Introduction to Digital Systems Design

Giuliano Donzellini · Luca Oneto
Domenico Ponta · Davide Anguita

Introduction to Digital Systems Design

Springer

Giuliano Donzellini
Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecomunicazioni (DITEN)
Università degli Studi di Genova
Genoa
Italy
e-mail: giuliano.donzellini@unige.it

Luca Oneto
Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS)
Università degli Studi di Genova
Genoa
Italy
e-mail: luca.oneto@unige.it

Domenico Ponta
Università degli Studi di Genova
Genoa
Italy
e-mail: ponta@unige.it

Davide Anguita
Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS)
Università degli Studi di Genova
Genoa
Italy
e-mail: davide.anguita@unige.it

ISBN 978-3-319-92803-6 ISBN 978-3-319-92804-3 (eBook)
https://doi.org/10.1007/978-3-319-92804-3

Library of Congress Control Number: 2018943258

Originally published in Italian as "Introduzione al Progetto di Sistemi Digitali", in 2018 by Springer Milano, Italy, Print ISBN 978-88-470-3962-9, Online ISBN 978-88-470-3963-6. The rights for the Italian language version of the text are owned by Springer-Verlag Italia S.r.l. 2018.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my wife Melina and my sons Sara and Paolo,
who supported me with their love in this work, leaving me
all the necessary time often subtracted to their needs.
To my Father and my Mother,
who I did not have time to thank as I would have liked.
Giuliano Donzellini

To my Mom and Dad for my absences.
To Federica for her presence.
Luca Oneto

To my wife Luisa for the gift of her time
and the lifelong encouragement and support.
To the memory of Sandro Chiabrera, teacher, mentor and friend.
Domenico Ponta

Dedicated to Sandro Ridella, my mentor and friend who,
after all of these years, never stops teaching me.
Davide Anguita

Foreword of Prof. Filippo Sorbello

It is a great pleasure to present the book *Introduction to Digital Systems Design* by my friends and colleagues Donzellini, Oneto, Ponta, and Anguita. This textbook is suited for first year students of Engineering and Computer Science. It starts from the theoretical bases of digital systems, chosen and treated at the right level of depth, proceeds toward the analysis and synthesis of combinational and sequential logic to reach its target of designing and simulating controller–datapath systems. A very high number of examples and exercises with related solutions are provided.

The evolution of electronic technologies has brought a wide diffusion of digital systems in every field of everyday life. Speed, density, and complexity of current digital circuits have been made possible by automatic design methodologies and technological progress.

The knowledge of the theoretical bases of logic networks is necessary to achieve a complete mastery of digital system architectures of different complexities and also for the correct use of automatic design tools based on hardware description languages (HDLs).

First year students possess neither the adequate programming and abstraction abilities nor the physics and electronics knowledge necessary to use HDLs properly. In this textbook, this difficulty is overcome by employing a simulation tool (*Deeds*), developed by one of the authors, which uses an user-friendly interface. *Deeds* is employed to simulate the behavior both of the circuits proposed in the textbook and of those that the learner will autonomously design and then verify. *Deeds* projects can be exported in HDL and tested on FPGA circuits.

The use of languages for hardware description, together with the knowledge of the theoretical bases of logic circuits, represent the keys to understanding the digital world.

I think that this book is a good tool to face this challenge, given the ability that the authors have shown in transferring the necessary theoretical and professional know-how in a text with clear contents, smooth layout, and pleasant aspect.

Palermo, Italy
March 2018

Filippo Sorbello

Foreword of Prof. Mauro Olivieri

The textbook written by Giuliano Donzellini, Luca Oneto, Domenico Ponta, and Davide Anguita is characterized by two features that distinguish it in the wide field of university textbooks introducing digital design.

The first feature is the focus on a well-defined group of notions and tools representing the basis for digital systems: combinational and sequential logic synthesis and the topics strictly connected with them. The book covers neither electronic circuits nor microprocessor systems, and by remaining within these limits, it allows great clarity, precision, completeness, and consistency for the learners, as it appears immediately to the reader by inspecting the high number of schematics and timing diagrams provided with the textbook. Besides, the availability of solved exercises, together with the simulation software *Deeds*, represents a key element that is always appreciated and requested by students.

The second feature is the balance between the theoretical structure and the practical implementation, which allows solid learning. Even though in the textbook the word "voltage" is never cited, a student using the book always has the impression he/she is studying an electronic system formalization. At the same time, the textbook avoids presenting the topics only as a series of practical design examples without a theoretical basis.

This peculiarity characterizes the approach of the "school" of digital systems at the University of Genoa, in respect of which I consider myself an outsider.

For these reasons, the textbook of Donzellini, Oneto, Ponta, and Anguita is a precious tool for a student willing to deeply understand the concepts belonging to the big world of digital electronics design.

Roma, Italy
March 2018

Prof. Mauro Olivieri

Preface

The large and ever-growing complexity of today's digital systems places heavy demands on educational systems that are in charge of training the new generations of designers or just providing a solid understanding of the digital world. Academic institutions struggle to keep the pace of technological advancements, and people, like the authors of this book, who are in charge of introductory- or intermediate-level education, have the responsibility to face the problem and make choices.

It is certainly obvious that a digital designer must be trained in the use of hardware description languages (HDLs) and it is nowadays a common practice to introduce them very early in the courses, substituting the traditional approach based on components and schematics. The choice to describe digital systems by HDL matches very well with the adoption of Field-Programmable Gate Arrays (FPGA) for the practical implementation of projects, using prototype boards provided by chip producers.

Nevertheless, it is our opinion that the adoption of HDL in a beginner course of logic networks with limited resources in terms of credits (as in our case) may present problems. We believe that it is not easy to build a solid understanding of the foundations by completely replacing logic components and schematics with HDL, which requires a level of abstraction and a familiarity with programming that beginner students generally do not possess.

What is more, employing a simulation and synthesis software developed by FPGA chip producers presents other problems. Tools developed for digital systems designers may not necessarily satisfy learning needs: their use is not immediate for students, who may end up using them partially and mechanically, with the risk of missing important basic concepts, hidden under the technicalities of HDL and tools.

It is therefore necessary that students acquire a solid foundation on which to build design abilities and, at the same time, adapt to the fast rate of technological innovation, while gaining familiarity with languages and design tools.

For these reasons, the textbook maintains a traditional approach to logic networks, described and designed through symbols and schematics, while taking into account today's state of the art when choosing topics and, especially, exercises and projects. This feature allows an optimal use of the book in university curricula that contain only one course on digital hardware, while providing a solid foundation for higher-level studies.

The book takes an original approach in introducing FPGA devices and VHDLs. The last chapter shows how projects similar to the ones presented earlier and tested by simulation only can be practically and quickly implemented on FPGA boards, using Deeds tools. The procedure offers the opportunity of an "hands-on" introduction to FPGA devices and VHDL.

The book is self-sufficient, since it supports the theoretical part with a huge number of examples and exercises, complete with their solutions. In courses that have room for a laboratory session, the symbiosis with the *Digital Electronics Education and Design Suite* (*Deeds*) simulation tool can be exploited, with important advantages. *Deeds* was developed recently by one of the authors (Giuliano Donzellini), with the precise target of supporting learning and laboratory activities for Information Engineering students. The strong connection with *Deeds* represents an important strength and the originality of our work, since all the schematics, examples, and design exercises included, from the easiest to the most complex, were created with *Deeds* and are available online for an immediate simulation.

The *Deeds* environment covers all the principal aspects of digital systems design, from combinational and sequential logic to finite state machines and embedded systems, thus allowing for the design and simulation of complex networks containing standard logic, finite state machines, user-defined components, and microprocessors, including their programming in *assembly* language.

Deeds has been developed with the idea of matching ease of use and almost professional features. The main differences between *Deeds* and a professional tool are represented by the friendliness of the user interface and the availability of a wide collection of teaching material and projects. Furthermore, *Deeds* is an "alive," continuously evolving system: updates are periodically available to improve existent tools and add new ones. The same is true for teaching materials.

The transition toward FPGA devices is supported by *Deeds* that allows to export any of its projects to a professional tool, in order to test it in FPGA hardware. *Deeds* bypasses the complexity of the process that is normally required by a specific professional software and does not require writing HDL code, which is automatically generated by *Deeds*. The rich teaching material of *Deeds* is hence redirected toward FPGA implementation, without substantial modifications.

However, after practicing with the automatic HDL code generation by *Deeds*, students can directly interact with FPGA tools, thus having the possibility to observe, modify, and reuse HDL code (VHDL in our case), making a gradual transition toward current design techniques.

Teaching Objectives

According to authors' experience, the whole content of the book, together with design exercises and simulations based on *Deeds*, may be developed in an introductory course to digital systems of at least nine credits.

In the following, we briefly report the content of the chapters, indicating in italics the topics that can be avoided without loss of continuity with the teaching project, for courses with a smaller number of credits:

1. Boolean Algebra and Combinational Logic
   - Classic approach that does not require preliminary knowledge.
     It is possible to skip Shannon's theorems.
2. Combinational Network Design
   - Synthesis and minimization with Karnaugh maps.
   - Standard combinational logic.
   - Propagation delays.
     Variable-Entered Maps and hazards may be omitted.
3. Numeral Systems and Binary Arithmetic
   - Classic approach.
   - Arithmetic networks.
     Binary negative numbers may be omitted, as well as BCD arithmetic.
4. Complements in Combinational Network Design
   - Minimization of expressions with Quine–McCluskey method.
     The entire chapter may be omitted.
5. Introduction to Sequential Networks
   - Intuitive transition from combinational to sequential logic.
   - Structure and operation of principal flip-flop types.
   - Dynamic flip-flop characteristics.
     It is possible to consider just "D" and "E" logic types and to skip their circuital details.
6. Flip-Flop-BasedSynchronous Networks
   - Introduction to synchronous flip-flop networks.
   - Sequential networks: registers and counters.
   - Techniques for timing analysis of synchronous networks.
     Counters and registers section may be reduced, as well as timing analysis of sequential networks.

7. Sequential Networks as Finite State Machines
   - FSM project, realized through ASM diagrams.
   - Solved exercises of ASM diagrams.
   - FSM synthesis with state tables and maps.
     FSM synthesis may be reduced, by omitting variable-entered maps, or completely left aside.
8. The Finite State Machine as System Controller
   - Design of Controller–Datapath systems.
   - Solved exercises on controller–datapath systems.
     This chapter applies all the material presented in the book to develop controller–datapath systems. The projects may be chosen according to the needs and level of the class.
9. Introduction to FPGA and HDL Design
   - Introduction to FPGA.
   - System prototyping on FPGA with *Deeds* tools.
   - Introduction to VHDL.
   - Examples of FPGA prototyping projects.
     This chapter requires the use of Deeds, and it is fully exploited when accompanied by laboratory activities.

How to Use the Book

The strict connection between this book and the *Deeds* tool suggests using it together with the simulation tools, both to verify and test concepts and procedures in an active way and to have a support for the solution of the exercises and the design of systems.

This "learning by doing" practice allows students to progressively build the analytic and design capabilities that represent the target to reach.

Giuliano Donzellini
Luca Oneto
Domenico Ponta
Davide Anguita
Genova, Italy

Digital Contents for the Book

This textbook contains theoretical parts, examples, exercises, and solutions. All the examples have also been implemented with the *Deeds* simulator that can be downloaded from the link:

https://www.digitalelectronicsdeeds.com

The Web site contains a description of the *Deeds*'s features, tutorials, and learning materials. The simulator does not require an Internet connection.

On the same Web site, as additional material, it is possible to find almost all the schematics and charts included in the book:

https://www.digitalelectronicsdeeds.com/books

Thanks to this material, it is possible to simulate with *Deeds* the proposed circuits and the exercises. The material has been organized by following the same structure of the book in order to make it easier to access. On the same Web site, future updates, corrections, and additions will be made available.

Contents

1 Boolean Algebra and Combinational Logic 1
1.1 Analog and Discrete Variables 1
1.2 Boolean Variables 3
1.3 Boolean Functions 4
1.4 Truth Tables 4
1.5 Definition of Boolean Algebra 5
1.6 The Fundamental Properties of Boolean Algebra 6
1.7 Other Operations 10
1.8 Functionally Complete Operation Sets 11
1.9 Shannon's Expansion Theorem 14
1.10 Level of Boolean Expressions 17
1.11 Literals 17
1.12 Minterms 18
1.13 Maxterms 18
1.14 Implicants 18
1.15 Prime Implicants 18
1.16 Combinational Networks 19
1.16.1 Example: Logical Network Analysis 19
1.16.2 Example: Two-Level Logical Network Analysis 20
1.16.3 Example: Circuit Schematic of a Logical Network (1) 20
1.16.4 Example: Circuit Schematic of a Logical Network (2) 21
1.16.5 Example: Defining the Behavior of a Logical Network 21
1.16.6 Example: Circuit Schematic from the Truth Table 22
1.16.7 Example: Controlling a Heating System 23
1.16.8 Example: Two Channels Multiplexer (Selector) 24

1.17 Exercises 27
1.18 Solutions 29

2 Combinational Network Design 33
2.1 Karnaugh Maps 33
2.2 Using Maps for AND-OR Synthesis 36
2.2.1 Implicants and Prime Implicants in Karnaugh Maps 38
2.2.2 Using Maps for Minimization 41
2.2.3 “Checkerboard” Maps 41
2.2.4 Examples of AND-OR Synthesis 42
2.3 OR-AND Synthesis 45
2.3.1 Synthesis of the Negated Function 46
2.4 NAND-NAND Synthesis 46
2.5 NOR-NOR Synthesis 47
2.6 Standard Combinational Networks 48
2.6.1 Decoders 48
2.6.2 Multiplexer 51
2.6.3 Demultiplexers 53
2.6.4 Seven-Segment Display Decoder 55
2.6.5 Seven-Segment BCD Decoder (using “don’t-cares”) 57
2.6.6 Using Multiplexers to Synthesize Combinational Networks 58
2.7 Variable-Entered Maps 60
2.7.1 Synthesizing Maps with Entered Variables 60
2.7.2 Entered Variables and Theorems of Expansion 63
2.8 Time Behavior of Combinational Networks 64
2.8.1 Definitions and Timing Models 64
2.8.2 Hazards 66
2.8.3 Elimination of Static Hazards 67
2.8.4 Notes on Eliminating Hazards 70
2.9 Exercises 70
2.9.1 Maps 70
2.9.2 Hazards 72
2.10 Solutions 73
2.10.1 Maps 73
2.10.2 Hazards 77

3 Numeral Systems and Binary Arithmetic 79
3.1 Binary Information 79
3.2 Binary Numbering System (BIN) 80
3.2.1 Converting from Binary System to Decimal 80
3.2.2 Converting from Decimal System to Binary 81
3.2.3 Maximum Representable Number 82
3.3 Octal Number System (OCT) 83

3.4 Hexadecimal Number System (HEX) 84
3.5 Others Binary Codes 85
3.5.1 Binary Coded Decimal 85
3.5.2 GRAY Codes 86
3.6 Binary Arithmetic 87
3.6.1 Addition 87
3.6.2 Subtraction 88
3.6.3 Products 89
3.7 BCD 8421 Arithmetic 89
3.8 Binary Rational Numbers 91
3.9 Arithmetic Networks 91
3.9.1 Half Adders 91
3.9.2 Full Adders 92
3.9.3 Ripple Carry Adder 93
3.9.4 Arithmetic Logic Unit (ALU) 94
3.10 Relative Numbers in Binary 95
3.10.1 Representation in “Module and Sign” Code 95
3.10.2 Complementation 97
3.10.3 Representation in “Ones’ Complement” Code 99
3.10.4 Representation in “Two’s Complement” Code 101
3.10.5 Sign Extension 102
3.11 Representation of Real Numbers 103
3.12 Alphanumeric Codes 104
3.13 Error Detection Codes: Parity Generator and Checker 106
3.14 Exercises 108
3.14.1 Binary Numbers 108
3.14.2 Signed Binary Numbers 108
3.14.3 Octal and Hexadecimal Numbers 109
3.15 Solutions 109
3.15.1 Binary Numbers 109
3.15.2 Signed Binary Numbers 111
3.15.3 Octal and Hexadecimal Numbers 112

4 Complements in Combinational Network Design 115
4.1 Minimizing Boolean Expressions with the Quine–McCluskey Method 115
4.1.1 The Expansion Phase 116
4.1.2 The Covering Phase 119
4.1.3 Incompletely Specified Functions 120
4.1.4 Optimizing the Covering Phase 120
4.1.5 Simultaneous Optimization of Multiple Functions 125

4.2 Exercises . . . 130
4.2.1 Quine–McCluskey: Single Function Synthesis . . . 130
4.2.2 Quine–McCluskey: Jointly Synthesis of Multiple Functions . . . 133
4.3 Solutions . . . 136
4.3.1 Quine–McCluskey: Synthesis of a Single Function . . . 136
4.3.2 Quine–McCluskey: Joint Synthesis of Multiple Functions . . . 137

5 Introduction to Sequential Networks . . . 139
5.1 From Combinational Networks to Sequential Networks . . . 139
5.1.1 Introductory Example . . . 140
5.1.2 Memorizing an Information Bit: Flip-Flops . . . 141
5.1.3 Flip-Flop Classification: Logical Type and Command Type . . . 142
5.2 Direct Command Flip-Flops . . . 143
5.2.1 SR Flip-Flop . . . 143
5.2.2 D Flip-Flop . . . 147
5.2.3 JK Flip-Flop . . . 148
5.3 Initialization of a Sequential Network . . . 149
5.3.1 Flip-Flop Initialization Inputs . . . 150
5.3.2 Generating an Initialization Signal . . . 152
5.4 Level-Enabled Flip-Flops . . . 153
5.4.1 SR-Latch Flip-Flop . . . 153
5.4.2 D-Latch Flip-Flop . . . 155
5.4.3 JK-Latch Flip-Flop . . . 156
5.5 Synchronization of Sequential Networks . . . 157
5.5.1 The Synchronization Signal . . . 157
5.5.2 Pulse Command in Level-Enabled Flip-Flops . . . 158
5.5.3 The “Clock” and the “Edge-Triggered Command” . . . 159
5.5.4 Master–Slave Structure . . . 160
5.6 Edge-Triggered Flip-Flops . . . 162
5.6.1 D-PET Flip-Flop . . . 162
5.6.2 E-PET Flip-Flop . . . 166
5.6.3 JK-PET Flip-Flop . . . 168
5.6.4 T-PET Flip-Flop . . . 170
5.6.5 Synchronous Initialization of Flip-Flops . . . 171
5.7 Timing Parameters of Flip-Flops . . . 171
5.7.1 Relationship Between Propagation and Hold Times . . . 172
5.7.2 Maximum Clock Frequency of a Network with Flip-Flops . . . 173
5.8 Flip-Flops: Graphic Symbols and Tables . . . 174
5.8.1 Logical Types . . . 174

5.8.2 Command Types 175
5.8.3 Excitation Tables 177
5.9 Exercises 178
5.10 Solutions 181

6 Flip-Flop-Based Synchronous Networks 183
6.1 Synchronous and Asynchronous Signals. 185
6.1.1 Synchronizer 186
6.1.2 Multistage Synchronization 187
6.2 Registers 188
6.2.1 Parallel Registers 188
6.2.2 Shift Registers 190
6.2.3 Shift Registers with Parallel Load 193
6.2.4 Universal Shift Register 196
6.3 Counters 199
6.3.1 Binary Counters 199
6.3.2 Counters with Enabling 205
6.3.3 Up/Down Counters. 207
6.3.4 "Universal" Counters 210
6.3.5 Asynchronous Counters 213
6.4 Network Analysis Examples 215
6.4.1 Example 1 215
6.4.2 Example 2 218
6.4.3 Example 3 221
6.4.4 Example 4 223
6.4.5 Example 5 224
6.5 Exercises 228
6.6 Solutions 248

7 Sequential Networks as Finite State Machines 259
7.1 Finite State Machines: Standard Model 260
7.1.1 Synchronous and Asynchronous Machines. 261
7.1.2 Moore and Mealy Machines 261
7.1.3 An Example of Synchronous Finite State Machine. 262
7.1.4 General Equations of the Next State and the Outputs . . . 263
7.2 ASM Diagrams. 265
7.2.1 Description of States 265
7.2.2 Inputs 269
7.2.3 Conditional Outputs 277
7.3 Examples of ASM Diagram Construction. 282
7.3.1 Introductory Examples 282
7.3.2 Pulse Generators with Adjustable Duty Cycle 291
7.3.3 Sequence Detector 296
7.3.4 Serial Synchronous Transmitters (2 bits) 298

7.3.5 Command Receiver with Serial Synchronous Interface 300
7.3.6 Serial Synchronous Receiver (2 bits) 303
7.3.7 Push-Button Handling 306
7.3.8 3-Bit Shift Registers 309
7.3.9 Sequential Networks with Conditional Outputs 312
7.3.10 Shift Register with XOR Tree 314
7.4 Synthesis of Synchronous FSM 316
7.4.1 State Assignment 317
7.4.2 Describing an FSM with a State Table 318
7.4.3 State Table Synthesis 320
7.4.4 Examples of Synchronous FSM Synthesis 323
7.5 Time Behavior of Synchronous FSM 338
7.5.1 FSM with no Inputs 338
7.5.2 FSM with Synchronous and Asynchronous Inputs 339
7.5.3 Synchronous Inputs 340
7.5.4 Asynchronous Inputs 341
7.5.5 FSM Outputs (Moore's Model) 342
7.5.6 FSM Outputs (Mealy's Model) 342
7.6 Exercises 343
7.6.1 Sequential Network Analysis in Terms of FSMs 343
7.6.2 FSM Design Based on Textual Specifications 346
7.7 Solutions 351
7.7.1 Sequential Network Analysis in Terms of FSMs 351
7.7.2 FSM Design Based on Textual Specifications 355

8 The Finite State Machine as System Controller 363
8.1 Digital Systems 363
8.2 Open Control Systems 364
8.2.1 2-Bit Serial Receiver 365
8.3 Feedback Control Systems 369
8.3.1 2-Bit Serial Receiver and Transmitter 369
8.3.2 Pulse Generator 372
8.3.3 8-Bit Serial Receiver 377
8.3.4 Light Dimmer 384
8.3.5 Combination Lock 388
8.3.6 Automatic Drink Dispenser 393
8.3.7 Programmable Square Wave Generator 397
8.3.8 Christmas Light Systems 401
8.4 Design Exercises 405
8.4.1 Design of the Controller of a Given Datapath 405
8.4.2 Design of a Controller–Datapath System 424

8.5 Design Tips 427
8.5.1 Design of a Controller–Datapath System 427
8.6 Solutions 434
8.6.1 Design of the Controller of a Given Datapath 434
8.6.2 Design of a Controller–Datapath System 452

9 Introduction to FPGA and HDL Design 465
9.1 Field-Programmable Gate Arrays 465
9.1.1 System Prototyping and FPGA 466
9.1.2 FPGA Board Examples 467
9.1.3 FPGA Architecture 470
9.1.4 JTAG Programming 472
9.1.5 FPGA Development Tools 475
9.1.6 Deeds Support for FPGA 478
9.2 Introduction to VHDL 485
9.2.1 VHDL Code from Deeds 486
9.2.2 Counter 487
9.2.3 Finite State Machine 491
9.2.4 Top-Level Entity 494
9.2.5 Other VHDL Examples 497
9.3 FPGA Prototyping Exercises 506
9.3.1 Synchronous Serial Communication System (8-bit) 506
9.3.2 Digital Chronometer 510
9.4 Solutions 514
9.4.1 Synchronous Serial Communication System (8-bit) 514
9.4.2 Digital Chronometer 515

Appendix A: The Powers of 2 519

Appendix B: State Diagrams 521

Appendix C: VHDL Code 525