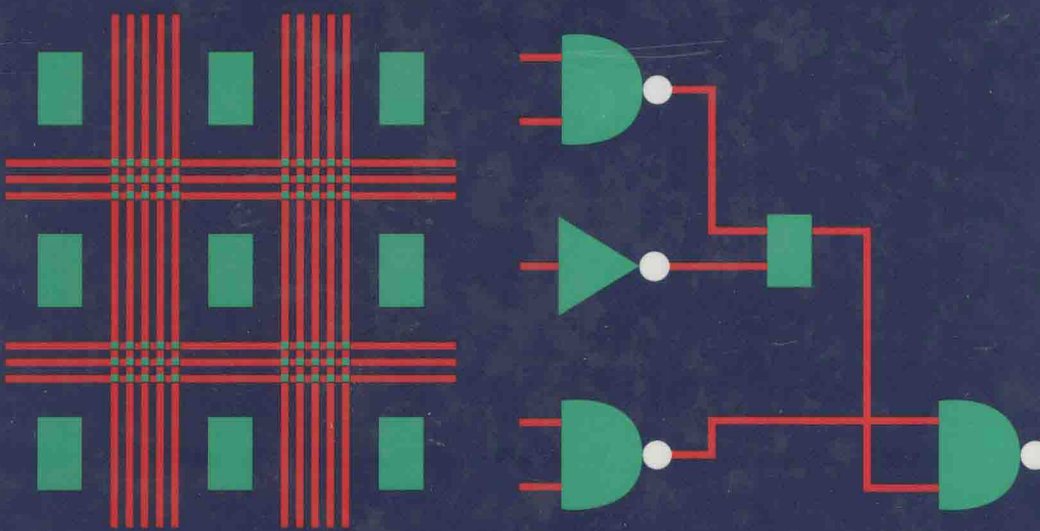
FIELD PROGRAMMABLE GATE ARRAYS

Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems



JOHN V. OLDFIELD
RICHARD C. DORF

FIELD-PROGRAMMABLE GATE ARRAYS

Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems

John V. Oldfield
Syracuse University

Richard C. Dorf
University of California, Davis

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York / Chichester / Brisbane / Toronto / Singapore

This text is printed on acid-free paper.

Library of Congress Cataloging in Publication Data:

Oldfield, John V., 1933–

Field-programmable gate arrays : reconfigurable logic for rapid prototyping and implementation of digital systems / John V. Oldfield, Richard C. Dorf.

p. cm.

"Wiley-Interscience publication."

Includes index.

ISBN 0-471-55665-3 (cloth)

1. Field programmable gate arrays. 2. Programmable array logic. I. Dorf, Richard C. II. Title.

TK7895.G36043 1995

621.39′5—dc20 94-20839

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

FIELD-PROGRAMMABLE GATE ARRAYS

To John Gray, Irene Buchanan,
and Tom Kean, pioneers
of reconfigurable logic

FOREWORD

Not much has changed in computing in the last 40 years.

Most computers still use the Von Neumann stored program model with serial execution of instructions, centralized main memory, and a bus tying it all together. We write algebraic formulas and symbolic constructs and surround them with various loop structures to sequentially process an array of data items. Language statements are translated into a fixed set of machine instructions which are then executed one (or perhaps two or three) at a time. Strange errors of unknown origin frequently occur, and even stranger incantations are often necessary to recover from them.

To be sure, the price and performance of computing hardware have improved dramatically over the years, on a curve of steepness and length that may never be equalled in any man-made technology. Computers have gotten dramatically easier to use, largely because user interfaces have become two-dimensional. Instead of typing arcane commands, we now merely push the mouse to a button displayed on the screen and click.

But the way in which computers are built and programmed hasn't changed all that much. It can be argued that languages, compilers, CAD/CAE tools, and so on have improved significantly, but *our underlying models of computation have remained basically unchanged since the earliest days of computing.*

The reasons for this are simple: Hardware is difficult and time-consuming to build and to change. Serial instruction execution is a simple, very general

paradigm. Parallel architectures can be more powerful, but are less general. A special-purpose circuit can always outperform a microprocessor-based implementation (assuming the same integrated circuit technology is used for both) *for a small class of problems.* Usually the higher performance is achieved through parallelism, either spatial (replication) or temporal (pipelining). The very specialization which provides this parallelism also necessarily limits the range of its application.

I first became aware of a way in which things could be different in 1968, when I came upon Sven Wahlstrom's article in *Electronics* magazine entitled "Programmable Logic Arrays—Cheaper by the Millions." Wahlstrom, Spandorfer, and other pioneers in this area (see references herein) had come up with a radical idea: instead of relying on permanent fuses or 'cutpoints' to customize an array of integrated circuitry, one could simply include additional gates to do the job. The result would be special-purpose hardware which was easily changed as well.

This idea was somewhat heretical when gates were precious, but most people knew this would not always be the case. I became sufficiently excited about this direction to adopt it as my doctoral dissertation research at Carnegie-Mellon University, and completed the thesis in early 1970. At my thesis oral defense, one committee member asked how long it might be before programmable logic would be a reality and in common use in computing hardware. I thought it might be 5–10 years, given that the first microprocessor was already dimly visible in the integrated future. I suspended work in this area awaiting levels of integration which would make my thesis practical.

Graduate students are often known for their great optimism, but they are by no means unique in this regard. In fact it was nearly 15 years before even the first real FPGA was introduced into the marketplace. Today, nearly 28 years after Wahlstrom's first patent was filed (1966), FPGAs are just beginning to have a significant impact. Two application areas seem to dominate their use so far: random 'glue' logic, and emulation of new integrated circuit designs.

But the real impact of FPGAs—*restructurable hardware*, more to the point—is yet to be felt. As the present authors duly point out, the ability to reconfigure hardware on the fly will have vast ramifications. Once suitable design tools and automatic methods become available, designers and programmers will be able to create custom hardware circuitry and pipelines to suit the problem at hand. I like the term "soft hardware," as it suggests that hardware will become as readily created and malleable as software. In a practical sense, this means that the turn-around time for custom hardware will be just as short as software development is today—seconds or minutes, instead of weeks or months.

And the greatest effect of this will be the freedom to think in terms of highly-concurrent hardware structures which implement the desired computation literally, in many cases directly, rather than by emulation or simulation. Von Neumann architectures will be used only when appropriate, and more organic, cus-

tomized architectures can begin to flourish. The computer will *be* the computation desired at that moment.

Programmable logic is not just a better way to do conventional designs, but a doorway to whole new domains.

RICHARD G. SHOUP

Interval Research
Palo Alto, California
June 1994

PREFACE

The field-programmable gate array (FPGA) is a relatively new type of component for the construction of electronic systems, particularly those using digital, or more correctly *logical*, circuit principles. It consists of an array of functional blocks along with an interconnection network, and as the name implies, its configuration can be determined in the field, that is, at the *point of application*. The specific function of each block and the connections between blocks are prescribed by the user. For the most part we will be concerned with *reprogrammable* FPGAs in this text, since they offer more flexibility than fuse-programmable devices. This choice is not a matter of taste. Already the FPGA market has a wide range of architectures and alternative ways of controlling configurations. We do not dismiss fuse-programmable architectures, which clearly already provide fast, economical, and compact implementations for logic designs. But it is almost like comparing a ROM-based computer with the more-general RAM-based one. Moreover, this field is at such an early stage of development that no one yet knows the most appropriate design methodologies and run-time environments. The FPGA takes its place in the continuing evolution of very-large-scale integrated (VLSI) circuit technology toward denser and faster circuits. It already provides, for many applications, an adequate number of transistors in a single chip package for the functional blocks, switches for the routing network, and the memory capacity to control both. The prospects for further increases in both circuit density and speed of operation are excellent, since VLSI circuit density continues to double every 2 to 3 years or so. Indeed, billion-transistor (10^9) chips are anticipated by the turn of the century.

Clearly, the ability to reconfigure an electronic component, either to correct an error, or to change an application, has economic benefits, but we suggest that the impact of the FPGA concept is much more profound, in that a logic system may be *dynamically* reconfigured to match changing circumstances. This is a new computing paradigm whose potential is far-reaching. Although present-day FPGA components have only a few thousand or tens of thousands of gates, future ones will have hundreds of thousands or eventually millions of gates. We may expect to see the development of pipelined and parallel processing in which the system configuration changes dynamically. For example, a graphic processing operation might use three processors for three-dimensional calculations and switch to two processors for screen-related calculations. An image processing system could divide computations into parallel tasks for a large number of processors realized within the same FPGA chip. While such schemes may seem futuristic, they suggest that the FPGA should not be thought of as just a VLSI component that replaces others, but as a concept with distinctive new possibilities.

Returning to present-day realizations, the FPGA has significant advantages for the development of prototype systems and their early introduction to the market. The benefits are similar to those associated with the introduction of the microprocessor in the late 1970s, such as programmability and adaptability, but with additional advantages in speed, compactness, and design protection. From an educational viewpoint, designing with FPGAs requires computer assistance at almost every stage of design, including detailed specification, simulation, placement, and routing, and calls for an overall systematic *design methodology*. We consider this to be an important aspect in the education of future engineers, which should improve both the performance and the quality of the systems they produce. At the same time, the increase in designer productivity makes it possible to consider alternative implementations at a higher level than previously, and should not cramp the creativity of a system designer.

ABOUT THE BOOK

The book is intended to serve the needs of several constituencies:

- As a text to accompany an undergraduate course in which students are introduced to FPGAs for the first time, and in which laboratory work is a likely accompaniment.
- As a graduate-level text, with introductory aspects as well as more advanced applications and interest in the underlying VLSI structures of FPGAs, computer-aided design (CAD) algorithms for functional and physical layout.
- For the professional engineer who wishes to obtain an appreciation of FPGA principles and prospects.

In all cases, we assume a familiarity with the basic principles of logic circuits and their realization as digital systems with commonly available components. We have paid more attention than is customary to the nature of the emerging industry and business aspects, since we believe that these should be of interest to all of the preceding categories of reader. VLSI technology has been an exciting business to be in during its formative years, and technical considerations cannot be divorced from financial and marketing aspects.

Chapter 1 discusses the alternative ways in which a logic system can be realized, and the variety of technologies available for the purpose. It considers the design process itself as worthy of study, and highlights various design methodologies that have emerged. Chapter 2 is mostly a review of logical and electrical aspects of digital systems with which the reader should already be familiar, with additional material on implementing state machines so as to take advantage of common FPGA properties. Chapter 3 reviews the architecture of FPGAs, along with the options and trade-offs for different approaches. While the industry is still in its formative years, radically different directions have been taken in setting up the underlying structures, and new architectures will continue to emerge. The chapter concludes with a discussion of appropriate benchmarks for FPGA comparison. Chapter 4 is concerned with CAD for FPGA applications, and attempts to give a comprehensive, manufacturer-independent view of the place of CAD tools, as well as some elementary and specific examples for two particular systems. In Chapter 5 there are a number of case studies of small- and modest-scale designs implemented on different FPGA architectures, along with a substantial example of a parallel controller that stretched the limits of an FPGA architecture and its supporting CAD software. Chapter 6 is concerned with advanced applications of FPGAs, including several large-scale examples in which the FPGA is used as a novel computing structure. The term "custom computing machine" has been coined to describe such systems, and a number of illustrations already exist to show the enormous potential of arrays of FPGA chips for some of the present-day "Grand Challenge" problems, such as high-speed string comparison for the Human Genome Project. Business issues are the focus of Chapter 7, and include a perspective on the start-up and growth of new companies in this field, as well as the relevance of market factors to FPGA applications in the context of other ways of implementing digital systems. The chapter concludes with a short review of intellectual property aspects—how to protect one's own or one's company's ideas in a highly competitive environment. The FPGA has interesting new possibilities for design protection, and can be made extraordinarily secure compared with other VLSI technologies. Chapter 8 was added to the text as late as possible and reviews some recent developments. Much of this chapter has been contributed by the manufacturers concerned, and we appreciate their cooperation. There is a Glossary containing explanations of the technical terms, particularly acronyms, used in the field.

Readers should be aware that we have not included full details of FPGA technical data, nor specific information on particular CAD support software. This decision is based on the fact that well-written technical handbooks are

available from the FPGA manufacturers and CAD software vendors, and also recognizes that the publication cycle of a text is long, compared with ever-changing computer software and the intervals between announcements of new FPGA families and devices. For the most part our examples are from Xilinx Inc. and Algotronix Ltd., and readers should contact their nearest distributor for technical information.

A few months before the manuscript was completed, Algotronix Ltd. was acquired by Xilinx Inc., and in the new circumstances two of the original authors, John Gray and Tom Kean, had to withdraw. We believe that the pioneering work they carried out on fine-grain FPGA structures will be recognized as a valuable and complementary alternative to more widely accepted methods used in the industry.

BACKGROUND AND APPLICABILITY

As mentioned earlier, we assume that the reader is already familiar with logic circuits, including the concepts of state machines and their realization. A university-level introductory course should be adequate, and even better if supplemented by laboratory experience, for example, with a logic breadboarding system. Chapter 2 provides relevant review material. In considering using this book as a text, it is worth giving thought to the long-term impact of this technology on both undergraduate and graduate education, since FPGAs could be relevant to a *sequence* of classes related to digital design and computer architecture.

This book could be used as a text for an introductory undergraduate- or graduate-level class on field-programmable logic arrays and their applications. This might be offered in electrical engineering, computer engineering, or computer science. It should be combined with exposure to a practical FPGA system, including supporting CAD hardware and software. It is planned to produce an Instructor's Manual which will include suggestions and advice, as well as worked examples, suggestions for projects, and solutions. This manual will also suggest useful sources of information. The previously listed classes could focus on one particular FPGA architecture, but should include comparisons with other architectures, if at all possible. We recommend that the broader aspects, including design methodologies and business issues, be included in such courses.

The material in the text can be usefully linked to an introductory VLSI design class. Often such courses emphasize full-custom design, and already include most of the basic elements found in an FPGA. While the underlying architecture is highly proprietary, many of the issues in full-custom design, such as global and local communication, memory, input/output pads, general-purpose logic blocks, are particularly relevant. The FPGA also allows much faster turnaround than for full-custom design, and exposure to this material gives students a more balanced perspective on digital system implementation.

The book could be used in conjunction with a project-oriented class, particularly for computer engineering or computer science students. Assuming that adequate CAD support is available, FPGAs should contribute to a considerable increase in student productivity, compared with traditional digital breadboard systems. An added advantage is the emphasis on a systematic approach to engineering design, and the high-quality design documentation that is readily produced from the CAD system. This is highly relevant to the emphasis placed on design by the U.S. Accreditation Board for Engineering and Technology (ABET), which defines "engineering design" as [ABET90]:

> . . . the process of devising a system, component, or process to meet desired needs. . . . Among the fundamental elements of the design process are the establishment of objectives and criteria, synthesis, analysis, construction, testing, and evaluation. The engineering component of a curriculum must include at least some of the following features: development of student creativity, use of open-ended problems, development and use of design methodology, formulation of design problem statements, consideration of alternative solutions, feasibility considerations, and detailed system descriptions. Further, it is essential to include a variety of realistic constraints such as economic factors, safety, reliability, aesthetics, ethics, and social impact.

We submit that FPGAs can provide an outstanding *design* environment for engineering education.

Computer architecture courses are often accompanied by the use of a simulator for trying out proposed architectures. The FPGA and its CAD simulators should be very attractive for such courses, with the added advantage of implementation at speeds much nearer to those of a real system than the slowness of a software simulator. For some of the less-complex FPGA architectures, it is possible to independently develop CAD software, including simulators, and much exploration can be done without the actual hardware parts. The demise of Algotronix as a source of parts should not discourage exploration of this and other novel architectures by simulation. We have already found such a simulator useful for debugging purposes, and they are easy to develop for fine-grain architectures.

At the graduate-level, FPGAs can provide useful target architectures for classes in CAD for digital systems. These include physical design issues such as partitioning, placement, and routing, along with logic simplification and simulation. Clearly, simpler architectures are to be preferred to more complex ones, and it may be necessary to approach the manufacturer for details of CAD file formats, and so forth. Even outside the classroom context, students should be encouraged to develop software aids for FPGAs. These goals would be easier to attain if FPGA vendors actively encouraged more open design systems for their products.

A superficial pass through the book may detect more emphasis on *wiring management* and *layout* issues than might be expected for a component that, in an ideal world, would be designed and configured totally automatically. There is a close parallel with full-custom VLSI design and much scope for designer

ingenuity by exploiting the inherent massive parallelism, particularly by pipelining.

In summary, the FPGA can provide a realistic implementation vehicle for complex digital systems that are beyond the limits of conventional breadboard logic, and where there is insufficient time for full-custom, standard-cell, or even mask-programmable gate array implementation.

ERRORS AND OMISSIONS

Despite the best efforts of the author team, errors will get through to the printed version. We would appreciate being informed of any such errors, and will endeavor to see that they are identified and corrected in subsequent printings.

ACKNOWLEDGMENTS

First we wish to express our sincere gratitude to our former coauthors John Gray and Tom Kean. Without their enthusiasm, experience, and cooperation the book would never have been started. The authors wish to thank a number of individuals who have contributed to the development of this book. Erik Dagless, Jonathan M. Saul, and Tomasz Kozlowski of the University of Bristol provided the material on parallel state machines in Chapter 2, and the two former authors provided the video controller example given in Chapter 5. Christopher Kappler of Syracuse University contributed the sorter design given in the same chapter, and, along with Kang Shen, developed the self-timed approach to genetic string matching described in Chapter 6. Roberto Melo provided the multiplier examples in Chapter 5 as part of a wider study that included standard-cell designs. Fred Furtek kindly agreed to the insertion of his original and thought-provoking approach to motion estimation that is reprinted, with permission, at the end of Chapter 6. Ted Hagelin of the Syracuse University College of Law provided much of the background on intellectual property issues in the latter half of Chapter 7, and Harold Burstyn made some important corrections to one of the author's interpretations. Amr Mohsen, Wayne Luk, Steven Sillich, and Steven Trimberger were very helpful in commenting on draft versions of the text. Ed Sibert of Syracuse University devoted countless late hours in the production of the final manuscript in Latex. Mary Haas-Wendel became involved in the development of the manuscript at a late stage. She made significant contributions to the business issues chapter based on her experience in the corporate world as well as providing much editorial help. Finally, there are a host of persons in both academia and industry who have encouraged the authors directly and indirectly, through papers and discussions.

The authors are of course solely responsible for the views expressed in this book.

BIBLIOGRAPHY

[ABET90] *Criteria for Accrediting Programs in Engineering*, United States Accrediting Board for Engineering and Technology, Inc., New York, 1990.

[NSF92] Committee on Physical, Mathematical, and Engineering Sciences Federal Coordinating Council for Science, Engineering, and Technology Office of Science and Technology, "Grand Challenges: High Performance Computing and Communications," National Science Foundation, Washington, D.C., 1992.

ACRONYMS

ABEL	Proprietary language for describing state machines
ABET	Accreditation Board for Engineering and Technology
A/D	Analog-Digital
ALU	Arithmetic-logic unit
ASIC	Application-specific integrated circuit
AT	IBM type of Personal Computer
CAD	Computer-aided design
CAE	Computer-aided engineering
CAL	Configurable Array Logic
CBIC	Cell-based integrated circuit
CHS	Configurable Hardware System
CLB	Configurable Logic Block
CLS	Configurable Logic Software
CPLD	Complex programmable logic device
CMOS	Complementary metal-oxide semiconductor
CUPS	Cell updates per second
D/A	Digital-analog
DES	Data Encryption Standard
DIL	Dual in-line
DRAM	Dynamic random-access memory
DSP	Digital signal processor

ECAD	Electronic computer-aided design
EDIF	Electronic Design Interchange Format
EEPROM	Electrically-erasable programmable read-only memory
EPLD	Electrically-programmable logic device
EPROM	Erasable programmable read-only memory
ESD	Electrostatic discharge
FET	Field-effect transistor
FPGA	Field-programmable gate array
FPCB	Field-programmable circuit board
FPIC	Field-programmable interconnect component
FPID	Field-programmable interconnect device
FPLA	Field-programmable logic array
FSM	Finite-state machine
GAL	Generic Array Logic
HDL	Hardware definition language
HP	Hewlett-Packard
IC	Integrated circuit
IOB	Input/output block
IOE	Input/output element
IP	Initial permutation
ISA	Industry-Standard Architecture
JTAG	Joint Test Action Group
LAB	Logic array block
LCA	Logic cell array
LE	Logic element
LED	Light-emitting diode
LFSR	Linear feedback shift register
LPM	Library of parameterized modules
LSI	Large-scale integration
MCUPS	Millions of cell updates per second
MIPS	Millions of instructions per second
MOS	Metal-oxide-semiconductor
MOSFET	MOS field-effect transistor
MPGA	Mask-programmable gate array
MIS	medium-scale integration
MUX	Multiplexer