Design of Large-Scale Symmetric Multiprocessors (SMPs) using Parallel Optical Interconnects

Ahmed Louri and Avinash Karanth Kodi
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ - 85721, USA
E-mail: louri@ece.arizona.edu

Abstract

In this paper, we address the primary limitation of bandwidth demands for address transaction in future cache coherent symmetric multiprocessors (SMPs). As a solution, we propose a scalable optical address sub-network called Symmetric Multiprocessor Network (SYMNET). SYMNET, not only has the ability to pipeline address requests, but also multiple address requests from different processors can propagate through the address sub-network simultaneously. This is in contrast to all electrical bus-based SMPs, where only a single request is broadcast on the physical address bus at any given point in time. The simultaneous propagation of multiple address requests in SYMNET increases the available address bandwidth and lowers the latency of the network, but the preservation of cache coherence can no longer be maintained with the usual fast snooping protocols. A modified snooping coherence protocol, Coherence in SYMNET (COSYM) is introduced to solve the coherence problem. We evaluated COSYM with a subset of Splash-2 benchmarks and compared it with the electrical bus-based MOESI protocol. Our simulation studies have shown a 5-66% improvement in execution time for COSYM as compared to MOESI for various applications. Simulations have also shown that the average latency for a transaction to complete using COSYM protocol was 5-78% better than the MOESI protocol. SYMNET improves system performance and scalability and that additional performance gains may be attained with further improvement in optical device technology.

Keywords
SMPs, optical interconnects, architectures, cache coherence.

1 INTRODUCTION

Symmetric multiprocessors (SMPs) dominate the server market as the most prevalent form of parallel processing commercially available. In SMPs, each address request is broadcast to all processors/memory modules connected to the network using a shared-bus. This address request is *snooped* by all the processors enabling simultaneous update or invalidation of cache blocks, thereby maintaining the caches coherent with low latency[1]. As the number of processors grows in the network, contention to acquire the bus also increases. The evolution of faster processors further aggravates the situation because the shared bus cannot run at speeds comparable to that of faster processors. This is because shared buses running at greater than 100Mhz face some fundamental problems such as wave reflection, impedance mismatch and parasitic capacitance which significantly limit the speed improvements[2], [3], [4]. Therefore the bus speed and the coherence overhead limit the rate at which address requests can be broadcast to all the processors/memory modules connected to the network[5], [6]. This in turn, limits the number of processors that can share the bus, affecting the scalability of SMP systems[6]. This address rate/bandwidth is the main scaling limit, which cannot follow the increasing demands of faster and large number of processors, limiting the scalability of shared-bus based SMPs.

In order to increase the address bandwidth, several techniques have been introduced. These techniques include split transaction buses[7], multiple address buses[6], physically separate address and data sub-networks[6] and moving from physically shared buses to logical buses which are implemented as point-to-point links. More aggressive solutions using multiple crossbars have been adopted to increase the address bandwidth by using a combination of snooping and directory cache coherence protocols in the FirePlane[8] design from Sun Microsystems. Directory protocols[9] are more scalable than snooping protocols, since the requests, responses (acknowledgements) and data responses need not be broadcast as in snooping protocols. Snooping protocols are more popular because they obtain data quickly (without indirection) and avoid the overhead of message sequencing as in directory protocols[10]. However, snooping protocols are limited to small systems as all transactions must be broadcast to all processors. New shared-memory architectures[8], [10] have moved away from implementing pure-snooping or pure-directory protocols to hybridization of cache coherence protocols by implementing both protocols within a single architecture model. Using current electrical technology, it remains a big challenge to have a large number of processors and at the same time implement fast, pure snooping cache-coherence protocols.

One technology that can provide high communication bandwidth, low latency and scalability is optical interconnection technology[2], [5]. The recent advances in optical in-

terconnect devices and packaging techniques such as multi-dimensional arrays of vertical cavity surface emitting lasers (VCSELs), arrays of photodetectors (PDs) and waveguide optics[11] are making optical interconnects a serious and potentially viable interconnect technology for parallel computing. The data transmission rate of a VCSEL is approximately $3-5$ Gbps. An array of such VCSELs enables address transmission with data rates in excess of $200-300$ Gbps[12], [13], [14]. This could satisfy the bandwidth demands of future SMPs[5]. Two unique properties of optics, namely unidirectional propagation and predictable path delays[15] are exploited in this paper to significantly reduce latency and increase the address bandwidth. Optical pulses can co-exist on the same optical line without interference if they are sufficiently separated. *This enables multiple address requests to propagate within the same waveguide/fiber simultaneously*. It can be argued that in electronic pipelined address buses, there could be several address requests in different phases of address translation such as bus arbitration, address transmission or waiting for the snoop response. *Our contention is that, in electronic SMPs, only a single address request is transmitted on the physical address bus at any given point in time*. Optically, we can easily have more than one address request in propagation simultaneously as discussed before. Moreover, scalability of optical interconnects depends on power budget constraints and not on coherence protocols as in electrical interconnects. These advantages provide us the impetus to look at optical technology to develop scalable SMPs with hundreds of processors while still using snooping cache coherence schemes.

Optical multiprocessors such as opto-electronic buses[15] photobus[16], U-bus[5], SPEED[17] and Lightning networks[18] have been previously reported. Optical networks mentioned above with the exception of the U-bus employ serial links to transmit address and data requests/responses using wavelength division multiplexing (WDM) technology. Moreover, directory cache coherence protocols are used to maintain coherency, which increases the latency as discussed before. U-bus extends the address bandwidth, but a new coherence protocol must be designed to maintain consistency across the caches. The optical solutions so far have not been able to integrate fast, pure-snooping cache coherence protocols and improve the address bandwidth demands to scale the architecture significantly.

This paper proposes an integrated solution to solve the address bandwidth requirements of large, scalable SMPs and still use fast snooping protocols to maintain cache coherence with low-latency using optical technology. An address sub-network, called Optical Symmetric Multiprocessor Network (SYMNET) using parallel optical interconnects is proposed using one-to-many communications. Parallel optical interconnects provide higher bandwidth-density product as compared to serial interconnects which provide higher bandwidth-distance product. SYMNET, not only has the ability to pipeline address requests, but also multiple address requests from different processors can propagate through the address sub-network simultaneously. The simultaneous insertion of multiple address requests complicates cache coherence. We have introduced a modified snooping coherence protocol, called Coherence in SYMNET (COSYM) and verified its correctness using several transient states. The use of transient states is not a new concept as it has been widely documented, but the transient states in our architecture is used to solve the write atomicity along with the snoop response requirements. COSYM relies completely on snooping protocols and avoids directory protocols altogether. SYMNET using the COSYM protocol is compared against electrical bus based systems using the MOESI protocol with Splash-2 benchmarks[19].

2 SYMNET:ADDRESS SUB-NETWORK

The proposed optical symmetric multiprocessor network, SYMNET is shown in figure 1. SYMNET consists of the processing elements/memory modules and an interconnection network. The interconnection network consists of two sub-networks; address and data sub-networks. The address and data sub-networks are separated, reducing the design complexity and enabling the design of large scalable SMPs. Several scalable data sub-networks have been reported[20], therefore this paper focuses only on the address sub-network. The address sub-network consists of two components, a transport part capable of transmitting multiple address requests and a control part which ensures collisionless transmission of these address requests. The transport part in SYMNET is implemented using bidirectional couplers/splitters and the control part is implemented using an optical token. In what follows, we describe the SYMNET address sub-network and then explain how the architecture is implemented.

The address sub-network follows a two-level hierarchical architecture design. The first level consists of grouping few processors on the boards using intra-board interconnections and the second level consists of interconnecting these boards by using inter-board interconnections. The inter-board and intra-board interconnections are constructed using bidirectional Y-splitter/coupler combination. Time division multiple access (TDMA) protocol is used as a control mechanism to achieve mutual exclusive access to the transport part. Several TDMA protocols such as pre-allocation-based protocols, reservation-based protocols with pre-allocated reservation control and token based TDMA protocols have been reported [18], [17]. In this paper, we consider an optical token based TDMA protocol with pre-allocation to prevent collision of address requests.

The basic building block of SYMNET address sub-network is shown in figure 2. The address sub-network is constructed using bidirectional Y-couplers, which provides two-way address transmission (see the inset in figure 2). The up-stream Y-couplers are used for combining the address requests from the processors. After reaching higher levels,

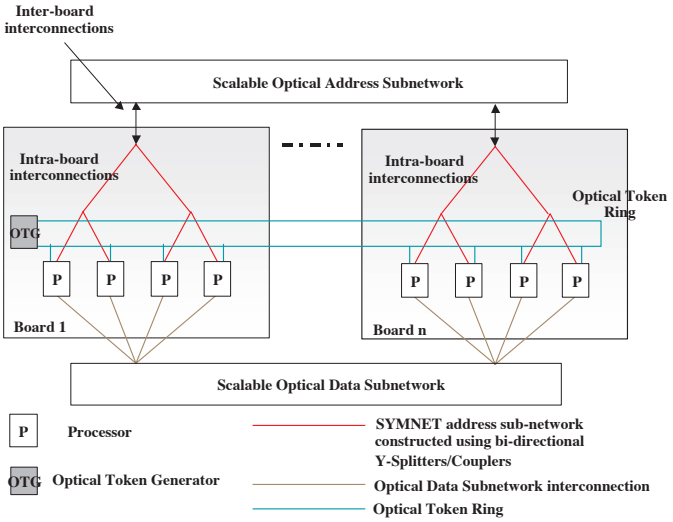

Figure 1. The proposed optical Symmetric Multiprocessor Network (SYMNET) in which processors are connected to two sub-network; address and data sub-networks.

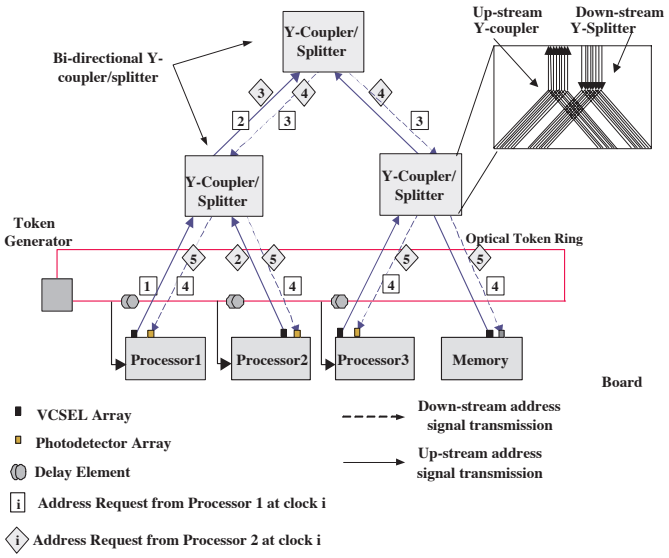

Figure 2. An overview of a single board in SYMNET. The processors are interconnected using bi-directional up-stream and down-stream Y-couplers/splitters.

this address request is re-routed through the downstream Y-splitters which enables broadcasting of the address requests to all the processors and memory modules. It should be noted that the broadcasting allows a request to reach all processors and memory at the same time. The optical token is a single pulse generated by the token generator. It provides a time reference for insertion of address requests into the sub-network by each individual processor. The optical token is tapped by the processor, which triggers the electronic interface to drive the address request. The token is delayed by using a delay element, which provides sufficient time to drive the electronics and also ensures that the address requests from successive processors are transmitted without collision. The address requests from different processors are pipelined which allows multiple requests to be propagating through the address sub-network. Using the properties of optics, namely unidirectional propagation and predictable path delays, it is possible in SYMNET to transmit multiple address requests *simultaneously on the same waveguide/fiber*. This is in contrast to all electrical shared-bus solutions where only a single address request is transmitted on the physical address buses at any given point in time. These address requests move up the hierarchy and then are re-transmitted back to all processors and memory simultaneously. This ensures that different requests from different processors are serialized in the global order of requests needed to maintain memory consistency.

The optical clock and the token generator are synchronized; thus successive processors receive the token every clock cycle. As shown in figure 2, in cycle 1 indicated by square shape, the optical token is received by processor 1, which transmits an address request. During cycle 2, when this address from processor 1 is in propagation at the next level of the address sub-network, the token is received by processor 2, which can transmit an address request. This is shown in the shape of a diamond in figure 2 . In cycle 3, the address request from processor 1 is being re-routed using the downward Y-splitter and at the same time the address request from processor 2 has moved up the address sub-network. The optical token is now received by processor 3, which can transmit an address. In cycle 4, the address request from processor 1 has reached all the processors, thereby the address request is broadcast to all the processors simultaneously. In cycle 5, the address request from processor 2 has reached all the processors. Broadcasting the address request results in simultaneous reception of the request by all the processors/memory modules enabling snooping of the same request, after which appropriate coherence action is taken as dictated by the snooping protocols. A detailed optical implementation of SYMNET using current optical devices has been submitted elsewhere.

3 CACHE COHERENCE IN SYMNET

Coherence in SYMNET, called COSYM is modified from the popular MOESI (*Modified, Owned, Exclusive, Shared, Invalid*)[21] protocol. In SYMNET, an address request is *issued* on a cache-miss. This request is *inserted* into the address sub-network when the optical token is received by the processor. The address request traverses through several Y-splitters and couplers, and then becomes *visible* to all processors simultaneously. In SYMNET, there is a fixed latency between the time when the address request is *inserted* into the address sub-network and when this request becomes *visible* to all processors. At the time of inserting an address request, there could be potentially other requests propagating through

the address sub-network affecting the same cache block for which the request is issued. COSYM protocol handles all race conditions that arises due to the simultaneous propagation of multiple address requests using several transient states. In what follows, we discuss the snoop response requirement, the working of COSYM protocol and how write-backs are handled.

3.1 Design Space for Snooping Protocols Implemented Optically

In the electrical networks, the snoop response is implemented using two wired-OR lines, *shared* and *owned*[9]. The processors sharing the block assert the shared line if the block is in the shared state or the owned line if the block is in any of the following states; E, M, or O. The shared snoop line could be asserted by more than one processor. In an optically interconnected multiprocessor system, if more than one pulse is inserted into the network as snoop response by multiple processors, collision of snoop responses from several processors result in erroneous response being received by the requestor as they operate at a single wavelength. Therefore, the constraint for the snoop response in our architecture is that it should be *a single response from a single processor*. To achieve this, we maintain an owner for every cache block shared. The owner is responsible for providing the snoop response. In case of a dirty block, the owner is the most recent processor which wrote to that block. In case of clean block, there could be several processors sharing the block. In order to determine a single owner, MOESI protocol is modified such that if a read miss request is issued to an E block, the block is upgraded to O, instead of S, thereby becoming the owner of the block. This does not change any other protocol constraints. Reads from the processor can still be satisfied and writes will still require an invalidation transaction to be issued. In the COSYM protocol, a single snoop response (HIGH or LOW) signal can determine all the relevant information required as follows:

Snoop High: Dirty block exists, memory need not respond to the requestor and if it was for a read request, the block is loaded in S state.

Snoop Low: No dirty block exists, memory responds with the data to the requestor and if it was for a read request the block is loaded in E state.

This simple strategy of responding with snoop results and recording of it, can lead to race conditions when simultaneous reads to the same block are issued by different processors. To illustrate this point, let us assume that processors P1, P2 and P3 share a block B and it is in E, I and I states in the three caches respectively. Processors P2 and P3 issue read miss requests to the same block B and P2's request becomes visible before P3's request. P2 becomes active after seeing its request and now begins reacting to other incoming requests. When P3's request becomes visible, there is a possibility that P2 may respond with the snoop signal. The reason why P2 may respond is, by default, a read-miss block is destined to be loaded in E state when the data is received. But, for this block, the owner, P1 does exist. This may result in both P1 and P2 responding with snoop response signals resulting in a potential race condition.

The solution for the race condition in reporting snoop response signals is that the processor should monitor all transactions after inserting a read-miss request and not when the request becomes visible. Therefore, in the above scenario, processor P3 should monitor transactions after inserting a read-miss request. When P2's read-miss request becomes visible, processor P3 makes note of P2's request by changing state such that the block will be loaded to S state upon data availability. Processor P2 should not respond to P3's request if its own snoop response is not received. At the same time, P2 should change state such that, if there is no snoop response, the block should be loaded in O state when the data is received and become the owner for the block. In the above example, P1 responds to P2's request by virtue of being in the E state and changes to O state. If there is no owner, processor P2 loads the block in O state when the data is available. For the write-miss case, race conditions do not arise, as the processor waits only for the address request to be serialized in the total order and not the snoop response.

We should note that it is very hard to verify a coherence protocol with many stable/transient states as it was indicated in[10]. In this paper, we attempt to provide a brief description of the verification methodology. The verification procedure for validating the coherence scheme is the table/graphical approach[10], [9] describing all the state transitions that occur when the processor issues new requests or when requests from other processors become visible. Our network is a completely ordered network, thereby providing a basis for snooping protocols to be implemented. The memory controller is also simplified with no dirty bit being present for every block. A transaction, once inserted into the network is assured to be completed without the processor having to retry the transaction. The responsibility of providing the snoop response rests solely with the owner of the block in our protocol. In case of a clean block, the owner is the cache which acquired the block first from the memory.

3.2 Implementation of COSYM Protocol

The cache controller reacts to two kinds of requests, issued either from the processor or from the interconnect. The address requests issued in SYMNET are read-miss, write-miss and upgrade/invalidation requests. The cache controllers make transitions based on their current state and current events. The transient state diagram, refer to figure 3, indicates the change in transient states according to different events. The events that cause the transitions are address request being issued from the processor, the request being inserted into the interconnect, the request becoming visible, receiving high/low snoop response for the request and finally receiving the data.

The various transient states in COSYM in case of a read

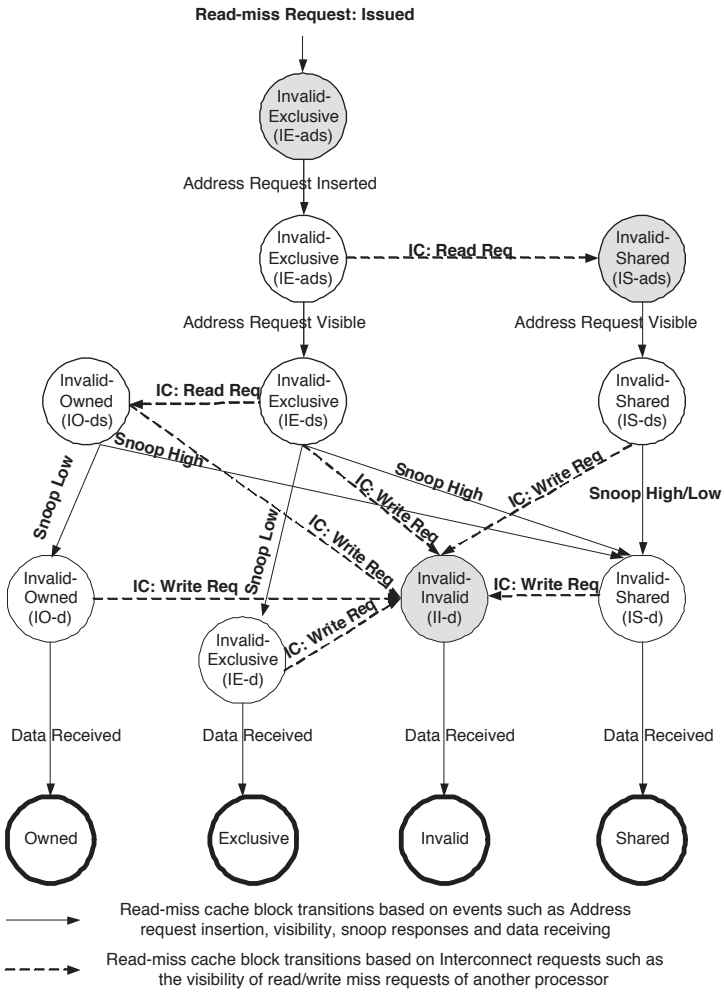

Figure 3. COSYM cache coherence protocol described using state diagram for all the transient states when a read miss occurs.

miss are shown in figure 3. The states indicated with gray shade imply that the protocol does not react to incoming address requests. The white circles, which are not bold, indicate the states in which the protocol reacts to transactions. The bold circles indicate the stable states. The text associated with the arrow from one state to another transient state indicates the event that caused the transition. Each transient state is indicated in the following manner[10]: *<present state>-<next state>-<abbreviation-"a/d/s">*. For example, when a read miss occurs, the transient state is indicated as: <Invalid-Exclusive (IE-ads)>. Invalid (I) indicates the present state, the next state is Exclusive (E) and "ads" stands for pending address, data and snoop response. When the address request is inserted, the cache is reactive to other requests issued to the same block. When the address request becomes visible, the state changes to <Invalid-Exclusive (IE-ds)> which indicates that the data and the snoop signal is pending. The other state reachable from IE-ads is IS-ads and IE-ds. IS-ads indicates that the processor has seen a read-miss request issued by another processor before its own request. The controller will not react to any other transaction until its own address request becomes visible. When the address request becomes visible, the transition takes place to either IE-ds or IS-ds depending on the previous state. If the block is in IE-ds and a read-miss request from another processor is visible, the block transits to IO-ds. All write-miss requests from the interconnect will result in the block being downgraded to II-d state. When the snoop signal is received, depending on whether it is high or low, the next possible states are IE-d, IO-d, IS-d or II-d. The snoop signal is not relevant when the state is in IS-<any> as the cache controller will load the block as shared irrespective of the snoop signal. Finally, when the data is received, the block makes the transition to E, S, O or I states depending on the previous transient state. A similar approach is adopted to verify the write-miss case and is not shown due to page limitation.

3.3 Write-back Handling

The snoop response is always provided by the owner of the block which informs both the memory whether to respond or not and the requesting processor whether to load the block in S or E state in case of a read-miss request. When the owned block itself is replaced, there are potential sharers in the system. So if a new request is issued, the caches sharing the block will not respond, memory responds and the issuing processor loads the block erroneously in the E state. Therefore there is a need to transfer the ownership of the block when the owned block is being replaced. The added advantage is that by transferring the ownership, no data transfer is required back to memory when a dirty block is evicted with potential sharers in the system. *This implies that the memory will be updated only when there are no sharers in the network*. In order to perform the above requirement, each cache block, in addition to tag, address bits and cache state, maintains *the next sharer for the block*. The next sharer is maintained only if the block is in either O or S state. If the block is in E or M state, then it is the only cached, valid copy in the system and there are no next sharers for the block. To illustrate this point, let's consider an example as shown in figure 4. Let us assume that P0 issues a read-miss request to block B as shown in figure 4(a). This is the first processor issuing the request, the block is loaded in the E state, the data is supplied by the memory and is indicated as P0(E). P0 has no next sharer for block B. In figure 4(b), P1 issues a read-miss request to block B, P0 provides the snoop high signal, supplies the data, sets the next sharer to P1, changes the state of block B to O and is indicated as P0(O). P2 loads the block in S state, has no next sharer and is indicated as P1(S). In figure 4(c), P2 issues a read-miss request to block B, P0 provides the snoop high signal, supplies the data, but does not set the next sharer. P1 sets the next sharer for the block B to P2. P2 loads the block in S state and has no next sharer. This continues, as new requests arrive, the last processor sets the next

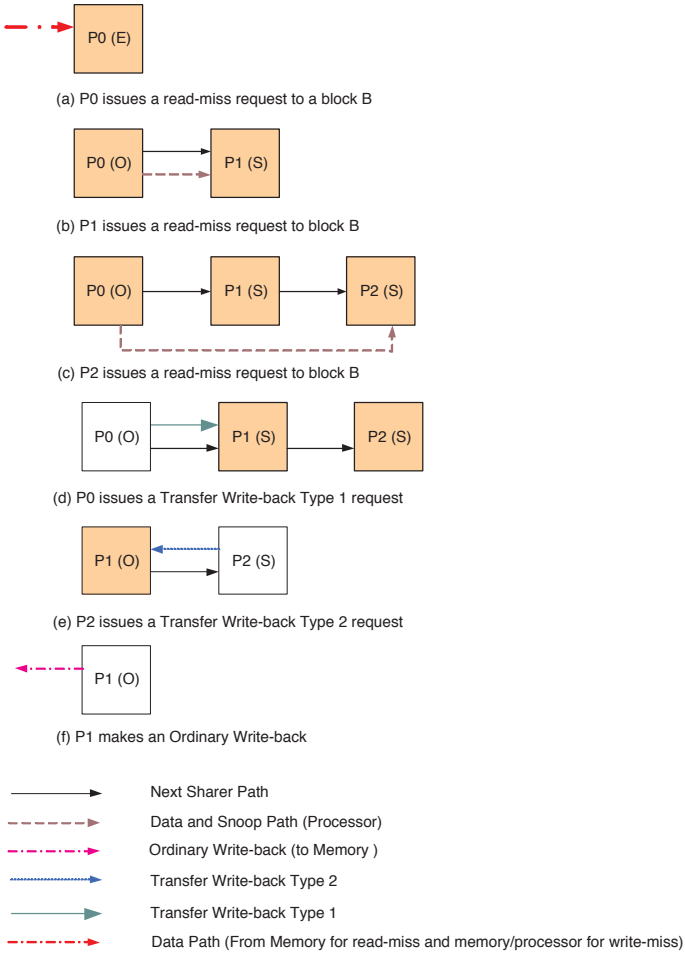

Figure 4. Figure 4(a) shows P0 issuing a read-miss request for block B, loads the block in E state and the data is supplied by memory. This is denoted as P0(E). In figure 4(b,c) processors P1 and P2 successively issue read-miss requests to the same block B. In figure 4(d,e,f), Transfer write-back Type 1, Type 2 and ordinary write-backs are shown.

sharer to be the processor that requested the block. A write-back to S/O block that transfers either the next sharer information or the ownership for the block constitutes a *transfer* write-back. We, therefore, define three types of write-back; transfer write-back Type 1, transfer write-back Type 2 and ordinary write-back.

Transfer Write-back Type 1: When an owned block is evicted from the cache, it transfers the ownership to the next sharer of the block, refer to figure 4(d). Here, the owner processor P0 transfers the ownership to processor P1. This is implemented by obtaining the optical token and inserting a transfer write-back type 1 request. This is acknowledged by processor P1.

Transfer Write-back Type 2: When a shared block is evicted from the cache, then the cache transfers the next sharer of the block to the previous sharer shown in figure 4(e). This is implemented by obtaining the token and inserting a transfer write-back type 2 request. In figure 4(e), P2 transfers the information of no sharer to P1. This is acknowledged by processor P1.

Ordinary Write-back: When a M/O block is evicted from the cache, then the block is written back to the memory. In figure 4(f), Processor P1 writes the victimized block B back to memory, since there are no sharers for the block. This does not cause another transaction to be inserted into the address sub-network. The evicted block in maintained in the write-back buffer, until the block is written back. Any intermediate transaction for this block will be responded by the controller as it is the owner of the block.

All transfer write-back transactions are acknowledged by the processor which changes the next sharer. When the transfer write-back transaction is snooped, no address request transaction is visible and no snoop response signal is needed. The snoop line can, therefore be used for acknowledgements when transfer write-back type 1 or 2 is snooped. An acknowledgement need not necessarily mean that the transaction has been completed. Race conditions may arise if simultaneous write-back requests for the same cache block is inserted by successive processors sharing the block. An algorithm has been devised to handle all subtle race conditions by re-issuing the transfer write-back request and has not been included here due to page limitations.

4 PERFORMANCE EVALUATION

We have chosen Limes (Linux Memory Simulator)[22], an event-driven execution simulator to evaluate the performance of SYMNET with electrical bus based systems considering realistic delays for address and data transactions. Limes models a single level cache and a blocking bus. We have extended the simulator to implement a two-level cache with a split transaction bus by merging or delaying conflicting requests for the electrical system. We assume that the data network and memory are contention-free to maximize the effects of the limited address bus bandwidth. The electrical SMP considered for comparison is similar in design to the Gigaplane and StarFire models[6], [23]. We do not compare SYMNET, a pure-snooping based SMP with the Fireplane[8] model as the Fireplane architecture follows a hybrid coherence model containing both the snooping and the directory-based protocols.

In this study, we use eight benchmarks, which cover a spectrum of memory sharing and access patterns from the SPLASH-2 suite[19], namely FFT with input data set $64K$ points; LU with 256×256,16×16 block; Ocean with 130×130; Radix with $1M$ integers,1024 radix; , Waternsquared with 512 molecules and Cholesky with $tk16.0$, to evaluate the performance of COSYM and MOESI protocols. We varied the number of processors from 2 to 32 to evaluate the performance of SYMNET. Each node of the simulated network contains 1 Ghz processor and has two cache levels, namely L1 and L2. The L1 cache is a 16 KByte

direct-mapped, with 32 byte block size and a write-through policy. The L2 cache is 64 KByte, 4-way set-associative with 32 byte block size and a write-back policy. Both the caches implement an LRU replacement policy. The access time to L2 cache is 4 processor clock cycles. Throughout this evaluation, we have considered processor clock cycles (pcc) as the base time unit for all measurements. All first-level cache read/write hits are assumed to take one processor clock cycle(pcc). The processor and the cache parameters are kept constant for both electrical and optical networks.

4.1 Electrical and Optical Simulation Parameters

In electrical SMPs, the address bandwidth is affected by several factors such as the bus speed, coherence protocol and the number of address buses. In electrical bus based SMPs[6], [7], [8], the processor clock is always a fraction of the system clock rate. For example, in the StarFire model, UltraSparc3 is clocked at $250Mhz$, where as the system bus is clocked at $83.3MHz$[6]. This implies that the system clock is around $1/3rd$ processor clock. With a 1GHz simulated processor, and an improved system clock, we assume that the system clock runs at $1/6th$ the processor clock. In the Gigaplane[23] architecture, it takes 2 cycles to broadcast a single address request. With the above assumption, it takes $12pcc$ to broadcast a single address request in our simulated address bus. This single address request per cycle (RPC) is denoted in all results as ($RPC = 1$). In the StarFire[6] architecture, processors snoop up to 2 address requests per cycle using 4 address buses. This case is simulated where each processor receives 2 address requests per cycle and is accomplished by reducing the number of cycles required to broadcast an address request to $6pcc$. This two address requests per cycle, is denoted in all results as ($RPC = 2$). Data network is contention-free and is implemented using a crossbar for both $RPC = 1$ and $RPC = 2$ cases. The number of cycles required for data transfer is fixed at 2 electrical network cycles irrespective of whether the memory or some cache responds as in StarFire[6] design. This results in $24pcc$ for data transfer in our simulated network for both the simulated electrical cases.

In SYMNET, the optical token is implemented such that the optical signal, generated by a laser source is split into two parts. One part of the optical signal is detected by the address port controller and the other part is delayed at the delay element implemented using a fiber loop. Now, we calculate the delay, D in transmitting an address request into the address sub-network by the address port controller. This delay should account for the signal detection, opto-electronic conversion and the rise time of address pulses driven by VCSEL arrays. The delay D, is given by:

$$D = \frac{S_p}{v_c} + 2.O_e + G_d + \frac{b}{m.V_d} \quad (1)$$

where S_p is the distance of separation between the delay element at processor n and the detector at processor $n+1$, v_c is the velocity of light in fibers, O_e is the latency of opto-electronic conversion, G_d is the gate delay faced by the token at the address port controller, m is the number of parallel links, b is the number of address bits (including one bit for snoop response) and V_d is the VCSEL data rate. O-E conversion takes place when the optical signal is detected by the address port controller and E-O conversion takes place when the address bits encoded as optical pulses are driven by the VCSEL array. It is assumed that a single gate delay is seen by the address port controller when it receives the token. Assuming that $S_p = 4cm$, $v_c = 2 \times 10^8$, $O_e = 75psec$, $G_d = 0.2nsec$, $V_d = 3Gbps$ and with $m = b$, D is estimated to be $0.88nsec$. The optical token should be seen by the next processor with a delay greater than $0.88nsec$ to prevent collision of address requests. Therefore, the other part of the optical signal at the delay element should be delayed by more than $0.88nsec$. Adding guard time to D, we assume the delay to be $1nsec$. Considering $1nsec$ as the required delay, we can estimate the length of the delay element to be $20cm$ ($= (2 \times 10^8) \times 1nsec$). The delay element is implemented by using a fiber loop 20cms in length. Therefore, the time taken by each processor to insert its address request is estimated to be $1nsec$ or $1pcc$. The delay encountered by a address transaction to be visible is equivalent to the number of stages in the address sub-network. This is assumed to be twice the logarithm of the number of processors connected in the address sub-network. The snoop response also takes similar number of cycles after the address is snooped. The data network considered for SYMNET is the optical crossbar[20] network. The data sub-network for SYMNET is also assumed to be conflict-free. The delay in data transfer for the optical network depends on the data rate of current multi-wavelength VCSEL arrays. At 5 Gbps VCSEL data rate, to transmit 32 byte block, it takes around 52nsec (= (32×8) / (5×10^9)) and this corresponds to $52pcc$.

4.2 Simulation Results

We determined the normalized execution time and the average delay for each read/write miss from the L2 cache for SYMNET and the electrical bus-based SMP varying from 2 to 32 processors.

Normalized Execution Time: Figure 5 shows the normalized execution time for varying number of processors for different applications. Normalized execution time is calculated by considering the maximum number of simulated cycles for a given application and given number of processors and dividing the simulated cycles for the other cases with the maximum number of cycles. For FFT, COSYM shows 25% improvement over MOESI protocol for $RPC = 1$ and 8% improvement for $RPC = 2$ running for 32 processors. For the LU application, the improvement in execution time is around 30-35% for both the cases. The best improvement is visible for Ocean application, where the improvement is over 62% for $RPC = 1$ and over 52% for $RPC = 2$ running for 32 processors. The improvement in performance for

radix is 38% for $RPC = 1$ for 32 processors. Cholesky and Water-nsquared applications show lower improvement in performance for COSYM protocol, with cholesky showing an improvement of 5% for $RPC = 1$ and water showing an improvement of 16-19% for both the cases. Cholesky showed higher transfer write-backs for 32 processors which increased the write-back time. This affected the execution time and showed better performance for MOESI protocol with $RPC = 2$ than COSYM protocol by almost 5%.

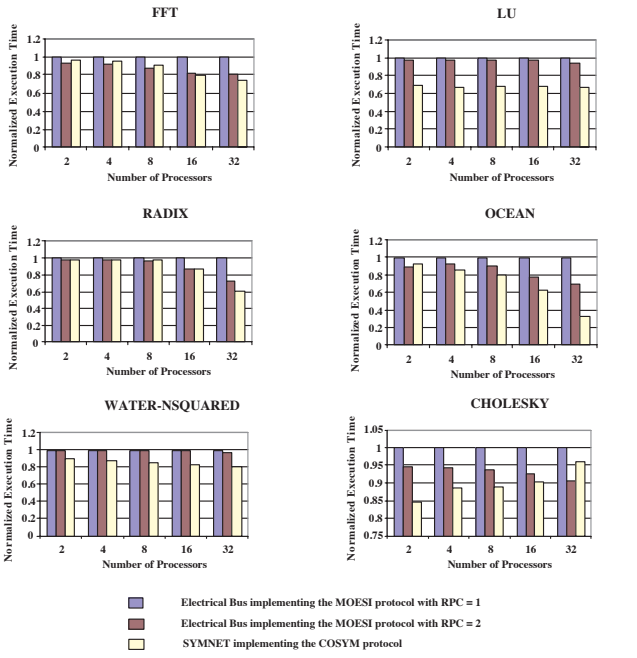

Figure 5. Normalized Execution time for processors varying from 2 to 32 for various Splash-2 benchmarks. The execution time for the electrical bus implementing the MOESI protocol with a single address request per cycle ($RPC = 1$), two address requests per cycle ($RPC = 2$) and the SYMNET address sub-network implementing COSYM protocol is shown. Normalized execution time is calculated for a given number of processors and a given application by considering the maximum number of simulated cycles and dividing the remaining two cases with the maximum value.

Average Latency: Figure 6 shows the normalized average delay for a transaction to be completed for both the electrical and the optical case. The delay in completing a transaction was calculated from the time the read/write miss was received by the L2 cache to the time the data was received by the L2 cache for each processor. The ratio of the total number of transactions to the total time consumed for all processors was used to determine the average delay. This delay was then normalized by considering the maximum average delay for a given application and then dividing all the remaining cases with that value. The average delay was much higher for the electrical case for all applications which increased linearly with the number of processors. The delay for $RPC = 1$ case was higher than $RPC = 2$ case as expected. Most applications showed lower delay for MOESI protocol for smaller configurations. As the number of processors increased, COSYM outperformed both the electrical cases. This is directly attributed to the saturation of the electrical bus. As the number of processors increases in the interconnect, the delay to acquire the bus also increases, thereby increasing the latency for a transaction to complete. The reduction in latency for FFT using the COSYM protocol is as high as 76% for $RPC = 1$ and 57% for $RPC = 2$. For LU, the reduction in latency for COSYM protocol ranged from 51% for $RPC = 1$ to 25% for $RPC = 2$. For Cholesky application, the latency for COSYM was a fraction (2%) better than MOESI with $RPC = 2$ condition for the same reason explained above.

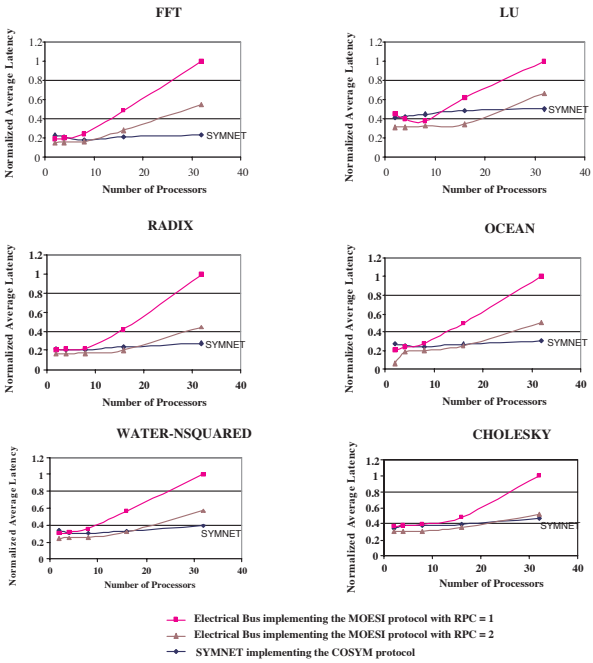

Figure 6. Normalized Average latency of a transaction for processors varying from 2 to 32 for various Splash-2 benchmarks. The average latency for the electrical bus implementing the MOESI protocol with a single address request per cycle ($RPC = 1$), two address requests per cycle ($RPC = 2$) and the SYMNET address sub-network implementing COSYM protocol is shown. Normalized average latency is calculated by considering the maximum value of average latency for a given application and dividing all the remaining cases with the maximum value.

In COSYM protocol, transfer write-backs use the address sub-network to insert write-back transactions. This is in contrast to the electrical case which uses the data sub-network for write-backs. Transfer write-backs, therefore require lesser number of cycles to complete in COSYM. The

overhead in the COSYM protocol is the transfer write-back Type 2 where, even a shared block evicted from the cache needs to inform the previous processor of the change in the next sharer. Our simulation studies have shown that in COSYM, the write-back time is dominated by the time required to evict the M/O block from the cache. The write-back latency in COSYM exceeds MOESI, and this is due to the fact that the write-back time for COSYM is almost twice as that for MOESI protocol. The average latency for a cache-miss transaction to complete is lower for the COSYM protocol. This is attributed to the SYMNET architecture which pipelines multiple address requests from different processors. The improvement in the average latency for cache-miss request offsets the increase in write-back latency in COSYM protocol. This results in improved execution time for COSYM as compared to electrical MOESI protocol as our simulation studies have shown.

When a transfer write-back transaction does not complete due to race conditions, the write-back transaction is re-issued. Our simulation studies for various applications and varying processors have shown that re-issuing of transactions is negligible compared to the total number of transfer write-backs that take place. For FFT, running for 32 processors, the number of re-issued transactions is only 0.002% of the total number of requests inserted into the address sub-network. Hence, the race condition which cause the write-back request to be re-inserted occurs for a very small percentage of write-backs.

The theoretical power budget analysis has shown that this architecture can scale up to 128 processors using current optical device technology, while still using fast snooping-based cache coherence protocols. This is significant improvement considering the largest pure-snoopy electrical SMP can support 64 processors[6]. The simulation results clearly indicate that the proposed optical SYMNET with COSYM as the coherence protocol provide a better support for scalable SMPs than their electrical counterparts.

5 CONLCUSION

In this paper, we addressed the primary limitation of address bandwidth in SMPs. As a solution, we propose a parallel optical interconnect based Symmetric Multiprocessor Network (SYMNET) and a modified cache coherence protocol called COSYM. SYMNET improves execution time and reduces the latency by pipelining multiple address requests from different processors simultaneously. Using the modified Limes simulator, we simulated SYMNET implementing the COSYM protocol from 2 to 32 processors. The improvement in execution time was seen for all applications ranging from 5% for cholesky to 66% for ocean. The average latency for the transaction also decreased by as much as 78% for various applications of the Splash-2 benchmarks.

Our objective to implement the snoop response signals optically resulted in handling write-backs differently with ownership transfer. In SYMNET, the overhead is to maintain the next sharer for every shared cache block. This concept has been previously introduced in multicast snooping protocol for mask prediction. Transfer write-back Type 1 improves system performance as the faster address sub-network is used instead of the slower data crossbar. Transfer write-back Type 2 affects system performance as even a block in S state needs to inform the change in sharing information. Techniques such as speculating the next sharer or random choosing a processor as the next sharer are currently being studied to reduce the number of Transfer write-back Type 2. The improvement in latency for an address transaction offsets the write-back latency, resulting in better performance for the COSYM protocol. The simulation results provide encouragement that SYMNET has the potential to match the bandwidth needs of future SMPs. Parallel optical interconnects and integrated waveguide technology makes SYMNET a viable solution for SMPs with distinct cost and performance advantages over traditional electronics. Greater improvements in terms of bandwidth, latency and scalability can be expected with further improvement in optical device technology.

Acknowledgement

This research is sponsored by NSF grant no. CCR-0000518. We would like to thank Prof.D.Litaize and Prof.J.Collet for first pointing out the limitations of SMPs in a quantitative manner, and for numerous comments and suggestions on this work.

References

[1] James R. Goodman, "Using cache memory to reduce processor-memory traffic," in *Proceedings of the 10th International Symposium on Computer Architecture*, June 1983, pp. 124–131.

[2] David A.B.Miller, "Rationale and challenges for optical interconnects to electronic chips," *Proceedings of the IEEE*, vol. 88, pp. 728–749, June 2000.

[3] Fong Pong, Michel Dubois, and Ken Lee, "Design and performance of smps with asynchronous caches," Tech. Rep. HPL-1999-149, Hewlett Packard, HP Laboratories Palo Alto, November 1999.

[4] H. S. Stone and J. Cocke, "Computer architecture in the 1990s," *IEEE Computers*, vol. 24, no. 9, pp. 30–38, Jan-Feb 1991.

[5] Jacques Henri Collet, Wissam Hlayhel, and Daniel Litaize, "Parallel optical interconnects may reduce the communication bottleneck in symmetric multiprocessors," *Applied Optics*, vol. 40, pp. 3371–3378, 2001.

[6] Alan Charlesworth, "Starfire: Extending the smp envelope," *IEEE Micro*, vol. 18, no. 1, pp. 39–49, Jan-Feb 1998.

[7] Mike Galles and Eric Williams, "Performance optimizations, implementation and verification of the sgi challenge multiprocessor," in *Proceedings of 27th Annual Hawaii International Conference on Systems Sciences*, 1996, pp. 134–143.

[8] Alan Charlesworth, "The sun fireplane smp interconnect in the sunfire 3800-6800," in *Hot Interconnects 9*, August 2001, pp. 37–42.

[9] David E. Culler, Jaswinder Pal Singh, and Anoop Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, San Fransisco, 1999.

[10] Daniel J. Sorin, Manoj Plakal, Anne E. Condon, Mark D. Hill, Milo M. Martin, and David A. Wood, "Specifying and verifying a broadcast and a multicast snooping cache coherence protocol," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, 2002.

[11] Y. S. Liu, R. J. Wojnarowski, W. A. Hennessy, P. A. Piacente, J. Rowlette, J. Stack, M. Kader-Kallen, Yue Liu, A. Peczalski, A. Nahata, and J. Yardley, "Plastic vcsel array packaging and high density polymer waveguides for board and backplane optical interconnect," in *Proceedings of the Electronic Components and Technology Conference*, 1998, pp. 999–1005.

[12] A.V.Krishnamoorthy, K.W.Goossen, L.M.F.Chirovsky, R.G.Rozier, P.Chandramani, S.P.Hui, J.Lopata, J.A.Walker, and L.A.D'Asaro, "16 x 16 vcsel array flip-chip bonded to cmos vlsi circuit," *IEEE Photonics Technology Letters*, vol. 12, no. 8, pp. 1073–1075, August 2000.

[13] Yue Liu, "Heterogeneous integration of oe arrays with si electronics and micro-optics," in *Proceedings of the Electronic Components and Technology Conference*, 2001, pp. 864–869.

[14] Y. S. Liu, R. J. Wojnarowski, W. A. Hennessy, J. P. Bristow, Yue Liu, A. Peczalski, J. Rowlette, A. Plotts, J. Stack, M. Kader-Kallen, J. Yardley, L. Eldada, R. M. Osgood, R. Scarmozzino, S. H. Lee, V. Ozgus, and S. Patra, "Polymer optical interconnect technology (point)-optoelectronic packaging and interconnect for board and backplane applications," in *Proceedings of the Electronic Components and Technology Conference*, 1996, pp. 308–315.

[15] Donald M. Chiarulli, Stephen P.Levitan, Rami G. Melhem, Manoj Bidnurkar, Robert Ditmore, Gregory Gravenstreter, Zicheng Guo, Chungming Qiao, Majd F. Sakr, and James P. Teza, "Optoelectronic buses for high-performance computing," in *Proceedings of the IEEE*, 1994, pp. 1701–1710.

[16] Paul Lukowicz, "The photobus smart pixel interconnection system for symmetric multiprocessing using workstation clusters," in *6th International Conference on Parallel Interconnects*, 1999, pp. 106–113.

[17] Joon-Ho Ha and T.M.Pinkston, "The speed cache coherence for an optical multi-access interconnect architecture," in *Proceedings of the 2nd International Conference on Massively Parallel Processing Using Optical Interconnections*, 1995, pp. 98–107.

[18] Patrick Dowd, James Perreault, John Chu, David C. Hoffmeister, Ron Minnich, Dan Burns, Frank Hady, Y. J. Chen, and M. Dagenais, "Lighnting network and systems architecture," *Journal of Lightwave Technology*, vol. 14, pp. 1371–1387, 1996.

[19] C.S.Woo, M.Ohara, E.Torrie, J.P.Singh, and A.Gupta, "The splash-2 programs: Characterization and methodological considerations," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, June 1995, pp. 24–37.

[20] Brian Webb and Ahmed Louri, "A class of highly scalable optical crossbar-connected interconnection networks (socns) for parallel computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 1, pp. 444–458, May 2000.

[21] Sweazey P. and A.J.Smith, "A class of compatible cache consistency protocols and their support by the ieee futurebus," in *Proceedings of the 13th Annual International Symposium on Computer Architecture*, May 1986, pp. 414–423.

[22] Igor Ikodinovic, Aleksander Milenkovic, Veljko Milutinovic, and Davor Magdic, "Limes: A multiprocessor simulation environment for pc platforms," in *PPAM*, September 1999.

[23] Ashok Singhal, David Broniarczyk, Fred Cerauskis, Jeff Price, Leo Yuan, Chris Cheng, Drew Doblar, Steve Fosth, Nalini Agarwal, Kenneth Harvey, Erik Hagersten, and Bjorn Liencres, "Gigaplane: A high performance bus for large smps," in *Proceedings Hot Interconnects 4*, 1996, pp. 97–112.