

Digital Design Using VHDL

A Systems Approach

This introductory textbook provides students with a system-level perspective and the tools they need to understand, analyze, and design digital systems. It goes beyond the design of simple combinational and sequential modules to show how such modules are used to build complete systems.

- All the essential topics needed to understand modern design practice are covered, including:
 - Design and analysis of combinational and sequential modules
 - Composition of combinational and sequential modules
 - Data and control partitioning
 - Factoring and composition of finite-state machines
 - Interface specification
 - System timing
 - Synchronization
- Teaches how to write VHDL-2008 HDL in a productive and maintainable style that enables CAD tools to do much of the tedious work.
- Covers the fundamentals of logic design, describing an efficient method to design combinational logic and state machines both manually and using modern CAD tools.

A complete introduction to digital design is given through clear explanations, extensive examples, and online VHDL files. The teaching package is completed with lecture slides, labs, and a solutions manual for instructors (available via www.cambridge.org/dallyvhdl). Assuming no previous digital knowledge, this textbook is ideal for undergraduate digital design courses that will prepare students for modern digital practice.

William J. Dally is the Willard R. and Inez Kerr Bell Professor of Engineering at Stanford University and Chief Scientist at NVIDIA Corporation. He and his group have developed system architecture, network architecture, signaling, routing, and synchronization technology that can be found in most large parallel computers today. He is a Member of the National Academy of Engineering, a Fellow of the IEEE, a Fellow of the ACM, and a Fellow of the American Academy of Arts and Sciences. He has received numerous honors, including the ACM Eckert-Mauchly Award, the IEEE Seymour Cray Award, and the ACM Maurice Wilkes Award.

R. Curtis Harting is a Software Engineer at Google and holds a Ph.D. from Stanford University. He graduated with honors in 2007 from Duke University with a B.S.E., majoring in Electrical & Computer Engineering and Computer Science. He received his M.S. in 2009 from Stanford University.

Tor M. Aamodt is an Associate Professor in the Department of Electrical and Computer Engineering at the University of British Columbia. Alongside his graduate students, he developed the GPGPU-Sim simulator. Three of his papers related to the architecture of general purpose GPUs have been selected as “Top Picks” by *IEEE Micro Magazine* and one as a “Research Highlight” by *Communications of the ACM magazine*. He was a Visiting Associate Professor in the Computer Science Department at Stanford University during his 2012–2013 sabbatical, and from 2004 to 2006 he worked at NVIDIA on the memory system architecture (“framebuffer”) of the GeForce 8 Series GPU.

“Dally and Harting blend circuit and architecture design in a clear and constructive manner on the basis of their exceptional experience in digital design.”

“Students will discover a modern and effective way to understand the fundamental underpinning of digital design, by being exposed to the different abstraction levels and views of computing systems.”

Giovanni De Micheli, *EPFL Switzerland*

“Bill and Curt have combined decades of academic and industry experience to produce a textbook that teaches digital system design from a very practical perspective without sacrificing the theoretical understanding needed to train tomorrow’s engineers. Their approach pushes students to understand not just what they are designing, but also what they are building. By presenting key advanced topics, such as synthesis, delay and logical effort, and synchronization, at the introductory level, this book is in the rare position of providing both practical advice and deep understanding. In doing so, this book will prepare students well even as technology, tools, and techniques change in the future.”

David Black-Schaffer, *Uppsala University*

“Everything you would expect from a book on digital design from Professor Dally. Decades of practical experience are distilled to provide the tools necessary to design and compose complete digital systems. A clear and well-written text that covers the basics and system-level issues equally well. An ideal starting point for the microprocessor and SoC designers of the future!”

Robert Mullins, *University of Cambridge and the Raspberry Pi Foundation*

“This textbook sets a new standard for how digital system design is taught to undergraduates. The practical approach and concrete examples provide a solid foundation for anyone who wants to understand or design modern complex digital systems.”

Steve Keckler, *The University of Texas at Austin*

“This book not only teaches how to do digital design, but more importantly shows how to do *good* design. It stresses the importance of modularization with clean interfaces, and the importance of producing digital artifacts that not only meet their specifications, but which can also be easily understood by others. It uses an aptly chosen set of examples and the Verilog code used to implement them.”

“It includes a section on the design of asynchronous logic, a topic that is likely to become increasingly important as energy consumption becomes a primary concern in digital systems.”

“The final appendix on Verilog coding style is particularly useful. This book will be valuable not only to students, but also to practitioners in the area. I recommend it highly.”

Chuck Thacker, *Microsoft*

“A terrific book with a terrific point-of-view of systems. Everything interesting – and awful – that happens in digital design happens because engineers must integrate ideas from bits to blocks, from signals to CPUs. The book does a great job of focusing on the important stuff, moving from foundations to systems, with the right amount of HDL (Verilog) focus to make everything practical and relevant.”

Rob A. Rutenbar, *University of Illinois at Urbana-Champaign*

Cambridge University Press
978-1-107-09886-2 - Digital Design Using VHDL: A Systems Approach
William J. Dally, R. Curtis Harting and Tor M. Aamodt
Frontmatter
[More information](#)

Digital Design Using VHDL

A Systems Approach

WILLIAM J. DALLY

Stanford University

R. CURTIS HARTING

Google, Inc.

TOR M. AAMODT

The University of British Columbia



CAMBRIDGE
UNIVERSITY PRESS

Cambridge University Press
978-1-107-09886-2 - Digital Design Using VHDL: A Systems Approach
William J. Dally, R. Curtis Harting and Tor M. Aamodt
Frontmatter
[More information](#)

CAMBRIDGE UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9781107098862

© Cambridge University Press 2016

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2016

Printed in the United Kingdom by TJ International Ltd. Padstow Cornwall

A catalog record for this publication is available from the British Library

Library of Congress Cataloging in Publication data

Dally, William J., author.

Digital design using VHDL : a systems approach / William J. Dally, Stanford University, California, R. Curtis Harting, Google, Inc., New York, Tor M. Aamodt, The University of British Columbia.
pages cm

Includes bibliographical references and index.

ISBN 978-1-107-09886-2 (Hardback : alk. paper)

1. Digital integrated circuits—Computer-aided design. 2. Electronic digital computers—Computer-aided design. 3. Digital electronics—Data processing. 4. VHDL (Computer hardware description language)
I. Harting, R. Curtis, author. II. Aamodt, Tor M., author. III. Title.

TK7868.D5D3285 2015

621.38150285/5133—dc23 2015021269

ISBN 978-1-107-09886-2 Hardback

Additional resources for this publication at www.cambridge.org/9781107098862

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

CONTENTS

Preface	<i>page</i> xv
Acknowledgments	xx

Part I Introduction

1 The digital abstraction	3
1.1 Digital signals	3
1.2 Digital signals tolerate noise	5
1.3 Digital signals represent complex data	8
1.3.1 Representing the day of the year	10
1.3.2 Representing subtractive colors	11
1.4 Digital logic functions	11
1.5 VHDL description of digital circuits and systems	13
1.6 Digital logic in systems	16
Summary	17
Bibliographic notes	18
Exercises	18
2 The practice of digital system design	22
2.1 The design process	22
2.1.1 Specification	22
2.1.2 Concept development and feasibility	24
2.1.3 Partitioning and detailed design	26
2.1.4 Verification	27
2.2 Digital systems are built from chips and boards	28
2.3 Computer-aided design tools	32
2.4 Moore’s law and digital system evolution	34
Summary	36
Bibliographic notes	36
Exercises	37

Part II Combinational logic

3 Boolean algebra	43
3.1 Axioms	43

Contents		
3.2	Properties	44
3.3	Dual functions	46
3.4	Normal form	47
3.5	From equations to gates	48
3.6	Boolean expressions in VHDL	51
	Summary	54
	Bibliographic notes	55
	Exercises	55
CMOS logic circuits		58
4.1	Switch logic	58
4.2	Switch model of MOS transistors	62
4.3	CMOS gate circuits	68
4.3.1	Basic CMOS gate circuit	69
4.3.2	Inverters, NANDs, and NORs	70
4.3.3	Complex gates	72
4.3.4	Tri-state circuits	75
4.3.5	Circuits to avoid	76
	Summary	77
	Bibliographic notes	78
	Exercises	78
Delay and power of CMOS circuits		82
5.1	Delay of static CMOS gates	82
5.2	Fan-out and driving large loads	85
5.3	Fan-in and logical effort	86
5.4	Delay calculation	89
5.5	Optimizing delay	92
5.6	Wire delay	94
5.7	Power dissipation in CMOS circuits	98
5.7.1	Dynamic power	98
5.7.2	Static power	99
5.7.3	Power scaling	100
	Summary	101
	Bibliographic notes	101
	Exercises	102
Combinational logic design		105
6.1	Combinational logic	105
6.2	Closure	106
6.3	Truth tables, minterms, and normal form	107
6.4	Implicants and cubes	110
6.5	Karnaugh maps	113
6.6	Covering a function	115

	Contents	vii
6.7	From a cover to gates	116
6.8	Incompletely specified functions	117
6.9	Product-of-sums implementation	119
6.10	Hazards	121
	Summary	123
	Bibliographic notes	124
	Exercises	124
7	VHDL descriptions of combinational logic	129
7.1	The prime number circuit in VHDL	129
7.1.1	A VHDL design entity	129
7.1.2	The <code>case</code> statement	131
7.1.3	The <code>case?</code> statement	134
7.1.4	The <code>if</code> statement	136
7.1.5	Concurrent signal assignment statements	136
7.1.6	Selected signal assignment statements	137
7.1.7	Conditional signal assignment statements	138
7.1.8	Structural description	138
7.1.9	The decimal prime number function	141
7.2	A testbench for the prime number circuit	143
7.3	Example: a seven-segment decoder	148
	Summary	153
	Bibliographic notes	154
	Exercises	154
8	Combinational building blocks	157
8.1	Multi-bit notation	157
8.2	Decoders	157
8.3	Multiplexers	163
8.4	Encoders	171
8.5	Arbiters and priority encoders	173
8.6	Comparators	180
8.7	Shifters	183
8.8	Read-only memories	184
8.9	Read–write memories	189
8.10	Programmable logic arrays	192
8.11	Data sheets	193
8.12	Intellectual property	195
	Summary	195
	Bibliographic notes	196
	Exercises	196
9	Combinational examples	199
9.1	Multiple-of-3 circuit	199

Contents

9.2	Tomorrow circuit	201
9.3	Priority arbiter	205
9.4	Tic-tac-toe	207
	Summary	214
	Exercises	215

Part III Arithmetic circuits

10	Arithmetic circuits	221
10.1	Binary numbers	221
10.2	Binary addition	224
10.3	Negative numbers and subtraction	230
10.4	Multiplication	237
10.5	Division	240
	Summary	244
	Exercises	245
11	Fixed- and floating-point numbers	250
11.1	Representation error: accuracy, precision, and resolution	250
11.2	Fixed-point numbers	252
11.2.1	Representation	252
11.2.2	Operations	255
11.3	Floating-point numbers	257
11.3.1	Representation	257
11.3.2	Denormalized numbers and gradual underflow	258
11.3.3	Floating-point multiplication	259
11.3.4	Floating-point addition/subtraction	260
	Summary	265
	Bibliographic note	265
	Exercises	265
12	Fast arithmetic circuits	269
12.1	Carry look-ahead	269
12.2	Booth recoding	276
12.3	Wallace trees	278
12.4	Synthesis notes	284
	Summary	286
	Bibliographic notes	287
	Exercises	287
13	Arithmetic examples	290
13.1	Complex multiplication	290
13.2	Converting between fixed- and floating-point formats	291

	Contents	ix
13.2.1 Floating-point format		291
13.2.2 Fixed- to floating-point conversion		293
13.2.3 Floating- to fixed-point conversion		297
13.3 FIR filter		298
Summary		300
Bibliographic note		300
Exercises		300

Part IV Synchronous sequential logic

14 Sequential logic	305
14.1 Sequential circuits	305
14.2 Synchronous sequential circuits	307
14.3 Traffic-light controller	309
14.4 State assignment	312
14.5 Implementation of finite-state machines	313
14.6 VHDL implementation of finite-state machines	316
Summary	324
Bibliographic notes	324
Exercises	324
15 Timing constraints	328
15.1 Propagation and contamination delay	328
15.2 The D flip-flop	331
15.3 Setup- and hold-time constraints	331
15.4 The effect of clock skew	334
15.5 Timing examples	336
15.6 Timing and logic synthesis	337
Summary	339
Bibliographic notes	340
Exercises	340
16 Datapath sequential logic	344
16.1 Counters	344
16.1.1 A simpler counter	344
16.1.2 Up/down/load counter	346
16.1.3 A timer	349
16.2 Shift registers	352
16.2.1 A simple shift register	352
16.2.2 Left/right/load (LRL) shift register	353
16.2.3 Universal shifter/counter	353
16.3 Control and data partitioning	356
16.3.1 Example: vending machine FSM	357

16.3.2	Example: combination lock	367
	Summary	372
	Exercises	372
17	Factoring finite-state machines	375
17.1	A light flasher	375
17.2	Traffic-light controller	382
	Summary	393
	Exercises	394
18	Microcode	398
18.1	Simple microcoded FSM	398
18.2	Instruction sequencing	402
18.3	Multi-way branches	408
18.4	Multiple instruction types	410
18.5	Microcode subroutines	414
18.6	Simple computer	420
	Summary	427
	Bibliographic notes	427
	Exercises	428
19	Sequential examples	431
19.1	Divide-by-3 counter	431
19.2	SOS detector	432
19.3	Tic-tac-toe game	439
19.4	Huffman encoder/decoder	439
19.4.1	Huffman encoder	440
19.4.2	Huffman decoder	442
	Summary	448
	Bibliographic note	448
	Exercises	448

Part V Practical design

20	Verification and test	453
20.1	Design verification	453
20.1.1	Verification coverage	453
20.1.2	Types of tests	454
20.1.3	Static timing analysis	455
20.1.4	Formal verification	455
20.1.5	Bug tracking	456
20.2	Test	456
20.2.1	Fault models	456

	Contents	xi
20.2.2 Combinational testing		457
20.2.3 Testing redundant logic		457
20.2.4 Scan		458
20.2.5 Built-in self-test (BIST)		459
20.2.6 Characterization		460
Summary		461
Bibliographic notes		462
Exercises		462

Part VI System design

21 System-level design		467
21.1 System design process		467
21.2 Specification		468
21.2.1 Pong		468
21.2.2 DES cracker		471
21.2.3 Music player		472
21.3 Partitioning		473
21.3.1 Pong		474
21.3.2 DES cracker		475
21.3.3 Music synthesizer		475
Summary		476
Bibliographic notes		477
Exercises		477
22 Interface and system-level timing		479
22.1 Interface timing		479
22.1.1 Always valid timing		479
22.1.2 Periodically valid signals		480
22.1.3 Flow control		481
22.2 Interface partitioning and selection		482
22.3 Serial and packetized interfaces		483
22.4 Isochronous timing		486
22.5 Timing tables		487
22.5.1 Event flow		488
22.5.2 Pipelining and anticipatory timing		488
22.6 Interface and timing examples		489
22.6.1 Pong		489
22.6.2 DES cracker		489
22.6.3 Music player		493
Summary		493
Exercises		494

23 Pipelines	497
23.1 Basic pipelining	497
23.2 Example pipelines	500
23.3 Example: pipelining a ripple-carry adder	502
23.4 Pipeline stalls	505
23.5 Double buffering	507
23.6 Load balance	511
23.7 Variable loads	512
23.8 Resource sharing	516
Summary	517
Bibliographic notes	518
Exercises	518
24 Interconnect	521
24.1 Abstract interconnect	521
24.2 Buses	522
24.3 Crossbar switches	524
24.4 Interconnection networks	527
Summary	529
Bibliographic notes	529
Exercises	530
25 Memory systems	532
25.1 Memory primitives	532
25.1.1 SRAM arrays	532
25.1.2 DRAM chips	534
25.2 Bit-slicing and banking memory	536
25.3 Interleaved memory	537
25.4 Caches	540
Summary	544
Bibliographic notes	545
Exercises	545

Part VII Asynchronous logic

26 Asynchronous sequential circuits	551
26.1 Flow-table analysis	551
26.2 Flow-table synthesis: the toggle circuit	554
26.3 Races and state assignment	558
Summary	562
Bibliographic notes	563
Exercises	563

	Contents	xiii
27 Flip-flops		566
27.1 Inside a latch		566
27.2 Inside a flip-flop		568
27.3 CMOS latches and flip-flops		571
27.4 Flow-table derivation of the latch		572
27.5 Flow-table synthesis of a D flip-flop		574
Summary		576
Bibliographic notes		577
Exercises		577
28 Metastability and synchronization failure		580
28.1 Synchronization failure		580
28.2 Metastability		581
28.3 Probability of entering and leaving an illegal state		584
28.4 Demonstration of metastability		585
Summary		589
Bibliographic notes		590
Exercises		590
29 Synchronizer design		592
29.1 Where are synchronizers used?		592
29.2 Brute-force synchronizer		593
29.3 The problem with multi-bit signals		595
29.4 FIFO synchronizer		596
Summary		604
Bibliographic notes		605
Exercises		605

Part VIII Appendix: VHDL coding style
and syntax guide

Appendix A: VHDL coding style	611
A.1 Basic principles	611
A.2 All state should be in explicitly declared registers	612
A.3 Define combinational design entities so that they are easy to read	614
A.4 Assign all signals under all conditions	615
A.5 Keep design entities small	617
A.6 Large design entities should be structural	617
A.7 Use descriptive signal names	618
A.8 Use symbolic names for subfields of signals	618
A.9 Define constants	618

A.10	Comments should describe intention and give rationale, not state the obvious	619
A.11	Never forget you are defining hardware	620
A.12	Read and be a critic of VHDL code	620
Appendix B: VHDL syntax guide		622
B.1	Comments, identifiers, and keywords	623
B.2	Types	623
B.2.1	Std_logic	624
B.2.2	Boolean	624
B.2.3	Integer	624
B.2.4	Std_logic_vector	625
B.2.5	Subtypes	625
B.2.6	Enumeration	626
B.2.7	Arrays and records	626
B.2.8	Qualified expressions	627
B.3	Libraries, packages, and using multiple files	627
B.4	Design entities	628
B.5	Slices, concatenation, aggregates, operators, and expressions	629
B.6	Concurrent statements	631
B.6.1	Concurrent signal assignment	632
B.6.2	Component instantiation	634
B.7	Multiple signal drivers and resolution functions	636
B.8	Attributes	638
B.9	Process statements	640
B.9.1	The process sensitivity list and execution timing	641
B.9.2	Wait and report statements	644
B.9.3	If statements	644
B.9.4	Case and matching case statements	644
B.9.5	Signal and variable assignment statements	646
B.10	Synthesizable process statements	648
B.10.1	Type 1: purely combinational	649
B.10.2	Type 2: edge-sensitive	649
B.10.3	Type 3: edge-sensitive with asynchronous reset	650
	References	653
	Index of VHDL design entities	658
	Subject index	660

PREFACE

This book is intended to teach an undergraduate student to understand and design digital *systems*. It teaches the skills needed for current industrial digital system design using a hardware description language (VHDL) and modern CAD tools. Particular attention is paid to system-level issues, including factoring and partitioning digital systems, interface design, and interface timing. Topics needed for a deep understanding of digital circuits, such as timing analysis, metastability, and synchronization, are also covered. Of course, we cover the manual design of combinational and sequential logic circuits. However, we do not dwell on these topics because there is far more to digital system design than designing such simple modules.

Upon completion of a course using this book, students should be prepared to practice digital design in industry. They will lack experience, but they will have all of the tools they need for contemporary practice of this noble art. The experience will come with time.

This book has grown out of more than 25 years of teaching digital design to undergraduates (CS181 at Caltech, 6.004 at MIT, EE121 and EE108A at Stanford). It is also motivated by 35 years of experience designing digital systems in industry (Bell Labs, Digital Equipment, Cray, Avici, Velio Communications, Stream Processors, and NVIDIA). It combines these two experiences to teach what students need to know to function in industry in a manner that has been proven to work on generations of students. The VHDL guide in Appendix B is informed by nearly a decade of teaching VHDL to undergraduates at UBC (EECE 353 and EECE 259).

We wrote this book because we were unable to find a book that covered the system-level aspects of digital design. The vast majority of textbooks on this topic teach the manual design of combinational and sequential logic circuits and stop. While most texts today use a hardware description language, the vast majority teach a TTL-esque design style that, while appropriate in the era of 7400 quad NAND gate parts (the 1970s), does not prepare a student to work on the design of a three-billion-transistor GPU. Today's students need to understand how to factor a state machine, partition a design, and construct an interface with correct timing. We cover these topics in a simple way that conveys insight without getting bogged down in details.

Outline of the book

A flow chart showing the organization of the book and the dependences between chapters is shown in Figure 1. The book is divided into an introduction, five main sections, and chapters about style and verification. Appendix B provides a summary of VHDL-2008 syntax.

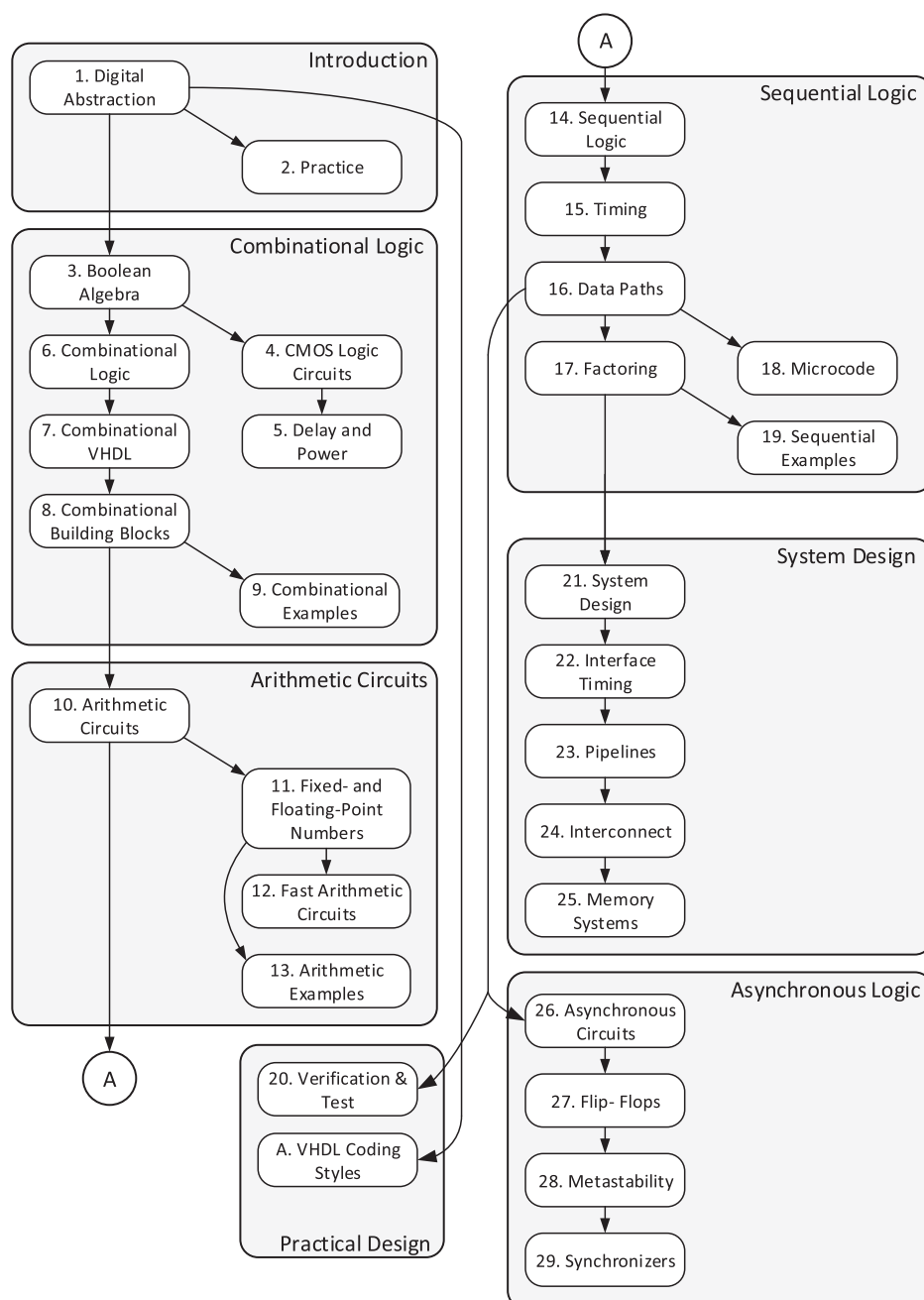


Figure 1. Organization of the book and dependences between chapters.

Part I Introduction

Chapter 1 introduces digital systems. It covers the representation of information as digital signals, noise margins, and the role of digital logic in the modern world. The practice of digital design in industry is described in Chapter 2. This includes the

design process, modern implementation technologies, computer-aided design tools, and Moore's law.

Part II Combinational logic

Chapters 3–9 deal with combinational logic circuits – digital circuits whose outputs depend only on the current values of their inputs. Boolean algebra, the theoretical underpinning of logic design, is discussed in Chapter 3. Switching logic and CMOS gate circuits are introduced in Chapter 4. Chapter 5 introduces simple models for calculating the delay and power of CMOS circuits. Manual methods for designing combinational circuits from basic gates are described in Chapter 6. Chapter 7 discusses how to automate the design process by coding behavioral descriptions of combinational logic in the VHDL hardware description language. Building blocks for combinational logic, decoders, multiplexers, etc. are described in Chapter 8, and several examples of combinational design are given in Chapter 9.

Part III Arithmetic circuits

Chapters 10–13 describe number systems and arithmetic circuits. Chapter 10 describes the basics of number representation and arithmetic circuits that perform the *four functions* $+$, $-$, \times , and \div on integers. Fixed-point and floating-point number representations and their accuracy are presented in Chapter 11. This chapter includes a discussion of floating-point unit design. Techniques for building fast arithmetic circuits, including carry look-ahead, Wallace trees, and Booth recoding, are described in Chapter 12. Finally, examples of arithmetic circuits and systems are presented in Chapter 13.

Part IV Synchronous sequential logic

Chapters 14–19 describe synchronous sequential logic circuits – sequential circuits whose state changes only on clock edges – and the process of designing finite-state machines. After describing the basics in Chapter 14, timing constraints are covered in Chapter 15. The design of *datapath* sequential circuits – whose behavior is described by an equation rather than a state table – is the topic of Chapter 16. Chapter 17 describes how to factor complex state machines into several smaller, simpler state machines. The concept of stored program control, and how to build finite-state machines using microcoded engines, is described in Chapter 18. This section closes with a number of examples in Chapter 19.

Part V Practical design

Chapter 20 and the Appendix discuss two important aspects of working on digital design projects. The process of verifying the correctness of logic and testing that it works after manufacturing are the topics of Chapter 20. The Appendix teaches the student proper VHDL coding style. It is a style that is readable, maintainable, and enables CAD tools to produce optimized hardware. Students should read this chapter before, during, and after writing their own VHDL.

Part VI System design

Chapters 21–25 discuss system design and introduce a systematic method for the design and analysis of digital systems. A six-step process for system design is introduced in Chapter 21. System-level timing and conventions for the timing of interfaces are discussed in Chapter 22. Chapter 23 describes pipelining of modules and systems, and includes several example pipelines. System interconnects including buses, crossbar switches, and networks are described in Chapter 24. A discussion of memory systems is given in Chapter 25.

Part VII Asynchronous logic

Chapters 26–29 discuss asynchronous sequential circuits – circuits whose state changes in response to any input change, without waiting for a clock edge. The basics of asynchronous design including flow-table analysis and synthesis and the problem of races are introduced in Chapter 26. Chapter 27 gives an example of these techniques, analyzing flip-flops and latches as asynchronous circuits. The problem of *metastability* and synchronization failure is described in Chapter 28. This section, and the book, closes with a discussion of synchronizer design – how to design circuits that safely move signals across asynchronous boundaries – in Chapter 29.

Teaching using this book

This book is suitable for use in a one-quarter (10-week) or one-semester (13-week) introductory course on digital systems design. It can also be used as the primary text of a second, advanced, course on digital systems.

There need not be any formal prerequisites for a course using this book. A good understanding of high-school-level mathematics is the only required preparation. Except for Chapters 5 and 28, the only place derivatives are used, the material does not require a knowledge of calculus. At Stanford, E40 (Introduction to Electrical Engineering) is a prerequisite for EE108A (Digital Systems I), but students often take EE108A without the prerequisite with no problems.

A one-quarter introductory course on digital systems design covers the material in Chapters 1, 3, 6, 7, 8, 10, (11), 14, 15, 16, (17), 21, 22, (23), 26, 28, and 29. For the one-quarter course we omit the details of CMOS circuits (Chapters 4 and 5), microcode (Chapter 18), and the more advanced systems topics (Chapters 24 and 25). The three chapters in parentheses are optional and can be skipped to give a slightly slower-paced course. In offerings of this course at Stanford, we typically administer two midterm examinations: one after covering Chapter 11, and the second after covering Chapter 22.

A one-semester introductory course on digital systems can use the three additional weeks to include the material on CMOS circuits and a few of the more advanced systems topics. A typical semester-long course covers the material in Chapters 1, 2, 3,

4, (5), 6, 7, 8, 9, 10, (11), 13, 14, 15, 16, (17), (18), (19), 21, 22, (23), (24), (25), 26, (27), 28, and 29.

This book can be used for an advanced course on digital systems design. Such a course covers the material from the introductory courses in more depth and includes advanced topics that were omitted from the introductory courses. Such a course usually includes a significant student project.

Materials

To support teaching with this book, the course website includes teaching materials: lecture slides, a series of laboratories, and solutions to selected exercises. The laboratories are intended to reinforce the material in the course and can be performed via simulation or a combination of simulation and implementation on FPGAs.

ACKNOWLEDGMENTS

We are deeply indebted to the many people who have made contributions to the creation of this book. This book has evolved over many years of teaching digital design at MIT (6.004) and Stanford (EE108A). We thank the many generations of students who took early versions of this class and provided feedback that led to constant refinement of our approach. Professors Subhasish Mitra, Phil Levis, and My Le have taught at Stanford using early versions of this material, and have provided valuable comments that led to many improvements. The course and book benefited from contributions by many great teaching assistants over the years. Paul Hartke, David Black-Shaffer, Frank Nothaft, and David Schneider deserve special thanks. Frank also deserves thanks for contributing to the exercise solutions. Teaching 6.004 at MIT with Gill Pratt, Greg Papadopolous, Steve Ward, Bert Halstead, and Anant Agarwal helped develop the approach to teaching digital design that is captured in this book. An early draft of the VHDL edition of this book was used for EECE 259 at the University of British Columbia (UBC). We thank the students who provided feedback that led to refinements in this version. The treatment of VHDL-2008 in Appendix B has been informed by several years of experience teaching earlier VHDL versions in EECE 353 at UBC using a set of slides originally developed by Professor Steve Wilton. Steve also provided helpful feedback on an early draft of the VHDL edition.

Julie Lancashire and Kerry Cahill at Cambridge University Press helped throughout the original Verilog edition, and Julie Lancashire, Karyn Bailey, and Jessica Murphy at Cambridge Press helped with the current VHDL edition. We thank Irene Pizzie for careful copy editing of the original Verilog edition and Abigail Jones for shepherding the original Verilog edition through the sometimes difficult passage from manuscript to finished project. We thank Steven Holt for careful copy editing of the present VHDL edition you see before you.

Finally, our families: Sharon, Jenny, Katie, and Liza Dally, and Jacki Armiak, Eric Harting, and Susanna Temkin, and Dayna and Ethan Aamodt have offered tremendous support and made significant sacrifices so we could have time to devote to writing.