

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Mikhail Kovalev Silvia M. Müller
Wolfgang J. Paul (Eds.)

A Pipelined Multi-core MIPS Machine

Hardware Implementation
and Correctness Proof

Volume Editors

Mikhail Kovalev
Wolfgang J. Paul
Saarland University, Saarbrücken, Germany
E-mail: {kovalev,wjp}@wjpsrver.cs.uni-saarland.de

Silvia M. Müller
IBM-Labor Böblingen, Böblingen, Germany
E-mail: smm@de.ibm.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-13905-0 e-ISBN 978-3-319-13906-7
DOI 10.1007/978-3-319-13906-7
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: Applied for

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Markus Richter, Heidelberg

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book is based on the third author's lectures on computer architecture, as given in the summer semester of 2013 at Saarland University. It contains a gate level construction of a multi-core machine with pipelined MIPS processor cores and a sequentially consistent shared memory. This opens the way to the formal verification of synthesizable hardware for multi-core processors in the future.

We proceed in three steps: i) we review pipelined single core processor constructions and their correctness proofs, ii) we present a construction of a cache-based shared memory which is kept consistent by the MOESI protocol and show that it is sequentially consistent, and iii) we integrate the pipelined processor cores into the shared memory and show that the resulting hardware simulates the steps of an abstract multi-core MIPS machine in some order. In the last step the reference machine consists of MIPS cores and a single memory, where the cores, together with the shared memory, execute instructions in a nondeterministic order.

In the correctness proofs of the last two steps a new issue arises. Constructions are in a gate level hardware model and thus deterministic. In contrast the reference models against which correctness is shown are nondeterministic. The development of the additional machinery for these proofs and the correctness proof of the shared memory at the gate level are the main technical contributions of this work.

October 2014

Mikhail Kovalev
Silvia M. Müller
Wolfgang J. Paul

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Overview	5
2	Number Formats and Boolean Algebra	7
2.1	Basics	8
2.1.1	Numbers, Sets, and Logical Connectives	8
2.1.2	Sequences and Bit-Strings	9
2.2	Modulo Computation	12
2.3	Geometric Sums	14
2.4	Binary Numbers	15
2.5	Two's Complement Numbers	18
2.6	Boolean Algebra	20
2.6.1	Identities	23
2.6.2	Solving Equations	25
2.6.3	Disjunctive Normal Form	26
3	Hardware	29
3.1	Digital Gates and Circuits	30
3.2	Some Basic Circuits	33
3.3	Clocked Circuits	41
3.3.1	Digital Clocked Circuits	41
3.3.2	The Detailed Hardware Model	44
3.3.3	Timing Analysis	49
3.4	Registers	54
3.5	Drivers and Main Memory	54
3.5.1	Open Collector Drivers and Active Low Signal	55
3.5.2	Tristate Drivers and Bus Contention	56
3.5.3	The Incomplete Digital Model for Drivers	60
3.5.4	Self Destructing Hardware	61
3.5.5	Clean Operation of Tristate Buses	64

3.5.6	Specification of Main Memory	69
3.5.7	Operation of Main Memory via a Tristate Bus	72
3.6	Finite State Transducers	75
3.6.1	Realization of Moore Automata	76
3.6.2	Precomputing Outputs of Moore Automata	78
3.6.3	Realization of Mealy Automata	80
3.6.4	Precomputing Outputs of Mealy Automata	81
4	Nine Shades of RAM	83
4.1	Basic Random Access Memory	83
4.2	Single-Port RAM Designs	85
4.2.1	Read Only Memory (ROM)	85
4.2.2	Multi-bank RAM	86
4.2.3	Cache State RAM	89
4.2.4	SPR RAM	90
4.3	Multi-port RAM Designs	92
4.3.1	3-port RAM for General Purpose Registers	92
4.3.2	General 2-port RAM	94
4.3.3	2-port Multi-bank RAM-ROM	95
4.3.4	2-port Cache State RAM	97
5	Arithmetic Circuits	99
5.1	Adder and Incrementer	99
5.2	Arithmetic Unit	101
5.3	Arithmetic Logic Unit (ALU)	106
5.4	Shift Unit	108
5.5	Branch Condition Evaluation Unit	113
6	A Basic Sequential MIPS Machine	117
6.1	Tables	118
6.1.1	I-type	118
6.1.2	R-type	119
6.1.3	J-type	119
6.2	MIPS ISA	120
6.2.1	Configuration and Instruction Fields	120
6.2.2	Instruction Decoding	123
6.2.3	ALU-Operations	124
6.2.4	Shift Unit Operations	126
6.2.5	Branch and Jump	127
6.2.6	Sequences of Consecutive Memory Bytes	129
6.2.7	Loads and Stores	130
6.2.8	ISA Summary	132
6.3	A Sequential Processor Design	133
6.3.1	Software Conditions	133
6.3.2	Hardware Configurations and Computations	134

6.3.3	Memory Embedding	135
6.3.4	Defining Correctness for the Processor Design	138
6.3.5	Stages of Instruction Execution	140
6.3.6	Initialization	141
6.3.7	Instruction Fetch	142
6.3.8	Instruction Decoder	143
6.3.9	Reading from General Purpose Registers	147
6.3.10	Next PC Environment	147
6.3.11	ALU Environment	150
6.3.12	Shift Unit Environment	151
6.3.13	Jump and Link	152
6.3.14	Collecting Results	153
6.3.15	Effective Address	153
6.3.16	Shift for Store Environment	154
6.3.17	Memory Stage	156
6.3.18	Shifter for Load	158
6.3.19	Writing to the General Purpose Register File	159
7	Pipelining	161
7.1	MIPS ISA and Basic Implementation Revisited	162
7.1.1	Delayed PC	162
7.1.2	Implementing the Delayed PC	163
7.1.3	Pipeline Stages and Visible Registers	164
7.2	Basic Pipelined Processor Design	167
7.2.1	Transforming the Sequential Design	167
7.2.2	Scheduling Functions	172
7.2.3	Use of Invisible Registers	175
7.2.4	Software Condition <i>SC-1</i>	176
7.2.5	Correctness Statement	177
7.2.6	Correctness Proof of the Basic Pipelined Design	178
7.3	Forwarding	190
7.3.1	Hits	190
7.3.2	Forwarding Circuits	190
7.3.3	Software Condition <i>SC-2</i>	191
7.3.4	Scheduling Functions Revisited	192
7.3.5	Correctness Proof	193
7.4	Stalling	196
7.4.1	Stall Engine	196
7.4.2	Hazard Signals	197
7.4.3	Correctness Statement	198
7.4.4	Scheduling Functions	198
7.4.5	Correctness Proof	203
7.4.6	Liveness	205

8	Caches and Shared Memory	207
8.1	Concrete and Abstract Caches	209
8.1.1	Abstract Caches and Cache Coherence	210
8.1.2	Direct Mapped Caches	212
8.1.3	k -way Associative Caches	214
8.1.4	Fully Associative Caches	216
8.2	Notation	219
8.2.1	Parameters	219
8.2.2	Memory and Memory Systems	219
8.2.3	Accesses and Access Sequences	220
8.2.4	Sequential Memory Semantics	221
8.2.5	Sequentially Consistent Memory Systems	222
8.2.6	Memory System Hardware Configurations	223
8.3	Atomic MOESI Protocol	224
8.3.1	Invariants	224
8.3.2	Defining the Protocol by Tables	226
8.3.3	Translating the Tables into Switching Functions	228
8.3.4	Algebraic Specification	230
8.3.5	Properties of the Atomic Protocol	234
8.4	Gate Level Design of a Shared Memory System	235
8.4.1	Specification of Interfaces	236
8.4.2	Data Paths of Caches	240
8.4.3	Cache Protocol Automata	247
8.4.4	Automata Transitions and Control Signals	249
8.4.5	Bus Arbiter	257
8.4.6	Initialization	260
8.5	Correctness Proof	261
8.5.1	Arbitration	261
8.5.2	Silent Slaves and Silent Masters	263
8.5.3	Automata Synchronization	264
8.5.4	Control of Tristate Drivers	269
8.5.5	Protocol Data Transmission	274
8.5.6	Data Transmission	277
8.5.7	Accesses of the Hardware Computation	279
8.5.8	Relation with the Atomic Protocol	301
8.5.9	Ordering Hardware Accesses Sequentially	305
8.5.10	Sequential Consistency	308
8.5.11	Liveness	310
9	A Multi-core Processor	311
9.1	Compare-and-Swap Instruction	312
9.1.1	Introducing CAS to the ISA	312
9.1.2	Introducing CAS to the Sequential Processor	313
9.2	Multi-core ISA and Reference Implementation	317
9.2.1	Multi-core ISA Specification	317

9.2.2	Sequential Reference Implementation	318
9.2.3	Simulation Relation	321
9.2.4	Local Configurations and Computations	323
9.2.5	Accesses of the Reference Computation	325
9.3	Shared Memory in the Multi-core System	326
9.3.1	Notation	326
9.3.2	Invisible Registers and Hazard Signals	327
9.3.3	Connecting Interfaces	329
9.3.4	Stability of Inputs of Accesses	330
9.3.5	Relating Update Enable Signals and Ends of Accesses ..	331
9.3.6	Scheduling Functions	334
9.3.7	Stepping Function	334
9.3.8	Correctness Proof	336
9.3.9	Liveness	341
References		345
Index		347