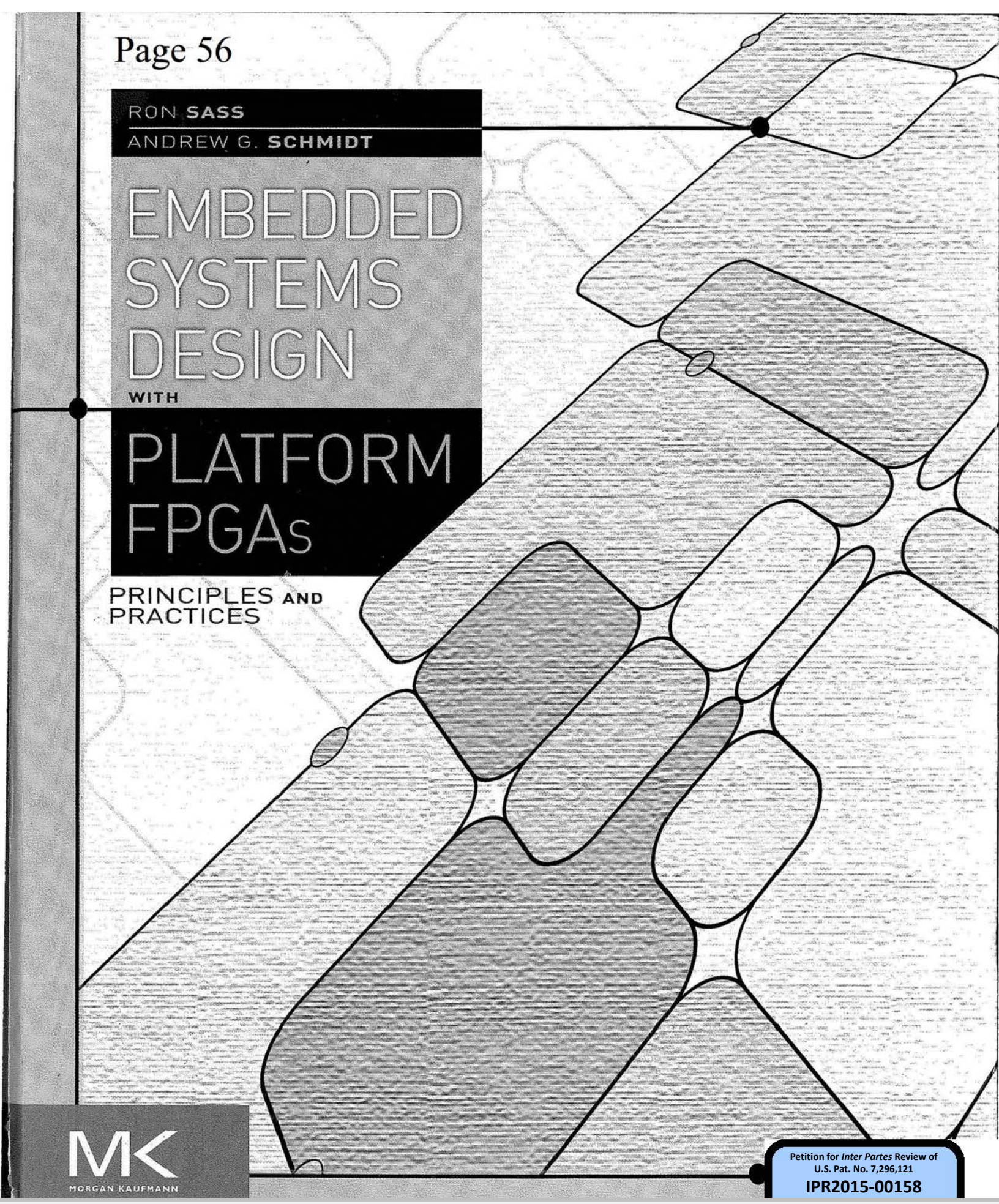
RON SASS
ANDREW G. SCHMIDT

EMBEDDED SYSTEMS DESIGN WITH PLATFORM FPGAs

PRINCIPLES AND PRACTICES



MK
MORGAN KAUFMANN

Acquiring Editor: Todd Green
Development Editor: Nate McFadden
Project Manager: André Cuello
Designer: Alisa Andreola

Morgan Kaufmann Publishers is an imprint of Elsevier.
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

This book is printed on acid-free paper.

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors assume any liability for any injury and/or damage to persons or property as a matter of product liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
Sass, Ronald (Ronald R.)
Embedded systems design with platform FPGAs: Principles and Practices / Ronald Sass.
p. cm.
Includes bibliographical references.
ISBN 978-0-12-374333-6
1. Embedded computer systems. 2. Field programmable gate arrays. I. Title.
TK7895.E42S27 2010
004.16–dc22

2010010632

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

For information on all Morgan Kaufmann publications
visit our Web site at *www.mkp.com* or *www.elsevierdirect.com*

Printed in the United States of America
10 11 12 13 5 4 3 2 1

Typeset by: diacriTech, India

hardware. Over time, it was recognized that the descriptions could be used to simulate hardware circuits on a general-purpose processor. This process of translating an HDL source into a form suitable for a general-purpose processor to mimic the hardware described is called ***simulation***. Simulation has proven to be an extremely useful tool for developing hardware and verifying the functionality before physically manufacturing the hardware. It was only later that people began to ***synthesize*** hardware, automatically generating the logic configuration for the specified device from the hardware description language.

Unfortunately, while simulation provided a rich set of constructs to help the designer test and analyze the design, many of these constructs extend beyond what is physically implementable within hardware (on the FPGA) or synthesize inefficiently into the FPGA resources. As a result, only a subset of hardware description languages can be used to synthesize designs to hardware. The objective of this section is to present two of the more popular hardware description languages, VHDL and Verilog. It is arguable to which is better suited for FPGA design; both are presented here for completeness, but it is left to the readers to decide which (if any) is best suited for their needs. We will focus on VHDL throughout the remainder of this book as it becomes redundant to give two examples, one in VHDL and one in Verilog, for every concept. That being said, we will also focus within this section on synthesizable HDL and how it maps to the previous section's components.

2.4.1. VHDL

VHDL, which stands for VHSIC[1] Hardware Description Language, describes digital circuits. In simulation, VHDL source files are analyzed and a description of the behavior is expressed in the form of a netlist. A ***netlist*** is a computer representation of a collection of logic units and how they are to be connected. The logic units are typically AND/OR/NOT gates or some set of primitives that makes sense for the target (4-LUTs, for example). The behavior of the circuit is exercised by providing a sequence of inputs. The inputs, called *test vectors*, can be created manually or by writing a program/script that generates them. The component that is generating test vectors and driving the device under test is typically called a *test bench*.

Synthesizable VHDL

In VHDL, there are two major styles or forms of writing hardware descriptions. Both styles are valid VHDL codes; however, they