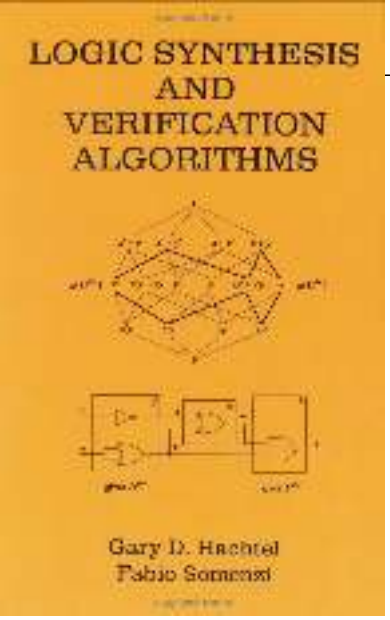
LOGIC SYNTHESIS
AND
VERIFICATION
ALGORITHMS
Gary D. Hachtel
Fabio Somenzi

LOGIC SYNTHESIS
AND
VERIFICATION ALGORITHMS

LOGIC SYNTHESIS AND VERIFICATION ALGORITHMS

by

Gary D. Hachtel
University of Colorado

Fabio Somenzi
University of Colorado

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 0-306-47592-8
Print ISBN: 0-7923-9746-0

Created in the United States of America

Visit Kluwer Online at: http://kluweronline.com
and Kluwer's eBookstore at: http://ebooks.kluweronline.com

To:

Linda, *Jordan,* and *Kira,*

and

Chiara and *Laura.*

Contents

I Introduction **1**

1 Introduction **5**

1.1 VLSI: Opportunity and Challenge 5
1.1.1 Manufacturing Technology 5
1.1.2 Design technology 6
1.1.3 Why VLSI 7
1.2 VLSI Processes 7
1.3 Design Styles 8
1.3.1 Design Decomposition 8
1.3.2 Logic (Circuit) Design Styles 10
1.4 Overview of Optimal Logic Synthesis 14
1.4.1 Area-Time Tradeoff Curves 15
1.4.2 The Technology Independent View — A Bit-Serial Full Adder Circuit 16
1.4.3 The Technology Dependent View — Technology Mapping . . . 18
1.4.4 Testing — Is What I Fabricated What I Wanted? 19
1.4.5 Graph Models and Finite State Machines 21
1.4.6 Successors and Predecessors 24
1.5 Graph Algorithms and Complexity 24
1.5.1 Complexity 24
1.5.2 Computing the Product of Sets of Sets 26
1.5.3 Longest Paths 27
1.5.4 Backtracing 29
1.5.5 Complexity of Computing the Longest Path 32
1.6 Asymptotic Complexity (or just complexity) 33
1.6.1 Worst Case Asymptotic Upper Bound Complexity 34
1.6.2 Complexity of Algorithms 36
1.6.3 Practical Complexities 36
1.7 Brief Summary of MOS Device Behavior 37
1.8 Notes 39
1.9 Summary 39
1.10 Problems 39

2 A Quick Tour of Logic Synthesis with the Help of a Simple Example 47
2.1 A Simple Case Conversion Circuit 47
2.2 First Refinement 49
2.3 The Transform Block 50
2.3.1 The CC Block 52
2.3.2 An Optimized Transform Block 53
2.4 The Command Interpreter 54
2.4.1 Checking for Equality 54
2.4.2 Optimizing the Command Interpreter 54
2.5 Technology Mapping 57
2.6 Problems 58

II Two Level Logic Synthesis 73

3 Boolean Algebras 77
3.1 Sets, Relations, and Functions 77
3.1.1 Sets 77
3.1.2 Relations 79
3.1.3 Reflexive Binary Relations 80
3.1.4 Functions 84
3.2 Partial Orders 85
3.2.1 Partially Ordered Sets 86
3.2.2 Hasse Diagrams 87
3.2.3 The Meet and Join Operations 87
3.2.4 Totally Ordered Sets, Well-Ordered Sets, and Induction 89
3.2.5 Lattices 90
3.2.6 Definition of Boolean Algebras 92
3.2.7 Examples and Properties of Boolean Algebras 92
3.3 Boolean Functions 95
3.3.1 Boolean Formulae 96
3.3.2 Boolean Functions 97
3.3.3 Boole's Expansion Theorem 98
3.3.4 The Minterm Canonical Form 99
3.3.5 Pseudo-Boolean Functions 101
3.3.6 The Boolean Algebra of n-variable Boolean Functions 101
3.3.7 Atoms of a Boolean Algebra 101
3.4 Don't Care Conditions as Boolean Function Algebra Intervals 103
3.4.1 Satisfiability Don't Care Conditions 104
3.4.2 Observability Don't Care Conditions 105
3.4.3 Deriving Don't Cares From and Interval Specification 106
3.5 Incomplete Specification of Boolean Functions 106
3.5.1 Incompletely Specified Switching Functions 106
3.5.2 Incompletely Specified Boolean Functions 107
3.6 Notes 108
3.7 Summary 108
3.8 Problems 108

4 Synthesis of Two-Level Circuits **127**
4.1 Design Optimality . . . 127
4.2 Two-Level Logic . . . 129
4.2.1 Cost Functions for Two-Level Implementations . . . 130
4.2.2 Minimality and Testability . . . 131
4.3 Sums of Products and Products of Sums . . . 132
4.4 Implicants and Prime Implicants . . . 134
4.4.1 Quine's Prime Implicant Theorem . . . 134
4.5 Iterated Consensus . . . 134
4.5.1 Consensus and Implications: A Digression . . . 135
4.5.2 The Tabular Method of Computing the Prime Implicants . . . 135
4.5.3 Iterated Consensus in General . . . 137
4.6 Recursive Computation of Prime Implicants . . . 138
4.7 Selecting a Subset of Primes . . . 141
4.8 The Unate Covering Problem . . . 143
4.8.1 Reduction Techniques . . . 146
4.8.2 Essential Columns or Variables . . . 146
4.8.3 Row or Constraint Dominance . . . 146
4.8.4 Column or Variable Dominance . . . 147
4.8.5 Systematically Exploring the Search Space . . . 148
4.8.6 Computation of the Lower Bound . . . 149
4.9 The Branch-and-Bound Algorithm . . . 152
4.9.1 Choice of the Splitting Variable . . . 154
4.9.2 Examples of Splitting and Lower Bounding . . . 155
4.9.3 The Unate Covering Problem as an Integer Linear Program . . . 160
4.10 Multiple Output Functions . . . 160
4.10.1 Multiple-Output Primes . . . 161
4.10.2 Formulating the Covering Problem . . . 163
4.10.3 Incompletely Specified Multiple-Output Functions . . . 163
4.11 Notes . . . 164
4.12 Summary . . . 165
4.13 Problems . . . 165

5 Heuristic Minimization of Two-Level Circuits **185**
5.1 Local Search . . . 185
5.1.1 Local Search Applied to Logic Minimization . . . 187
5.1.2 A Simple Local Search Algorithm for Logic Minimization . . . 190
5.2 Checking for Equivalence and Tautology . . . 191
5.2.1 Unate Functions . . . 194
5.2.2 Additional Speed-Up Techniques for Tautology Checking . . . 197
5.2.3 Examples of Tautology Checks . . . 199
5.3 Choosing the Right Direction . . . 200
5.3.1 Recursive Complementation . . . 201
5.3.2 Using the OFF-set in the Expansion . . . 203
5.4 Identifying Essential Primes . . . 203
5.5 Multiple-Valued Logics . . . 204

5.6 Notes . . . 205
5.7 Summary . . . 206
5.8 Problems . . . 206

6 Binary Decision Diagrams (BDDs) 219
6.1 Representing Logic Functions with BDDs . . . 220
6.1.1 Binary Decision Diagrams by Way of Examples . . . 220
6.1.2 Formal Definition of BDDs . . . 222
6.1.3 How to Build the BDD for f . . . 225
6.1.4 Reduced BDDs . . . 226
6.1.5 Why Ordering is Important . . . 230
6.2 Design Considerations for a BDD Package . . . 231
6.3 Algorithms . . . 233
6.3.1 The ITE Algorithm . . . 234
6.3.2 Complement Edges . . . 237
6.3.3 The Computed Table . . . 238
6.3.4 Conditioning of the ITE Calls . . . 238
6.3.5 The ITE_CONSTANT Algorithm . . . 240
6.4 Notes . . . 243
6.5 Summary . . . 244
6.6 Problems . . . 244

III Models of Sequential Systems 251

7 Models of Sequential Systems 255
7.1 Introduction to Finite State Machines . . . 255
7.2 Synthesis of Finite State Machines . . . 257
7.3 FSMs: Definitions, Notation, and Examples . . . 261
7.3.1 Examples . . . 261
7.3.2 Incomplete Specification . . . 263
7.4 FSM Minimization for Completely Specified Machines . . . 265
7.4.1 Identifying the Equivalent States of an FSM . . . 265
7.4.2 State Equivalence Checking: the Partition/Refinement Approach 269
7.4.3 Finding the Reduced Machine . . . 272
7.4.4 Moore Machines and DFAs . . . 272
7.4.5 The Iterative Collapsing Approach . . . 273
7.4.6 Summary of State Equivalence Checking Methods . . . 275
7.5 Graph Algorithms for FSM Traversal . . . 275
7.5.1 Graphs, Subgraphs, and Components . . . 276
7.5.2 Graph Traversal — Breadth First Search . . . 278
7.5.3 Traversal — Depth First Search . . . 280
7.5.4 Finding the SCCs of a Directed Graph . . . 282
7.5.5 Shortest Paths . . . 286
7.6 Models of Sequential Systems . . . 289
7.7 FSTs: Strings, Runs, Reachability and Products . . . 292
7.7.1 Finite State Transition Structures . . . 292

7.7.2 NFAs and ε-moves . . . 295
7.7.3 FSTs as Labeled Digraphs . . . 295
7.7.4 Strings, Tapes and Runs of FSTs . . . 297
7.7.5 Product of FSTs . . . 298
7.8 FSM Equivalence Checking . . . 300
7.8.1 Strings which Distinguish Two Machines . . . 300
7.8.2 Building the Product Machine . . . 301
7.8.3 Equivalence Identification by Isomorphism . . . 305
7.9 Reachability Analysis . . . 305
7.9.1 FSM Traversal Using Binary Decision Diagrams . . . 305
7.10 Symbolic FSM State Traversal . . . 308
7.10.1 Transition Relations and Symbolic Image Computation . . . 308
7.11 Notes . . . 312
7.12 Summary . . . 313
7.13 Problems . . . 313

8 Synthesis and Verification of Finite State Machines 325
8.1 Minimization of Incompletely Specified Machines . . . 325
8.1.1 Finding the Compatible Pairs . . . 328
8.1.2 Finding the Maximal Compatibles . . . 329
8.1.3 Finding the Prime Compatibles . . . 329
8.1.4 Setting up the Covering Problem . . . 332
8.1.5 Forming the Reduced Table . . . 334
8.2 The Binate Covering Problem . . . 335
8.2.1 Formulation of BCP . . . 337
8.2.2 Reduction Techniques . . . 337
8.2.3 Choice of the Splitting Variable and Bounding . . . 340
8.2.4 Maximal independent set. . . . 340
8.2.5 Choice of the branching column . . . 341
8.2.6 Infeasible problems. . . . 341
8.2.7 An Example of Reductions . . . 342
8.3 State Encoding . . . 343
8.3.1 Practical Encoding Algorithms . . . 343
8.4 Decomposition and Encoding . . . 347
8.4.1 Partitions . . . 348
8.4.2 Partitions with Substitution Property . . . 350
8.4.3 Computation of the S.P. Partitions . . . 352
8.4.4 General Decomposition and State Encoding . . . 354
8.5 Notes . . . 356
8.6 Notes . . . 357
8.7 Summary . . . 357
8.8 Problems . . . 357

9 Finite Automata **369**
9.1 Finite Automata and Regular Languages . . . 370
9.1.1 String Acceptance . . . 372
9.1.2 Languages of Finite Automata . . . 373
9.1.3 Complements of Languages . . . 376
9.1.4 Examples . . . 377
9.2 DFA Synthesis . . . 378
9.2.1 Determinization of FSTs and FAs . . . 383
9.2.2 The Subset Construction . . . 383
9.2.3 The Deterministic Image . . . 385
9.3 ω-Regular Automata . . . 387
9.4 Formal Verification with L-Automata . . . 390
9.4.1 ω-Regular Languages . . . 390
9.5 ω-regular Language Containment . . . 392
9.5.1 Lifting Acceptance Conditions to a Product L-Automaton . . . 393
9.5.2 Example of Product L-Automaton . . . 393
9.5.3 BDD Representation of Cycle Sets and Recur Edges . . . 394
9.5.4 The Language Containment Algorithm . . . 395
9.5.5 Example of Containment Check . . . 396
9.6 Notes . . . 397
9.7 Summary . . . 397
9.8 Problems . . . 398

IV Multilevel Logic Synthesis **405**

10 Multi-Level Logic Synthesis **409**
10.1 Introduction . . . 409
10.1.1 Networks and Algebraic Operations . . . 410
10.2 Representation Issues and Choices . . . 412
10.2.1 Alternate Node Representations . . . 413
10.3 Representing Switching Functions in Factored Form . . . 417
10.3.1 Factored Forms . . . 417
10.3.2 Algebraic and Boolean Expressions . . . 418
10.3.3 Algebraic and Boolean Factored Forms . . . 419
10.3.4 Value of a Factorization . . . 420
10.3.5 Equivalent, Maximal, and Optimum Factorizations . . . 420
10.3.6 Size, Unateness, and Cofactors of a Factored Form . . . 422
10.4 Division . . . 422
10.5 Kernels and Co-Kernels . . . 425
10.5.1 Computation of Co-Kernels and Kernels . . . 427
10.6 Heuristic Factoring Algorithms . . . 428
10.6.1 Generic Factoring Algorithm . . . 429
10.6.2 Quick Factor . . . 433
10.6.3 Good Factor . . . 434
10.6.4 Boolean Factor . . . 434
10.6.5 Summary of Factoring Algorithms . . . 435

10.6.6 Rectangle Covering 436
10.7 Decomposition and Restructuring 436
10.7.1 Algebraic Resubstitution 436
10.7.2 Selective Node Elimination 437
10.7.3 Extraction 439
10.8 Notes 440
10.9 Summary 441
10.10 Problems 441

11 Multi-Level Minimization **455**
11.1 Introduction 455
11.2 Boolean Networks 456
11.2.1 Network Cost 459
11.3 Don't Cares in Multi-Level Networks 461
11.3.1 Satisfiability Don't Cares 461
11.3.2 Observability Don't Cares 462
11.3.3 Use of Don't Cares in Minimization 462
11.3.4 Internal and External Don't Cares 463
11.3.5 External Satisfiability Don't Care Conditions 463
11.3.6 External Observability Don't Care Conditions 463
11.4 Internal Satisfiability Don't Cares 464
11.5 Observability Don't Cares 465
11.5.1 Computing ODCs with the Boolean Difference 468
11.6 Prime and Irredundant Networks 468
11.7 Two-Level Minimization with Multi-Level Don't Cares 469
11.8 Notes 470
11.9 Summary 470
11.10 Problems 471

12 Automatic Test Generation for Combinational Circuits **475**
12.1 Introduction 475
12.2 Faults and Fault Models 476
12.3 Automatic Test Generation 478
12.3.1 Excitation and Sensitization 478
12.3.2 A Simple Test Generation Algorithm 481
12.3.3 Implications and Backtracking 483
12.3.4 Choice of the Decision Variables 486
12.3.5 Putting the Pieces Together 488
12.4 Redundancy Removal 488
12.5 Notes 492
12.6 Summary 492
12.7 Problems 492

13 Technology Mapping **505**
13.1 Graph Covering and Technology Mapping 506
13.2 Choice of Base Functions 507
13.3 Creating the Subject Graph 508
13.4 The DAG-Covering Problem 509
13.5 Tree Covering by Dynamic Programming 509
13.6 Decomposition 512
13.7 Delay Optimization and Graph Covering 513
13.8 Notes 514
13.9 Summary 514
13.10 Problems 515

A ASCII Codes **523**

B Supplementary Problems **525**

Bibliography **537**

Index **555**

List of Figures

1.1 MOS gates. 8
1.2 A six-transistor gate array cell. 12
1.3 A three-input NAND gate obtained from the cell of Figure 1.2. 12
1.4 Organization of a channeled gate array. 12
1.5 Two-input look-up table for FPGAs (Field Programmable Gate Arrays). 13
1.6 Area-Delay tradeoff curves. 15
1.7 Bit-serial adder circuit. 17
1.8 Bit-serial adder circuit after technology mapping. 19
1.9 Bit-serial adder circuit with fault $s5A1$ asserted. 19
1.10 Finite State Machine for Majority Circuit. 22
1.11 A simple directed graph. 23
1.12 Logic Graph of 1-bit full adder. The gate outputs are the vertices of the graph and the nets connecting gate outputs to gate inputs are the edges of the graph. 25
1.13 Procedure for Intersecting 2 sets of sets. 26
1.14 28
1.15 A weighted directed acyclic graph. 29
1.16 31
1.17 A function $F(n)$ in the set $O(\log_2(n))$ and also in the set $\Omega(\log_2(n))$. 35
1.18 The FSM corresponding to the driver circuit of Figure 1.1 (middle) . 38
1.19 Complex CMOS gate for Problem 1. 40
1.20 Solution of Problem 1. 40
1.21 Circuit for Problem 4. 41
1.22 Procedure LEVELIZE1. 43

2.1 Interface of the example circuit. 48
2.2 Block diagram for LUNC. 49
2.3 Block diagram for the transform block. 51
2.4 Procedure CHANGECASE 52
2.5 Block diagram for the CC block. 52
2.6 Circuit schematic for the optimized transform block. 53
2.7 Block diagram for the command interpreter. 54
2.8 Procedure SWITCH 55
2.9 Circuit schematic for an equality checker. 56
2.10 Circuit schematic for the optimized command interpreter. 56
2.11 Circuit schematic for the technology-mapped decoder of the command interpreter. 58

2.12 Iterative scheme for the 8-bit comparator of Problem 3. 60
2.13 Circuit for Problem 4. 62

3.1 Venn Diagrams for illustrating set inclusion. 79
3.2 Matrix and graph representations of a binary relation. 80
3.3 Illustration of image and preimage. 86
3.4 Examples of posets. 87
3.5 Examples of lattices. 90
3.6 The Boolean algebra defined over the power sets of $\{a, b\}$ and $\{a, b, c\}$. 93
3.7 The Boolean algebra of the Boolean functions of two variables over B = {0,1}. 102
3.8 The interval $[xy', x + y]$ (represented by solid lines). 103
3.9 A simple example relating intervals in a Boolean function algebra to satisfiability and observability don't care conditions. 104
3.10 Hasse Diagram for Problem 14. 112
3.11 Hasse Diagram for Problem 18. 113
3.12 Lattice for Problem 18. 113
3.13 Partially ordered set (poset) for Problem 26. 115
3.14 Partially ordered set (poset) for Problem 27. 115
3.15 Hasse Diagrams for Problem 31. 117
3.16 Hasse Diagram for Problem 32. 117
3.17 Lattice for Problem 33. 118
3.18 Lattice of the Boolean functions of one variable over the Boolean algebra $B = \{0, a, b, 1\}$. (Problem 58.) 125

4.1 Tradeoff of area for speed for optimal designs. 128
4.2 NMOS NAND-NAND PLA. 130
4.3 Tabular Method Applied to $f = x'y' + wxy + x'yz' + wy'z$. 136
4.4 Tabular Method Applied to an Incompletely Specified Function. . . . 137
4.5 Example of Recursion Tree for the Computation of Prime Implicants. 140
4.6 A Function with a Cyclic Core. 142
4.7 Algorithm for computing an MIS. 151
4.8 Recursion Tree for a Covering Problem. 152
4.9 Example of Search Tree. 153
4.10 Branch-and-Bound Algorithm for the Unate Covering Problem. 154
4.11 A search tree produced by Procedure BCP. 156
4.12 A Two-Output Function that Illustrates the Importance of Sharing Common Terms. 161
4.13 Two Implementations for the Multiple-Output Function of Figure 4.12. 161
4.14 Tabular Method Applied to the Multiple-Output Function of Figure 4.12. 163
4.15 Recursion Tree for Problem 14. 172
4.16 Recursion Tree for Problem 20. 175

5.1 A Pictorial Representation of Local Search. 186
5.2 A Convex Optimization Problem. 186
5.3 A Non-Convex Optimization Problem. 187

5.4 A Function with an Initial Cover *(a)* and after the Expansion of an Implicant *(b)*. . . . 187
5.5 A Function and an Initial Cover Illustrating Output Expansion. . . . 188
5.6 The Cover of Figure 5.5 after the Expansion of an Output Part. . . . 188
5.7 A Function and an Initial Cover Illustrating Input Reduction. 189
5.8 Simple Minimization Loop. . . . 190
5.9 A Circuit that is Simplified by MAKE_SPARSE. . . . 191
5.10 Example where the Directions in which Cubes are Expanded Matters. *(a)*: Initial Cover. *(b)*: After Reduction. *(c)*: After Expansion in the Right Direction. . . . 200
5.11 The Interconnection of Sub-Circuits Gives Rise to Encoding Problems. 205

6.1 A MUX circuit and the corresponding BDD. . . . 221
6.2 A binary decision diagram. . . . 221
6.3 Another BDD. . . . 222
6.4 An optimal BDD. . . . 223
6.5 BDDs for typical functions. . . . 224
6.6 Partial BDD after expansion with respect to b. . . . 225
6.7 Partial BDD after expansion with respect to b and c. . . . 226
6.8 Final BDD. . . . 227
6.9 Non-reduced BDD. . . . 227
6.10 Two isomorphic subgraphs. . . . 228
6.11 Merging two isomorphic subgraphs. . . . 228
6.12 Elimination of a redundant node. . . . 229
6.13 BDD illustrating the advantages of a good ordering. . . . 230
6.14 BDD illustrating the drawbacks of a bad ordering. . . . 231
6.15 Shared BDD. . . . 232
6.16 Two-argument operators expressed in terms of ITE. . . . 234
6.17 Pseudo-code of the ITE algorithm. . . . 236
6.18 Example of application of ITE. . . . 236
6.19 Equivalent pairs of functions. . . . 237
6.20 Pseudo-code of the ITE_CONSTANT algorithm. . . . 240
6.21 An example of computation by ITE_CONSTANT. . . . 241
6.22 BDDs f, g, h and ITE (f, g, h). . . . 242
6.23 BDD for Problem 1. . . . 245
6.24 Solution for Problem 1. . . . 245
6.25 BDD for Problem 2. . . . 246
6.26 Solution for Problem 3. . . . 247
6.27 Solution for Problem 4. . . . 248
6.28 Pseudo-code of the APPLY algorithm. . . . 249
6.29 Pseudo-code of the OR operation. . . . 249
6.30 Solution for Problem 6. . . . 249

7.1 Simple Sequential Circuit. . . . 256
7.2 State Transition Graph for the Circuit of Figure 7.1. . . . 257
7.3 Simplified FSM Design Flow. . . . 257
7.4 An FSM with Redundant States. . . . 258

7.5 A Finite State Machine. . . . 261
7.6 Example of State Transition Graph. . . . 262
7.7 Tabular Representations of FSMs. . . . 262
7.8 STG of the symbolic LUNC FSM. . . . 263
7.9 Example of Incompletely Specified FSM. . . . 264
7.10 Machine Equivalent to the One of Figure 7.9. . . . 264
7.11 The STG of a simple FSM. . . . 265
7.12 The STG of an FSM equivalent to the one of Figure 7.11. . . . 266
7.13 Procedure for Finding Equivalent States of an FSM. . . . 268
7.14 The STG of an FSM in which all state pairs are equivalent. . . . 270
7.15 Flow Table for a Completely Specified Mealy Machine. . . . 271
7.16 Flow Table for a Completely Specified Moore Machine. . . . 273
7.17 Result of Reducing the FSM of Figure 7.16. . . . 274
7.18 First Collapsed Flow Table. . . . 274
7.19 Second Collapsed Flow Table. . . . 275
7.20 A simple undirected graph. . . . 276
7.21 A digraph and its strong components . . . 277
7.22 A directed graph representing the connectivity of the circuit of Figure 1.12. . . . 277
7.23 Procedure for basic Breadth First Search. . . . 279
7.24 A directed acyclic graph. . . . 279
7.25 Algorithm for Depth First Search Traversal of graph $G = (V, E)$ from start vertex u (first call). . . . 281
7.26 DAG with nodes labeled by [*preorder*, *postorder*, *lowlink*]. . . . 281
7.27 Recursive procedure for depth first search, modified to identify SCCs. 283
7.28 Algorithm for popping the *SCC* stack in DFS_SCC. . . . 284
7.29 DAG with labeled edges. . . . 284
7.30 Procedure for finding shortest paths in a weighted graph. . . . 287
7.31 A weighted directed acyclic graph. . . . 288
7.32 Models of finite-state transition systems. . . . 289
7.33 A Finite State Transition Structure. . . . 293
7.34 The STGs of Tables 7.2 and 7.3. . . . 294
7.35 NFA example with ϵ-moves. . . . 295
7.36 The FST of the Mead-Conway Traffic Controller. . . . 296
7.37 Product of FSTs. . . . 299
7.38 Product of Nondeterministic FSTs. . . . 299
7.39 Product Machine for Equivalence Checking. . . . 301
7.40 Encoded Product Machine for Equivalence Checking. . . . 302
7.41 Product of two equivalent FSMs. . . . 302
7.42 Procedure for equivalence checking a product machine. . . . 303
7.43 Procedure for finding a shortest error trace. . . . 304
7.44 A simple BDD representing the characteristic function of the set S. . 307
7.45 Two non-equivalent FSMs . . . 310
7.46 Product of the Two FSMs of Figure 7.45. . . . 310
7.47 The STGs of two equivalent FSMs. . . . 314
7.48 The STG of a modulo 3 counter. . . . 315

7.49 The STG of an FSM to be minimized. . . . 315
7.50 Procedure for finding 1-equivalent states of an FSM. . . . 316
7.51 Procedure for finding equivalent states of an FSM. . . . 317
7.52 A Completely Specified Flow Table. . . . 319
7.53 STG for the Flow Table of Figure 7.52. . . . 319
7.54 A Completely Specified Flow Table. . . . 321
7.55 Minimized Flow Table for Figure 7.54. . . . 321
7.56 Flow Table for Problem 14. . . . 321
7.57 A simple directed graph. . . . 322
7.58 Another simple undirected graph. . . . 322
7.59 Partial labeling of directed acyclic graph 7.59. . . . 323

8.1 An incompletely specified Moore machine. . . . 326
8.2 Another incompletely specified Moore machine. . . . 326
8.3 Reduced machine obtained from the one of Figure 8.2. . . . 327
8.4 Machine obtained from the one of Figure 8.2 by state splitting. . . . 327
8.5 A flow table and its compatibility table. . . . 328
8.6 A flow table to illustrate the computation of prime classes. . . . 330
8.7 Compatibility table for the flow table of Figure 8.6. . . . 330
8.8 Prime compatibles for the flow table of Figure 8.6. . . . 332
8.9 Algorithm for computing prime compatibles. . . . 333
8.10 Reduced flow table obtained from the one of Figure 8.6. . . . 335
8.11 Reduced flow table obtained from the one of Figure 8.10 by heuristic choices of the next state entries. . . . 335
8.12 Branch and bound algorithm for binate covering. . . . 336
8.13 Example FSM for the discussion of state encoding. . . . 344
8.14 Attraction graph for the FSM of Figure 8.13. . . . 344
8.15 Attraction graph produced by the fanout-oriented algorithm of MUSTANG. . . . 346
8.16 An assignment derived by the fanout-oriented algorithm. . . . 346
8.17 An assignment derived by the fanin-oriented algorithm. . . . 347
8.18 Example of FSM with parallel decomposition. . . . 350
8.19 Components of the FSM of Figure 8.18. . . . 351
8.20 Structure of the parallel decomposition. . . . 351
8.21 Structure of the serial decomposition. . . . 352
8.22 Example of FSM with serial decomposition. . . . 352
8.23 Independent component for the FSM of Figure 8.22. . . . 352
8.24 First step in the construction of the dependent component. . . . 353
8.25 Second step in the construction of the dependent component. . . . 353
8.26 Example FSM for the computation of the S.P. partitions. . . . 353
8.27 S.P. partition lattice for the example of Figure 8.26. . . . 354
8.28 Example FSM for encoding based on partition pairs. . . . 355
8.29 Schematic for the encoding of the machine of Figure 8.28. . . . 356
8.30 An incompletely specified flow table. . . . 357
8.31 Compatibility table for the flow table of Figure 8.30. . . . 358
8.32 Search tree for Problem 3. . . . 359

8.33 Result of minimizing the flow table of Figure 8.30 using maximal compatibles only. . . . 359
8.34 Flow table for Problem 7. . . . 361
8.35 Compatibility table for the flow table of Figure 8.34. . . . 362
8.36 Covering table for Problem 7. . . . 364
8.37 Search tree for the covering problem of Figure 8.36. . . . 364
8.38 Result of minimizing the flow table of Figure 8.34. . . . 365
8.39 Flow table for Problem 9. . . . 365
8.40 Matrices S and Z for Problem 9. . . . 366
8.41 Attraction graph for the fanout-oriented algorithm. . . . 366
8.42 Encoding for the fanout-oriented algorithm. . . . 367
8.43 Matrices $\hat{S}$ and X for Problem 9. . . . 367
8.44 Attraction graph for the fanin-oriented algorithm. . . . 367
8.45 Encoding for the fanin-oriented algorithm. . . . 368

9.1 Physical implementation of a Finite Automaton. . . . 370
9.2 A DFA accepting all strings ending in 111. . . . 370
9.3 An NFA (Nondeterministic Finite Automaton). . . . 371
9.4 Procedure for deciding string acceptance. . . . 373
9.5 An NFA (top) and DFA (bottom) accepting the language of Example 9.1.1. 375
9.6 A DFA abstracted from the modulo 3 counter of Problem 5 of Page 314. 376
9.7 The complement of the DFA of Figure 9.6. . . . 376
9.8 A simple DFA. . . . 377
9.9 Binary Parse Tree for $(a^*b)^*$. . . . 379
9.10 Rule for constructing an NFA which accepts the product of two regular languages. . . . 379
9.11 Incorrect rule for constructing an NFA which accepts the product of two regular languages. . . . 380
9.12 Rule for constructing an NFA which accepts the union of two regular languages. . . . 381
9.13 Rule for constructing an NFA which accepts the closure of two regular languages. . . . 382
9.14 NFA whose language is $(a^*b)^*$. . . . 382
9.15 DFA whose language is $(a^*b)^*$. . . . 384
9.16 Algorithm SUBSET_CONSTRUCTION for determinizing a given NFA. . . . 386
9.17 An FST and its deterministic image. . . . 387
9.18 An L-automaton for expressing a safety property in formal verification. 387
9.19 An L-automaton recognizing a class of tapes (infinite strings with at most two b inputs after an a, unless there is an intervening c). . . . 389
9.20 An L-automaton recognizing a tapes containing an infinite number of *Bach* substrings. . . . 390
9.21 Illustration of $\lim(\mathcal{L})$ and $\mathcal{L}^\omega$. . . . 391
9.22 Automata Accepting $\lim(\mathcal{L})$ and $\mathcal{L}^\omega$. . . . 392
9.23 An example of a product automaton. . . . 394
9.24 Procedure LANGUAGE_CONTAINMENT. . . . 396
9.25 Illustration of non-containment in cycle set. . . . 396

9.26 Language containment test on the product automaton of Figure 9.23. 397
9.27 Flow table equivalent Moore machine for Problem 1. 399
9.28 A DFA for recognizing a certain string. 400
9.29 The DFA for Problem 5. 401
9.30 An L-automaton expressing a liveness property in formal verification. 402
9.31 A simple L-automaton. 403

10.1 Example of Local Optimization. 410
10.2 Another Example of Local Optimization. 410
10.3 Example of Circuit Restructuring. 411
10.4 Example of Boolean Network. 411
10.5 A CMOS Complex Gate Implementing $f = ((a + bc)(c + d))'$. 415
10.6 A Simple Gate Implementation of $f = ((a + bc)(c + d))'$. 415
10.7 NAND and NOR Decompositions. 416
10.8 Factoring Tree for $((a' + b)(a + e)(a + b') + e'$ 420
10.9 Weak Division Algorithm. 425
10.10Procedure GEN_FACTOR. 432
10.11Procedures QUICK_FACTOR, QUICK_DIVISOR, and ONE_LEVEL-0_KERNEL. 433
10.12Procedure for good factorization 434
10.13Procedure BOOL_FACTOR. 435
10.14Procedure QUICK-EXTRACTION. 439
10.15Factoring Tree for Problem 5. 443
10.16Factoring Tree for Problem 8. 443
10.17Boolean Network for Problem 24. 449
10.18Boolean Network for Problem 24 after Resubstitution. 449
10.19Boolean Network for Problem 26 after Extraction. 451

11.1 Example for Boolean Network. (Input and Output Elements are Buffers and are Considered Part of the Network.) 459
11.2 An Example Network for the Computation of Observability Don't Cares. 467
11.3 Network for Problem 3. 472
11.4 Boolean Network for Problem 4. 473
11.5 Simplified Boolean Network for Problem 4. 473
11.6 Boolean Network for Problem 6. 473
11.7 Simplified Boolean Network for Problem 6. 474
11.8 Circuit for Problem 7. 474

12.1 A short-circuit in a CMOS inverter. 476
12.2 Stuck-at faults. 476
12.3 Equivalent faults. 477
12.4 A simple combinational circuit. 478
12.5 Another simple combinational circuit. 479
12.6 A redundant combinational circuit. 479
12.7 A combinational circuit. 480
12.8 Use of compound values. 481
12.9 Frontier element ($G4$) and unjustified element ($G1$). 482
12.10Decision tree for the example of Figure 12.7. 483

12.11Example of implications. 483
12.12Another example of implications. 484
12.13Schneider's example. 484
12.14ATPG example. 485
12.15Decision tree for the example of Figure 12.14. 486
12.16Example of backtrace. 487
12.17A redundant circuit. 489
12.18Irredundant circuit derived from the one of Figure 12.17. 489
12.19Circuit with multiple redundancies that cannot be simultaneously removed. 490
12.20Circuit where the removal of one redundancy exposes another redundancy. 490
12.21Circuit of Figure 12.20 after the removal of the only redundancy. . . . 490
12.22Circuit of Figure 12.21 after the removal of the remaining redundancy. 490
12.23A 2-bit carry-skip adder. 491
12.24Combinational circuit for Problems 1–4. 493
12.25A decision tree for Problem 2. 494
12.26Combinational circuit for Problems 5–9. 495
12.27A decision tree for Problem 5. 495
12.28Circuit for Problem 6. 495
12.29Circuit for Problem 8. 496
12.30Decision tree for Problem 8. 496
12.31Circuit of Figure 12.26 after removal of one redundancy. 497
12.32Circuit of Figure 12.31 after removal of one redundancy (top) and after further removal of the inverter pair (bottom). 497
12.33Circuit for Problem 10. 497
12.34Circuit for Problem 11. 498
12.35Circuit of Figure 12.34 after removal of "input of Gate 2 connected to x_4 stuck-at-1." 499
12.36Circuit of Figure 12.34 after removal of "x_5 stuck-at-1." 500
12.37Circuit of Figure 12.36 after removal of "input of Gate 5 connected to x_4 stuck-at-0." 500

13.1 Splitting a DAG into a Forest of Trees. 510
13.2 A Subject Tree and its Matches. 511
13.3 The Two Possible Patterns for a Four-Input NAND Gate. 512
13.4 Two Possible Decompositions of the Same Circuit. 513
13.5 Library Patterns for Four-Input NOR and Three-Input OR. 515
13.6 Library of Pattern Trees. 516
13.7 Best Solution Trace. 517
13.8 Final Cover. 517
13.9 Modified Library of Pattern Trees. 517
13.10Best Modified Solution Trace. 518
13.11Modified Final Cover. 518
13.12Boolean Network for Problem 3. 519
13.13Boolean Network for Problems 4 and 5. 519

A.1 Table of ASCII Codes. 523

B.1 Boolean Network for Problem 13. 527
B.2 Boolean Network for Problem 14. 528
B.3 Boolean Network for Problem 15. 528
B.4 Boolean Network for Problem 16. 529
B.5 Circuit for Problem 17. 529
B.6 Circuit for Problem 18. 530
B.7 Circuit for Problem 19. 530
B.8 Circuit for Problem 20. 531
B.9 Circuit for Problem 21. 532

List of Tables

1.1 Data trace for Procedure LONGEST_PATH 30
1.2 Comparative growth of log, polynomial, polylog, and exponential functions 37
1.3 Computing times on a 10MIP's computer, assuming unit coefficients for each complexity function 37
1.4 Data trace for Procedure LONGEST_PATH, applied to Figure 1.7. . . . 45

3.1 Mapping of a simple function f. 85
3.2 Mapping of a two Boolean formulae representing the same Boolean function f. 96

7.1 Partial data trace for Procedure SHORTEST_PATH, applied to the graph of Figure 7.31($v_0 = 0$). 288
7.2 The δ mapping for a deterministic FST with initial state A. 294
7.3 The δ mapping for a nondeterministic FST with initial state A. 294
7.4 Data trace for procedure shortest_path 324

10.1 Cube Intersection Table. 427
10.2 Extended Cube Intersection Table. 428

Preface