Multiprocessor Consistency and Synchronization Through Transient Cache States

Erik Hagersten, Anders Landin and Seif Haridi
Swedish Institute of Computer Science
Box 1263
S-164 28 Kista, Sweden

Abstract

A transient state can be used to mark a cache line for which an access have started, but not yet completed. It can be used to implement cache-coherence protocols for split transaction buses.

Transient states can also be used to implement nonblocking writes, i.e. the processor never stalls on a write, while providing processor consistency for a certain class of networks. This has earlier only been achieved for looser forms of consistency at an extra hardware cost.

The same technique can be used to resolve data dependencies at runtime, implementing a functionality similar to that of Dataflow's I-structure memory, at no extra hardware cost.

Keywords: Multiprocessor, hierarchical architecture, hierarchical buses, multilevel cache, split-transaction bus, cache coherence, transient cache state, synchronization, processor consistency.
1 INTRODUCTION

Shared-memory multiprocessors with caches local to the processors often rely on a cache-coherence protocol to enable the programmer to observe a shared-memory view of the system. The protocol also provides some consistency level, specifying the order in which processors observe accesses from each other.

We have earlier shown how transient states can be used to implement a cache-coherence protocol for hierarchical buses [Hagersten et al., 1990]. The protocol makes use of transient states to implement a split-transaction protocol, allowing a bus to be released between a request and its reply. The consistency level provided by that protocol is the strongest level most often assumed by the programmer: sequential consistency [Lamport, 1979].

The weaker orders of consistency [Dubois et al., 1986, Gharachorloo et al., 1990] that have been proposed rely on special synchronization operations recognized by hardware. They allow for accesses to bypass each other, resulting in improved performance. However, the solutions ask for extra hardware support and introduce a new model to the programmer.

Processor consistency is an intermediate level of consistency introduced by Goodman [Goodman, 1989]. It provides a programming model closer to that of sequential consistency, while providing nearly the same performance as the weaker orders of consistency [Gharachorloo et al., 1991].

This paper discusses the use of transient states in processor caches. We have summarized the assumptions about the rest of the architecture in Section 2. The next section, Section 3, describes the general use of transient states in a cache-coherent protocol. Section 4 describes the sequentially consistent protocol reported earlier [Hagersten et al., 1990]. It features an efficient handling of write-invalidate, where a write acknowledge might be received before all other copies are erased. The protocol introduced in Section 5 implements processor consistency, allowing multiple outstanding writes, and where reads are allowed to bypass writes, by the use of transient states at (almost) no extra hardware cost.

The paper ends with a section about synchronization showing how transient states can be used to implement a synchronization similar to that of the I-structure memory in Dataflow [Arvind and Nikhil, 1989].