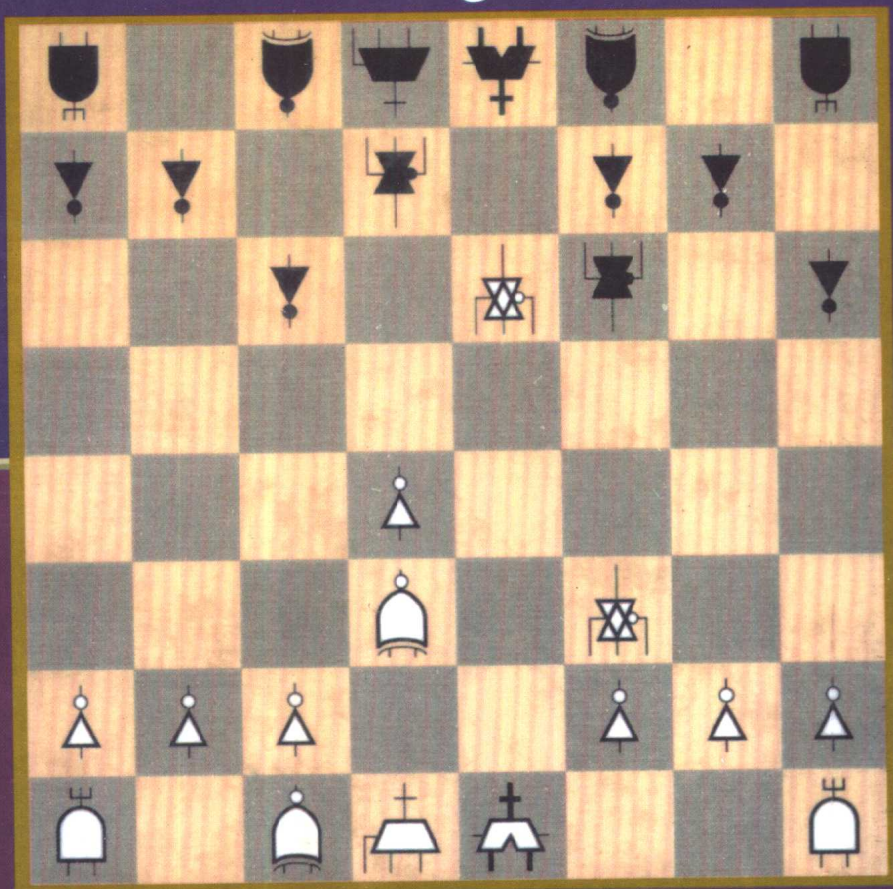



时代教育 • 国外高校优秀教材精选

# 数字逻辑设计(VHDL)基础

## Fundamentals of Digital Logic with VHDL Design (英文版)



(美) 斯蒂芬·布朗 (Stephen Brown) 著  
兹翁科·弗拉内希奇 (Zvonko Vranesic)

 机械工业出版社  
CHINA MACHINE PRESS

 McGraw  
Hill

Education

时代教育·国外高校优秀教材精选

# 数字逻辑设计(VHDL)基础 (英文版)

**Fundamentals of Digital Logic with VHDL Design**

(美) 斯蒂芬·布朗 (Stephen Brown) 著  
兹翁科·弗拉内希奇 (Zvonko Vranesic)



机械工业出版社



Education

Stephen Brown, Zvonko Vranesic

**Fundamentals of Digital Logic with VHDL Design**

ISBN: 0-07-012591-0

Copyright © 2000 by the McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed in any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Authorized English language reprint edition jointly published by McGraw-Hill Education (Asia) Co. and China Machine Press. This edition is authorized for sale in the People's Republic of China only, excluding Hong Kong, Macao SAR and Taiwan. Unauthorized export of this edition is a violation of the Copyright Act. Violation of this law is subject to Civil and Criminal Penalties.

本书英文影印版由机械工业出版社和美国麦格劳-希尔教育出版(亚洲)公司合作出版。此版本仅限在中华人民共和国境内(不包括香港、澳门特别行政区及台湾地区)销售。未经许可之出口,视为违反著作权法,将受法律之制裁。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 McGraw-Hill 公司激光防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号:图片:01-2002-1878 号

**图书在版编目(CIP)数据**

数字逻辑设计(VHDL)基础=Fundamentals of Digital Logic with VHDL Design/(美)朗(Brow, S.), (美)弗拉内希奇(Vranesic, Z.)著. —北京:机械工业出版社, 2002. 8

时代教育. 国外高校优秀教材精选

ISBN 7-111-10640-7

I. 数… II. ①朗…②弗… III. ①数字逻辑—逻辑设计—高等学校—教材—英文②硬件描述语言, VHDL—程序设计—高等学校—教材—英文 IV. TP302. 2

中国版本图书馆 CIP 数据核字(2002)第 053664 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑:周娟 封面设计:鞠杨

责任印制:付方敏

北京铭成印刷有限公司印刷·新华书店北京发行所发行

2002 年 5 月第 1 版第 1 次印刷

1000mm×1400mm B5·26.625 印张·1029 千字

定价:63.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

本社购书热线电话(010) 68993821、68326677—2527

封面无防伪标均为盗版

# 国外高校优秀教材审定委员会

主任委员：杨叔子

委员（按姓氏笔画为序）：

王先逵	王大康	白峰杉	史荣昌	朱孝禄
陆启韶	张润琦	张策	张三慧	张福润
张延华	吴宗泽	吴麒麟	张宋心	李俊峰
佘远斌	陈文楷	陈立周	俞正光	赵汝嘉
翁海珊	龚光鲁	章栋恩	黄永畅	谭泽光

# 序

CPLD 和 FPGA 这类复杂的用户可编程逻辑器件，正以其编程方便、集成度高、速度快、价格低等特点越来越受到电子设计者的青睐。VHDL 硬件描述语言是数字逻辑电路设计者的得力工具，是现代电子设计的基础语言。硬件描述语言的种类很多，成为 IEEE 标准的语言有两种：VHDL 和 VerilogHDL 语言。

目前已出版的很多关于 VHDL 语言的教材和参考书都是从语法规则、程序结构两个方面介绍的，从语言基础讲述数字逻辑的程序设计。

本书则是数字逻辑设计方面的入门教材，在介绍传统的数字逻辑电路基本概念的同时，引入 VHDL 语言的描述方法。书中采用 MAX+Plus II CAD 软件工具，其目的是在基本概念和实际应用之间，通过电子设计自动化（EDA）的 CAD 方法建立它们的桥梁，使初学者在学习数字逻辑电路的同时逐步掌握 VHDL 语言的描述方法。全书给出 100 多个设计例子，可边学习，边上机实验，在实验中掌握 VHDL 语言的描述方法，使本书更适合初学者自学。书中的例子由浅入深，对较难的设计例子进行了清楚的说明，从而构成本书的特色。本书由于起点较低，也可作为双语教材使用。

陈文楷

北京工业大学

2002 年 4 月

# 出版说明

随着我国加入 WTO，国际间的竞争越来越激烈，而国际间的竞争实际上也就是人才的竞争、教育的竞争。为了加快培养具有国际竞争力的高水平技术人才，加快我国教育改革的步伐，教育部近来出台了一系列倡导高校开展双语教学、引进原版教材的政策。以此为契机，机械工业出版社拟于近期推出一系列国外影印版教材，其内容涉及高等学校公共基础课，以及机、电、信息领域的专业基础课和专业课。

引进国外优秀原版教材，在有条件的学校推动开展英语授课或双语教学，自然也引进了先进的教学思想和教学方法，这对提高我国自编教材的水平，加强学生的英语实际应用能力，使我国的高等教育尽快与国际接轨，必将起到积极的推动作用。

为了做好教材的引进工作，机械工业出版社特别成立了由著名专家组成的国外高校优秀教材审定委员会。这些专家对实施双语教学做了深入细致的调查研究，对引进原版教材提出许多建设性意见，并慎重地对每一本将要引进的原版教材一审再审，精选再精选，确认教材本身的质量水平，以及权威性和先进性，以期所引进的原版教材能适应我国学生的外语水平和学习特点。在引进工作中，审定委员会还结合我国高校教学课程体系的设置和要求，对原版教材的教学思想和方法的先进性、科学性严格把关，同时尽量考虑原版教材的系统性和经济性。

这套教材出版后，我们将根据各高校的双语教学计划，举办原版教材的教师培训，及时地将其推荐给各高校选用。希望高校师生在使用教材后及时反馈意见和建议，使我们更好地为教学改革服务。

机械工业出版社

2002 年 3 月

# CONTENTS

序

出版说明

## Chapter 1

### DESIGN CONCEPTS 1

- 1.1 Digital Hardware 2
  - 1.1.1 Standard Chips 4
  - 1.1.2 Programmable Logic Devices 4
  - 1.1.3 Custom-Designed Chips 5
- 1.2 The Design Process 6
- 1.3 Design of Digital Hardware 8
  - 1.3.1 Basic Design Loop 8
  - 1.3.2 Design of a Digital Hardware Unit 9
- 1.4 Logic Circuit Design in this Book 12
- 1.5 Theory and Practice 14
  - References 15

## Chapter 2

### INTRODUCTION TO LOGIC CIRCUITS 17

- 2.1 Variables and Functions 18
- 2.2 Inversion 21
- 2.3 Truth Tables 22
- 2.4 Logic Gates and Networks 23
  - 2.4.1 Analysis of a Logic Network 24
- 2.5 Boolean Algebra 27
  - 2.5.1 The Venn Diagram 30
  - 2.5.2 Notation and Terminology 34
  - 2.5.3 Precedence of Operations 34
- 2.6 Synthesis Using AND, OR, and NOT Gates 35
  - 2.6.1 Sum-of-Products and Product-of-Sums Forms 37
- 2.7 Design Examples 41
  - 2.7.1 Three-Way Light Control 42
  - 2.7.2 Multiplexer Circuit 43
- 2.8 Introduction to CAD Tools 45
  - 2.8.1 Design Entry 46
  - 2.8.2 Synthesis 48
  - 2.8.3 Functional Simulation 49
  - 2.8.4 Summary 49

### 2.9 Introduction to VHDL 51

- 2.9.1 Representation of Digital Signals in VHDL 52
- 2.9.2 Writing Simple VHDL Code 52
- 2.9.3 How *Not* to Write VHDL Code 54
- 2.10 Concluding Remarks 55
  - Problems 56
  - References 60

## Chapter 3

### IMPLEMENTATION TECHNOLOGY 61

- 3.1 Transistor Switches 63
- 3.2 NMOS Logic Gates 65
- 3.3 CMOS Logic Gates 68
  - 3.3.1 Speed of Logic Gate Circuits 75
- 3.4 Negative Logic System 76
- 3.5 Standard Chips 77
  - 3.5.1 7400-Series Standard Chips 77
- 3.6 Programmable Logic Devices 81
  - 3.6.1 Programmable Logic Array (PLA) 81
  - 3.6.2 Programmable Array Logic (PAL) 84
  - 3.6.3 Programming of PLAs and PALs 86
  - 3.6.4 Complex Programmable Logic Devices (CPLDs) 88
  - 3.6.5 Field-Programmable Gate Arrays 92
  - 3.6.6 Using CAD Tools to Implement Circuits in CPLDs and FPGAs 96
- 3.7 Custom Chips, Standard Cells, and Gate Arrays 97
- 3.8 Practical Aspects 100
  - 3.8.1 MOSFET Fabrication and Behavior 100
  - 3.8.2 MOSFET On-Resistance 104
  - 3.8.3 Voltage Levels in Logic Gates 105
  - 3.8.4 Noise Margin 107
  - 3.8.5 Dynamic Operation of Logic Gates 108
  - 3.8.6 Power Dissipation in Logic Gates 111
  - 3.8.7 Passing 1s and 0s Through Transistor Switches 112
  - 3.8.8 Fan-in and Fan-out in Logic Gates 114

- 3.9 Transmission Gates 120
  - 3.9.1 Exclusive-OR Gates 121
  - 3.9.2 Multiplexer Circuit 122
- 3.10 Implementation Details for SPLDs, CPLDs, and FPGAs 123
  - 3.10.1 Implementation in FPGAs 129
- 3.11 Concluding Remarks 131
  - Problems 132
  - References 141

## **Chapter 4**

### **OPTIMIZED IMPLEMENTATION OF LOGIC FUNCTIONS 143**

- 4.1 Karnaugh Map 144
- 4.2 Strategy for Minimization 152
  - 4.2.1 Terminology 153
  - 4.2.2 Minimization Procedure 154
- 4.3 Minimization of Product-of-Sums Forms 158
- 4.4 Incompletely Specified Functions 160
- 4.5 Multiple-Output Circuits 161
- 4.6 NAND and NOR Logic Networks 165
- 4.7 Multilevel Synthesis 167
  - 4.7.1 Factoring 168
  - 4.7.2 Functional Decomposition 171
  - 4.7.3 Multilevel NAND and NOR Circuits 177
- 4.8 Analysis of Multilevel Circuits 180
- 4.9 Cubical Representation 185
  - 4.9.1 Cubes and Hypercubes 185
- 4.10 Minimization Using Cubical Representation 189
  - 4.10.1 Generation of Prime Implicants 189
  - 4.10.2 Determination of Essential Prime Implicants 192
  - 4.10.3 Complete Procedure for Finding a Minimal Cover 194
- 4.11 Practical Considerations 196
- 4.12 CAD Tools 197
  - 4.12.1 Logic Synthesis and Optimization 198
  - 4.12.2 Physical Design 199
  - 4.12.3 Timing Simulation 201
  - 4.12.4 Summary of Design Flow 202
  - 4.12.5 Examples of Circuits Synthesized from VHDL Code 204
- 4.13 Concluding Remarks 210
  - Problems 211
  - References 214

## **Chapter 5**

### **NUMBER REPRESENTATION AND ARITHMETIC CIRCUITS 217**

- 5.1 Positional Number Representation 218
  - 5.1.1 Unsigned Integers 218
  - 5.1.2 Conversion between Decimal and Binary Systems 219
  - 5.1.3 Octal and Hexadecimal Representations 220
- 5.2 Addition of Unsigned Numbers 222
  - 5.2.1 Decomposed Full-Adder 226
  - 5.2.2 Ripple-Carry Adder 227
  - 5.2.3 Design Example 228
- 5.3 Signed Numbers 228
  - 5.3.1 Negative Numbers 228
  - 5.3.2 Addition and Subtraction 232
  - 5.3.3 Adder and Subtractor Unit 236
  - 5.3.4 Radix-Complement Schemes 237
  - 5.3.5 Arithmetic Overflow 241
  - 5.3.6 Performance Issues 242
- 5.4 Fast Adders 243
  - 5.4.1 Carry-Lookahead Adder 243
- 5.5 Design of Arithmetic Circuits Using CAD Tools 250
  - 5.5.1 Design of Arithmetic Circuits Using Schematic Capture 250
  - 5.5.2 Design of Arithmetic Circuits Using VHDL 253
  - 5.5.3 Representation of Numbers in VHDL Code 256
  - 5.5.4 Arithmetic Assignment Statements 258
- 5.6 Multiplication 262
  - 5.6.1 Array Multiplier for Unsigned Numbers 263
  - 5.6.2 Multiplication of Signed Numbers 264
- 5.7 Other Number Representations 267
  - 5.7.1 Fixed-Point Numbers 267
  - 5.7.2 Floating-Point Numbers 267
  - 5.7.3 Binary-Coded-Decimal Representation 269
- 5.8 ASCII Character Code 273
  - Problems 276
  - References 280



**Chapter 6****COMBINATIONAL-CIRCUIT  
BUILDING BLOCKS 281**

- 6.1 Multiplexers 282
  - 6.1.1 Synthesis of Logic Functions Using Multiplexers 287
  - 6.1.2 Multiplexer Synthesis Using Shannon's Expansion 288
- 6.2 Decoders 295
  - 6.2.1 Demultiplexers 298
- 6.3 Encoders 300
  - 6.3.1 Binary Encoders 300
  - 6.3.2 Priority Encoders 301
- 6.4 Code Converters 302
- 6.5 Arithmetic Comparison Circuits 304
- 6.6 VHDL for Combinational Circuits 304
  - 6.6.1 Assignment Statements 305
  - 6.6.2 Selected Signal Assignment 305
  - 6.6.3 Conditional Signal Assignment 308
  - 6.6.4 Generate Statements 312
  - 6.6.5 Concurrent and Sequential Assignment Statements 315
  - 6.6.6 Process Statement 315
  - 6.6.7 Case Statement 321
- 6.7 Concluding Remarks 324
  - Problems 326
  - References 330

**Chapter 7****FLIP-FLOPS, REGISTERS,  
COUNTERS, AND A SIMPLE  
PROCESSOR 331**

- 7.1 Basic Latch 333
- 7.2 Gated SR Latch 335
  - 7.2.1 Gated SR Latch with NAND Gates 337
- 7.3 Gated D Latch 338
  - 7.3.1 Effects of Propagation Delays 340
- 7.4 Master-Slave and Edge-Triggered D Flip-Flops 341
  - 7.4.1 Master-Slave D Flip-Flop 341
  - 7.4.2 Edge-Triggered D Flip-Flop 342
  - 7.4.3 D Flip-Flops with Clear and Preset 344
- 7.5 T Flip-Flop 346
  - 7.5.1 Configurable Flip-Flops 349
- 7.6 JK Flip-Flop 349

- 7.7 Summary of Terminology 350

## 7.8 Registers 350

- 7.8.1 Shift Register 351
- 7.8.2 Parallel-Access Shift Register 352

## 7.9 Counters 353

- 7.9.1 Asynchronous Counters 353
- 7.9.2 Synchronous Counters 356
- 7.9.3 Counters with Parallel Load 360

## 7.10 Reset Synchronization 360

## 7.11 Other Types of Counters 364

- 7.11.1 BCD Counter 364
- 7.11.2 Ring Counter 365
- 7.11.3 Johnson Counter 366
- 7.11.4 Remarks on Counter Design 367

## 7.12 Using Storage Elements with CAD Tools 367

- 7.12.1 Including Storage Elements in Schematics 367
- 7.12.2 Using Latches and Flip-Flops in VHDL Code 370
- 7.12.3 Using VHDL Sequential Statements for Storage Elements 370

## 7.13 Using Registers and Counters with CAD Tools 375

- 7.13.1 Including Registers and Counters in Schematics 375
- 7.13.2 Registers and Counters in VHDL Code 378
- 7.13.3 Using VHDL Sequential Statements for Registers and Counters 379

## 7.14 Design Examples 387

- 7.14.1 Bus Structure 387
- 7.14.2 Simple Processor 400
- 7.14.3 Reaction Timer 413

## 7.15 Concluding Remarks 418

- Problems 418
- References 424

**Chapter 8****SYNCHRONOUS SEQUENTIAL  
CIRCUITS 427**

## 8.1 Basic Design Steps 429

- 8.1.1 State Diagram 429
- 8.1.2 State Table 431
- 8.1.3 State Assignment 431
- 8.1.4 Choice of Flip-Flops and Derivation of Next-State and Output Expressions 433

- 8.1.5 Timing Diagram 435
- 8.1.6 Summary of Design Steps 436
- 8.2 State-Assignment Problem 440
  - 8.2.1 One-Hot Encoding 442
- 8.3 Mealy State Model 444
- 8.4 Design of Finite State Machines Using CAD Tools 449
  - 8.4.1 VHDL Code for Moore-Type FSMs 449
  - 8.4.2 Synthesis of VHDL Code 451
  - 8.4.3 Simulating and Testing the Circuit 454
  - 8.4.4 An Alternative Style of VHDL Code 456
  - 8.4.5 Summary of Design Steps When Using CAD Tools 456
  - 8.4.6 Specifying the State Assignment in VHDL Code 458
  - 8.4.7 Specification of Mealy FSMs Using VHDL 459
- 8.5 Serial Adder Example 463
  - 8.5.1 Mealy-Type FSM for Serial Adder 463
  - 8.5.2 Moore-Type FSM for Serial Adder 464
  - 8.5.3 VHDL Code for the Serial Adder 467
- 8.6 State Minimization 470
  - 8.6.1 Partitioning Minimization Procedure 473
  - 8.6.2 Incompletely Specified FSMs 480
- 8.7 Design of a Counter Using the Sequential Circuit Approach 482
  - 8.7.1 State Diagram and State Table for a Modulo-8 Counter 482
  - 8.7.2 State Assignment 482
  - 8.7.3 Implementation Using D-Type Flip-Flops 484
  - 8.7.4 Implementation Using JK-Type Flip-Flops 485
  - 8.7.5 Example—A Different Counter 489
- 8.8 FSM as an Arbiter Circuit 492
  - 8.8.1 Implementation of the Arbiter Circuit 496
  - 8.8.2 Minimizing the Output Delays for an FSM 499
  - 8.8.3 Summary 499
- 8.9 Analysis of Synchronous Sequential Circuits 500
- 8.10 Algorithmic State Machine (ASM) Charts 504
- 8.11 Formal Model for Sequential Circuits 507
- 8.12 Concluding Remarks 509
  - Problems 509
  - References 513

## Chapter 9

### ASYNCHRONOUS SEQUENTIAL CIRCUITS 515

- 9.1 Asynchronous Behavior 516
- 9.2 Analysis of Asynchronous Circuits 519
- 9.3 Synthesis of Asynchronous Circuits 528
- 9.4 State Reduction 540
- 9.5 State Assignment 555
  - 9.5.1 Transition Diagram 558
  - 9.5.2 Exploiting Unspecified Next-State Entries 561
  - 9.5.3 State Assignment Using Additional State Variables 565
  - 9.5.4 One-Hot State Assignment 569
- 9.6 Hazards 571
  - 9.6.1 Static Hazards 572
  - 9.6.2 Dynamic Hazards 576
  - 9.6.3 Significance of Hazards 578
- 9.7 A Complete Design Example 579
  - 9.7.1 The Vending-Machine Controller 579
- 9.8 Concluding Remarks 583
  - Problems 585
  - References 590

## Chapter 10

### DIGITAL SYSTEM DESIGN 591

- 10.1 Building Block Circuits 592
  - 10.1.1 Flip-Flops and Registers with Enable Inputs 592
  - 10.1.2 Shift Registers with Enable Inputs 593
  - 10.1.3 Static Random Access Memory (SRAM) 595
  - 10.1.4 SRAM Blocks in PLDs 600
- 10.2 Design Examples 600
  - 10.2.1 A Bit-Counting Circuit 600
  - 10.2.2 ASM Chart Implied Timing Information 601
  - 10.2.3 Shift-and-Add Multiplier 603
  - 10.2.4 Divider 612
  - 10.2.5 Arithmetic Mean 623
  - 10.2.6 Sort Operation 629
- 10.3 Clock Synchronization 639
  - 10.3.1 Clock Skew 640
  - 10.3.2 Flip-Flop Timing Parameters 641

- 10.3.3 Asynchronous Inputs to Flip-Flops 644
- 10.3.4 Switch Debouncing 645
- 10.4 Concluding Remarks 645
- Problems 647
- References 651

## Chapter 11

### TESTING OF LOGIC CIRCUITS 653

- 11.1 Fault Model 654
  - 11.1.1 Stuck-at Model 654
  - 11.1.2 Single and Multiple Faults 655
  - 11.1.3 CMOS Circuits 655
- 11.2 Complexity of a Test Set 655
- 11.3 Path Sensitizing 657
  - 11.3.1 Detection of a Specific Fault 659
- 11.4 Circuits with Tree Structure 661
- 11.5 Random Tests 662
- 11.6 Testing of Sequential Circuits 665
  - 11.6.1 Design for Testability 665
- 11.7 Built-in Self-Test 669
  - 11.7.1 Built-in Logic Block Observer 673
  - 11.7.2 Signature Analysis 675
  - 11.7.3 Boundary Scan 676
- 11.8 Printed Circuit Boards 676
  - 11.8.1 Testing of PCBs 678
  - 11.8.2 Instrumentation 679
- 11.9 Concluding Remarks 680
- Problems 680
- References 683

## Appendix A

### VHDL REFERENCE 685

- A.1 Documentation in VHDL Code 686
- A.2 Data Objects 687
  - A.2.1 Data Object Names 687
  - A.2.2 Data Object Values and Numbers 687
  - A.2.3 SIGNAL Data Objects 687
  - A.2.4 BIT and BIT\_VECTOR Types 688
  - A.2.5 STD\_LOGIC and STD\_LOGIC\_VECTOR Types 688
  - A.2.6 STD\_ULOGIC Type 689
  - A.2.7 SIGNED and UNSIGNED Types 690
  - A.2.8 INTEGER Type 690
  - A.2.9 BOOLEAN Type 691
  - A.2.10 ENUMERATION Type 691
  - A.2.11 CONSTANT Data Objects 692
  - A.2.12 VARIABLE Data Objects 692
  - A.2.13 Type Conversion 692
  - A.2.14 Arrays 693
- A.3 Operators 693
- A.4 VHDL Design Entity 694
  - A.4.1 ENTITY Declaration 695
  - A.4.2 ARCHITECTURE 695
- A.5 Package 697
- A.6 Using Subcircuits 698
  - A.6.1 Declaring a COMPONENT in a Package 700
- A.7 Concurrent Assignment Statements 701
  - A.7.1 Simple Signal Assignment 701
  - A.7.2 Assigning Signal Values Using OTHERS 703
  - A.7.3 Selected Signal Assignment 704
  - A.7.4 Conditional Signal Assignment 704
  - A.7.5 GENERATE Statement 705
- A.8 Defining an Entity with GENERICS 707
- A.9 Sequential Assignment Statements 707
  - A.9.1 PROCESS Statement 708
  - A.9.2 IF Statement 708
  - A.9.3 CASE Statement 709
  - A.9.4 LOOP Statements 710
  - A.9.5 Using a Process for a Combinational Circuit 710
  - A.9.6 Statement Ordering 711
  - A.9.7 Using a VARIABLE in a PROCESS 712
- A.10 Sequential Circuits 716
  - A.10.1 A Gated D Latch 717
  - A.10.2 D Flip-Flop 717
  - A.10.3 Using a WAIT UNTIL Statement 719
  - A.10.4 A Flip-Flop with Asynchronous Reset 719
  - A.10.5 Synchronous Reset 719
  - A.10.6 Instantiating a Flip-Flop from a Library 721
  - A.10.7 Registers 721
  - A.10.8 Shift Registers 723
  - A.10.9 Counters 725
  - A.10.10 Using Subcircuits with GENERIC Parameters 725
  - A.10.11 A Moore-Type Finite State Machine 728
  - A.10.12 A Mealy-Type Finite State Machine 731
  - A.10.13 Manual State Assignment for a Finite State Machine 731

- A.11 Common Errors in VHDL Code 734
- A.12 Concluding Remarks 738
- References 738

## Appendix B

### TUTORIAL 1 739

- B.1 Introduction 740
  - B.1.1 Getting Started 740
- B.2 Design Entry Using Schematic Capture 743
  - B.2.1 Specifying the Project Name 744
  - B.2.2 Using the Graphic Editor 744
  - B.2.3 Synthesizing a Circuit from the Schematic 750
  - B.2.4 Performing Functional Simulation 751
  - B.2.5 Using the Message Processor to Locate and Fix Errors 755
- B.3 Design Entry Using VHDL 757
  - B.3.1 Specifying the Project Name 757
  - B.3.2 Using the Text Editor 757
  - B.3.3 Synthesizing a Circuit from the VHDL Code 759
  - B.3.4 Performing Functional Simulation 759
  - B.3.5 Using the Message Processor to Debug VHDL Code 760
- B.4 Design Entry Using Truth Tables 760
  - B.4.1 Using the Waveform Editor 760
  - B.4.2 Create the Timing Diagram 761
  - B.4.3 Synthesizing a Circuit from the Waveforms 763
- B.5 Mixing Design-Entry Methods 764
  - B.5.1 Creating a Schematic that Includes a Truth Table 764
  - B.5.2 Synthesizing and Simulating a Circuit from the Schematic 765
  - B.5.3 Using the Hierarchy Display 766
  - B.5.4 Concluding Remarks 767

## Appendix C

### TUTORIAL 2 769

- C.1 Implementing a Circuit in a MAX 7000 CPLD 770
  - C.1.1 Using the Compiler 771
  - C.1.2 Selecting a Chip 772
  - C.1.3 Viewing the Logic Synthesis Options 773
  - C.1.4 Examining the Implemented Circuit 774
  - C.1.5 Running the Timing Simulator 775
  - C.1.6 Using the Floorplan Editor 776

- C.2 Implementing a Circuit in a FLEX 10K FPGA 779
- C.3 Downloading a Circuit into a Device 781
- C.4 Making Pin Assignments 783
  - C.4.1 Assigning Signals to Pins in the Floorplan Editor 785
  - C.4.2 Making Pin Assignments Permanent 786
- C.5 Concluding Remarks 787

## Appendix D

### TUTORIAL 3 789

- D.1 Design Using Hierarchical VHDL Code 790
  - D.1.1 The Full-Adder Subcircuit 790
  - D.1.2 The Ripple-Carry Adder Code 790
  - D.1.3 Alternative Style of Code for the Ripple-Carry Adder 795
  - D.1.4 Using the Timing Analyzer Module 795
- D.2 Using an LPM Module 796
- D.3 Design of a Sequential Circuit 800
  - D.3.1 Using the Graphic Editor 800
  - D.3.2 Synthesizing a Circuit and Using the Timing Simulator 806
  - D.3.3 Using the Timing Analyzer 807
  - D.3.4 Using VHDL Code 807
- D.4 Design of a Finite State Machine 809
  - D.4.1 Implementation in a CPLD 810
  - D.4.2 Implementation in an FPGA 813
- D.5 Concluding Remarks 815

## Appendix E

### COMMERCIAL DEVICES 817

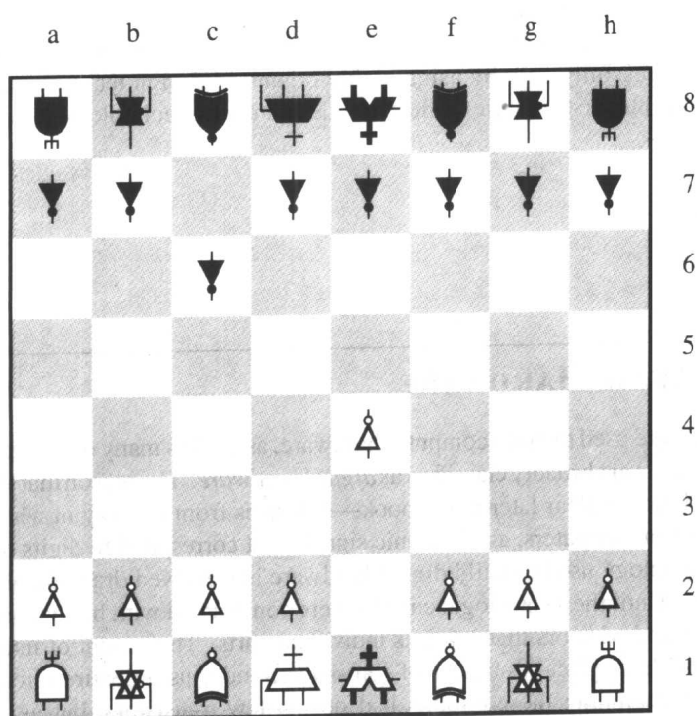
- E.1 Simple PLDs 818
  - E.1.1 The 22V10 PAL Device 818
- E.2 Complex PLDs 820
  - E.2.1 Altera MAX 7000 821
- E.3 Field-Programmable Gate Arrays 822
  - E.3.1 Altera FLEX 10K 823
  - E.3.2 Xilinx XC4000 826
- E.4 Transistor-Transistor Logic 827
  - E.4.1 TTL Circuit Families 829
- References 830

## INDEX 831

# chapter

# 1

## DESIGN CONCEPTS



1. e2-e4, c7-c6

This book is about logic circuits—the circuits from which computers are built. Proper understanding of logic circuits is vital for today's electrical and computer engineers. These circuits are the key ingredient of computers and are also used in many other applications. They are found in commonly used products, such as digital watches, various household appliances, CD players, and electronic games, as well as in large systems, such as the equipment for telephone and television networks.

The material in this book will introduce the reader to the many issues involved in the design of logic circuits. It explains the key ideas with simple examples and shows how complex circuits can be derived from elementary ones. We cover the classical theory used in the design of logic circuits in great depth because it provides the reader with an intuitive understanding of the nature of such circuits. But throughout the book we also illustrate the modern way of designing logic circuits, using sophisticated *computer aided design (CAD)* software tools. The CAD methodology adopted in the book is based on the industry-standard design language called VHDL. Design with VHDL is first introduced in Chapter 2, and usage of VHDL and CAD tools is an integral part of each chapter in the book.

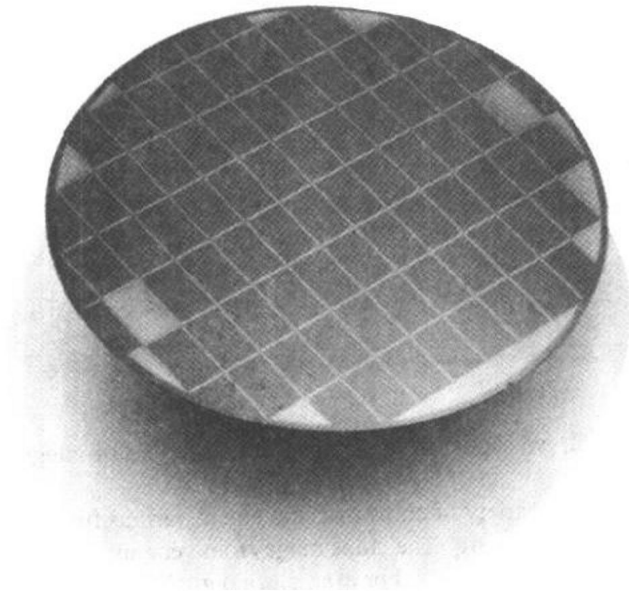
Logic circuits are implemented electronically, using transistors on an integrated circuit chip. With modern technology it is possible to fabricate chips that contain tens of millions of transistors, as in the case of computer processors. The basic building blocks for such circuits are easy to understand, but there is nothing simple about a circuit that contains tens of millions of transistors. The complexity that comes with the large size of logic circuits can be handled successfully only by using highly organized design techniques. We introduce these techniques in this chapter, but first we briefly describe the hardware technology used to build logic circuits.

---

## 1.1 DIGITAL HARDWARE

Logic circuits are used to build computer hardware, as well as many other types of products. All such products are broadly classified as *digital hardware*. The reason that the name *digital* is used will become clear later in the book—it derives from the way in which information is represented in computers, as electronic signals that correspond to digits of information.

The technology used to build digital hardware has evolved dramatically over the past four decades. Until the 1960s logic circuits were constructed with bulky components, such as transistors and resistors that came as individual parts. The advent of integrated circuits made it possible to place a number of transistors, and thus an entire circuit, on a single chip. In the beginning these circuits had only a few transistors, but as the technology improved they became larger. Integrated circuit chips are manufactured on a silicon wafer, such as the one shown in Figure 1.1. The wafer is cut to produce the individual chips, which are then placed inside a special type of chip package. By 1970 it was possible to implement all circuitry needed to realize a microprocessor on a single chip. Although early microprocessors had modest computing capability by today's standards, they opened the door for the information processing revolution by providing the means for implementation of affordable personal computers. About 30 years ago Gordon Moore, chairman of Intel Corporation, observed that integrated circuit technology was progressing at an astounding rate, doubling the number of transistors that could be placed on a chip every 1.5 to 2 years.



**Figure 1.1** A silicon wafer (courtesy of Altera Corp.).

This phenomenon, informally known as *Moore's law*, continues to the present day. Thus in the early 1990s microprocessors could be manufactured with a few million transistors, and by the late 1990s it has become possible to fabricate chips that contain more than 10 million transistors.

Moore's law is expected to continue to hold true for at least the next decade. A consortium of integrated circuit manufacturers called the Semiconductor Industry Association (SIA) produces an estimate of how the technology is expected to evolve. Known as the *SIA Roadmap* [1], this estimate predicts the minimum size of a transistor that can be fabricated on an integrated circuit chip. The size of a transistor is measured by a parameter called its *gate length*, which we will discuss in Chapter 3. A sample of the SIA Roadmap is given in Table 1.1. In 1999 the minimum possible gate length that can be reliably manufactured is  $0.14\text{ }\mu\text{m}$ . The first row of the table indicates that the minimum gate length is expected to reduce steadily to about  $0.035\text{ }\mu\text{m}$  by the year 2012. The size of a transistor determines how many transistors can be placed in a given amount of chip area, with the current maximum being about 14 million transistors per  $\text{cm}^2$ . This number is expected to grow to 100 million transistors by the year 2012. The largest chip size is expected to be about  $1300\text{ mm}^2$  at that time; thus chips with up to 1.3 billion transistors will be possible! There is no doubt that this technology will have a huge impact on all aspects of people's lives.

The designer of digital hardware may be faced with designing logic circuits that can be implemented on a single chip or, more likely, designing circuits that involve a number of chips placed on a *printed circuit board (PCB)*. Frequently, some of the logic circuits can be realized in existing chips that are readily available. This situation simplifies the design task and shortens the time needed to develop the final product. Before we discuss the design

**Table 1.1** A sample of the SIA Roadmap

	Year					
	1999	2001	2003	2006	2009	2012
Transistor gate length	0.14 $\mu\text{m}$	0.12 $\mu\text{m}$	0.10 $\mu\text{m}$	0.07 $\mu\text{m}$	0.05 $\mu\text{m}$	0.035 $\mu\text{m}$
Transistors per $\text{cm}^2$	14 million	16 million	24 million	40 million	64 million	100 million
Chip size	800 $\text{mm}^2$	850 $\text{mm}^2$	900 $\text{mm}^2$	1000 $\text{mm}^2$	1100 $\text{mm}^2$	1300 $\text{mm}^2$

process in more detail, we should introduce the different types of integrated circuit chips that may be used.

There exists a large variety of chips that implement various functions that are useful in the design of digital hardware. The chips range from very simple chips with low functionality to extremely complex chips. For example, a digital hardware product may require a microprocessor to perform some arithmetic operations, memory chips to provide storage capability, and interface chips that allow easy connection to input and output devices. Such chips are available from various vendors.

For most digital hardware products, it is also necessary to design and build some logic circuits from scratch. For implementing these circuits, three main types of chips may be used: standard chips, programmable logic devices, and custom chips. These are discussed next.

### 1.1.1 STANDARD CHIPS

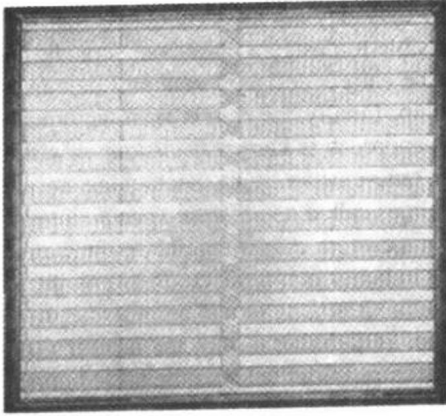
Numerous chips are available that realize some commonly used logic circuits. We will refer to these as *standard chips*, because they usually conform to an agreed-upon standard in terms of functionality and physical configuration. Each standard chip contains a small amount of circuitry (usually involving fewer than 100 transistors) and performs a simple function. To build a logic circuit, the designer chooses the chips that perform whatever functions are needed and then defines how these chips should be interconnected to realize a larger logic circuit.

Standard chips were popular for building logic circuits until the early 1980s. However, as integrated circuit technology improved, it became inefficient to use valuable space on PCBs for chips with low functionality. Another drawback of standard chips is that the functionality of each chip is fixed and cannot be changed.

### 1.1.2 PROGRAMMABLE LOGIC DEVICES

In contrast to standard chips that have fixed functionality, it is possible to construct chips that contain circuitry that can be configured by the user to implement a wide range of different logic circuits. These chips have a very general structure and include a collec-





**Figure 1.2** A field-programmable gate array chip (courtesy of Altera Corp.).

tion of *programmable switches* that allow the internal circuitry in the chip to be configured in many different ways. The designer can implement whatever functions are needed for a particular application by choosing an appropriate configuration of the switches. The switches are programmed by the end user, rather than when the chip is manufactured. Such chips are known as *programmable logic devices (PLDs)*. We will introduce them in Chapter 3.

Most types of PLDs can be programmed multiple times. This capability is advantageous because a designer who is developing a prototype of a product can program a PLD to perform some function, but later, when the prototype hardware is being tested, can make corrections by reprogramming the PLD. Reprogramming might be necessary, for instance, if a designed function is not quite as intended or if new functions are needed that were not contemplated in the original design.

PLDs are available in a wide range of sizes. They can be used to realize much larger logic circuits than a typical standard chip can realize. Because of their size and the fact that they can be tailored to meet the requirements of a specific application, PLDs are widely used today. One of the most sophisticated types of PLD is known as a *field-programmable gate array (FPGA)*. FPGAs that contain more than 100 million transistors will soon be available [2,3]. A photograph of an FPGA chip that has 10 million transistors is shown in Figure 1.2. The chip consists of a large number of small logic circuit elements, which can be connected together using the programmable switches. The logic circuit elements are arranged in a regular two-dimensional structure.

### 1.1.3 CUSTOM-DESIGNED CHIPS

PLDs are available as off-the-shelf components that can be purchased from different suppliers. Because they are programmable, they can be used to implement most logic circuits found in digital hardware. However, PLDs also have a drawback in that the programmable switches consume valuable chip area and limit the speed of operation of implemented cir-