

# Digital Design

## 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 Verilog 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 accurate, clear explanations, extensive examples, and online Verilog files. The teaching package is completed with lecture slides, labs, and a solutions manual for instructors. 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 has many years of experience working in industry and academia, previously holding positions at Bell Labs, Caltech and MIT, and consulting for Digital Equipment, Cray Research and Intel. 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. He has published over 200 papers in these areas, holds over 75 issued patents, and is an author of the textbooks *Digital Systems Engineering* and *Principles and Practices of Interconnection Networks*.

**R. Curtis Harting** is a Ph.D. candidate at 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. His primary research interest is in computer architecture, focusing on parallel, high-performance, and energy-efficient design.

‘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 Prof. 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 provides 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 to practitioners in the area. I recommend it highly.’

**Chuck Thacker**, *Microsoft*

Cambridge University Press  
978-0-521-19950-6 - Digital Design: A Systems Approach  
William J. Dally and R. Curtis Harting  
Frontmatter  
[More information](#)

# Digital Design

## A Systems Approach

**WILLIAM J. DALLY**  
**R. CURTIS HARTING**  
Stanford University



Cambridge University Press  
978-0-521-19950-6 - Digital Design: A Systems Approach  
William J. Dally and R. Curtis Harting  
Frontmatter  
[More information](#)

CAMBRIDGE UNIVERSITY PRESS  
Cambridge, New York, Melbourne, Madrid, Cape Town,  
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press  
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

[www.cambridge.org](http://www.cambridge.org)  
Information on this title: [www.cambridge.org/9780521199506](http://www.cambridge.org/9780521199506)

© Cambridge University Press 2012

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 2012

Printed in the United States of America

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

*Library of Congress Cataloging in Publication data*

Dally, William J.

Digital design : a systems approach / WILLIAM J. DALLY, R. CURTIS HARTING.  
pages cm

Includes bibliographical references and index.

ISBN 978-0-521-19950-6 (hardback)

1. Digital electronics. 2. Electronic systems. I. Harting, R. Curtis. II. Title.

TK7868.D5D328 2012

621.39'1 – dc23 2012021831

ISBN 978-0-521-19950-6 Hardback

Additional resources for this publication at [www.cambridge.org/9780521199506](http://www.cambridge.org/9780521199506)

---

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

<b>1 The digital abstraction</b>	<b>3</b>
1.1 Digital signals	3
1.2 Digital signals tolerate noise	5
1.3 Digital signals represent complex data	9
1.3.1 Representing the day of the year	10
1.3.2 Representing subtractive colors	11
1.4 Digital logic functions	12
1.5 Verilog description of digital circuits and systems	14
1.6 Digital logic in systems	15
Summary	16
Bibliographic notes	17
Exercises	17
<b>2 The practice of digital system design</b>	<b>21</b>
2.1 The design process	21
2.1.1 Specification	22
2.1.2 Concept development and feasibility	23
2.1.3 Partitioning and detailed design	26
2.1.4 Verification	26
2.2 Digital systems are built from chips and boards	27
2.3 Computer-aided design tools	32
2.4 Moore's law and digital system evolution	33
Summary	35
Bibliographic notes	36
Exercises	36

## Part II Combinational logic

<b>3</b>	<b>Boolean algebra</b>	41
3.1	Axioms	41
3.2	Properties	42
3.3	Dual functions	44
3.4	Normal form	45
3.5	From equations to gates	47
3.6	Boolean expressions in Verilog	49
	Summary	51
	Bibliographic notes	52
	Exercises	52
<b>4</b>	<b>CMOS logic circuits</b>	55
4.1	Switch logic	55
4.2	Switch model of MOS transistors	59
4.3	CMOS gate circuits	66
4.3.1	Basic CMOS gate circuit	66
4.3.2	Inverters, NANDs, and NORs	67
4.3.3	Complex gates	69
4.3.4	Tri-state circuits	72
4.3.5	Circuits to avoid	74
	Summary	75
	Bibliographic notes	75
	Exercises	76
<b>5</b>	<b>Delay and power of CMOS circuits</b>	79
5.1	Delay of static CMOS gates	79
5.2	Fan-out and driving large loads	82
5.3	Fan-in and logical effort	84
5.4	Delay calculation	87
5.5	Optimizing delay	89
5.6	Wire delay	92
5.7	Power dissipation in CMOS circuits	95
5.7.1	Dynamic power	96
5.7.2	Static power	97
5.7.3	Power scaling	97
	Summary	98
	Bibliographic notes	99
	Exercises	99

<b>6</b>	<b>Combinational logic design</b>	103
6.1	Combinational logic	103
6.2	Closure	104
6.3	Truth tables, minterms, and normal form	105
6.4	Implicants and cubes	108
6.5	Karnaugh maps	112
6.6	Covering a function	114
6.7	From a cover to gates	115
6.8	Incompletely specified functions	116
6.9	Product-of-sums implementation	118
6.10	Hazards	120
	Summary	122
	Bibliographic notes	123
	Exercises	123
 <b>7</b>	 <b>Verilog descriptions of combinational logic</b>	 128
7.1	The prime number circuit in Verilog	128
7.1.1	A Verilog module	129
7.1.2	The <code>case</code> statement	130
7.1.3	The <code>case</code> statement	132
7.1.4	The <code>assign</code> statement	133
7.1.5	Structural description	134
7.1.6	The decimal prime number function	136
7.2	A testbench for the prime number circuit	137
7.3	Example: a seven-segment decoder	141
	Summary	146
	Bibliographic notes	146
	Exercises	147
 <b>8</b>	 <b>Combinational building blocks</b>	 150
8.1	Multi-bit notation	150
8.2	Decoders	150
8.3	Multiplexers	155
8.4	Encoders	161
8.5	Arbiters and priority encoders	165
8.6	Comparators	170
8.7	Shifters	173
8.8	Read-only memories	173
8.9	Read-write memories	177
8.10	Programmable logic arrays	180
8.11	Data sheets	181
8.12	Intellectual property	182

Summary	183
Bibliographic notes	184
Exercises	184
<b>9 Combinational examples</b>	<b>187</b>
9.1 Multiple-of-3 circuit	187
9.2 Tomorrow circuit	189
9.3 Priority arbiter	192
9.4 Tic-tac-toe	193
Summary	201
Exercises	202
<b>Part III Arithmetic circuits</b>	
<b>10 Arithmetic circuits</b>	<b>207</b>
10.1 Binary numbers	207
10.2 Binary addition	210
10.3 Negative numbers and subtraction	216
10.4 Multiplication	223
10.5 Division	226
Summary	230
Exercises	230
<b>11 Fixed- and floating-point numbers</b>	<b>236</b>
11.1 Representation error: accuracy, precision, and resolution	236
11.2 Fixed-point numbers	238
11.2.1 Representation	238
11.2.2 Operations	241
11.3 Floating-point numbers	243
11.3.1 Representation	243
11.3.2 Denormalized numbers and gradual underflow	245
11.3.3 Floating-point multiplication	245
11.3.4 Floating-point addition/subtraction	247
Summary	250
Bibliographic note	251
Exercises	251
<b>12 Fast arithmetic circuits</b>	<b>255</b>
12.1 Carry look-ahead	255
12.2 Booth recoding	260
12.3 Wallace trees	265
12.4 Synthesis notes	269



Summary	271
Bibliographic notes	272
Exercises	272
<b>13 Arithmetic examples</b>	<b>275</b>
13.1 Complex multiplication	275
13.2 Converting between fixed- and floating-point formats	276
13.2.1 Floating-point format	276
13.2.2 Fixed- to floating-point conversion	279
13.2.3 Floating- to fixed-point conversion	282
13.3 FIR filter	283
Summary	284
Bibliographic note	285
Exercises	285
<b>Part IV Synchronous sequential logic</b>	
<b>14 Sequential logic</b>	<b>291</b>
14.1 Sequential circuits	291
14.2 Synchronous sequential circuits	293
14.3 Traffic-light controller	296
14.4 State assignment	299
14.5 Implementation of finite-state machines	300
14.6 Verilog implementation of finite-state machines	303
Summary	309
Bibliographic notes	309
Exercises	310
<b>15 Timing constraints</b>	<b>314</b>
15.1 Propagation and contamination delay	314
15.2 The D flip-flop	317
15.3 Setup- and hold-time constraints	318
15.4 The effect of clock skew	321
15.5 Timing examples	323
15.6 Timing and logic synthesis	324
Summary	326
Bibliographic notes	327
Exercises	327
<b>16 Datapath sequential logic</b>	<b>331</b>
16.1 Counters	331
16.1.1 A simpler counter	331

16.1.2	Up/down/load counter	333
16.1.3	A timer	336
16.2	Shift registers	338
16.2.1	A simple shift register	338
16.2.2	Left/right/load (LRL) shift register	339
16.2.3	Universal shifter/counter	340
16.3	Control and data partitioning	342
16.3.1	Example: vending machine FSM	342
16.3.2	Example: combination lock	348
	Summary	356
	Exercises	357
<b>17</b>	<b>Factoring finite-state machines</b>	<b>360</b>
17.1	A light flasher	360
17.2	Traffic-light controller	367
	Summary	378
	Exercises	378
<b>18</b>	<b>Microcode</b>	<b>383</b>
18.1	Simple microcoded FSM	383
18.2	Instruction sequencing	387
18.3	Multi-way branches	394
18.4	Multiple instruction types	397
18.5	Microcode subroutines	400
18.6	Simple computer	403
	Summary	406
	Bibliographic notes	408
	Exercises	411
<b>19</b>	<b>Sequential examples</b>	<b>414</b>
19.1	Divide-by-3 counter	414
19.2	SOS detector	415
19.3	Tic-tac-toe game	422
19.4	Huffman encoder/decoder	422
19.4.1	Huffman encoder	423
19.4.2	Huffman decoder	427
	Summary	430
	Bibliographic note	430
	Exercises	430

## Part V Practical design

<b>20</b>	<b>Verification and test</b>	435
	20.1 Design verification	435
	20.1.1 Verification coverage	435
	20.1.2 Types of tests	437
	20.1.3 Static timing analysis	437
	20.1.4 Formal verification	438
	20.1.5 Bug tracking	438
	20.2 Test	438
	20.2.1 Fault models	439
	20.2.2 Combinational testing	439
	20.2.3 Testing redundant logic	440
	20.2.4 Scan	441
	20.2.5 Built-in-self-test (BIST)	442
	20.2.6 Characterization	443
	Summary	444
	Bibliographic notes	444
	Exercises	445

## Part VI System design

<b>21</b>	<b>System-level design</b>	449
	21.1 System design process	449
	21.2 Specification	450
	21.2.1 Pong	451
	21.2.2 DES cracker	452
	21.2.3 Music player	454
	21.3 Partitioning	456
	21.3.1 Pong	456
	21.3.2 DES cracker	457
	21.3.3 Music synthesizer	457
	Summary	459
	Bibliographic notes	459
	Exercises	460
<b>22</b>	<b>Interface and system-level timing</b>	461
	22.1 Interface timing	461
	22.1.1 Always valid timing	461
	22.1.2 Periodically valid signals	462
	22.1.3 Flow control	463

22.2	Interface partitioning and selection	465
22.3	Serial and packetized interfaces	465
22.4	Isochronous timing	468
22.5	Timing tables	469
22.5.1	Event flow	470
22.5.2	Pipelining and anticipatory timing	471
22.6	Interface and timing examples	471
22.6.1	Pong	471
22.6.2	DES cracker	472
22.6.3	Music player	473
	Summary	476
	Exercises	476
<b>23</b>	<b>Pipelines</b>	479
23.1	Basic pipelining	479
23.2	Example pipelines	482
23.3	Example: pipelining a ripple-carry adder	484
23.4	Pipeline stalls	488
23.5	Double buffering	489
23.6	Load balance	494
23.7	Variable loads	494
23.8	Resource sharing	499
	Summary	500
	Bibliographic notes	500
	Exercises	501
<b>24</b>	<b>Interconnect</b>	504
24.1	Abstract interconnect	504
24.2	Buses	505
24.3	Crossbar switches	507
24.4	Interconnection networks	510
	Summary	512
	Bibliographic notes	512
	Exercises	513
<b>25</b>	<b>Memory systems</b>	515
25.1	Memory primitives	515
25.1.1	SRAM arrays	515
25.1.2	DRAM chips	517
25.2	Bit-slicing and banking memory	519
25.3	Interleaved memory	521
25.4	Caches	524
	Summary	528

Bibliographic notes	528
Exercises	528
<b>Part VII Asynchronous logic</b>	
<b>26 Asynchronous sequential circuits</b>	<b>533</b>
26.1 Flow-table analysis	533
26.2 Flow-table synthesis: the toggle circuit	536
26.3 Races and state assignment	540
Summary	544
Bibliographic notes	545
Exercises	545
<b>27 Flip-flops</b>	<b>548</b>
27.1 Inside a latch	548
27.2 Inside a flip-flop	551
27.3 CMOS latches and flip-flops	553
27.4 Flow-table derivation of the latch	555
27.5 Flow-table synthesis of a D-flip-flop	557
Summary	559
Bibliographic notes	559
Exercises	559
<b>28 Metastability and synchronization failure</b>	<b>563</b>
28.1 Synchronization failure	563
28.2 Metastability	564
28.3 Probability of entering and leaving an illegal state	567
28.4 Demonstration of metastability	570
Summary	573
Bibliographic notes	574
Exercises	574
<b>29 Synchronizer design</b>	<b>576</b>
29.1 Where are synchronizers used?	576
29.2 Brute-force synchronizer	577
29.3 The problem with multi-bit signals	579
29.4 FIFO synchronizer	581
Summary	588
Bibliographic notes	589
Exercises	589

<b>Appendix A: Verilog coding style</b>	592
A.1 Basic principles	592
A.2 All state should be in explicitly declared registers	593
A.3 Define combinational modules so they are easy to read	594
A.4 Assign all variables under all conditions	596
A.5 Keep modules small	597
A.6 Large modules should be structural	599
A.7 Use descriptive signal names	599
A.8 Use symbolic names for subfields of signals	599
A.9 Define constants	600
A.10 Comments should describe intention and give rationale, not state the obvious	601
A.11 Never forget you are defining hardware	602
A.12 Read and be a critic of Verilog code	602
References	604
Index of Verilog modules	609
Subject index	611

## 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 (Verilog) 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, and 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.

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-esqe 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.

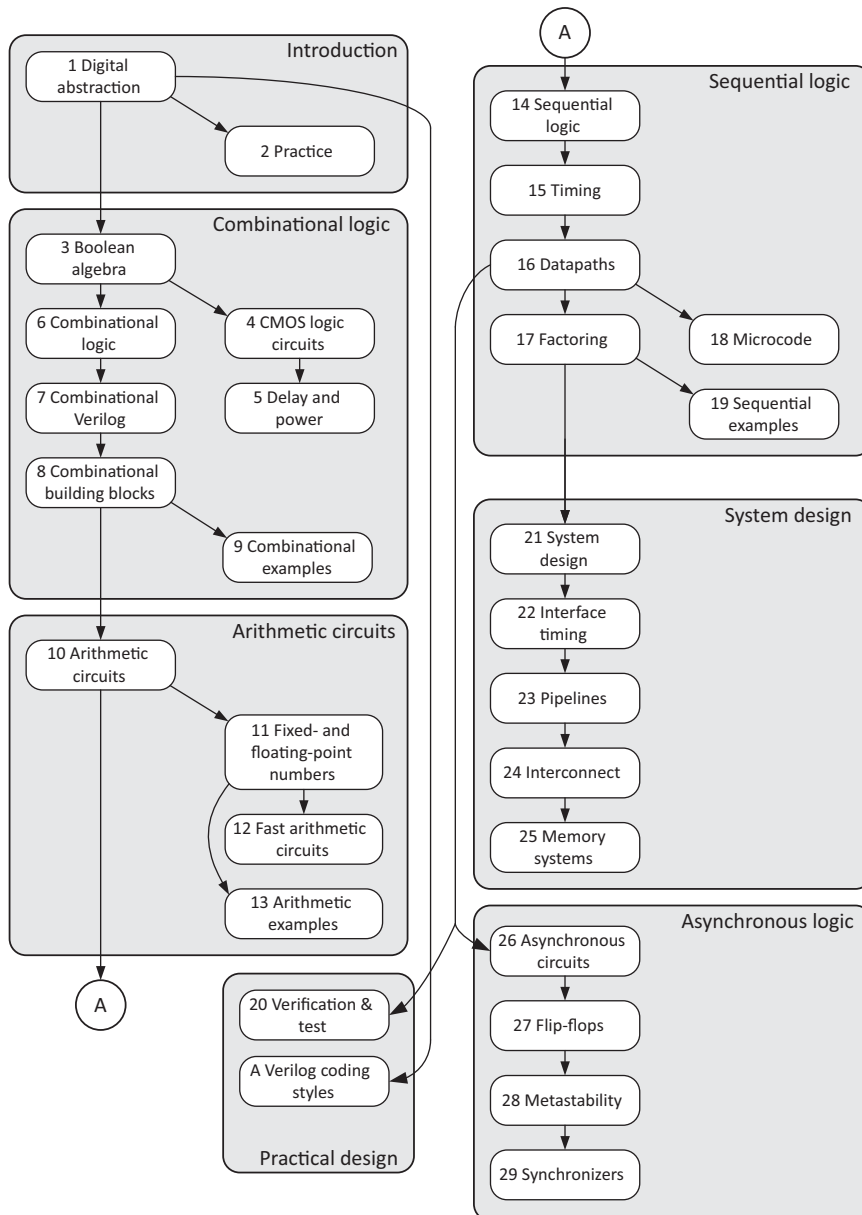


Figure 1. Organization of the book and dependence between chapters.

## 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 Verilog 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 Verilog 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 Verilog.

## 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 midterms: 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, 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 including: 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.

Julie Lancashire and Kerry Cahill at Cambridge University Press have helped us throughout the project. We thank Irene Pizzie for careful copy editing and Abigail Jones for shepherding the book through the sometimes difficult passage from manuscript to finished project.

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