DIGITAL LOGIC DESIGN

This introductory textbook is a complete teaching tool for turning students into logic designers in one semester, beginning with basic gates and ending with the specification and implementation of simple microprocessors. It shows how to use rigorous mathematical language to accurately define models, specify functionality, describe designs, prove correctness, and analyze cost and delay.

Each chapter first describes new concepts and then gives extensive applications and examples of these new ideas. Assuming no prior knowledge of discrete mathematics, the authors introduce all the necessary background in propositional logic, asymptotics, graphs, hardware, and electronics.

Important features of the presentation are the following:

- All material is presented in full detail, with every claim proved.
- Algorithmic solutions are offered for tasks such as logical simulation, computation of propagation delay, and minimum clock period.
- Connections are drawn from the physical analog world to the digital abstraction.
- The language of graphs is used to describe formulas and circuits.
- Hundreds of examples and exercises enhance understanding.

The extensive Web site http://www.eng.tau.ac.il/~guy/Even-Medina/ includes teaching slides, links to Logisim and a DLX assembly simulator, and other supplements.

Guy Even is a professor in the School of Electrical Engineering at Tel Aviv University, Israel.

Moti Medina is a PhD student in the School of Electrical Engineering at Tel Aviv University, Israel.

Cambridge University Press
978-1-107-02753-4 - Digital Logic Design: A Rigorous Approach
Guy Even and Moti Medina
Frontmatter
More information

DIGITAL LOGIC DESIGN

A Rigorous Approach

GUY EVEN
Tel Aviv University, Israel

MOTI MEDINA
Tel Aviv University, Israel

CAMBRIDGE
UNIVERSITY PRESS

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

Cambridge University Press
32 Avenue of the Americas, New York, NY 10013-2473, USA

www.cambridge.org
Information on this title: www.cambridge.org/9781107027534

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

Even, Guy.
Digital logic design : a rigorous approach / Guy Even, Moti Medina.
pages cm
Includes bibliographical references and index.
ISBN 978-1-107-02753-4
1. Logic design. I. Medina, Moti, 1979– II. Title.
TK7868.L6E94 201
621.39′5–dc23 2012015723

ISBN 978-1-107-02753-4 Hardback

Additional resources for this publication at http://www.eng.tau.ac.il/~guy/Even-Medina/

Contents

List of Algorithms

Preface

This book is an introductory textbook on the design and analysis of digital logic circuits. It has been written after 15 years of teaching hardware design courses in the School of Electrical Engineering at Tel Aviv University. The main motivation for writing a new textbook was the desire to teach hardware design rigorously. By rigorously, we mean that mathematical language and exposition are used to define the model, to specify functionality, to describe designs, to prove correctness, and to analyze cost and delay. We believe that students who study formal courses such as algebra and calculus can cope well with a rigorous approach. Moreover, they are likely to benefit from this approach in many ways.

The book covers the material of an introductory course in digital logic design, including an introduction to discrete mathematics. It is self-contained; it begins with basic gates and ends with the specification and implementation of a simple microprocessor. The goal is to turn our students into logic designers within one semester.

The rest of this preface deals with the rationale, structure, and audience of the book. We conclude with a list of the book's highlights, some of which are new to a hardware design text.

HOW TO ACQUIRE INTUITION

It is not fashionable these days to emphasize mathematical rigor. Mathematical rigor is perceived as an alienating form that dries out the passion for learning and understanding. Common teaching tactics avoid rigor (i.e., the holy definition–theorem–proof) and resort to examples. Since intuition is what really matters (and we, of course, agree with that!), in the rare cases when one feels compelled to provide a proof, the following strategy is employed. First, intuition precedes the proof in an attempt to explain in advance what the proof does and why it actually works (is this part actually an apology for what is about to come?). Then, a long proof follows using partially defined terms. All we can say is that this strategy is in complete disregard of the statement "When you have to shoot, shoot. Don't talk" (as stated by Tuco in *The Good, the Bad, and the Ugly*).

Recall the great endeavor of nineteenth-century mathematicians to formalize the calculus of real functions. Weierstrass and others undertook the task of providing a formal abstraction of the presumably well-understood notions of real numbers, real functions, continuous functions, and so on. We still remember our surprise when Weierstrass's function was first described to us: continuous everywhere and differentiable nowhere. The lesson is clear: intuition is gradually acquired and must be based on solid fundamentals.

What does this have to do with digital design? The quest for intuition is confronted by the fact that it is hard to formulate precise statements about objects such as digital circuits. Our approach is to give students a solid, rigorous basis for their intuition. Of course, examples are easy to follow but might give students the false impression that they understand the topic. We have seen many brilliant students in engineering disciplines who find it hard to acquire intuition based only on examples. Such students can easily cope with a rigorous exposition in which delicate issues are not hidden or brushed aside.

LEARN FROM THE SUCCESS OF DATA STRUCTURES AND ALGORITHMS

We believe that successful teaching means that a student can implement the material from the course. After studying data structures, a student should be able to program search trees, sorting, and hashing. We believe that the same goal should be set for a logic design course. Unfortunately, most textbooks describe various circuits, and provide examples for why they work, but do not train engineers who can actually design digital circuits.

The goal of this book is to bring students to a level that will enable them to understand a specification of a combinational or synchronous circuit, to design it, to prove the correctness of their design, and to be able to analyze the efficiency of the design (i.e., delay and cost).

We do not restrict this goal to isolated circuits. We show how a system is built from different circuits working in concert. In fact, we present a simple microprocessor, the design of which combines multiple modules, including an arithmetic logic unit (with an adder, logical operators, and a comparator), a shifter, a file register (with the general-purpose registers), and main memory.

THE KNOWLEDGE HIGHWAY

Our goal is to turn our students into logic designers within one semester. To meet this goal, we follow a bottom-up approach that begins with the basics and ends with a simple microprocessor. We solidify the presentation by using mathematical notations and statements and by defining the abstraction precisely. The effort spent on a formal approach pays off simply because it enables us to teach more material, in more depth, and in a shorter time. It is not surprising that toward the end of the course, students will not only be able to design nontrivial modules but will also be able to identify errors in designs and suggest ways to correct these errors.

OUR TEACHERS

When writing this book, the first author and, by transitivity, the second author were mainly influenced by three people: Shimon Even, Ami Litman, and Wolfgang Paul.

It was Shimon Even who stated (1) never complain or be surprised by the students' lack of knowledge—just teach it! (2) digital design is the same art as algorithm design; the only difference is the model of computation; and (3) identify the methods and be systematic; in other words, turn digital design into a discipline.

It was Ami Litman who demanded (1) always verify that your abstraction makes sense; don't hesitate to refute the model by introducing absurd consequences; (2) introduce a design by a sequence of evolutionary modifications, starting with a simple straightforward yet costly design and ending with an evolved yet efficient design; each modification preserves functionality and hence the final design is correct—describe each modification as a general transformation that can be applied in a wide variety of settings; and (3) focus on large instances—optimization of small instances depends on the technology and is not likely to reveal insights.

Wolfgang Paul's rules are (1) formulate a precise specification and prove that the design satisfies the specification; (2) write the equations that describe the delay and cost—solving these equations asymptotically is nice, but from a practical point of view, it suffices to solve them numerically for the sizes one needs to actually design; and (3) keep in mind that the goal is to design a correct, well-understood system. Avoid fancy optimizations that eventually impede this goal. This rule applies both for teaching and for actual design.

OUR STUDENTS

Our students are electrical engineering undergraduate students in their second or third semester. The students lack background in discrete mathematics, and the first part of the book deals with filling this gap. This is considered the easy part of the course.

Following the logic design course, our students take courses on devices (both analog and digital). Students who choose the computer track also study computer organization and computer architecture and practice digital design in a lab with an FPGA platform. In this lab, they implement the simplified DLX microprocessor described in Part IV of the book. This implementation is from basic gates (e.g., no library modules, such as adders, are used). At the end of the lab, the students program a nontrivial program in assembly language and execute it on their design.

Apart from training the students in logic design, we also teach discrete methods that are used in data structures and algorithms. In particular, we focus on induction and recursion, trees and graphs, and recurrence equations.

STRUCTURE OF THE BOOK

The book consists of four parts: (I) "Preliminaries," (II) "Combinational Circuits," (III) "Synchronous Circuits," and (IV) "A Simplified DLX."

The first part of the book is a short introduction to discrete mathematics. We made an effort to include only topics in discrete math that are actually used in the other parts. This is considered the easy part of the course; however, it is essential for students who lack background in discrete mathematics. In addition, this part helps students get used to working with definitions, mathematical notation, and proofs.

The second part of the book is its heart. It focuses on Boolean functions and on methods for building circuits that compute Boolean functions. We begin by representation by Boolean formulas, for example, sums of products and products of sums. This establishes the connection between Boolean functions and propositional logic. We then define combinational gates and combinational circuits and define two quality measures: cost and propagation delay.

The study of combinational circuits begins with circuits that have a topology of a tree. At this point we introduce lower bounds on the number of gates and the propagation delay of a combinational circuit that implements a Boolean function such as the OR of n bits. Logical simulation is presented in an algorithmic fashion using topological ordering of a directed acyclic graph. The same approach is used for computing the propagation delay of a combinational circuit.

We proceed with a variety of combinational circuits (e.g., decoders, encoders, selectors, shifters, and adders). Designs are presented in a parametric fashion, where the parameter is the length of the input. Whenever possible, designs are recursive and proofs are by induction.

Chapter 10, in Part II, explains the digital abstraction. The purpose of this chapter is to build a bridge between the analog world and the digital world.

Synchronous circuits are studied in the third part of the book. We first introduce the clock signal and edge-triggered D-flip-flops. Only one type of flip-flop is discussed in detail. This discussion explains the different timing parameters of a flip-flop, including an explanation of why so many parameters are required. Other types of flip-flops are considered as finite state machines with two states and are implemented using a D-flip-flop and additional combinational logic. Synchronous circuits are viewed in two ways: (1) memory modules, such as registers, random access memory (RAM), and read-only memory (ROM) and (2) finite state machines, including their analysis and synthesis.

Algorithmic issues related to synchronous circuits include logical simulation and calculation of the minimum clock period. These algorithms are presented via reductions to combinational circuits.

Students who have studied the first three parts of the book should have a good idea of what computer-aided design tools for designing digital circuits do.

The last part of the book deals with the design of a simple microprocessor. Connections are made between the machine language, assembly, high-level programming, and the instruction set architecture (ISA). We present an implementation of the simple microprocessor using the modules from Parts II and III. The design methodology is to present the simplest microprocessor implementation that supports the ISA. We present an unpipelined multicycle microprocessor based on a simple datapath and a small finite state machine.

HOW TO USE THIS BOOK

This book is written as a textbook for an introductory course in digital design for undergraduate students in electrical engineering and computer science. The following material is considered advanced and may be omitted: Section 5.6*, "More on Unique Binary Representation;" Chapter 8, "Computer Stories: Big Endian versus Little Endian;" Section 9.6*, "Minimization Heuristics;" Chapter 10, "The Digital Abstraction;" and Sections 17.3*–17.5*. Advanced material as well as advanced questions and examples are marked with an asterisk.

When we teach this course, we spend roughly five weeks on Part I, five weeks on Part II, and five weeks on Parts III and IV. We suggest starting the course very rigorously and gradually relaxing rigor when repeating a proof technique that was used before.

Logic design, like swimming, cannot be taught without immersion. We therefore include homework assignments in which students practice logic design using a schematic entry tool and a logic simulator. We found the open source Logisim software both easy to use and powerful enough for our purposes.

We also use a DLX assembly simulator so that students can practice assembly programing of constructs in high-level programming (e.g., if-then-else statements, loops, arrays).

HIGHLIGHTS

Here we list the main highlights of the book:

1. *The book is self-contained.* We do not assume that students have any prior knowledge of discrete math, propositional logic, asymptotics, graphs, hardware, electronics, and so on.
2. *A complete teaching tool.* In each chapter, we tried to make a clear separation between (1) conceptual parts containing new materials, (2) applications and examples that are based on this new material, and (3) problems. There are many benefits to this approach for both the teacher and the student. One clear advantage is that the examples can be covered in greater detail during recitations.
3. *"Nothing is hidden."* We adhere to the rule that all the details are complete and every claim is proven.
4. *Methodology as a "ritual."* Each design is presented in four steps: specification, design, correctness proof, and analysis of delay and cost. The specification formally defines what a circuit should do. Without a formal specification, it is impossible to prove correctness. Most designs are described using recursion, and correctness is usually proved using

induction. Finally, analysis of cost and delay is carried out by formulating recurrence equations and solving them.

5. *The recursion–induction pair.* Instead of designing circuits for specific input lengths, we consider the task of designing circuits with a parameter n specifying the length of the inputs. For example, we consider addition of n-bit numbers, n:1-selectors, and so on. These designs are described recursively. The first advantage is that we present a precise and formal definition of the design for any input length. The second advantage is that we prove the correctness of the design for any input length. Naturally, the proof is carried out using induction.
6. *Modeling circuits as graphs.* We use the language of graphs to describe formulas and circuits. Boolean formulas are defined by parse trees. Circuits are defined using directed graphs. This approach enables us to present a clean and precise definition of propagation delay and minimum clock period using longest paths in a directed graph. With small effort, it is possible to extend this approach to the more elaborate setting of nonuniform delays between input and output ports of gates.
7. *Lower bounds.* We prove simple lower bounds on the cost and the delay of a combinational circuit that implements Boolean functions. The ability to formally state that a design is an optimal adder design is remarkably powerful. Our lower bounds are stated in terms of the number of inputs on which an output depends (i.e., the "cone" of an output). These lower bounds are easy to apply to all the Boolean functions that are discussed.
8. *Algorithmic approach.* Tasks such as logical simulation, computation of propagation delay, and minimum clock period are presented as algorithmic problems. Algorithms are presented for solving these problems, and the correctness of these algorithms is proven.

   For example, the algorithmic approach is used to teach timing analysis, as follows: we present an algorithm, prove its correctness, and run it on an example. In this fashion, the topic of timing analysis is described in a precise and concise fashion that does not require lengthy examples. One may ask, why not teach about timing analysis with different delays for different transitions (i.e., the time required for transition of the output from zero to one does not equal the time required for the transition from one to zero)? Indeed, this question pertains to the lasting argument about the usefulness of worst case analysis. We resort to worst case timing analysis simply because it is intractable to decide whether the output of a combinational circuit ever equals one (see Section 9.5).
9. *Relations to analog world.* In Chapters 10 and 17, we connect the physical analog world to the digital abstraction. Two physical phenomena are discussed in detail: noise and metastability. We show how noise is overcome by using different thresholds for inputs and outputs. We show how metastability is mitigated using the timing parameters of a flip-flop (i.e., setup time, hold time, contamination delay, and propagation delay). We explicitly mention issues that cannot be resolved within the digital abstraction (e.g., reset controller).
10. *Zero propagation delay as functional model.* In Chapter 18, on memory modules, we introduce the *zero delay model*. In the zero delay model, transitions of all signals are instantaneous. This means that the flip-flop's output at a certain cycle equals the value of the input sampled during the previous cycle. This simplified discrete timing model is used for specifying and simulating the functionality of circuits with flip-flops. The

advantage of this approach is that it decouples the issues of functionality and timing into two separate issues.

KARNAUGH MAPS

A quick comparison of this book with other books on logic design will reveal that we mention *Karnaugh maps* (Karnaugh, 1953) only very briefly in Section 9.6.6. There is a good reason for this brief mentioning.

Karnaugh maps are a technique for finding the minimum number of products in a sum-of-products representation of a Boolean function. The input to the technique of *Karnaugh maps* is the truth table of the Boolean function. Thus the input to this technique is exponential in the number of variables and therefore cannot be considered efficient. In addition, the maps are actually two-dimensional tables and are convenient to use for at most four variables. Experts, of course, are proud that they can use this technique also for five and even six variables! Given that the technique of *Karnaugh maps* has an exponential running time and is limited to few variables, we do not think it is an important issue in logic design. One should bear in mind that the difference between a reasonable representation and the best representation for a function over six variables is constant. Moreover, with such small functions, even exhaustive search makes sense if one is really interested in finding the "best" representation.

Teachers insisting on teaching heuristics for finding the minimum number of products in a sum-of-products representation of a Boolean function can teach the Quine–McCluskey heuristic (Quine, 1952, 1955; McCluskey, 1956) Our presentation of the Quine–McCluskey heuristic uses a layered graph over the implicants instead of a tabular approach. We hope that this choice favors notions over notation. Unfortunately, the full details of the heuristic require almost 10 pages. We therefore marked this section with an asterisk.

RECURRENCE EQUATIONS

We use recurrences to describe the cost and delay of circuits defined recursively. We do not introduce the "master theorem" for solving recurrences. The reason is that we find this theorem to be too general for the students at this stage (they do learn it later in the algorithms course). Instead, we resort to solving the specific recurrences we encounter later in the book.

REFERENCES

There are many books on discrete mathematics. Two discrete math books that also treat Boolean algebra and logic design are by McEliece et al. (1989) and Mattson (1993).

There are many books on logic design and computer structure. We were mainly influenced by the book of Müeller and Paul (1996) in the choice of combinational circuits and the description of the processor. We use the simplified timing diagrams from the notes of Litman (2003). These notes also helped with the description of the digital abstraction and flip-flops. The book by Ward and Halstead (1990) describes, among other things, the problem of metastability, arbitration, and the abstraction provided by an instruction set architecture. The book by Ercegovac et al. (1998) uses a hardware description language to design circuits. The book by Ercegovac and Lang (2003) deals with computer arithmetic.

Most textbooks do not introduce Boolean formulas via parse trees. In the book by Howson (1997), propositional logic is described by trees.

More material on finite automata (aka finite state machines) appears in the book by Hopcroft et al. (1979). The book by Savage (1997) starts with basic hardware and ends with advanced material in computability.

The DLX processor architecture was designed by Patterson and Hennessy (1994) as an educational architecture that demonstrates the principles of a RISC processor without the elaborate details of a commercial processor. Our simplified DLX architecture is based on it and on the simplified architecture designed in the RESA lab in Wolfgang Paul's group at the University of the Saarland. See also the book by Müeller and Paul (1996) for a concise description of the DLX architecture and its implementation.

BOOK HOME PAGE

The home page of the book is: http://www.eng.tau.ac.il/~guy/Even-Medina/. We plan to maintain this home page so that it contains the following:

- Slides that we use for teaching
- Errata and a simple form for reporting errors
- Links to simulators (logisim and a DLX assembly simulator)
- Supplementary material

Finally, we would like to thank the anonymous reviewers. Reports of mistakes (all of which are solely our fault) would be greatly appreciated.

The photo on the book cover is a closeup from a photo of Y/Surf/Struc* by Marc Fornes that we saw in the exhibition in Centre Pompidou, Paris. We thank Marc Fornes for sending us the photo and for the permission to use it.

Guy Even and Moti Medina
Tel Aviv, March 2012