SYMBOLIC MODEL CHECKING

SYMBOLIC MODEL CHECKING

by

Kenneth L. McMillan
Carnegie Mellon University

Springer Science+Business Media, LLC

Library of Congress Cataloging-in-Publication Data

McMillan, Kenneth L.
Symbolic model checking / by Kenneth L. McMillan.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4613-6399-6 ISBN 978-1-4615-3190-6 (eBook)
DOI 10.1007/978-1-4615-3190-6
1. Electronic digital computers--Circuits--Design--Data processing. 2. Symbolic circuit analysis--Data processing.
3. Logic design--Data processing. I. Title.
TK7888.4.M43 1993
621.39'2--dc20 93-24859
CIP

Dedicated to the memory of William L. McMillan

CONTENTS

FOREWORD

Logical errors in sequential circuit designs and protocols are an important problem for hardware designers. Such errors can delay getting a new product on the market or cause the failure of a computer system that is already in use. The most widely used method for verifying such systems is based on extensive simulation and can easily miss significant errors when the number of possible states of the system is very large. Although there has been considerable research on the use of theorem provers, term-rewriting systems, and proof checkers to verify such systems, these techniques are time consuming and often require a great deal of manual intervention.

Over the past ten years, my research group at Carnegie Mellon University has developed an alternative approach to verification called *temporal logic model checking*. In this approach specifications are expressed in a propositional temporal logic, and circuit designs and protocols are modeled as state–transition systems. An efficient search procedure is used to determine automatically if the specifications are satisfied by the transition systems. The initial idea for model checking appeared in a paper that I wrote with my first graduate student, Allen Emerson, in 1981. The earliest example that we verified using this technique was a version of the *alternating bit protocol* with 251 states. Neither of us could have possibly predicted the size of the transition systems that are now routinely checked by this method.

The main disadvantage of the original model checking algorithm was the *state explosion problem* which occurred if the system being verified had many components that could make transitions in parallel. Because of this problem, many researchers in formal verification predicted that model checking would never be practical for large circuits and protocols. This view was reinforced by early implementations which could only handle transition systems with a few thousand states and, therefore, were only useful for verifying systems with a small number of components. As late as 1987, no one could verify transition systems with more than a million states by using model checking techniques.

During the last five years, the size of the transition systems that can be verified

by model checking techniques has increased dramatically. The initial breakthrough was made in the fall of 1987 by one of my graduate students, Ken McMillan, who realized that the explicit adjacency-list representation that we had been using for transition systems severely limited the size of the circuits and protocols that we could verify. By representing transition systems implicitly using Binary Decision Diagrams (BDDs), he was able to handle some examples that had more than 10^{20} states. McMillan made this observation independently of the work by Coudert and Madre on using BDDs for checking equivalence of deterministic finite state machines. In fact, he made the key observation during the first six weeks of his first year as a graduate student at CMU. Refinements of the BDD-based techniques by two other graduate students, Jerry Burch and David Long, have pushed the state count up to more than 10^{100}. By combining model checking with various abstraction techniques, my research group has been able to handle even larger circuits and protocols. In one example, we were able to verify a pipelined ALU with more than 10^{1300} states.

For this insight McMillan was a cowinner of the 1992 ACM Doctoral Dissertation Award. This book is his Ph.D. thesis. In my opinion, it is the best introduction to temporal logic model checking that has been written. The SMV model checker described in this thesis is based on a dataflow-oriented hardware description language which can be annotated by specifications expressed in the temporal logic CTL. The model checker extracts a transition system from a program in the SMV language and uses a BDD-based search algorithm to determine whether the system satisfies the CTL specifications. If the transition system does not satisfy some specification, the verifier will produce an execution trace that shows why the specification is false. These traces are particularly useful in finding and correcting errors in complex systems.

The SMV system has been distributed widely, and many circuits and protocols have now been verified with it. Perhaps, the most impressive example of its use is the verification of the cache coherence protocol described in the IEEE Futurebus+ standard (IEEE Standard 896.1–1991). Although development of the Futurebus+ cache coherence protocol began in 1988, all previous attempts to validate the protocol were based entirely on informal techniques. In the summer of 1992 David Long constructed a precise model of the protocol in the SMV language and then used McMillan's model checker to show that the resulting transition system satisfied a formal specification of cache coherence. He was able to find a number of previously undetected errors and potential errors in the design of the protocol. To the best of my knowledge this is the first time that an automatic verification tool has been used to find errors in an IEEE standard.

I believe that the examples in this thesis provide convincing evidence that SMV can be used to debug real industrial designs. Nevertheless, much work remains to be done to realize the full potential of this new verification technique. In addition to describing the most powerful model checking system that has been developed, McMillan's thesis provides an excellent introduction to these new areas of research. Certainly, the most important problem is to characterize the class of circuits that have small BDD representations and, therefore, can be verified by symbolic model checking techniques. McMillan gives what I think is the best characterization to date of such circuits. From the examples that we have considered already, it is clear that use of abstraction and induction will be crucial in reasoning about large systems in the future. McMillan's thesis illustrates how these techniques can be applied in a number of realistic examples. His thesis also demonstrates convincingly that the state explosion problem is reduced if the possible executions of a concurrent system are modeled by partially ordered event structures instead of by execution traces or computation trees with interleaving of events. Several other important research topics are discussed in addition to the three that I have listed. Because his thesis opens up so many exciting areas for future research, I am positive that it will continue to have a profound influence in research on formal verification methods in the years to come.

Edmund M. Clarke, Jr.

Carnegie Mellon University

PREFACE

This book is a revised edition of my doctoral thesis, submitted in 1992 to Carnegie Mellon Univerity. As such, it is chiefly a report of research in the area of automatic formal verification. However, I hope that it can also serve as an introduction to the area for those interested in applying the symbolic model checking technique to verify their own designs, or in evaluating the method with a view to deciding what verification tools might be integrated into their own design systems. Thus, I have not assumed familiarity with such topics as temporal logic or model checking (though certainly a more thorough treatment of these topics can be obtained from other sources). In addition, I have tried to keep the emphasis on the practical side, using realistic examples whenever possible. The most involved of these is the verification of the cache consistency protocols of the Encore Gigamax multiprocessor, to which a chapter is devoted. Those readers interesting in applying the methods will find chapters 1–5 to be of the greatest interest. On the other hand, chapters 6–9 are more oriented toward researchers in the field. These chapters cover research topics that are more open-ended, and less clear in their implications. They will also require, I think, a greater degree of patience and mathematical fortitude on the part of the reader in order to decipher them. Nonetheless, chapter 7 in particular, on induction, may be of interest to the practically minded, since it extends the treatment of the cache protocol example to cover systems of arbitrary size. I hope that the presentations in chapters 6–9 may stimulate others to extend the work.

The contents of the book can be summarized roughly as follows: The concept of formal verification and the state explosion problem are introduced in chapter 1. Very briefly, if a system can be modeled as a finite state machine, we can very efficiently prove properties about that system expressed in temporal logic by a technique called model checking, the subject of chapter 2. Unfortunately, finite state models of *concurrent* systems (*eg.*, computer hardware) grow exponentially in size as the number of components of the system increases. This is known widely as the *state explosion problem*, and has limited finite state verification methods to small systems.

Chapter 3 introduces a method of skirting this problem called *symbolic model checking*. This technique avoids constructing the finite state model by representing it symbolically as a Boolean formula. This allows efficient methods of manipulating Boolean algebra to be applied to the model checking process. In particular, a representation called Binary Decision Diagrams can be applied. A significant result from chapter 3 identifies a structural class of sequential circuits which can be represented for the purpose of model checking with Binary Decision Diagrams of low complexity. This is born out by experimental results on example circuits and systems, including the Encore cache protocol.

In chapter 4, a language is developed for describing sequential circuits and protocols at various levels of abstraction. This language has a synchronous dataflow semantics, but allows nondeterminism and supports interleaving processes with shared variables. A system called SMV can automatically verify programs in this language with respect to temporal logic formulas, using the symbolic model checking technique.

Chapter 5 describes the verification of the cache protocol, expressing it in the SMV language, detailing what properites can be proved, and explaining the performance of the model checking algorithm in terms of the theory developed in chapter 3. The symbolic model checking technique revealed subtle errors in this protocol, resulting from complex execution sequences that would occur with very low probability in random simulation runs. This highlights an important aspect of model checking – the ability to generate counterexamples.

Chapter 6 generalizes the symbolic model checking technique from the simple temoral logic described in chapter 2, to a much more expressive logic called the Mu-Calculus. Several application of the Mu-Calculus are described.

Chapter 7 develops a technique that uses model checking to prove properties of systems of arbitrary size (generated by simple rules). The proof is checked automatically, but the user must supply a special model called a *process invariant* to act as an inductive hypothesis. A process invariant is developed for the distributed cache protocol, allowing properties of systems with an arbitrary number of processors to be proved.

Chapter 8 deals with the question of computing the state equivalence relation between finite state machines, relying heavily on the material in chapter 5. Benchmark results using a variety of optimizations are presented.

Finally, in chapter 9, an alternative method is developed for avoiding the state explosion in the case of asynchronous control circuits. This technique is based

the unfolding of Petri nets, and is used to check for hazards in a distributed mutual exclusion circuit.

This book came about because of the influence, advice and assistance of a large number of people, whom I would like to thank. By far the most significant influence on the work is that of Ed Clarke, my advisor at CMU. His work is the foundation of almost everything in it. Ed got me started in the field by teaching me what I know about verification, and opening doors for me in the research community. His enthusiasm for my early, somewhat untutored efforts in the field gave me the confidence to pursue my own ideas. Another important influence on this work and its author is Bob Kurshan. In more than once instance, his insistence on solving a particular problem led to a general solution in an unexpected way.

The work described in this book also rests rather heavily on that of Randy Bryant. To a certain extent, the symbolic model checking technique resulted from the fortunate coincidence of my being at the same institution as Randy. Finding an application for this technique was also a more or less serendipitous occurrence, and for this I have to thank the people at Encore Computer Corporation, including Dyung Van Le, Jim Schwalbe and Drew Wilson, for taking an interest in formal verification, and generously allowing me to use their system as an application (and to publish it no less!).

I must also thank David Long and Jerry Burch, who contributed substantially to the ideas in this book. I owe special thanks to David, who is a font of information, and who helped me to solidify ideas in countless discussions, when no doubt he had more important things to do.

Thanks to Robert Brayton and the CAD group at the University of California at Berkeley, who gave my thesis a very careful and intelligent reading, and to Allan Fisher. It was always a pleasure to have discussions with Allan, and no doubt some of his insights can be found interspersed in these pages. Of course, I am indebted to all the good people who make the Carnegie Mellon School of Computer Science and its facilities work. CMU provides a unique environment for graduate students, and I consider myself privileged to have been a part of it.

Finally, to Tracy Slein, the one indispensible person in the whole process - I can't thank you enough.

SYMBOLIC MODEL CHECKING