

Founding Editors

Gerhard Goos

Karlsruhe Institute of Technology, Karlsruhe, Germany

Juris Hartmanis

Cornell University, Ithaca, NY, USA

Editorial Board Members

Elisa Bertino

Purdue University, West Lafayette, IN, USA

Wen Gao

Peking University, Beijing, China

Bernhard Steffen 

TU Dortmund University, Dortmund, Germany

Gerhard Woeginger 

RWTH Aachen, Aachen, Germany

Moti Yung

Columbia University, New York, NY, USA

More information about this series at <http://www.springer.com/series/7407>

Petro Lutsyk · Jonas Oberhauser · Wolfgang J. Paul

A Pipelined Multi-Core Machine with Operating System Support

Hardware Implementation and Correctness Proof

Authors

Petro Lutsyk
Saarland University
Saarbrücken, Germany

Jonas Oberhauser
Saarland University
Saarbrücken, Germany

Wolfgang J. Paul
Saarland University
Saarbrücken, Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-43242-3 ISBN 978-3-030-43243-0 (eBook)
<https://doi.org/10.1007/978-3-030-43243-0>

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2020

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.

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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Building on previous work, we present at the gate level construction and correctness proof of a multi-core machine with pipelined processors and extensive operating system support. More precisely, the machine under consideration has the following features:

- MIPS instruction set architecture (ISA) for application and for system programming
- cache coherent memory system
- store buffers in front of the data caches
- interrupts and exceptions
- memory management units (MMUs)
- pipelined processors: the classical five-stage pipeline is extended by two pipeline stages for address translation
- local interrupt controller (ICs) supporting inter-processor interrupts (IPIs)
- I/O-interrupt controller and a disk

An overall correctness proof for such a machine poses several challenges, which are overcome with the following technical contributions:

- a new pipeline control which allows rollbacks of instructions from multiple pipeline stages while allowing to complete memory accesses that have already started
- a new induction scheme which permits to treat all rollbacks in such a pipeline in a uniform way
- modeling of ISA by a nondeterministic transition system with guard conditions
- proof technology for exploiting software conditions in such a machine, which only have to hold if the ISA computation abstracted from the hardware computation obeys the guard conditions

February 2020

Petro Lutsyk
Jonas Oberhauser
Wolfgang J. Paul

Contents

1	Introductory Material	1
1.1	So What?	2
1.1.1	The Element of Chance in Multi-Core Programming	2
1.1.2	Standards and Verification in Classical Architecture versus System Architecture	2
1.2	Overview	4
1.3	Basics	17
1.3.1	Sets, Cross Products and Sequences	17
1.3.2	Boolean Operators	22
1.3.3	Binary and Two's Complement Numbers	22
1.3.4	Memory	24
2	On Hierarchical Hardware Design	29
2.1	Syntax	30
2.1.1	Buses	30
2.1.2	I/O Buses	31
2.1.3	Basic Hardware Units	35
2.1.4	Subunit Declarations	36
2.1.5	Composite Hardware Units	37
2.1.6	Implicit Labels and Types in Hardware Schematics	40
2.1.7	Size Parameters	42
2.1.8	Recursive Constructions	44
2.1.9	Global Naming	46
2.1.10	Paths, Cycles and Depth	49
2.2	Semantics	54
2.2.1	Values of Input Signals	54
2.2.2	Configurations	56
2.2.3	Hardware Computations	60
2.2.4	Circuit Evaluation	61
2.2.5	Values of Buses and Outputs of Units	63
2.2.6	Updating Configurations	64
2.2.7	Streamlining Notation	65
2.3	Modular Reasoning	67
2.3.1	Tracking Paths	68

2.3.2	Evaluating Circuits	71
2.3.3	The Easy Common Case	76
2.3.4	Cycles of Buses Between Units	78
2.3.5	Updating Configurations	81
2.4	Expressions and Assignments as Syntactic Sugar	82
2.4.1	Syntax	82
2.4.2	Implementation	85
2.4.3	Semantics	87
3	Hardware Library	91
3.1	Circuit Library	91
3.1.1	Basic Circuits	91
3.1.2	Arithmetic Circuits	102
3.1.3	Branch Condition Evaluation Unit	106
3.2	Control Automata	106
3.3	Random Access Memory (RAM)	108
3.3.1	Basic Design	108
3.3.2	Read Only Memory (ROM)	110
3.3.3	R-W-RAM	111
3.3.4	R-RW-RAM	112
3.3.5	Two port ROM	113
3.3.6	bw-R-RW-RAM	113
3.3.7	bw-R-RW-RAM-ROM	115
3.3.8	GPR-RAM	117
3.3.9	SPR-RAM	118
3.3.10	R-RW-SPR RAM	121
3.4	Register Control	122
3.4.1	Set-Clear Flip-Flop	122
3.4.2	Stabilizer Latch	123
3.5	Control of Tri-State Buses	123
3.5.1	Justifying the Operating Conditions	123
3.5.2	Controlling Tri-State Buses	126
3.5.3	Arbitration	127
3.5.4	Combined Arbiter and Tri-State Bus	130
3.6	Exercises	134
4	Basic Processor Design	137
4.1	MIPS ISA	138
4.1.1	Instruction Tables	138
4.1.2	Configuration and Instruction Fields	141
4.1.3	Instruction Decoding	143
4.1.4	Reading out Data from Register Files	144
4.1.5	Moves	145
4.1.6	ALU-operations	145
4.1.7	Shift Unit Operations	146

4.1.8	Branch and Jump	147
4.1.9	Memory Operations	149
4.1.10	ISA Summary	151
4.1.11	Software Conditions	152
4.2	A Sequential Processor Design	152
4.2.1	Hardware Configurations and Simulation Relation	152
4.2.2	Memory Embedding	153
4.2.3	Overview of the Hardware	156
4.2.4	Initialization and Instruction Fetch	157
4.2.5	Straight Forward Constructions	158
4.2.6	Data Accesses	166
4.2.7	Absence of Cycles	173
4.3	Repackaging the Induction Step	174
4.3.1	Numbers and Correctness of Circuit Stages	175
4.3.2	Use of Signals for Instruction Execution	178
5	Pipelining	183
5.1	General Concepts	184
5.1.1	ISA, Circuit Stages and Software Conditions	184
5.1.2	Cost Effectiveness of Pipelining	186
5.1.3	Notation	187
5.2	Basic Pipelined Processor	189
5.2.1	Pipeline Registers	189
5.2.2	Trivial Stall Engine	192
5.2.3	Scheduling Functions	194
5.2.4	Delayed PC	197
5.2.5	Correctness Statement and Additional Software Condition	198
5.2.6	Intuition	199
5.2.7	Software Conditions	200
5.2.8	Correctness Proof	200
5.3	Forwarding	206
5.3.1	Forwarding Circuits	206
5.3.2	Correctness	207
5.4	Stalling	209
5.4.1	Stall Engine with Hazard Signals	209
5.4.2	Correctness and Liveness	212
5.4.3	Proof Summary	213
5.5	Final Remark: Getting Rid of Software Conditions	214
6	Cache Memory Systems	217
6.1	Reviewing A Concrete Example	218
6.1.1	Construction	218
6.1.2	Memory Abstraction	219
6.2	Specification	220
6.2.1	Signals of the Processor Interface	221

6.2.2	Protocol	221
6.2.3	Operating Conditions for the CMS	222
6.2.4	External Access Sequence	223
6.2.5	Sequential Consistency	224
6.2.6	Ordering External Accesses by their End Cycle	225
6.2.7	Liveness	227
6.3	Using a Cache Memory System in a Single Pipeline	227
6.3.1	Connecting Interfaces	227
6.3.2	Stability of Inputs of Accesses	228
6.3.3	Relating Update Enable Signals and Ends of Accesses	228
6.3.4	Software Conditions and Main Induction Hypothesis	229
6.3.5	External Accesses to the Data Cache Ending in the Current Cycle	232
6.3.6	Induction Step	232
6.3.7	Proof Summary	236
6.4	Exercises	236
6.5	Errata	239
7	Interrupt Mechanism	243
7.1	Specification	244
7.1.1	Special Purpose Registers Revisited	244
7.1.2	Types of Interrupts	245
7.1.3	MIPS ISA with Interrupts	246
7.1.4	Specification of Most Internal Interrupt Event Signals	249
7.1.5	Instruction and Data Accesses Revisited	250
7.2	Sequential Implementation	251
7.2.1	PC Environment	252
7.2.2	Cause Processing Environment	252
7.2.3	Accesses and Write Signals	253
7.2.4	Special Purpose Register File	254
7.2.5	Software Conditions and Correctness	254
7.3	Pipelining a Processor With Interrupts	258
7.4	Stalling, Rollback and Pipe Drain	259
7.4.1	Stall Engine and Scheduling Functions with Rollbacks and Stabilized Full Bits	261
7.4.2	Properties of the New Stall Engine	266
7.4.3	Liveness and Correctness in the Presence of Rollback	274
7.4.4	Pipelining Signals	282
7.4.5	Sampling the External Interrupt Signal	283
7.5	Pipelining and Forwarding	285
7.5.1	Data Paths	285
7.5.2	Forwarding Data	287
7.5.3	Connecting the new Rollback Engine	288
7.5.4	Program Counter Environment	292
7.5.5	Write and Clock Enable Signals	293

7.5.6	Connecting the Cache Memory System	293
7.5.7	Stability of the Instruction Memory Address	294
7.6	Stating Processor Correctness	296
7.6.1	Software Condition for Self Modifying Code	296
7.6.2	A New Induction Hypothesis	296
7.7	Properties of Liveness	298
7.7.1	Properties needed for Correctness	299
7.7.2	Towards the Liveness Proof	302
7.8	Proving Processor Correctness	308
7.8.1	Induction Scheme	308
7.8.2	Induction Step of the Inner Induction	311
7.8.3	Correctness of Memory Update	311
7.8.4	Pipeline Correctness with Rollback Request	312
7.8.5	Pipeline Correctness without Rollback Request	314
7.9	Completing the Liveness Proof	324
7.10	Proof Summary	326
7.11	Exercises	329
8	Self Modification, Instruction Buffer and Nondeterministic ISA	331
8.1	ISA	332
8.1.1	Nondeterminism, Oracle Inputs and Stepping Functions	332
8.1.2	Guard Conditions and Software Conditions	334
8.1.3	Configurations, Instruction Buffer and Guard Conditions	335
8.1.4	Instruction Execution	336
8.2	Hardware correctness	339
8.2.1	Inputs for Processor Steps	339
8.2.2	Stepping Function	340
8.2.3	Software Condition and Stating Processor Correctness	344
8.2.4	Properties of <i>pseq</i>	348
8.2.5	Induction Step	351
8.2.6	Correctness of Memory Update	356
8.2.7	Pipeline Correctness	357
8.3	Liveness and Guard condition	362
8.4	A Simple Instruction Buffer Reduction Theorem	363
8.5	Summary	367
8.5.1	ISA	367
8.5.2	Specifying Processor Correctness	367
8.5.3	Correctness Proof	368
8.6	Exercises	369
9	Memory Management Units	371
9.1	Walks and ISA MMU	374
9.1.1	Page Tables	374
9.1.2	Walks	375
9.1.3	TLBs	378

9.1.4	Translation Requests	378
9.2	ISA	379
9.2.1	Configuration	379
9.2.2	TLB Steps	380
9.2.3	Instruction Buffer Steps	381
9.2.4	Processor Steps	382
9.3	MMU Construction	386
9.3.1	TLB	386
9.3.2	Walking Unit and MMU	390
9.3.3	Correctness	395
9.4	Pipelined Processor with MMUs	404
9.4.1	Instruction Address	406
9.4.2	Connecting MMUs to the Pipeline	406
9.4.3	Caches	408
9.4.4	Forwarding	410
9.4.5	PC Environment	411
9.4.6	Interrupts and Cause Pipe	412
9.4.7	Stall Engine	413
9.5	Correctness	414
9.5.1	Stability of Inputs of Accesses	415
9.5.2	Sequential Order and Local Sequential Index of CMS steps	416
9.5.3	Relating Update Enable Signals and Ends of Accesses	416
9.5.4	Inputs for Processor Steps	417
9.5.5	Stepping Function	417
9.5.6	Induction Hypothesis	420
9.5.7	Correctness for Components Outside the Pipeline	421
9.5.8	Correctness for Pipeline Registers	427
9.5.9	Proof Summary	430
10	Store Buffers	433
10.1	Specification of Store Buffers	435
10.2	Extending ISA	438
10.2.1	Processor Steps	439
10.2.2	Store Buffer Steps	440
10.2.3	Instruction Buffer and MMU Steps	441
10.3	Store Buffer Construction	441
10.3.1	Basic Queue with Forwarding	441
10.3.2	Improved Queue Design	449
10.4	Pipelined Processor with Store Buffer	451
10.4.1	Draining Instructions	452
10.4.2	Arbitration and Control	453
10.4.3	Connecting the Data Paths	456
10.4.4	Computing the Busy Signal	459
10.4.5	Stability of Inputs	460
10.5	Correctness	460

10.5.1	Definition of Stepping Function	460
10.5.2	Simulation Relation	463
10.5.3	Managing the Induction Step	464
10.5.4	Induction Step	466
10.5.5	Preparations	467
10.5.6	Correctness for Components Outside the Pipeline	476
10.5.7	Correctness for Pipeline Registers	478
10.5.8	Proof Summary	480
10.6	A Simple Store Buffer Reduction Theorem	481
10.7	Exercises	486
11	Multi-Core Processors	489
11.1	Defining Basic Multi-Core ISA	489
11.2	Local ISA Configurations and Computations	491
11.3	Hardware	492
11.3.1	Connecting Processors to the Cache Memory System	492
11.4	Correctness	493
11.4.1	Straight Forward Generalizations of Arguments for Single Core Processors	493
11.4.2	Liveness and Committed	495
11.4.3	Sampling Signals from Outside the Pipeline	495
11.4.4	Ordering Instructions by the Cycle they Pass The Memory Stage	495
11.4.5	Software Conditions and Correctness Statement	497
11.4.6	Induction Step	499
11.4.7	Numbering Accesses to Data Ports	499
11.5	Instruction Buffer Reduction	502
11.6	Exercises	503
12	Advanced Programmable Interrupt Controllers (APICs)	505
12.1	Specification	507
12.1.1	Processor Steps	510
12.1.2	Store Buffer Steps	512
12.1.3	MMU and Instruction Buffer Steps	513
12.1.4	SIPI Steps	513
12.1.5	IPI Delivery Steps	513
12.1.6	Initial Configuration	515
12.2	Construction	515
12.2.1	Interrupt Bus	515
12.2.2	Interface of Local APIC	516
12.2.3	Construction of Local APIC	518
12.2.4	Placing the APICs on the Interrupt Bus	523
12.2.5	Cause Pipe and Stalling	523
12.2.6	Memory Operations to the APIC	525
12.3	Correctness	527

12.3.1	Stepping Function	527
12.3.2	Software Conditions	531
12.3.3	Induction Hypothesis	532
12.3.4	Scheduling Diagram for APIC	534
12.3.5	Induction Step	535
12.3.6	Correctness of Store Buffers	538
12.3.7	Validity	540
12.3.8	Correctness of ICR	541
12.3.9	Correctness of IRR	547
12.3.10	Correctness of ISR	553
12.3.11	Remaining Registers	557
12.3.12	Correctness of Components outside the Pipeline	558
12.3.13	Correctness of Pipeline	560
12.3.14	Summary	563
12.4	Exercises	563
13	Adding a Disk	565
13.1	Hardware Disk as a Basic Unit	566
13.2	Connecting the Disk with the Processors	570
13.2.1	Operating Conditions of the Device Bus	573
13.3	ISA	574
13.4	Correctness	575
13.4.1	Stepping Function	575
13.4.2	Software Condition for Disk	577
13.4.3	Induction Hypothesis	578
13.4.4	Induction Step	579
13.4.5	Validity	580
13.4.6	Operating Condition of Disk	581
13.4.7	Disk Correctness	584
13.4.8	Correctness of Non-Pipelined Components	586
13.4.9	Correctness of the Pipeline	586
13.5	Exercises	588
14	I/O APIC	591
14.1	Specification of I/O APIC	592
14.1.1	I/O APIC Steps	593
14.1.2	EOI Mechanism and the Local APIC	594
14.2	Construction	595
14.2.1	Local APIC Revisited	596
14.2.2	I/O APIC	600
14.2.3	Connecting Everything	604
14.3	Correctness Proof	606
14.3.1	Stepping Function	606
14.3.2	Software Conditions	607
14.3.3	Induction Hypothesis	608

14.3.4	Induction Step	609
14.3.5	Validity	610
14.3.6	Correctness of the EOI Pending Register	613
14.3.7	Correctness of the Redirection Register	615
14.3.8	Correctness of Non-Pipelined Components	621
14.3.9	Correctness of Pipelined Registers	623
14.4	Exercises	624

References	627
-------------------	------------