

---

---

**LOGIC SYNTHESIS  
AND  
VERIFICATION ALGORITHMS**

by

**Gary D. Hachtel**  
University of Colorado

**Fabio Somenzi**  
University of Colorado

 Springer

# Contents

<b>I</b>	<b>Introduction</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>5</b>
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

<b>2</b>	<b>A Quick Tour of Logic Synthesis with the Help of a Simple Example</b>	<b>47</b>
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
<b>II</b>	<b>Two Level Logic Synthesis</b>	<b>73</b>
<b>3</b>	<b>Boolean Algebras</b>	<b>77</b>
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 . . . . .	106
3.6	Notes . . . . .	107
3.7	Summary . . . . .	108
3.8	Problems . . . . .	108

<b>4</b>	<b>Synthesis of Two-Level Circuits</b>	<b>127</b>
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
<b>5</b>	<b>Heuristic Minimization of Two-Level Circuits</b>	<b>185</b>
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
<b>6</b>	<b>Binary Decision Diagrams (BDDs)</b>	<b>219</b>
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
<b>III</b>	<b>Models of Sequential Systems</b>	<b>251</b>
<b>7</b>	<b>Models of Sequential Systems</b>	<b>255</b>
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 $\epsilon$ -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
<b>8</b>	<b>Synthesis and Verification of Finite State Machines</b> . . . . .	<b>325</b>
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

<b>9</b>	<b>Finite Automata</b>	<b>369</b>
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	$\omega$ -Regular Automata . . . . .	387
9.4	Formal Verification with $L$ -Automata . . . . .	390
9.4.1	$\omega$ -Regular Languages . . . . .	390
9.5	$\omega$ -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
<b>IV</b>	<b>Multilevel Logic Synthesis</b>	<b>405</b>
<b>10</b>	<b>Multi-Level Logic Synthesis</b>	<b>409</b>
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
<b>11</b>	<b>Multi-Level Minimization</b>	<b>455</b>
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
<b>12</b>	<b>Automatic Test Generation for Combinational Circuits</b>	<b>475</b>
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



<b>13 Technology Mapping</b>	<b>505</b>
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
<b>A ASCII Codes</b>	<b>523</b>
<b>B Supplementary Problems</b>	<b>525</b>
<b>Bibliography</b>	<b>537</b>
<b>Index</b>	<b>555</b>