** Cross-section and top view of our coupler.

as shown in Figure 3. We still favor this design over a regular TL because it simplifies implementation and reduces the circuitry needed in each node and along the ring.

*2) Infinite Loop Prevention:* Amplifiers in our ring compensate for losses in signal strength. So the signal, once injected by a transmitter, would continue to circle around the ring infinitely and prohibit transmission of new signals. To prevent such infinite loops, each transmitter controls a pass-gate that connects the previous coupler's "main" ring connection to its own. This pass-gate is normally in the connected state, i.e., the signal is allowed to propagate freely. However, when a transmitter is about to transmit a packet, it first puts the pass-gate it controls into the disconnected state. This transforms the ring into a chain so the signal reaches all the other nodes but does not continue along the ring for the second time. Note that each pass-gate only introduces a single transistor delay into the signal's path. This delay improves with technology scaling, making these pass-gates both cost- and energy-efficient.

As shown in Figure 2b, the receiver of each node is connected before the node's pass gate, so sender is the last to receives its own signal. This can be used for error detection, although we omit that discussion because our ring has low error rates ($10^{-15}$ or less). More importantly, this arrangement allows our new arbitration approach (Section III-B).

*3) Signal (De)Modulation:* The modulation method is a tradeoff between receiver/transmitter complexity, their latency, error rates, and the bit rates that can be achieved within a given frequency band. Typically, high bit rates require modulations with multiple bits per "symbol" (e.g. signal level in AM modulation) with more complex and longer-delay transmitters/receivers. Also, such modulations have smaller differences between symbols. Furthermore, they tend to require complex and long-delay approaches to reduce the error rates.

The primary goal for our TL ring is to minimize its latency and cost. Consequently, we use a very simple form of ultra-wideband modulation: on-off keying of a high-frequency clock signal. With this modulation, one or more consecutive pulses of the clock represent one bit, and these pulses are either transmitted (bit is one) or not (bit is zero).

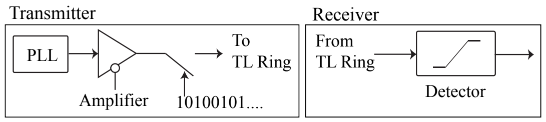

**Fig. 4:** Transmitter and receiver design for TL ring.

The pulse frequency and bit rate should correspond to the central frequency and half-width of the available spectrum of the transmission medium. Our coupler design has available spectrum that scales up as its length scales down. For 1 mm length, the available spectrum is 20–60 GHz. Thus we use a 40 GHz clock signal and conservatively modulate it with two pulses per bit to get 20 Gbit/s of raw throughput. As cores get smaller (and transistors faster) with technology scaling, our design can be modified to use 0.5 mm coupler length. The available spectrum would be 40–120 GHz, so 40 Gbit/s can be transmitted using an 80 GHz signal clock and two pulses per bit.

The receiver and transmitter for this modulation approach are simple (Figure 4). The transmitter contains (1) a PLL to generate pulses, (2) a pass-gate to control whether the pulses get onto the transmission line and (3) an amplifier (composed of two appropriately sized CMOS inverters) that mostly provides impedance matching. The receiver consists of two components: a detector (capacitor and a few CMOS transistors) to identify if a pulse is present in each bit position and a fast shift register (clocked by the same PLL used for the transmitter) to collect the high-rate bits so they can be used by logic operating at "normal" on-chip clock frequencies (e.g. 1 GHz). The PLLs of all transmitters can be phase-locked to the existing skew-compensated global clock signal, which is used for mesh routers and cores to derive the clock signal.

*4) End-to-End Latency on the Ring:* The latency has five components: (1) TL propagation time that largely depends on the total distance traveled between the farthest-apart transmitter and receiver, (2) transistor circuit latencies that depend on the complexity of the circuitry in transmitters, amplifiers, and receivers, which altogether continues to improve with technology scaling, (3) delays introduced by fundamental requirements of signal processing in detectors, (4) packet transmission latency that depends on the number of bits in the packet and the time needed to transmit one bit (Section III-A3), and finally (5) queuing and arbitration delays that we addressed in Sections IV-B and III-B respectively.

We now summarize the first three components of the latency: first for the transmitter, then for the TL and amplifiers on the ring, and finally for the receiver.

Transmitter delay consists of a single CMOS pass-gate and an amplifier (Figure 4) that connect the PLL to the transmit port of the TL coupler. The pass-gate delay is only a few picoseconds and improves with technology scaling. The amplifier latency is about 50 ps in 45 nm technology and scales with technology [41] (we conservatively skip the 32 nm technology node and assume 25 ps for 22 nm and 13 ps for 10 nm technology). Note that the amplifier is needed mainly for impedance matching if the PLL signal is strong enough. In our simulations, we find that this amplifier can be replaced by an inverter, with transistors sized for 50 Ω output impedance.

Propagation speed along our UTL is 0.0075 ns/mm (from the AWR [42] EM simulator). For a 16 mm×16 mm die with 64 cores, the ring is 156.4 mm long, with 16 amplifiers along the way. When the number of cores is increased to 256 (in 10 nm technology), the ring length is 286.4 mm with 32 (faster) amplifiers along the way. Hence, the propagation delay (transmitter output to receiver input) is expected to be 1.6 ns for a 64-core chip in 22 nm technology (156.4 mm×0.0075 ns/mm+16 amps×0.025 ns/amp), and 2.6 ns for a 256-core chip in 10 nm (286.4 mm×0.0075 ns/mm+32 amps×0.013 ns/amp).

At the receiver, the delay is dominated by the delay of a detector, which is a signal-processing delay determined by the frequency of the received RF signal. For the 40 GHz AM detector used in our design, the delay was 25 ps for one pulse. Since we use two pulses per bit, the second pulse can be used for robustness (at the cost of adding another 25ps to the delay), or it can simply be ignored.

### B. Disconnect-Based Arbitration Mechanism

With a separate arbitrator, the request/grant signals must be sent to/from the arbitrator by a separate set of TLs (which is expensive), or through long-latency wires (which leads to much longer overall latency for ring-steered packets). Instead, we use the ring itself for arbitration by adapting the Token Ring approach from nanophotonic broadcast rings [24]—the sender transmits a token down the ring when it is done, and the first prospective sender node that gets the token becomes the new sender. However, in nanophotonic interconnects, a dedicated wavelength is used for the token signal, whereas our ring with simple modulation uses all of its bandwidth for one channel. Therefore, we attach a token sequence at the end of the packet. The token sequence consists of a few disconnect-delay bits, a token bit that is initially equal to 1, and a few reconnect-delay bits. Each potential sender looks for the end of the current packet, disconnects the ring using its pass-gate and receives the token bit. If the token bit is zero, it reconnects the ring. This way, only the first potential sender will receive the token bit with a value of 1—the disconnect in our modulation scheme results in a 0 value (no pulses) continuing down the ring so other prospective senders receive 0 value for the arbitration bit. Our EM and circuit simulations show that disconnect and reconnect delays of only 2 bits are sufficient, so arbitration "spends" only five bits worth of throughput per packet.

### C. Broadcasts on the TL Ring

A packet sent on our TL ring is "seen" by all other cores, so we convert multicast packets (e.g., when sending out invalidations) into broadcasts. However, we use the baseline directory protocol for all coherence actions, even though a ring can be used as an ordered interconnect to support efficient snooping [43]. In our approach, most packets still use an unordered (mesh) interconnect, which would lead to ordering inconsistencies if both snoopy and the directory-based actions can occur on the same cache block. A solution for this is outside the scope of this paper.

## IV. Traffic Steering

The UTL ring presented in Section III provides low-latency and efficient arbitration, but its aggregate throughput is significantly lower than that of wire-based switched NoCs. Instead of boosting the ring throughput using multiple rings, sophisticated modulation and even higher signaling frequencies that cause major increase in cost and design complexity, we combine the low-latency-but-limited-throughput UTL ring with a traditional switched interconnect. This approach is based on an insight that the latency benefit from using the ring varies from message to message, so we can boost the benefit from the low-latency ring by using it only for packets that are expected to show a lot of benefit from the reduced latency while the remaining (majority of) packets continue to use the high-throughput switched interconnect.

Our steering approach has three components: (1) a scoring mechanism that predicts the benefit from ring-steering a given packet (Section IV-A), (2) a threshold management mechanism that controls ring throughput consumption (Section IV-B) and (3) an emergency re-steering mechanism for sudden rises in ring contention (Section IV-C).

### A. Benefit Estimation and Packet Scoring

Our scheme rates each packet and assigns a score according to how much benefit it can expect from being steered to the ring instead of the mesh. This score $S$ is a sum of two components: latency benefit $S_{lat}$ and criticality score $S_{crit}$. The latency benefit $S_{lat}$ is the difference in packet latency, which requires future knowledge and is thus estimated. So, $S_{lat} \approx L_{mesh} - L_{ring}$, where $L_{mesh}$ and $L_{ring}$ are the estimated latencies for the mesh and the ring, respectively.

Because the main latency factors in the ring and mesh are different, they have separate estimators. The mesh latency mainly depends on the packet's hop count and network congestion. Without congestion, latency is proportional to hop count. However, in our experiments, a 5-hop packet has latencies from 19 cycles (no contention) to 56 cycles (heavy contention), so it is important to account for contention when estimating $L_{mesh}$. For the TL ring, the range of packet latencies can be even larger without some form of demand control. Ring-steered packet latency can be as low as 4–6 cycles without contention (4 cycles to transmit a 64-bit packet at 16 Gbit/s, plus 0–2 cycles of propagation latency depending on destination). In our experiments with *lu-cn*, most packets do have reasonable latency (82.7% are below 20 cycles), but some (about 5%) have latencies above 100 (and up to 2600) cycles due to contention when we randomly steer packets to TL ring with 50% chance.

**Our mesh latency estimator** uses a small cache (*m-cache*) in each node to track latencies of recent packets with a given hop count. The *m-cache* has $s$ rows, one row for each possible hop count in the mesh. Each row stores $n$ most recently observed latencies for that hop count for messages sent by that node. In our experiments, $s = 14$ ($8 \times 8$ mesh for 64-cores) and $n = 4$. We store 8-bit values (255 represents all latencies $\geq$255), so the *m-cache* uses only 64 bytes on each core. We use three prediction strategies: (1) most recent latency, (2) average of latest $\frac{n}{2}$ latencies and (3) average of all $n$ latencies. All three use the same state (the *m-cache*), and they offer a tradeoff between quick adaptation and ability to filter outliers. Each node picks the best-performing strategy as $L_{mesh}$ for its score calculation, and tracks accuracy of each strategy with a saturating counter that is incremented by 2 when that strategy's prediction is the closest to actual latency and decremented by 1 otherwise. To do this, we add short timestamps to outgoing packets, compute latency at the destination node, and piggyback this information to a packet sent in the opposite direction.

**Our ring latency estimator** considers $l$, the contention-free latency for the packet, and two contention-related factors: 1) probability of finding the ring idle ($p_{idle}$) and 2) expected wait time if the ring is not idle ($t_{queue}$). The packet's overall latency $L_{ring}$ on the ring is estimated as $L_{ring} = l + t_{queue} \times (1 - p_{free})$. To estimate $p_{idle}$ and $t_{queue}$, for the most recent $K$ packets seen on the ring, each node records 1) the number of elapsed cycles since the previous packet, i.e. $t_k = T_k - T_{k-1}$ and 2) the distance ($d_k$) along the ring between the previous and the current transmitter. The probability of finding the ring idle is estimated using the history of elapsed time: over the last $K$ ring accesses the total elapsed time was $\sum_{i=1}^{K} t_k$ cycles while the ring was busy for $l \times K$ cycles. Therefore, the ring is expected to be free with a probability of $p_{free} = 1 - \frac{l \times K}{\sum_{i=1}^{K} t_i}$.

To estimate $t_{queue}$, we first estimate $p_{core}$, the probability that a core will use the ring if given the opportunity ("live" token with arbitration bit **1** passes by). Over the last $K$ packets, the number of cores that were given this opportunity was $\sum_{i=1}^{K} d_i$, but only $K$ of them used the ring. Thus we estimate $p_{core} = \frac{K}{\sum_{i=1}^{K} d_i}$. If the ring-distance between the current transmitter and this node is $k$, the expected number of packets that will be transmitted before this node can get the "live" token is $k \times p_{core}$, and the expected wait time for a packet on this node is $l \times k \times p_{core}$. Note that this assumes no packets are already queued for the ring at this node. If there are $w$ already-queued packets on this node, the first of those packets will be transmitted when this node gets the live token, and then for each subsequent packet the node will have to wait until every node around the ring gets the opportunity to transmit—if there are $N$ cores in the ring, each "circle" around the ring is expected to take $l$ cycles for our own packet (which we know will be sent) plus $l \times (N-1) \times p_{core}$ cycles for packets from other $N-1$ nodes. The overall waiting time for a packet is then $t_{queue} = l \times k \times p_{core} + w \times (l + l \times (N-1) \times p_{core})$.

**The criticality adjustment** $S_{crit}$ is used to account for known and/or estimated effects of the packet's latency on the actual performance. In principle, one can use instruction-level criticality estimator that attributes some of the message's latency to overall stall time and uses it for future messages from the same instruction. To simplify hardware, we use a simple approach that assigns a constant penalty to messages that are unlikely to be critical. Examples include write-back messages, speculative replies and their confirmations when various protocol optimizations are used.

### B. Threshold-Based Throughput Control

Note that the overall latency benefit of having the TL ring can be roughly estimated as the product of the number of ring-steered messages and the average latency benefit per message. Without any throughput control, too many packets would be steered to the ring under high-load conditions. An equilibrium would eventually be reached at a point where ring contention is so high that it offsets the latency advantage for most messages. The result would be that the ring still carries a small fraction of the traffic while provides very little latency advantage for that traffic. Thus, our throughput mechanism aims to maintain a ring utilization level that is low enough to preserve its low latency, but high enough not to waste the ring throughput. The "Goldilocks" utilization level depends on the distribution of mesh latencies in the application, efficiency of arbitration on the ring and many other factors. However, we found that target utilizations around 75% tend to work well for our ring—it keeps queuing delay at a low level, but still utilizes a significant portion of the available throughput of the ring.

Our approach to controlling utilization of the ring is to maintain a threshold for scores used in steering decisions. Instead of steering packets based on whether its expected-benefit score is positive or negative, we now steer a packet only when its score exceeds the threshold. To control ring utilization, we adjust the threshold upward when the ring is busier than desired, or decrease the threshold if the ring utilization is too low.

The ring utilization is measured locally at each core using a TL ring access counter. Each core increments the counter when a packet is observed. For every $T$ cycles, the counter is destructively read and ring utilization is calculated as $\frac{l \times n}{T}$, where the counter value is $n$ and the TL ring propagation latency is $l$. In our implementation, we update utilization every 512 cycles ($T = 512$), and increment or decrement the threshold by 1 if the utilization is above or below a preset target utilization.

Finally, we note that high ring utilization also creates significant priority inversion problems—a high-scored message on one core waits for other cores' lower-scored messages that are sent on the ring already. This priority-inversion problem can get resolved with score-aware arbitration at a cost of more ring throughput wasted on arbitration. However, when ring utilization is lower (e.g. below 85%) such priority inversion is rare, so score-aware arbitration is counter-productive—the penalty from larger arbitration sequences is higher than the benefit from eliminating priority inversion.

### C. Emergency Re-steering

Our steering mechanism is based on estimated latencies. If the actual latency of TL-steered packet is higher than estimated, e.g. due to a sudden burst of ring-steered messages, it may take a while for our throughput control to compensate. During that time, ring-steered messages may experience very long latencies. To avoid this, we implement an emergency re-steering mechanism, which periodically checks entries in the local TL packet queue and resteers a packet to the mesh if it has waited too long. To find such packets without keeping much state, we use a single bit in each TL packet queue entry. We set this bit to 0 when the packet is enqueued. Periodically, we check these bits. If the bit is 0, we set it to 1. If the bit is already set, we resteer the packet. With this scheme, if checks are performed every $T$ cycles, the maximum penalty from ring-steering a packet is limited to $2 \times T$.

## V. Implementation Cost

Our transmission-line (TL) ring has two main sources of cost—the use of metal layers for the coupler-based TL ring, and the use of silicon area for active components of the design.

| Active Element | Area ($mm^2$) | Power (mW) | # Used | Total Area ($mm^2$) | Total Power (W) |
|---|---|---|---|---|---|
| Amplifier [38] | 0.017 | 28 | 16 | 0.272 | 0.448 |
| Detector [39] | 0.00024 | 0.84 | 64 | 0.015 | 0.054 |
| Total | – | – | – | 0.287 | 0.502 |

**TABLE I:** The cost of main active components used in TL ring for 64-core processor in 22 nm technology.

In terms of metal area, a coupler-based TL should be implemented in the top two (global) metal layers. However, its "ground plane" should be in the lowest global metal layer, and the layers in-between should be clear of metal and cannot be used for any local circuitry. Hence, the coupler-based TL uses all global metal layers. However, switches are needed only at connection points between couplers, and amplifiers are required at every eighth connection point. This use of switches and amplifiers at millimeter-scale distances allows our TL ring to be placed-and-routed without affecting the layout of circuitry within each core.

The main drawback of using transmission lines is in their large width—a microstrip coupler implementation occupies a 20 $\mu$m-wide area in two metal layers. The coupler consists of two strips, each only 4 $\mu$m wide, but to achieve the desired amount of coupling, the spacing between the strips has to be about 10 $\mu$m. Furthermore, other wires or transmission lines cannot go between its "ground plane" conductors without creating significant crosstalk. This metal use is significantly larger than traditional global wires that have only a 0.135 $\mu$m pitch in 45 nm technology with proportional scaling in future technologies. This difference in metal area consumption reinforces the argument in favor of *combining* a fast TL based ring with the traditional switched interconnects, rather than trying to replace a traditional high-bandwidth interconnect with a much more expensive TL-based interconnect.

A conservative estimate of the total metal area necessary for our TL ring with 64 processors in 22 nm technology is 31.28 $mm^2$—two coupled lines (transmitter, receiver and the next coupler) for each coupler, 156.4 mm total length of the TL ring, 20 $\mu$m total width in 10 layers for a total of 156.4 mm$\times$0.020 mm$\times$10. Although this appears expensive, it should be noted that it represents only 1.22% of the total metal area for a 16 mm$\times$16 mm chip with 10 metal layers. Furthermore, the only requirement for "ground plane" conductor is that it should not carry RF signals, so this wide conductor can be leveraged for Vdd or Vss distribution.

The main active components are the amplifiers between transmission line segments and amplitude detector in each receiver (Table I). For amplifier design, we model a low-noise amplifier (LNA) that can be implemented in a CMOS process [38]. In the original 45 nm implementation, this amplifier occupies 0.017 $mm^2$ of chip area. For the detector, we model a compact CMOS-based design [39]. In the original 90 nm technology, this detector only occupies 0.00024 $mm^2$ of chip area, and can be expected to scale well because it entirely consists of transistors and very small capacitors and inductors.

Finally, we note that the design of transmission lines can pose some restrictions on the traditional use of metal layers, and may result in sub-optimal placement of non-TL component such as mesh links in metal layers. However, such elements are less affected by physical location and shape, and can often be relocated with negligible impact on performance and metal budget. Finally. our TL-coupler design operates at much higher frequencies (e.g. 20–60 GHz) and filters out everything below 20 GHz, so traditional signals (which are below 20 GHz) will not interfere with the TL design and layout.

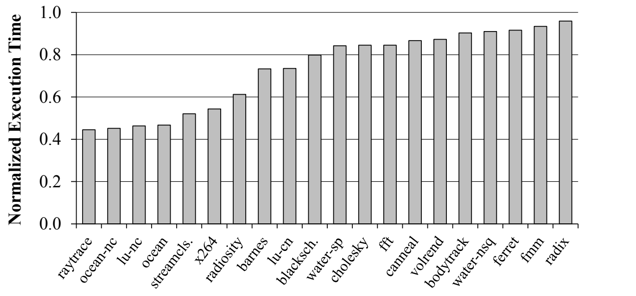

**Fig. 5:** Normalized execution time with infinite throughput on-chip interconnect (single cycle latency).

## VI. Performance Evaluation

For performance evaluation, we implement a detailed model of our TL ring's message and arbitration timing as well as its traffic steering in SESC [44], an open-source cycle-accurate architectural simulator. Our baseline system is a 64-tile CMP system. A tile contains a 2-issue out-of-order core at 1 GHz, a 32 KB instruction L1 cache, a 32 KB L1 data cache, a 1 MB slice of the L2 cache, a directory slice and a mesh router. The latency of an off-chip memory access is 200 cycle, with memory controllers located in each corner of the chip. The packet-switched interconnects are modeled in detail using the cycle accurate Booksim simulator [45], which we integrated into SESC. Our evaluation is based on 8$\times$8 2D mesh with 128 bit channel with one cycle delay. We model 8 virtual channels (24 buffers) per router port and packets are routed using Dimension-Order Routing (DOR).

We evaluate our scheme with parallel applications from SPLASH-2 [46] and PARSEC 3 [47] benchmark suites. For SPLASH-2, enlarged input set sizes are used to achieve execution of at least 1 billion instructions. For PARSEC, we use sim-medium inputs. We measure the execution time only in the parallel section as intended by the benchmark designers, denoted as region of interest.

Not all 20 applications in SPLASH-2 and PARSEC can benefit from better on-chip interconnects. Figure 5 shows results with an ideal interconnect that transmits every packet in a single cycle with no contention, and in half of the applications this only improves overall performance by $<$20%. These applications either suffer from high rates of off-chip memory access (e.g., *fft* and *radix* show 98.7% and 85.0% of accesses go off-chip), or exploit memory- or instruction-level parallelism to hide on-chip access latencies. In the remainder of our evaluation, we focus on applications whose performance is sensitive to NoC characteristics. We thus use only applications that have at least a 20% execution time reduction with ideal interconnect, called *latency-critical applications*.

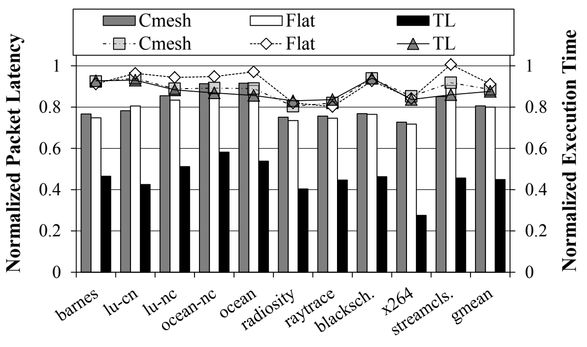

**Fig. 6:** Normalized packet latency to the baseline mesh (left axis) of concentrated mesh (Cmesh), flattened butterfly (Flat) and TL ring with execution time (right axis).

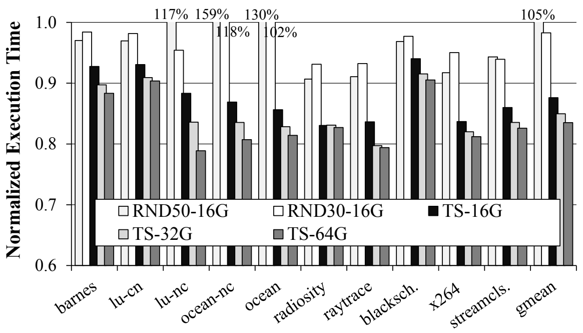

**Fig. 7:** Normalized execution time of traffic steering (TS) with varying TL ring bandwidth, random (RND).

## *A. Impact of Latency Reduction with TL*

Signal propagation speed in a our TL ring is much faster than traditional on-chip wire. As a result, ring-steered packets on average have 55% lower latency than the packets in the baseline mesh interconnect despite serial transmission of bits on a single transmission line and some delays (Figure 6). Furthermore, TL ring provides lower latency when compared with advanced interconnects: ring-steered packets show 44.3% and 43.9% lower latency over concentrated mesh (*Cmesh*) and flattened butterfly (*Flat*), respectively. Note that Cmesh and Flat provide some latency reduction for all long-distance packets through a more complex topology, which includes long links. In contrast, the TL ring provides much lower latency but only for 13% to 44% of the on-chip messages and 2.0% to 9.9% of the traffic (bits), depending on the application. The overall average packet latency for *ring+mesh* is thus slightly higher than for *Cmesh* or *Flat*.

However, in terms of execution time, our ring+mesh combination outperforms both advanced switched interconnects. Ring+mesh provides a 12.4% execution time reduction, compared to 11.5% and 8.9% reduction achieved by using *Cmesh* and *Flat*. This is because our steering scheme concentrates the ring's large packet latency reduction on packets that benefit the most. Furthermore, our *ring+X* approach can be gainfully applied to advanced switched interconnects, i.e. a ring+*Cmesh* or ring+*Flat* combination provides a performance benefit beyond that provided by *Cmesh* or *Flat* alone (Section VI-D).

## *B. Latency Estimation and Re-Steering*

Our steering is based on latency estimation of mesh and TL ring. Our estimation of mesh latency has $<$30% error for $>$80% of packets in all applications, except in *streamcluster* where "only" 72% of packets have $<$30% latency estimation error. For TL latency estimation, we find that 80% of packets can be estimated within 6 cycles of actual latency, except in *ocean* and *ocean-nc* where "only" about 75% of packets are estimated within 6 cycles of actual latency. Note that our steering scheme uses a threshold that specifies the minimum allowed score (i.e., gap between estimated TL and mesh latency), so small estimation errors are unlikely to change the steering decision. Only a small fraction of packets experience high estimation error and are steered to a sub-optimal network, with negligible impact on overall performance.

Re-steering is also rare—in our experiments, only 0.27% of packets are re-steered on average. The highest re-steering (in *streamcluster*) is only 2.3%. Recall that these packets wait in the TL queue first, and are then sent to the mesh, so they suffer a higher latency than they would in the mesh-alone baseline. This penalty can be controlled by how often we check ring-enqueued packets for possible re-steering (period $T$ in Section IV-C). If $T$ is too short, packets that could have benefited from the ring are send to the mesh after already suffering a latency penalty while waiting for the ring. If $T$ is too long, the ring can suffer longer periods of saturation (and high overall latency) during certain message traffic changes. In our experiments, the re-steering period does change how many packets are re-steered, but even with short re-steering periods (10 cycles) the number of re-steered packets is still relatively low (1.7% on average) and the impact on execution time is also marginal (0.3% worse than with the optimal $T$). Interestingly, our results show that the optimal re-steering period (24 cycles) is very close to the average packet latency in the mesh.

## *C. Effect of Ring Bandwidth and Steering*

Figure 7 shows application execution times with a mesh+ring, normalized to the mesh-only baseline. To evaluate the effectiveness of our traffic steering scheme (shown as *TS*), we also show *RND50* and *RND30* steering, which randomly steer packets to the ring with 50% and 30% probability, respectively. These non-adaptive random steering approaches provide limited execution time reduction for some applications, mostly those that have less message traffic (e.g., *radiosity*). However, for applications like *ocean-nc*, where on-chip traffic can be significant, sending 50% of packets to TL ring causes too much contention, which results in significant slowdown (59% execution time increase in *ocean-nc*), and even 30% of the traffic proves too much in some applications. On average, *RND30* provides only a 1.7% execution time reduction, and *RND50* results in a 5% execution time *increase*. In some lower-traffic applications, *RND50* provides better performance than *RND30*, because it allows more of the (light) traffic to benefit from the ring's low latency. However, even in these applications, our adaptive steering (*TS*) shows significantly better performance—partly because it allows nearly all traffic to benefit from the TL when traffic is very light, and partly because it is more judicious in selecting which messages to ring-steer during higher-traffic periods. Finally, we tried other simple (non-adaptive) policies, e.g. steering all short packets

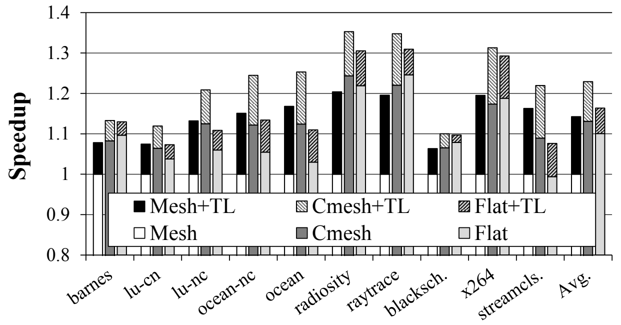

**Fig. 8:** Effect of combining a 16GB/s TL ring with a mesh, a concentrated mesh (Cmesh), and a flattened butterfly (Flat).

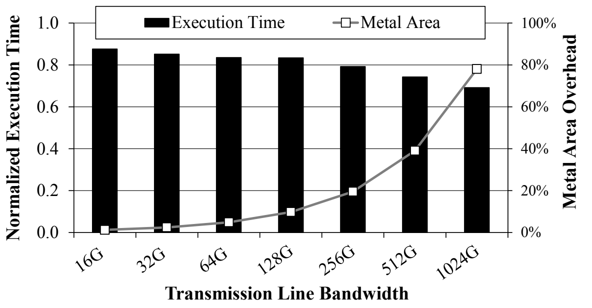

**Fig. 9:** Normalized execution times (averaged for all benchmarks) for configurations with various TL ring bandwidth.

to the ring. These policies still significantly underperform our adaptive steering, for similar (lack of adaptation) reasons.

### *D. Effect of Advanced Switched Networks*

In our combined interconnect, the switched interconnect handles most of the on-chip traffic. While we have used a mesh as a baseline switched interconnect, our approach can be applied to any switched interconnect where there is a significant variation in packet latency depending on the destination. To confirm this, we evaluate our approach using a concentrated mesh (*Cmesh*) and a flattened butterfly (*Flat*) as the baseline. The *Cmesh* we use in our experiments is 4-concentrated (4 cores per switch) and in *Flat*, we halved the link width. These results are shown in Figure 8. Note that the speedups are based on the baseline mesh result. We find that our TL ring provides a 8.7% improvement in speedup (1.23×) over the *Cmesh* (1.13×), and 5.7% improvement (1.16×) over *Flat* (1.10×). These results show that the benefits of adding our TL ring are not entirely additive with the benefits of using a more advanced switched interconnect. However, it should be noted that the flattened butterfly is one of the best-performing switched networks that is still suitable for on-chip implementation, so a 5.7% performance improvement in exchange for 1.2% of the chip's metal area may be a good tradeoff that would be difficult to match by further improvements in switched network topology.

### *E. Sensitivity to TL Ring Bandwidth*

Figure 9 shows average execution time reduction as TL ring bandwidth increases, e.g. by using more than one such ring or more sophisticated signaling. We observe diminishing marginal benefit from each doubling of TL bandwidth, while the costs increase dramatically: a 16 Gb/s TL provides 12.4% execution time reduction; 32 Gb/s yields an additional 2.6% execution time reduction but at 2× the cost, and 64 Gb/s yields only another 1.5% improvement but doubles the cost again, etc. This is because our steering chooses those messages that get the most latency reduction per ring-steered bit, so each addition to the ring's bandwidth allows it to also carry "second-best" packets in addition to the "very-best" ones. This leads to a decreasing cost-benefit trend, which confirms our initial insight that a limited-throughput TL interconnect can provide most of the benefit that would be obtained from a higher-throughput (but proportionally more expensive) TL interconnect.

### *F. Non-Latency-Critical Applications*

Although our evaluation is focused on *latency-critical* applications, our scheme provides some performance improvement (2.9% on average) for *non-latency-critical* applications, and does not hurt performance in any applications.

### *G. Power Benefit*

The main sources of power consumption at TL ring are active components (Table I). Without any power-hungry switches/routers, the overhead is near constant regardless of the load on the TL ring (0.5W). Therefore, the overall power benefit is equal to the reduction in power consumption on baseline NoC due to the reduced traffic (5.0% on average), minus the (nearly constant) TL ring power overhead.

## VII. Conclusion

This paper describes an approach where a low-latency unswitched interconnect is synergistically used with a high-throughput switched interconnect in a many-core system. We designed and introduced a ring of transmission line coupler structures with low latency and low throughput. We combined the structures with a low-latency arbitration mechanism that allows efficient use of this ring by many cores. Then, we create a new adaptive packet steering policy to judiciously use the limited throughput available on the low-latency interconnect. Our experiments show that the ring+mesh combination adds only 1.3% to the chip area, but provides a 12.4% execution time reduction for parallel applications on average, compared to the mesh alone.

## Acknowledgment

This work was supported in part by the National Science Foundation under grants 0903470 and 0964647. Any opinions, findings, and conclusions or recommendations in this paper are ours alone, and do not necessarily reflect the views of NSF.

## References

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip inteconnection networks," in *Proceedings of the 38th Annual Design Automation Conference*, 2001, pp. 684–689.

[2] C. Grecu, P. P. Pande, A. Ivanov, and R. Saleh, "Structured interconnect architecture: a solution for the non-scalability of bus-based SoCs," in *Proc. 14th ACM Great Lakes Symposium on VLSI*, 2004, pp. 192–195.

[3] R. Lu, A. Cao, and C.-K. Koh, "SAMBA-Bus: A high performance bus architecture for system-on-chips," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 1, pp. 69–79, 2007.

[4] C. Fallin, X. Yu, G. Nazario, and O. Mutlu, "A high-performance hierarchical ring on-chip interconnect with low-cost routers," *SAFARI Technical Report No. 2011-007*, Sep. 2011.

[5] International Technology Roadmap for Semiconductors, "ITRS - 2009 edition," *http://www.itrs.net*, 2009.

[6] N. Srivastava and K. Banerjee, "Performance analysis of carbon nanotube interconnects for VLSI applications," in *Proceedings of Int'l Conference on Computer Aided Design*, Nov. 2005.

[7] P. Wolkotte, G. Smit, N. Kavaldjiev, J. Becker, and J. Becker, "Energy model of Networks-on-Chip and a Bus," in *Proceedings of Int'l Symposium on System-on-Chip*, 2005, pp. 82–85.

[8] M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, vol. 17, pp. 34–39, 1997.

[9] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proceedings of the 31st Annual Int'l Symposium on Computer Architecture*, 2004, pp. 188–197.

[10] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proc. 20th Int'l Conference on Supercomputing*, 2006, pp. 187–198.

[11] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *Proc. 40th Int'l Symposium on Microarchitecture*, 2007, pp. 172–182.

[12] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: towards the ideal interconnection fabric," in *Proc. 34th Int'l Symposium on Computer Architecture*, 2007, pp. 150–161.

[13] B. Grot, J. Hestness, S. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *Proceedings of the 15th Int'l Symposium on High Performance Computer Architecture*, 2009, pp. 163–174.

[14] R. E. Collins, *Field Theory of Guided Waves*. McGraw-Hill, 1960.

[15] S. Bourduas and Z. Zilic, "A hybrid ring/mesh interconnect for network-on-chip using hierarchical rings for global routing," in *Proceedings of the First Int'l Symposium on Networks-on-Chip*, 2007, pp. 195–204.

[16] 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 *Proceedings of the 15th Int'l Symposium on High Performance Computer Architecture*, 2009, pp. 175–186.

[17] K.-L. Tsai, F. Lai, C.-Y. Pan, D.-S. Xiao, H.-J. Tan, and H.-C. Lee, "Design of low latency on-chip communication based on hybrid NoC architecture," in *8th IEEE Int'l NEWCAS Conference*, 2010, pp. 257–260.

[18] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in Multi-Core Architectures: Understanding mechanisms, overheads and scaling," in *Proc. 32nd Int'l Symposium on Computer Architecture*, 2005, pp. 408–419.

[19] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian, "Towards scalable, energy-efficient, bus-based on-chip networks," in *Proc. 16th Int'l Symposium on High Performance Computer Architecture*, 2010, pp. 1–12.

[20] M. Hayenga and M. Lipasti, "The NoX router," in *Proc. 44th Int'l Symposium on Microarchitecture*, 2011, pp. 36–46.

[21] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proceedings of the 17th Int'l Symposium on High Performance Computer Architecture*, 2011, pp. 144–155.

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

[23] C. R. Johns and D. A. Brokenshire, "Introduction to the cell broadband engine architecture," *IBM J. Res. Dev.*, vol. 51, no. 5, pp. 503–519, 2007.

[24] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System implications of emerging nanophotonic technology," in *Proc. 35th Int'l Symposium on Computer Architecture*, 2008, pp. 153–164.

[25] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. B. Carter, "Interconnect-aware coherence protocols for chip multiprocessors," in *Proc. 33rd Int'l Symposium on Computer Architecture*, 2006, pp. 339–351.

[26] J. L. Abellán, J. Fernández, and M. E. Acacio, "Efficient and scalable barrier synchronization for many-core cmps," in *Proceedings of the 7th ACM Intl. Conf. on Computing frontiers*, 2010, pp. 73–74.

[27] J. Oh, M. Prvulovic, and A. Zajic, "TLSync: support for multiple fast barriers using on-chip transmission lines," in *Proceeding of the 38th Annual Int'l Symposium on Computer Architecture*, 2011, pp. 105–116.

[28] M.-C. F. Chang, J. Cong, A. Kaplan, C. Liu, M. Naik, J. Premkumar, G. Reinman, E. Socher, and S.-W. Tam, "Power reduction of CMP communication networks via RF-interconnects," in *Proceedings of the 41st Annual Int'l Symposium on Microarchitecture*, 2008, pp. 376–387.

[29] T. Krishna, A. Kumar, P. Chiang, M. Erez, and L.-S. Peh, "NoC with near-ideal express virtual channels using global-line communication," *Symposium on High-Performance Interconnects*, pp. 11–20, 2008.

[30] B. M. Beckmann and D. A. Wood, "TLC: Transmission Line Caches," in *Proceedings of the 36th Annual IEEE/ACM Int'l Symposium on Microarchitecture*, 2003, pp. 43–54.

[31] A. Carpenter, J. Hu, J. Xu, M. Huang, and H. Wu, "A case for globally shared-medium on-chip interconnect," in *Proceedings of the 38th Annual Int'l Symposium on Computer Architecture*, 2011.

[32] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi, "Phastlane: a rapid transit optical routing network," in *Proceedings of the 36th Annual Int'l Symposium on Computer Architecture*, 2009, pp. 441–450.

[33] N. Kirman and J. F. Martínez, "A power-efficient all-optical on-chip interconnect using wavelength-based oblivious routing," in *Proceedings of the fifteenth Int'l Conference on Architectural Support for Programming Languages and Operating Systems*, 2010, pp. 15–28.

[34] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," in *Proc. 36th Int'l Symposium on Computer Architecture*, 2009, pp. 429–440.

[35] A. Ganguly, K. Chang, S. Deb, P. Pande, B. Belzer, and C. Teuscher, "Scalable hybrid wireless network-on-chip architectures for multicore systems," *IEEE Transactions on Computers*, vol. 60, no. 10, pp. 1485–1502, 2011.

[36] E. Yablonovitch, "Can nano-photonic silicon circuits become an intra-chip interconnect technology?" in *Proceedings of the 2007 IEEE/ACM Int'l Conference on Computer Aided Design*, 2007, pp. 309–309.

[37] A. Carpenter, J. Hu, O. Kocabas, M. Huang, and H. Wu, "Enhancing effective throughput for transmission line-based bus," in *Proc. 39th Int'l Symposium on Computer Architecture*, 2012, pp. 165–176.

[38] J. Borremans, S. Thijs, M. Dehan, A. Mercha, and P. Wambacq, "Low-cost feedback-enabled LNAs in 45nm CMOS," in *Proceedings of ESSCIRC*, 2009, pp. 100–103.

[39] A. Oncut, B. Badalawa, and M. Fujishima, "60GHz-pulse detector based on CMOS nonlinear amplifier," in *IEEE Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems (SiRF)*, Jan. 2009.

[40] A. R. Djordjević, V. M. Napijalo, D. I. Olćan, and A. G. Zajić, "Wideband multilayer directional coupler with tight coupling and high directivity," *Microwave and Optical Technology Letters*, vol. 54, no. 10, pp. 2261–2267, 2012.

[41] M. Sinha, S. Hsu, A. Alvandpour, W. Burleson, R. Krishnamurthy, and S. Borkar, "High-performance and low-voltage sense-amplifier techniques for sub-90nm SRAM," in *Proceedings of the IEEE Int'l Conference on Systems-on-Chip*, 2003, pp. 113–116.

[42] AWR, "Applied Wave Research," El Segundo, CA, USA, 2010.

[43] M. R. Marty and M. D. Hill, "Coherence ordering for ring-based chip multiprocessors," in *Proceedings of the 39th Annual IEEE/ACM Int'l Symposium on Microarchitecture*, 2006, pp. 309–320.

[44] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, "SESC simulator," 2005, http://sesc.sourceforge.net.

[45] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[46] S. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. 22th Int'l Symposium on Computer Architecture*, 1995.

[47] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.