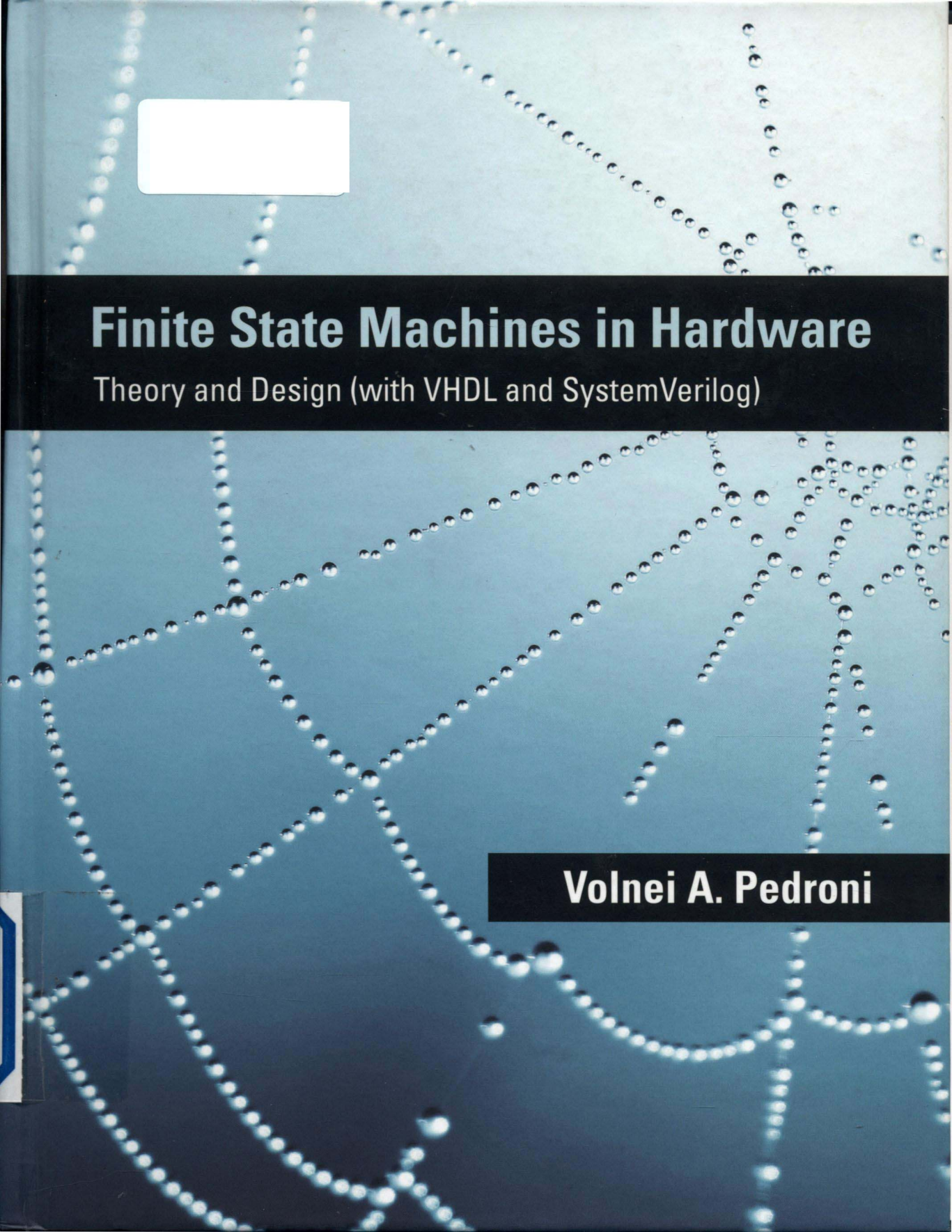

Finite State Machines in Hardware
Theory and Design (with VHDL and SystemVerilog)
Volnei A. Pedroni

Finite State Machines in Hardware

Theory and Design (with VHDL and SystemVerilog)

Volnei A. Pedroni

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Stone Sans and Stone Serif by Toppan Best-set Premedia Limited, Hong Kong. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Pedroni, Volnei A.
Finite state machines in hardware : theory and design (with VHDL and SystemVerilog) / Volnei A. Pedroni.
pages cm
Includes bibliographical references and index.
ISBN 978-0-262-01966-8 (hardcover : alk. paper) 1. SystemVerilog (Computer hardware description language) 2. VHDL (Computer hardware description language) 3. Sequential machine theory—Data processing. 4. Computer systems—Mathematical models. I. Title.
TK7885.7.P443 2013
621.39'2—dc23

2013009431

10 9 8 7 6 5 4 3 2 1

Finite State Machines in Hardware

Preface

This book deals with the crucial issue of implementing Finite State Machines (FSMs) in hardware, which has become increasingly important in the development of modern, complex digital systems.

Because FSM is a modeling technique for synchronous digital circuits, a detailed review of synchronous circuits in general is also presented, to enable in-depth and broad coverage of the topic.

A new classification for FSMs from a hardware perspective is introduced, which places any state machine under one of three categories: *regular machines*, *timed machines*, or *recursive machines*. The result is a clear, precise, and *systematic* approach to the construction of FSMs in hardware.

Many examples are presented in each category, from datapath controllers to password readers, from car alarms to multipliers and dividers, and from triggered circuits to serial data communications interfaces.

Several of the state machines, in all three categories, are subsequently implemented using VHDL and SystemVerilog. It starts with a review of these hardware description languages, accompanied by new, detailed templates. The subsequent designs are always complete and are accompanied by comments and simulation results, illustrating the design's main features.

Numerous exercises are also included in the chapters, providing an invaluable opportunity for students to play with state machines, VHDL and SystemVerilog languages, compilation and simulation tools, and FPGA development boards.

In summary, the book is a complete, modern, and interesting guide on the theory and physical implementation of synchronous digital circuits, particularly when such circuits are modeled as FSMs.

Acknowledgments

I want to express my gratitude to Bruno U. Pedroni for his invaluable help and suggestions during the initial phase of the book. I am also grateful to the personnel at MIT Press, especially Marc Lowenthal, acquisitions editor, for his assistance during the early phases of the book; and Marcy Ross, production editor, for her excellent work and endless patience during the editing and production phases.

Contents

1 The Finite State Machine Approach

1.1 Introduction

This chapter presents fundamental concepts and introduces new material on the finite state machine (FSM) approach for the modeling and design of sequential digital circuits.

A summary of the notation used in the book is presented in table 1.1.

1.2 Sequential Circuits and State Machines

Digital circuits can be classified as *combinational* or *sequential*. A combinational circuit is one whose output values depend solely on the present input values, whereas a sequential circuit has outputs that depend on previous system states. Consequently, the former is memoryless, whereas the latter requires some sort of memory (generally, D-type flip-flops [DFFs], reviewed in section 2.2).

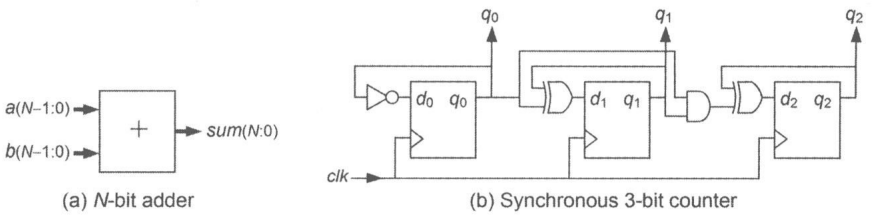
An example of a combinational circuit is presented in figure 1.1a, which shows an *N*-bit adder; because the present sum is not affected by previous sums computed by the circuit, it is combinational. An example of sequential circuit is depicted in figure 1.1b, which shows a synchronous three-bit counter (it counts from 0 to 7); because its output depends on the system state (for example, if the current output is 5, then the next will be 6), it is a sequential circuit. Note the presence of a clock signal in the latter.

An often advantageous model for sequential circuits is presented in figure 1.2a, which consists of a combinational logic block in the forward path and a memory (DFFs) in the feedback loop. When this architecture is used, a *finite state machine* (FSM) results. Note that the state presently stored in the memory is called *pr_state*, and the state to be stored by the DFFs at the next (positive) clock transition is called *nx_state*.

An example of such a modeling technique is depicted in figure 1.2b, which shows the same circuit of figure 1.1b, now reorganized according to the architecture of

Table 1.1

Item	Representation	Examples
Signal names	In italic	*a*, *x*, *clk*, *rst*, *ena*, *WE*
Active-low signal names	In italic, followed by an *n*	*WEn*, *rstn*, *rst_n*
Single-bit values	Within a pair of single quotes	'0', '1', 'X', '–', 'Z'
Multi-bit values	Within a pair of double quotes	"00", "1000", "ZZZZ"
Integers	Without quotes	1000, 5, –256
Allowed bit values	'0' or 'L' for low logic level '1' or 'H' for high logic level 'X' or '–' for "don't care" 'Z' for high impedance	*y* = '0' or *y* = 'L' *y* = '1' or *y* = 'H' *y* = 'X' or *y* = '–' *y* = 'Z'
Bit indexing (outside VHDL or SystemVerilog codes)	Between parentheses, with a colon	*x*(7:0) means that *x* has 8 bits, *x*(7) is the most significant bit, *x*(0) is the least significant bit
Reset and clear signals	- Called *reset* (*rst*) when asynchronous (resets the circuit regardless of the clock) - Called *clear* (*clr*) when synchronous (effective only at the proper clock edge)	if *rst* = '1' then … if *clr* = '1' then …
Transition conditions in state diagrams	& means *and* | means *or* ! and ≠ mean *not* or *different* ¯ (bar) and ′ mean *not* or *inversion* 'X' and '–' mean "don't care" for a single-bit value "XX…" and "– –…" mean "don't care" for a multi-bit value – means "don't care" for an integer	if *a* = '1' & *b* = '0' then … if *a* = '1' | *b* = '0' then … if *x* != *a* then … or if *x* ≠ *a* then … *y* = *x*′ *x* = '1' & *y* = '–' If (*a* = "111" & *b* = "0–0") | *c* = "000" then … *m* = 5 & *n* = –



(a) *N*-bit adder (b) Synchronous 3-bit counter

Figure 1.1
Examples of (a) combinational and (b) sequential circuits.