An Optical Interconnection Network and a Modified Snooping Protocol for the Design of Large-Scale Symmetric Multiprocessors (SMPs)

Ahmed Louri, *Senior Member*, *IEEE*, and Avinash Karanth Kodi, *Student Member*, *IEEE*

Abstract—In Symmetric Multiprocessors (SMPs), the cache coherence overhead and the speed of the shared buses limit the address/snoop bandwidth needed to broadcast transactions to all processors. As a solution, a scalable address subnetwork called Symmetric Multiprocessor Network (SYMNET) is proposed in which address requests and snoop responses of SMPs are implemented optically. SYMNET not only uses passive optical interconnects that increases the speed of the proposed network, but also pipelines address requests at a much faster rate than electronics. This increases the address bandwidth for snooping, but the preservation of cache coherence can no longer be maintained with the usual snooping protocols. A modified coherence protocol, Coherence in SYMNET (COSYM), is introduced to solve the coherence problem. COSYM was evaluated with a subset of Splash-2 benchmarks and compared with the electrical bus-based MOESI protocol. The simulation studies have shown a 5-66 percent 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 percent better than the MOESI protocol. It is also seen that SYMNET can scale up to hundreds of processors while still using fast snooping-based cache coherence protocols, and additional performance gains may be attained with further improvement in optical device technology.

Index Terms—SMPs, parallel optical interconnects, cache coherence, scalable optical networks.

1 Introduction

Symmetric multiprocessors (SMPs) are attractive parallel computers widely used, since they provide a global physical address space and a symmetric access to the entire memory with increased flexibility and programmability [1], [2], [3]. SMPs use fast snooping protocols to maintain cache coherence by broadcasting every request on the shared bus connecting all modules. Contention to acquire the bus in addition to faster processing capabilities of current processors degrades the performance of the shared-bus. As the bus speed increases, the processor boards connected to the bus behave as stubs resulting in reflections of bus signals [3]. Other fundamental problems such as impedance mismatch, stray capacitance, $\triangle I$ noise, and crosstalk [3], [4] significantly affect the speed improvements of the shared bus. Therefore, the bus speed and the coherence overhead limit the rate at which address requests can be broadcast to all the processors/memory modules [5], [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.

1.1 Electrical Solutions for the Address Bandwidth Problem

Several techniques have been used to increase the address bandwidth in SMPs that include split transaction buses [7], multiple address buses [6], and physically separate address and data subnetworks [6]. Current commercial implementations have replaced the shared-bus topology by implementing each bus with a pipelined broadcast tree constructed from point-to-point links [6]. 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 [2] design from Sun MicroSystems. Asynchronous caches [3] are implemented by using a deeply pipelined memory system with parallel-link interconnection and queues through which the memory and the processors communicate. Timestamp snooping protocol [8] reduces the execution time by broadcasting requests over an indirect interconnection network. As seen above, shared-bus topology is currently being replaced by scalable interconnection networks so that when the system is scaled, additional processors are incorporated into the network by simply using more commodity switching chips. Unfortunately, there are a limited number of data ports available per switching chip, and more switching chips are usually added to accommodate for the additional point-to-point connections. As a consequence, a significant delay occurs in the additional switching stages, which in turn dramatically increases the latency. It has been reported that scaling from a medium-scale multiprocessor to a large-scale multiprocessor increases memory access latency by 60 percent. Switching/routing delays alone account for almost 40 percent of the increased latency. Additional switching stages also result in further slowing the clock, for example, even with a changed coherence protocol, the system clock in the Fireplane architecture from Sun Microsystems runs at a rate of 150 Mhz, where as the UltraSparc 3 processor is clocked over 750Mhz [2]. In addition, the effective delay on electrical lines depends on temperature. For example, copper's resistivity changes by 40 percent over 100 degree centigrade. The rise time on electrical lines typically is proportional to the

• *The authors are with the Department of Electrical and Computer Engineering, University of Arizona, 1230 E. Speedway Blvd., Tucson, AZ 85721. E-mail: {louri, avinashk}@ece.arizona.edu.*

Manuscript received 26 Nov. 2002; revised 16 Mar. 2004; accepted 21 May 2004.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 117845.

resistance of the line (this applies to both RC and LC transmission lines), and the signal delay on RC lines is proportional to the resistance. Moreover, implementing broadcast algorithms using electrical interconnects becomes more complex and expensive as the number of processors increases in the system. Therefore, by using current electrical interconnect technology, it remains a big challenge to have a large number of processors, and at the same time implement fast broadcast snooping cache coherence protocols.

1.2 Optical Interconnects for Address Bandwidth Limitation

One technology that can provide high communication bandwidth, low latency, and scalability is optical interconnection technology [9], [4], [5]. The recent advances in optical interconnect devices and packaging techniques such as multidimensional arrays of vertical cavity surface emitting lasers (VCSELs), arrays of photodetectors (PDs), and waveguide optics [10] are making optical interconnects a serious and potentially viable interconnect technology for parallel computing. The data transmission rate of a VCSEL is approximately $3\text{-}5Gb/s$. An array of such VCSELs enables address transmission with data rates in excess of $200\text{-}300Gb/s$ [11], [12], [10], [19]. This could satisfy the bandwidth demands of future SMPs. Optical pulses can coexist on the same optical line without interference if they are sufficiently separated. Even though requests can be pipelined by electrical interconnects, the rate at which requests from successive processors can be pipelined using optical technology is much greater and can be achieved without using any expensive multiplexer/demultiplexer circuits. In addition, the delay on optical signal or clock paths (fibers or waveguides) is not strongly dependent on temperature as in the case of electronics, and signal or clock edges do not degrade substantially over the scale of a computer room. It is likely possible to retain absolute timing accuracy in the delivery of optical signals of 10-100 ps over several meters [4]. Moreover, optical interconnect-based systems are easily scalable either by adding extra wavelengths or fiber channels in order to incorporate additional processors. 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 subnetwork, called Optical Symmetric Multiprocessor Network (SYMNET) using parallel optical interconnects, is proposed with one-to-many communications. Parallel optical interconnects provide higher-bandwidth density product as compared to serial interconnects which provide higher bandwidth product. SYMNET not only uses passive optical interconnects that increases the speed of the proposed network, but also pipelines address requests at a much faster rate than electronics. By using passive optical technology as opposed to more complex optical switching networks, the optical signal transfer in SYMNET is much faster since there is no optical switching or conversion. This results in increasing the address bandwidth for snooping and provides the impetus to develop scalable SMPs with hundreds of processors using optical interconnects, while using snooping cache coherence schemes. However, the preservation of cache coherence can no longer be maintained with the usual snooping protocols. We have introduced a modified snooping coherence protocol, called Coherence in SYMNET (COSYM), and verified its correctness using several transient states. SYMNET using the COSYM protocol is compared against electrical bus-based systems using the MOESI protocol with Splash-2 benchmarks [13].

1.3 Related Work

Optical bus-based multiprocessor systems using coincident pulse technique provide optical solutions to the problems of bus design in areas of address bus arbitration, device addressing, and data transfer [14]. However, the problem of cache coherence is not addressed. The photobus smart pixel interconnection system for shared-memory multiprocessors use optical buses for broadcasting the address requests, but arbitration is implemented using electronic buses leading to buffering of address requests at the smart pixel VLSI chip [15]. The constraints of access arbitration is eliminated in the U-bus [5] design for SMPs. U-bus extends the address bandwidth, but a new coherence protocol must be designed to maintain consistency across the caches. In the SPEED [16] architecture, write requests are broadcast using the snooping protocol and read requests are unicast using the directory protocol. The I-SPEED coherence protocol used for this architecture implements a single owner for dirty blocks to preserve the consistency of caches. Optical networks discussed so far employ serial links to transmit address requests and data responses between the source and the destination using wavelength division multiplexing (WDM) technology. 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.

2 SYMNET: ADDRESS SUBNETWORK

The proposed optical symmetric multiprocessor network, SYMNET is shown in Fig. 1. Fig. 1a shows SYMNET consisting of the processing elements/memory modules and an interconnection network. The interconnection network consists of two subnetworks: address and data subnetworks. The address and data subnetworks are separated, reducing the design complexity and enabling the design of large scalable SMPs. Scalable data subnetworks have been studied elsewhere [17], therefore this paper focuses only on the address subnetwork. The address subnetwork 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 subnetwork and then explain how the architecture is implemented.

The address subnetwork follows a two-level hierarchical architecture design. The first level consists of grouping few processors on the boards using intraboard interconnections and the second level consists of interconnecting these boards by using interboard interconnections. The interboard and intraboard 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 preallocation-based protocols, reservation-based protocols with preallocated reservation control, and token-based TDMA protocols have been reported [18], [16]. In this

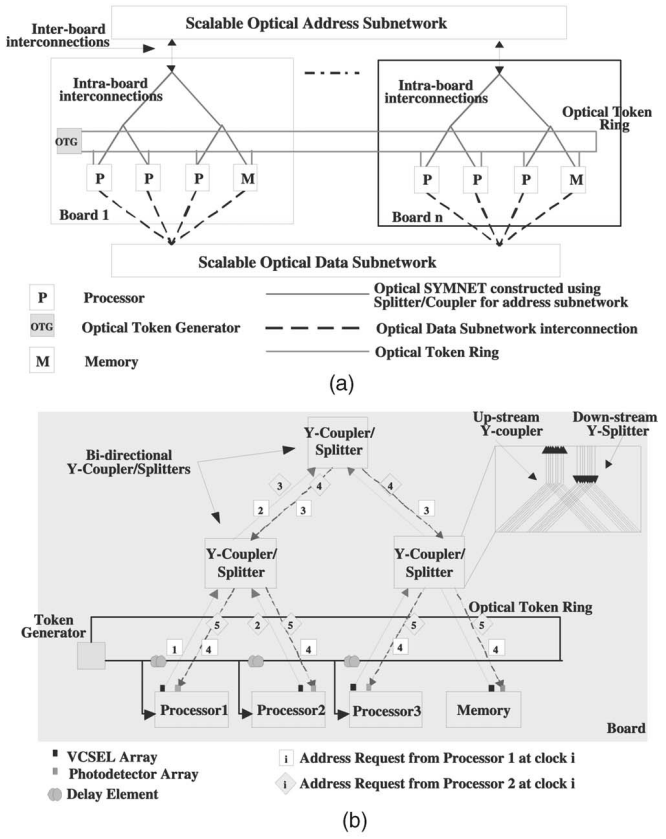

Fig. 1. (a) The proposed optical Symmetrical Multiprocessor Network (SYMNET) in which the processors are connected to two subnetworks: address subnetwork and data subnetwork. (b) An overview of a single board of SYMNET, which shows the interconnections for four processors. The processors are connected to bidirectional Y-couplers by optical waveguides/fibers.

paper, we consider an optical token-based TDMA protocol with preallocation to prevent collision of address requests.

The basic building block of SYMNET address subnetwork is shown in Fig. 1b. The address subnetwork is constructed using bidirectional Y-couplers, which provides two-way address transmission (see the inset in Fig. 1b). The upstream Y-couplers are used for combining the address requests from the processors. After reaching higher levels, this address request is rerouted 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. This is a very useful feature for the cache coherence protocol design to be discussed later. The optical token is generated by a high-powered ($10mW$) and high-frequency ($10Ghz$) laser source. This optical signal is split (50/50), such that one part of the signal (50 percent) is received by the address port controller and the other part (50 percent) is delayed by the delay element implemented using a fiber of length $20cms$. This could lead to significant losses in the token ring. One way to avoid the effects of these losses, is to regenerate the token signal for every board (containing up to four processors). The optical token provides a time reference for insertion of address requests into the subnetwork 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. These address requests move up the hierarchy and then are retransmitted 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 [1].

The optical clock and the token generator are synchronized; thus, successive processors receive the token every clock cycle. As shown in Fig. 1b, 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 subnetwork, the token is received by processor 2, which can transmit an address request. This is shown in the shape of a diamond in Fig. 1b. In cycle 3, the address request from processor 1 is being rerouted using the downward Y-splitter and, at the same time, the address request from processor 2 has moved up the address subnetwork. 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. 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. In cycle 5, the address request from processor 2 has reached all the processors. At the same time, processors snoop on the request of processor 1 received in cycle 4. Similar to electrical implementations, cache access priority is with address subnetwork controller rather than the processor-side controller. The snoop response, if one exists will be provided by only a single processor as is explained later. This response will be broadcast in the next cycle, i.e., cycle 6 using the dedicated snoop line shown in Fig. 2a. In addition, in cycle 6, processors receive the request from processor 3, snoop on the request from processor 2, and respond to this request in cycle 7. This process repeats in a pipelined fashion, processors receive an address request, snoop on the previously received request, and transmit the snoop response in the next cycle. The snoop response for processor 1 is received in cycle 9, for processor 2 is received in cycle 10, and so on. Depending on the snoop response, either the memory or the owner of the block will respond with data. The processor that requested the block, maintains a count down timer to know the correct cycle in which it expects to receive the snoop response.

2.1 SYMNET Implementation Details

The key component of SYMNET is the VCSEL/PD arrays capable of transmitting at data rates in excess of $3Gb/s$ per channel, which results in providing aggregate data rates of several Gbps. High-performance GaAs and InGaAs-based selectively oxidized or proton implanted top-emitting, bottom-emitting VCSEL arrays emitting at $780nm$ to $980nm$ have been widely reported in the literature [10], [19], [20], [12]. Optical polymers are increasingly considered as highly versatile elements that can be readily transformed into single-mode, multimode, and microoptical waveguide/fiber structures as they exhibit excellent thermal stability, low optical loss, mechanical robustness, and have demonstrated capability in a variety of demanding applications. The low loss in optical polymers [10] makes them an attractive material for constructing the 2×1 couplers, 1×2 splitters,

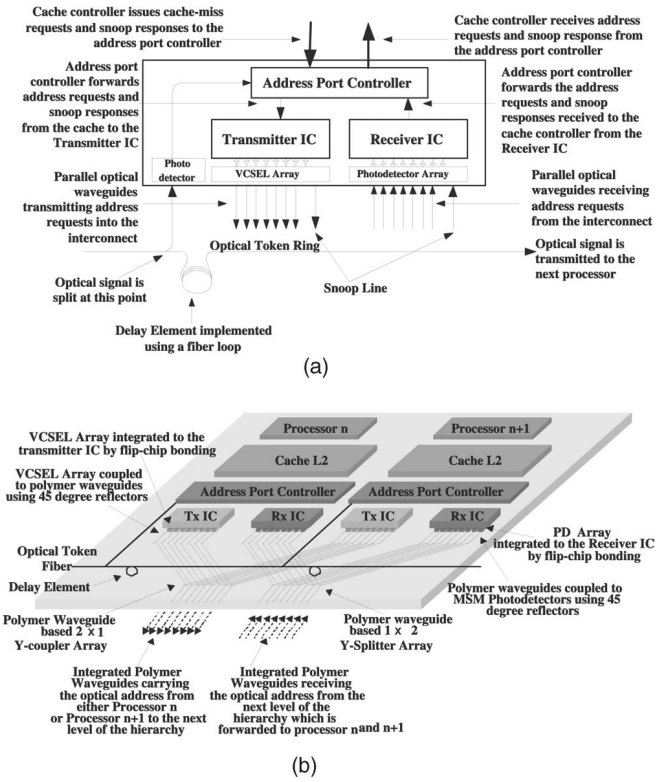

Fig. 2. A Functional block diagram of the processor, cache, cache controller, address port controller, optical token ring, delay element, VCSEL, and Photodetector arrays. (a) Functional block diagram specifying the address port controller, VCSEL, and Photodetector arrays. (b) Interboard interconnection between two adjacent processors n and n + 1.

and for routing optical pulses from VCSELs to these couplers/splitters in the SYMNET interconnection network. Details of the optical components and their implementation can be found in a separate publication [21].

The implementation details of SYMNET are shown in Fig. 2. In Fig. 2a, on a cache-miss, the address request is forwarded to the address port controller by the cache controller. The optical token is generated by a high-powered laser source. This optical signal is split, such that one part of the signal is received by the address port controller and the other part is delayed by the delay element implemented using a fiber loop. When the token is received, the address port controller forwards the address request to the transmitter IC which drives the VCSEL arrays. Optical interconnects based on complimentary-metal-oxide-semiconductor CMOS/VCSEL technology have been widely proposed for high-performance computing applications [12]. A similar, hybrid integration [20] using flip-chip bonding of CMOS-VCSEL arrays is followed in our design. The optical pulses emitted from the VCSEL array propagate through integrated polymer 2×1 up-stream couplers reach the root of the address subnetwork and is transmitted back using 1×2 down-stream splitters. The address bits encoded as light pulses are eventually detected by the photodetector after amplification and is returned back to the address port controller from the CMOS receiver IC. The received address request is then forwarded to the cache controller for processing the received address request. Note that in Fig. 2a, in addition to the parallel optical address bits, a snoop bit is also implemented for snoop responses to be discussed later. In Fig. 2b, the interconnection between two successive processors n and $n+1$ on a system board is shown. A more detailed implementation of SYMNET can be found in a separate publication [21].

3 Cache Coherence in SYMNET

Coherence in SYMNET, called COSYM, is modified from the popular MOESI (*Modified, Owned, Exclusive, Shared, Invalid*) [1] protocol. In SYMNET, an address request is *issued* on a cache-miss. This request is *inserted* into the address subnetwork 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 subnetwork and when this request becomes *visible* to all processors. This is in contrast to electrical bus-based SMPs, where the address request becomes visible immediately after the request is broadcast on the bus [1]. Therefore, in electrical SMPs, the cache controller is aware of all previous outstanding requests in the system at the time of inserting the new request. In SYMNET, when the token is received, the cache controller inserts an address request into the address subnetwork. At the time of inserting an address request, the cache controller is unaware of other requests propagating through the address subnetwork affecting the same cache block for which the request is inserted. 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* [1]. 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 the same 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, and this makes the processor which was initially in E state, the owner of the shared 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.

3.2 Race Condition in Reporting Snoop Responses

The snoop response signal also traverses the address subnetwork (see Fig. 2a) using the snoop line and the requesting processor and the memory receive this response [6], [1]. 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. 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. Similarly, when a processor is waiting after inserting a read-miss request, it does not respond to write requests which appear before its own request. The reason for that is the processor writing to the block becomes the owner and the snoop response for the read-miss request will be provided by the owner.

3.3 Verification Methodology for the Cache Coherence Protocol

We should note that it is very hard to verify a coherence protocol with many stable/transient states as it was indicated in [22], [3]. 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 [22], [1] 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.

COSYM: States and Events. The stable states in COSYM have the same functionality as in the MOESI protocol [1]. Table 1 describes the transient states used in COSYM. This representation is similar to the table-based method adopted for verifying multicast protocol [22]. The active/inactive status indicates whether the transient state reacts to incoming address requests. The cache controller reacts to two kinds of requests, issued either from the processor or from the interconnect. The address requests issued by the cache controller 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 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. For the stable state transitions in COSYM, read hit occurs when the block is any stable state except I state. Write hit occurs when the block is in either the M state, or when the block is upgraded from the E state to M state. When a read request is issued by another processor to a block in the E state, the block is upgraded to O state. All write requests from the interconnect will result in the block being downgraded to I state from all the other stable states.

COSYM Transient State Diagram. The transient diagram of the COSYM protocol in case of a cache-miss are shown in Fig. 3. The read-miss and write-miss cases are combined in this figure due to strict page limitations. The states indicated with gray shade imply that the protocol is not reactive to requests yet. The white circles, which are not bold, indicate the states in which the protocol reacts to transactions. The text associated with the arrow from one state to another transient state indicates the event that caused the transition. The transitions due to cache-miss requests issued from the interconnect are shown with a dotted line. Each transient state is indicated in the following manner [22]: < *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 read-miss 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 are 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. 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, or IS-d. Finally, when the data is received, the block makes the

TABLE 1
Cache Controller Transient States

Transient State	*Status*	*Description*
IE-ads	inactive	invalid, issued read request, has not inserted the address request into the network
IE-ads	active	invalid, inserted read request, yet to receive address, snoop and data response
IE-ds	active	invalid, issued read request, received address, yet to receive snoop and data response
IE-d	active	invalid, issued read request, received address and snoop response(low), yet to receive data response
IS-ads	inactive	invalid,issued read request, yet to receive address, data and snoop response
IS-ds	active	invalid, issued read request, received address, yet to receive snoop and data response
IS-d	active	invalid, issued read request, received address and snoop response (high), yet to receive data response
IO-ds	active	invalid, issued either read or write request, the next stable state is known to be Owned, received address, yet to receive snoop and data response
IO-d	active	invalid, issued either read request or write request, the next stable state is known to be Owned, received address and snoop response, yet to receive data response
IM-ad	inactive	invalid, issued or/and inserted write request, yet to receive address and data response
IM-d	active	invalid, issued write request, received address, yet to receive data response
S/O,M-a	active	owned/shared, issued write request, yet to receive the address request
II-d	inactive	invalid, next state is known to be Invalid, yet to receive data response

transition to E, S, O, or I states depending on the previous transient state.

In case of a write-miss, the block maybe in I, S, or O states. If a write-miss occurs, the block transits to IM-ad, indicating pending address and data. The snoop signal is not relevant to the issuing processor as it only needs the data to write to the block. The snoop signal is still provided by the owner, if one exists, to the memory controller. Once the write-miss request is visible, it is serialized in the global order of requests and transits to IM-d which indicates that the data is not received. Any intermediate read-request causes the block to change to IO-d state. When the data is received, depending upon the previous state, the block is loaded in M, O, or I state. If a write-miss occurs and the cache does have a valid block (O/S), it issues an invalidation request and transits to O/S,M-a state. The data is already valid and the processor waits only for the invalidation request to become visible. When the cache block is in O/S,M-a state, if a write request is issued from the interconnect, the block transits to IM-ad, which indicates that both data and address are pending and the controller reacts in a similar fashion as explained.

3.4 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* [22]. 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 Fig. 4. Let us assume that P0 issues a read/write-miss request to block B as shown in Fig. 4a. If it is a read-miss request, this is the first processor issuing the request, the block is loaded in the E state and the data is supplied by the memory. If the request is a write-miss, the memory or the owner of the block supplies the data and P0 loads the block in M state. As both these conditions make P0 the first processor to load the valid block, the notation used here is P0(E/M). In Fig. 4b, 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 Fig. 4c, 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 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-backs: ordinary write-back, transfer write-back Type 1, and transfer write-back Type 2.

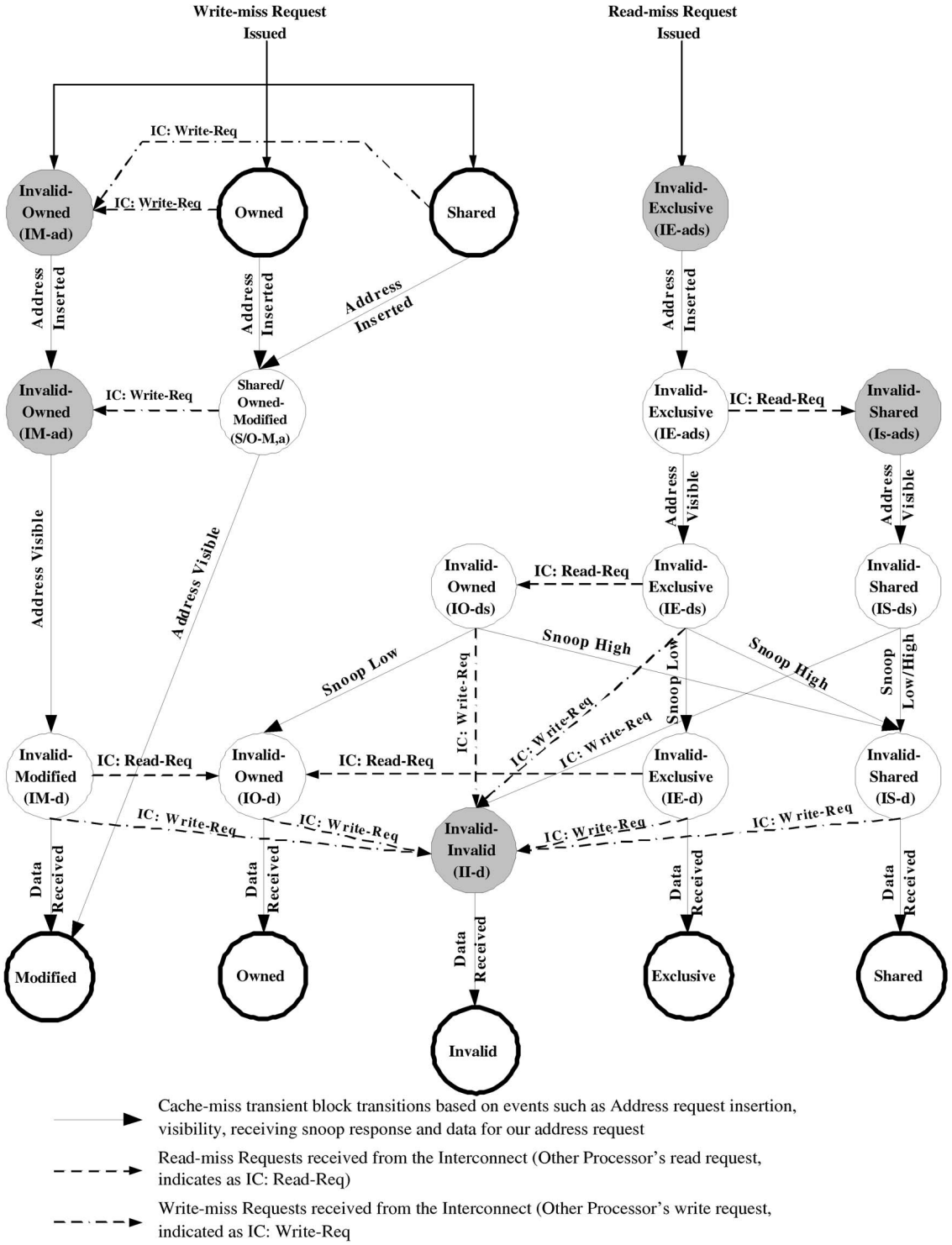

Fig. 3. COSYM cache coherence protocol described using state diagram for all the transient states when a cache miss occurs. The various events that cause the transition are issuing of address requests, inserting address requests, visibility of address requests, and receiving snoop signals and data. The transitions caused by address requests made by other processors are indicated IC: Read/Write Req in the above figure. The arrow from one transient state to another indicates either the event that caused the transition or another processor's request received from the interconnect.

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 Fig. 4d. 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. This is implemented by obtaining the token and inserting a transfer write-back type 2 request. In Fig. 4e, P2 transfers the information of no sharer to P1. This is acknowledged by processor P1.

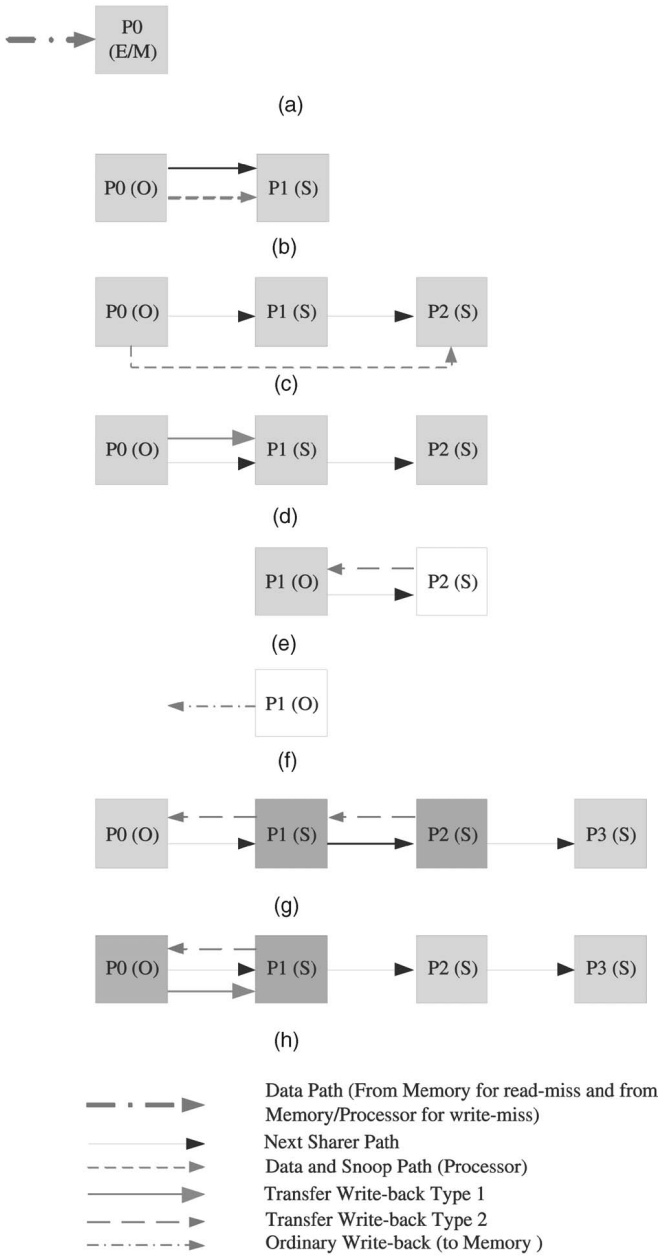

Fig. 4. (a) P0 issuing a read-miss/write-miss request for block B. If it is a read-miss request, the data is supplied by memory and P0 loads the block in E state. If it is a write-miss request, the data is supplied by either memory or some cache and P0 loads the block in M state. As both these conditions, makes P0 the first processor to load the valid block, the notation used here is P0(E/M). In (b), processor P1's read-miss request is satisfied by processor P0. Block B in P0 transits to O, indicated as P0(O), supplies the data and sets P1 as the next sharer. P1 loads the block in S state, indicated as P1(S). P1 issues a read-miss requst to block B. In (c), P2 issue read-miss requests to the same block B. The data/snoop response is provided by P0 and P1 sets the next sharer as P2. P2 loads the block in S state. In (d), Transfer write-back Type 1 (ownership transfer) is issued by P0. In (e), Transfer write-back Type 2 (next sharer transfer) is issued by P2 Type 2. In (f), P1 performs an ordinary write-back. (g) and (h) show race conditions in COSYM protocol. P2 issues a Transfer Write-back Type 2 request and P1 issues a Transfer Write-back Type 2 request. PO's write-back request becomes visible before P1's write-back request.

Ordinary Write-Back. When a modified block is evicted from the cache, then the block is written back to the memory. In Fig. 4f, Processor P1 writes the block back to memory since there are no sharers for the block. This does not cause another transaction to be inserted into the address subnetwork. 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 since it is the owner for the block.

4 Performance Evaluation

4.1 Simulation Methodology and Architectural Assumptions

We have chosen Limes (Linux Memory Simulator) [23], 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 access are contention-free to maximize the effects of the limited address bus bandwidth. We compare SYMNET with the traditional electrical bus-based SMPs implementing the MOESI protocol with a subset of Splash-2 suite benchmarks [13]. The electrical SMP considered for comparison is similar in design to the Gigaplane and StarFire models [6]. We do not compare SYMNET which is a pure-snooping-based SMP with the Fireplane [2] model for two reasons: 1) the Fireplane architecture follows a hybrid coherence model containing both the snooping and the directory-based protocols and 2) the coherence protocol of the Fireplane has not been published at this time.

Benchmarks. In this study, we use eight benchmarks, which cover a spectrum of memory sharing and access patterns from the SPLASH-2 suite [13], namely, FFT with input data set $64K$ points; LU with 256×256, 16×16 block; Ocean with 130×130; Radix with $1M$ integers, $1,024$ radix; Water-nsquared with 512 molecules; FMM with $16K$ particles; Barnes-hut with $1K$ particles, 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. Unfortunately, due to the complexities of full-system simulation, we were unable to simulate for more than 32 processors for some applications. However, applications such as FFT, LU, Radix, Water, and Ocean showed similar trends when simulated for 64 processors.

Processor/Cache Parameters. 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 four cycles. The processor and the cache parameters are kept constant while simulating both electrical and optical networks. All first-level cache read/write hits are assumed to take one processor clock cycle (pcc). Throughout this evaluation, we have considered processor clock cycles (pcc) as the base time unit for all measurements.

Electrical 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 [2], [6], [7], the processor clock is always a fraction of the system clock rate. For example, in the StarFire model, UltraSparc2 is clocked at $250Mhz$, whereas 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 [6] architecture, it takes two 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 two address requests per cycle using four address buses. This case is simulated where each processor receives two 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 two 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 $RPC = 1$ and $RPC = 2$ cases.

Optical Simulation Parameters. In SYMNET, the optical token is implemented such that the optical signal, generated by a laser source, is split as shown in Fig. 2a. 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. In order to calculate the time lag between two successive processors inserting address requests into the interconnect so that the address requests do not collide, we evaluate the value of D, i.e., the delay induced by the delay element. This delay should account for the distance of propagation of the signal from the delay element to the next processor $n + 1$, detection of the signal by the photodetector, and the rise time of address pulses driven by VCSEL arrays at processor n. The delay D, can be determined as:

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

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. If the delay element at the processor n is close to the the detector at processor $n + 1$, we can approximate this distance to be $2 - 5cms$. The velocity of light in fiber/waveguide is $v_c = 2 \times 10^8$. The O-E conversion depends on the data rate of the photodetector. Current high data rate pin photodetectors are available at $3 - 5Gbps$, this value can be estimated to be $O_e = 200psec$. VCSELs are commercially available in the region of $3 - 5Gbps$ and with $m = b$, D can be estimated to be $0.8nsec$. The optical token should be seen by the next processor with a delay greater than $0.8nsec$ to prevent collision of address requests. Therefore, the other part of the optical signal at the delay element should be delayed by more than $0.8nsec$. Considering an expected delay skew in the region of $0.1nsec$, we consider that the delay D should 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 an address transaction to be visible is equivalent to the number of stages in the address subnetwork. This is assumed to be twice the logarithm of the number of processors connected in the address subnetwork. The snoop response also takes similar number of cycles after the address is snooped. The data network considered for SYMNET is the SOCN [17] network. SOCN is an optical crossbar constructed using VCSEL/PD arrays and diffraction grating. The data subnetwork for SYMNET is also assumed to be conflict-free. The simulated cycles to transfer the data in SYMNET varies and this depends on current optical devices. While conflict in wavelength allocation can cause additional delays [17], for simplicity, we assume that the number of wavelengths available is equal to the number of processors simulated. The delay in data transfer for the optical network depends on the data rate of current multiwavelength VCSEL arrays. At 5 Gbps VCSEL data rate, to transmit 32 byte block, it takes around $52nsec$ ($= (32 \times 8)/(5 \times 10^9)$) and this corresponds to $52pcc$.

4.2 Simulation Results

We determined the execution time, the average delay for cache miss transaction, and the average delay for write-back transaction for SYMNET and the electrical bus-based SMP varying from 2 to 32 processors. We also measured the percentages of different types of write-backs and the effects of these write-backs on the address bandwidth in SYMNET.

Normalized Execution Time. Fig. 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. The remaining two cases for a given number of processors, are normalized to this maximum value. For FFT, COSYM shows 25 percent improvement over MOESI protocol for $RPC = 1$ and 8 percent improvement for $RPC = 2$ running for 32 processors. For the LU application, the improvement in execution time is around 30-35 percent for both the cases. The best improvement is visible for Ocean application, where the improvement is more than 62 percent for $RPC = 1$ and over 52 percent for $RPC = 2$ running for 32 processors. The improvement in performance for Radix is 38 percent 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 percent for $RPC = 1$ and Water shows an improvement of 16-19 percent for both cases. COSYM protocol improves execution time by 20 percent and 15 percent for Barnes-Hut and FMM applications for 32 processors, respectively.

Normalized Average Latency. Fig. 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 cache miss request 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 this value. The average delay was much higher for the electrical

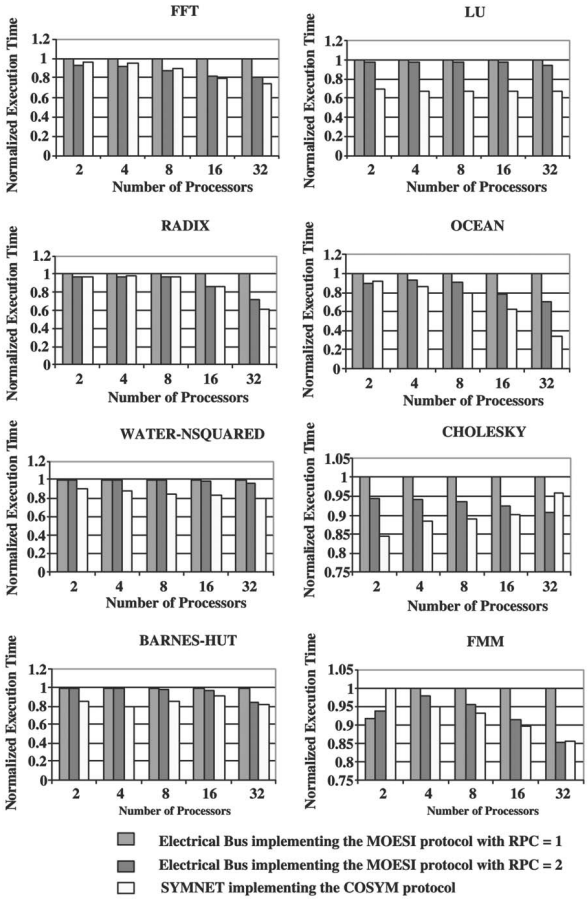

Fig. 5. Normalized Execution time for processors varying from 2 to 32 for all Splash-2 benchmarks. The execution time for the electrical bus implementing the MOESI protocol with a single address request per cycle ($RPC = 1$), two address requests per cycle ($RPC = 2$), and the SYMNET address subnetwork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

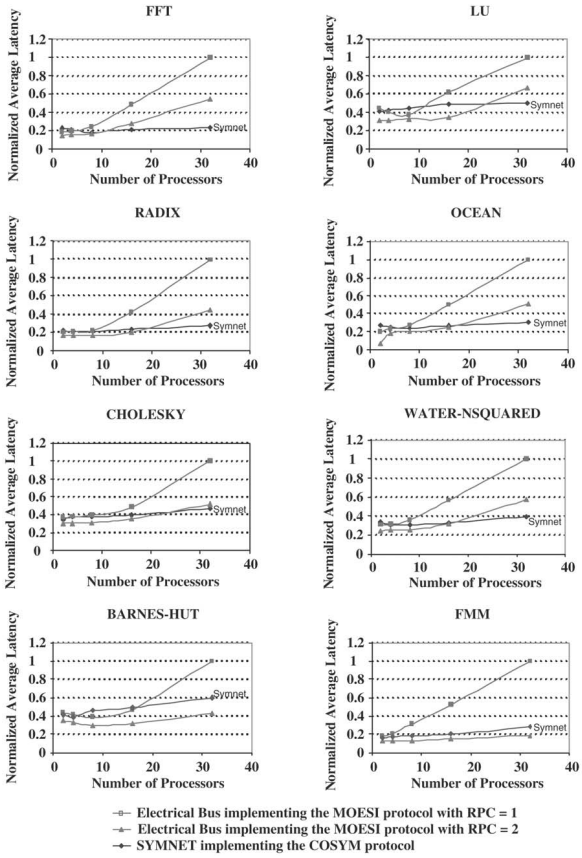

Fig. 6. Normalized Average latency of a transaction for processors varying from 2 to 32 for all Splash-2 benchmarks. The average latency for the electrical bus implementing the MOESI protocol with a single address request per cycle ($RPC = 1$), two address requests per cycle ($RPC = 2$), and the SYMNET address subnetwork 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.

case for all applications which increased linearly with the number of processors. The delay for the single address request per cycle case was higher than the two address requests per cycle case as expected. Most applications showed lower delay for MOESI protocol for smaller configurations. As the number of processors increased, COSYM outperformed both 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 COSYM model with a faster address interconnect and a data crossbar provides much better performance for all the cases. The reduction in latency for FFT using the COSYM protocol is as high as 76 percent for $RPC = 1$ and 57 percent for $RPC = 2$. For LU, the reduction in latency for COSYM protocol ranged from 51 percent for $RPC = 1$ to 25 percent for $RPC = 2$. Barnes-Hut and FMM showed slightly better performance for $RPC = 2$ condition than COSYM protocol. COSYM still outperformed Barnes-Hut and FMM for $RPC = 1$ condition by as much as 40 percent and 85 percent, respectively.

Normalized Average Write-Back Time. Fig. 7a shows the average time taken to complete write-back transactions for COSYM and MOESI protocols. The electrical cases using the MOESI protocol for both the conditions showed lower write-back time than COSYM protocol. This is attributed to the number of cycles required to perform the write-back, since electrically, it takes 24 clock cycles as compared to 52 cycles required for the COSYM protocol. The assumption that data transfer in SYMNET takes 52 cycles is based on current optical device technology and is expected to improve in the future.

Percentages of Transfer and Ordinary Write-Backs. Fig. 7b shows the transfer write-back type 1, transfer write-back type 2, and ordinary write-backs as a percentage of the total number of write-backs for COSYM protocol for different applications. As the number of processors increases, percentage of transfer write-backs also increases. All applications other than LU and radix have transfer write-backs lesser than 20 percent of all write-backs. For 32 processors, for Radix application, the transfer write-back jumps to almost 40 percent of the total number of write-backs. The Ocean application shows a low number of transfer write-backs for all the cases. For Ocean, this is approximately 8-10 percent of all write-backs for 32 processors. As more processors are connected in the interconnect, the ownership of the dirty block is more often transferred between processors than writing it back to memory every now and then.

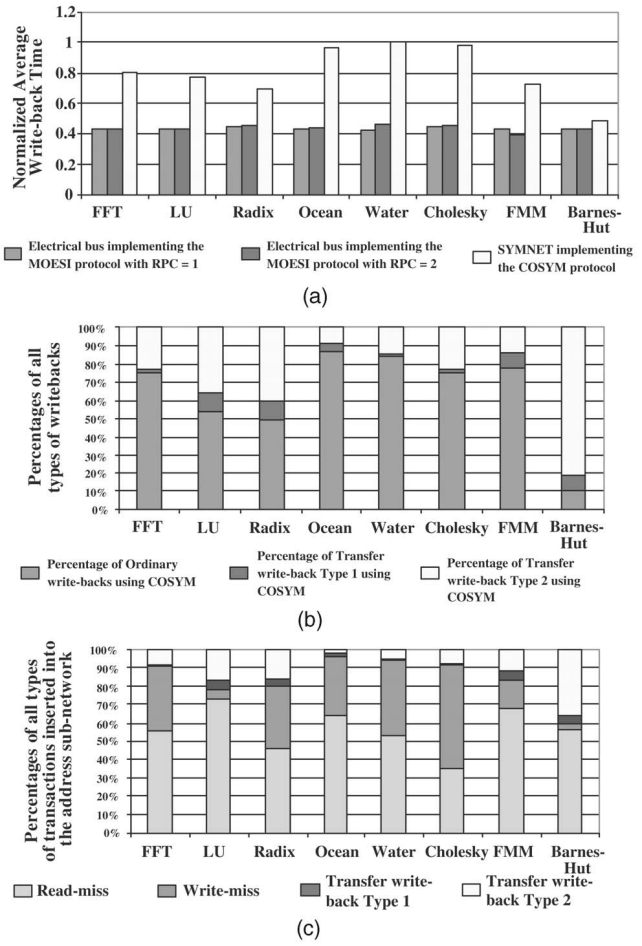

Fig. 7. (a) shows the normalized average write-back time required for 32 for the Splash-2 benchmarks. Normalized average write-back time is calculated by considering the maximum value of average write-back time for a given application and dividing all the remaining cases with the maximum value. (b) shows the percentages of different types of write-backs occurring in COSYM protocol for 32 processors. (c) shows the percentages of all types of transactions inserted into the address subnetwork for 32 processors. Note that the ordinary write-backs use the data subnetwork and are not plotted here. The reissued transactions are insignificantly less as compared to other transactions and, therefore, are not visible in the figure.

Effects of Transfer Write-Backs on Address Bandwidth. Fig. 7c shows the percentages of all types of transactions inserted into the address subnetwork. It should be noted that ordinary write-back transactions are not inserted into the address subnetwork as they are written back using the data subnetwork. The dominant requests propagating through the address subnetwork are read and write misses for all applications. All applications, except Barnes-hut, have transfer write-backs less than 10-15 percent of all transactions inserted into the address subnetwork. The reissued write-back transactions are insignificantly less as compared to other transactions and therefore, are not visible in Fig. 7c.

There are four distinct differences in implementing the write-backs in the COSYM protocol, as compared to electrical MOESI protocol. First, the E block is upgraded to O state when a read-miss is issued by another processor. The drawback of this is, if the processor needs to write-back this block in O state, and if there are no sharers, then this clean block is unnecessarily written back to memory. Second, COSYM stores the next sharer information for every shared block and this sharing information is transferred by issuing transfer write-back Type 1 and Type 2 transactions. Third, an overhead in the COSYM protocol is the transfer write-back Type 2 transaction, where even a shared block evicted from the cache needs to inform the previous processor of the change in the next sharer. Fourth, a write-back transaction may be reissued if a transfer write-back Type 1 or 2 is not completed due to race conditions as discussed before.

The first difference can be resolved by adding another stable state to COSYM, such that the block transits from E to this new state, rather than to the O state, and becomes the owner of the block. If a write-back is issued to this new state and if there are no sharers, then the block can be evicted without writing it back to memory. The second and the third differences are related to the snoop response requirements. The concept of maintaining the next sharer has been previously introduced in multicast protocol [22]. Transfer write-back Type 2 is an overhead and may increase with the number of processors. 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. For the fourth difference, our simulation studies for various applications and varying processors have shown that reissuing of transactions is negligible compared to the total number of transfer write-backs that take place. For all applications the number of reissued transactions is less than 0.01 percent of the total number of requests inserted into the address subnetwork. Therefore, the race conditions, which cause the write-back request to be reinserted, occur for a very small percentage of write-backs.

5 Conclusion

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 and compared it with the electrical bus-based MOESI protocol using Splash-2 benchmarks from 2 to 32 processors. Our simulation studies have shown a 5 to 66 percent 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 to 78 percent better than the MOESI protocol. Our objective to implement the snoop response signals optically resulted in handling write-backs differently with ownership transfer. The improvement in latency for an address transaction offsets the write-back latency, resulting in better performance for the COSYM protocol as our simulation studies have shown. 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 significant performance advantages over traditional electronics. Greater improvements in terms of bandwidth, latency, and scalability can be expected with further improvement in optical device technology.

ACKNOWLEDGMENTS

This research is sponsored by the US National Science Foundation grant no. CCR-0000518. The authors would like to thank Professor D. Litaize for pointing out the limitations of SMPs in a quantitative manner. They would also like to thank Professor J. Collet for many fruitful discussions and suggestions.

REFERENCES

[1] D.E. Culler, J.P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach.* San Francisco, Calif.: Morgan Kaufmann, 1999.

[2] A. Charlesworth, "The Sun Fireplane SMP Interconnect in the Sunfire 3800-6800," *Hot Interconnects 9,* pp. 37-42, Aug. 2001.

[3] F. Pong, M. Dubois, and K. Lee, "Design and Performance of SMPS with Asynchronous Caches," Technical Report HPL-1999-149, Hewlett Packard, HP Laboratories Palo Alto, Nov. 1999.

[4] D.A.B. Miller, "Rationale and Challenges for Optical Interconnects to Electronic Chips," *Proc. IEEE,* vol. 88, pp. 728-749, June 2000.

[5] J.H. Collet, W. Hlayhel, and D. Litaize, "Parallel Optical Interconnects May Reduce the Communication Bottleneck in Symmetric Multiprocessors," *Applied Optics,* vol. 40, pp. 3371-3378, 2001.

[6] A. Charlesworth, "Starfire: Extending the SMP Envelope," *IEEE Micro,* vol. 18, no. 1, pp. 39-49, Jan.-Feb. 1998.

[7] M. Galles and E. Williams, "Performance Optimizations, Implementation and Verification of the SGI Challenge Multiprocessor," *Proc. 27th Ann. Hawaii Int'l Conf. Systems Sciences,* pp. 134-143, 1996.

[8] M.M. Martin, D.J. Sorin, A. Ailamaki, A.R. Alameldeen, R.M. Dickson, C.J. Mauer, M. Plakal, M.D. Hill, and D.A. Wood, "Timestamp Snooping: An Approach for Extending SMPS," *Proc. Ninth Int'l Conf. Architectural Support for Programming Languages and Operating Systems,* pp. 25-36, Nov. 2000.

[9] J.H. Collet, D. Litaize, J.V. Campenhut, C. Jesshope, M. Desmulliez, H. Thienpont, J. Goodman, and A. Louri, "Architectural Approaches to the Role of Optics in Mono and Multiprocessor Machines," *Applied Optics,* special issue on optics in computing, vol. 39, pp. 671-682, 2000.

[10] Y.S. Liu, R.J. Wojnarowski, W.A. Hennessy, P.A. Piacente, J. Rowlette, J. Stack, M. Kader-Kallen, Y. Liu, A. Peczalski, A. Nahata, and J. Yardley, "Plastic VCSEL Array Packaging and High Density Polymer Waveguides for Board and Backplane Optical Interconnect," *Proc. Electronic Components and Technology Conf.,* pp. 999-1005, 1998.

[11] F. Mederer, F. Eberhard, I. Ecker, R. Jager, J. Joos, C. Jung, M. Kicherer, R. King, P. Schnitzer, H. Unold, D. Wiedenmann, and K.J. Ebeling, "High Performance Selectively Oxidized VCSELs and Arrays for Parallel High-Speed Optical Interconnects," *Proc. 50th Electronic Components and Technology Conf.,* pp. 1242-1251, May 2000.

[12] R. Pu, E.M. Hayes, C.W. Wilmsen, K.D. Ohoquette, H.Q. Hou, and K.M. Geib, "Comparison of Techniques for Bonding VCSELs Directly to ICS," *J. Optical Soc. Am.,* vol. 1, pp. 324-329, 1999.

[13] C.S. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The Splash-2 Programs: Characterization and Methodological Considerations," *Proc. 22nd Ann. Int'l Symp. Computer Architecture,* pp. 24-37, June 1995.

[14] D.M. Chiarulli, S.P. Levitan, R.G. Melhem, M. Bidnurkar, R. Ditmore, G. Gravenstreter, Z. Guo, C. Qiao, M.F. Sakr, and J.P. Teza, "Optoelectronic Buses for High-Performance Computing," *Proc. IEEE,* pp. 1701-1710, 1994.

[15] P. Lukowicz, "The Photobus Smart Pixel Interconnection System for Symmetric Multiprocessing Using Workstation Clusters," *Proc. Sixth Int'l Conf. Parallel Interconnects,* pp. 106-113, 1999.

[16] J.-H. Ha and T.M. Pinkston, "A New Token-Based Channel Access Protocol for Wavelength Division Multiplexed Multiprocessor Interconnects," *J. Parallel and Distributed Computing,* vol. 60, no. 2, pp. 169-188, Feb. 2000.

[17] B. Webb and A. Louri, "A Class of Highly Scalable Optical Crossbar-Connected Interconnection Networks (SOCNS) for Parallel Computing Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 11, no. 1, pp. 444-458, May 2000.

[18] P. Dowd, J. Perreault, J. Chu, D.C. Hoffmeister, R. Minnich, D. Burns, F. Hady, Y.J. Chen, and M. Dagenais, "Lighnting Network and Systems Architecture," *J. Lightwave Technology,* vol. 14, pp. 1371-1387, 1996.

[19] A. Lindstrom, "Parallel Links Transform Networking Equipment," *FiberSystems Int'l,* pp. 29-32, Feb. 2002.

[20] A.V. Krishnamoorthy and K.W. Goossen, "Optoelectronic-VLSI: Photonic Integrated with VLSI Circuits," *IEEE J. Selected Topics in Quantum Electronics,* vol. 4, pp. 899-912, 1998.

[21] A. Louri and A.K. Kodi, "Parallel Optical Interconnection Network for Address Transactions in Large-Scale Cache Coherent Symmetric Multiprocessors," *IEEE J. Selected Topics in Quantum Electronics,* special issue on optical interconnects, vol. 9, pp. 667-676, 2003.

[22] D.J. Sorin, M. Plakal, A.E. Condon, M.D. Hill, M.M. Martin, and D.A. Wood, "Specifying and Verifying a Broadcast and a Multicast Snooping Cache Coherence Protocol," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, 2002.

[23] I. Ikodinovic, A. Milenkovic, V. Milutinovic, and D. Magdic, "Limes: A Multiprocessor Simulation Environment for PC Platforms," *Proc. Int'l Conf. Parallel Processing and Applied Math.,* Sept. 1999.

Ahmed Louri received the MS degree in computer engineering in 1984 and the PhD degree in computer engineering in 1988, both from the University of Southern California (USC), Los Angeles. He is currently a full professor of electrical and computer engineering at the University of Arizona, Tucson, and the chairman of the Computer Engineering Program. He is also the director of the Optical Networking and Parallel Processing Laboratory. His research interests include computer architecture, parallel processing, optical computing systems, and optical interconnection networks. He has published numerous journal and conference articles on the above topics. He was the recipient of the Best Article of 1991 Award from IEEE Micro, the recipient of the US National Science Foundation Research Initial Award, the recipient of the Advanced Telecommunications Organization of Japan Fellowship, Ministry of Post and Telecommunications, Japan, the recipient of the Centre Nationale de Recherche Scientifique (CNRS), Fellowship, France, and the recipient of the Japanese Society for the Promotion of Science Fellowship. Dr. Louri has served as a member of the technical program committee of several conferences including OSA Topical Meetings on Optics in Computing and the OSA/IEEE Conference on Massively Parallel Processors using Optical Interconnects, among others. He has also served as the general chair of the Workshop on Optics in High-Performance Computing, at Euro-Par '96 Lyon, France. His Web address is www.ece.arizona.edu/department/ocppl. Dr. Louri is a senior member of the IEEE and a member of OSA.

Avinash Karanth Kodi received the BEngg degree in electronics and communication from Manipal Institute of Technology, Mangalore University, Mangalore, India, in 1999 and the MS degree in computer engineering in 2003 from the University of Arizona, Tucson, where he is currently pursuing the PhD degree. His research interests include design of high-speed optical interconnects for shared-memory multiprocessors, parallel processing, and cache coherence protocols. He is a student member of the IEEE.