



McGraw-Hill Series on
Computer Engineering

LOGIC SYNTHESIS

Srinivas Devadas
Abhijit Ghosh
Kurt Keutzer

Logic Synthesis

Srinivas Devadas

Abhijit Ghosh

Kurt Keutzer

McGraw-Hill, Inc.

New York San Francisco Washington, D.C. Auckland Bogotá
Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore
Sydney Tokyo Toronto

Library of Congress Cataloging-in-Publication Data

Devadas, Srinivas.

Logic synthesis / Srinivas Devadas, Abhijit Ghosh, Kurt Keutzer.
p. cm. — (McGraw-Hill series on computer engineering)

Includes bibliographical references and index.

ISBN 0-07-016500-9

1. Integrated circuits—Very large scale integration—Design—Data processing.

2. Logic design—Data processing. 3. Computer-aided design.

I. Ghosh, Abhijit. II. Keutzer, Kurt William. III. Title.

IV. Series: Series on computer engineering.

TK7874.D478 1994

621.39'5—dc20

93-41524

CIP

Copyright © 1994 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

2 3 4 5 6 7 8 9 0 DOC/DOC 9 0 9 8 7 6

ISBN 0-07-016500-9

*The sponsoring editor for this book was Stephen S. Chapman.
Printed and bound by R. R. Donnelley & Sons Company.*

Information contained in this work has been obtained by McGraw-Hill, Inc., from sources believed to be reliable. However, neither McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Other Computer Engineering Books of Interest

PERRY • *VHDL*, Second Edition 0-07-049434-7

ROSENSTARK • *Transmission Lines in Computer Engineering*
0-07-053953-7

PICK • *VHDL Techniques, Experiments, and Caveats* 0-07-049906-3

KIELKOWSKI • *Inside SPICE* 0-07-911525-X

MASSABRIO, ANTOGNETTI • *Semiconductor Device Modeling with
SPICE*, Second Edition 0-07-002469-3

DEWAR, SMOSNA • *Microprocessors* 0-07-016639-0

*To order or receive additional information on these
or any other McGraw-Hill titles, in the United States
please call 1-800-822-8158. In other countries, contact
your local McGraw-Hill representative.*

BC14BCZ

Preface

The dramatic increase in designer productivity over the past decade in the area of *very large scale integrated* (VLSI) circuit design is the direct result of the development of sophisticated *computer-aided design* (CAD) tools. Today, designers routinely describe the functionality of a circuit in a *hardware description language* (HDL) (which is a high-level description akin to a programming language) and use synthesis tools to produce optimized circuit layouts that can be sent off to a chip manufacturer to be fabricated on a silicon integrated circuit.

The two major areas in VLSI synthesis that have enabled vastly improved design turnaround times are logic synthesis and layout synthesis. The logic synthesis process consists of the translation of the input HDL description into a gate-level circuit and the optimization of the gate-level circuit. An optimized layout is produced for the final gate-level circuit by the layout synthesis process. Both the logic and layout synthesis process require the solution of difficult combinatorial optimization problems. In this book we focus solely on the logic synthesis process.

Switching and automata theory form the cornerstones of logic synthesis. Combinatorial problems associated with optimizing switching circuits abound in logic synthesis. In order to meet the demands of the designers, logic optimization systems have to be versatile and efficient. Versatility implies that the system should be able to target a variety of design parameters such as circuit area, delay, power dissipation, and testability. Efficiency implies that the system should be able to produce near-optimum or at least acceptable results for large VLSI circuits with reasonable CPU time expenditure.

The development of logic optimization systems that are versatile and efficient posed one of the major challenges for CAD in the 1980s. Today, thanks to a large amount of research and developmental effort such systems are in wide use among integrated circuit designers. This book focuses on describing the basic principles of logic design as well as the practical aspects of engineering a logic

synthesis system.

Very few individuals will themselves undertake to implement a logic synthesis system, but we feel that understanding the core principles of logic synthesis will be of use to a number of communities.

One community consists of educators in computer science and electrical engineering. In general an educator is looking for material that both disciplines the intellect of the student as well as prepares the student for practical problems the student is likely to encounter. In particular, the modern educator searching to find material that is relevant to the logical design of VLSI circuits has been faced with choosing either the classical works on switching and automata theory or anthologies of collected articles on logic synthesis and optimization methods. A large body of theory as well as many practical algorithms and methodologies to design logic circuits have been developed by researchers in logic synthesis, but so far theoretical and practical results have only been documented in numerous articles. This book attempts to fill the gap between the classical books written in the 1970s and modern logic optimization articles.

Another community consists of integrated circuit designers who are presently using logic synthesis to design integrated circuits. For these designers the former skills of handcrafting transistor-level layout or manually entering the schematics of a carefully designed gate-level implementation of a circuit are being replaced by the skill of writing efficient HDL models of integrated circuits. In order to develop the skill of writing HDL models of circuits that will result in efficient implementations, it is necessary for a designer to understand the basic principles underlying logic synthesis and optimization. One of the recurrent obstacles for a hardware designer using synthesis is the assumption that two functionally equivalent HDL models will produce similar circuits after logic synthesis and optimization. While we do not present a primer on HDL model development, it is our hope that by making the designer understand the capabilities and limitations of logic optimization software the designers will be able to build more efficient circuits within a synthesis framework. It is our hope that the discipline of logic synthesis will play just as central a role to the training and education of circuit designers as the discipline of compilers plays in the education and training of software developers.

A final community we wish to address are those fellow researchers in CAD who are working in logic synthesis or a related area. Research in logic synthesis is a very satisfying enterprise because improvements in algorithms can immediately translate into smaller or

faster circuits, and smaller and faster circuits have substantial commercial impact. Researchers wishing to get up to speed in logic synthesis have also been forced to rely on the classical works on switching and automata theory or on the anthologies of collected articles on logic synthesis and optimization. Here we hope to provide these researchers with a self-contained reference book that covers most of the principal synthesis and optimization techniques.

This book is organized into ten chapters. We provide an introduction to synthesis, verification, and testing in Chapter 1. The translation of an HDL model into a netlist of gates is introduced in Chapter 2. While integrated circuits implement sequential circuits, the most successful logic synthesis and optimization techniques have focused on the combinational portions. Therefore, in the remainder of the book we focus on the combinational portions of the circuits. The core algorithms for logic optimization were initially developed on two-level circuits and most easily understood in that context. For these reasons we will introduce these circuits first in Chapter 3. In the following chapters, we deal with the problems of minimizing two-level circuits so as to improve area and speed. We focus on testing a two-level circuit under various models of faulty behavior. We show how logic transformations applied to minimize the two-level circuit's area affect the testability of the circuit. While logic optimization techniques were first developed on two-level circuits, multilevel circuits are of much greater practical importance. In Chapters 6 through 9 we deal with the problems of synthesizing multilevel combinational logic circuits for minimal area, maximal speed, and high testability. Strong relationships between the area, speed, and testability of a circuit are highlighted throughout the book. We summarize the state-of-the-art in logic synthesis in Chapter 10.

Circuit representations and data structures cut across all facets of design such as synthesis, testing, and verification. At the combinational or sequential circuit level, Boolean functions are manipulated in various ways. The search for more efficient representations of Boolean functions is ceaseless, mainly because discovering such representations can have a significant impact on synthesis, testing, and verification problems. In this book we describe commonly used representations for combinational circuits and their advantages and disadvantages when applied to particular problems in synthesis and test.

Since we intend this book to be useful to CAD researchers, educators, and VLSI designers, we have included considerable de-

tail in the description of the various algorithms. Because we cannot comprehensively present the full panorama of logic optimization techniques, we present those techniques that have proven to be most useful in practice.

Acknowledgements

Over the years, several people have helped to deepen our understanding of VLSI synthesis, test generation, and synthesis for testability. We thank Jonathan Allen, Pranav Ashar, Robert Brayton, Michael Bryan, Gaetano Borriello, Raul Camposano, Steve Carlson, Tim Cheng, Aart De Geus, Giovanni De Micheli, Alfred Dunlop, Gary Hachtel, Niraj Jha, Charles Leiserson, Michael Lightner, Bill Lin, Tony Ma, Sharad Malik, Rick McGeer, Richard Newton, Paul Penfield, Sudhakar Reddy, Richard Rudell, Alexander Saldanha, Alberto Sangiovanni-Vincentelli, Fabio Somenzi, Kanwar Jit Singh, Albert Wang, Ruey-sing Wei, Jacob White, Tom Williams, and Wayne Wolf.

Some of the problems in this book have been taken or modified from Robert Brayton's class notes. Material modified from a variety of other sources is acknowledged by citations in the text.

Lastly, we thank Sulochana Devadas, Eliane Setton, Cate Hunter, and the rest of our families for their continual patience and encouragement.

Contents

Preface	xiii
Acknowledgements	xvii
1 Introduction	1
1.1 Digital Integrated Circuits	2
1.2 IC Design Methodology	2
1.3 Transistor-Level Layout	3
1.4 Gate-Level Entry	3
1.5 Initial Use of Logic Optimization	5
1.6 Emergence of Synthesis-Based Design	5
1.7 A Logic Synthesis Design Methodology	6
1.7.1 Behavioral Modeling	7
1.7.2 Register-Transfer Level Modeling	8
1.7.3 Two-level Logic Optimization	10
1.7.4 Multilevel Logic Optimization	10
1.7.5 Technology Mapping	11
1.7.6 Physical Design	11
1.8 Verification	12
1.8.1 Design Verification	12
1.8.2 Implementation Verification	13
1.9 Manufacture Testing	13
1.9.1 Fault Detection	14
1.9.2 Fault Models	14
1.10 Synthesis For Testability	16
1.11 Outline	17
References	18
2 Translation from HDL Descriptions	27
2.1 Introduction	27
2.2 A Policy for Synthesis from a HDL	28
2.2.1 Description Style	28
2.2.2 Supported Language Constructs	29

2.3	Synthesis Examples from VHDL	29
2.3.1	Entities and Architectures	30
2.3.2	Bit Vectors and Sequential Statements	30
2.3.3	Arithmetic Operators	32
2.3.4	If-Then-Else Statements	33
2.3.5	Wait Statements	33
2.3.6	Parameterizable Circuits	35
2.3.7	Greatest Common Divisor Circuit	37
2.4	Dependence on Input Description Style	39
	Problems	41
	References	42
3	Two-Level Combinational Circuits	45
3.1	Introduction	45
3.2	Terminology	46
3.3	Programmable Logic Arrays	50
3.4	Primality and Irredundancy Properties	51
3.5	Boolean Operations on Logic Functions	52
3.6	Operations on Cubes and Covers	52
3.6.1	Cube Intersection	53
3.6.2	Disjoint SHARP	53
3.6.3	Single Cube Containment	54
3.7	Complexity of Two-Level Circuits	55
	Problems	55
	References	57
4	Synthesis of Two-Level Circuits	59
4.1	Two-Level Boolean Minimization	59
4.2	The Quine-McCluskey Method	60
4.2.1	Prime Implicant Generation	60
4.2.2	Prime Implicant Table	61
4.2.3	Essential Prime Implicants	61
4.2.4	Dominated Columns	62
4.2.5	Dominating Rows	64
4.2.6	A Branching Covering Strategy	64
4.3	Two-Level Tautology	64
4.3.1	Unate Functions	65
4.3.2	Tautology Procedure	66
4.3.3	Example	68
4.4	Complementation	69
4.4.1	Basic Procedure	69

4.4.2	Special Cases	71
4.4.3	Unate Complementation	71
4.4.4	Example	72
4.5	Exact Minimization Methods	73
4.5.1	Prime Implicant Generation	74
4.5.2	Reduced Prime Implicant Table Generation	76
4.5.3	Branch-and-Bound Covering	79
4.6	Heuristic Minimization Methods	81
4.6.1	Heuristics Based on Exact Minimization	81
4.6.2	Heuristics Based on Iterative Improvement	81
4.6.3	ESPRESSO Minimization Loop	84
4.6.4	EXPAND	85
4.6.5	IRREDUNDANT	86
4.6.6	REDUCE	87
	Problems	89
	References	91
5	Testability of Two-Level Circuits	93
5.1	Introduction	93
5.2	Fault Models	93
5.2.1	Single-Stuck-At Fault Model	94
5.2.2	Multiple-Stuck-At Fault Model	95
5.2.3	Bridging Fault Model	96
5.2.4	Gate Delay Fault Model	96
5.2.5	Transistor Stuck-Open Fault Model	97
5.2.6	Path Delay Fault Model	99
5.2.7	Complexity of Test Generation	99
5.3	Single Stuck-At Faults	100
5.3.1	Conditions for Testability	100
5.3.2	Synthesis for Full Testability	101
5.3.3	Test Generation Methods	103
5.4	Multiple Stuck-At Faults	104
5.4.1	Conditions for Testability	104
5.4.2	Synthesis for Full Testability	106
5.4.3	Test Generation Methods	107
5.5	Timing Analysis Terminology	107
5.6	Robust and Nonrobust Testing	109
5.6.1	Introduction	109
5.6.2	Hazard-Free Robust Path Delay Faults	109
5.6.3	General Robust Path Delay Faults	111
5.6.4	Hazard-Free Robust Gate Delay Faults	112

5.6.5	General Robust Gate Delay Faults	113
5.6.6	Hazard-Free Robust Stuck-Open Faults	114
5.7	Hazard-Free Robust Path Delay Faults	115
5.7.1	Conditions for Testability	115
5.7.2	Synthesis for Maximal Testability	117
5.7.3	Test Generation Methods	121
5.8	General Robust Path Delay Faults	122
5.9	Hazard-Free Robust Gate Delay Faults	123
5.9.1	Conditions for Testability	123
5.9.2	Synthesis for Maximal Testability	124
5.9.3	Test Generation Methods	125
5.10	Hazard-Free Robust Stuck-Open Faults	125
	Problems	126
	References	127
6	Multilevel Combinational Circuits	129
6.1	Boolean Networks	129
6.2	Special Classes of Circuits	130
6.2.1	Fan-out-Free Circuits	130
6.2.2	Leaf-DAG Circuits	130
6.2.3	Algebraically Factored Circuits	131
6.2.4	Multiplexor-Based Circuits	131
6.3	Binary Decision Diagrams	131
6.4	Ordered Binary Decision Diagrams	132
6.4.1	Reduced Ordered Binary Decision Diagrams	134
6.4.2	Canonicity Property	136
6.4.3	Reduction	137
6.4.4	Complementation	138
6.4.5	Cofactor	138
6.4.6	APPLY	139
6.4.7	Circuit Equivalence using ROBDDs	141
6.4.8	Ordering Heuristics	143
6.4.9	Improvements to ROBDDs	144
6.4.10	Multiplexor-Based Networks	146
	Problems	147
	References	148
7	Synthesis of Multilevel Circuits	151
7.1	Logic Transformations	152
7.1.1	Decomposition	152
7.1.2	Extraction	153

7.1.3	Factoring	153
7.1.4	Substitution	154
7.1.5	Elimination	154
7.2	Division and Common Divisors	155
7.3	Algebraic Division	156
7.3.1	Computing the Quotient	156
7.3.2	Kernels and Algebraic Divisors	157
7.3.3	Computing the Kernels	158
7.3.4	Factoring Algorithm	161
7.3.5	Extraction and Resubstitution Algorithm	162
7.3.6	Algebraic Resubstitution with Complement	163
7.4	Rectangles and Rectangle Covering	163
7.4.1	Definitions	164
7.4.2	Rectangles and Kernels	165
7.4.3	Common-Cube Extraction	166
7.4.4	Kernel Intersection	168
7.4.5	Rectangle Algorithms	171
7.5	Boolean Division	176
7.6	Don't-Care-Based Optimization	177
7.6.1	Satisfiability Don't-Cares	178
7.6.2	Observability Don't-Cares	179
7.6.3	Don't-Care Generation	180
7.6.4	ROBDD implementation	182
7.6.5	Range Computation	182
7.7	Technology Mapping	185
7.7.1	Introduction	185
7.7.2	Technology Libraries	186
7.7.3	Cost Models	187
7.7.4	Graph Covering	188
7.7.5	Choice of Atomic Pattern Set	189
7.8	Technology Mapping by Tree Covering	190
7.8.1	Tree Covering Approximation	190
7.8.2	Partitioning the Subject Graph	191
7.8.3	Technology Decomposition	192
7.8.4	Tree Matching Techniques	192
7.8.5	Optimal Tree Covering	193
7.8.6	Inverter-Pair Heuristic	195
7.8.7	Extension to Nontree Patterns	196
7.8.8	Delay Optimization	197
7.8.9	Conclusions	198
7.9	Field Programmable Gate Arrays	198

7.9.1	FPGA Architectures	199
7.9.2	FPGA Terminology	201
7.9.3	FPGA Logic Block Architectures	202
7.9.4	FPGA Routing Architecture	206
7.10	FPGA Synthesis Methods	209
7.10.1	Lookup-Table-Based Architectures	210
7.10.2	Multiplexor-Based Architectures	213
	Problems	215
	References	219
8	Delay of Multilevel Circuits	225
8.1	Component and Circuit Delay	225
8.1.1	Component Delay Calculation	226
8.1.2	Circuit Delay Calculation	227
8.2	Timing Analysis and Verification	229
8.2.1	Topological Timing Analysis	229
8.2.2	False Paths in an Adder	231
8.2.3	Delay Models and Modes of Operation	232
8.2.4	Transition Mode and Monotone Speedup	233
8.2.5	Floating Mode and Monotone Speedup	236
8.2.6	Static Sensitization	237
8.2.7	Static Cosensitization	239
8.2.8	True Floating Mode Delay	240
8.3	Floating Mode Delay Computation	244
8.3.1	The PODEM Algorithm	245
8.3.2	Cube Simulation	248
8.3.3	Timed Test Generation	252
8.3.4	Backtrace	255
8.4	Technology-Independent Optimization	256
8.4.1	Circuit Restructuring	256
8.5	The Speedup Algorithm	257
8.5.1	Definitions	258
8.5.2	Outline of the Algorithm	259
8.5.3	Weight of the Critical Nodes	260
8.5.4	Minimum Weighted Cutset	261
8.5.5	Partial Collapsing	262
8.5.6	Timing Decomposition	262
8.5.7	Kernel-Based Decomposition	263
8.5.8	AND-OR Decomposition	264
8.5.9	Controlling the Algorithm	267
8.6	Technology Mapping for Delay	268