Practical Design Verification

Improve design efficiency and reduce costs with this practical guide to formal and simulation-based functional verification. Giving you a theoretical and practical understanding of the key issues involved, expert authors explain both formal techniques (model checking and equivalence checking) and simulation-based techniques (coverage metrics and test generation). You get insights into practical issues including hardware verification languages (HVLs) and system-level debugging. The foundations of formal and simulation-based techniques are covered too, as are more recent research advances including transaction-level modeling and assertion-based verification, plus the theoretical underpinnings of verification, including the use of decision diagrams and Boolean satisfiability (SAT).

Dhiraj K. Pradhan is Chair of Computer Science at the University of Bristol, UK. He previously held the COE Endowed Chair Professorship in Computer Science at Texas A & M University, also serving as Founder of the Laboratory of Computer Systems there. He has also worked as a Staff Engineer at IBM, and served as the Founding CEO of Reliable Computer Technology, Inc. A Fellow of ACM, the IEEE, and the Japan Society of Promotion of Science, Professor Pradhan is the recipient of a Humboldt Prize, Germany, and has numerous major technical publications spanning more than 30 years.

Ian G. Harris is Associate Professor in the Department of Computer Science, University of California, Irvine. He is an Executive Committee Member of the IEEE Design Automation Technical Committee (DATC) and Chair of the DATC Embedded Systems Subcommittee, as well as Chair of the IEEE Test Technology Technical Committee (TTTC) and Publicity Chair of the IEEE TTTC Tutorials and Education Group. His research interests involve the testing and validation of hardware and software systems.
Practical Design Verification

Edited by

DHIRAJ K. PRADHAN
University of Bristol, UK

IAN G. HARRIS
University of California, Irvine
Contents

List of contributors

1 Model checking and equivalence checking 1
Masahiro Fujita

1.1 Introduction 1

1.2 Techniques for Boolean reasoning 2
1.2.1 Binary decision diagrams (BDDs) 3
1.2.2 Boolean satisfiability checker 6
1.2.3 Automatic test-pattern generation (ATPG) techniques 8

1.3 Model checking techniques 11
1.3.1 Property description with temporal logic 11
1.3.2 Basic algorithms of CTL model checking 14
1.3.3 Symbolic model checking 16
1.3.4 Practical model checking 20

1.4 Equivalence-checking techniques 22
1.4.1 Definition of equivalent designs 23
1.4.2 Latch-mapping problem 23
1.4.3 Practical combinational equivalence checking 24
1.4.4 Sequential equivalence checking (SEC) 28

1.5 Techniques for higher-level design descriptions 35

1.6 References 47

2 Transaction-level system modeling 51
Daniel Gajski and Samar Abdi

2.1 Taxonomy for TLMs 51
2.1.1 Granularity-based classification of TLMs 52
2.1.2 Objective-based classification 60

2.2 Estimation-oriented TLMs 62
2.2.1 Result-oriented modeling (ROM) 63
2.2.2 Similarity to TLM 63
2.2.3 Optimistic modeling 64
2.2.4 Measurements 64
2.3 Synthesis-oriented TLMs 65
  2.3.1 Universal bus channel (UBC) 67
  2.3.2 Transducer 75
  2.3.3 Routing 79
  2.3.4 TLMs for C-based design 80
  2.3.5 Synthesizable TLMs in practice: MP3 decoder design 83
2.4 Related work on TLMs 89
2.5 Summary and conclusions 90
2.6 References 90

3 Response checkers, monitors, and assertions 92
Harry Foster

3.1 Introduction 92
  3.1.1 Identifying what to check 92
  3.1.2 Classifying design behavior 93
  3.1.3 Observability and controllability 96
3.2 Testbench verification components 97
3.3 Assertion-based verification 99
  3.3.1 Brief introduction to SystemVerilog assertion 100
3.4 Assertion-based bus monitor example 102
  3.4.1 Basic write operation 104
  3.4.2 Basic read operation 105
  3.4.3 Unpipelined parallel bus interface requirements 106
  3.4.4 Unpipelined parallel bus interface coverage 108
  3.4.5 Analysis communication in the testbench 110
3.5 Summary 111
3.6 References 112

4 System debugging strategies 113
Wayne H. Wolf

4.1 Introduction 113
4.2 Debugging tools 114
  4.2.1 Logic analyzers and pattern generators 115
  4.2.2 Power measurement 116
  4.2.3 In-circuit emulators 117
  4.2.4 Emulators 117
  4.2.5 Profilers 117
  4.2.6 CPU simulators 118
4.3 Debugging commands 118
4.4 Functional debugging 119
4.5 Performance-oriented debugging 119
4.6 Summary 120
4.7 References 121
5  Test generation and coverage metrics  122  
Ernesto Sánchez, Giovanni Squillero, and Matteo Sonza Reorda

5.1 Introduction  122
5.2 Coverage metrics  128
5.3 Classification of coverage metrics  131
5.3.1 Code coverage metrics  131
5.3.2 Metrics based on circuit activity  136
5.3.3 Metrics based on finite-state machines  137
5.3.4 Functional coverage metrics  140
5.3.5 Error- (or fault-) based coverage metrics  141
5.3.6 Coverage metrics based on observability  143
5.4 Coverage metrics and abstraction levels of design  144
5.5 Stimuli generation methods  145
5.5.1 Manual generation  146
5.5.2 Automatic generation  147
5.6 Acknowledgements  151
5.7 References  151

6  SystemVerilog and Vera in a verification flow  154  
Shireesh Verma and Ian G. Harris

6.1 Introduction  154
6.2 Testbench components  155
6.2.1 Design under verification  156
6.2.2 Monitor  156
6.2.3 Checker  157
6.2.4 Scoreboard  158
6.2.5 Stimulus  159
6.3 Verification plan  160
6.4 Case study  160
6.4.1 DUV  160
6.4.2 Verification plan  163
6.4.3 Testbench  163
6.5 Summary  171
6.6 References  172

7  Decision diagrams for verification  173  
Maciej Ciesielski, Dhiraj K. Pradhan, and Abusaleh M. Jabir

7.1 Introduction  173
7.2 Decision diagrams  175
7.2.1 Binary decision diagrams (BDDs)  175
7.2.2 Beyond BDDs  181
7.3 Binary moment diagrams (BMDs)  183
7.4 Taylor expansion diagrams (TEDs)

7.4.1 Related work
7.4.2 Motivation
7.4.3 The Taylor series expansion
7.4.4 Reduction and normalization
7.4.5 Canonicity of Taylor expansion diagrams
7.4.6 Complexity of Taylor expansion diagrams
7.4.7 Composition of Taylor expansion diagrams
7.4.8 Design modeling and verification with TEDs
7.4.9 Implementation and experimental results
7.4.10 Limitations of TED representation
7.4.11 Conclusions and open problems

7.5 Representation of multiple-output functions over finite fields

7.5.1 Previous work
7.5.2 Background and notation
7.5.3 Graph-based representation
7.5.4 Reduction
7.5.5 Variable reordering
7.5.6 Operations in \( \text{GF}(N) \)
7.5.7 Multiple-output functions in \( \text{GF}(N) \)
7.5.8 Further node reduction
7.5.9 Representing characteristic functions in \( \text{GF}(N) \)
7.5.10 Evaluation of functions
7.5.11 Experimental results
7.5.12 Conclusions

7.6 Acknowledgements
7.7 References

8 Boolean satisfiability and EDA applications

8.1 Introduction
8.2 Definitions
8.2.1 Propositional formulas and satisfiability
8.2.2 Boolean circuits
8.2.3 Linear inequalities over Boolean variables
8.2.4 SAT algorithms
8.3 Extensions of SAT
8.4 Applications of SAT in EDA
8.4.1 Combinational equivalence checking
8.4.2 Automatic test-pattern generation

Contents
Contributors

Samar Abdi
Center for Embedded Computer Systems, University of California, Irvine, USA

Maciej Ciesielski
University of Massachusetts, Amherst, USA

Harry Foster
Mentor Graphics Corporation, USA

Masahiro Fujita
The University of Tokyo, Japan

Daniel Gajski
Center for Embedded Computer Systems, University of California, Irvine, USA

Ian G. Harris
University of California, Irvine, USA

Abusaleh M. Jabir
Oxford Brookes University, UK

Joao Marques-Silva
University of Southampton, UK

Dhiraj K. Pradhan
University of Bristol, UK

Matteo Sonza Reorda
Politecnico di Torino, Italy

Ernesto Sánchez
Politecnico di Torino, Italy
List of contributors

Giovanni Squillero
Politecnico di Torino, Italy

Shireesh Verma
Conexant Systems, Inc., USA

Wayne H. Wolf
Georgia Institute of Technology, USA