Low Latency Scheduling Algorithm for Shared Memory Communications over Optical Networks

Muhammad Ridwan Madarbux, Anouk Van Laer and Philip M. Watts
Department of Electronic and Electrical Engineering
University College London
London, United Kingdom, WC1E 7JE
Email: m.madarbux@ucl.ac.uk

***Abstract*—Optical Network on Chips (NoCs) based on silicon photonics have been proposed to reduce latency and power consumption in future chip multi-core processors (CMP). However, high performance CMPs use a shared memory model which generates large numbers of short messages, typically of the order of 8-256B. Messages of this length create high overhead for optical switching systems due to arbitration and switching times. Current schemes only start the arbitration process when the message arrives at the input buffer of the network. In this paper, we propose a scheme which intelligently uses the information from the memory controllers to schedule optical paths. We identified predictable patterns of messages associated with memory operations for a 32 core x86 system using the MESI coherency protocol. We used the first message of each pattern to open the optical paths which will be used by all subsequent messages thereby eliminating arbitration time for the latter. Without considering the initial request message, this scheme can therefore reduce the time of flight of a data message in the network by 29% and that of a control message by 67%. We demonstrate the benefits of this scheduling algorithm for applications in the PARSEC benchmark suite with overall average reductions in overhead latency per message, of 31.8% for the streamcluster benchmark and 70.6% for the swaptions benchmark.**

I. Introduction

Photonic networks-on-chip (NoC) based on advances in silicon photonics have been proposed as one of a range of solutions to the serious problems of energy consumption and thermal management in chip multiprocessors (CMP) [1]–[6] due to fundamentally lower power consumption. Figure 1) shows a shared distributed memory CMP with optical cross bar assumed in this work. However, whereas electronic networks, such as meshes [7] and crossbars [8] rely on multiple hops between sequential elements, efficient optical NoC require end-to-end optical paths to be setup in advance of communication and the resulting latency overhead of arbitration and control message transmission between cores and a central switch can be significant for the short (8–256 B) messages produced by shared memory computer systems. Figure 2(a) shows the overhead latency comprising of request transmission, arbitration and grant transmission latencies. Various schemes have been proposed to overcome this overhead latency and the non-scalable nature of centralised arbitration including speculative transmission [1], [9], distributed arbitration [1], optical arbitration [2], avoiding arbitration using a single write multiple read topology [4] or partitioning the network into smaller optical switch sections separated by optical-electrical-optical converters and electronic buffering [4], [6]. These schemes all involve an increase in the number of optical components and/or the complexity of the control plane. Alternatively, optical circuit switching allows long lived flows to be efficiently transferred with low arbitration overhead [5], although a backup electronic network is usually necessary to transfer small messages. The authors have previously investigated setting up long lived circuits between shared memory cores which have dense sharing requirements. Initial results [10] showed that, with ideal circuit setup decisions made on less than 1 μs time periods, a large proportion of PARSEC application traffic can be routed onto the circuit switch. However, further investigation has shown that adding background traffic from the operating system considerably reduces the benefits.

In this paper, we propose and evaluate a low latency scheduling algorithm for an optical shared memory network which reduces the number of control path messages and arbitrations required, thus improving both latency and power consumption. Rather than waiting for each message to appear in the input buffer before starting the arbitration process, our algorithm intelligently uses a knowledge of the traffic patterns produced by the cache coherence protocol to setup optical paths and hence eliminate the arbitration latency for many messages. The rest of the paper is organized as follows: Section II describes the predictable communication patterns produced by cache coherence protocols and our scheduling algorithm which we use to efficiently schedule optical paths. Section III outlines our evaluation methodology including the baseline optical switch system and our proposed algorithm. Section IV presents results from simulations of the baseline and proposed networks operating on a 32 core x86 system running the PARSEC benchmark suite, and finally, section V discusses the system implications of these results and further work.

II. Proposed Scheduling Algorithm

In order to investigate the traffic patterns within shared memory CMPs, we collected traces of all communication within the parallel phase of ten PARSEC benchmark applications [11] running on a simulated 32 core x86 system with a Linux operating system and MESI coherence protocol using the gem5 simulator [12]. Ideal contention-free interconnect

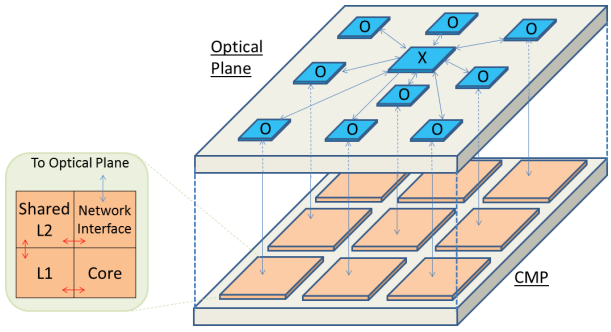

Fig. 1: A Chip Multiprocessor (CMP) with compute tiles consisting of processor, private L1 and distributed shared L2 cache interconnected with a photonic crossbar. O = optical network port, X = optical switch

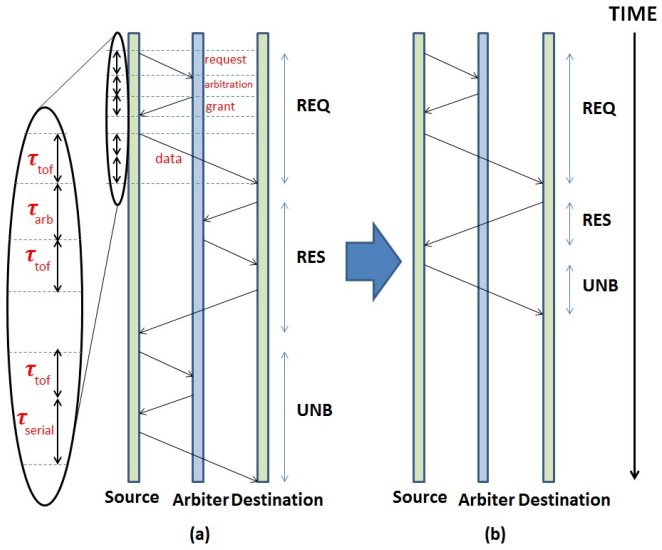

Fig. 2: Timing model of (a) baseline (b) proposed scheduling algorithm. τ_{tof}=time of flight latency between ports and switch, τ_{arb}=arbitration time and τ_{serial}=serialization latency.

was used to generate the traces in order to eliminate the effect of the network at this stage.

Analyzing the traces, it was found that the communication consists of eight common patterns of messages exchanged between any pair of cores, initiated by the cache coherence protocol. The messages consist of request messages (REQ, 8B for read/72B for write), response messages (RES, 8B/72B) and unblock messages (UNB, 8B). Although each of the different benchmarks had very different traffic matrices, their communications consisted mostly of patterns involving mostly two, three and five messages, from which it can be concluded that the patterns depend on the coherence protocol rather than the benchmark.

Figure 3 shows examples of the most commonly occurring three and five message patterns, both representing a store request. In the three message pattern, shown in Figure 3(a), the L1 cache of the source port is requesting exclusive access

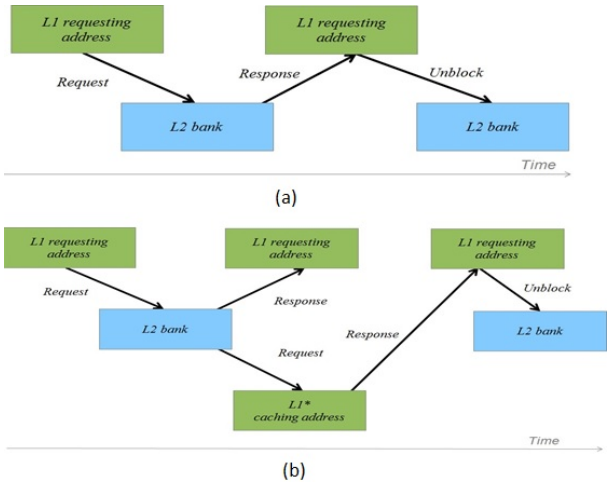

Fig. 3: Most common patterns of three and five messages

to an address in memory by sending a REQ message to the L2 of the destination port. The L2 will then block the address and return a RES message with the blocked address. Finally, an UNB message will be sent between the L1 and L2 caches at the destination port before being sent across to the source port to make the address available exclusively by the L1 of the source port. Similarly, Figure 3(b) represents the most commonly occurring five message pattern consisting of two REQ, two RES and an UNB. The purpose of the five message pattern is very similar to the three message one except that the data comes from a different L1 cache. The two message patterns (not shown) consist of a REQ followed by a RES message.

The main point to note is that each of these patterns is an series messages exchanged between two ports where, once the initial REQ message has entered the network, the rest of the pattern is predictable and can be used to schedule a bidirectional optical circuit for the entire duration of the pattern. From Figure 2, it can be observed that the latency of subsequent messages in the pattern is reduced by twice the time of flight in the network plus the arbitration time, $2\tau_{tof} + \tau_{arb}$. The bidirectional circuit will finally be closed only after all the messages in the specific pattern have been transferred. It also shows the latency savings that would arise from a three message pattern. In this particular situation, the latency of the pattern would be reduced by $4\tau_{tof} + 2\tau_{arb}$. Clearly, longer patterns have proportionately greater latency savings.

There is one refinement required to the algorithm concerning the situation in which the requested memory address is not in any cache on the chip and must be retrieved from main memory. Memory access can take hundreds of clock cycles (set to 600 clocks in the simulations reported in this paper). Holding open optical circuits for this length of time would block other messages waiting to use the switch. Therefore in this case, the destination port signals to the arbiter to tear down the circuits. The memory response and unblock message are

then treated as a separate pattern.

In spite of the considerable latency benefits of this method, there is one obvious drawback: while the bidirectional circuit is setup, there is a period of time in which other messages to or from the ports are blocked and will need to be queued for future transmission. The effectiveness of the algorithm therefore depends on whether the latency savings made by keeping the optical path open outweigh the additional latency introduced by additional queuing of request messages. In the following section, we investigate this using simulations of the proposed algorithm against a baseline switch arbiter.

III. Methodology

Simulations of an optical crossbar switch [13] and arbiter were performed using the 32-core x86 traces as input. Figure 4 represents the arrangement of two cores connected by the optical switch. The switch has two sections; (1) a path allocator/arbiter with a FIFO which will receive control messages from the cores requesting setup and tear down of optical paths and (2) the optical switch with an 80 Gb/s data path (e.g. 8 wavelengths of 10 Gbps), chosen because this bandwidth (and hence serialsation latency) gave optimum results for full system simulations of a CMP interconnected with an optical cross-bar in our previous work [14].

Starting from a position in which all ports are free, when a message enters the source port data FIFO, it will send a path request message to the allocator which enters the request FIFO. If there is an available switch path, a grant message is sent to the source port and the allocator opens a unidirectional optical circuit from the source port to the destination port in the case of the baseline scheduler or a bidirectional circuit in the case of proposed scheduler. To deal with contentions, the round-robin method [15] was to ensure fair allocation. As only output port arbitration is required, our previous 45 nm synthesis results show that the allocator can be scaled up to 64 ports while maintaining single cycle allocation with a clock frequency of up to 1.2 GHz [16]. In these simulations, we conservatively assumed a clock frequency of 1.2 GHz for the 32-port network. Once the grant message is received, the source port router will serialize the message and send it via the now open optical path. With the message sent, in the case of the baseline scheduler, the arbiter will bring down the path and proceed to deal with the oldest request in the request FIFO. In the case of the proposed algorithm, the bidirectional circuit remains open until the memory transaction is complete with subsequent messages in the pattern bypassing the data FIFO. If another memory transaction between the same two cores is initiated before the previous pattern is completed, the circuit is extended, further increasing the latency savings.

From [17], the die area of a 32 core CMP was estimated to be $1400mm^2$. Assuming a topology whereby the switch and arbiter are situated in the center of the die, the maximum distance of any core from the arbiter does not exceed $\sqrt{1400} = 37\ mm$. Assuming an optical link using waveguides with $n_{eff} = 4$ for both control (parallel wavelengths) and data paths, τ_{tof} is 0.49 ns, that is, 59 % of a network clock cycle.

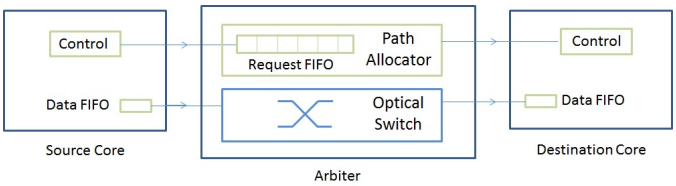

Fig. 4: Simulated optical switch interconnect with electronic sections in green and optical sections in blue

Therefore, the time taken for request and grant messages was rounded up to 1 clock cycle, which explains why a different network frequency (<2 GHz) can be used from the one in the trace file. Similarly the data transfer time ($2\tau_{tof} + \tau_{serial}$) was rounded up to the nearest integer number of clock cycles. At the source and destination ports, the processing time between obtaining a grant and sending the message is taken to be one clock cycle.

IV. Results

Figure 5(a) quantifies the average latency savings per memory transaction (or per pattern) including the network latencies and computation times. Patterns of 2, 3 and 5 messages are included in the figures, therefore, for the proposed algorithm, these times correspond to the average time that the optical circuit remains open. The latency savings per memory transaction ranges from 14.8% for swaptions to 23.8% for vips. Figure 5(b) shows the average overhead latency per message, that is, the latency required for the message to be sent in the network without considering the serialisation time and the time of flight of the data message through the network. This metric eliminates the latency difference between data and control messages and therefore clearly demonstrates the effect of the proposed scheduling algorithm. A significant reduction in the overhead latency ranging from 31.8% reduction for streamcluster up to 70.6% for swaptions is observed.

Although the proposed algorithm shows significant reductions in latency, it is also expected to have increased use of the request FIFO as the bidirectional circuits block other memory transactions. Figures 5(c) and (d) illustrate the extent of the additional queuing and explain why this does not significantly diminish the latency savings. Figure 5(c) shows that the average waiting time in the request FIFO increases using the proposed algorithm, up to a factor of nearly 5 for freqmine and vips. Although this will tend to increase latency compared with the baseline scheduler, Figure 5(d) shows that requests which use the request FIFO for more than one clock cycle constitute a very small proportion of the total communication. Here, for both freqmine and vips benchmarks, less than 2% of total communication will remain in the request FIFO. Even Swaptions, which has the densest communication pattern of the ten investigated, does not exceed 17.9% for the proposed algorithm in the request FIFO. Hence, the overall effect is a substantial latency saving.

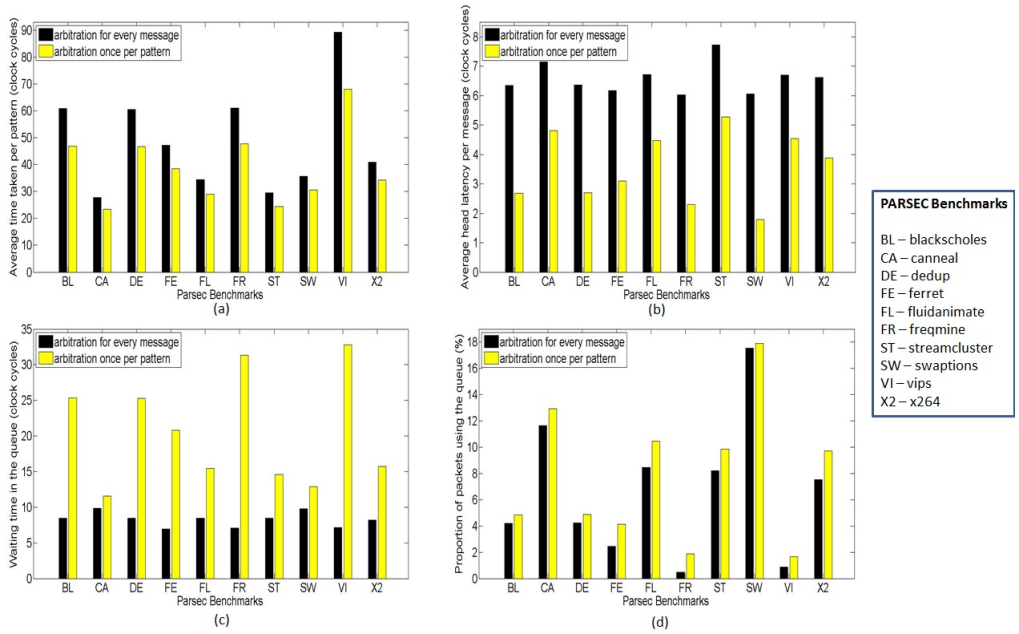

Fig. 5: Simulation results comparing the baseline and proposed scheduling algorithms (a) time taken to complete a pattern of messages (b) individual message latency; (c) average waiting time in request FIFO and (d) percentage of total communication that uses the request FIFO

V. Conclusions

The results in this paper demonstrate that a scheduling algorithm which performs arbitration once per memory transaction rather than per message offers significant overhead latency reductions of up to 70.6%. Despite the potential for blocking other communication while the memory transaction is in progress, the proportion of messages using the request FIFO are relatively low ranging from 1.5% for vips to 17.9% for swaptions. The results show that this additional queuing does not significantly affect overall latency savings. However, for future work we intend to build a full system simulation model of the proposed algorithm within the gem5 framework in order to measure the effect on overall application performance.

This paper demonstrated the latency benefits of the proposed algorithm for CMPs with an optical NoC. However, far greater benefits can be obtained for larger shared memory systems which span multiple chips such as high performance, multiple-socket servers. In this case, the additional time of flight τ_{tof} ensures that the control and arbitration overhead latency saving is further increased. In addition, the proposed algorithm potentially allows greater port counts to be accommodated without a substantially increased arbitration overhead as using several clock cycles for arbitration will only affect the initial REQ message in a pattern.

Acknowledgment

The authors would like to thank Timothy Jones (Univeristy of Cambridge) for his help in setting up the gem5 simulator and for stimulating discussions. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/I004157/2.

References

[1] A. Shacham *et al.*, "Building ultralow-latency interconnection networks using photonic integration," *IEEE Micro*, vol. 27, no. 4, 2007.
[2] D. Vantrease *et al.*, "Corona: System implications of emerging nanophotonic technology," in *Int. Symp. on Comput. Archit.*, 2008.
[3] A. Krishnamoorthy *et al.*, "Computer systems based on silicon photonic interconnects," *Proc. of the IEEE*, vol. 97, no. 7, 2009.
[4] Y. Pan *et al.*, "Firefly: Illuminating future network-on-chip with nanophotonics," in *Proc. Int. Symp. on Comput. Archit.*, 2009.
[5] G. Hendry *et al.*, "Analysis of photonic networks for a chip multiprocessor using scientific applications," in *3rd ACM/IEEE Int. Symp. on Networks-on-Chip*, 2009.
[6] G. Hendry *et al.*, "Time-division-multiplexed arbitration in silicon nanophotonic networks-on-chip for high-performance chip multiprocessors," *Journal of Parallel and Distributed Comput.*, vol. 71, no. 5, 2011.
[7] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. Design Autom. Conf.*, 2001.
[8] J. Shin *et al.*, "A 40 nm 16-core 128-thread sparc soc processor," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, 2011.
[9] I. Iliadis and C. Minkenberg, "Performance of a speculative transmission scheme for scheduling-latency reduction," *IEEE/ACM Trans. on Networking*, vol. 16, no. 1, 2008.
[10] P. Watts *et al.*, "Requirements of low power photonic networks for distributed shared memory computers," in *Opt. Fib. Comm. Conf.*, 2011.
[11] C. Bienia *et al.*, "The parsec benchmark suite: Characterization and architectural implications," Princeton University, Tech. Rep., 2008.
[12] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, 2011.
[13] A. Poon *et al.*, "Cascaded Microresonator-Based Matrix Switch for Silicon On-Chip Optical Interconnection," *Proc. of the IEEE*, vol. 97, no. 7, 2009.
[14] A. Laer *et al.*, "Full system simulation of optically interconnected chip multiprocessors using gem5," in *Optical Fiber Comm. Conf.*, 2013.
[15] E. Shin *et al.*, "Round-robin arbiter design and generation," in *Proc. Int. Symp. on Syst. Synthesis*, 2002.
[16] P. Watts *et al.*, "Energy implications of photonic networks with speculative transmission," *IEEE/OSA Jour. of Opt. Comm. and Netw.*, vol. 4, no. 6, 2012.
[17] J. Zhao *et al.*, "Cost-aware three-dimensional (3d) many-core multiprocessor design," in *Proc. 47th Design Autom. Conf.*, 2010.