ATAC: On-Chip Optical Networks for Multicore Processors

James Psota Jonathan Eastep Jason Miller Theodoros Konstantakopoulos Michael Watts
Mark Beals Jurgen Michel Kim Kimerling Anant Agarwal
Massachusetts Institute of Technology
Cambridge, MA 02139
{psota,eastep,jasonm,tkonsta,mbeals,jmichel,lckim,agarwal}@mit.edu, mwatts@sandia.gov

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: AMD and Intel have released 2-core chips with 4-core offerings coming soon. Forecasts show that by 2014, processors may contain thousands of cores. But will current processor architectures (especially their interconnection mechanisms) scale to thousands of cores and will programming such systems be tractable? This abstract argues that current mulitcore architectures will not scale to thousands of cores and introduces ATAC, a new processor architecture that attempts to solve these issues. ATAC integrates an on-chip optical broadcast communication network to drastically improve the performance and energy scalability and ease of programmability of multicore processor architectures for parallel applications.

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 computational elements on a single chip, the extent to which they can be used to improve performance is limited by the cost of communication among them. As computation is spread across the elements, communication of intermediate values accounts 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 broadcast operations because each broadcast ties up many resources. Besides performance, broadcasts 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. The Raw microprocessor [3] consists of a 2-D mesh of cores where each core can communicate directly 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 may 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 are charged with orchestrating both computation and communication, a challenge that has been evident as long as parallel computers have existed. Further, the techniques that are currently used to program multicores will not scale to thousands of cores. Neither bus-based nor point-to-point multiprocessors are likely to scale to thousands of cores since contention would overwhelm the network and coordinating thousands of processors is extremely difficult. Furthermore, broadcast and all-to-all operations on these systems are possible but expensive.

The ATAC processor architecture addresses these contention and programmability issues using on-chip optical communications technologies to replace or augment electrical communication channels. Current research in optical communications technologies is making 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 128 WDM channels would match the bandwidth of a 128-bit electrical bus. Optical waveguides, however, can also transmit data at higher speeds than electrical wires. This virtually eliminates the heterogeneous distance-dependent cost function for communication that complicates multicore programming. In addition to speed, optical signaling can use less power than electrical signaling, especially for long wires because optical waveguides have relatively low loss and do not require periodic repeaters as long electrical wires do. Using these new optical technologies, ATAC processor will provide programming transparency, power efficiency, high bandwidth (both on-chip and off-chip), and performance scalability. The remainder of this abstract presents a brief introduction to the ATAC architecture and programming model.

2. Architecture Overview

The ATAC architecture provides an ISA that offers an interface to both processing elements and communication mechanisms through suitable high-level abstractions. As illustrated in Figure 1, ATAC consists of a 2-D mesh of cores, each containing a processor pipeline and electrical and optical network resources. While the initial prototypes will not meet this target, ATAC is targeted for an 11 nm process in 2014, and will have about 4096 cores. Each core will have a simple MIPS-style processing pipeline. Indeed, ATAC favors simpler cores in order to pack more of them onto the chip and allow its designers to focus on the interconnection network.

ATAC cores are interconnected via an optical waveguide ring bus that is functionally similar to a fully-connected, bi-directional point-to-point network for message-passing. The waveguide bus passes through every core and incorporates WDM to seemlessly overlap simultaneous communications without contention. Additionally, it leverages the improved propagation speed of optical signals to eliminate the heterogeneous, distance-dependent cost of communication between cores; any pair of cores on the chip communicate with O(1) latency. To contrast, employing WDM is not

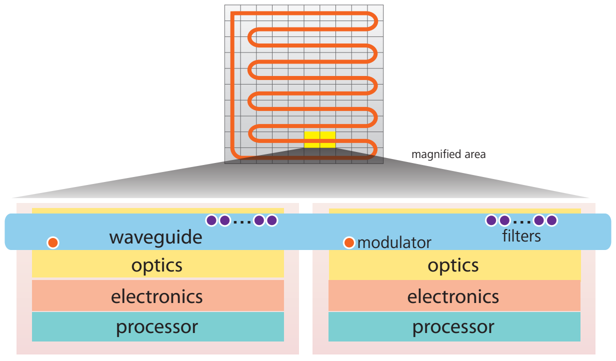

Figure 1. 10x10 core ATAC chip showing two adjacent cores

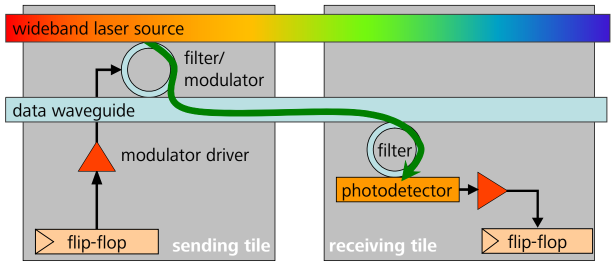

Figure 2. Optical transmission of one bit between two cores

feasible with traditional electrical interconnect, and wire delay and power considerations limit the scalability of electrical buses, forcing alternatives with distance-dependent communication latencies. ATAC's unique network architecture simplifies programming, as programmers need only specify the recipient of messages without having to deal with the complicated routing, non-uniform latencies, and bottlenecks inherent to most electrical interconnects.

The optical network also allows a core to broadcast values to some or all of the other cores. This broadcast mechanism, which will scale to an arbitrary number of cores, is one of the differentiators of the ATAC architecture from a performance scalability perspective, as electrical broadcast mechanisms of traditional multicore architectures do not scale. The network also allows simultaneous all-to-all communications, where all cores broadcast at once.

The process of sending one bit from one core to another is illustrated in Figure 2. ATAC's enabling technology is an add/drop optical network. A modulator is an optical component that writes 1s and 0s onto the network. Each core has one modulator per bit statically tuned to a frequency that is dedicated to that core. The modulator is turned on and off by a driver that is controlled by the processor electronics. The optical signal is transmitted over a Si waveguide at approximately one-third the speed of light. The signals are received using optical filter detectors, each statically tuned to receive a particular wavelength. For an N-core ATAC chip, each core contains N x M filters, where M is the bit-width of the network, enabling each core to receive from any other core. The filter channels the light onto a photodetector, which converts the incoming optical signal into an electrical signal that is buffered and stored in a flip-flop. The data is fed into a FIFO and ultimately into a CAM from which the receiving core's processor can read the value. The CAM facilitates the ATAC programming model, discussed in Section 3.

3. Programming the ATAC

Traditional cluster computing programming models squander the multicore opportunity. Message passing and shared memory paradigms, often employed with MPI [2] and OpenMP [1], were designed assuming high-overhead communication. Algorithms run on such systems require large caches and need large chunks of work to minimize communication. Multicore, on the other hand, should have low-overhead and high-bandwidth communication that is cheaper than memory accesses, which results in smaller per-core memories, and hence, less power. ATAC attempts to achieve this goal by offering programmers low-latency and high-bandwidth communication resources that encourage a programming model that is communication-centric rather than memory-centric.

ATAC employs a message-passing communication model, where a message can be sent to one, some, or all cores on the chip. Messages are buffered by the receivers in a CAM structure, and received by the processor pipeline using a CAM lookup. Buffered messages can be indexed by sender, by type (data, memory read, etc.), or using wildcards (*e.g.,* "receive from any sender"). ATAC programmers will have access to a high-level API that resembles MPI, and includes efficient broadcast and other global operations, as well as standard point-to-point communications. One of the goals of the ATAC architecture is to offer programmers a high-level API that is at least as straightforward as MPI, but with the performance of hand-coded and hand-optimized parallel programs.

One application where we can apply the ATAC API and exploit ATAC's unique broadcast capabilities is N-body particle-particle simulation, which has applications to astronomy, molecular dynamics, fluid dynamics and other areas. Various algorithms improve performance on traditional architectures, sacrificing accuracy for performance. The N-body problem exhibits $O(N^2)$ intrinsic parallelism because updating the state for a body requires the superposition of contributions from each of the other $N-1$ bodies. Each body broadcasts its state to the other $N-1$ bodies; each body then incorporates incoming states to update its state for the next simulation step. Traditional approximating algorithms avoid the $O(N)$ simultaneous broadcasts and long-distance point-to-point communication that becomes necessary as N grows large. ATAC, on the other hand, provides equidistant point-to-point communication between any two cores, allowing as many simultaneous, contention-free broadcasts as there are WDM channels in the optical network. On traditional architectures, the distance-based cost function of communication between physically distant cores limits the extent to which parallelization improves performance and adds considerable development effort to achieve high performance. ATAC allows the use of a straightforward algorithm and enables it to scale much further.

4. Conclusion

The ideas presented in this abstract will hopefully enable multicore processors to scale well into thousands of cores. The ATAC team plans to begin implementing a prototype version of ATAC in hardware over the next year.

Acknowledgments This work was sponsored by the Defense Advanced Research Projects Agency's UNIC program funded under ARL contract #W911NF-07-1-0034, NDSEG, and the NSF.

References

[1] Openmp: Simple, portable, scalable smp programming. http://www.openmp.org.

[2] M. Forum. A message passing interface standard. Technical report, University of Tennessee, Knoxville, 1994.

[3] Taylor et al. Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams. In *ISCA*, 2004.