

# FSM-based Digital Design using Verilog HDL

# FSM-based Digital Design using Verilog HDL

**Peter Minns**  
**Ian Elliott**

*Northumbria University, UK*



John Wiley & Sons, Ltd

Copyright © 2008 John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,  
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): [cs-books@wiley.co.uk](mailto:cs-books@wiley.co.uk)  
Visit our Home Page on [www.wileyeurope.com](http://www.wileyeurope.com) or [www.wiley.com](http://www.wiley.com)

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1T 4LP, UK, without the permission in writing of the Publisher. Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or emailed to [permreq@wiley.co.uk](mailto:permreq@wiley.co.uk), or faxed to (+44) 1243 770620.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

#### *Other Wiley Editorial Offices*

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, ONT, L5R 4J3

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

#### *British Library Cataloguing in Publication Data*

A catalogue record for this book is available from the British Library

ISBN 978-0470-06070-4

Typeset in 10/12 pt Times by Thomson Digital, Noida, India

Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire

# Contents

<b>Preface</b>	<b>xi</b>
<b>Acknowledgements</b>	<b>xv</b>
<b>1 Introduction to Finite-State Machines and State Diagrams for the Design of Electronic Circuits and Systems</b>	<b>1</b>
1.1 Introduction	1
1.2 Learning Material	2
1.3 Summary	21
<b>2 Using State Diagrams to Control External Hardware Subsystems</b>	<b>23</b>
2.1 Introduction	23
2.2 Learning Material	23
2.3 Summary	38
<b>3 Synthesizing Hardware from a State Diagram</b>	<b>39</b>
3.1 Introduction to Finite-State Machine Synthesis	39
3.2 Learning Material	40
3.3 Summary	66
<b>4 Synchronous Finite-State Machine Designs</b>	<b>67</b>
4.1 Traditional State Diagram Synthesis Method	67
4.2 Dealing with Unused States	69
4.3 Development of a High/Low Alarm Indicator System	71
4.3.1 Testing the Finite-State Machine using a Test-Bench Module	75
4.4 Simple Waveform Generator	76
4.4.1 Sampling Frequency and Samples per Waveform	78
4.5 The Dice Game	79
4.5.1 Development of the Equations for the Dice Game	81

4.6	Binary Data Serial Transmitter	83
4.6.1	The RE Counter Block in the Shift Register of Figure 4.15	87
4.7	Development of a Serial Asynchronous Receiver	88
4.7.1	Finite-State Machine Equations	91
4.8	Adding Parity Detection to the Serial Receiver System	92
4.8.1	To Incorporate the Parity	92
4.8.2	$D$ -Type Equations for Figure 4.26	94
4.9	An Asynchronous Serial Transmitter System	95
4.9.1	Equations for the Asynchronous Serial Transmitter	98
4.10	Clocked Watchdog Timer	100
4.10.1	$D$ Flip-Flop Equations	102
4.10.2	Output Equation	102
4.11	Summary	103
<b>5</b>	<b>The One Hot Technique in Finite-State Machine Design</b>	<b>105</b>
5.1	The One Hot Technique	105
5.2	A Data Acquisition System	110
5.3	A Shared Memory System	114
5.4	Fast Waveform Synthesizer	116
5.4.1	Specification	117
5.4.2	A Possible Solution	118
5.4.3	Equations for the $d$ Inputs to $D$ Flip-Flops	119
5.4.4	Output Equations	120
5.5	Controlling the Finite-State Machine from a Microprocessor/Microcontroller	120
5.6	A Memory-Chip Tester	123
5.7	Comparing One Hot with the more Conventional Design Method of Chapter 4	126
5.8	A Dynamic Memory Access Controller	127
5.8.1	Flip-Flop Equations	131
5.8.2	Output Equations	131
5.9	How to Control the Dynamic Memory Access from a Microprocessor	132
5.10	Detecting Sequential Binary Sequences using a Finite-State Machine	134
5.11	Summary	143
<b>6</b>	<b>Introduction to Verilog HDL</b>	<b>145</b>
6.1	A Brief Background to Hardware Description Languages	145
6.2	Hardware Modelling with Verilog HDL: the Module	147
6.3	Modules within Modules: Creating Hierarchy	152
6.4	Verilog HDL Simulation: a Complete Example	155
	References	162
<b>7</b>	<b>Elements of Verilog HDL</b>	<b>163</b>
7.1	Built-In Primitives and Types	163

7.1.1	Verilog Types	163
7.1.2	Verilog Logic and Numeric Values	167
7.1.3	Specifying Values	169
7.1.4	Verilog HDL Primitive Gates	170
7.2	Operators and Expressions	172
7.3	Example Illustrating the Use of Verilog HDL Operators: Hamming Code Encoder	185
7.3.1	Simulating the Hamming Encoder	188
References		195
<b>8</b>	<b>Describing Combinational and Sequential Logic using Verilog HDL</b>	<b>197</b>
8.1	The Data-Flow Style of Description: Review of the Continuous Assignment	197
8.2	The Behavioural Style of Description: the Sequential Block	198
8.3	Assignments within Sequential Blocks: Blocking and Nonblocking	204
8.3.1	Sequential Statements	204
8.4	Describing Combinational Logic using a Sequential Block	209
8.5	Describing Sequential Logic using a Sequential Block	217
8.6	Describing Memories	229
8.7	Describing Finite-State Machines	240
8.7.1	Example 1: Chess Clock Controller Finite-State Machine	245
8.7.2	Example 2: Combination Lock Finite-State Machine with Automatic Lock Feature	252
References		265
<b>9</b>	<b>Asynchronous Finite-State Machines</b>	<b>267</b>
9.1	Introduction	267
9.2	Development of Event-Driven Logic	269
9.3	Using the Sequential Equation to Synthesize an Event Finite-State Machine	272
9.3.1	Short-cut Rule	275
9.4	Implementing the Design using Sum of Product as used in a Programmable Logic Device	276
9.4.1	Dropping the Present State $n$ and Next State $n + 1$ Notation	277
9.5	Development of an Event Version of the Single-Pulse Generator with Memory Finite-State Machine	277
9.6	Another Event Finite-State Machine Design from Specification through to Simulation	280
9.6.1	Important Note!	280
9.6.2	A Motor Controller with Fault Current Monitoring	281
9.7	The Hover Mower Finite-State Machine	285
9.7.1	The Specification and a Possible Solution	285
9.8	An Example with a Transition without any Input	289
9.9	Unusual Example: Responding to a Microprocessor-Addressed Location	291
9.10	An Example that uses a Mealy Output	293
9.10.1	Tank Water Level Control System with Solutions	293
9.11	An Example using a Relay Circuit	296

9.12 Race Conditions in an Event Finite-State Machine	299
9.12.1 Race between Primary Inputs	300
9.12.2 Race between Secondary State Variables	300
9.12.3 Race between Primary and Secondary Variables	300
9.13 Wait-State Generator for a Microprocessor System	301
9.14 Development of an Asynchronous Finite-State Machine for a Clothes Spinner System	304
9.15 Caution when using Two-Way Branches	309
9.16 Summary	312
References	312
<b>10 Introduction to Petri Nets</b>	<b>313</b>
10.1 Introduction to Simple Petri Nets	313
10.2 Simple Sequential Example using a Petri Net	318
10.3 Parallel Petri Nets	319
10.3.1 Another Example of a Parallel Petri Net	323
10.4 Synchronizing Flow in a Parallel Petri Net	324
10.4.1 Enabling and Disabling Arcs	325
10.5 Synchronization of Two Petri Nets using Enabling and Disabling Arcs	326
10.6 Control of a Shared Resource	327
10.7 A Serial Receiver of Binary Data	329
10.7.1 Equations for the First Petri Net	333
10.7.2 Output	333
10.7.3 Equations for the Main Petri Net	333
10.7.4 Outputs	333
10.7.5 The Shift Register	334
10.7.6 Equations for the Shift Register	334
10.7.7 The Divide-by-11 Counter	335
10.7.8 The Data Latch	335
10.8 Summary	336
References	336
<b>Appendix A: Logic Gates and Boolean Algebra Used in the Book</b>	<b>337</b>
A.1 Basic Gate Symbols Used in the Book with Boolean Equations	337
A.2 The Exclusive OR and Exclusive NOR	338
A.3 Laws of Boolean Algebra	338
A.3.1 Basic OR Rules	339
A.3.2 Basic AND Rules	339
A.3.3 Associative and Commutative Laws	340
A.3.4 Distributive Laws	340
A.3.5 Auxiliary Law for Static 1 Hazard Removal	341
A.3.5.1 Proof of Auxiliary Rule	341
A.3.6 Consensus Theorem	342
A.3.7 The Effect of Signal Delay in Logic Gates	343
A.3.8 De Morgan's Theorem	343

A.4	Examples of Applying the Laws of Boolean Algebra	345
A.4.1	Example: Converting AND–OR to NAND	345
A.4.2	Example: Converting AND–OR to NOR	345
A.4.3	Logical Adjacency Rule	345
A.5	Summary	346
<b>Appendix B: Counting and Shifting Circuit Techniques</b>		<b>347</b>
B.1	Basic Up and Down Synchronous Binary Counter Development	347
B.2	Example for a 4-Bit Synchronous Up-Counter Using <i>T</i> -Type Flip-Flops	349
B.3	Parallel-Loading Counters: Using <i>T</i> Flip-Flops	352
B.4	Using <i>D</i> Flip-Flops to Build Parallel-Loading Counters with Cheap Programmable Logic Devices	353
B.5	Simple Binary Up-Counter: with Parallel Inputs	354
B.6	Clock Circuit to Drive the Counter (And Finite-State Machines)	355
B.7	Counter Design using Don't Care States	355
B.8	Shift Registers	357
B.9	Asynchronous Receiver Details of Chapter 4	358
B.9.1	The 11-Bit Shift Registers for the Asynchronous Receiver Module	360
B.9.2	Divide-by-11 Counter	362
B.9.3	Complete Simulation of the Asynchronous Receiver Module of Chapter 4	364
B.10	Summary	365
<b>Appendix C: Tutorial on the Use of Verilog HDL to Simulate a Finite-State Machine Design</b>		<b>367</b>
C.1	Introduction	367
C.2	The Single Pulse with Memory Synchronous Finite-State Machine Design: Using Verilog HDL to Simulate	367
C.2.1	Specification	367
C.2.2	Block Diagram	367
C.2.3	State Diagram	368
C.2.4	Equations from the State Diagram	368
C.2.5	Translation into a Verilog Description	369
C.3	Test-Bench Module and its Purpose	372
C.4	Using SynaptiCAD's VeriLogger Extreme Simulator	376
C.5	Summary	378
<b>Appendix D: Implementing State Machines using Verilog Behavioural Mode</b>		<b>379</b>
D.1	Introduction	379
D.2	The Single-Pulse/Multiple-Pulse Generator with Memory Finite-State Machine Revisited	379
D.3	The Memory Tester Finite-State Machine in Section 5.6	383
D.4	Summary	386
<b>Index</b>		<b>387</b>

# Preface

This book covers the design and use of finite state-machines (FSMs) in digital systems. It includes stand-alone applications and systems that use microprocessors, microcontrollers, and memory controlled directly from the FSM, as well as other common situations found in practical digital systems. The emphasis is on obtaining a good understanding of FSMs, how they can be used, and where to use them.

The popular and widely used Verilog hardware description language (HDL) is introduced and applied to the description and verification of many of the designs in the book. In addition to logic gate and Boolean equation-level styles of Verilog description, there is also a chapter covering the use of HDL at the so-called behavioural level, whereby a design is described using the high-level features provided by Verilog HDL.

There is also a chapter using the One Hot technique, commonly used to implement FSMs in field programmable gate arrays with examples on the development of dynamic memory access (DMA) controllers and data sequence detectors. Asynchronous (event-driven) FSMs not requiring a clock signal are covered in a chapter using a technique that allows rapid development of reliable systems. A chapter on the use of Petri-net-based controllers is included, allowing parallel-based digital FSMs to be developed.

In the development of digital systems, microcontrollers have been used for many years to control digital inputs and outputs, as well as process analogue information. Now, using the techniques in this book, FSM-based designs can be implemented using a deterministic model, the state diagram, as a design aid. Once developed, the state diagram can be used to implement the final system using either Boolean equations obtained directly from the state diagram, or a behavioural Verilog HDL description, again developed directly from the state diagram. External devices, such as memory, address counters and comparators, can be implemented either from the Boolean equations that define their operation or via behavioural-level descriptions in Verilog HDL.

The book is targeted at undergraduate final-year students of Electrical, Electronic, and Communications Electronic Engineering, as well as postgraduate students and practising Electronic Design Engineers who want to know how to develop FSM-based systems quickly. The book will assume an understanding of basic logic design and Boolean algebra, as would be expected of a final-year undergraduate. The book sequence follows.

The first three chapters are in the form of a linear frame programmed learning format to help the reader learn the essential concepts of synchronous FSM design.

This set of notes has been used with undergraduate final-year students at our university for some years now and has been well received by the students. These chapters cover the idea of basic FSM design and synthesis. Once this is covered, the book reverts to a more familiar text. However, the first three chapters, being linear, can be read in the same style as the more familiar text if the reader desires.

A breakdown of the chapters in the book now follows.

Chapter 1 contains an introduction to FSMs, the Mealy and Moore models of an FSM, differences between synchronous (clock-driven) FSMs and asynchronous (event-driven) FSMs, the state diagram and how it can be used to indicate sequential behaviour and the inputs and outputs of a system. This follows with a number of examples of FSMs to illustrate the way in which they can be developed to meet particular specifications.

Chapter 2 covers the use of external hardware and how this hardware can be controlled by the FSM. The examples include how to create wait states using external timers, how to control analogue-to-digital converters, and memory devices. This opens up the possibilities of FSM-based systems that are not normally covered in other books.

Chapter 3 is a continuation of the programmed learning text, looking at synthesization of state diagrams using *T* flip-flops and *D* flip-flops, as well as initialization techniques.

The remaining chapters of the book will be in a more conventional format.

Chapter 4 covers synchronous (clock-driven) FSM examples, some with simulation. This chapter gives some practical examples commonly found in real applications, such as a digital waveform synthesizer and asynchronous serial transmit and receive blocks.

Chapter 5 is an introduction to the use of 'One Hotting' in synchronous FSM design. Amongst the examples covered is a DMA controller and serial bit stream code detection.

Chapter 6 is an introduction to Verilog HDL and how to use it at the gate level and the Boolean equation level, together with how to combine different modules to form a complete system.

Chapter 7 introduces the basic lexical elements of the Verilog HDL. Emphasis is placed on those aspects of the language that support the description of synthesizable combinational and sequential logic.

Chapter 8 takes a more detailed look at the Verilog HDL, with emphasis on behavioural modelling of FSM designs. It covers using an HDL to implement synchronous FSMs at the behavioural level - with examples.

Chapter 9 is an introduction to asynchronous (event-driven) design of FSMs from initial concepts through to the design of asynchronous FSMs to given specifications. This will also include a brief discussion of race problems with asynchronous designs and how to overcome them.

Chapter 10 is an introduction to synchronous Petri nets, and how they can be used to implement both sequential and parallel FSMs. Petri nets allow the design of parallel FSMs with synchronized control. This chapter shows how a Petri net can be designed and synthesized as an electronic circuit using *D*-type flip-flops.

Each chapter contains examples with solutions, many of which have been used by the authors in real practical systems.

There is a CD-ROM included with the book containing a digital simulation program to aid the reader in learning and verifying the many examples given in the book. The program

is based on Verilog HDL. This tool has been used to simulate most of the examples in the book.

Also on the CD-ROM are folders containing many of the book's examples, complete with test-bench descriptions to allow the simulations to be run directly on a PC-based computer.

**Peter Minns BSc(H) PhD CEng MIET**

**Ian Elliott BSc(H) MPhil CEng MIET**

*Newcastle Upon Tyne*

# Acknowledgements

We would like to thank all those who have helped in the proof reading of this book. In particular, we thank Safwat Mansi for his proof reading of our ideas and his helpful suggestions. In addition, Kathleen Minns is thanked for her help in the proof reading of the entire manuscript.

We would like to thank our editors, Emily Bone, Laura Bell, Kate Griffiths and Nicky Skinner for their help over the time that we have worked on this book. Thanks also to Caitlin Flint for her help with the marketing of the book.

Special thanks go to Donna Mitchell at SynaptiCAD for her help with Appendix C on the use of the VeriLogger Extreme Simulation Program on the CD-ROM with this book. Also, Gary Covington for his help in creating the CD-ROM.

Finally, thanks to our wives, Helen and Kathleen, for putting up with our frequent disappearances during the preparation of the book.

Any errors are, of course, entirely the responsibility of the authors.