TLSync: Support for Multiple Fast Barriers Using On-Chip Transmission Lines

Jungju Oh
jungju@gatech.edu

Milos Prvulovic
milos@cc.gatech.edu

Alenka Zajic
azajic@cc.gatech.edu

Georgia Institute of Technology
Atlanta, GA, USA

ABSTRACT

As the number of cores on a single-chip grows, scalable barrier synchronization becomes increasingly difficult to implement. In software implementations, such as the tournament barrier, a larger number of cores results in a longer latency for each round and a larger number of rounds. Hardware barrier implementations require significant dedicated wiring, e.g., using a reduction (arrival) tree and a notification (release) tree, and multiple instances of this wiring are needed to support multiple barriers (e.g., when concurrently executing multiple parallel applications).

This paper presents TLSync, a novel hardware barrier implementation that uses the high-frequency part of the spectrum in a transmission-line broadcast network, thus leaving the transmission line network free for non-modulated (baseband) data transmission. In contrast to other implementations of hardware barriers, TLSync allows multiple thread groups to each have its own barrier. This is accomplished by allocating different bands in the radio-frequency spectrum to different groups. Our circuit-level and electromagnetic models show that the worst-case latency for a TLSync barrier is 4ns to 10ns, depending on the size of the frequency band allocated to each group, and our cycle-accurate architectural simulations show that low-latency TLSync barriers provide significant performance and scalability benefits to barrier-intensive applications.

Categories and Subject Descriptors

C.1.2 [**PROCESSOR ARCHITECTURES**]: Multiprocessors—*Interconnection architectures*

General Terms

Design, Performance

Keywords

Multi-core, Synchronization, Barrier, Transmission Line

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISCA'11, June 4–8, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0472-6/11/06 ...$10.00.

1. INTRODUCTION

With growing numbers of cores per chip and increasing wire latencies, barrier synchronization becomes increasingly challenging. Simplistic software-only implementations suffer from serialization in counting arrivals, so their latency increases in proportion to the number of cores. Hierarchical barriers reduce this contention, but require multiple rounds of point-to-point synchronization and an inefficient broadcast via a shared memory location. Hardware barrier networks provide lower latency but require a dedicated chip-spanning network whose latency has a very unfavorable technology-scaling trend. Worse, future many-core chips are likely to execute several parallel applications, virtual machines, etc., each of which may need a separate barrier network to synchronize its threads. Because the number of barrier networks (and their total cost) is a design-time decision, designers must decide between a) providing multiple expensive barrier networks that may be unused and b) not providing enough of them and potentially forcing applications to fall back to a much slower software barrier implementation.

Transmission lines have been proposed as a low-latency solution for on-chip interconnects, e.g. for express links in the on-chip network [10, 17] or for connecting distant banks in large caches [5]. In addition to providing much lower signaling latencies than traditional wires, transmission lines can be used to transmit multiple modulated signals at different RF frequencies. These advantages are tempered by increased cost and significant signal processing delays for demodulation and decoding of RF signals.

In this paper, we exploit the advantages of transmission lines to address the latency and multiple-barrier support issues faced by traditional barrier networks. Our hardware barrier support, which we call TLSync, allocates a relatively small part of the transmission line's available spectrum to each barrier group, allowing a single chip-spanning transmission-line network to support many (tens) of barriers simultaneously. We minimize signal processing latencies by not relying on data transmission in our barrier implementation. Instead, a TLSync barrier uses a simple signal presence test to determine when all cores in the group have arrived to the barrier. In essence, a TLSync barrier performs a wired-AND operation using the presence/absence of a "tone" at the barrier's allocated frequency to replace the traditional signaling using high and low voltage level on a wire. Each core transmits into the transmission line a low-power "tone" at its barrier's allocated frequency and stops transmitting when it finally arrives to the barrier. The receiver in each core is a simple tone detector tuned to the barrier's frequency, and a

core can exit the barrier when this receiver no longer detects a "tone" at the barrier's frequency.

To obtain accurate latency estimates for the proposed TLSync mechanism, we also provide a careful analysis of how the underlying chip-spanning transmission-line network can be designed. This analysis resulted in several important insights into the tradeoffs that are faced when using transmission lines (TLs) for on-chip transmission of RF signals. Our key observation in this regard is that the end-to-end latency of RF transmission is dominated by fundamental signal processing delays that must be accounted for, and that careful modeling of TLs is needed to account for severe signal weakening and degradation when multiple transmitters and/or receivers are connected along one transmission line.

The rest of the paper is organized as follows. Section 2 describes the background and related work, Section 3 describes our TLSync scheme, Section 4 provides more detail and analysis of the implementation of transmission-line networks and transmitters and receivers used for TLSync, and Section 5 presents our evaluation in terms of performance and area cost. Finally, Section 6 presents our conclusions.

2. BACKGROUND AND RELATED WORK

Researchers have long acknowledged the importance of efficient barrier mechanisms in highly-parallel systems, and many past supercomputer or massively parallel multicomputer systems had dedicated fast barrier hardware. Examples include a combining omega-switch network for fast fetch-and-add (barrier counter increment) in the NYU Ultracomputer [13], a single-stage combining network [14], and a dedicated bus for test-and-set messages in Sequent systems [4], dedicated networks in CM-5 [18] and Cray T3D [11], router extensions to support a virtual barrier tree in Cray T3E [26], and "global interrupt network" wired-OR in Blue Gene/L [3]. In the chip-multiprocessor arena, the MIT Multi-ALU processor [16] has a single-cycle barrier instruction implemented using global wires (that still had single-cycle latencies at the time). These dedicated barrier networks can only support one barrier at a time, whereas future many-core systems will likely execute several (potentially many) multi-threaded applications, each with its own barrier synchronization. Virtual barrier networks, such as the one in T3E, possibly bolstered with additional express links [24], are the exception to this—they can provide multiple virtual barrier trees, albeit at a cost of increased contention and latency. Beckmann and Polychronopolous [6] proposed a dedicated global interconnect for barriers that uses a register with zero-detect logic. However, the extension of this design for multiple concurrent barriers replicates much of the cost for each supported barrier, an approach that does not scale to large number of cores that may need many (ten or more) concurrent barriers. Sampson et al. introduce Barrier Filter [23], a lightweight barrier mechanism that uses cache invalidation as a barrier arrival signal and starves the cache fill until all threads arrive to the barrier. Unfortunately, the invalidation traffic to the centralized filter (which determines when all cores have arrived) can be a problem in many-core processors.

For future many-core chips, all these barrier networks also face the problem of increasing latencies for global wires. A barrier network spans the entire chip, so the wire length traversed by arrival and notification signals is expected to slowly increase as the total chip size increases [15], while the wire latency per unit distance quickly gets worse [15].

Transmission lines (TLs) have been proposed as a low-latency alternative to traditional global wires. The long delays in traditional wires are a result of changing the potential of the entire wire, whose capacitance is proportional to its length. Future technology scaling trends [15] are expected to significantly increase these already long per-millimeter delays. In contrast, TLs propagate signals as electromagnetic waves, at speeds that are close to the speed of light in the conductor material. This allows signals to cross the entire chip with sub-nanosecond propagation latencies, at least an order of magnitude faster than with traditional wires. This makes TLs very attractive for on-chip communication. However, TLs have several constraints that limit their usability: 1) they are more expensive than traditional wires because they are much wider (often by an order of magnitude or more) and occupy several (two or three) metal layers, 2) design of TLs requires much more modeling and planning because high frequency signals are significantly affected by the TL's shape (e.g., bends and turns) and implementation (e.g., material and thickness of the metal and insulator), 3) TLs require more sophisticated circuitry to transmit and receive their signals—instead of directly connecting logic gate inputs to a wire, TL receivers range from sense amplifiers to complete radio-frequency (RF) demodulation and decoding, depending on the type of the signal that is being used, and 4) data rates that can be achieved with relatively simple receivers (e.g., sense amplifiers) are limited.

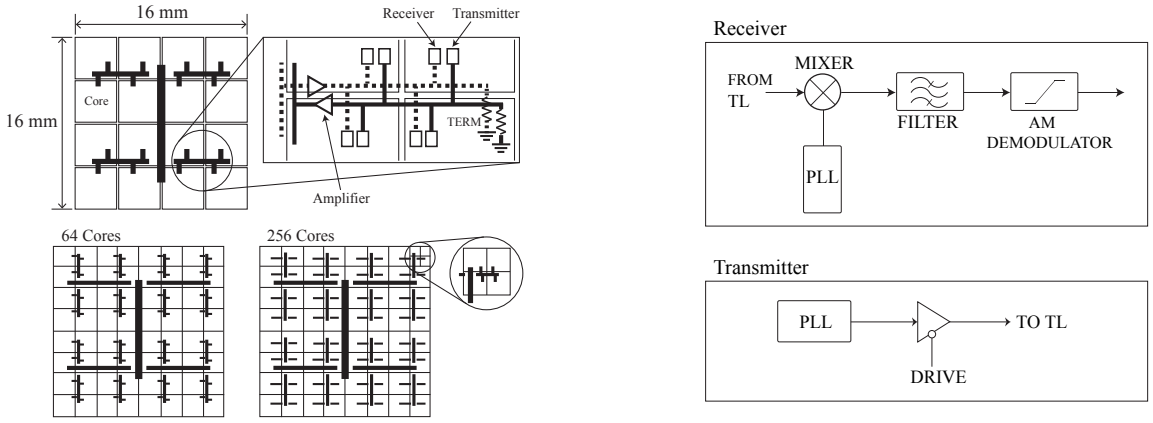
As a result of these considerations, TLs are not likely to entirely replace global wires. Instead, TLs are likely to be used strategically, e.g. to provide low-latency global connections for cache banks in very large caches [5], "express" long-distance network-on-chip (NoC) links [10, 17], or as a chip-wide frequency-multiplexed bus [9].

Recently, a global network of TLs with baseband (non-modulated) signals has been proposed for barrier synchronization [1]. This hardware barrier has a lower latency than one built out of traditional wires. However, this scheme can still support only one barrier group at a time. Considering the large cost of TLs, it is unlikely that future chips will include multiple such global networks in order to support multiple barriers.

3. DESIGN OF TLSYNC

We propose a novel hardware barrier support which uses high-frequency RF signals for barriers, while still leaving the transmission line (TL) usable for baseband and low-frequency RF data transmission. This barrier, which we call TLSync, supports multiple simultaneous barriers by allocating a separate RF frequency band to each barrier, and uses very simple modulation to minimize cost and demodulation latencies. Finally, TLSync accommodates the fact that weakening and distortion in TLs get worse as the frequency of the signals grows—the high-frequency RF bands we use are unlikely to be of practical use for data transmission, but are good enough for the simpler signals used for our barriers.

Our TLSync barrier support is conceptually very similar to traditional wired-OR [3], wired-NOR [27], and wired-AND approaches, which differ mainly in the interpretation of voltage levels (high/low vs. 1/0). A traditional wired-AND barrier would have a single chip-spanning wire, which



(a) Chip-Spanning Transmission-Line Network

(b) Receiver/Transmitter block diagram

Figure 1: Overall design of the TL network and TLSync transmitters and receivers.

would be connected to the supply voltage (logical 1) with a pull-up resistor and to each core by pull-down drivers. These drivers keep the wire at low (logical 0) voltage in spite of the current passing through the pull-up resistor. A core that has not yet arrived to the barrier would keep its pull-down driver active, thus keeping the wire at a low voltage level. When the core reaches the barrier, it simply deactivates its pull-down driver and starts monitoring the wire. When the last core has arrived, no pull-down drivers remain active and the pull-up resistor brings the wire's potential to a high value, which signals to all cores that the barrier is complete.

Unfortunately, an actual chip-spanning wired-AND circuit would have a huge latency because the wire has a large capacitance. For example, in the 45nm technology, a wired-and circuit for 16 4mm × 4mm cores (for a total chip size of 16mm × 16mm) would have a 34.7ns delay[1]—nearly 100 cycles for a 3GHz core. This latency grows quickly as technology scales—in 32nm technology, 32 such cores would fit on the same die area and the chip-spanning wired-AND would have a 100ns delay[2], and in 22nm technology the wired-OR for 64 cores would have a 312ns latency [3].

The latency of a traditional wired-AND circuit can be improved using a reduction tree for the AND operation and then a notification tree to distribute the final result to all the cores. With this approach, the chip-spanning barrier network would have 8.7ns, 23.4ns, and 59.4ns latency for the 45nm/16-core, 32nm/32-core, and 22nm/64-core chips used in the previous example. This latency can be further improved by using optimal repeater spacing along each wire segment—the delay drops to only 1.3ns, 2.4ns, and 3.4ns for the same three chips, but at a cost of using silicon (for repeaters) at frequent intervals along the wire's path. This use of silicon also interferes with the layout of the devices (cores, caches, etc.) along the path of the wire.

Even with the repeated-wire approach, the technology scaling trend is unfavorable and each reduction/notification barrier tree only provides support for a single barrier (i.e., only one parallel application). If there are multiple groups of threads that synchronize on independent barriers (e.g., several multi-core applications running concurrently), separate hardware barrier support would be needed for each group.

Our TLSync barrier retains the conceptual simplicity of wired-AND circuits and their main advantage—there is no contention between arriving cores, so when the last core arrives to the barrier, all cores are released after a fixed latency, even if all cores arrive nearly simultaneously. However, TLSync eliminates the problems of latency scaling (by using transmission lines) and of supporting multiple groups (by using different frequency bands for different groups).

Logical zero and one in a TLSync barrier are represented by the presence or absence of a signal at a particular frequency (the simplest form of binary amplitude modulation). Each group of cores that uses barrier synchronization is dynamically assigned to a particular frequency, thus separating it from other groups without requiring a separate set of transmission-lines (TLs) for each group. Each core has a transmitter and a receiver (Figure 1(b)), both tuned to the frequency assigned to the group the core belongs to. The transmitter outputs a continuous signal at this frequency into the TL network, and the receiver detects the presence of such a signal. When a core arrives to the barrier, it stops transmitting. When the last core in a barrier-synchronized group has arrived and stops transmitting, after a short delay (propagation and receiver delay) all the cores in the group detect the absence of the signal and can continue past the barrier. Because different groups of cores can use different frequencies, many barriers can be implemented simultaneously using the same TL network and the same transmitter/receiver circuitry in each core. The only change from using a single, fixed, frequency for all cores is to make the transmitter and receiver tunable (see Section 4).

We note that the design of a TL based broadcast network is not a straight-forward task. The number of cores can be large and a single TL segment can have a limited number of connections (see Section 4). To overcome this problem, we use multiple TL segments connected using amplifiers instead of routers and repeaters, so all of the segments still form the same broadcast medium (see Figure 1(a)). Section 4 also provides implementation details for the TL broadcast net-

[1]The chip-spanning wire would have a total length of 64mm (16mm×4) to connect all 16 cores, while ITRS [15] reports a 0.54ns/mm latency for an unrepeated (continuous) wire in 45nm technology.

[2]Total wire length grows to 88mm and wire delay is expected to be 1.13ns/mm in 32nm technology

[3]Total wire length of 126mm, with 2.48ns/mm delay

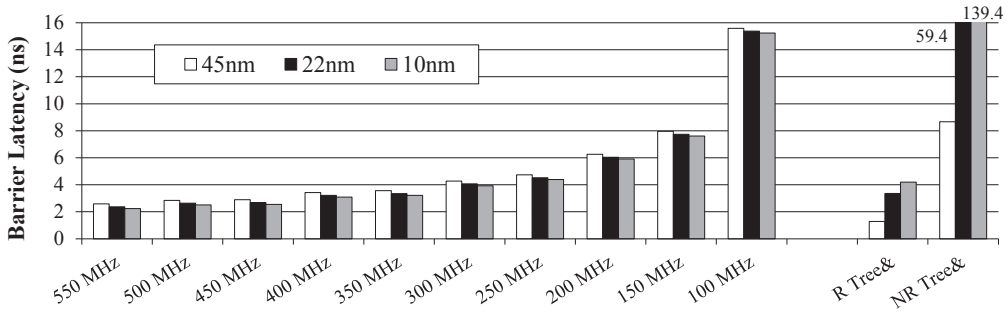


Figure 2: TLSync Barrier latency for different frequency band allocations.

work and for transmitters and receivers used in our implementation of TLSync barriers.

The total delay of a TLSync barrier is composed of three types of latencies: 1) the TL propagation time, which largely depends on the total distance traveled between the farthest-apart transmitter and receiver, 2) transistor circuit latencies, which depend on the complexity of the circuitry in transmitters, amplifiers, and receivers, but continues to improve with technology scaling, and 3) delays introduced by fundamental requirements of signal processing in filters and demodulators. We now summarize the various components of TLSync barrier latency, in the order in which these delays are encountered by the transmitted signal.

The transmitter delay contains only a single CMOS pass-gate, which directly connects the output of the digital phase-locked loop (PLL) to the transmission line (see Figure 1(b)). This delay is only a few picoseconds and improves with technology scaling. Propagation speed along transmission lines is nearly constant at 0.0075ns/mm [19] and is largely unchanged by technology scaling. For a 16mm×16mm die with 16 cores, the worst-case length of TL segments traversed by a signal in our broadcast network is 24mm and there are three amplifiers along the way—as shown in Figure 1(a), there is a local *collection* segment (4mm) with four transmitters, then an amplifier feeds the collected signals to the central collection segment (8mm). The overall collected signal is amplified and sent onto the central distribution segment (8mm again), then amplified again and sent to the local distribution segment (4mm again). Each amplifier introduces 0.05ns of delay [28], for a total propagation delay (transmitter output to receiver output) of 0.33ns. When the number of cores is increased to 64 (in 22nm), the longest path is 32mm (8mm, 6mm, and 2mm central, quadrant, and local segments for collection and for distribution). There are now five amplifiers along each path, but their latency has improved with technology [28] to 0.025ns, so the total propagation latency is 0.365ns. With 256 cores in 10nm technology, the path is 36mm with seven (even faster) amplifiers, for a total propagation latency of 0.358ns (36mm × 0.0075ns/mm + 7 amplifiers × 0.0125ns/amplifier).

At the receiver, a mixer in 45nm technology introduces a 0.5ns delay [2], which is expected to improve to 0.25ns and 0.13ns with technology scaling to 22nm and 10nm. Finally, band-pass filter and amplitude tracker (AM demodulator) latency is dominated by fundamental signal processing delays. The demodulator's delay depends on the frequency of its input signal (filtered mixer's output). We use a 1.8GHz signal, resulting in a 0.28ns demodulator latency (half of the period). The filter creates a fundamental tradeoff between latency and frequency band allocation[4]. When we allocate to a barrier larger spectrum around its "tone" frequency, we achieve faster filtering but reduce the number of barrier groups that can be supported. We experimented with frequency bands as narrow as 100MHz to as wide as 500MHz, designing the filter to have a frequency profile just steep enough to guarantee that signals at surrounding frequencies will not result in false positives of false negatives for a barrier. Filter designs that worked well in most cases were 3-stage digital infinite impulse response (IIR) filters, with delays of 1.74ns, 2.32ns, 3.17ns, 5.15ns, and 14.48ns when using 500MHz, 400MHz, 300MHz, 200MHz, and 100MHz bands, respectively [19][5].

The total end-to-end delay for our barrier implementation, from the time the last core arrives to the barrier to the time all processors are notified, is shown in Figure 2 for different frequency band allocations and technology nodes. This delay includes propagation and receiver (mixer, filter, demodulator) delays. For comparison, we also show latencies for a traditional-wire reduction-notification tree barrier with optimal repeater spacing and sizing ("R Tree&") and without repeaters ("NR Tree&"). Recall that a traditional-wire tree can only support one barrier, and that optimally-repeated wires rely on using silicon along the wire's path.

We observe that, at the 45nm node, our barrier implementation has a latency of 2.85ns with a 500MHz band assigned to each group. Note that with 16 cores there are at most eight 2-core groups, so 500MHz per group still leaves more than 5GHz of spectrum for other uses. However, the optimally-repeated traditional barrier tree has a latency of only 1.28ns, and with only 16 cores only a few "R Tree&" networks may be needed on a chip.

However, at future technology nodes the latency of a traditional barrier network sharply increases while the number of cores (and thus potential barrier groups) also increases. In contrast, the latency of the TLSync barrier implementation

[4]The fundamental filter delay constraint is a close cousin of the sampling theorem—to preserve a signal at one frequency while suppressing another whose frequency differs by x GHz, the filter must "observe" the signals for $1/x$ ns. Actual filters have small additional circuit delays and can save some time by having less suppression for "neighboring" signals, but their delay is still primarily determined by this fundamental constraint.

[5]The actual filter design was for 450MHz, 350MHz, etc. pass-band, leaving 25MHz on each side of the pass-band to reduce interference from neighboring signals in order to avoid use of even more filter stages.

improves slightly, as faster transistor circuits compensate for longer paths traversed by the signals. More importantly, the same TL network can support multiple (many) simultaneous barriers that can be active when many cores are available.

We also observe that, regardless of technology node, filter delays result in significant barrier latency increase when the frequency band allocated to a synchronization group is 200MHz or smaller, and that other latency components are dominant when the allocation is more than 450MHz per group. Between 250MHz and 450MHz per group, there is a reasonable tradeoff between spectrum consumption and barrier latency. One way to manage this tradeoff is to vary size of the band at runtime, depending on how many barriers are active. With fewer than 10 barrier-synchronized thread groups, each group would get a 500MHz band to support a 2.64ns barrier latency at 22nm and 2.50ns latency at 10nm. With more groups, progressively smaller bands would be allocated to each group, with a small gradual increase in barrier latency experienced by each group.

4. IMPLEMENTATION DETAILS

4.1 Using TLs to Connect Many Cores

Our TLSync mechanism relies on a chip-wide broadcast network built with transmission lines (TLs). Ideally, this broadcast would be provided by connecting all transmitters and receivers to the same TL (that winds its way through the chip). However, each such connection changes the electromagnetic field that propagates along the TL, thus weakening and distorting the signals represented by this field. Our detailed simulations of electromagnetic propagation in transmission lines (see Section 4.3 for details) show that, for on-chip distances, the length of the TL segment is almost irrelevant—longer segments do have more signal attenuation (weakening) and reflection (which causes distortion), but the main factor that determines the overall attenuation and distortion is the number of connections.

To overcome this problem, we use multiple TL segments instead of a single TL (see Figure 1(a)). These segments are connected using amplifiers instead of routers and repeaters, so all of the segments still form the same broadcast medium. In light of the results from our electromagnetic simulations (Section 4.3), we use TL segments with at most five connections. Our simulations show that the attenuation of such a segment can still be compensated for by a relatively simple amplifier, and that reflected signals are still weak enough to not interfere much with transmitted signals.

When connecting multiple segments using amplifiers, we must avoid positive feedback loops. In practice, this means that we cannot connect a TL segment to another one using amplifiers in both directions, and we cannot form a ring of segments. Instead, the TL network we use consists of a quaternary tree of input segments, possibly a shared root segment, and a quaternary tree of output segments. Transmitters from four cores are connected to a leaf "collection" segment, which feeds an amplifier. The outputs of four such amplifiers are connected to a second-level collection segment, which feeds an amplifier connected to a third-level collection segment, etc. If the number of cores on chip is a power of four, this tree has a root collection segment that feeds an amplifier whose output is connected to the root distribution segment, which feeds four amplifiers connected to distribution segments, etc. until eventually each leaf distribution segment feeds four receivers (see Figure 1).

If the number of cores is not a power of four, the root collection and distribution TL segments each have fewer connections, so a single root segment can be used for both. For example, with 32 nodes, eight leaf collection segments feed two second-level collection segments, and two second-level distribution segments feed the eight leaf distribution segments. As a result, all four second-level segments (two for collection and two for distribution) can be connected using a single root segment.

By placing the root TL segments in the middle of the chip and by carefully laying out the others, the total length of TL that must be traversed by any signal (from a transmitting core to the farthest-away receiving one) is close to the sum of the width and length of the chip, plus a few millimeters of extra distance to avoid having turns in a transmission line segment—a segment with a turn would have additional attenuation and reflection, and would require a separate analysis to determine how many connections can be allowed.

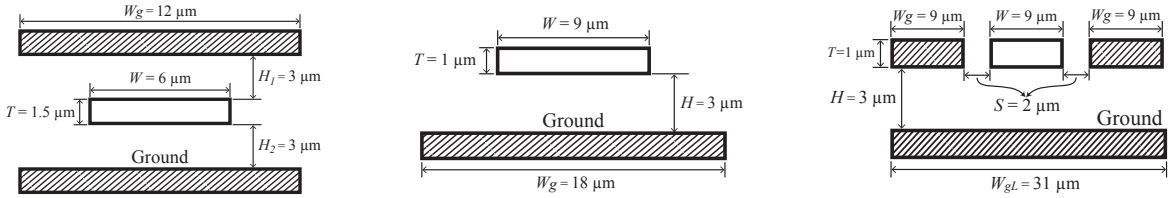
4.2 Transmitter and Receiver Design

When designing transmitters and receivers for a multicore broadcast network based on transmission lines (TLs), the main goal is to preserve signal fidelity without introducing a significant delay. This is not a simple task, because signals with different frequencies propagate differently in TLs—lower-frequency signals propagate less quickly than high-frequency ones, but high-frequency signals suffer more attenuation and more distortion due to crosstalk, skin effect, and other artifacts of electromagnetic wave propagation. These effects affect non-modulated high-data-rate signals: rapid transitions have very high-frequency components and are thus significantly distorted; this limits the data rate that can be achieved without sophisticated modulation and multiplexing methods. On the other hand, sophisticated modulations and multiplexing could introduce signal processing delays that are incompatible with on-chip latency requirements. In fact, most of the high-data-rate modulation and multiplexing schemes used in modern telecommunications rely on digital processing in the receiver, with milliseconds-long delays. Such delays are several orders of magnitude longer than those needed for on-chip communications.

As a result of these considerations, practical on-chip data broadcasts are likely to use simple signaling methods (e.g., baseband signaling without modulation) and have relatively low data rates. Therefore, a significant frequency range below the TL cut-off frequency (see Section 4.3) may be unused, because signal integrity at those frequencies is not sufficient for data transmission without sophisticated (and long-latency) modulation schemes. These frequencies can still be used for other mechanisms that require low-latency broadcasts, such as our TLSync barrier synchronization.

In light of this discussion, TLSync uses the simplest form of binary amplitude modulation (i.e., the presence or absence of a signal at a particular frequency) to indicate a logical zero or one. Transmitter delay is very small—when a core reaches the barrier, it simply turns off the pass-gate that connects the output of the PLL to the transmission line. However, the receiver design is more challenging.

To implement different groups of cores that use different frequencies for barrier synchronization, the only change from using a single, fixed, frequency for all cores is to make the



(a) Stripline (b) Microstrip (c) Coplanar

Figure 3: Realizations of transmission lines.

transmitter and receiver tunable. A naive approach to providing tunability would require a tunable PLL in the transmitter, and a tunable filter (demodulator) in the receiver, but a tunable digital filter would have significant additional delays. Fortunately, a viable receiver can be implemented using a fixed-frequency filter by leveraging the same tunable PLL used in the core's transmitter. For this, the received signal first goes into a *mixer*, which combines the signal with a reference signal from the PLL. The output of the mixer is a *down-converted* signal—signals from the input signal are represented in the output, but their frequencies are reduced by the frequency of the reference signal[6]. The output from the mixer is then fed to a band-pass filter whose frequency is constant. Tuning is achieved by controlling the frequency of the reference signal. For example, if the filter's pass band is centered at 1.8GHz and the group's assigned frequency is 8GHz, the reference signal would be generated at 6.2GHz. To generate reference signals for transmitters and receivers, we use digital PLLs that can be implemented efficiently. Tuning is achieved by changing the PLL's divider and multiplier factors, without changing phase compensation and other hard-to-design circuitry of the PLL.

Because our barrier only requires detection of the presence or absence of a "tone" at a given frequency, the band-pass filter used for the barrier receiver needs not have a very steep frequency profile—it only needs to have low attenuation at "tone" frequency of the band allocated to the group and large-enough attenuation for "neighboring" tones, but the attenuation profile for frequencies in-between need not be sharply defined. This allows this filter to be both less expensive and faster: as explained in Section 3, the delay introduced by the filter fundamentally depends on how sharply its frequency profile is defined. Finally, the output of the band-pass filter is then fed to an AM demodulator—a diode and a small capacitor that tracks the amplitude of a high-frequency signal. The delays in the filter and AM demodulator are heavily dependent on the width of the filter's pass-band and on the frequency at which the AM demodulator is operating. In the proposed system, we use 1.8GHz for AM demodulator's frequency and several hundred MHz for the filter's pass-band.

4.3 Transmission Line Design

A transmission line (TL) consists of a signal conductor (wire), surrounding dielectric, and one or more grounding conductors that run in parallel with the signal wire. There are three main realizations of TLs that are potentially suitable for on-chip implementation: *stripline*, *microstrip*, and *coplanar waveguide*, as shown in Figure 3.

In designing TLs, the important characteristics are attenuation (forward loss), reflection (return loss), and propagation delay at different frequencies. These characteristics are, in turn, affected by not only dimensions of the conductor, but also the thickness and type of dielectric that separates it from the ground plane, by the fact that the "ground plane" is of finite dimensions, and by the size, shape, and impedance of any connections, bends, etc. This is further complicated by the fact that attenuation, reflection, and delay of TLs are not constant across all frequency bands—typically, there is a range of frequencies within which the TL has acceptable characteristics. A TL behaves as a low-pass filter—the attenuation slowly worsens as the frequency of the signal increases up to some frequency (called the cut-off frequency), with rapidly worsening attenuation beyond the cut-off frequency. The end result of this is that any relatively accurate characterization of TLs requires an electromagnetic (EM) propagation simulator (ADS, MWO, Linpar, etc.).

We used MWO electromagnetic simulator [19] to determine appropriate on-chip implementations for all three realizations of TLs. In this analysis, we focused on achieving good transmission characteristics (attenuation, reflection, latency) in a relatively large spectrum, while minimizing the total area consumption in the affected metal layers. After careful consideration, we selected a 1–10GHz spectrum as the target. Although a similar-sized spectrum at higher frequencies (e.g., at 56–65GHz, which is often used for on-chip RF circuitry in solid-state literature) would result in smaller and somewhat faster amplifiers, at higher frequencies the attenuation in TLs is much higher. To compensate for this, we would need larger number of amplifiers and very sophisticated analog circuitry at transmitters and receivers, which will increase the delay. As a result, we chose to operate in the <10GHz part of the spectrum. The optimized dimensions for all three realizations of TLs in a 45nm process are shown in Figure 3. The smallest dimensions are obtained for stripline realization, which we choose for our design.

As a result of all these considerations and limitations, on-chip TLs must be designed carefully, especially when the geometry of TLs is further complicated by having more than just a single receiver and a single transmitter at the endpoints of the line. We find that these considerations have a major effect on the characteristics of the TL. As an example, consider a straight 16mm run of stripline implemented as in Figure 3(a). With a single transmitter (at one end) and a single receiver (at the other end) that are both impedance-matched to the TL in the 1–10GHz frequency range, forward loss (attenuation) is 5dB at 1GHz and 10dB at 10GHz, return loss (attenuation of the reflected signal) is 15dB at

[6]The output of the mixer also contains up-converted signals, whose frequency is the sum of input and reference frequencies. These signals are eliminated by the filter that follows.

Parameters	Small-Core	Large-Core
Core Frequency, Issue Width, ROB Size	1 GHz, 2 (Out-of-Order), 64	3 GHz, 4 (Out-of-Order), 128
Cache Line Size	64 Byte	64 Byte
L1 Instruction Cache	32 KB, Private, 2 way, 1 cycle	32 KB, Private, 2 way, 1 cycle
L1 Data Cache	32 KB, Private, 2 way, 1 cycle	32 KB, Private, 4 way, 3 cycles
L2 Cache	16 MB, Shared, 16 way, 6 cycles	256 KB, Private, 8 way, 5 cycles
L3 Cache	–	32 MB, Shared, 16 way, 23 cycles
Memory Latency	110 cycles	323 cycles
TLSync Barrier Latency (350/300/200/150MHz)	4/5/6/9 cycles	12/13/16/25 cycles

Table 1: Simulation Setup

1GHz and 10dB at 10GHz, and propagation delay is 110ps. Although forward loss is significant (the received signal retains only 10% of the transmitted power), the signal is still significantly above the noise level and can be amplified using a low-noise linear amplifier. The return loss and delay, however, are excellent: the reflected signal is weaker[7] than the one being sent, so reflection causes little interference, and the delay is much lower than that of a traditional 16mm wire implementation (even with optimal repeater spacing). However, if we have 15 additional transmitters (at 1mm intervals along the TL), the forward loss, return loss, and propagation delay for the transmitter at the end of the TL become 44dB, 7dB, and 139ps at 10GHz. This signal is unlikely to be received successfully—the forward loss suppresses the signal well into the noise level, and the reflected signal is nearly 10,000 times stronger than the weakened forward signal. This loss occurs because impedance cannot be matched equally well in a wide frequency range (more than a few GHz), resulting in a slight-to-moderate impedance mismatch at some frequencies. This mismatch adds significant additional attenuation and reflection.

A practical implementation must have relatively few receiver/transmitter connections on the same TL segment, with careful planning and design of those connections. Through careful experimentation, modeling, and EM simulation, we found that up to 6-7 transmitters (and an amplifier at the receiving end) can be safely implemented on a single transmission segment. Reception segments are a bit more forgiving, allowing for 10-12 receivers (each with its own amplifier) to reside on a segment fed by an amplifier. Our TLSync design is conservatively engineered to have five connections (a transmitter and four receivers, or four transmitters and a receiver) on each segment.

5. EXPERIMENTAL EVALUATION

We evaluate the performance impact of our TLSync barrier in terms of raw barrier latency, and in terms of execution time on Livermore loop kernels [21], hand-parallelized Dithering application from EEMBC benchmark [12] and the Streamcluster application from the PARSEC benchmark [7]. Other PARSEC benchmarks use barriers rarely and their performance is not affected much by barrier latencies.

For comparison, we also show results for a optimized implementations of the widely-used centralized sense reversal software barrier (shown as "Centralized"), the tournament barrier (among the most scalable software barrier implementations [20]), and the hardware-assisted Barrier Filter [23] implementation. In all three implementations, we carefully arrange shared variables to avoid unnecessary coherence traffic and use the best-performing variant (e.g., ping-pong with I-Cache for Barrier Filter).

We obtain our performance results using the SESC [22] cycle-accurate simulator. We model two types of cores, summarized in Table 1, with core counts from 4 to 256. The "Small-Core" setup models a "many-smaller-cores" approach that can achieve large core counts (64 and beyond) in the near future, while the "Large-Core" setup models the approach of using fewer high-performance cores, which can be expected to achieve large core counts later. The last row in Table 1 shows TLSync barrier latency with 350, 300, 200, and 150MHz bands allocated to each barrier in terms of clock cycles in each type of cores.

5.1 Barrier Latency

We measure the raw barrier latency by executing (in parallel) a tight loop with 64 barriers in the loop body, then dividing the execution time with the number of completed barriers.

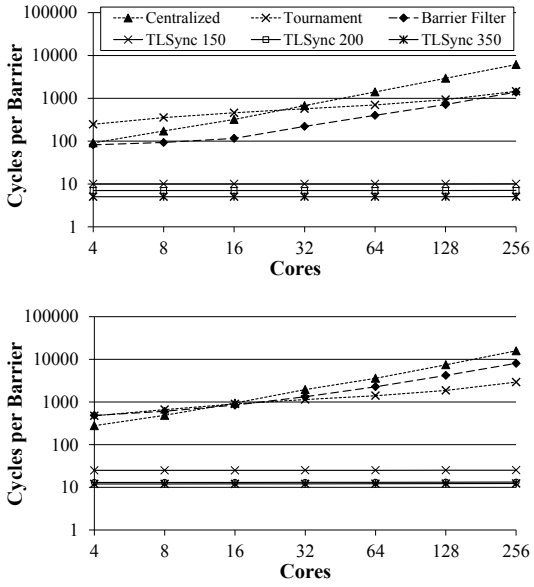


Figure 4: Raw barrier latency in Small-Core (top) and Large-Core (bottom) configurations.

Figure 4 shows the results from this experiment. In both configurations, TLSync barriers have very low latency which stays nearly constant as the number of cores increases. This is because the number of cores affects the barrier latency

[7]At 10GHz and beyond, the reflected signal becomes too strong, which is part of the reason why the usable spectrum for this transmission line ends at 10GHz

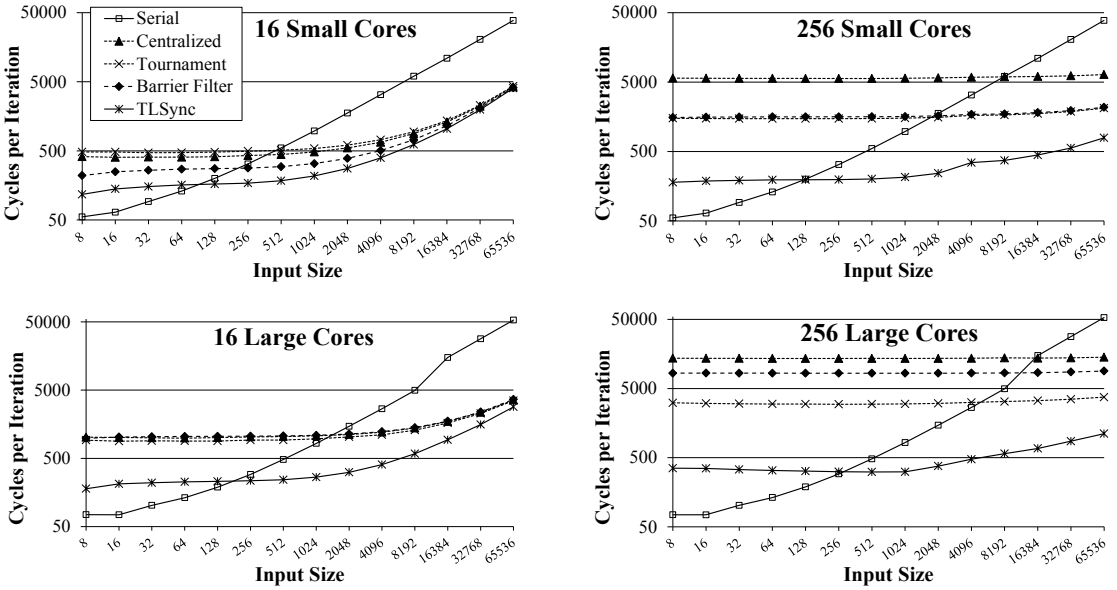


Figure 5: Performance of Livermore Loop 2 for various barrier mechanisms.

only slightly—e.g., from 16 to 256 cores (assuming the same 16mm by 16mm chip size) the longest path for signal propagation changes from 24mm to 36mm and there are four more amplifiers on that path. The resulting additional delay is only 290ps (less than one cycle even for "Large-Core"). The width of the frequency band allocated to the barrier changes its latency noticeably (note the logarithmic scale in Figure 4), but even with only a 150MHz band the TLSync barrier outperforms the others by an order of magnitude for small core counts and by two orders of magnitude for large core counts. As expected, the centralized sense reversal barrier outperforms the tournament barrier up to a certain point (16 small cores, 8 large cores). This is because the tournament barrier executes more code in each core but with little serialization among cores, whereas the centralized barrier uses a simple counter increment but serializes increments from different cores. Barrier Filter reduces barrier latency, but its scaling trend is similar to the other two because there is still serialization in the barrier filter logic that receives notifications from each arriving core.

5.2 Livermore Kernels

Livermore loops [21] have long been used in architecture and compiler research for testing the scaling of parallel performance. We use Livermore Kernels 2, 3 and 6 to evaluate the performance impact of different barrier implementations for workloads with large amounts of fine-grain parallelism. In parallelizing these kernels, we followed the same methodology that was used for Barrier Filter work [23], and we carefully aligned data structures to cache line boundaries and assigned data in cache-line-sized chunks to minimize coherence traffic. For the TLSync barrier, we only show the results when a 350MHz band is allocated to a barrier group, because results for bands sizes from 150MHz to 350MHz do not perceptibly differ from each other.

Figure 5 shows cycles per loop iteration in Kernel 2 for different barrier implementations across various input sizes. For comparison, each chart also shows the results for sequential execution ("Serial"). In each chart, as input size increases, more iterations are assigned to each core and the impact of barrier latency decreases. With 16 cores (left half of Figure 5), the difference in performance between TLSync and other barrier implementations almost disappears when the input size reaches 32,768. However, when the number of cores is increased to 256 (right half of Figure 5), the same work is split among more cores and the impact of barrier latency increases—even with the input size of 65,536, the TLSync barrier provides speedups of at least 2X relative to tournament and Barrier Filter barriers. We also observe that barrier latency has a larger impact with large cores, because they finish the work between barriers faster and experience longer barrier latencies. This means that, as technology scales, barrier latency will have more impact regardless of whether the increased transistor count is used to improve performance of each individual core, to put more cores on the chip, or for a combination of both.

Figure 6 shows the results of our experiments on Livermore kernel 3. Small and large cores have nearly identical trends, so we omitted Small Core results for brevity.

With 16 cores, the trends are similar to those observed for Livermore Loop 2. Loop 3 computes a simple inner product of two vectors and is parallelized by computing a partial sum-of-products in each thread, then combining the partial sums in the "main" thread. With 256 cores, the parallel portion of the execution is finished much faster than with 16 threads, but the serial part takes longer (256 partial sums to add instead of 16) and dominates the execution time until input sizes become sufficiently large. This causes "Serial" execution to perform better than any of the parallel executions until input sizes reach about 512. The same effect reduces the impact of barrier latency for small inputs and moves the convergence point for their performance beyond the 8192 input size.

Livermore Loop 6 is a reduction for a general linear equation, and is parallelized using the coordinate axis transformation [23]. The results for Loop 6 are shown in Figure 7.

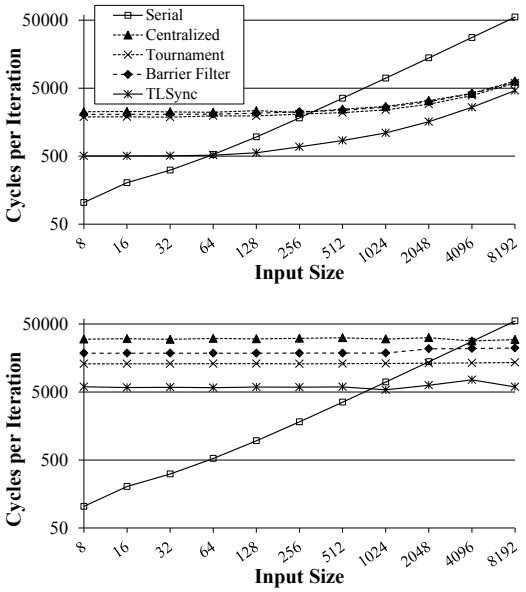


Figure 6: Performance of Livermore Loop 3 with various barrier mechanisms with 16 (top) and 256 (bottom) cores.

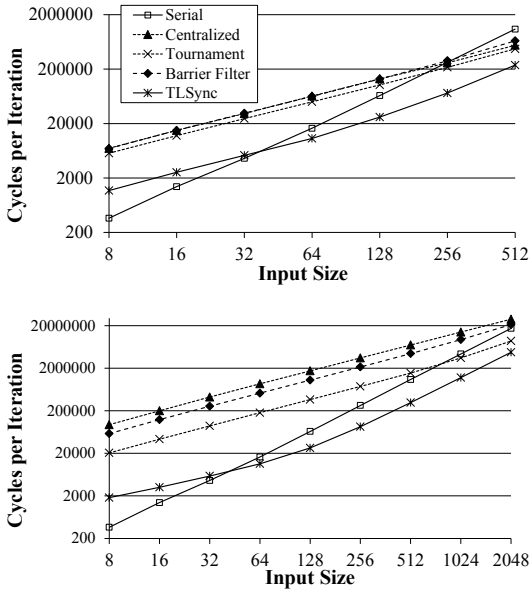


Figure 7: Performance of Livermore Loop 6 with various barrier mechanisms with 16 (top) and 256 (bottom) cores.

Again, we only show Large Core results for brevity because Small core results show nearly identical trends.

The trends for 16-core execution are nearly identical to those for Loop 2 and Loop3—the lower barrier latency in TLSync results in better overall performance, with a trend toward eventual convergence as input sizes increase. When the core count increases to 256, the difference between TLSync and longer-latency barrier implementations increases because each core has less work between barriers and the barrier latency becomes more significant.

In addition to lowering execution times, an important effect of a low-latency barrier implementation is that it reduces the input size necessary to achieve a parallel speedup. In all three Livermore loops (Figures 5, 6, and 7) we observe that with TLSync barriers the break-even point (where the parallel execution time is equal to serial execution time) occurs at smaller input sizes than with longer-latency barrier implementations. This is very important for parallelization of non-scientific applications—unlike scientific applications where ever-larger input sizes are the norm, input sizes for many consumer applications are either determined by standards (e.g., for video and audio) or grow less quickly than core counts due to other limitations, e.g., display resolution and human perception thresholds for games.

5.3 Streamcluster and Dithering

To examine the impact of various barrier mechanisms on overall performance in non-scientific applications, we use the Streamcluster benchmark from PARSEC [7] and the Dithering benchmark from the EEMBC [12] benchmark suite. Streamcluster is an RMS kernel that solves an online clustering problem using a streaming algorithm. To synchronize threads between relatively short tasks along the stream, it uses a large number of barriers (8,772 for 1K data points). Dithering is an embedded benchmark that converts an image into a error-diffused image suitable for printing using the Floyd-Steinberg dithering algorithm. We hand-parallelized the application to exploit fine-grained parallelism by processing each row concurrently. In our implementation, a pair of barriers were used to synchronize error propagation between rows.

Figure 8 shows the parallel speedup of the two applications for different barrier implementations and numbers of cores. We used 1K data points for the Streamcluster, and a 4,096×3,072 image for the Dithering application. Each application executed 8,772 and 6,144, respectively. In addition to the centralized, tournament, and Barrier Filter barrier implementations, we compare TLSync performance to a hardware barrier with reduction and notification trees implemented with traditional wires with optimally-spaced repeaters (shown as "R Tree&") Recall that this hardware barrier implementation requires a separate barrier tree for each barrier group, and that the optimally-repeated chip-spanning wiring of this implementation requires repeater placement at frequent intervals along each wire, which interferes with the layout of "local" logic (e.g., cores and caches) that the wire is traversing. We observe that hardware barriers (both variants of TLSync and "R Tree&") show nearly identical performance than software-only and Barrier Filter implementations. This result is in-line with our results from Livermore loops, and leads to two important conclusions regarding the performance impact of barriers. First, various scalable low-latency barriers have similar performance on real workloads in spite of differences in actual barrier latency—when barrier latency is low enough (tens of cycles or less), execution time between barriers dominates the execution time and further reduction in barrier latency provides minimal returns. Second, the advantage of scalable low-latency barriers increases significantly as the core count increases, not only up to the point when the application performance stops scaling for other reasons (e.g., Streamcluster), but also beyond the point (e.g., Dithering). As noted

Active Element	Area (mm^2)	Power (mW)	# Used	Total Area (mm^2)	Total Power (W)
Amplifier [8]	0.008	8	9	0.08	0.07
Mixer [25]	0.005	1.46	16	0.08	0.02
Filter [29, 31]	0.07	10	16	1.12	0.16
AM demodulator [30]	0.002	3.03	16	0.02	0.05
Total	–	–	–	1.30	0.30

Table 2: The cost of main active components used in TLSync for a 16-core processor in 45nm technology.

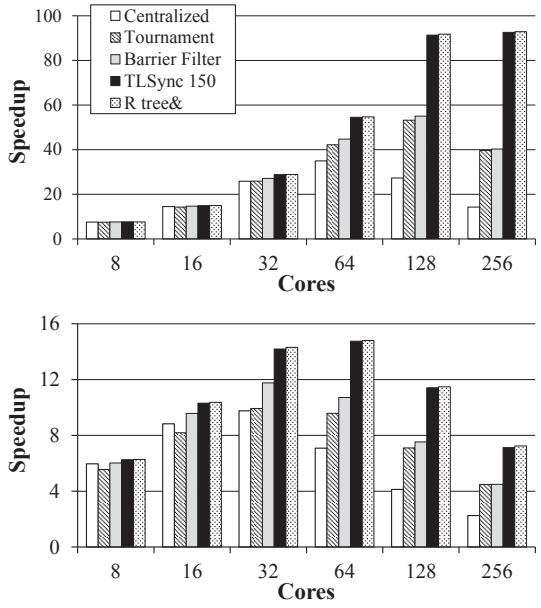


Figure 8: Performance of Dithering (top) and Streamcluster (bottom) with various barrier mechanisms.

earlier, this is because larger core counts result in less work per core between barriers, turning barrier latencies into a larger fraction of overall execution time, and thus increasing the performance impact of barriers.

5.4 Implementation Cost

Our transmission line (TL) broadcast network with TLSync barrier support has two main sources of cost—the use of metal layers for the actual TL, and the use of silicon area for active components of the design.

In terms of metal area, a TL should be implemented in the top (global) metal layers. Because a TL is shielded from the circuitry below by its "ground plane" conductor, it can easily be routed over any local circuitry. Additionally, our TL broadcast network only requires amplifiers at connection points between segments, at millimeter-scale distances from each other. This allows our TL network to be routed without affecting the placement of circuitry within each core.

The main drawback of using TLs is in their large width: a stripline implementation occupies a 12μm-wide strip in three metal layers. Although the main conductor is only 6μm wide, an additional 3μm on each side must be free of other wires or TLs to keep crosstalk within tolerable bounds. This metal use is significantly larger than for traditional global wires, which have only a 0.135μm pitch in 45nm technology, with proportional scaling in future technologies. As explained in Section 3, a single chip-spanning TL broadcast tree can support multiple barrier groups with a very low latency, and the total metal area needed for this tree in 45nm technology is only 1.7mm^2—two lines (input and output) for each segment, 24mm total length of all segments, using 12μm each in three layers for a total area of 2×24mm×0.012mm×3. Although this appears expensive, it should be noted that it represents only 0.07% of the total metal area for a 16mm×16mm chip with 10 metal layers. Furthermore, the only requirement for "ground plane" conductors is that they should not carry RF signals, so these wide conductors can be leveraged for Vdd and Vss distribution. Alternatively, if the metal layers used for Vdd and Vss are placed above and below the TL's main conductor metal layer, there would be no need for separate "ground plane" conductors, thus reducing the metal area demands for TLs by a factor of three.

The main active components in TLSync are the amplifiers between transmission line segments, and the mixer, filter, and AM demodulator (amplitude tracker) circuitry in each receiver. Table 2 summarizes the chip area occupied by each active component, together with energy consumed by that component. We use the resistive-feedback low-noise amplifier with a high gain from [8], the low-voltage ultra-wideband down-conversion mixer from [25], the ultra-compact active bandpass filter from [31] with low-power enhancements from [29], and the compact CMOS-based amplitude tracker from [30]. Because the original filter and amplitude tracker were built in 180nm and 350nm CMOS technology, their area and energy numbers were scaled to 45nm technology using conservative assumptions: we assumed that area is reduced in linear (not quadratic) proportion to feature size, and that power is reduced only because of the reduction in supply voltage (this preserves the current in analog circuitry to ensure its correct operation after scaling). Note that all components are chosen to scale well and avoid the use of large passive (e.g., inductor, capacitor, and resistor) components, so their actual scaling is likely to be better than described above.

6. CONCLUSIONS

This paper presents TLSync, a novel hardware barrier implementation that uses the high-frequency part of the spectrum in a transmission-line broadcast network, thus leaving the transmission line network free for non-modulated (baseband) data transmission. In contrast to other implementations of hardware barriers, TLSync allows multiple thread groups to each have its own barrier. This is accomplished by allocating different bands in the radio-frequency spectrum to different groups.

Our circuit-level and electromagnetic models show that the worst-case latency for a TLSync barrier would be between 4ns and 10ns, depending on the size of the frequency band allocated to the barrier. Our cycle-accurate simulation

results show that such TLSync barriers 1) provide significant performance improvements for barrier-intensive multi-core applications and 2) improve scalability over software and some hardware-assisted implementations. Since the performance improvement is not very sensitive to variations in barrier latency caused by allocating larger or smaller (up to a point) frequency bands to the TLSync barrier, tens of different barriers can be supported using a single physical transmission-line chip-spanning tree while still leaving the transmission line free for baseband (unmodulated) data transmission.

Overall, we find that TLSync barrier support is a very good candidate for cost-effectively providing scalable and low-latency barrier synchronization for multiple thread groups in future many-cores.

7. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grants 1017638 and 0903470, and by the Semiconductor Research Corporation under contract 2009-HJ-1977. the authors would also like to thank Chris Hughes (Intel) for highly valuable feedback on an early draft of this paper.

8. REFERENCES

[1] J. L. Abellán, J. Fernández, and M. E. Acacio. Efficient and scalable barrier synchronization for many-core cmps. In *7th ACM Intl. Conf. on Computing frontiers*, pages 73–74, 2010.

[2] Advanced Design System. Agilent Technologies, Santa Clara CA, USA, 2010.

[3] G. Almási, C. Archer, J. G. Castaños, J. A. Gunnels, C. C. Erway, P. Heidelberger, X. Martorell, J. E. Moreira, K. Pinnow, J. Ratterman, B. D. Steinmacher-Burow, W. Gropp, and B. Toonen. Design and implementation of message-passing services for the Blue Gene/L supercomputer. *IBM J. Res. Dev.*, 49:393–406, 2005.

[4] B. Beck, B. Kasten, and S. Thakkar. VLSI assist for a multiprocessor. In *second Intl. Conf. on Architectual Support for Prog. Lang. and Operating Sys.*, pages 10–20, 1987.

[5] B. M. Beckmann and D. A. Wood. TLC: Transmission Line Caches. In *36th annual IEEE/ACM Intl. Symp. on Microarchitecture*, pages 43–54, 2003.

[6] C. J. Beckmann and C. D. Polychronopoulos. Fast barrier synchronization hardware. In *1990 ACM/IEEE Conf. on Supercomputing*, pages 180–189, 1990.

[7] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *17th Int'l. Conf. on Parallel Architectures and Compilation Techniques (PACT)*, 2008.

[8] J. Borremans, S. Thijs, M. Dehan, A. Mercha, and P. Wambacq. Low-cost feedback-enabled LNAs in 45nm CMOS. In *Proc. of ESSCIRC '09*, pages 100–103, 2009.

[9] A. Carpenter, J. Hu, J. Xu, M. Huang, and H. Wu. A case for globally shared-medium on-chip interconnect. In *38th annual Intl. Symp. on Computer Architecture*, 2011.

[10] 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 *41st annual IEEE/ACM Intl. Symp. on Microarchitecture*, pages 376–387, 2008.

[11] Cray Research, Inc. *CRAY T3D System Architecture Overview*, 1993.

[12] E. M. B. Consortium. EEMBC benchmark. `www.eembc.org`.

[13] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir. The NYU ultracomputer - designing a MIMD, shared-memory parallel machine. In *9th Intl. Symp. on Computer Architecture*, pages 27–42, 1982.

[14] W. T.-Y. Hsu and P.-C. Yew. An effective synchronization network for hot-spot accesses. *ACM Trans. Comput. Syst.*, 10:167–189, 1992.

[15] Intl. Technology Roadmap for Semiconductors. ITRS - 2008 update, 2008. `http://www.itrs.net`.

[16] S. W. Keckler, W. J. Dally, D. Maskit, N. P. Carter, A. Chang, and W. S. Lee. Exploiting fine-grain thread level parallelism on the MIT multi-ALU processor. In *25th annual Intl. Symp. on Computer architecture*, pages 306–317, 1998.

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

[18] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak. The network architecture of the Connection Machine CM-5. In *fourth annual ACM Symp. on Parallel Algorithms and Architectures*, pages 272–285, 1992.

[19] Microwave Office. Applied Wave Research, El Segundo, CA, USA, 2010.

[20] R. Nanjegowda, O. Hernandez, B. Chapman, and H. H. Jin. Scalability evaluation of barrier algorithms for openmp. In *5th Int'l Workshop on OpenMP: Evolving OpenMP in an Age of Extreme Parallelism*, pages 42–52, 2009.

[21] T. Peters. Livermore loops coded in c, 1992. `http://www.netlib.org/benchmark/livermorec`.

[22] 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.

[23] J. Sampson, R. Gonzalez, J.-F. Collard, N. P. Jouppi, M. Schlansker, and B. Calder. Exploiting fine-grained data parallelism with chip multiprocessors and fast barriers. In *39th IEEE/ACM Int'l Symp. on Microarchitecture*, pages 235–246, 2006.

[24] J. Sartori and R. Kumar. Low-overhead, high-speed multi-core barrier synchronization. In *5th Intl. Conf. on High Performance Embedded Architectures and Compilers*, pages 18–34, 2010.

[25] O. Schmitz, S. Hampel, C. Orlob, M. Tiebout, and I. Rolfes. Body effect up- and down-conversion mixer circuits for low-voltage ultra-wideband operation. *Analog Integrated Circuits and Signal Processing*, 64:233–240, 2010.

[26] S. L. Scott. Synchronization and communication in the T3E multiprocessor. In *seventh Intl. Conf. on Arch. Support for Prog. Lang. and Operating Sys.*, pages 26–36, 1996.

[27] S. Shang and K. Hwang. Distributed hardwired barrier synchronization for scalable multiprocessor clusters. *IEEE Trans. on Parallel and Distributed Systems*, 6:591–605, 1995.

[28] 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 *IEEE Intl. Conf. on Systems-on-Chip*, pages 113–116, 2003.

[29] Y. Sun, C. Jeong, I. Lee, J. Lee, and S. Lee. A 50-300-MHz low power and high linear active RF tracking filter for digital TV tuner ICs. In *2010 IEEE Custom Integrated Circuits Conf. (CICC)*, pages 1–4, 2010.

[30] A. Valdes-Garcia, R. Venkatasubramanian, R. Srinivasan, J. Silva-Martinez, and E. Sanchez-Sinencio. A CMOS RF RMS detector for built-in testing of wireless transceivers. In *Proc. of 23rd IEEE VLSI Test Symp.*, pages 249–254, 2005.

[31] Y. Zheng and C. Saavedra. Ultra-compact MMIC active bandpass filter with wide tuning range. *Electronics Letters*, 44(6):424–425, 2008.