Taking Advantage of the SHECS-Based Critical Sections in the Shared Memory Parallel Architectures

Tomasz Madajczak[1,2]

[1] Gdansk University of Technology, Faculty of Electronic, Telecommunication and Informatics, Dep. of Computer Systems Architectures, ul. Narutowicza 11/12, 80-233 Gdansk, Poland
[2] Intel Technology Poland Sp. z o.o. ul. Slowackiego 173, 80-298 Gdansk, Poland
Tomasz.Madajczak@intel.com

Abstract. This document presents a new method for implementing critical sections in the shared memory parallel architectures such as multithreaded multiprocessors integrated on a die. The method bases on Shared Explicit Cache System (SHECS) implemented in the multiprocessor. The document presents the concept of system architecture equipped with SHECS, the algorithm to implement operating system or application level locking service, and the results obtained with the method simulation on the network processor Intel[1] IXP2800.

1 Introduction

The Multicore Shared Memory Parallel Architectures are becoming very popular thanks to mass availability of the multicore SMP (symmetric multi-processors) systems such as multicore IA (Intel Architecture) processors and multicore specialized RISC systems such as the IXA (Intel Exchange Architecture) network processors. The level of parallelism in such systems is enhanced by the hardware threading technologies such as simultaneous multithreading (SMT) and switch-on-event multithreading (SoEMT), respectively. Efficient use of shared resources is always connected with the need of a critical section implementation. Multicore and multiprocessors systems have the ability to rely on specific hardware support and capabilities to synchronize the particular processor cores within the die or the integrated platform. Hardware techniques for critical sections are available in the network processors and they are very efficient, but not always universal [1]. Software techniques are still available for the SMP systems and for cases when the hardware support isn't flexible.

Thus, there is a need for a flexible hardware method that would address the perspectives of multithreaded multiprocessors systems. This document presents such a new method based on the use of SHECS being an additional, manageable, explicit cache system. It is derived from the Folding method [2].

[1] Intel is a registered trademark of Intel Corporation in the United States and other countries.

R. Wyrzykowski et al. (Eds.): PPAM 2005, LNCS 3911, pp. 26–33, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1.1 Folding Method in Network Processors

The Folding method was introduced in the network processors Intel IXP2000 [3] as a universal method for programming software critical section for the threads of a single microengine (that is a RISC processor). The method uses the microengine's internal local memory and CAM (content addressable memory) lookup engine that comprises an internal explicit cache system managed with software.

Folding caches the read data to be modified. It manages with the following critical section scenario that firstly reads a resource, then modifies it, and finally writes back the modified data. The read resource is stored within this system and considered locked if its use counter is greater than 0. Folding may occasionally lost the order of threads entering the critical section, because in case of finding locked entry the algorithm repeats the entering. This order lost means that the critical section may be starving for some threads, as they have no luck and they always repeat entering. Extending the Folding method onto a number of processors is not an easy task and also starving critical sections aren't useful in the general purpose parallel systems.

All these issues are solved in the SHECS-based method. It bases on the new explicit cache memory system architecture that eliminates the Folding's limitations and disadvantages. Thus, the SHECS-based method is more universal, flexible, and may have more applications.

1.2 Cache Coherency

The Parallel Shared Memory Architectures built with using general purpose processors with internal caches may have implemented a mechanism for enforcing the coherence of internal caches. Hardware solutions for this problem are presented in [4][5], while software algorithms are discussed in [6][7]. Generally there are two approaches: implicit methods that hide the problem for the system user or software, and explicit methods assuming that the system user or software is aware of the problem, treats cache as a normal shared resource and solves it with cache-locking [8][9] or other locking method. The introduced method addresses the problem in the similar way as explicit methods. It assumes explicit locking with integrated data transfers of the most recent cached data value and additionally it copes with hardware threading.

2 The Concept of SHECS-Based Critical Sections

2.1 SHECS Architecture

SHECS consists of CAM banks controller, a number of CAM banks and some shared fast SRAM memory for caching[2]. Fig. 1 shows that SHECS should be connected in a similar way as shared memory to the parallel processors P1-PN. The key assumption is providing a number of CAM banks that can

[2] The concept of Shared Explicit Cache System is patent pending in the U.S. patent office.