Parallel Computer Architecture

A HARDWARE/SOFTWARE APPROACH

David E. Culler
Jaswinder Pal Singh
with Anoop Gupta

Apple, Inc. et al. v.
Memory Integrity, LLC
IPR2015-00159. -00161,
-00163, -00172
EXHIBIT
Memory Integrity - 2002

DOCKET ALARM
Find authenticated court documents without watermarks at docke

Senior Editor	Denise E.M. Penrose
Director of Production and Manufacturing	Yonie Overton
Senior Production Editor	Elisabeth Beller
Editorial Coordinator	Meghan Keeffe
Cover Design	Martin Heirakuji Graphic Design
Cover Photo	Image copyright © 1998 PhotoDisc, Inc.
Text Design	Mark Ong, Side by Side Studios
Copyeditor	Jennifer McClain
Proofreaders	Jennifer McClain, Jeff Van Beuren, Ken DellaPenta, Christine Sabooni
Compositor	Nancy Logan
Illustrators	Nancy Logan, Dartmouth Publishing, Inc., Cherie Plumlee
Indexer	Steve Rath

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances where Morgan Kaufmann Publishers, Inc. is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA

Telephone	415/392-2665
Facsimile	415/982-2665
Email	mkp@mkp.com
WWW	http://www.mkp.com

Order toll free 800/745-7323

Advice, Praise, and Errors: Any correspondence related to this publication or intended for the authors should be addressed to the Editorial and Sales Office of Morgan Kaufmann Publishers, Inc., Dept. PCA APE or sent electronically to *pca@mkp.com*. Please report errors by email to *pcabugs@mkp.com*. Please check the errata page at *http://www.mkp.com/pca* to see if the bug has already been reported and fixed.

© 1999 Morgan Kaufmann Publishers, Inc.
All rights reserved
Printed in the United States of America

Transferred to Digital Printing, 2011

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science and Technology Rights Department in Oxford, UK. Phone: (44) 1865 843830, Fax: (44) 1865 853333, e-mail: permissions@elsevier.co.uk. You may also complete your request on-line via the Elsevier homepage: http://www.elsevier.com by selecting "Customer Support" and then "Obtaining Permissions".

Library of Congress Cataloging-in-Publication Data
Culler, David E.
Parallel computer architecture: a hardware/software approach /
David E. Culler, Jaswinder Pal Singh, with Anoop Gupta.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-1-55860-343-1 ISBN-10: 1-55860-343-3
1. Parallel computers. 2. Computer architecture. I. Singh, Jaswinder Pal. II. Gupta, Anoop. III. Title.
QA76.58.C85 1999
004'.35—dc21 98-28034
CIP

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

The check to determine if a bus transaction is relevant to a cache is essentially the same tag match that is performed for a request from the processor. The action taken may involve invalidating or updating the contents or state of that cache block and/or supplying the latest value for that block from the cache to the bus.

A snoopy cache coherence protocol ties together two basic facets of computer architecture that are also found in uniprocessors: bus transactions and the state transition diagram associated with a cache block. Recall that the first component—the bus transaction—consists of three phases: arbitration, command/address, and data. In the arbitration phase, devices that desire to initiate a transaction assert their bus request, and the bus arbiter selects one of these and responds by asserting its grant signal. Upon grant, the selected device places the command, for example, read or write, and the associated address on the bus command and address lines. All devices observe the address and, in a uniprocessor, one of them recognizes that it is responsible for the particular address. For a read transaction, the address phase is followed by data transfer. Write transactions vary from bus to bus according to whether the data is transferred during or after the address phase. For most buses, a responding device can assert a wait signal to hold off the data transfer until it is ready. This wait signal is different from the other bus signals because it is a wired-OR across all the processors; that is, it is a logical 1 if any device asserts it. The initiator does not need to know which responding device is participating in the transfer, only that there is one and whether it is ready.

The second basic facet of computer architecture leveraged by a cache coherence protocol is that each block in a uniprocessor cache has a state associated with it, along with the tag and data, which indicates the disposition of the block, (e.g., invalid, valid, dirty). The cache policy is defined by the *cache block state transition diagram*, which is a finite state machine specifying how the disposition of a block changes. Transitions for a cache block occur upon access to that block or to an address that maps to the same cache line as that block. (We refer to a cache block as the actual data, and a line as the fixed storage in the hardware cache, in exact analogy with a page and a page frame in main memory.) While only blocks that are actually in cache lines have hardware state information, logically, all blocks that are not resident in the cache can be viewed as being in either a special "not present" state or in the "invalid" state. In a uniprocessor system, for a write-through, write-no-allocate cache (Hennessy and Patterson 1996), only two states are required: valid and invalid. Initially, all the blocks are invalid. When a processor read operation misses, a bus transaction is generated to load the block from memory and the block is marked valid. Writes generate a bus transaction to update memory, and they also update the cache block if it is present in the valid state. Writes do not change the state of the block. If a block is replaced, it may be marked invalid until the memory provides the new block, whereupon it becomes valid. A write-back cache requires an additional state per cache line, indicating a "dirty" or modified block.

In a multiprocessor system, a block has a state in each cache, and these cache states change according to the state transition diagram. Thus, we can think of a block's cache state as being a vector of p states instead of a single state, where p is the number of caches. The cache state is manipulated by a set of p distributed finite state

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

machines, implemented by the cache controllers. The state machine or state transition diagram that governs the state changes is the same for all blocks and all caches, but the current state of a block in different caches is different. As before, if a block is not present in a cache we can assume it to be in a special "not present" state or even in the invalid state.

In a snooping cache coherence scheme, each cache controller receives two sets of inputs: the processor issues memory requests, and the bus snooper informs about bus transactions from other caches. In response to either, the controller may update the state of the appropriate block in the cache according to the current state and the state transition diagram. It may also take an action. For example, it responds to the processor with the requested data, potentially generating new bus transactions to obtain the data. It responds to bus transactions by updating its state and sometimes intervenes in completing the transaction. Thus, a *snooping protocol* is a distributed algorithm represented by a collection of cooperating finite state machines. It is specified by the following components:

- the set of states associated with memory blocks in the local caches
- the state transition diagram, which takes as inputs the current state and the processor request or observed bus transaction and produces as output the next state for the cache block
- the actions associated with each state transition, which are determined in part by the set of feasible actions defined by the bus, the cache, and the processor design

The different state machines for a block are coordinated by bus transactions.

A simple invalidation-based protocol for a coherent write-through, write-no-allocate cache is described by the state transition diagram in Figure 5.5. As in the uniprocessor case, each cache block has only two states: invalid (I) and valid (V) (the "not present" state is assumed to be the same as invalid). The transitions are marked with the input that causes the transition and the output that is generated with the transition. For example, when a controller sees a read from its processor miss in the cache, a BusRd transaction is generated, and upon completion of this transaction the block transitions up to the valid state. Whenever the controller sees a processor write to a location, a bus transaction is generated that updates that location in main memory with no change of state. The key enhancement to the uniprocessor state diagram is that when the bus snooper sees a write transaction on the bus for a memory block that is cached locally, the controller sets the cache state for that block to invalid, thereby effectively discarding its copy. (Figure 5.5 shows this bus-induced transition with a dashed arc.) By extension, if any processor generates a write for a block that is cached by any of the others, all of the others will invalidate their copies. Thus, multiple simultaneous readers of a block may coexist without generating bus transactions or invalidations, but a write will eliminate all other cached copies.

To see how this simple write-through invalidation protocol provides coherence, we need to show that for any execution under the protocol a total order on the mem-

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.