

Efficient Memory Synchronization Techniques for Embedded System Architectures

Iris Bahar², Cesare Ferri², Maurice Herlihy², Tali Moreshet¹, Amber Viescas¹

1. Swarthmore College

2. Brown University

aviesca1@swarthmore.edu, tali@swarthmore.edu

Abstract

As multi-processor systems proliferate, designers for embedded systems must consider new memory designs that avoid common problems such as deadlock and priority inversion. In this study, the MPARM Simulator runs numerous benchmarks designed to test a transactional memory system—previously only studied for application for general-purpose computers—and compare energy and performance to traditional lock-based memory systems. The analysis considers new concerns with embedded systems—specifically, energy efficiency, and addresses them by introducing a transactional cache shutdown policy.

Introduction

As embedded systems, such as those found in cell phones and digital cameras, become more complex, memory requirements become more stringent. Individual threads must alter data atomically; in other words, changes made to the database must be visible to all processes near-instantly, to ensure that each process has the most recent data. To date, lock-based systems are the most common way of accomplishing this task.

Lock-based systems are simple in principle: each piece of data is guarded by a lock. In order to access that data, the process must acquire the lock. The process releases the lock when finished. There are several problems in this system that are difficult to work around, and put much of the burden on the individual programmer. In addition, designers of memory systems must choose how large of a data section they want to protect with a single lock.

Transactional Memory for Embedded Systems

Transactional memory has the potential to avoid many of these problems, increasing parallel efficiency. It is an optimistic system—meaning it assumes memory conflicts are rare and plans accordingly. A transactional memory system uses a transactional cache, in which a processor stores old data and potential new data. While the process writes the new data, the snoop device monitors cache coherency. If the data has not been altered by another process, the transaction commits, and the “potential” data becomes available to all processors. Otherwise, the transaction aborts, the new data is discarded, and the process is rolled back to the point before the transaction was attempted. A short backoff period follows a failed

transaction, after which the transaction is attempted again.

Although the implementation of transactional memory in both hardware and software has been widely studied, this study focused mainly on the hardware implementations of transactional memory, which is more appropriate for embedded systems.

When dealing with embedded systems, energy consumption is an especially important issue. TM systems often save energy by virtue of being faster than locking systems, and by avoiding “lock spin.” However, by default, the Transactional Cache can be a huge drain on energy. In this study, different shutdown techniques are pursued, but only two prove to be effective: the “default” policy in which the TC is always on (useful for systems with many transactions) and an “aggressive shutdown” policy in which the TC is shut down after every transaction. Aggressive shutdown is especially useful when the **critical section**, or percentage of data shared by all processes, is small.

Simulation Setup

The MPARM simulator, written in SystemC to simulate a multi-core ARM processor, was the main environment for testing TM and its comparison with lock-based systems. Inside the environment, memory systems manage data storage for several different benchmarks, each using different data structures. The Metrics Micro Benchmark Suite is a set of benchmarks that simulate normal workload on a matrix algorithm. The Red-black Benchmark simulates multiple red-black binary trees, and the Skiplist Benchmark focuses on linked lists, the latter of which has the greatest potential for parallelism.

Results

We found that results vary for different benchmarks, with the size of critical section and number of processor cores. The Metrics Micro Benchmark gains the most improvement from aggressive shutdown protocol, but the Red-black and Skiplist Benchmarks do best when the TC is not shut down.

Overall, when dealing with many cores (4 or more), transactional memory provides significant improvement over a lock-based system, when considering latency and energy together—using TM results in up to an 80% reduction in the energy-delay product of the system. Even with two cores, when the critical section is large, TM performs competitively to a lock-based system.