A Composite and Scalable Cache Coherence Protocol for Large Scale CMPs

Yi Xu[†], Yu Du[‡], Youtao Zhang[‡], Jun Yang[†]
[†] Department of Electrical and Computer Engineering
[‡] Department of Computer Science
University of Pittsburgh, Pittsburgh, Pennsylvania, USA
[†]{yix13, juy9}@pitt.edu, [‡]{fisherdu, zhangyt}@cs.pitt.edu

ABSTRACT

The number of on-chip cores of modern chip multiprocessors (CMPs) is growing fast with technology scaling. However, it remains a big challenge to efficiently support cache coherence for large scale CMPs. The conventional snoopy and directory coherence protocols cannot be smoothly scaled to many-core or thousand-core processors. Snoopy protocols introduce large power overhead due to enormous amount of cache tag probing triggered by broadcast. Directory protocols introduce performance penalty due to indirection, and large storage overhead due to storing directories.

This paper addresses the efficiency problem when supporting cache coherency for large-scale CMPs. By leveraging emerging optical on-chip interconnect (OP-I) technology to provide high bandwidth density, low propagation delay and natural support for multicast/broadcast in a hierarchical network organization, we propose a composite cache coherence (C^3) protocol that benefits from direct cache-to-cache accesses as in snoopy protocol and small amount of cache probing as in directory protocol. Targeting at quickly completing coherence transactions, C^3 organizes accesses in a three-tier hierarchy by combining a mix of designs including local broadcast prediction, filtering, and a coarse-grained directory. Compared to directory-based protocol [18], our evaluations on a thousand-core CMP show that C^3 improves performance by 21%, reduces network latency of coherence messages by 41% and saves network energy consumption by 5.5% on average for PARSEC applications.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Processor Architectures—*Multiple Data Stream Architectures (Multiprocessors)*; B.3.2 [**Memory Structures**]: Design Styles—*Cache memories, shared memories*; B.4.3 [**Hardware**]: Input/Output and Data Communications—*Interconnections (subsystems)*

General Terms

Performance

ICS'11, May 31–June 4, 2011, Tucson, Arizona, USA.

Keywords

Cache Coherence Protocol, Thousand-Core, CMP, Optical Network, Nanophotonics

1. INTRODUCTION

Chip multiprocessors (CMP) have emerged as a promising microarchitecture for keeping up performance with integration density [20, 41]. Today, the number of on-chip cores has reached low hundreds, e.g. Intel's 80-core Terascale chip [48] and nVidia's 128-core Quadro GPU [40]. In near future, the on-chip core count will likely to reach upper hundreds or even a thousand [11].

Designing hardware cache coherence protocols for future many-core CMPs is challenging. There are mainly two classes of protocols to enforce cache coherence: snooping-based and directory-based protocols [3]. There have been many proposals for small- and mid- scale CMPs [4, 5, 17, 23, 24, 34, 35, 46]. However, supporting coherence for large scale CMPs remains difficult. Snoopy protocols introduce large power overhead due to enormous amount of cache tag probing triggered by broadcast. Directory protocols introduce large storage and performance overhead due to directory store and lookup. In addition, the efficiency of coherence protocols depends on the network topology. Traditional topologies, such as Mesh, face challenge in performance scalability. For example, core-to-core communications require up to 64 hops for a 1024-core mesh network, making it prohibitively expensive to support indirection in directory protocols. Mesh-based snoopy protocols [5, 4, 23], on the other hand, require additional hardware to support broadcast and message ordering, making it difficult to scale with increased core count and cache sizes.

Recent efforts have embraced the emerging nanophotonic technologies for networks-on-chip. Optical interconnects (OP-I) naturally supports broadcast, provides high bandwidth density, low propagation delay, and low power consumption for remote communications [8, 36, 37]. However, few optical designs address the intrinsic scalability issues of coherence protocols in large-scale CMPs. For example, optical crossbars [29, 42, 49] did not reduce the cache tag probing overhead in snoopy protocols. Hence, new designs are necessary for large-scale cache coherence protocols to be efficient and scalable.

In this paper, we address the cache coherence problem for thousand-core CMPs leveraging benefits of OP-I. We first alleviate the scalability pressure by shrinking the network size with cache clustering and network concentration. We then utilize optical crossbars to provide inter-cluster communi-

cations for high bandwidth and low network diameter. We greatly reduce snooping messages through local broadcast with prediction, and message filtering. Finally, we use extremely small directories to avoid unnecessary global broadcast. Comparing to various directory and snoopy protocols in a thousand-core architecture, our composite protocol achieves 21% performance improvement and 5.5% energy savings with modest storage. The last-level cache probing is only 5.8% of a snoopy protocol. In summary, we make these contributions:

- We introduce a composite cache coherence (C^3) protocol that strives to achieve both direct cache-to-cache accesses as in snoopy protocol, and small amount of cache probing as in directory protocol.
- We integrate recent research and technology advances including OP-I, 3D die-stacking and network concentration to design a viable NoC architecture for thousand-core CMPs.
- We design a new mechanism to order coherence messages in global network to remove racing situations and ensure correctness.

The remainder of this paper is organized as follows. Section 2 presents the background of nanophotonics. Section 3 discusses the details of the proposed C^3 protocol as well as architecture design of thousand-core CMP. The implementation details are described in Section 4. The experimental results are analyzed in Section 5. Section 6 summarizes the related work. Finally, Section 7 concludes this paper.

2. NANOPHOTONICS BACKGROUND

Recent ITRS report [21] identified limitations in using metal wires for global links: (i) the wire performance does not scale well; (ii) long RC wires require large number of repeaters that consume significant portion of total power; and (iii) the slow increase of pin count restricts the bandwidth between core and memory. In contrast, nanophotonic links can provide high bandwidth density, low propagation delay, communication-target-independent power consumption, and natural support for multicast/broadcast. Recent advances in photonic devices and integration technology have made it a promising candidate for future global interconnects.

A typical optical network includes off-chip laser source that provides on-chip light, waveguides that route optical signal, ring modulators that convert electrical signals to optical ones, and ring filters to detect lights and translate it into electrical signals. Fig. 1 illustrates a typical dense wavelength division multiplexing (DWDM) nanophotonic link. Light of multiple wavelengths is generated by the off-chip laser source and then coupled into the on-chip waveguide. Though on-chip light source exists [51], it consumes significant precious on-chip power and area [37]. Waveguide, which uses high refraction index material as the core part and low index material on the outside, confines and guides the light around the chip with losses on the order of 0.1dB/mm [30]. Since light of different wavelengths can be transmitted and modulated in the single waveguide, DWDM technology enables multiple data channels per waveguide, providing high network bandwidth density. At the sender side, electrical signals are imprinted to laser light by wavelength-selective silicon modulators that absorb and pass the light for signal '0' and '1' respectively. The modulators are built based on ring resonators of less than $50\mu m^2$/ring [37] performing at 20Gbps. At the receiver side, the optical filter extracts the light of specific wavelength from the waveguide and transfers it to the photo detector which can be built with CMOS-compatible germanium to convert optical signals to electrical ones which are then passed to amplifiers. In global broadcast, optical signal is able to reach the furthest receiver in the network because of the low-loss property of those optical filters in the intermediate nodes. The power and energy loss of different optical modules are listed in Section 4.2.

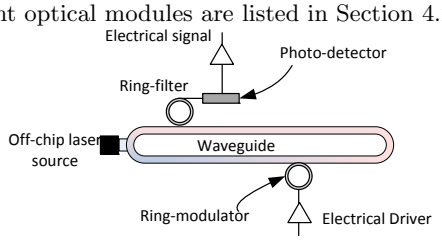

Figure 1: Photonic components.

3. COMPOSITE CACHE COHERENCE (C^3) PROTOCOL DESIGN

In this section, we present the composite cache coherence protocol to address the efficiency problem in large-scale CMPs. We first discuss the chip architecture and cache organization that our protocol is based upon, and then elaborate the protocol details.

3.1 Thousand-Core Chip Architecture

Fig. 2 illustrates an overview of our 1000-core chip architecture. It adopts emerging 3D die-stacking technology [10, 27] to integrate CMOS logic and memories into different layers. All cores are in one layer and caches are in other layers. In our design, there are three levels of cache in the hierarchy: L1 and L2 are private; they are relatively small and fast. L3 (LLC) is shared; it is relatively large and slow. L1 is placed in the core layer for fast access. L2 and L3 are stacked atop.

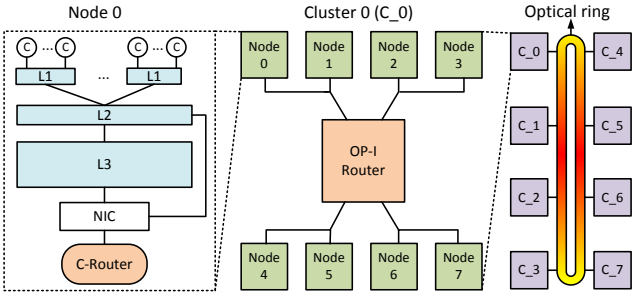

Figure 2: A thousand-core cache and NoC architecture.

3.1.1 Cache Clustering

In typical small-scale CMPs, the private cache tile of each core is directly connected to an NoC node. However, this would be inappropriate for thousand-core processors because the NoC size would be too large and average latency would too high. Also, each node's traffic injection rate is not very high because they are from a single core's private cache misses, indicating that such NoC is not very efficient. Therefore, a better way is to employ the concentration technique to share the network channel among core-cache tiles [6, 28]. We choose to cluster several cores and their L1 and L2 cache slices. Sharing caches results in less global coherence traffic, better utilization of the limited cache capacity and network resources. However, too much sharing may create

contention in cache, especially in L1 which is performance critical. Hence, we let four cores share one L1, and four L1s share one L2, similar to the design of the Rock processor [15]. Each L2 is connected to one concentrated router (C-Router) via network interface module (NIC), resulting in only 64 nodes in the NoC, as shown in left part of Fig. 2.

3.1.2 Optical Interconnection Topology

To support high traffic volume generated from up to thousand on-chip cores, we employ the single-write-multiple-read (SWMR) optical buses [29, 37, 42] because it provides contention free connections between senders and receivers. Each node in the network has a dedicated channel to send requests and all remaining nodes listen to the channel to receive requests. We choose this topology due to its contention-free communication, high performance and low design complexity. However, the power consumption of this design increases quadratically with the network size. This is because all microrings need to consume the "listening" power and there are N^2 microrings (N is the network size) in the crossbar. We have already reduced the network size from 1000 to 64 through cache clustering. But a 64×64 optical crossbar is still quite large. Hence, a hierarchical network architecture [29, 37] is necessary to keep the crossbar small. Adopting such a hierarchy divides the network into C clusters. Each cluster has $64/C$ nodes that share one OP-I router. The SWMR crossbar then connects all OP-I routers to form a network of C nodes.

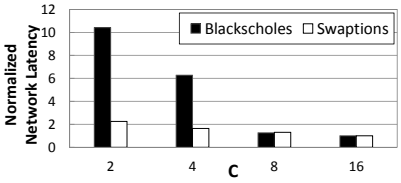

Figure 3: Network latency with different Cs.

Determining an appropriate C represents the design tradeoff between bandwidth and power in 1) the aggregated traffic load per cluster. If cluster size is large, then the bandwidth requirement within a cluster may be high for the OP-I router, resulting in contention delay and performance degradation; 2) the power consumed by microrings. If the cluster size is small, then C is large, which results in quadratic increase in power consumption as discussed earlier; 3) the power consumed by laser source. More ring resonators on a waveguide will cause more energy loss during light propagation, which leads to higher laser power at the source. Fig. 3 compares the network latency of two applications with different Cs. The 8- and 16-cluster networks obtain better performance over others. We choose C=8 in this paper. Electrical network is used to connect 8 L2 banks within each OP-I router (Fig. 2). These links are bidirectional. A coherence message generated by a node (L2 bank) is first sent to either local L3 if it is the home node, or the NIC that packs it as a network message and then sends to the corresponding C-router which routes the message through local electrical network to the OP-I router and then onto the global OP-I network (Fig. 2).

Data messages are much larger than coherence messages. The latter often requires broadcast or multicast while the former is always unicast. Since the laser power for broadcast or multicast is higher than unicast [8, 29], it is inefficient to use the same network for both coherence and data messages. In our scheme, we use a separate network that employs a reservation-assisted SWMR crossbar for data messages [42], which saves filter power and laser source power. Lastly, we use multiple network layers to supply necessary bandwidth required by both networks. This is more effective than increasing the network bandwidth directly [16]. DWDM technology is adopted to implement multiple network layers through partitioning the wavelengths according to address space. The details of the network layers used are discussed in section 4.2.

3.2 The C³ Protocol

The goal of the C^3 protocol is to provide as much as possible direct cache-to-cache communication between requester and destination as in a snoopy protocol, while minimizing the amount of cache tag probing, and storage space as required in a directory protocol.

3.2.1 Local Broadcast for Reads

Let us first perform a simple traffic analysis on a set of PARSEC workloads [9] to motivate our design.

The left bar in Fig. 4 shows the probability of a read request finding a sharer ("shared data"), or the owner ("dirty data") in its local cluster, i.e., if the requested data present in a local L2 cache bank. On average, there is a 50% chance that a read can find the desired data locally, without having to use the global optical crossbar. For some benchmarks such as `blackscholes` and `swaptions`, the probability can be as high as 85%. This simple observation suggests that for read requests, it is beneficial to perform a local broadcast. When it is likely to obtain the target data directly from a local L2 cache, this design choice is not only fast but also energy efficient.

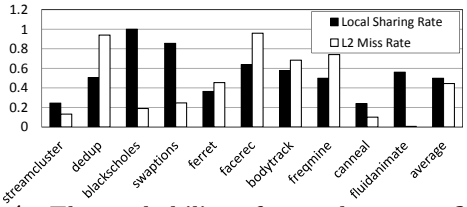

Figure 4: The probability of a read request finding a sharer/owner in the local cluster upon L2 hit and L2 miss rate of read requests.

However, blindly broadcasting read requests is not always beneficial because 1) some benchmarks such as `canneal` have low local sharing rate; 2) some benchmarks do not even have many reads that can find owner or sharers in global L2 due to L2 miss, as is also plotted in Fig. 4. Some of workloads cannot fit in private L2 cache and miss rate is over 90%. Hence, performing a local broadcast for every read in those benchmarks indeed introduces extra cycles and energy. To prevent unnecessary broadcast in those cases, we implement a simple 1-bit predictor for every core in each OP-I router to determine if a broadcast should be performed. If a local owner or sharer is found in the previous broadcast, then the bit is set, and the next read request will also be broadcast. Otherwise, the bit is reset, preventing the next local broadcast. If the later activity, e.g. from directory, indicates there is a local owner or sharer, the bit is set again. In Section 5, we will show that the simple 1-bit predictor can remove above 60% blind local broadcast.

Although a large portion of read requests benefit from local broadcast, write requests cannot because local forwarding is only allowed when the the owner is in the local cluster

and dirty. Otherwise the write requests have to be broadcast to other clusters to invalidate all the sharers. Fig. 5 shows the possibility of the write request finishing transactions locally.The benchmarks not shown here hardly have any owner status during our experimental phase. As we can see the success rate is much lower than that of a read request in Fig. 4.

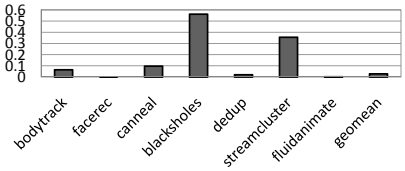

Figure 5: The probability of a write request finding a owner in the local cluster.

3.2.2 Filtering Requests for Global Broadcasts

In this section, we discuss how to handle requests that either skip local broadcast (writes) or fail to find a local sharer/owner (reads). We take advantage of direct cache-to-cache transfer using snooping, which can quickly locate the owner of the requested data. But broadcasting such requests to all nodes still incurs enormous probing energy costs and long queuing delay at L2 cache side. In our design, we use carefully designed filters to remove unnecessary broadcast overhead in snooping. That is, requests that hit in filters are turned into direct message or multicast messages. Requests that missed filters are sent directly to the home node. Most of them are also global L2 misses and should be sent to L3 anyway.

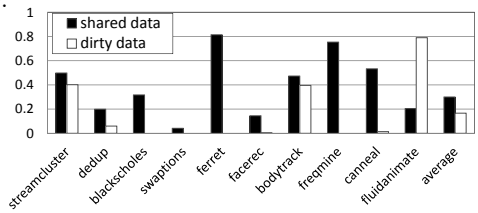

Figure 6: Categorizing read requests. There are more shared data than dirty data.

We put filters in OP-I routers to capture broadcast messages upon first sight. To determine what information is kept in filters, we analyzed read requests that are satisfied by read sharers and dirty owner in Fig. 6. We can see that for most benchmarks such as `ferret` and `freqmine`, shared-data reads predominate the total reads. For others such as `fluidanimate`, `streamcluster` and `bodytrack`, dirty-data reads occupy significant portion of the total reads. For the rest, i.e., `swaptions` and `facerec`, most read requests are simply L2 misses. Thus, if we store sharers in the filter, we can filter out more global broadcast messages. In addition, we can further reduce invalidation messages inside the cluster by utilizing the information of the sharer list in the filter.

Based on above observations, each entry of the filter stores an 8-bit *cluster* sharer vector and an 8-bit *local* sharer ID, as shown in Fig. 7. Each bit of the cluster sharer vector indicates if there is at least one sharer in the cluster. The local sharer records sharers within the local cluster. We use an example to illustrate how filter reduces snoopy messages upon write and read requests in Fig. 7. The request is first broadcast to all the OP-I routers through the optical waveguide. Once an OP-I router receives a remote write/read request, it queries both the cluster and local filter. If there are more than one cluster sharers, the filters spontaneously pick one cluster closest to the requester, such as C_3 in the figure, to forward data. Other cluster sharers will invalidate their local sharers if it is a write request. Note that if there is a dirty owner, both the cluster and the local sharer vector should contain a single "1". Hence, we also implicitly store the dirty owner for requests, covering larger percent of broadcast. We also want to make a note that all filters are always synchronized, which we will discuss with details in Section 3.2.3. Later in Section 5, we will show that our filter together with the predicted local broadcast result in extremely small amount of cache probing while providing direct cache-to-cache transfer for majority of read and write requests. The former is almost comparable to that in a directory based coherence protocol while the latter is similar to that in a snooping protocol.

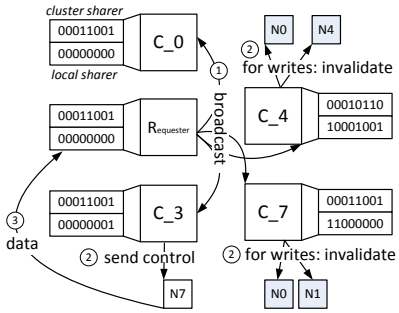

Figure 7: An example of filter access upon a request. A write request has extra local invalidation messages.

We implement the simple MSI coherence states in our protocol to remove remaining snoopy messages. "M" stands for "modify" (dirty data); "S" stands for "share" (clean data) and "I" indicates that no data copy in L2 cache. Keeping directory helps to handle messages such as L2 misses — they cannot benefit from further snooping. Using a coarse-grained vector directory helps to control the storage overhead, making it scalable to large scale CMPs. We use only an 8-bit vector to record either cluster sharers or the precise owner ID of the data.

Putting everything together, upon a local L2 *read* miss, the C-router first sends the request to its OP-I router and consults the predictor to determine if the request should be broadcast locally. If yes, the router sends message to all the local nodes and waits for their responses. Upon a hit, the requester receives data from a local sharer. Comparing to traditional directory protocols, a request with successful local prediction does not use the global network, which saves hop count and is more energy efficient. However, the write requests omit this step due to its low success rate.

For writes and those reads that failed to find sharer/owner in local broadcast, a global broadcast is then necessary. Upon seeing such messages, every OP-I router looks up in the filters trying to locate the corresponding owner/sharer ID. If a sharer/owner is found in the requester's cluster, the router simply forwards the data inside this cluster locally. Otherwise, the cluster (with a sharer) closest to the requester will forward the data. Finally, the OP-I router of the home node's cluster will forward the message to the home node to update the directory.

3.2.3 Message Ordering and Synchronization

The correctness of cache coherence protocols depends on the ordering in handling request racing. For example, two writes requesting the same data should be ordered such that

only one owner exists at any time. In this section we discuss this ordering problem and present our mechanism to guarantee the correctness.

Ordering in global network. If a single transmission channel is used for broadcast, it is essentially a global ordered bus which needs arbitration for multiple requests. To improve the throughput of this network, we used as many channels as the number of nodes: 8. The problem arising from such a multi-bus broadcast network is how to order the broadcast messages sent by different nodes. In a conventional snoopy protocol implemented on a shared bus, the arbitration procedure of the bus will naturally grant permissions in a specific order for the receivers. In a conventional directory-based protocol, the directory in the home node orders the requests. In this paper, we use the filters in OP-I routers as the natural ordering points. Since the requests in the optical network can be received in one cycle (details in section 4), broadcast messages sent at the same cycle will be received by every optical router simultaneously. Next, messages arriving at the same cycle are processed in the same order in all the filters as arbitration algorithms are the same. Thus in the ordered global network, every filter observes the same sequence of broadcast messages.

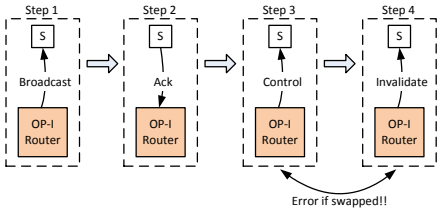

Figure 8: An example of ordering scheme in local network. The in-flight broadcast transaction is preferred over the global write/invalidation requests.

Ordering in local network. When arbitrating among local broadcast request and remote write/invalidation messages, we always check whether there is an in-flight local broadcast request before forwarding global ones to the local nodes. A write request will change cache status, so incorrect order of message handling would result in errors of an ongoing snoopy message. For example in Fig 8, a read request is first broadcast locally by the OP-I router. A local sharer S responds with a positive Ack to the router. Next the router should send a control message to S to require data forwarding. If, however, the router grants a remote invalidation message to the same address over the control message, the data in S would be invalidated before being forwarded to the requester, which is an error. Hence, only after the transaction of local broadcast is finished (the control message is sent or no positive Ack is received), can an invalidation/write message use the local link between the OP-I router and destination nodes. Note that it does not create starvation of remote write requests because the buffer size for the local broadcasts in the router is limited, so is the number of in-flight messages. Besides new broadcast is not issued until current invalidation message is sent. Second, messages sent by directory have fixed index of Virtual Channel (VC) and network layer via address mapping. The directory serves as ordering point if filters miss. The acknowledgments of invalidation and owners are not necessarily in the ordered network. Thus the ordered messages leaving from directory should stay in order until they arrive at the corresponding nodes. The same network layer and virtual channel can ensure that local arbitration in router will not reorder them to maintain the correctness.

Filter synchronization. The main reason of using same filters for every cluster is to let home node be aware if other node (the owner/sharer) is in charge of data forwarding. When the request is broadcast in the global network, filter in the home cluster can tell if the request hits in any other filter and if home node should send control message to the owner node without receiving the responds from other clusters, introducing additional network latency and messages. So in our protocol, filters in different clusters should be synchronized. Otherwise, it may introduce error: different filters indicate different owner/sharer. Filter is updated immediately after it is queried by current request sent via global network. However not all the local broadcast request is used to modify filter in case that they introduces differences. Only successful broadcast request meaning that predicted local broadcast helps find a local sharer and related reply messages could update the filter if the entry already exists and the status of the entry remains the same to avoid changes on replacement. Most of the actions of update are finished with the implication of the read/write requests. The network bandwidth overhead for filter updating is small as only write back message and eviction of sharers in whole cluster result broadcast in optical network. The filters in all the clusters are cache structures which have the same settings, initialization, and replacement policy. Besides, the accessing requests which update the filters are broadcast to all the clusters, arriving at the same cycle and arranged in the same order. Hence, the synchronization of filter is achieved via these organizations. Because filter space is not statically allocated for each cluster. The actual occupancy for the entries belonging to one of the clusters can vary vastly during this period of time. It helps improve the usage efficiency of the filter space.

4. IMPLEMENTATION

4.1 Router Microarchitecture

The OP-I router connects each local cluster to the global optical network. Since our optical network employs SWMR crossbar, each router includes eight input/output local ports, one output port that transfers flit in optical network (through Electric/Optical conversion), and seven input ports from optical detectors. Messages from local nodes are stored in virtual-channel (VC) buffer. If it is sent by the home node, the VC is assigned based on two address bits, as mentioned in section 3.2.3. Otherwise, there is no need for static mapping. After buffering message, the broadcast predictor and route unit are acquired. Based on the prediction result, message is either routed to local nodes or global network and forwarded to the switch allocation (SW) stage. If the message is to be broadcast in global network, the SW is similar to conventional two-stage design. Otherwise, it competes the local output ports with messages from remote clusters.

The messages from optical network go through a crossbar, which is designed for load balance and message ordering, before being stored into a shared buffer [43]. The buffer index of each message is exactly the same in all routers. And the filters process messages in the buffers in the same order, which depends on the arriving time of the message and a priority order based on sender's ID for the simultaneous

requests. To reduce the unfairness, all routers periodically rotate the priority of the clusters.

To obtain the usage of electrical links to the local nodes, the winners from both local and global networks send requests to the round-robin based arbiter. When current local broadcasts do not finish, meaning that the OP-I router does not receive or respond to corresponding reply messages, the arbiter will not grant write and invalidation messages from remote clusters or following local broadcast requests to guarantee the order. However other type requests are processed normally and does not require waiting for on-fighting local broadcasts. Upon passing the arbitration, the messages stored in the buffer go through the second switch to be forwarded to the local ejected ports. And one credit per message is sent back to increment the buffer count in upstream node or router.

To ensure that message arrives at different clusters in the same cycle, the cycle time should accommodate the propagation delay in the worst case. We assume a 400 mm^2 die size. And the light speed is 10.45 ps/mm, the latency of E/O and O/E conversion is 75ps [29]. Therefore the network can support 1GHz adequately. The OP-I router requires 4 pipeline stages: buffering, (prediction and routing)/filter accessing, switch allocation and transferring to downstream router/local node. For the router implemented in directory protocol, it also has 4 stages except that the second stage is replaced by VC allocation. VC help improve the throughput of the un-ordered network, which is feasible for directory-based coherence protocol.

4.2 Power Model

To estimate the power consumption of on-chip network, including electrical and optical components, we adopted the nanophotonics power model in [7]. The total dynamic energy consumption of the optical link is 158 fJ/bit. This includes energy consumption from modulators, detectors, amplifiers, driver and clock circuits. In addition, to maintain corresponding wavelength, micro-rings require consistent heating power as they are sensitive to temperature. This introduces 16-32fJ/bit/heater [25] thermal tuning energy cost.

Our on-chip optical network adopts dense wavelength division multiplexing (DWDM) support such that 64 wavelengths can be transmitted in single waveguide. Each modulator can run at 20 Gb/s bit rate [49]. There are five-layer coherence network of 64-bit width and single data layer of 512-bit width. There are in total 2.6 k micro-rings for both modulators and filters. The same network design is used to evaluate our proposed C^3 protocol and traditional snoopy and directory protocols. The only difference is that traditional full-vector directory protocol has wider invalidation message size and requires an extra of 0.2k rings. To summarize, the thermal power required to tune the rings over a temperature range of 20K is 104 mW and 112 mW [25] for C^3 and directory protocols, respectively.

The power of off-chip laser is large enough to sustain all types of light loss such that the detector can receive sufficient optical power. In table 1, we listed the losses from different optical components [25, 30]. We performed detailed power calculation of our optical network that consists of 4 broadcast network layers for read/write requests, one unicast network for control messages, and one unicast data network. The estimated laser power is 582 mW. The directory requires a multi-cast network with twice bandwidth of ours for invalidation, the laser power is 326 mW as the aggregative broadcast bandwidth is less than ours. In this study, the electrical power estimations of buffers, crossbars and electrical links are modeled at 32nm technology, which is scaled from 45nm PTM HSPICE device models [45] at 1.1V and temperature $90°C$.

Photonic device	Loss (dB)	Photonic device	Loss (dB)
Waveguide loss	0.5/cm	Waveguide bend	0.005
Splitter	0.2	Coupler	1
Modulator insertion	0.1	Detector insertion	0.1
Filter drop	1.5	Ring through	0.01
Laser efficiency	30%	detector sensitivity (μw)	10

Table 1: Optical losses of different optical components [25, 30].

5. EVALUATION

We evaluated and compared the performance and energy-efficiency of proposed C^3 protocol with directory-based, snoopy-based coherence and DASH [31] protocols. In the evaluation, we used the same hierarchical network (see section 3.1.1).

5.1 Simulation Methodology

We extended Noxim [39], a cycle-accurate NoC SystemC-based simulator, to model our proposed hierarchical network (Fig. 2) that uses electrical concentration organization and optical crossbar respectively for intra- and inter- cluster communications. The routing for our topology is deterministic routing within each cluster. We modeled all major components of the network and additional components in C^3: predictors, filters and other control logics. The energy models for both electrical and optical modules are augmented. Table 2 lists other essential parameters.

Execution Pipeline	2GHz, in-order, 2-issue
Core Organization	8 clusters, 8 nodes per cluster, 4 groups per node, 4 cores per group. Total 64 nodes, 256 groups and 1K cores
Private L1I/D	32KB per cluster, 8-way, 64B line, 2-cycle hit time, write-through, shared by 4 cores in the same group
Local Shared L2	2MB per node, 16-way, 64B line, 10-cycle hit time, write-back, shared by 4 clusters in the same node
Global Shared LLC	Distributed 3D stacking, 4 layers, 64 tiles per layer. 2GB, 16-way, 512B line, 30-cycle hit time, write-back
Memory Controller	Four, located in the edges of the chip
Interconnect	Hierarchical optical-electrical network, 8 clusters with 8 nodes per cluster, 1GHz frequency, 4-stage router.

Table 2: Chip configuration.

Since existing OS may not be scalable to thousand-core architecture, we choose a user level simulator to collect cache transaction traces. We extended PTLsim [44] to simulate parallel kernels with up to thousands of threads. The detailed processor parameters are listed in Table 2. We used workloads from PARSEC [9] to evaluate different protocols, similar to the approach in the thousand-core simulation in ATAC [37]. For each PARSEC workload, we used *sim-large* input set and collected all L2 cache transactions into trace files. Four memory controllers are placed along the edges of

the chip. Since a large LLC is integrated in our architecture, the memory bandwidth in our system is not a bottleneck.

We compared C^3 to three widely adopted cache coherence protocol designs:

Snoop-based protocol — `snoopy`. Every request is broadcast to all the nodes through the ordered network. To enable simulation with high core count, we alleviate the bandwidth overhead by removing the respond messages of the snoopy request, and implemented a simple 1-bit directory in LLC to indicate if the line is clean. If the line is dirty, the owner forwards the data copy to the requester, otherwise, the home node is in charge of data forwarding.

Directory protocol — `directory`. We evaluated a full-vector MSI directory protocol. Although a full directory keeps all sharers of the shared cache line and helps to remove unnecessary probings, it brings significant storage and power overheads and needs large invalidation messages. In this protocol, the directory serves as the order point to ensure correctness.

ACKwise protocol — `ATAC` [37]. ATAC is a directory based protocol that only stores limited number (5 in this paper) of sharers, in a 1024-core CMP. Broadcast is still required if sharers are more than 5. The number of sharers is tracked to reduce the acknowledgment messages.

DASH protocol — `DASH` [31]. DASH protocol is a hybrid design of directory and snoopy protocols. Each request is broadcast inside the cluster firstly. If it can not be served locally, it will be sent to global network to access directory. The sharing information in the directory is the bit vector of pointers to clusters. Thus additional local broadcast is necessary to look for the owner node or to invalidate the sharers inside the cluster.

5.2 Filter Configuration

Filter efficiency is the key point of C^3 performance since higher hit rate indicates more direct cache-to-cache accesses and less directory queries. In addition, filter hits reduce the number of write-initiated local broadcasts. The filter efficiency depends on its capacity, set-associativity, replacement policy and throughput. We adopted a recently proposed replacement policy — dynamic re-reference interval prediction (DRRIP) [22], in the filter to achieve better hit rate. Each filter has 2 read/write ports to achieve improved throughput without incurring significant area and power overheads.

We performed a study of hit rates with different configurations of capacity and set-associativity. The estimations of delay, energy and area under different settings are based on CACTI [12]. Both energy consumption and area overhead increase almost linearly with the capacity of filter but much slower with the set-associativity. In our assumption, filter access completes within one cycle. This timing constraint (1ns) limit the number of entries to be less than 16K. Besides, the hit rate shows visible increase when the set count goes up from 1K to 8K. Therefore `C`^{3} employs the 8K-set 16-way filter design which incurs 0.28% of total area overhead.

5.3 Performance Evaluation

In this section, we evaluated the efficiency of filters and broadcast predictors and overall performance of four different cache coherence protocols.

Filter efficiency. We compared the effectiveness of our proposed cache-based filter, to an owner predictor that predicts the locations of dirty lines [1]. The latter is a two-level prediction design. The first-level predictor predicts whether a cache miss can be served by a cache-to-cache forwarding based on the PC of the instruction that generates the miss. The second-level predictor further improves the prediction accuracy with the combination of the PC and the load/store addresses. We chose its default implementation parameters, which has 2K L1 entries and 16K L2 entries. Each L2 entry includes 4 prediction nodes. The first-level table is indexed by the lowest 11-bit of PC. The second-level table is indexed by the XOR result of the tag address of the missing cache line and bits from 2 to 15 of the PC.

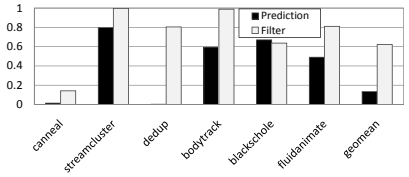

Figure 9: Comparisons of efficiency in locating owner between owner prediction and filter.

Fig. 9 depicts the percentage of owners caught by either predictor or filter. The benchmarks not shown here hardly have any owner status during our experimental phase. The geometric means of success rate are 13.4% and 62.2% respectively. In particular, workloads `facerec` and `freqmine` have above 98% success rates. To summarize, the filter is effective in finding the owners for majority requests.

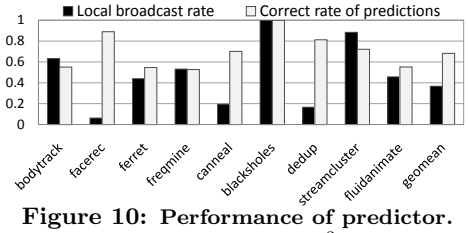

Figure 10: Performance of predictor.

Predictor efficiency. Although `C`^{3} uses 1-bit predictor with trivial overhead, it helps to remove a large number of snoopy and reply messages, as shown in Fig. 10. The geometric mean of broadcast rate from our predictor is 36.7% while `DASH` requires 100% local broadcasts. And the broadcast rate of workload is compatible with local sharing rate. For example, as we can see in Fig. 4, the local sharing rate of `blackscholes` is above 90%, meaning that most local broadcasts are necessary for fast data forwarding. Fig. 10 shows that the broadcast rate of `blackscholes` reaches 99%. The figure also presents the the correct prediction rate of our proposed predictor which is over 50% on average and up to 98% for `blackscholes` case.

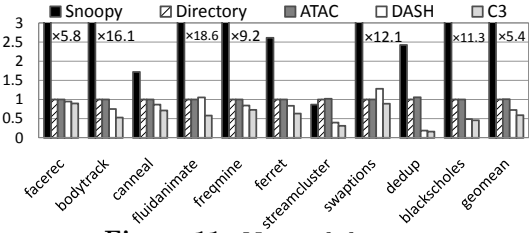

Figure 11: Network latency.

Network latency. We then integrated filters and predictors with our coarse-vector directory, and compared `C`^{3} protocol with three baseline designs. The network latency is the total delay of round-trip for the request and its reply, including the queuing delay at network and cache, the time spent on the optical and electrical network, and the access latency of filters and directory/cache.

Fig. 11 compared the network latency of C^3, snoopy, ATAC, DASH (all are normalized to directory). From the figure, C^3 protocol outperforms directory and ATAC by 41% on average and a maximum of 84% (in streamcluster). It is mainly due to the high hit rates of the filter, which removes remote accesses to the directory. The access latency of L2 cache is lower than that of L3 due to its smaller sizes. Although ATAC requires broadcast when sharers are more than 5, it does not inccur significant network latency compared to full-vector directory since cases with more than five sharers are rare and both the global optical network and local concentration topology support broadcast for invalidation messages.

On average, C^3 outperforms DASH protocol by 18.7%. This is because the failed local broadcast could still gain benefits from filters to visit the owner directly, and successful prediction could reduce redundant local broadcasts.

Snoopy protocol does not perform well almost in every protocol due to the bottleneck of the bandwidth through most of the benchmarks. And the contention in the network results in long queuing delay since the buffers of the router are full and the packet cannot be injected into the network.

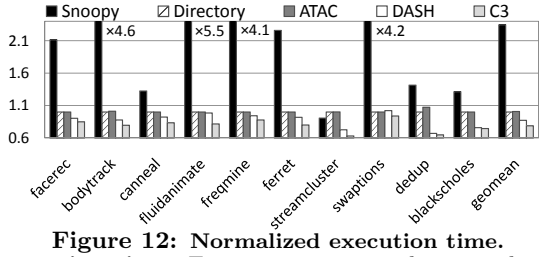

Figure 12: Normalized execution time.

Execution time. Fig. 12 summarizes the normalized execution time of all the protocols (normalized to directory). C^3 protocol consistently outperforms other protocols with a maximum reduction of 35% over tt directory or ATAC and 17% over DASH. This is due to shorter network latency in C^3 protocol (Fig. 11). Snoopy only performs better than directory in streamcluster when the traffic is low. In other cases, snoopy is not a scalable option for a many-core CMP.

5.4 Broadcast Overhead

One major drawback of snoopy is the amount of activities triggered by a broadcast message: all local cache tags are probed for the coherence status of the requested data. This not only generates significant energy overhead but also harms performance. We measured the amount of redundant snooping messages, i.e., those that do not have a copy of requested data and so the lookup is redundant. Over 90% of total broadcast traffic is unnecessary for PARSEC benchmarks [9].

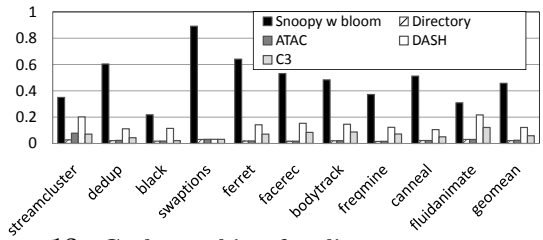

Figure 13: Cache probing for directory, snoopy protocol with bloom filter [47], DASH and C^3, normalized to snoopy protocol without filter.

C^3 adopted a mix of three-level blocking scheme to effectively remove most of the redundancy: (1) local broadcast predictor is able to reduce the unnecessary local broadcasts. Successful local broadcasts also prevent global broadcasts; (2) The filters in OP-I routers also reduces the global snoopy; (3) coarse-grained vector directory indicates the location of the owner or cluster index of shares to avoid snooping every cache. The comparisons with other protocol designs are shown in Fig. 13. One can see that the predictor removes about half number of messages and is able to reduce the number of snoopy messages to only 5.8% and 12% on average of snoopy, with and without bloom filter, respectively. In ATAC, if the number of sharers is over five, broadcasting invalidation messages could generate more traffic than in full-map directory (e.g. streamcluster), though overall, ATAC is low on cache probing. Redundancy only exists in two cases of C^3 compared to precise full-map directory protocol: (1) local broadcast to look for sharer/owner and (2) invalidation. So the snoopy traffic is still higher than directory protocol, however the storage overhead is only 16.4% of a baseline directory.

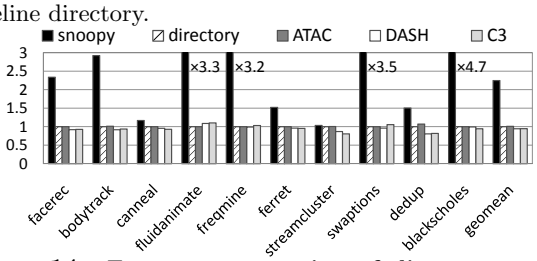

Figure 14: Energy consumption of directory, snoopy protocol with bloom filter [47], DASH and C^3, normalized to snoopy protocol without filter.

5.5 Energy Comparison

We also estimated the energy consumption of on-chip optical and electrical networks in different protocols. Fig. 14 demonstrates that both C^3 and DASH save about 5% energy over directory. This is because: (1) they shorten the execution time, which results in the overall reduced leakage energy consumption. (2) the average path of coherence messages in C^3 and DASH is shorter as most requests can be served either by a local broadcast or by the filter. This helps to reduce the communication energy further. (3) coarse-grain directory requires small message size (8-bit vs 64-bit in the baseline directory) and less number of micro-rings to transfer the message, thus C^3 saves both communication and heating power. On average C^3 has slightly larger energy consumption than DASH.

6. PRIOR ART

There are a myriad of prior research works on coherence protocol optimization, network optimizations for better cache coherence management and nanophotonic based interconnection designs. In this section, we categorize and discuss those works.

Optical interconnect. Optical interconnect (OP-I) often adopts crossbar topology to achieve fast all-to-all connection [29, 49]. The overhead of micro-rings becomes a serious issue for high node count, requiring high optical and thermal tuning power. Different topologies such as Clos [25] or mesh [14] were proposed to reduce the number of optical devices although with an increase of dynamic energy consumption in intermediate routers. Adopting hybrid networks of both metal and OP-I connections, utilizing OP-I in global communication [37, 42], and/or clustering [29] were

proposed to shrink the crossbar size. A reservation mechanism [42] was proposed to tune off the idle rings to avoid optical energy loss of broadcast on the data channels. Flex-iShare [43] and wavelength-based routing [30] presented the topologies that allow channel or wavelength sharing among the network nodes to improve the power-efficiency. However, they both have to deal with contention introduced by multiple senders, which results in additional network latency due to waiting for the dedicated token or acknowledgment.

For snoopy-based coherence traffic, Corona [49] and optical bus [29] leveraged single broadcast bus and arbitration scheme while spectrum [32] employed antennas to form multiple broadcast channels. In this paper, we adopted contention-free SWMR [42] crossbar topology and the reservation scheme for the wide data channel to save optical power, considering its broadcast support and low design and fabrication complexity. But we also leverage the benefit of the topology and optical technology to introduce a new cache coherence policy.

Cache coherence protocol. A major obstacle that restricts the adoption of directory-based protocol in large-scale CMPs is its prohibitive storage overhead as the size of full-map directory scheme [13] increases linearly with the core count. Optimizations have been proposed to deal with this problem, e.g., the schemes that save limited pointers with broadcast backup [3, 37], WayPoint that uses sparse directory [26] to allow evicting entries to memory, and tagless directory (TL) [52] that builds sharing information based on a set of bloom filters. Coarse vector scheme has been proven to have similar or even better performance than limited pointer scheme and generate less traffic than sparse directory [18]. Thus we employ the coarse vector as one of building blocks in this paper, which enables us to achieve comparable performance as snoopy protocol but with small storage overhead.

The major challenge to scale snoopy protocol is that enormous cache tag look-ups consume significant network bandwidth and power. Several schemes have been proposed [4, 38, 47] to address this problem through filtering redundant messages. However, when scaling this approach to a thousand cores, challenges remain such as the dilemma between filter storage space and its efficiency, and network overhead of updating traffics for previous filter designs. In this paper, we analyze the traffic of thousand-core CMPs using real traces and propose a mix of three building components, (1) local broadcast predictor, (2) filters in the router, and (3) coarse-vector directory, to eliminate the majority of unnecessary broadcast messages.

Another difficulty to implement snoopy protocol is to maintain ordering in the network. Uncorq [46] presented an approach to ensure the correctness for ring-based network, which does not scale well when the number of cores grows to one thousand. Token coherence protocol [34, 35] can achieve direct cache-to-cache transfer in un-ordered network by decoupling the performance from correct substrate. The order is guaranteed by requesting and forwarding tokens. However tokenB protocol requires additional storage for tokens and broadcasting requests. Atomic coherence [50] serializes the transactions with mutexes to reduce additional racing transitions in directory protocol but brings performance penalty due to atomic coherence and large number of mutexes occupy non-trivial bandwidth. In our case, we implemented an ordered network to ensure the correctness.

Several hybrid cache coherence protocols have been previously proposed. DASH [31] uses bus-based snoopy to maintain consistency for intra-cluster cache coherence and full-vector directory for inter-cluster coherence. However, every coherence request demands at least one local broadcast which brings unnecessary snoopy messages and extra network latency. If local broadcast fails, additional access to directory is always required. In our scheme, broadcast predictor and filters introduce much less overhead and harvests more direct cache-to-cache accesses. The hybrid coherence protocol proposed in [19] decouples the read and write traffic to reduce contentions on the broadcast network. The directory scheme is used exclusively for read over optical fully-connected network while the remaining messages are handled by snoopy-based protocol. This results in worse write latency than that through a shared bus. Our work differs from this design as follows: (1) The C^3 protocol is not designed to speedup any specific request type. Both read and write requests can benefit from snoopy as it eliminates the extra indirection. (2) Write requests in C^3 do not always require global broadcasting as both the filter and the directory help to remove unnecessary broadcast messages.

Bandwidth adaptive snooping hybrid (BASH) [33] is a protocol that employs snoopy protocol when network bandwidth is plentiful and switches to directory protocol when bandwidth becomes limitation. It neither addresses the probing overhead problem in snoopy protocol nor optimizes the performance under heavy traffic loads. In contrast, our scheme let one request achieve direct access without looking up all the cache-tags. Owner prediction [1] and sharer prediction [2] introduced an approach to improve the performance of directory protocol by predicting the owner or sharers. Their works were motivated by some observations, such as the total number of sharers is small, 1 in some cases, are based on a small-scale CMP. It may not stand in a large-scale CMP or use a different application. Besides, the accesses extend to the 4-step when the prediction fails. Moreover, the prediction is not always allowed [1]. The filter in our scheme shows better performance than the predication in section 5.

7. CONCLUSION

The conventional snoopy and directory coherence protocols face several challenges when scaling to many-core or thousand-core CMPs. Snoopy protocol incurs enormous amount of cache tag probing. Directory protocol requires an indirection through directory in all coherence requests, and potentially large storage overhead from the directory itself. We observed that while the emerging OP-I network provides high bandwidth density, low propagation delay and natural support for multicast, it cannot satisfy the overwhelming broadcast requests in snoopy protocols.

In this paper, we addressed the efficiency problem by proposing C^3, a composite cache coherence (C^3) protocol that integrates a set of of effective building blocks including local broadcast predictor, filtering, and a coarse-grained directory, to improve the performance of the coherence transactions on large-scale CMPs. C^3 helps to achieve direct cache-to-cache accesses as in snoopy protocol and small amount of cache probing as in directory protocol. Our experimental results showed that on average, it reduces the snoopy messages to only 5.8% of that in snoopy protocol, improves performance by 21%, reduces network latency of coherence messages by 41% and saves network energy consumption

by 5.5%. Thus the proposed C^3 protocol with hierarchical optical interconnect support is a promising candidate for implementing cache coherence scheme in large-scale CMPs.

8. ACKNOWLEDGMENT

This work is supported in part by NSF grants CAREER CNS-0747242 and CNS-1012070.

9. REFERENCES

[1] M. E. Acacio, et. al., "Owner Prediction for Accelerating Cache-to-Cache Transfer Misses in a cc-NUMA Architecture," In *SC,* 2002.
[2] M. E. Acacio, et. al., "The use of Prediction for Accelerating Upgrade Misses in a cc-NUMA Multiprocessors," In *PACT,* pp. 155-164, 2002.
[3] A. Agarwal, et. al., "An Evaluation of Directory Schemes for Cache Coherence," In *ISCA,* pp.353-362, 1988.
[4] N. Agarwal, et. al., "In-Network Coherence Filtering: Snoopy Coherence without Broadcasts," In *MICRO,* 2009.
[5] N. Agarwal, et. al., "In-Network Snoop Ordering: Snoopy Coherence on Unordered Interconnects," In *HPCA,* 2009.
[6] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp onchip networks," In *ICS,* pp.187-198, 2006.
[7] C. Batten and et. al., "Building Manycore Processor-to-DRAM Networks with Monolithic Silicon Photonics," In *High Performance Interconnects,* pp.21-30, 2008.
[8] S. Beamer, et. al., "Re-Architecting DRAM Memory Systems with Monolithically Integrated Silicon Photonics," In *ISCA,* pp.117-128, 2010.
[9] C. Bienia, et. al., "The parsec benchmark suite: Characterization and architectural implications," In *PACT,* pp.72-81,2008.
[10] B. Black, et. al., "Die stacking (3d) microarchitecture," In *MICRO* pp. 469-479, 2006.
[11] S. Borkar, "Thousand core chips - a technology perspective," In *DAC,* pp.746-749, 2007.
[12] CACTI, http://www.hpl.hp.com/research/cacti/
[13] L. M. Censier and P. Feautrier, " A New Solution to Coherence Problems in Multicache Systems," In *IEEE Trans. on Computers,* pp. 1112-1118, 1978.
[14] M. J. Cianchetti, et. al., "Phastlane: A Rapid Transit Optical Routing Network," In *ISCA,* pp.441-450, 2009.
[15] S. Chaudhry, et. al., "Rock: A High-Performance Sparc CMT Processor," In *IEEE Micro*, 29(2):6-16, 2009.
[16] W. J. Dally and B. Towles, "Principles and practices of Interconnection Networks," Morgan Kaufmann, 2004.
[17] N. Eisley, et. al., "In-network cache coherence," In *MICRO,* pp. 321-332, 2006.
[18] A. Gupta, et. al., "Reducing Memory and Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes," In *ICPP,* pp. 312-321, 1990.
[19] J.-H. Ha and T. M. Pinkston, " A Hybrid Cache Coherence Protocol for a Decoupled Multi-Channel Optical Network: SPEED DMON, " In *ICPP,* pp.164-171, 1996.
[20] L. Hammond, et. al., "A single-chip multiprocessor," In *IEEE Computer,* 30(9):79-85, 1997.
[21] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors," http://www.itrs.net/Links/2009ITRS/Home2009.htm, 2009.
[22] A. Jaleel, et. al., "High performance cache replacement using re-reference interval prediction (RRIP)," In *ISCA,* pp.60-71, 2010.
[23] N. Enright-Jerger, et. al., " Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support," In *ISCA,* 2008.
[24] N. Enright-Jerger, et. al., "Virtual Tree Coherence: Leveraging Regions and In-Network Multicast Trees for Scalable Cache Coherence," In *MICRO,* 2008.
[25] A. Joshi, et. al., "Silicon-Photonic Clos Networks for Global On-Chip Communication," In *NOCS,* 2009.
[26] J. H. Kelm, et. al., "WAYPOINT: scaling coherence to 1000-core architectures," In *PACT,* pp. 99-109, 2010.
[27] T. Kgil, et. al., "Picoserver:Using 3d stacking technology to enable a compact energy efficient chip multiprocessor," In *ASPLOS,* pp. 117-128, 2006.
[28] J. Kim, et. al., "Flattened butterfly topology for on-chip networks," In *MICRO,* 2007.
[29] N. Kirman, et. al., " Leveraging optical technology in future bus-based chip multiprocessors," In *MICRO,* pp. 492-503, 2006.
[30] N. Kirman and J. Martinez, "An efficient all-optical on-chip interconnect based on oblivious routing," In *ASPLOS,* 2010.
[31] D. Lenoski, et. al., " Design and Scalble Shared-Memory Multiprocessors: The DASH Approach," In *COMPCON,* pp. 62-67, 1990.
[32] Z. Li, et. al., "Spectrum: A Hybrid Nanophotonic-Electric On-Chip Network," In *DAC,* pp. 575-580, 2009.
[33] M. M. K. Martin, et. al., "Bandwidth Adaptive Routing," In *HPCA,* 2002.
[34] M. M. K. Martin, et. al., "Token Coherence: Decoupling Performance and Correctness," In *ISCA,* 2003.
[35] M. Marty, et. al., "Improving multiple-cmp systems using token coherence," In *HPCA,* 2005.
[36] D. Miller, "Rationale and Challenges for Optical Interconnects to Electronic Chips," In *Proceedings of the IEEE,* 88(6):728-749, 2000.
[37] G. Kurian, et. al., "ATAC: A 1000-core cache-coherent processor with on-chip optical network," In *PACT,* pp.447-488, 2010.
[38] A. Moshovos, et. al., "JETTY: Filtering snoops for reduced energy consumption in SMP servers," In *HPCA,* pp.85-96, 2001.
[39] "Noxim, An Open Network-on-Chip Simulator," http://noxim.sourceforge.net
[40] nVidia, "Quadro fx 3700m, " `http://www.nvidia.com/object/product_quadro_fx_3700_m_us.html.`
[41] K. Olukotun, et. al., "The case for a single-chip multiprocessor," In *ASPLOS,* pp. 2-11, 1996.
[42] Y. Pan, et. al., " Firefly: Illuminating Future Network-on-Chip with Nanophotonics," *Int. Symp. on Computer Architecture, ISCA'09,* pp. 429-440, 2009.
[43] Y. Pan, et. al., "FlexiShare: Channel Sharing for an Energy-Efficient Nanophotonic Crossbar," *In Int. Symp. on High-Performance Computer Architecture (HPCA),* 2010.
[44] PTLsim. http://www.ptlsim.org/
[45] PTM interconnect model. http://www.eas.asu.edu/ ptm/ interconnect.html
[46] K. Strauss, et. al., "Uncorq: Unconstrained snoop request delivery in embedded-ring multiprocessors," *In Int. Symp. on Micorarchitecture,* pp. 327-342, 2007.
[47] A. N. Udipi, et. al., "Towards Scalable, Energy-Efficient Bus-Based On-Chip Networks," *In Int. Symp. on High-Performance Computer Architecture (HPCA),* pp. 1 - 12, 2010.
[48] S. Vangal, et. al., "An 80-tile 1.28tflops network-on-chip in 65nm cmos," *In IEEE Int. Solid-State Circuits Conf.,* pp. 98-590, 2007.
[49] D. Vantrease, et. al., "Corona: System implications of emerging nanophotonic technology," *In Int. Symp. on Computer Architecture,* pp.153-164, 2008.
[50] D. Vantrease, et. al., "Atomic Coherence: Leveraging Nanophotonics to Build Race-Free Cachec Coherence Protocols," *In Int. Symp. on High-Performance Computer Architecture (HPCA),* 2011.
[51] J. Xue, et. al., "An Intra-Chip Free-Space Opitcal Interconnect," *Int. Symp. on Computer Architecture, ISCA* 2010.
[52] J. Zebchuk, et. al., "A Tagless Coherence Directory," *In Int. Symp. on Microarchitecture, MICRO,* pp. 423-434, 2009.