

High-Level Verification

Sudipta Kundu · Sorin Lerner · Rajesh K. Gupta

High-Level Verification

Methods and Tools for Verification
of System-Level Designs

With Chapter 6 contributed by Malay K. Ganai
and Chapter 8 contributed by Zachary Tatlock

Sudipta Kundu
Synopsis Inc.
NW Thorncroft Dr. 2174
97124 Hillsboro
USA
sudiptakundu@gmail.com

Rajesh K. Gupta
Department of Computer Science
and Engineering
University of California, San Diego
Gillman Drive 9500
92093-0404 La Jolla
USA
rgupta@ucsd.edu

Sorin Lerner
Department of Computer Science
and Engineering
University of California, San Diego
Gillman Drive 9500
92093-0404 La Jolla
USA
lerner@ucsd.edu

ISBN 978-1-4419-9358-8 e-ISBN 978-1-4419-9359-5
DOI 10.1007/978-1-4419-9359-5
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011928550

© Springer Science+Business Media, LLC 2011

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

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

Preface

Given the growing size and heterogeneity of Systems on Chip (SOC), the design process from initial specification to chip fabrication has become increasingly complex. The growing complexity provides incentive for designers to use high-level languages such as C, SystemC, and SystemVerilog for system-level design. While a major goal of these high-level languages is to enable verification at a higher level of abstraction, allowing early exploration of system-level designs, the focus so far has been on traditional testing techniques such as random testing and scenario-based testing.

This book focuses on the rapidly growing area of high-level verification. We envision a design methodology that relies upon advances in synthesis techniques as well as on incremental refinement of the design process. These refinements can be done manually or through elaboration tools. In this book, we discuss verification of specific properties in designs written using high-level languages as well as checking that the refined implementations are equivalent to their high-level specifications. The novelty of each of these techniques is that they use a combination of formal techniques to do scalable verification of system designs completely automatically.

The verification techniques fall into two categories: (a) methods for verifying properties of high-level designs and (b) methods for verifying that the translation from high-level design to a low-level Register Transfer Language (RTL) design preserves semantics. Taken together, these two parts guarantee that properties verified in the high-level design are preserved through the translation to low-level RTL. By performing verification on the high-level design, where the design description is smaller in size and the design intent information is easier to extract, and then checking that all refinement steps are correct, we describe a hardware development methodology that provides strong and expressive guarantees that are difficult to achieve by directly analyzing the low-level RTL code.

We expect the reader to gain appreciation and knowledge of the recent advances in raising the abstraction for design verification tasks. While the complexity of the problem is not lost on a typical reader, this book would ultimately present a positive outlook on the engineering solutions to the problem that the reader can use in practice.

Acknowledgments

Acknowledgment is not a mere formality but a genuine opportunity to express the sincere thanks to all those; without whose active support and encouragement this book would not have been successful.

This book would not have been possible without contribution from Dr. Malay K. Ganai and Mr. Zachary Tatlock. Dr. Ganai shared his expertise on bounded model checking in Chap. 6 and Mr. Tatlock shared his knowledge on once-and-for-all verification techniques in Chap. 8.

Significant part of this book is based on Dr. Sudipta Kundu's doctoral dissertation work. To this end, we express our thanks to Prof. Ranjit Jhala, Prof. Ingolf Krueger, Prof. Bill Lin, Dr. Yuvraj Agarwal, Federic Doucet, Ross Tate, and Pat Rondon and to all the faculty members, staffs, and graduate students of the Department of Computer Science and Engineering for their continuous help and support.

Our sincere thanks to Chao Wang, Nishant Sinha, Aarti Gupta, and all other members of the System LSI and Software Verification group at NEC Laboratories America for uncountably many interesting discussion and explanation of key concepts in Verification.

We thank Prof. Alan J. Hu for his valuable insights and comments on the initial version of the translation validation part of this book.

We also thank the members of the MESL and Progsys group for their ceaseless effort, constant encouragement, and also for the endless good times during the making of this book.

We also agree that words are not enough to express our indebtedness and gratitude toward our families, to whom we owe every success and achievements of our life. Their constant support and encouragement under all odds has brought us where we stand today.

Contents

1	Introduction	1
1.1	Overview of High-Level Verification	2
1.2	Overview of Techniques Covered in this Book	4
1.2.1	High-Level Property Checking	4
1.2.2	Translation Validation	5
1.2.3	Synthesis Tool Verification	6
1.3	Contributions of the Book	7
1.4	Book Organization	8
2	Background	11
2.1	High-Level Design	11
2.2	RTL Design	12
2.3	High-Level Synthesis	12
2.4	Model Checking	15
2.4.1	Simple Elevator Example	16
2.4.2	Property Specification	18
2.4.3	Reachability Algorithm	19
2.5	Concurrent Programs	20
2.5.1	Representation of Concurrent Programs	20
2.5.2	Partial-Order Reduction	21
2.6	Summary	23
3	Related Work	25
3.1	High-Level Property Checking	25
3.1.1	Explicit Model Checking	25
3.1.2	Symbolic Model Checking	27
3.2	Translation Validation	29
3.2.1	Relational Approach	29
3.2.2	Model Checking	30
3.2.3	Theorem Proving	31

3.3	Synthesis Tool Verification	31
3.3.1	Formal Assertions	32
3.3.2	Transformational Synthesis Tools	33
3.3.3	Witness Generator	33
3.4	Summary	35
4	Verification Using Automated Theorem Provers	37
4.1	Satisfiability Modulo Theories	38
4.2	Hoare Logic	39
4.3	Weakest Preconditions	40
4.4	Additional Complexities for Realistic Programs	44
4.4.1	Path-Based Weakest Precondition	44
4.4.2	Pointers	46
4.4.3	Loops	49
5	Execution-Based Model Checking for High-Level Designs	51
5.1	Verification of Concurrent Programs	51
5.2	Overview of SystemC	52
5.3	Problem Statement	52
5.4	Overview of Execution-Based MC for SystemC Designs	52
5.5	SystemC Example	53
5.6	SystemC Simulation Kernel	55
5.6.1	Nondeterminism	55
5.7	State Transition System	56
5.8	The EMC-SC Approach	58
5.8.1	Static Analysis	59
5.8.2	The Explore Algorithm	60
5.9	The Satya Tool	63
5.10	Experiments and Results	64
5.10.1	FIFO Benchmark	64
5.10.2	TAC Benchmark	65
5.11	Further Reading	65
5.12	Summary	66
6	Bounded Model Checking for Concurrent Systems:	
	Synchronous Vs. Asynchronous	67
6.1	Introduction	67
6.1.1	Synchronous Models	69
6.1.2	Asynchronous Models	70
6.1.3	Outline	71
6.2	Concurrent System	72
6.2.1	Interleaving (Operational) Semantics	72
6.2.2	Axiomatic (Non-Operational) Semantics	74
6.2.3	Partial Order	74

6.3	Bounded Model Checking	75
6.4	Concurrent System: Model	76
6.5	Synchronous Modeling	77
6.6	BMC on Synchronous Models	79
6.6.1	BMC Formula Sizes	81
6.7	Asynchronous Modeling	81
6.8	BMC on Asynchronous Models: CSSA-Based Approach	83
6.8.1	Thread Program Constraints: Ω_{TP}	83
6.8.2	Concurrency Constraints: Ω_{CC}	83
6.8.3	BMC Formula Sizes	84
6.9	BMC on Asynchronous Models: Token-Based Approach	84
6.9.1	MAT-Based Partial Order Reduction	86
6.9.2	Independent Modeling	89
6.9.3	Concurrency Constraints	90
6.9.4	BMC Formula Sizes	91
6.10	Comparison Summary	92
6.11	Further Reading	93
6.12	Summary	93
7	Translation Validation of High-Level Synthesis	97
7.1	Overview of Translation Validation	97
7.2	Overview of the TV-HLS Approach	98
7.3	Illustrative Example	99
7.3.1	Translation Validation Approach	101
7.3.2	Simulation Relation	101
7.3.3	Checking Algorithm	103
7.3.4	Inference Algorithm	103
7.4	Definition of Refinement	106
7.5	Simulation Relation	108
7.6	The Translation Validation Algorithm	109
7.6.1	Checking Algorithm	109
7.6.2	Inference Algorithm	112
7.7	Equivalence of Transition Diagrams	115
7.8	Experiments and Results	116
7.8.1	Automatic Refinement Checking of CSP Programs	116
7.8.2	SPARK: High-Level Synthesis Framework	118
7.9	Further Reading	120
7.10	Summary	121
8	Parameterized Program Equivalence Checking	123
8.1	Overview of Synthesis Tool Verification	123
8.1.1	Once-And-For-All Vs. Translation Validation	124
8.2	Overview of the PEC Approach	124

8.3	Illustrative Example	125
8.3.1	Expressing Loop Pipelining	126
8.3.2	Parameterized Programs	126
8.3.3	Side Conditions	127
8.3.4	Executing Optimizations	127
8.3.5	Proving Correctness of Loop Pipelining	128
8.3.6	Parameterized Equivalence Checking	128
8.3.7	Bisimulation Relation	128
8.3.8	Generating Constraints	130
8.3.9	Solving Constraints	131
8.4	Parameterized Equivalence Checking	132
8.4.1	Bisimulation Relation	133
8.4.2	Architectural Overview	133
8.5	GenerateConstraints Module	134
8.6	SolveConstraints Module	137
8.7	Permute Module	137
8.8	Experiments and Results	140
8.9	Execution Engine	142
8.10	Further Reading	143
8.11	Summary	144
9	Conclusions and Future Work	147
9.1	High-Level Property Checking	147
9.2	Translation Validation	148
9.3	Synthesis Tool Verification	148
9.4	Future Work	148
	References	151
	Index	163

Acronyms

ATP	Automated Theorem Prover
BDD	Binary Decision Diagram
BMC	Bounded Model Checking
CCFG	Concurrent Control Flow Graph
CDFG	Control Data Flow Graph
CEGAR	Counter Example Guided Abstraction Refinement
CFG	Control Flow Graph
CSG	Conflict Sub-Graph
CSP	Communicating Sequential Processes
CSSA	Concurrent Static Single Assignment
CTL	Computation Tree Logic
FIFO	First In First Out
FSM	Finite State Machine
FSMD	Finite State Machine with Datapath
GALS	Globally Asynchronous Locally Synchronous
HDL	Hardware Description Language
HLD	High-Level Design
HLS	High-Level Synthesis
HLV	High-Level Verification
HTG	Hierarchical Task Graph
ICs	Integrated Circuits
LLS	Language of Labeled Segments
LTL	Linear Temporal Logic
MAT	Mutually Atomic Transaction
MC	Model Checking
OSCI	Open SystemC Initiative
PEC	Parameterized Equivalence Checking
POR	Partial-Order Reduction
RTL	Register Transfer Level
SAT	SATisfiability
SMC	Symbolic Model Checking
SMT	Satisfiability Modulo Theory
TLM	Transaction Level Modeling
TV	Translation Validation

