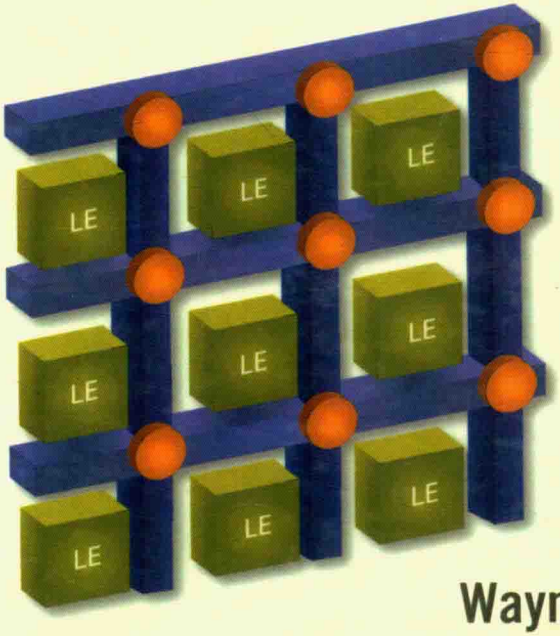
经 典 原 版 书 库

基于FPGA的系统设计

（英文版）

CD-ROM Included

FPGA-Based System Design



附赠 CD-ROM

Wayne Wolf

（美）Wayne Wolf 著

机械工业出版社
China Machine Press

经 典 原

基于FPGA的系统设计

（英文版）

FPGA-Based System Design

（美） Wayne Wolf 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2005-4393

图书在版编目（CIP）数据

基于FPGA的系统设计（英文版）/（美）沃尔夫（Wolf, W.）著. －北京：机械工业出版社，2005.9

（经典原版书库）

书名原文：FPGA-Based System Design

ISBN 7-111-17267-1

Ⅰ. 基…　Ⅱ. 沃…　Ⅲ. 现场可编程门阵列－系统设计－英文　Ⅳ. TP332.1

中国版本图书馆CIP数据核字（2005）第100805号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京京北制版厂印刷·新华书店北京发行所发行

2005年9月第1版第1次印刷

718mm×1020mm 1/16·34印张

印数：0 001 - 3 000册

定价：65.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔

滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

Preface

This book is an experiment. Shortly after completing the third edition of *Modern VLSI Design*, I came to realize that an increasing number of digital designs that used to be built in custom silicon are now implemented in field programmable gate arrays (FPGAs). While traditional VLSI system design won't go away any time soon, an increasing number of designers will work with FPGAs and many of them will never design a custom chip.

However, designers of large FPGA-based systems really do need to understand the fundamentals of VLSI in order to make best use of their FPGAs. While it is true that many system designers simply treat the FPGA as a black box, that approach makes the system designer miss many opportunities to optimize the design to fit within the FPGA. The architecture of FPGAs is largely determined by VLSI constraints: logic element structures, programmable interconnect structures, interconnection networks, configuration, pinout, etc. Understanding how the characteristics of VLSI devices influence the design of FPGA fabrics helps the designer better understand how to take advantage of the FPGA's best characteristics and to minimize the influence of its limitations.

Consider, for example, the interconnection networks in FPGAs. Most modern FPGA architectures provide designers with several different types of wires: local, general-purpose, and global. Why do all these different types of connections exist? Because wires become harder to drive as they grow in length; the extra circuitry required to drive long wires makes them more expensive. Understanding how these different types of interconnect work helps a designer decide whether a particular logic connection requires one of the more expensive types of wires.

Today's FPGAs are truly amazing. High-end FPGAs can hold several million gates. Several FPGAs incorporate one or more CPUs on-chip to provide a complete embedded computing platform. Many of the techniques for designing such large systems are the same whether they are built using FPGAs or custom silicon. This is particularly true when we want to make best use of the silicon characteristics of VLSI structures.

As a result of these advances in VLSI systems, I decided to use *Modern VLSI Design* as a starting point for a new book on FPGA-based system design. Readers of *Modern VLSI Design* will recognize material from most of the chapters in that book. I have extracted material on VLSI characteristics, circuits, combinational and sequential logic design, and system architectures. However, I have also added quite a bit of new material, some of which is specific to FPGAs and some of which is simply a new (and I hope better) look at digital system design.

One of my major goals in writing this book was to provide a useful text for both designers interested in VLSI and those who simply want to use FPGAs to build big systems. Chapter 2 of this book is devoted to a review of VLSI: fabrication, circuits, interconnect characteristics, etc. Throughout the rest of the book, I have tried to break out most details of VLSI design into separate sections that can be skipped by the reader who is not interested in VLSI. However, those who want to understand more about the design of FPGAs as VLSI devices can read this material at their leisure.

Chapter 3 is devoted to a survey of FPGA fabrics—the basic programmable structures of FPGAs. The commercial offerings of companies change all the time, so this chapter is not meant to be a replacement for a data book. Its goal is to introduce some basic concepts in FPGAs and to compare different approaches to solving the basic problems in programmable logic. What to do with these FPGA structures is the subject of the rest of the book.

Chapters 4 and 5 go into detail about combinational and sequential logic design, respectively. They describe methods for the specification and optimization of digital logic to meet the major goals in most design efforts: size, speed, and power consumption. We introduce both Verilog and VHDL in this book. While this book is not intended as a definitive reference on either language, hardware description languages are the medium of choice today for designing digital systems. A basic understanding of these languages, as well as of the fundamentals of hardware description languages, is essential to success in digital system design. We also study the tools for optimizing logic and sequential machine designs in order to understand how to best make use of logic and physical synthesis.

Chapter 6 looks at the structure of large digital systems. This chapter introduces register-transfer design as a methodology for structuring digital designs. It uses a simple DSP as a design example. This DSP is not intended as a state-of-the-art CPU design, but it does allow us to consider a large number of different design problems in a single example.

Chapter 7 caps off the book by studying large-scale systems built with FPGAs. Platform FPGAs that include CPUs and FPGA fabrics allow designers to mix hardware and software on a single chip to solve difficult design problems. Multi-FPGA systems can be used to implement very large designs; a single multi-FPGA system can be programmed to implement many different logic designs.

So what will happen to ASIC design? I don't think it will go away—people will still need the high density and high performance that only custom silicon provides. But I think that FPGAs will become one of the major modes of implementation for digital systems.

Xilinx has graciously allowed us to include CDs that contain the Xilinx Student Edition (XSE) tools. The examples in this book were prepared with these tools and you can follow along with the examples using the tools. You can also use them to create your own examples. Having a working set of tools makes it much easier to practice concepts and I greatly appreciate Xilinx's help with including these books.

You can find additional materials for the book at the Web site:

http://www.princeton.edu/~wolf/fpga-book

The Web site includes overheads for the chapters, pointer to additional Web materials, some sample labs, and errata. Properly accredited instructors can obtain a copy of the instructor's manual by contacting the publisher. I hope you enjoy this book; please feel free to email me at wolf@princeton.edu.

I'd like to thank the students of ELE 462 in the spring of 2003, who were patient with my experimentation on the traditional VLSI course. I'd also like to thank Jiang Xu and Li Shang, my teaching assistants that semester, who improved our infrastructure for FPGA design and helped me debug the DSP design. Mike Butts and Mohammed Khalid gave valuable advice on partitioning algorithms. Steven Brown, Jonathan Rose, Zvonko Vranesic, and William Yu provided figures from their papers to be put directly into the book, providing data straight from the source for the reader as well as simplifying my life as a typesetter. I greatly appreciate the thorough review of a draft of this manuscript given by Andre De Hon, Carl Ebeling, Yankin Tanurhan, and Steve Trimberger; they all made many excellent suggestions on how to improve this book. I greatly appreciate the efforts of Ivo Bolsens, Anna Acevedo, and Jeff Weintraub for access to the knowledge of Xilinx in general and permission to include the Xilinx ISE disks with this book in particular. And, of course, I'd like to thank my editor Bernard Goodwin for his tireless efforts on behalf of this book. All the problems remaining in the book, both small and large, are of course my responsibility.

Wayne Wolf

Princeton, New Jersey

Table of Contents

1 FPGA-Based Systems

FPGAs in system design.
FPGAs *vs.* ASICs.
Design methodologies.

1.1 Introduction

This chapter will set the stage for the rest of the book. The next section talks about some basic concepts in Boolean algebra and schematics. Section 1.3 introduces FPGAs and describes their importance. Section 1.4 describes how we use FPGAs to design complex digital systems.

1.2 Basic Concepts

This section introduces some basic concepts in logic design. If this material is review for you it will help to establish some terminology that we will use throughout the rest of the book. If the material is not review then it should be of use for the rest of the book.

1.2.1 Boolean Algebra

combinational logic functions

We use Boolean algebra to represent the logical functions of digital circuits. Shannon [Sha38] showed that networks of switches (such as light switches)

could be modeled by Boolean functions. Today's logic gates are usually not built from switches but we still consider Boolean algebra to be a fundamental representation. We use Boolean algebra to describe **combinational** (not combinatorial) **logic** functions. Our Boolean functions describe combinations of inputs; we do not use functions with existential ($\exists x\, f(x)$) or universal ($\forall x$ g(x)) quantification.

Figure 1-1 Symbols for functions in logical expressions.

name	symbol
NOT	', ~
AND	· , ∧ , &
NAND	|
OR	+ , ∨
NOR	NOR
XOR	⊕
XNOR	XNOR

notation

We will use fairly standard notation for logic expressions: if a and b are variables, then a' (or $\bar{a}$) is the complement of a, $a \cdot b$ (or ab) is the AND of the variables, and $a + b$ is the OR of the variables. In addition, for the NAND function $(ab)'$ we will use the | symbol [1], for the NOR function $(a + b)'$ we will use $a\ NOR\ b$, and for exclusive-or ($a\ XOR\ b = ab' + a'b$) we will use the ⊕ symbol. (Students of algebra know that XOR and AND form a ring.) Figure 1-1 summarizes the names and symbols for common logic functions.

We use the term **literal** for either the true form (a) or complemented form (a') of a variable. Understanding the relationship between logical expressions and gates lets us study problems in the model that is simplest for that problem, then transfer the results.

1. The Scheffer stroke is a dot with a negation line through it. C programmers should note that this character is used as OR in the C language.

algebraic rules

Let's review some basic rules of algebra that we can use to transform expressions. Some are similar to the rules of arithmetic while others are particular to Boolean algebra:

- **idempotency**: $a \cdot a = a$, $a + a = a$.
- **inversion**: $a + a' = 1$, $a \cdot a' = 0$.
- **identity elements**: $a + 0 = a$, $a \cdot 1 = a$.
- **commutativity**: $a + b = b + a$, $a \cdot b = b \cdot a$.
- **null elements**: $a \cdot 0 = 0$, $a + 1 = 1$.
- **involution** : $(a')' = a$.
- **absorption:** $a + ab = a$.
- **associativity**: $a + (b + c) = (a + b) + c$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
- **distributivity**: $a \cdot (b + c) = ab + ac$, $a + bc = (a + b)(a + c)$.
- **De Morgan's laws**: $(a + b)' = a' \cdot b'$, $(a \cdot b)' = a' + b'$.

completeness and irredundancy

Although Boolean algebra may seem abstract, some of the mathematical results it provides directly relate to the physical properties of logic circuits. Two problems in Boolean algebra that are of importance to logic design are **completeness** and **irredundancy**.

completeness

A set of logical functions is **complete** if we can generate every possible Boolean expression using that set of functions—that is, if for every possible function built from arbitrary combinations of +, ·, and ', an equivalent formula exists written in terms of the functions we are trying to test. We generally test whether a set of functions is complete by inductively testing whether those functions can be used to generate all logic formulas. It is easy to show that the NAND function is complete, starting with the most basic formulas:

- 1: $a|(a|a) = a|a' = 1$.
- 0: $\{a|(a|a)\}|\{a|(a|a)\} = 1|1 = 0$.
- a': $a|a = a'$.
- ab: $(a|b)|(a|b) = ab$.
- $a + b$:$(a|a)|(b|b) = a'|b' = a + b$.

From these basic formulas we can generate all the formulas. So the set of functions {|} can be used to generate any logic function. Similarly, any formula can be written solely in terms of NORs.