Nanophotonic Barriers

Nathan Binkert[‡], Al Davis[§], Mikko Lipasti[†], Robert Schreiber[‡], Dana Vantrease[†]
[‡]Hewlett-Packard Laboratories, [§]University of Utah [†]University of Wisconsin - Madison

Abstract

Multi-threaded programs employ barriers to temporally synchronize across threads. All threads must reach the barrier before any thread may advance past the barrier. A barrier is a form of global synchronization; the efficiency with which the barrier is processed extends the program's critical path. Large delays between the time the last thread reaches a barrier and when all threads receive notification that the barrier has been released can add significantly to program execution time.

In this paper, we show how nanophotonic devices can be configured to provide low latency, low power hardware barriers. The mechanism uses the presence of light in a waveguide to indicate that all threads have arrived at the barrier. DWDM (dense wave division multiplexing) allows multiple barriers to be processed in parallel with only a single physical waveguide.

1 Introduction

Multicore chips allow low-latency communication between cores, making fine-grained parallelism and on-chip data sharing cheap. However, global operations, such as barriers, are still expensive. Prior work has shown that barriers can consume a large portion of program execution time [11, 2]. This is problematic from an application programmer's perspective since barriers are a simple way to achieve synchronization and enforce correct ordering of memory accesses, but the penalty of using them can be surprisingly high.

Software barriers enjoy great popularity, in part because they are flexible and easy to implement. Mellor-Crummey and Scott [6] survey a variety of software barrier implementations and their tradeoffs. Software barriers are inherently interprocessor communication abstractions. Their performance is determined by the latency of a cache to cache transfer across the machine, a cost measured in hundreds of cycles. The best software barriers incur $O(\log N)$ such communications where N is the number of barrier participants.

Several efficient electrical hardware barrier mechanisms have been proposed and built to overcome the performance limitations of software barriers. Most have been proposed for SMP systems and use a dedicated barrier-interconnect network[4, 1, 5, 8]. In chip-multiprocessors, Sampson et al [7] propose using a hardware-assisted barrier, implemented in a shared level of the memory hierarchy. Their barrier takes hundreds of cycles between the last thread arriving and all threads being notified. Alternatively, they report a completely dedicated, centralized electronic barrier mechanism of Beckmann and Polychronopoulos [1], at 64 nodes, has approximately a 40 cycle latency.

To the best of our knowledge, there is only one prior proposal for an optical barrier mechanism [3]. In this proposal, each node broadcasts its identity onto an optical waveguide bus when it reaches a barrier. All other nodes listen to the broadcast and use the broadcaster's identity to decrement a locally maintained counter. If more than one node tries to simultaneously use the broadcast channel, then the conflicting accesses must be resolved in a serial manner. Finally, it performs the barrier function electronically, not optically.

In this paper, we show that it is possible to use nanophotonics to build a simple, fast, all-optical barrier.

2 Photonic Barrier

2.1 Implementation

In general, a barrier mechanism must complete the following steps: (1) barrier initialization, (2) collecting arrivals, (3) barrier release/notification. Often the steps above are repeated during seperate dynamic invocations of a static barrier.

The nanophotonic barrier network is a single waveguide, as shown in Figure 1(a). Each of the N nodes has both a light diverter and a light detector that are attached to the waveguide. The diverters and detectors are specialized incarnations of the ring resonator [12]. There are many possible layouts, but the key is that the waveguide passes by all nodes twice. In Figure 1, for example, the waveguide wraps around the nodes, with the diverters on one side of the nodes and the detectors on the other.

To initialize the barrier, each participating node diverts light from the waveguide. Non-participating nodes do not divert the light. As participating nodes arrives at the barrier, they stop diverting the light. When all nodes have arrived

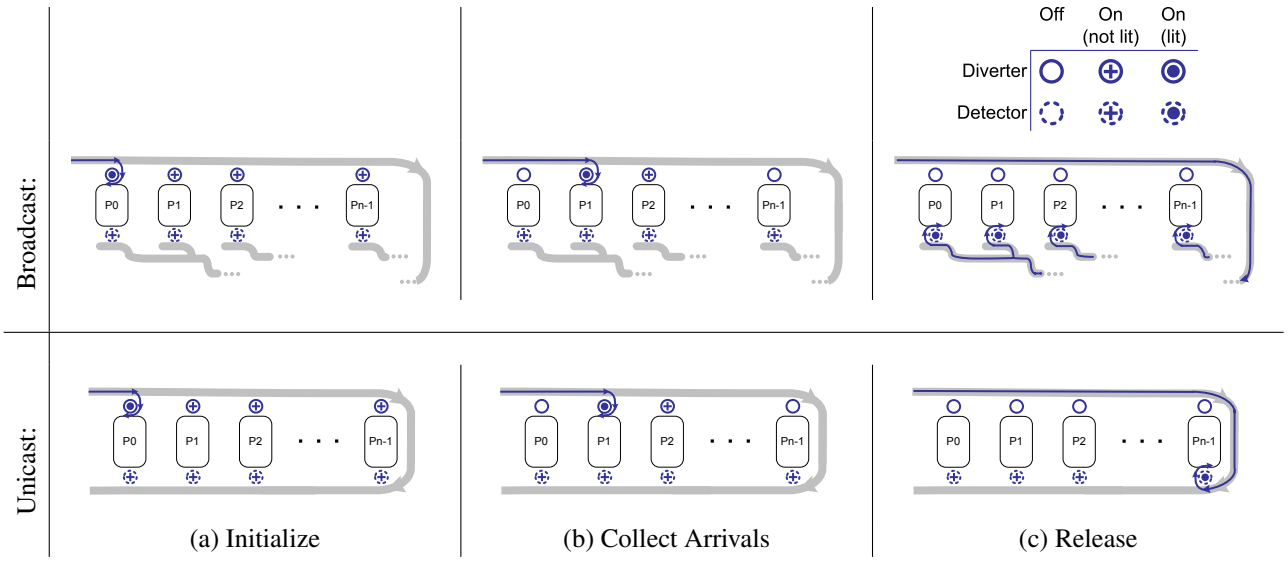


Figure 1: (a) Step 1: Initialize. Each participating node begins by diverting the light. No detectors detect light. (b) Step 2: Collect Arrivals. As participants reach the barrier, diverters cease diverting. Thus far, $P0$ and P_{n-1} have arrived at the barrier. No detectors detect light. (c) Step 3: Release. All nodes have arrived. As the unicast design shows above, notification is serial, and each node must sequentially detect and then turn off its detector to allow others a chance at the light. In the broadcast design, all nodes quickly detect the light with splitters.

at the barrier, the light passes by the inactive light diverters and begins notification. In the next section, we describe two optical notification alternatives.

Collecting arrivals (Step 2) is reflected in a software-visible barrier arrival operation which turns off the node's light diverter. The node then zeros a barrier-status bit, in memory or in a processor status register, and then waits for this bit to be set. The arrival and sensing of the light signal indicates that all nodes have arrived at the barrier. It's arrival triggers two things. First the barrier is re-initialized, and all nodes must be made to divert the light again. Then, the local status register bit is set, permitting the node to proceed. To ensure that all nodes re-divert the light before any is allowed to continue with computation, we set a timer at each node that delays the next arrival at the barrier until enough time elapses to ensure that all nodes have seen the light signal and re-initialized it. This timing interval is set to be the maximum latency of the light path plus the diversion time. Thus, an all-hardware mechanism is used.

With DWDM, the design can easily be extended to support several barriers simultaneously—each barrier is assigned a different wavelength and has its own status bit.

The optical barrier is scalable, in part because of the technology and because of the distributed nature of the mechanism. The distributed nature of the design allows each node to perform its barrier-arrival operation independently of the other nodes. That is, if two nodes arrive at the barrier at the same time, they may both stop diverting light in parallel, and thus the time between the last node's arrival at the barrier and when the participants are notified is only dependent upon the time it takes light to traverse the length of the waveguide. The electrical counterpart of this approach would be a wired-OR structure which doesn't scale well due to the linear relationship between wire capacitance and wire length.

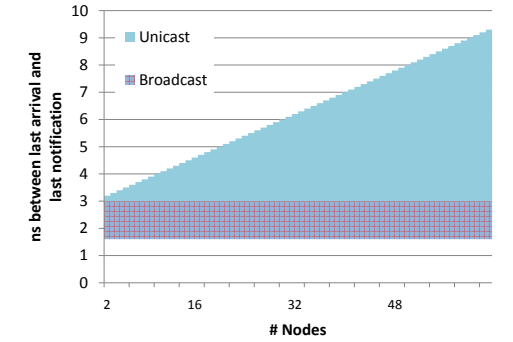
2.2 Unicast vs. Broadcast

The detectors shown in Figure 1 must all be able to sense the presence and absence of light. In real implementations, detecting the presence of light effectively removes the light from the waveguide[1]. This leaves two implementation options for the barrier release operation: unicast and broadcast. Neither implementation is obviously better since there is a tradeoff between latency and power/area.

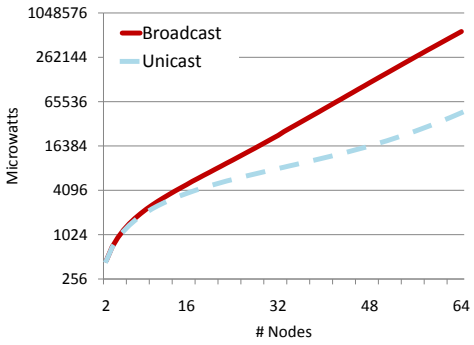
The broadcast is implemented using a splitter device that takes one waveguide and splits it into multiple waveguides, where a fraction of the input power is sent down each subsequent waveguide branch. Multiple splitters are employed to build a broadcast tree. This tree requires $O(N)$ power, and requires a bundle of waveguides $O(\log N)$ wide. The latency of barrier release is $O(L)$ where L is the length of the waveguide bundle.

The alternative, unicast, implementation avoids the need for splitters and is $O(1)$ in power and area, at the cost of the

[1]It is theoretically possible to remove only a fraction of the light, but this sort of implementation is impractical for high node counts (N).



Range in Notification Latency



Power

Figure 2: Latency analysis is for a 64-node system. Power analysis is for a range of nodes and for a single waveguide containing 64 barriers.

latency being $O(N+L)$. The unicast alternative uses diverters at the detector site to divert light to a detector. When the barrier is initialized, in addition to all nodes turning on their regular diverters, they also turn on their detectors. When barrier release occurs, the first node node's detector will remove the light from the waveguide and detect the presence of the light. Once detected, that node then immediately disables its detector, allowing the light to continue propagating down the waveguide to the next node. The process repeats until all nodes have detected a barrier release.

Figure 2 illustrates the tradeoffs. For latency, we looked at the critical timing between the last node to arrive and the last node to be notified. Our analysis assumes a Corona-like snake [10] with an 1.6 ns (8-cycle) single round-trip light-propagation latency. We assume that a node can cease detection within 1 clock edge of a 5 GHz clock. For a 64-node system, Figure 2 shows that, only when the number of participants is small does the unicast barrier perform competitively with the broadcast barrier.

For power, we use the power model described in [9]. We assume the interconnect is 26 cm long for 64 nodes and scale it linearly for smaller numbers of nodes. Broadcast's power increases at a much faster rate than unicast in part because the laser power must simultaneously be able to drive all nodes. At 64 nodes, broadcast uses over one-half a watt, approximately 12 times more power than unicast. As more nodes are added, this gap will widen.

3 Future Work and Conclusions

Since there can only be a limited number of physical hardware barriers, applications would either have to limit their use or they must be shared among multiple threads. Additionally, hardware barriers also do not readily cope with dynamic thread scheduling and thread migration. We plan to work on both problems by virtualizing our hardware barrier.

Barriers are a conceptually simple and useful synchronization construct for programmers. Unfortunately, they are expensive to implement efficiently in many-core chips. In this paper, we show how nanophotonic devices can be configured to provide low-cost and low-latency hardware barriers.

References

[1] C. J. Beckmann and C. D. Polychronopoulos. Fast barrier synchronization hardware. In *Supercomputing '90*, pages 180–189, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[2] P. Bryan, J. Beu, T. Conte, P. Faraboschi, and D. Ortega. Our Many-core Benchmarks Do Not Use That Many Cores. In *Workshop on Duplicating, Deconstructing, and Debunking (WDDD)*, 2009.

[3] M. Davis Jr and U. Ramachandran. A Distributed Hardware Barrier in an Optical Bus-Based Distributed Shared Memory Multiprocessor. In *Proceedings of International Conference on Parallel Processing*, pages 1–228, 1992.

[4] W. T.-Y. Hsu and P.-C. Yew. An effective synchronization network for hot-spot accesses. *ACM Trans. Comput. Syst.*, 10(3):167–189, 1992.

[5] 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 (extended abstract). In *SPAA '92*, pages 272–285, New York, NY, USA, 1992. ACM.

[6] J. Mellor-Crummey and M. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Transactions on Computer Systems (TOCS)*, 9(1):65, 1991.

[7] J. Sampson, R. Gonzalez, J. Collard, N. Jouppi, M. Schlansker, and B. Calder. Exploiting fine-grained data parallelism with chip multiprocessors and fast barriers. In *MICRO-39*, pages 235–246. IEEE Computer Society, 2006.

[8] S. L. Scott. Synchronization and communication in the t3e multiprocessor. In *ASPLOS '96*, pages 26–36, New York, NY, USA, 1996. ACM.

[9] D. Vantrease, N. Binkert, R. Schreiber, and M. H. Lipasti. Light Speed Arbitration and Flow Control for Nanophotonic Interconnects. In *MICRO-42*, 2009.

[10] 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 *ISCA-35*, pages 153–164, 2008.

[11] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *ISCA*, Jun 1995.

[12] Q. Xu, B. Schmidt, S. Pradhan, and M. Lipson. Micrometre-scale silicon electro-optic modulator. *Nature*, 435, May 2005.