

The background of the cover is a microscopic view of a circuit board, showing numerous small, circular components arranged in various patterns. A white rectangular label is positioned in the upper left corner.

Finite State Machines in Hardware

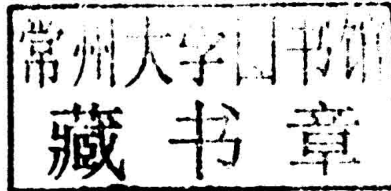
Theory and Design (with VHDL and SystemVerilog)

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

© 2013 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu.

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

Preface xi

Acknowledgments xiii

1 The Finite State Machine Approach 1

- 1.1 Introduction 1
- 1.2 Sequential Circuits and State Machines 1
- 1.3 State Transition Diagrams 4
- 1.4 Equivalent State Transition Diagram Representations 6
- 1.5 Under- and Overspecified State Transition Diagrams 8
- 1.6 Transition Types 11
- 1.7 Moore-to-Mealy Conversion 12
- 1.8 Mealy-to-Moore Conversion 14
- 1.9 Algorithmic State Machine Chart 15
- 1.10 When to Use the FSM Approach 16
- 1.11 List of Main Machines Included in the Book 17
- 1.12 Exercises 18

2 Hardware Fundamentals—Part I 21

- 2.1 Introduction 21
- 2.2 Flip-Flops 21
- 2.3 Metastability and Synchronizers 24
- 2.4 Pulse Detection 28
- 2.5 Glitches 29
- 2.6 Pipelined Implementations 32
- 2.7 Exercises 33

3 Hardware Fundamentals—Part II 39

- 3.1 Introduction 39
- 3.2 Hardware Architectures for State Machines 39
- 3.3 Fundamental Design Technique for Moore Machines 41

3.4	Fundamental Design Technique for Mealy Machines	44
3.5	Moore versus Mealy Time Behavior	46
3.6	State Machine Categories	47
3.7	State-Encoding Options	49
3.7.1	Sequential Binary Encoding	49
3.7.2	One-Hot Encoding	50
3.7.3	Johnson Encoding	50
3.7.4	Gray Encoding	50
3.7.5	Modified One-Hot Encoding with All-Zero State	51
3.7.6	Other Encoding Schemes	52
3.8	The Need for Reset	52
3.9	Safe State Machines	54
3.10	Capturing the First Bit	56
3.11	Storing the Final Result	58
3.12	Multimachine Designs	60
3.13	State Machines for Datapath Control	62
3.14	Exercises	67
4	Design Steps and Classical Mistakes	73
4.1	Introduction	73
4.2	Classical Problems and Mistakes	73
4.2.1	Skipping the State Transition Diagram	73
4.2.2	Wrong Architecture	73
4.2.3	Incorrect State Transition Diagram Composition	74
4.2.4	Existence of State Bypass	75
4.2.5	Lack of Reset	75
4.2.6	Lack of Synchronizers	76
4.2.7	Incorrect Timer Construction	76
4.2.8	Incomplete VHDL/SystemVerilog Code	76
4.2.9	Overregistered VHDL/SystemVerilog Code	78
4.3	Design Steps Summary	79
5	Regular (Category 1) State Machines	81
5.1	Introduction	81
5.2	Architectures for Regular (Category 1) Machines	82
5.3	Number of Flip-Flops	84
5.4	Examples of Regular (Category 1) Machines	84
5.4.1	Small Counters	84
5.4.2	Parity Detector	85
5.4.3	Basic One-Shot Circuit	86
5.4.4	Temperature Controller	88
5.4.5	Garage Door Controller	89

5.4.6	Vending Machine Controller	90
5.4.7	Datapath Control for an Accumulator	91
5.4.8	Datapath Control for a Greatest Common Divisor Calculator	93
5.4.9	Generic Sequence Detector	95
5.4.10	Transparent Circuits	96
5.4.11	LCD, I ² C, and SPI Interfaces	97
5.5	Exercises	97
6	VHDL Design of Regular (Category 1) State Machines	105
6.1	Introduction	105
6.2	General Structure of VHDL Code	105
6.3	VHDL Template for Regular (Category 1) Moore Machines	107
6.4	Template Variations	111
6.4.1	Combinational Logic Separated into Two Processes	111
6.4.2	State Register Plus Output Register in a Single Process	112
6.4.3	Using Default Values	112
6.4.4	A Dangerous Template	113
6.5	VHDL Template for Regular (Category 1) Mealy Machines	114
6.6	Design of a Small Counter	116
6.7	Design of a Garage Door Controller	120
6.8	Design of a Datapath Controller for a Greatest Common Divisor Calculator	123
6.9	Exercises	126
7	SystemVerilog Design of Regular (Category 1) State Machines	129
7.1	Introduction	129
7.2	General Structure of SystemVerilog Code	129
7.3	SystemVerilog Template for Regular (Category 1) Moore Machines	130
7.4	SystemVerilog Template for Regular (Category 1) Mealy Machines	133
7.5	Design of a Small Counter	135
7.6	Design of a Garage Door Controller	137
7.7	Design of a Datapath Controller for a Greatest Common Divisor Calculator	140
7.8	Exercises	141
8	Timed (Category 2) State Machines	143
8.1	Introduction	143
8.2	Architectures for Timed (Category 2) Machines	144
8.3	Timer Interpretation	146
8.3.1	Time Measurement Unit	146
8.3.2	Timer Range	146
8.3.3	Number of Bits	146

8.4	Transition Types and Timer Usage	147
8.5	Timer Control Strategies	147
8.5.1	Preliminary Analysis	148
8.5.2	Timer Control Strategy #1 (Generic)	149
8.5.3	Timer Control Strategy #2 (Nongeneric)	150
8.5.4	Time Behavior of Strategies #1 and #2	151
8.6	Truly Complementary Time-Based Transition Conditions	153
8.7	Repetitively Looped State Machines	154
8.8	Time Behavior of Timed Moore Machines	155
8.9	Time Behavior of Timed Mealy Machines	156
8.10	Number of Flip-Flops	158
8.11	Examples of Timed (Category 2) Machines	158
8.11.1	Blinking Light	159
8.11.2	Light Rotator	160
8.11.3	Switch Debouncer	161
8.11.4	Reference-Value Definer	163
8.11.5	Traffic Light Controller	166
8.11.6	Car Alarm (with Chirps)	167
8.11.7	Password Detector	168
8.11.8	Triggered Circuits	170
8.11.9	Pulse Shifter	172
8.11.10	Pulse Stretchers	173
8.12	Exercises	176
9	VHDL Design of Timed (Category 2) State Machines	185
9.1	Introduction	185
9.2	VHDL Template for Timed (Category 2) Moore Machines	185
9.3	VHDL Template for Timed (Category 2) Mealy Machines	189
9.4	Design of a Light Rotator	191
9.5	Design of a Car Alarm (with Chirps)	194
9.6	Design of a Triggered Monostable Circuit	198
9.7	Exercises	201
10	SystemVerilog Design of Timed (Category 2) State Machines	207
10.1	Introduction	207
10.2	SystemVerilog Template for Timed (Category 2) Moore Machines	207
10.3	SystemVerilog Template for Timed (Category 2) Mealy Machines	210
10.4	Design of a Light Rotator	212
10.5	Design of a Car Alarm (with Chirps)	214
10.6	Design of a Triggered Monostable Circuit	217
10.7	Exercises	220

11	Recursive (Category 3) State Machines	221
11.1	Introduction	221
11.2	Recursive (Category 3) State Machines	222
11.3	Architectures for Recursive (Category 3) Machines	223
11.4	Category 3 to Category 1 Conversion	224
11.5	Repetitively Looped Category 3 Machines	225
11.6	Number of Flip-Flops	226
11.7	Examples of Recursive (Category 3) State Machines	226
11.7.1	Generic Counters	226
11.7.2	Long-String Comparator	228
11.7.3	Reference-Value Definer	229
11.7.4	Reference-Value Definer with Embedded Debouncer	231
11.7.5	Datapath Control for a Sequential Multiplier	232
11.7.6	Sequential Divider	234
11.7.7	Serial Data Receiver	236
11.7.8	Memory Interface	237
11.8	Exercises	240
12	VHDL Design of Recursive (Category 3) State Machines	245
12.1	Introduction	245
12.2	VHDL Template for Recursive (Category 3) Moore Machines	245
12.3	VHDL Template for Recursive (Category 3) Mealy Machines	248
12.4	Design of a Datapath Controller for a Multiplier	249
12.5	Design of a Serial Data Receiver	252
12.6	Design of a Memory Interface	256
12.7	Exercises	261
13	SystemVerilog Design of Recursive (Category 3) State Machines	265
13.1	Introduction	265
13.2	SystemVerilog Template for Recursive (Category 3) Moore Machines	265
13.3	SystemVerilog Template for Recursive (Category 3) Mealy Machines	267
13.4	Design of a Datapath Controller for a Multiplier	268
13.5	Design of a Serial Data Receiver	271
13.6	Design of a Memory Interface	273
13.7	Exercises	278
14	Additional Design Examples	279
14.1	LCD Driver	279
14.1.1	Alphanumeric LCD	279
14.1.2	Typical FSM Structure for Alphanumeric LCD Drivers	283
14.1.3	Complete Design Example: Clock with LCD Display	284

14.2	I ² C Interface	290
14.2.1	I ² C Bus Structure	290
14.2.2	Open-Drain Outputs	291
14.2.3	I ² C Bus Operation	292
14.2.4	Typical FSM Structure for I ² C Applications	295
14.2.5	Complete Design Example: RTC (Real-Time Clock) Interface	296
14.3	SPI Interface	305
14.3.1	SPI Bus Structure	305
14.3.2	SPI Bus Operation	306
14.3.3	Complete Design Example: FRAM (Ferroelectric RAM) Interface	307
14.4	Exercises	315
15	Pointer-Based FSM Implementation	319
15.1	Introduction	319
15.2	Single-Loop FSM	319
15.3	Serial Data Transmitter	321
15.4	Serial Data Receiver	322
15.5	SPI Interface for an FRAM	325
15.6	Exercises	329
	Bibliography	331
	Index	333

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 <i>italic</i>	<i>a</i> , <i>x</i> , <i>clk</i> , <i>rst</i> , <i>ena</i> , <i>WE</i>
Active-low signal names	In <i>italic</i> , followed by an <i>n</i>	<i>WEn</i> , <i>rstn</i> , <i>rst_n</i>
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 <i>reset</i> (<i>rst</i>) when asynchronous (resets the circuit regardless of the clock) - Called <i>clear</i> (<i>clr</i>) when synchronous (effective only at the proper clock edge)	if $rst = '1'$ then ... if $clr = '1'$ then ...
Transition conditions in state diagrams	& means <i>and</i> means <i>or</i> ! and \neq mean <i>not</i> or <i>different</i> - (bar) and ' mean <i>not</i> or <i>inversion</i> '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 $x \neq a$ then ... $y = x'$ $x = '1'$ & $y = '-'$ If $(a = "111" \& b = "0-0")$ $c = "000"$ then ... $m = 5$ & $n = -$

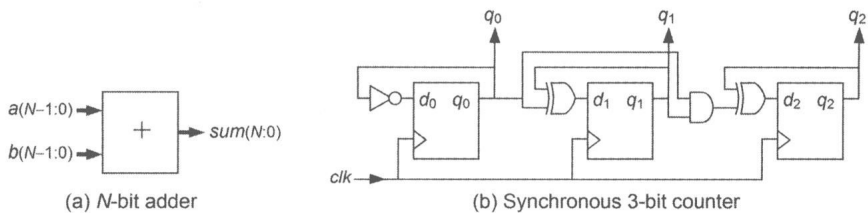


Figure 1.1

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