Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2009-018 May 5, 2009

ATAC: A Manycore Processor with On-Chip Optical Network

Jason Miller, James Psota, George Kurian, Nathan Beckmann, Jonathan Eastep, Jifeng Liu, Mark Beals, Jurgen Michel, Lionel Kimerling, and Anant Agarwal

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

ATAC: A Manycore Processor with On-Chip Optical Network

Jason Miller, James Psota, George Kurian, Nathan Beckmann, Jonathan Eastep,
Jifeng Liu, Mark Beals, Jurgen Michel, Lionel Kimerling, Anant Agarwal
Massachusetts Institute of Technology, Cambridge, MA
CSAIL Technical Report, April 2009

Abstract

Ever since industry has turned to parallelism instead of frequency scaling to improve processor performance, multicore processors have continued to scale to larger and larger numbers of cores. Some believe that multicores will have 1000 cores or more by the middle of the next decade. However, their promise of increased performance will only be reached if their inherent scaling and programming challenges are overcome. Meanwhile, recent advances in nanophotonic device manufacturing are making chip-stack optics a reality—interconnect technology which can provide significantly more bandwidth at lower power than conventional electrical analogs. Perhaps more importantly, optical interconnect also has the potential to enable new, easy-to-use programming models enabled by an inexpensive broadcast mechanism.

This paper introduces ATAC, a new manycore architecture that capitalizes on the recent advances in optics to address a number of the challenges that future manycore designs will face. The new constraints—and opportunities—associated with on-chip optical interconnect are presented and explored in the design of ATAC. Furthermore, this paper introduces ACKwise, a novel directory-based cache coherence protocol that takes advantage of the special properties of ATAC to achieve high performance and scalability on large-scale manycores. Early performance results show that a 1000-core ATAC chip achieves a speedup of as much as 39% when compared with a similarly sized manycore with an electrical mesh network.

1. Introduction

The trend in modern microprocessor architectures is clear: multicore is here. As silicon resources become increasingly abundant, processor designers are able to place more and more cores on a chip with massive multicore chips on the horizon. Many industry pundits have predicted manycores with 1000 or more cores by the middle of the next decade. But will current processor architectures (especially their interconnection mechanisms) scale to thousands of cores and will programming such systems be tractable? This paper argues that current multicore architectures will not scale to thousands of cores and introduces ATAC (pronounced ā-tack), a new processor architecture that addresses these issues. ATAC integrates an on-chip optical broadcast communication network within a mesh based tiled multicore architecture to significantly improve the performance, energy scalability, and ease of programmability of multicore processors [1].

Current multicore architectures will not allow performance to scale with Moore's Law for several important classes of parallel applications. Although Moore's Law enables increasing numbers of cores on a single chip, the extent to which they can be used to improve performance is limited both by the cost of communication among the cores and off-chip memory bandwidth. Although our research is investigating the application of optical interconnect technology to both problems, this paper focuses on the on-chip interconnect challenge. As computation is spread across multiple cores on a chip, distribution of instructions to the cores, and communication of intermediate values between cores account for an increasing fraction of execution time. As processors scale to larger numbers of cores, wire delay causes the cost of communication between any two cores to grow relative to the physical separation of those cores. This effect can be multiplied if communications between different pairs of cores interfere with each other by contending for communication resources. The outlook is particularly dismal for applications that require a lot of global communication operations (e.g., broadcasts to maintain cache coherence) because each such operation ties up many resources. Besides performance, global communication operations such as broadcast can also be power-hungry since the electrical signal must be copied many times.

State-of-the-art multicore chips employ one of two strategies to deal with interconnection costs. Small-scale multicores typically interconnect cores using a bus. This simple design, where communication costs are small and uniform, does not scale efficiently to larger numbers of cores. As the number of cores on a bus increases, the length of the bus wires increase, forcing the use of a slower clock. Also, since all cores share the same bus, contention increases very quickly. A more scalable interconnection strategy used by some multicores is a point-to-point network where communication is exposed to software. For example, the Raw microprocessor [17] consists of a 2-D array of cores where each core can communicate di-

rectly with its neighbors. This avoids long global wires but communication between distant cores requires multiple hops. This design allows for much less contention as long as communication patterns do not physically overlap in the mesh. However, for applications with irregular, statically unpredictable or broadcast communication patterns, contention is unavoidable and will become unacceptable as processors are scaled to thousands of cores.

The scalability of today's multicore architectures is also threatened by the challenge of programming them. Multicore programmers must spatially and temporally orchestrate computation and communication if they want to extract high performance from the hardware. Regardless of which existing network topology is used, dynamic communication patterns which result in network contention will make it extremely difficult to coordinate hundreds or thousands of processors. Even simple functions like supplying instructions to all the cores is difficult, even for the simple case where all cores run the same program (this popular approach is called SPMD, single program, multiple data, programming). Further, current multicore programming techniques do not provide mechanisms that are sufficiently fine-grained for scheduling inter-core communication for high performance. Broadcast and all-to-all communication operations inherent in many coherence, instruction distribution, and synchronization protocols present even greater challenges since most existing scalable multicore network architectures do not support these operations intrinsically, but rather emulate them via multiple point-to-point messages.

The ATAC processor architecture addresses these contention and programmability issues using on-chip optical communications technologies to augment electrical communication channels. Current research in optical communications technologies is making vast strides at integrating optoelectronic components with standard CMOS fabrication processes. ATAC leverages these advances to eliminate communication contention using Wavelength Division Multiplexing (WDM). WDM allows a single optical waveguide to simultaneously carry multiple independent signals on different wavelengths. For example, a single waveguide with the same switching speed as its electrical counterpart and with 64 WDM channels would match the bandwidth of a 64-bit electrical bus. Optical waveguides, however, can also transmit data at higher speeds than electrical wires (a function of the index of refraction of the waveguide material for the optical signal; and a function of the RC delays, the dielectric materials (SiO_2) surrounding the wires, and the delay of required repeaters for the electrical signal). This virtually eliminates the heterogeneous distance-dependent cost function for communication between a pair of cores on any electrical interconnect, which complicates multicore programming. Optical signaling can also use less power than electrical signaling (especially for long wires) because optical waveguides have relatively low loss and therefore do not require periodic repeaters.

The ATAC architecture capitalizes on the strengths of

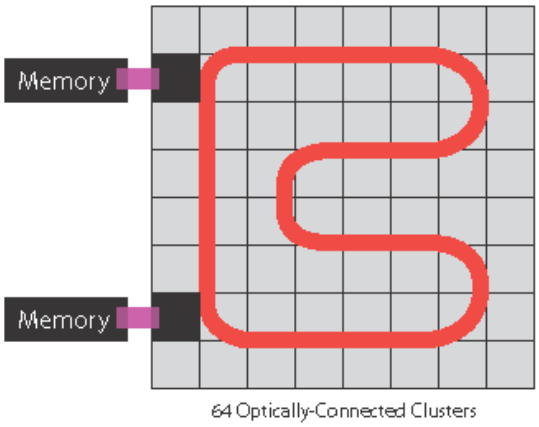

Figure 1. High-level ATAC architecture

nanophotonic technology to implement a high-performance, low-latency, contention-free global communication network. The network consists of a set of optical waveguides that snake through the chip and loop around on themselves to form a continuous loop as shown in Figure 1. Optical Hubs transmit data by modulating a laser light source and injecting it into the loop. The signal then quickly propagates around the loop and can be received by all of the other Hubs in a single operation. Thus every message on the optical network has the potential to be a highly efficient global broadcast. Filtering at the receiving Hubs can limit the scope of the message to create multicast or unicast messages.

While this paper focuses on the architecture of ATAC and how a cache coherence protocol could best utilize such an architecture, note that ATAC enables new programming models that could significantly improve the programmability of large scale manycores. For instance, ATAC provides an inexpensive broadcast operation, which allows new programming models that encourage lots of on-chip data sharing and efficient instruction distribution in the simple SPMD programming style. The broadcast mechanism also enables large-scale cache coherence where widespread sharing isn't frowned upon. New run time systems and APIs that take advantage of inexpensive global communication operations could substantially improve programmers' productivity and ability to achieve great performance without herculean effort. Issues surrounding programming ATAC are outside the scope of this paper and are the subject of ongoing research.

Using these new optical technologies, ATAC processors have the potential to decrease programming difficulty, raise power efficiency, provide high bandwidth (both on-chip and off-chip), and enable performance scalability. The remainder of this paper is organized as follows. Section 2 provides context for the ATAC architecture, focusing on the constraints imposed on the architecture by the optical and electronic devices. Section 3 provides an overview of the ATAC architecture, in-

cluding its processing, communication, and memory mechanisms. Section 4 introduces the ACKwise cache coherence protocol. Section 5 evaluates the ATAC architecture using the ACKwise protocol and provides a preliminary set of results, focusing on how ATAC enables high performance cache coherence across 1000 cores. Section 6 follows with a detailed discussion of related work, and Section 7 concludes the paper.

2. Optical Technology Background

Over the past few decades optical interconnection technology has found its way from long-haul telecommunications to wide area networks to enterprise backbones to datacenters and, more recently, to individual computer racks [16].

Recent advances in electronic-photonic integration continue this trend of greater integration, smaller distances, and higher bandwidths [13], [11], [10]. Optical interconnect is starting to be seen at the board and module level. Recent research [12] has shown that optical devices can be built using standard CMOS processes and optics will soon begin to replace global wires and on-chip buses [2]. The integration of photonic interconnects into chips has the potential to address some of the greatest challenges facing future large-scale multicore processors.

The ATAC architecture is enabled by these recent advances in electronic-photonic integration. ATAC has been designed with a substantial understanding of the limits of both state-of-the-art and soon-to-come optical devices. This section presents a brief overview of these devices and their contraints. The key elements in a nanophotonic network such as the one employed by the ATAC chip include: the "optical power supply" light source; waveguides to carry optical signals; modulators to place signals into the waveguides; and detectors to receive signals from the waveguides. This section discusses each of these components and describes the complete path for transmitting data optically.

In ATAC the light source, or "optical power supply", is generated by off-chip lasers and coupled into an on-chip waveguide. On-chip light sources exist, but consume large quantities of precious on-chip power and area. The power consumption of an off-chip laser is roughly 1.5 W, with 0.2 W of optical power ending up in the on-chip waveguide. Multiple lasers can be used to generate an array of wavelengths, useful for Wavelength Division Multiplexing (WDM) schemes.

Waveguides are the on-chip channels by which light is transmitted. They guide and confine light by a combination of a high-refractive-index material on the inside of the waveguide and a low-refractive-index material on the outside (the cladding). Waveguides can be made out of either silicon (Si) or polymer. Due to the fact that Si waveguides can be packed onto a chip at much higher densities and that modulators for Si can be made much more compactly, the ATAC design employs Si waveguides. These waveguides can be manufactured in a standard CMOS process, as both the waveguide and cladding materials are commonly used elsewhere. ATAC requires waveguides with losses of less than 0.3dB/cm and total power capacity of about 10 mW, both of which are achievable with Si.

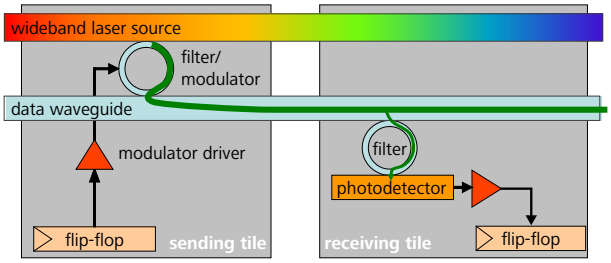

Figure 2. Optical transmission of one bit between two cores

To communicate information on a waveguide, several components are used: the optical power supply, an optical filter, a modulator, and the electronic modulator driver. The optical filter is a ring resonator that couples only a specific wavelength from the power supply waveguide to the data waveguide. The exact wavelength, as well as the spacing between wavelengths, is determined by the ring resonator dimensions. Further tuning can be achieved by changing the ring's temperature or by injecting charge into the ring. The modulator is an optical device that imprints a digital signal on the laser light by varying the absorption in the device. Modulators are used to translate an electrical signal (amplified by the modulator driver) into an optical signal, and can therefore be thought of as an "optical switch", placing values onto optical waveguides. The modulators used in the ATAC design have characteristics that are expected to be reached by designs available in 2012: insertion loss of 1dB; area less than 50 μm^2; modulation rate higher than 20 Gbps; energy required to switch less than 25 fJ; and average power consumption of 25 μW at 1 GHz [9].

At the receiving end of a waveguide additional components are used to receive the signal and convert it to an electrical signal. An optical filter (also known as a "ring resonator"), is used to extract light of a particular wavelength from the data waveguide and transfer it to a photodetector. As with modulators, optical filters must be tuned to "listen" to a particular frequency during manufacturing. The photodetector is an extremely sensitive optical device which absorbs photons and outputs an electrical signal. The photodetector proposed for ATAC at the 11nm node has a responsitivity of greater than 1 A/W and 3dB bandwidth performance at larger than 20 GHz. It has an area footprint of less than 20 μm^2. Furthermore, the expected capacitance of the photodetector is less than 1 fF [7]. In current technology nodes, the output of the photodetector would need to be amplified by a power-hungry TIA (transimpedance amplifier) before it could be used to drive a digital circuit. However, starting with the 22nm node, the smaller transistor input

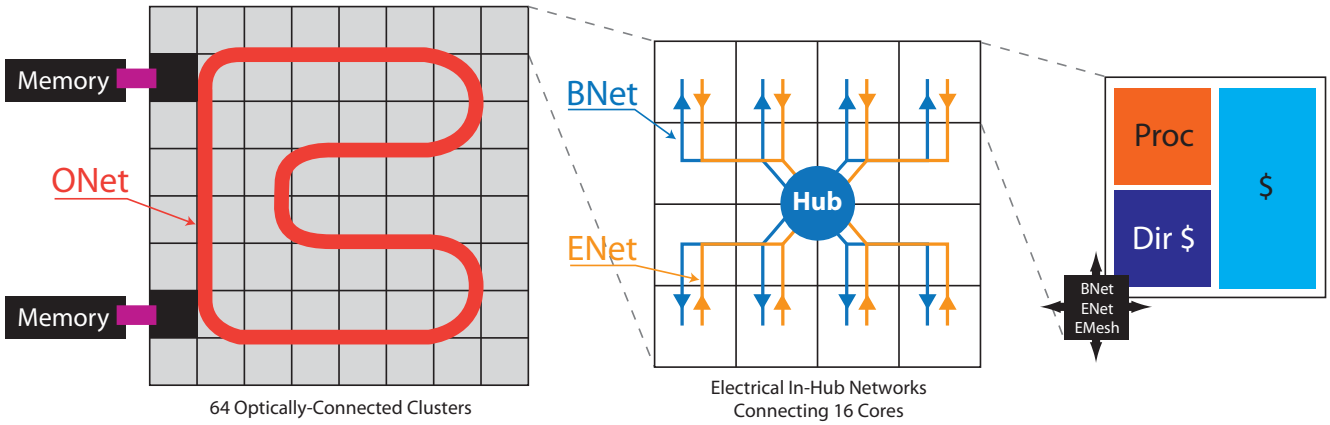

Figure 3. ATAC architecture overview

capacitances will allow the photodetector to directly drive a digital circuit, greatly reducing power consumption.

Figure 2 puts all of these elements together, showing how one bit is transmitted from a flip-flop of one core to a flip-flop of another core. In this figure, the core on the left shows the components relevant to sending and the core on the right shows the components relevant to receiving; however, in the actual chip all cores would contain both sets of components. From end to end, the process for sending a bit on the ATAC's optical network is as follows. The flip-flop signals the modulator driver to send either a 0 or a 1. The modulator driver, which consists of a series of inverter stages, drives the modulator's capacitive load. The modulator couples light at its pre-tuned wavelength λ_i from the optical power source and encodes either a 0 or 1 onto the data waveguide. The optically-encoded data signal traverses the waveguide at approximately one-third the speed of light and is detected by a filter that is also tuned to wavelength λ_i. Photons are detected by the photodetector and received by a flip-flop on the receiver side. Note that Figure 2 shows where a TIA would be needed to amplify the photodetector output, even though it would not be necessary for an ATAC chip targeting the 11nm technology node.

3. Architecture Overview

The ATAC processor architecture is a tiled multicore architecture combining the best of current scalable electrical interconnects with cutting-edge on-chip optical communication networks. The tiled layout uses a 2-D array of simple processing cores, each containing a single- or dual-issue, in-order RISC pipeline, L1 data and instruction caches. ATAC uses a sophisticated directory-based cache coherence scheme which is described later. A portion of the distributed cache-coherence directory is also located in each core. The ATAC architecture is targeted at an 11nm process in 2019, and will have at least 1000 cores (for the purposes of this paper, we assume a 1024-core chip).

The cores in an ATAC processor are connected through two networks: the electrical EMesh and the optical/electrical ANet. The EMesh is a conventional 2-D point-to-point electrical mesh network like those seen in other multicore processors [17, 6]. The EMesh is ideal for predictable, short-range communication. The ANet employs state-of-the-art optical technology to enable low-latency, energy-efficient, contention-free global communication. The core of the ANet is the all-optical ONet shown in Figure 3. The ANet also contains two small electrical structures called the ENet and BNet that are used to interface with the ONet. The ANet is especially useful for long-distance communication or global operations such as broadcasts. Other projects have studied networks like the EMesh in detail so the remainder of this paper focuses primarily on the ANet.

3.1. ONet Optical Network

The key to efficient global communication in a large ATAC chip is the optical ONet. The ONet provides a low-latency, contention-free connection between a set of optical endpoints called Hubs. Hubs are interconnected via waveguides that visit every Hub and loop around on themselves to form continuous rings (see Figure 3). Each Hub can place data onto the waveguides using an optical modulator and receive data from the other Hubs using optical filters and photodetectors. Because the data waveguides form a loop, a signal sent from any Hub will quickly reach all of the other Hubs. Thus every transmission on the ONet has the potential to be a fast, efficient broadcast.

To avoid having all of these broadcasts interfere with each other, the ONet uses wavelength division multiplexing (WDM). Each Hub has modulators tuned to a unique wavelength to use when sending and contains filters that allow it to receive signals on all the wavelengths. This eliminates con-

tention and the need for arbitration in the optical network. In addition, the improved propagation speed of optical signals eliminates the heterogeneous, distance-dependent cost of communication between cores; any pair of Hubs on the chip can communicate with low, fixed latency instead of the one-cycle-per-hop delay found in point-to-point networks. Taken together, these features mean that the ONet is functionally similar to a fully-connected, bi-directional point-to-point network with an additional broadcast capability.

WDM is a key differentiator of the ATAC architecture from a performance scalability perspective. WDM allows a single waveguide to simultaneously carry bits of many overlapping communications. To contrast, an electrical wire typically carries a single bit. Whereas ATAC may share a single waveguide medium between a large number of simultaneous communication channels, implementing multiple simultaneous communication channels in the electrical domain requires additional physical wires. For network operations that are expensive to implement in the electrical domain (such as broadcast), the ATAC approach greatly improves efficiency.

The ATAC architecture was carefully designed taking into account the physical limitations and constraints of both the optical (see Section 2) and electronic devices. Based on these constraints, the ONet as described above should scale to at least 64 (and possibly as many as 100) Hubs. This limit is based on several factors: 1) the total range of wavelengths over which the optical devices can be tuned divided by the minimum spacing between wavelengths, 2) the total amount of optical power a waveguide can carry divided by the minimum amount that each photodetector needs to receive to reliably register a signal, and 3) the maximum length of a waveguide based on the acceptable propagation losses.

These limits can be overcome using multiple waveguides and dividing the communication channels between them. However, eventually the area needed for the optical components will become the limiting factor. The ONet's optical components and photonic interconnect can be placed on a separate layer in the CMOS stack, and can therefore overlap the electrical components to which they connect. However, for a 400 mm^2 chip, the entire area would be consumed by an ONet with approximately 384 Hubs. Since we believe that chips will eventually grow to thousands of cores, some sharing of Hubs will certainly be needed. Therefore, for the purposes of this paper, we take the simple approach and assume that the ONet is limited to 64 Hubs.

Because of this limit, the set of 1024 cores are broken into 64 clusters of 16 cores that each share an optical Hub. The resulting architecture can be seen in Figure 3. The ONet interconnects 64 symmetric clusters with a 64-bit wide optical waveguide bus that snakes across the chip. Each cluster contains 16 cores and an ONet Hub. Within a cluster, cores communicate electrically with each other using the EMesh and with the Hub using two networks called the ENet and BNet. The ENet is an electrical mesh that is used only to send data

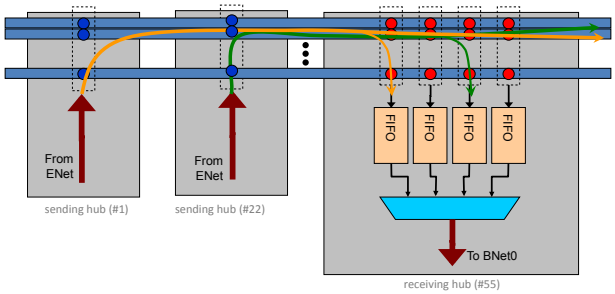

Figure 4. Hub-to-hub communication over the ONet

from cores within a cluster to the Hub for transmission on the ONet. The BNet is an electrical broadcast tree that is used to forward data that the Hub receives from the ONet down to the cores.

Sending data using the ANet is shown in more detail in Figure 4. Messages from the cores arrive at the Hubs on the ENet. Each Hub then retransmits the data on the ONet using its unique wavelength. Note that this allows the two Hubs shown to send their data simultaneously without interference. The ONet consists of a bundle of waveguides: 64 for data, 1 for backwards flow control, and several for metadata. The metadata waveguides are used to indicate a message type (e.g.,memory read, barrier, raw data) or a message tag (for disambiguating multiple messages from the same sender). The receiving Hub captures both of the values simultaneously into sender-Hub-specific FIFOs. These values are then propagated to the cores using the BNet.

The broadcast mechanism of the ATAC architecture is another key differentiator. Optical technology provides a way to build fast, efficient broadcast networks whereas electrical mechanisms do not. When using optical components instead of electrical components, signals may travel farther and be tapped into by more receivers before they need be regenerated. With electrical components, regeneration is accomplished via buffers or sizing-up of transistors for increased drive strength. When these electrical mechanisms are extensively employed, as they would be in a large electrical broadcast network, it leads to high power consumption and poor scaling.

3.2. Cache Subsystem

The cores in ATAC each contain a simple processor with L1 data and instruction caches. While the detailed core architecture is outside the scope of this paper, the cache coherence system is an important aspect of ATAC that is now briefly described. This protocol, known as *ACKwise* is described in more detail in Section 4.

The data caches across all cores on the ATAC chip are kept coherent using a directory-based MOESI coherence pro-

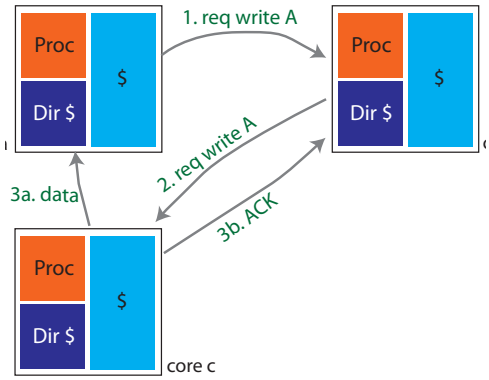

Figure 5. Cache coherence example.

tocol [15]. The directory is distributed evenly across the cores. Furthermore, each core is the "home" for a set of addresses (the allocation policy of addresses to homes is statically defined). Figure 5 shows a high level view of three cores, each located in different clusters across an ATAC chip. Each core contains a processor, data cache, and directory cache. For the purposes of this example, assume that the directory cache always hits. The arrows on the diagram depict a typical cache coherence operation: a write miss of address A by core a. Note that in this example, core b is the "home" of address A, and the initial cache state of address A is "owned" (O state) by core c. All coherence traffic is sent over the ANet. The sequence of actions needed to preserve coherence is as follows:

0. the processor in core a tries to write to address A and misses
1. core a sends a write request to core b, the home of address A
2. core b doesn't have the data in its cache, but core c is listed as the "owner" of it so core b sends a request to write address A on behalf of core a

3a. core c forwards the data for address A to core a and updates its cache state for address A to "invalid"

3b. core c sends an ACK back to the directory on core b

4. the cache line containing address A is updated on core a and the cache state is set to "modified"
5. the processor on core a writes a value to address A

3.3. External Memory Subsystem

When cores need to communicate with external memory, they do so via several on-chip memory controllers. Each memory controller replaces a cluster of cores and therefore has its own dedicated optical Hub. After receiving requests on the optical network, the memory controller communicates with external DRAM modules through standard I/O pins. Replies are then sent back to the processing cores through the optical network. By varying the number of clusters replaced by memory controllers, different ATAC chips with different ratios of compute power to memory bandwidth can be produced.

The primary task of the memory controller is to translate requests from the processing cores into transactions on a memory I/O bus. The choice of I/O bus technology is independent of the on-chip network architecture since the memory controller is performing a translation. In the near term, it is possible that ATAC processors would use standard electrical I/O to interface to off-the-shelf DRAM modules (e.g., DDR3 DIMMs). However, once the costs of integrating optical components into the chip and system have been paid, it seems logical to leverage those technologies for external I/O as well. Optical I/O has several potential advantages over electrical I/O including: lower power consumption, increased bandwidth, decreased pin-count, and great noise immunity.

A detailed design for an optical memory subsystem is beyond the scope of this paper and is left to future work. However, we can assume that an optical memory bus would consist of some number of on-chip waveguides that are coupled to external fibers or waveguides. Existing technology would require fibers to be mechanically bonded to the periphery of the chip. However, on-going research suggests that it should be possible to design free-space couplers that could transfer a signal from an on-chip waveguide to a waveguide embedded in a printed circuit board without a permanent mechanical connection. This would essentially create optical "pins" on a chip that could replace electrical pins.

Note that such optical pins would be able to transmit far more data than electrical pins. Each optical pin could carry up to 64 wavelengths of light at speeds of up to 20 GHz. The actual transmission speed would likely be limited by design trade-offs in the electrical circuits driving the optical components. We estimate that optical I/O pins operating at 5 GHz should be practical. Thus a single optical I/O pin will have a bandwidth of at least 320 Gb/s (or 40 GB/s). Contrast this with a 64-bit DDR3 memory bus which has a total peak data transfer rate of 12.8 GB/s. Thus an optical memory bus consisting of a single fiber or waveguide can reduce pin count by a factor of 64 while simultaneously increasing bandwidth by a factor of 3. These characteristics will make optical I/O highly desirable for large ATAC chips that require several memory controllers to meet their DRAM bandwidth needs.

4. Cache Coherence Protocol

This section presents *ACKwise*, a cache coherence protocol derived from a MOESI-directory based protocol [15]. Each directory entry in this protocol, as shown in figure 6 is similar to one used in a limited directory scheme [4] but with a few modifications. The 4 fields in each directory entry are as follows: (1) **State**: This field specifies the state of the cached block(s) associated with this directory entry (one of the MOESI states);

Figure 6. Structure of a Directory Entry

(2) **Global(G)**: This field states whether the number of sharers for this data block exceeds the capacity of the sharer list. If so, a broadcast is needed to invalidate all the cached blocks corresponding to this address when a cache demands exclusive ownership of this data block; (3) **KeeperID**: This field holds the ID of a core which contains an up-to-date copy of this data block. This copy could be clean (as in Exclusive(E) and Shared(S) state) or could be dirty (as in Modified(M) and Owned(O) state); and (4) **Sharers**$_{1-5}$: This field represents the sharer list. It can hold upto 5 core IDs.

When the number of sharers exceeds 6 (including the keeper), the **Global(G)** bit is set so that any number of sharers beyond this point can be accommodated. Once the global (G) bit is set, the sharer list (*Sharers*$_{1-5}$) holds the total number of sharers of this data block. In other words, once the global(G) bit is set, the directory has only the following information about a data block: (a) KeeperID; and (b) Number of sharers.

4.1. Operation of the ACKwise Protocol

When a request for a shared copy of a data block is issued, the directory controller first checks the state of the data block in the directory cache. (a) If the state is *Invalid(I)*, it forwards the request to the memory controller. The memory controller fetches the data block from memory and sends it directly to the requester. It also sends an acknowledgement to the directory. The directory changes the state of the data block to *Exclusive(E)* and sets the *KeeperID* field to the ID of the requester. (b) If the state is one of the valid states (i.e.,one of *MOES*), it forwards the request to the *Keeper*. The *Keeper* forwards the data block directly to the requester and sends an acknowledgement to the directory. Appropriate state changes happen in the cache block of the *Keeper* and the directory according to the rules of the *MOESI* protocol [15]. The directory controller also tries to add the ID of the requester to the sharer list. This is straightforward if the *global(G)* bit is clear and the sharer list has vacant spots. If *global(G)* bit is clear but the sharer list is full, it sets the *global(G)* bit and stores the total number of sharers (in this case, 7 (= 6+1)) in the sharer list. If the *global(G)* bit is already set, then it increments the number of sharers by one.

When a request for an exclusive copy of a data block is issued, the directory controller first checks the state of the data block in the directory cache. (a) If the state is *Invalid(I)*, the sequence of actions followed is the same as that above except that the state of the data block in the directory is set to *Modified(M)* instead of *Exclusive(E)*. (b) If the state is one of the valid states (i.e.,one of *MOES*), then the directory controller performs the following 2 actions: (i) It forwards the request to the *Keeper*. (ii) If the global bit is clear, it multicasts an invalidation message to the cores in the sharer list. Else, if the global bit is set, it broadcasts an invalidation message (to all the cores on the chip). Now, the *Keeper*, on receiving the forwarded request sends the data block directly to the requester, acknowledges the directory and invalidates its cache block. The other sharers invalidate their cache blocks and acknowledge the directory. The directory controller expects as many acknowledgements as the number of sharers (encoded in the sharer list if the *global(G)* bit is set and calculated directly if the *global(G)* bit is clear). After all the acknowledgements are received, the directory controller sets the state of the data block to *Modified(M)*, the *global(G)* bit to 0 and the *KeeperID* field to the ID of the requester.

Due to the broadcast capabilities of ATAC as described in section 3, the sending of multicast and broadcast messages can be achieved easily. A multicast invalidation message (referred to in the previous paragraph) is synthesized by prepending the sharer list to the invalidation address. This message is then sent on the electrical mesh network (ENet) followed by the optical ring (ONet). The receiving Hubs then filter out the invalidation messages directed to cores within them and forward it through one of the broadcast networks (BNet$_0$ or BNet$_1$). A broadcast invalidation message, on the other hand, is received by all the Hubs and broadcasted to all the cores within each Hub.

This protocol is named *ACKwise* because it keeps track of the number of sharers after the capacity of the sharer list has been exceeded and expects acknowledgements only from those many sharers in response to a broadcast invalidation message. The broadcast capabilities of *ATAC* coupled with this simple sharer tracking mechanism enable the *ACKwise* protocol to scale to 1000 cores.

A direct consequence of using this protocol is that silent evictions must be avoided. A detailed diagram showing the operation of this protocol and a table showing the set of coherence messages involved are presented in Appendix A.

5. Evaluation

To demonstrate the capabilities of the ATAC network (*ANet*) over a pure electrical mesh network denoted by *pEMesh*, we evaluate the performance of a cache coherent shared memory synthetic benchmark. The on-chip communication network's workload consists of the cache coherence messages that arise while running this synthetic benchmark. Results show that *ANet* is superior to *pEMesh* due to its higher bandwidth, lower latency, and broadcast capabilities.

5.1. Methodology

The methodology for the evaluation is described in great detail in Appendix B, but the salient aspects of the analytical model are presented here.

System Parameters	Value
CPI of Non-Memory Instructions	0.6
Number of Cores	1024
Number of Clusters	64
Off-Chip Memory Bandwidth	280 GB/s
Frequency of a Core	1 GHz
Cache Access Time	1 ns
Cache Line Size	64 bytes
Memory Access Time	0.1 μs
Single Hop Latency through Electrical Mesh	1 ns
Propagation Time though Optical Waveguide	2.5 ns
Number of Lanes (in an optical channel)	2
Number of Electrical Broadcast ($BNet_i$) Networks	2
Link Width of the pure Electrical Mesh (*pEMesh*)	2 flits

Table 1. Baseline System Configuration

Due to the impracticality of simulating many-core systems such as ATAC with current simulators, we built an analytical model of processor performance. The model is based on an in-order processor model focusing on latency of memory requests. It takes into account queueing delay in the on-chip network as well as off-chip. All network traffic generated by cache coherence messages is modeled and contributes to queueing delay.

The model distinguishes between different on-chip networks based on the expected transit time for a single flit in the network and the cost of generating different types of messages in the cache coherence protocol.

Comparing *ANet* to the pure electrical mesh (*pEMesh*), *ANet* differs because the average transit time is much less than an electrical mesh. In *ANet*, a flit goes through the optical waveguide and a short distance at the sender and receiver within a cluster. This time is, on average, much less than the time required in an electrical mesh, in which the message must be forwarded across the breadth of the chip.

Furthermore, *ANet* is much more efficient at multicast and broadcast traffic generated by the cache coherence protocol. In *ANet*, these messages consume a single packet along the optical waveguide (as described in section 4), and therefore their bandwidth is quite small. In a pure electrical mesh, a broadcast is forwarded to every core in the mesh, consuming enormous power and causing congestion.

5.2. Results

This sections details the different experiments conducted to measure the performance of *ANet* against that of *pEMesh*. The system parameters used are shown in Table 1 and the synthetic benchmark characteristics are shown in Table 2. The characteristics of the synthetic benchmark have been derived using the PARSEC benchmark suite [5]. Unless otherwise mentioned, these parameters are used in the performance studies conducted in this section.

Recall that the ATAC Network, *ANet* is made up of an electrical mesh network (*ENet*), an optical network (*ONet*) and two electrical broadcast networks ($BNet_0$ & $BNet_1$). The link widths of *ENet*, $BNet_0$ and $BNet_1$ are 1 flit (= 32 bits) each. Each optical channel consists of 2 lanes. Each lane comprises of a particular wavelength from a total of 32 waveguides (1 wavelength each from 1 waveguide) and hence is 1 flit wide. The *ONet* consists of 64 such optical channels (one for each cluster). The link width of the pure electrical mesh network, *pEMesh* with which we compare the performance of *ANet* is 64 bits (= 2 flits).

Parameter	Value
Frequency of Data References	0.3
Fraction of Reads in Data References	$2/3$
Fraction of Writes in Data References	$1/3$
Cache Miss Rate	4%
Average Number of Sharers	4
Fraction of Memory Requests going off-chip	0.7
Fraction of Memory Write Requests that cause Invalidation Broadcasts	0.1

Table 2. Benchmark Characteristics

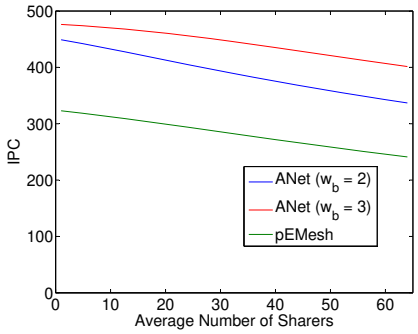

Figure 7. Performance of *ANet* vs *pEMesh* as a function of Average Number of Sharers

First, we describe how the number of electrical broadcast networks for *ANet* is calculated. The ratio of the number of electrical broadcast networks to the number of lanes in an optical channel is given by $f_u + f_m n_m + f_b C$ where f_u, f_m and f_b are the fraction of unicast, multicast and broadcast messages respectively, n_m is the average number of receivers (in terms of number of clusters) of a multicast message and $C(= 64)$ is the total number of clusters. This ratio is calculated to be ~ 1.15 in our experiments taking only cache coherence messages into account. In figure 8(a), we vary the number of broadcast networks (w_b) from 1 to 5. We observe that the performance of *ANet* almost reaches its peak when there are 3 broadcast networks. Note that $3 = \lceil 1.15 \cdot 2 \rceil$. Since the performance improvement when the number of broadcast networks is increased from 2 to 3 is relatively small, a reasonable choice would be to opt for 2 broadcast networks (corresponding to 2 lanes in an optical channel) to save on cost and area.

From figure 8(a), we also observe that with a single electrical broadcast network, *pEMesh* outperforms *ANet*. This can

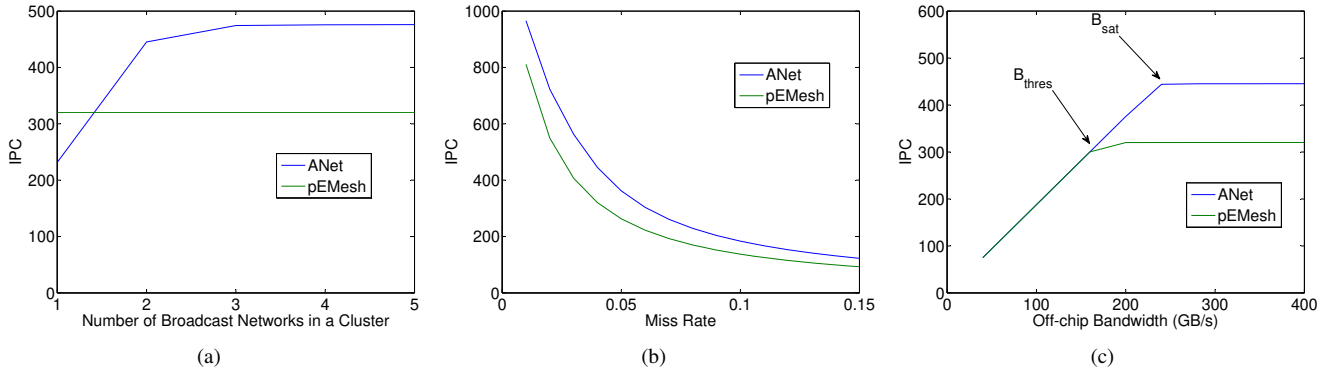

Figure 8. Performance of *ANet* vs *pEMesh* as a function of (a) Number of Broadcast Networks; (b) Miss Rate; and (c) Off-Chip Bandwidth

be attributed to the high queueing delays at the receiving Hub due to the extremely low broadcast bandwidth of the receiving cluster. In all other cases, *ANet* outperforms *pEMesh*.

Next, we study the effect of miss rates on the performance of *ANet* and *pEMesh*. The miss rate is varied from 1% to 15%. From figure 8(b), we observe that the performance of *ANet* is on an average 33.6% above that of *pEMesh*. Hence, *ANet* can outperform *pEMesh* over a wide range of applications with varying miss rates.

The effect of off-chip memory bandwidth on the performance of *ANet* and *pEMesh* is illustrated in figure 8(c). In this graph, we vary the off-chip bandwidth from 40 GB/s to 400 GB/s in steps of 40 GB/s. Observe that the performance of the chip is solely dominated by the memory bandwidth at low bandwidths, and it becomes sensitive to the on-chip network only when the memory bandwidth reaches a certain threshold (B_{thres} as shown in figure 8(c)). Once this threshold is exceeded, we observe that the performance of *ANet* is above that of *pEMesh* by 39%. We also observe that the performance of both *ANet* and *pEMesh* saturates after the memory bandwidth reaches another threshold (B_{sat}) since the queueing delays when going off-chip are extremely small beyond this point. Stated in a different way, beyond B_{sat}, the off-chip bandwidth provided exceeds what the application demands.

The effect of the average sharing density of a data block on the performance of *ANet* and *pEMesh* is shown in figure 7. In this performance study, we vary the average number of sharers from 1 to 64. As the number of sharers of a data block increases, the performance of *pEMesh* degrades due to the increase in number of broadcasts needed to invalidate cached copies of a data block (due to exclusive requests for that data block). On the other hand, in *ANet*, broadcasts are cheap on the optical network but lead to a significant increase in queueing delay at the receiving Hub due to the small number of broadcast networks ($w_b = 2$). The number of broadcast networks is decided based on evaluations where the average number of sharers is 4 (common case). Due to the above reasons, the performance improvement of *ANet* over *pEMesh* increases very slowly as the sharing density increases. To illustrate the effect of the number of broadcast networks on performance when the sharing density becomes large, we plot the performance of ANet when the number of broadcast networks is 3 ($i.e.,\ w_b = 3$) also in figure 7. We observe that the performance of *ANet* when $w_b = 3$ exceeds that when $w_b = 2$ by a large margin when the average number of sharers is high. When $w_b = 3$, the performance improvement of *ANet* over *pEMesh* across the studied sharing density range is 57.5%.

5.3. Discussion

In our experimental studies, we find that the ATAC network(*ANet*) outperforms a pure electrical mesh network (*pEMesh*) due to its higher bandwidth, lower latency and broadcast capabilities. To demonstrate this fact, we measure the contribution to the average memory latency due to the following 3 factors: (1) On-Chip Base Latency; (2) On-Chip Queueing Delay; and (3) Off-Chip Latency.

The on-chip base latency is the average on-chip latency of a memory request assuming infinite bandwidth. The on-chip queueing delay is the contribution to the average memory latency due to the queueing delays at the sending and receiving Hubs in the ATAC network (*ANet*) or due to the contention delay in the pure electrical mesh (*pEMesh*). The off-chip latency is the contribution due to the DRAM access time and the queueing delays while going off-chip. This factor is almost the same for both *ANet* and *pEMesh* since we assume the same off-chip bandwidth for both of them.

From our experimental study using the system configuration shown in Table 1 and the benchmark characteristics shown in Table 2, we observe that the average memory access time of the synthetic benchmark is 6.26 on *ANet* and 9.26 on *pEMesh*. Out of 6.26 on *ANet*, the contribution of the on-chip base latency is 2.71 (43.3%) and that of the on-chip queueing delay is 0.78 (12.5%). Out of 9.26 on *pEMesh*, the contribution of the

on-chip base latency is 5.12 (55.3%) and that of the on-chip queueing delay is 1.37 (14.8%). The contribution due to the off-chip latency is 2.77 in both the cases. (All numbers in this paragraph are reported in terms of processor cycles).

From the above figures, we conclude that *ANet* outperforms *pEMesh* in terms of both on-chip bandwidth and base latency. The on-chip base latency is 47.1% lesser in *ANet* as compared to *pEMesh* while the on-chip queueing delay is 43.1% lesser in *ANet* as compared to *pEMesh*. *ANet* also outperforms *EMesh* in its broadcast capability but could do so more significantly if the number of broadcast networks is increased in proportion to the amount of broadcast traffic (as illustrated in figure 7).

6. Related Work

CMOS-compatible nanophotonic devices are an emerging technology. Therefore there have only been a few architectures proposed that use them for on-chip communication: Corona [8], the optical cache-coherence bus of Kirman et al [14], and the switched optical NoC of Shacham et al [3].

The Corona architecture [8] uses optics for two different on-chip networks: an optical crossbar and an optical broadcast bus. Corona is similar to ATAC in that it uses a snaking waveguide pattern to connect 64 clusters of cores. However, the two designs differ in the way that they assign communication channels. Corona assigns a physical channel to each receiver and uses WDM to send multiple bits of a dataword simultaneously. ATAC assigns a physical channel to each sender and uses WDM to carry multiple channels in each waveguide. The Corona design requires arbitration between senders wishing the communicate with the same receiver. The ATAC design uses more optical components but allows all communication to occur independently and without any form of contention. This creates a uniform communication latency and allows the receiver to choose the order in which it processes messages.

Kirman et al [14] discuss the integration of optical technology into the cache-coherence bus of a future CMP. The design of their network is very similar to ATAC, consisting of a top-level optical broadcast bus which feeds into small electrical networks connecting groups of cores. However, the ATAC network is designed to be a general communication mechanism whereas Kirman's network is limited to snooping cache coherence traffic. We believe that their snooping architecture is unlikely to scale to hundreds or thousands of cores. Our *ACKwise* coherence protocol is designed to make efficient use of the optical broadcast network and scale to thousands of cores.

Shacham et al [3] propose a novel hybrid architecture in which they combine a photonic mesh network with electronic control packets. Whereas ATAC uses photonics to create an efficient broadcast network, Shacham et al construct a switched mesh network using similar components. To overcome the current impossibilities of directly buffering or processing optical information, they use an electronic control network to setup photonic switches in advance of the optical signal transmission. This scheme is therefore still partially limited by the properties of electrical signal propagation. It only becomes efficient when a very large optical payload follows the electrical packet. ATAC, on the other hand, leverages the efficiencies of optical transmission for even a single word packet.

7. Conclusion

The recent advances of optical technology have certainly inspired confidence in computer architects that optics may very well continue to make its way into smaller and smaller packages; just as optical interconnect has moved from connecting cities to connecting data centers, it seems likely that it will soon connect chips and on-chip components.

Overall, this paper presented a novel manycore architecture that scales to 1000 cores by embracing new technology offered by recent advances in nanophotonics. This paper also introduced ACKwise, a novel directory-based cache coherence protocol that takes advantage of special properties of ATAC to achieve high performance. Early analytical model based studies of performance show that a 1000-core ATAC chip achieves a speedup of as much as 39% when compared with a similarly sized manycore with an electrical mesh network. Our evaluations also provide significant insight into how the different parameters of an opto-electric network (like ANet) should be tuned to perform well together.

References

[1] ATAC: All-to-All Computing Using On-Chip Optical Interconnects. In *BARC*, 1/2007.

[2] The International Technology Roadmap for Semiconductors (ITRS) Technology Working Groups, 2008. http://public.itrs.net.

[3] A. Shacham, B.G.Lee, A.Biberman, K.Bergman, and L.P.Carloni. Photonic NoC for DMA Communications in Chip Multiprocessors. In *IEEE Hot Interconnects*, August 2007.

[4] Anant Agarwal et al. An evaluation of directory schemes for cache coherence. In *ISCA*, 1988.

[5] Christian Bienia et al. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *PACT*, 2008.

[6] I. Corporation. Intel's Teraflops Research Chip. http://techresearch.intel.com/articles/Tera-Scale/1449.htm.

[7] D. A. et al. High performance, waveguide integrated Ge photodetectors, 2007.

[8] D. V. et al. Corona: System Implications of Emerging Nanophotonic Technology. In *ISCA*, 2008.

[9] J. F. L. et al. Waveguide-integrated, ultra-low energy GeSi electro-absorption modulators, 2008.

[10] J. M. et al. Advances in Fully CMOS Integrated Photonic Circuits, 2007.

[11] M. B. et al. Process flow innovations for photonic device integration in CMOS, 2008.

[12] R. Kirchain and L. Kimerling. A roadmap for nanophotonics, 2007.

[13] J. F. Liu and J. Michel. High Performance Ge Devices for Electronic-Photonic Integrated Circuits, 2008.
[14] N.Kirman et al. Leveraging Optical Technology in Future Bus-based Chip Multiprocessors. In *MICRO*, 2006.
[15] Paul Sweazey and Alan Jay Smith. A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus. In *ISCA*, 1986.
[16] C. Schow. Optical Interconnects in Next-Generation High-Performance Computers. OIDA 2008 Integration Forum, 2008.
[17] Taylor et al. Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams. In *ISCA*, 2004.

APPENDIX

A. Cache Coherence Protocol

The coherence messages involved in the protocol described in section 4 are shown in table 3. Numbers assigned to each type of coherence message are used in the next section for building an analytical model to study the performance of the ATAC network compared with a normal electrical mesh.

Figure 9 shows the dynamics of the cache coherence protocol. This figure can be understood from the description of the protocol given in section 4. In this figure, **R** stands for Requester, **H** stands for Home (core on which the directory is present), **K** stands for Keeper (as explained in section 4) and **M** stands for Memory controller.

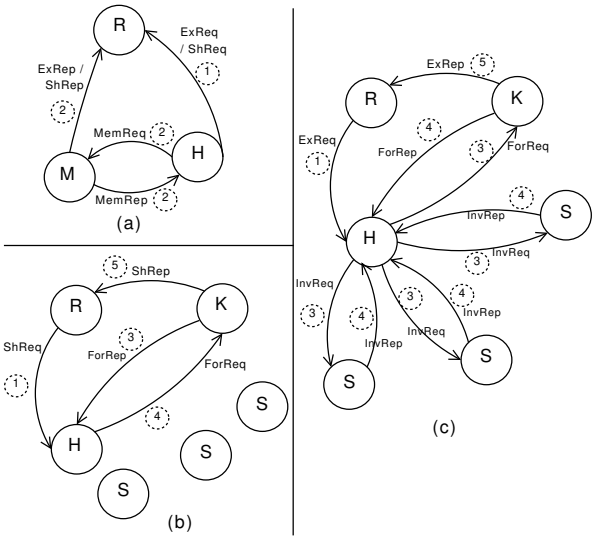

Figure 9. Operation of the ACKwise cache coherence protocol

B. Analytical Performance Model

We developed an analytical model of processor performance for the given cache coherence protocol. It is built to model two different on chip networks: an electrical mesh network and the ATAC optical network. The model focuses on the latency and queueing delay of memory requests in the on-chip networks. Queueing delay to main memory is also modeled. Parameters to the model are listed in Table 4. All times are measured in cycles, except where otherwise stated. Message lengths are measured in flits.

Parameter	Description
$\text{CPI}_{\text{non-mem}}$	The CPI of non-memory instructions.
N	The number of cores.
n	The number of cores per cluster.
M	The number of memory controllers.
B	The off-chip bandwidth (GB/s).
f_{core}	Frequency of a core.
t_{cache}	Average access time in the cache.
t_{mem}	Memory access time.
t_{mesh}	Hop time through the electrical mesh.
t_{opt}	Time through optical waveguide (including transitions between electrical and optical).
f_{mem}	Frequency of data references.
f_r	Frequency of data references that are reads.
f_w	Frequency of data references that are writes. ($f_r + f_w = 1$)
m_r	Miss rate of reads.
m_w	Miss rate of writes.
p_0	Probability that a miss encounters no sharers in the directory. (And therefore must go off-chip.)
k	The maximum number of sharers allowed before using broadcast.
p_k	Probability that a miss encounters between 1 and k sharers in the directory.
E_k	Expected number of sharers, given that there is at least one sharer.
$E_{C,k}$	Expected number of unique clusters containing sharers, given that there is at least one sharer. (Any cluster could contain up to n sharers, so $E_{C,k} \leq E_k$.)
ℓ_A	Length of an address or acknowledgement packet.
ℓ_D	Length of a data packet.
ℓ_M	Length of a multicast packet.
w_{opt}	Number of lanes in an optical channel.
w_{broad}	Number of electrical broadcast networks.
w_{elec}	Link Width of electrical mesh network (in flits).

Table 4. Parameters to the analytical model.

CC Messages	Msg Number	Description
ExReq	1	Request for an exclusive copy of a cache block
ExRep	2/5	An exclusive copy of a cache block returned to its requester (Memory Controller - 2, Keeper - 5)
ShReq	1	Request for a shared copy of a cache block
ShRep	2/5	A shared copy of a cache block returned to the requester (Memory Controller - 2, Keeper - 5)
InvReq	3	Invalidation Request from Directory to a sharer
InvRep	4	Acknowledgement for InvReq
ForReq	3	Request (ShReq/ExReq) forwarded from the Directory to the Keeper
ForRep	4	Acknowledgement for ForReq
MemReq	2	Request forwarded from the Directory to the Memory Controller
MemRep	2	Acknowledgement from the Memory Controller to the Directory

Table 3. Coherence Messages in a MOESI Directory CC protocol

B.1. CPI

The performance of a processor is computed as a function of the CPI of each core, which is calculated using a simple in-order processor model. Because of the different actions taken in the cache coherency protocol, reads and writes are modeled separately. The basic equation for CPI is,

$$\text{CPI} = \text{CPI}_{\text{non-mem}} + f_{\text{mem}}(t_{\text{cache}} + f_r m_r t_{r\text{-miss}} + f_w m_w t_{w\text{-miss}}) \quad (1)$$

Where $t_{r\text{-miss}}$ and $t_{w\text{-miss}}$ represent the latency of read misses and write misses, respectively.

We now show in detail how the miss latency is derived. The latency of memory references is determined by the three-hop latency in steps 1, 3, and 5 in the protocol (or, equivalently, steps 1 and two hops in 2 if the cache line is not shared) (refer table 3 above). The latency of a message of length ℓ is given by $t_{\text{flit}} + (\ell - 1)$, where t_{flit} is the latency of a single flit through the network, assuming uniformly distributed senders and receivers. t_{flit} depends on the type of network being modeled, and will be derived for ATAC and mesh networks later. Referring to the protocol above, we have

$$\begin{aligned} t_{r\text{-miss}} =& p_0(\text{Latency off-chip}) + (1-p_0)(\text{Latency on-chip}) \\ t_{r\text{-miss}} =& p_0\left[3t_{\text{flit}} + (t_{\text{mem}} + Q_{\text{mem}}) + 2(\ell_A - 1) + (\ell_D - 1)\right] \\ &+ (1-p_0)\left[3t_{\text{flit}} + 2(\ell_A - 1) + (\ell_D - 1)\right] \end{aligned} \quad (2)$$

Q_{mem} gives the queueing delay off-chip. The expression for $t_{w\text{-miss}}$ is very similar, except in that case we must also account for the slightly longer multicast packets that can occur when there are less than k sharers.

B.2. Queueing Delay

There are two queueing delays of interest: the queueing delay off-chip and the queueing delay in the on-chip network. In either case, delay is modeled by an M/D/1 queueing model with infinite queues. In this model, queueing delay is given by

$$Q = \frac{\lambda}{2\mu(\lambda - \mu)} \quad (3)$$

With λ the arrival rate and μ the service rate.

The off-chip queuing delay is slightly simpler, so that is derived first. To get the delay in terms of cycles, we must express the off-chip bandwidth in flits per cycle.

$$\mu_{\text{mem}} = \frac{B}{M f_{\text{core}} w} \quad (4)$$

Where w is the width of the network. The arrival rate is the miss rate to memory per cycle. Miss rates are given per instruction, so the arrival rate is inversely proportional to CPI. The per-core arrival rate is given by

$$\lambda_{\text{mem,core}} = \frac{1}{\text{CPI}} p_0 f_{\text{mem}}(f_r m_r + f_w m_w)$$

This gives an overall arrival rate off chip of[1]

$$\lambda_{\text{mem}} = \frac{N \lambda_{\text{mem,core}} \ell_D}{M} \quad (5)$$

Q_{mem} is given by substituting equations (4) and (5) into equation (3).

On-chip queueing delay is modeled under the same framework, but the arrival and service rates depend on the network being modeled. This is covered in the next section.

B.3. On-Chip Networks

This section derives t_{flit}, the mean transit time of a single flit through the network for uniformly distributed senders and receivers.

[1] It is assumed that addresses use separate control lines off-chip and do not contend with data traffic. Therefore queueing delay is dominated by the much larger data packets.

B.3.1. ATAC

Define t_{ATAC} as t_{flit} for the ATAC architecture. From the architecture above,

$$t_{\text{ATAC}} = (d_{\text{send}} + d_{\text{recv}})t_{\text{mesh}} + t_{\text{opt}} + Q_{\text{send}} + Q_{\text{recv}} \quad (6)$$

Here, d_{send} and d_{recv} are the mean distances traveled in the electrical mesh within a cluster at the sender and receiver. Specifically, $d_{\text{send}} = \frac{\sqrt{n}}{2}$ and $d_{\text{recv}} = \log_2 n$. We assume that contention on the electrical mesh within a cluster is insignificant, as it is limited by the bandwidth onto the optical waveguide and will never have high utilization.

Q_{send} and Q_{recv} are the queueing delay at modulator and detector, respectively. We derive Q_{send} using equation (3). μ_{send} is simply the width of the optical network, $\mu_{\text{send}} = w_{\text{opt}}$. The arrival rate is more complicated. It can be split by defining c_r and c_w as the total network cost (i.e., number of flits generated) for a read and write, respectively. This yields,

$$\lambda_{\text{send}} = \frac{f_{\text{mem}}}{\text{CPI}} \cdot (f_r m_r c_r + f_w m_w c_w) \quad (7)$$

c_r and c_w are computed by referring to the cache coherence protocol. c_r is simpler than c_w and nearly identical for both ATAC and electrical meshes, so we only give the derivation of c_w.

$$\begin{aligned} c_w = \;& \ell_A && \text{Initial request} \\ & + p_0(\ell_D + 2\ell_A) && \text{Off-chip traffic} \\ & + p_k\ell_M + (1 - p_0 - p_k)\ell_A && \text{Invalidates} \\ & + (1 - p_0)E_k\ell_A && \text{Replies} \\ & + (1 - p_0)\ell_D && \text{Data to requestor} \end{aligned}$$

Q_{recv} is derived similarly. The service rate is $\mu_{\text{recv}} = w_{\text{broad}}$. The only additional complication is that a single message on the optical waveguide can have several clusters as its destination. Therefore, the arrival rate at the receiver exceeds that of the senders by a factor of the mean number of destinations for a message. This is computed very similarly to λ_{send} — we condition on the type of message based on the arrival rates for reads and writes, and multicast and broadcast packets from step 3 of the protocol are more heavily weighted.

B.3.2. Electrical Mesh Networks

Define t_{elec} as t_{flit} for an electrical mesh network. By the model given in REFERENCE ANANT'S PAPER HERE!,

$$t_{\text{elec}} = d_{\text{p2p}}(t_{\text{hop}} + Q_{\text{elec}}) \quad (8)$$

Where d_{p2p} is the mean number of hops between sender and receiver, or $d_{\text{p2p}} = \sqrt{N}$. Q_{elec} is the queueing delay at each hop. By SAME REFERENCE HERE!,

$$Q_{\text{elec}} = \frac{3\rho}{1 - \rho} \frac{d_{\text{p2p}} - 2}{d_{\text{p2p}}} \quad (9)$$

ρ is the utilization of the mesh, which is given by $\frac{\lambda_{\text{elec}}}{\mu_{\text{elec}}}$. As before, $\mu_{\text{elec}} = w_{\text{elec}}$. λ_{elec} is derived exactly as λ_{send}, except the costs of read and write misses are different. For reads, c_r simply increases by a factor of d_{p2p}. Writes are complicated by broadcasts in an electrical network, which are forwarded $d_{\text{broad}} = N - 1$ times,[2] and the lack of a true multicast mechanism. This gives,

$$\begin{aligned} c_w = \;& d_{\text{p2p}}\ell_A \\ & + d_{\text{p2p}}p_0(\ell_D + 2\ell_A) \\ & + d_{\text{p2p}}p_k E_k \ell_A + d_{\text{broad}}(1 - p_0 - p_k)\ell_A \\ & + d_{\text{p2p}}(1 - p_0)E_k\ell_A \\ & + d_{\text{p2p}}(1 - p_0)\ell_D \end{aligned}$$

B.4. Solving for CPI

A complication with this model is that the CPI as given by equation (1) is dependent on several queueing delays that themselves depend on the CPI. Finding a consistent solution algebraically seems hopeless due to the nonlinearity of the queueing models and the large number of parameters.

Numerical methods address this, however. Unfortunately, iteration to a fixed point is ineffective because of the extreme instability of the CPI. But an equally simple solution works since the right side of equation (1), when viewed as a function of CPI, is monotonically decreasing. Therefore one can use binary search to find the fixed point. We used this method to find solutions within 0.001 accuracy.

B.5. Sources of Error

There are a few known sources of error in the model:

- Instruction cache misses are not modeled. It is assumed that since the instruction miss rate is low and will never generate invalidates or dirty cache blocks, the additional traffic in the network is not a first order effect.
- Queueing delay at the directory itself is not modeled. With high utilization, there could be multiple outstanding requests that force delay in the processing of later requests.
- Evictions are not modeled. Evictions would not increase the latency of misses, but would increase the utilization of the network and consequently the queueing delay.
- Congestion control in the networks is not modeled. We assume infinite queues in our queueing model, but in reality under high loads the networks will engage in some form of congestion control that limits performance.

[2] The simplest way to see this is that the message must be sent to every core except the sender, but a more rigorous counting argument produces the same result.