Scalability of Weak Consistency in NoC based Multicore Architectures

Abdul Naeem[1], Xiaowen Chen[1, 2], Zhonghai Lu[1] and Axel Jantsch[1]

[1] Department of Electronic Systems,
School of Information and Communication Technology,
Royal Institute of Technology
Stockholm, Sweden
{abduln, xiaowenc, zhonghai, axel}@kth.se

[2] Institute of Microelectronics and Microprocessor,
School of Computer Science,
National University of Defense Technology,
Changsha, China
xwchen@nudt.edu.cn

Abstract—**In Multicore Network-on-Chip, it is preferable to realize distributed but shared memory (DSM) in order to reuse the huge amount of legacy code. Within DSM systems, memory consistency is a critical issue since it affects not only performance but also the correctness of programs. In this paper, we investigate the scalability of the weak consistency model, which may be implemented using the concept of a transaction counter. Our experimental results compare synchronization latencies for various network sizes, topologies and lock positions in the network. Average synchronization latency rises exponentially for mesh and torus topologies as the network size grows. However, torus limits the synchronization latency in comparison to mesh. For mesh topology network average synchronization latency is also slightly affected by the lock position with respect to the network center.**

Keywords-Synchronization, Scalability, Memory consistency, Distributed shared memory.

I. INTRODUCTION

The general trend in processor development has been shifted from single sequential processor to parallel multicore systems. Most computer companies AMD, Intel, Sun, ARM and IBM have shifted their next generation designs to be based on multicore systems [1, 2, 3]. Distributed nodes are integrated by scalable, reliable, high-bandwidth and low-latency network-on-chip (NoC) in multicore systems. NoC based multicores (McNoC) are promising solutions to modern and future processor design challenges [4, 5]. In order to reuse the huge amount of legacy code distributed but shared memory organization is preferred. Multi-threaded applications running on McNoC architectures suffer with problem of memory consistency. Various memory consistency models (MCMs) have been proposed as alternative solutions [6]. Sequential consistency does not allow performance optimizations due to strictness in the program order. Relaxed consistency models (weak ordering, release consistency) allow these optimizations and improve system performance [7]. Weak consistency distinguishes shared memory accesses as synchronization and data operations. Atomic synchronization operations on reserved shared variables (locks) protect shared data operations (critical section). Data operations can be reordered and overlapped in weak consistency model.

We investigated scalability of the transaction counter based weak consistency model in the NoC based system. The model was tested with the lock maintained in the shared address space. This paper does not compare it with any other consistency model. We explored and compared synchronization latencies only for weak consistency model for various network sizes, topologies and lock position in the network. As the network size scales average and maximum synchronization latencies increase exponentially for both mesh and torus topologies. It is due to the network congestion, delay and waiting time in relevant virtual channel to acquire the lock. The results indicate that synchronization overhead is significant in larger networks. Torus topology limits this synchronization latency as compared to mesh topology in larger networks. For small network sizes up to 4 nodes synchronization latencies in torus and mesh topologies are almost the same. Results show a huge difference in synchronization latencies of torus and mesh topologies for very large network sizes. We also investigated that position of lock with respect to network center also affect synchronization latency in mesh topology.

The rest of the paper is organized as follows. In the next section, we describe related work to scalability analysis of synchronization in multicore systems. In section III, weak memory consistency is discribed. In section IV, transaction counter based weak consistency for NoC based systems has been focused.Section V describes our simulation results and scalability analysis of the consistency model in the NoC system. Section VI summarizes our contribution and future plan.

II. RELATED WORK

NoC work has so far mainly focused on architectural and modeling aspects. Very few researchers have worked on scalability analysis of synchronization among cores in NoC based multiprocessors system. Oreste Villa et al. [8] performed quantitative analysis to understand how different topologies behave when dealing with different SW/HW barriers implementations in NoC System. Four different barriers were implemented and evaluated for a number of cores (from 4 to 128) and five different network topologies. Different scaling behaviors were observed for some barriers with respect to theoretically expected. Simple network topologies proved to be more efficient than complex and highly connected topologies. However their work focuses off chip main memory rather than on chip distributed shared memory. Petrini et al. [9] analyzed

scalability of HW and SW based barriers designed and implemented in a programmable network interface card for quadrics interconnection networks. This work evaluated that HW approach is efficient than SW approach in the presence of network contention. Without network contention both algorithms can be used interchangeably on a flat-tree topology for systems of 64-128 nodes. But this work is not related to on chip scalability analysis. Sarita V et al. [7] discussed memory consistency issues with an emphasis on the system optimizations they allow. They proposed the counter to realize weak memory consistency in general multiprocessor parallel systems. The proposed counter keeps track of outstanding data operations between two synchronization operations. We implemented the counter based approach in specific NoC based distributed shared memory multi-cores system. Petro et al. [10] explored the reordering of synchronization and data operations due to diverse paths, routing scheme and physical location of target in the network. CPU directly uses physical addresses to avoid TLB flushes. This particular application also did not use virtual memory.

III. WEAK MEMORY CONSISTENCY

Sequential consistency requires strict program order for individual processor and sequential order among multiprocessors in parallel system. Sequential consistency enforces order for each individual shared memory access according to program order. Sequential consistency does not allow performance optimizations in the hardware (cache, interconnection network) and software (compiler reordering, register allocation) for multiprocessors system [7]. Relaxed memory consistency models (weak consistency, release consistency) permit such optimizations. Relaxed models relax program order and enhance system performance as compared to sequential consistency. Relaxed models rely on synchronizations among multiple processors in the system. It makes programmer responsible for concurrency issues. Weak consistency model classifies shared memory accesses as synchronization and data operations. Weak consistency enforces the following global orders on shared memory accesses:

- Synchronization to data
- Data to synchronization
- Synchronization to Synchronization

Data operations before, after and between synchronization operations can be reordered (1, 2, 3) as shown in Figure 1. All previous data operations must be completed before issuance of synchronization operation and vice versa. There is interference problem between synchronization and data operations. We illustrate transaction counter based solution to this problem in the upcoming section.

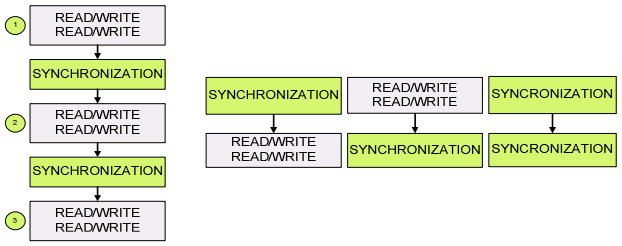

Figure 1. a) Weak Ordering b) Global orders to enforce

IV. TRANSACTION COUNTER BASED WEAK CONSISTENCY FOR NOC BASED PLATFORM

A. NoC Platform

Figure 2a shows a homogenous McNoC having same types of nodes. Each node represents a typical processor-memory (PM) node in the NoC based platform. 2D mesh topology for 4x4 network size is given in the same Figure. PM node consists of a processor, synchronization handler, transaction counter, network interface and local memory as shown in Figure 2b. The network interface performs packetization, de-packetization, queuing and connects a PM node to the NoC. Router uses routing algorithm to rout the packets to proper destinations. Memories are preferably distributed because the traditional centralized memory system has become the performance, power and area bottleneck in on-chip systems [11]. In our synthetic McNoC platform each node can have a local memory. All local memories can logically form a single global memory address space. Software developers can get benefit from ease of shared memory programming.

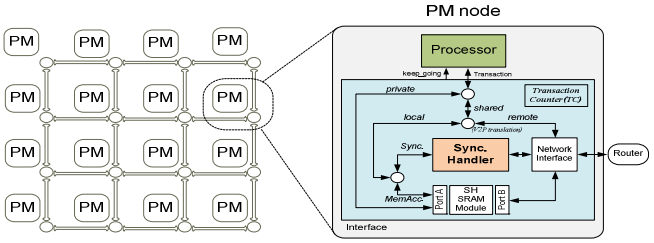

Figure 2. a) Homogeneous McNoC b) PM node

Synchronization handler is mainly composed of a synchronization variable pool, scheduling logic, two physical channels and a crossroad. Synchronization variable pool contains N locks one bit each. These locks have special reserved addresses in shared address space known to software developers. Programmer can use it by relevant programming language construct. There are two access ports for these locks one from local processor and other from network. Two physical channels respond simultaneously to two synchronization requests from local processor and network. Crossroad dispatches the synchronization requests to the proper physical channels. Scheduling logic controls the crossroad to determine directions of synchronization requests. It also organizes the two physical channels to be N virtual channels logically. Synchronization handler monitors lock's status, maintains the virtual channels and perform correct actions on the coming synchronization requests. It handles lock requests in an efficient way to reduce the contention, overhead and improve the response time. To access a shared lock in sequential order (mutually exclusive) synchronization handler architecture uses logical virtual channel per lock to maintain synchronization requests over the same lock.

B. Transaction counter based weak consistency

We adopted the transaction counter based approach [7] to realize the weak memory consistency model in the McNoC. Transaction counter in each node keeps track of outstanding data operations. Counter is incremented and decremented by issuance and completion of data operations correspondingly. It is not affected by synchronization operations. Counter zero value indicates completion of all previously issued data operations. Synchronization operations are not issued until the transaction counter becomes zero. Figure 3 illustrates shared

memory access operations that are initiated by processor in each node. After virtual to physical address translation shared memory access is checked whether it is in the local or remote node. Local shared memory accesses are accomplished within the same node. For remote shared memory accesses message passing is performed to remote node. Shared memory accesses operations are classified into synchronization and data operations. Local data operations in critical section of code are issued to shared locations (1) within the same node. A data operation may be memory read (load) or write (store) operation and is completed by either local data return or write acknowledgment respectively (5-1). Local data operations issued to shared locations (2) in non-critical section of code are also completed locally within the same node either by local data returns or write acknowledgments (5-2). Issuance and completion of these local data operations affects transaction counter in the local node. Synchronization operations are not issued to local or remote synchronization handlers until transaction counter in the local node becomes zero. Issuance and completion of these synchronization operations does not affect transaction counter in the local node. Local synchronization operations are issued (3) to local memory mapped synchronization handler and are completed by synchronization acknowledgments (5-3).

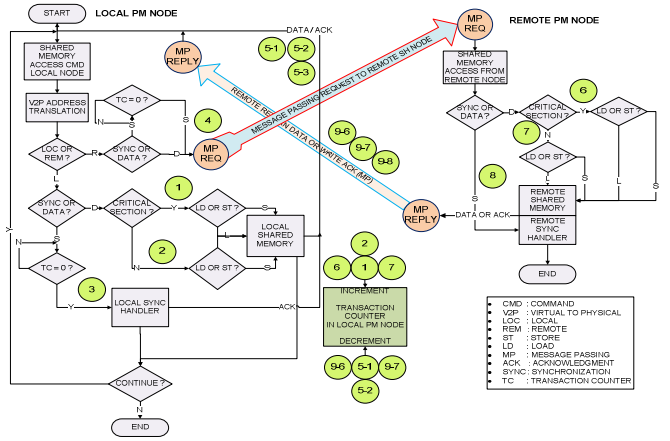

Figure 3. Transaction counter based weak consistency in NoC based system

For remote memory accesses (data, synchronization operations) message passing (4) is carried out to remote node and network. Remote shared data operations in critical section of code are issued to remote shared locations (6). Remote data operations are completed either by remote data returns or write acknowledgments (9-6). Similarly (7) and (9-7) are for remote data operations in non-critical section of code. Issuance and completion of these remote data operations also affect the same transaction counter in the local node. Remote synchronization operations (8) are issued to remote memory mapped synchronization handler. Overall transaction counter in each node is incremented with issuance of local data operations (1, 2) and remote data operations (6, 7). It is decremented by completion of previously issued local data operations (5-1, 5-2) and remote data operations (9-6, 9-7). It is not affected by local synchronization operations (3, 5-3) and remote synchronization operations (8, 9-8).

V. EXPERIMENTS AND RESULTS

We analyzed scalability of weak consistency model in the NoC system. We does not compare it with any other scheme. Tests were performed for various network sizes, topologies and lock position in the network and synchronization latencies were compared. In our experimental setup processor is replaced by stimulus in each node to initiate both data and synchronization operations. Synchronization handler in each node deals with synchronization operations. Each synchronization handler has 256 locks in synchronization variable pool maintained in shared address space. These locks can be accessed by two ports. Local processor accesses these locks locally via one port and remote processors (through network) via other port. Two physical channels in a synchronization handler hold requests from local processor and network. Each lock is associated with a virtual channel in synchronization handler. Each virtual channel combines together the synchronization requests in the two physical channels over the same dependent lock. Scheduling logic monitors locks status. It acquires a lock on synchronization request if the lock is available otherwise place the request in relevant virtual channel for its turn. Scheduling logic release a lock on synchronization request directly if there is no acquire request waiting for the same lock. Otherwise lock acquire ownership is transferred from releasing request to the oldest awaiting acquire request. Up-down 32 bit transaction counter avoids interference between synchronization and data operations. The network interface in each node connects that particular node with the NoC. Our proposed NoC supports both 2D mesh and torus topologies. Priority based round-robin arbitration and X-Y deterministic routing with X-first are used. Our test platform uses distributed shared memory architecture.

A. *Impact of network size*

Both lock and critical section were maintained in the center node of the network for the tests. Mesh topology was considered for these particular experiments. Tests were performed by increasing size of the network from single core to 8x8 multicores in the system. Results in Figure 4 indicate considerable increase in average and maximum synchronization latencies with growth of the network size. Synchronization latency increases exponentially with scaling of network. Synchronization latency is most for 8x8 network size. Average synchronization latency for 8x8 network size is approximately 1130 times larger than that of single core. Average synchronization latency is in the order of 1130 cycles for 8x8 network size. While that for single core it is one cycle. Increasing trend in synchronization latency is mainly due to increasing network traffic, delay and waiting time in relevant virtual channel for acquiring the same dependent lock. The results indicate that synchronization overhead is substantial in larger networks.

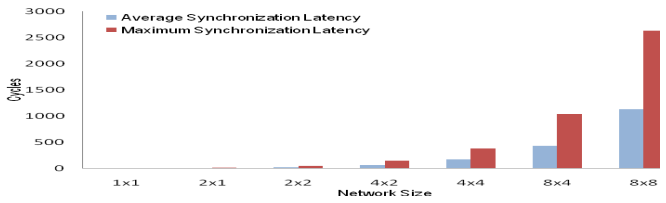

Figure 4. Impact of Network Size

B. *Impact of network topology*

The same set of experiments was also performed for torus topology. We examined affect of network topologies on synchronization latency in addition to the network size. Average synchronization latencies were compared for both topologies with increasing network size as shown in Figure 5.

Average synchronization latencies for both the network topologies are almost same for smaller network sizes up to 4 nodes. For larger network sizes the difference between average synchronization latencies become apparent and wider. Average synchronization latencies increases exponentially for both the topologies as the network size grows. For mesh topology as mentioned earlier average synchronization latency for 8x8 network size is approximately 1130 times of single core, where as for tours it is 860 times. For 8x8 network size difference in average synchronization latencies for both the topologies is the highest (270 cycles). For smaller network sizes up to 4 cores the difference is almost zero. The increasing behavior of the difference is also exponential. A huge difference in synchronization latencies for both topologies is predicted in very large sizes networks. Torus performed better and is scalable due to increasing path diversity in its structure as compared to mesh counterpart.

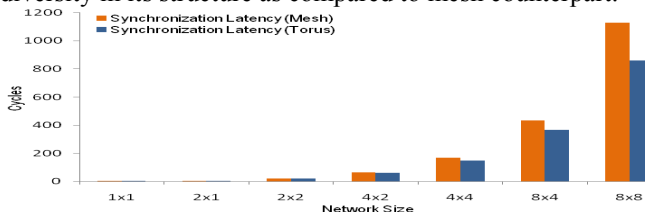

Figure 5. Impact of Network Topology

C. *Impact of lock position in the network*

Impact of lock position in the network on synchronization latency was also studied. We experimented on 7x7 size mesh network. Critical section was maintained in the center node of the network. Lock position was changed node by node in the network. Due to symmetry in the network we got the same synchronization latency for a specific nodes group. Nodes are categorized into various groups (NG1 to NG10) according to different synchronization latencies and are given in Table I.

TABLE I. NODES GROUP

NG1 : 00,60,06,66.	NG6 : 12,21,41,52,14,25,45,54.
NG2 : 10,01,05,16,50,61,65,56.	NG7 : 31,53,35,13.
NG3 : 02,20,04,26,40,62,64,46.	NG8 : 22,42,24,44.
NG4 : 03,30,63,36.	NG9 : 23,32,43,34.
NG5 : 11,51,15,55.	NG10 : 33.

NG1 has four corner nodes and NG10 has single center node in the network. Node number is XY coordinates in the 2D plane. Average synchronization latencies are highest when lock is maintained in the corner nodes as given in Figure 6.

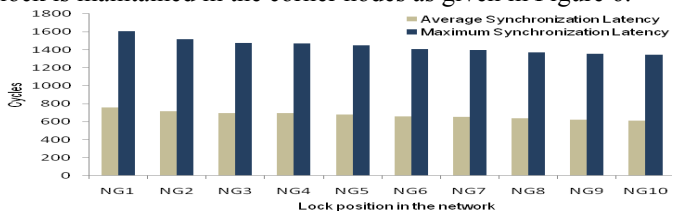

Figure 6. Impact of lock position in the mesh network

It is due to non-uniform access time for synchronization. Synchronization latency gradually decreases as the lock position is changed from corner to center of the network. Average synchronization latency at corner nodes is the most (760 cycles). Which is approximately 1.24 times larger than lock is positioned in center node of the network. Position of lock with respect to network center affects synchronization latency to some extent.

VI. CONCLUSION AND FUTURE WORK

The primary goal is to analyze scalability of critical synchronization operations in transaction counter based weak consistency in NoC based MPSoC platform. We observed that transaction counter avoided interference problem between data and synchronization operations. Average synchronization latencies increases exponentially for both mesh and torus topologies as network size scales. Torus topology limits this synchronization latency as compared to mesh topology. We also analyzed that position of lock with respect to network center also effect synchronization latency slightly in the mesh network. Our experiment results show that system performance is limited by synchronization performance in very large scale networks. The network topologies and lock positions with respect to network center also effect synchronization performance. In the future optimized network designs are desired. Also, we will study the scalibility of other relaxed memory consistency models like the release consistency model.

ACKNOWLEDGMENTS

This work has been supported partially by the FP7 EU project Mosart under contract number IST-215244, the SI/HEC joint scholarships program of Pakistan and Sweden.

REFERENCES

[1] PHAM D.C., AIPPERSPACH T., BOERSTLER D., ET AL.: 'Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor', IEEE J. Solid-State Circuits, 2006, 41, (1), pp. 179–196

[2] BELL S., EDWARDS B., AMANN J., ET AL.: 'TILE64TM processor:A 64-core SoC with mesh interconnect'. Digest of TechnicalPapers, IEEE Int. Solid-State Circuits Conf., February 2008, vol. 51, pp. 88–598

[3] STACKHOUSE B., CHERKAUER B., GOWAN M., ET AL.: 'A 65-nm 2- billion-transistor quad-core Itaniumw processor'. Digest of Technical Papers, IEEE Int. Solid-State Circuits Conf., February 2008, vol.51, pp.92–598.

[4] L. Benini and G. D. Micheli. Networks on Chip: A new SoC paradigm. IEEE Computer, 35(1):70–78, January 2002.

[5] W. J. Dally and B. Towles. Route packets, net wires: on-chip inteconnectoin networks. In DAC'01: Proceedings of the 38th conference on Design automation, pages 684–689, New York, NY, USA, 2001. ACM Press.

[6] S. V. Adve and K. Gharachorloo, "Shared Memory Consistency Models: A Tutorial," IEEE Computer, Vol. 29 No. 12, pp. 66–76, Dec. 1996.

[7] Sarita V. Adve and Kourosh Hgarachorloo, Shared Memory Consistency Models: A Tutorial, Digital Western Research Laboratory, report no. 95/7, Palo Alto, California 94301 USA, September 1995.

[8] O. Villa, G. Palermo, C. Silvano, "Efficiency and Scalability of Barrier Synchronization on NoC Based Many-core Architectures". In Proceedings of CASES 2008- International Conference on Compilers, Architectures and Synthesis for Embedded Systems. Atlanta, Georgia, USA, October 2008, pp. 81-90.

[9] F. Petrini, S. Coll, E. Frachtenberg, and A. Hoisie. Hardware- and software-based collective communication on the quadrics network. In NCA '01: Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA'01), page 24, Washington, DC, USA, 2001. IEEE Computer Society.

[10] F. Petrot, A.Greiner, P. Gomez, "On cache coherency and memory consistency issues in NoC based shared memory multiprocessor SoC architectures", 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools, 2006, Pages: 53-60.

[11] E.J. Marinissen, B. Prince, D. Keltel-Schulz and Y. Zorian, "Challenges in embedded memory design and test", Proceedings of Design, Automation and Test in Europe Conference (DATE'05), vol. 2, pp. 722-727, Mar. 2005.