
QUICK START GUIDE TO VERILOG

QUICK START GUIDE TO VERILOG

1ST EDITION

Brock J. LaMeres

 Springer

Brock J. LaMeres
Department of Electrical & Computer Engineering
Montana State University
Bozeman, MT, USA

ISBN 978-3-030-10551-8 ISBN 978-3-030-10552-5 (eBook)
<https://doi.org/10.1007/978-3-030-10552-5>

Library of Congress Control Number: 2018968403

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Cover credit: © MRMake | Dreamstime.com - Binary Code Photo

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The classical digital design approach (i.e., manual synthesis and minimization of logic) quickly becomes impractical as systems become more complex. This is the motivation for the modern digital design flow, which uses hardware description languages (HDL) and computer-aided synthesis/minimization to create the final circuitry. The purpose of this book is to provide a quick start guide to the Verilog language, which is one of the two most common languages used to describe logic in the modern digital design flow. This book is intended for anyone that has already learned the classical digital design approach and is ready to begin learning HDL-based design. This book is also suitable for practicing engineers that already know Verilog and need quick reference for syntax and examples of common circuits. This book assumes that the reader already understands digital logic (i.e., binary numbers, combinational and sequential logic design, finite state machines, memory, and binary arithmetic basics).

Since this book is designed to accommodate a designer that is new to Verilog, the language is presented in a manner that builds foundational knowledge first before moving into more complex topics. As such, Chaps. 1–6 provide a comprehensive explanation of the basic functionality in Verilog to model combinational and sequential logic. Chapters 7–11 focus on examples of common digital systems such as finite state machines, memory, arithmetic, and computers. For a reader that is using the book as a reference guide, it may be more practical to pull examples from Chaps. 7–11 as they use the full functionality of the language as it is assumed the reader has gained an understanding of it in Chaps. 1–6. For a Verilog novice, understanding the history and fundamentals of the language will help form a comprehensive understanding of the language; thus it is recommended that the early chapters are covered in the sequence they are written.

Bozeman, MT, USA

Brock J. LaMeres

Acknowledgments

For Kylie. Your humor brings me laughter and happiness every day. Thank you.

Contents

1: THE MODERN DIGITAL DESIGN FLOW	1
1.1 HISTORY OF HARDWARE DESCRIPTION LANGUAGES	1
1.2 HDL ABSTRACTION	4
1.3 THE MODERN DIGITAL DESIGN FLOW	8
2: VERILOG CONSTRUCTS	13
2.1 DATA TYPES	13
2.1.1 <i>Value Set</i>	14
2.1.2 <i>Net Data Types</i>	14
2.1.3 <i>Variable Data Types</i>	15
2.1.4 <i>Vectors</i>	15
2.1.5 <i>Arrays</i>	16
2.1.6 <i>Expressing Numbers Using Different Bases</i>	16
2.1.7 <i>Assigning Between Different Types</i>	17
2.2 VERILOG MODULE CONSTRUCTION	17
2.2.1 <i>The Module</i>	18
2.2.2 <i>Port Definitions</i>	18
2.2.3 <i>Signal Declarations</i>	19
2.2.4 <i>Parameter Declarations</i>	20
2.2.5 <i>Compiler Directives</i>	20
3: MODELING CONCURRENT FUNCTIONALITY IN VERILOG	23
3.1 VERILOG OPERATORS	23
3.1.1 <i>Assignment Operator</i>	23
3.1.2 <i>Continuous Assignment</i>	23
3.1.3 <i>Bitwise Logical Operators</i>	24
3.1.4 <i>Reduction Logic Operators</i>	25
3.1.5 <i>Boolean Logic Operators</i>	25
3.1.6 <i>Relational Operators</i>	25
3.1.7 <i>Conditional Operators</i>	26
3.1.8 <i>Concatenation Operator</i>	26
3.1.9 <i>Replication Operator</i>	27
3.1.10 <i>Numerical Operators</i>	27
3.1.11 <i>Operator Precedence</i>	28
3.2 CONTINUOUS ASSIGNMENT WITH LOGICAL OPERATORS	29
3.2.1 <i>Logical Operator Example: SOP Circuit</i>	29
3.2.2 <i>Logical Operator Example: One-Hot Decoder</i>	30
3.2.3 <i>Logical Operator Example: 7-Segment Display Decoder</i>	31
3.2.4 <i>Logical Operator Example: One-Hot Encoder</i>	34
3.2.5 <i>Logical Operator Example: Multiplexer</i>	36
3.2.6 <i>Logical Operator Example: Demultiplexer</i>	36

3.3	CONTINUOUS ASSIGNMENT WITH CONDITIONAL OPERATORS	37
3.3.1	<i>Conditional Operator Example: SOP Circuit</i>	38
3.3.2	<i>Conditional Operator Example: One-Hot Decoder</i>	39
3.3.3	<i>Conditional Operator Example: 7-Segment Display Decoder</i>	40
3.3.4	<i>Conditional Operator Example: One-Hot Decoder</i>	40
3.3.5	<i>Conditional Operator Example: Multiplexer</i>	41
3.3.6	<i>Conditional Operator Example: Demultiplexer</i>	42
3.4	CONTINUOUS ASSIGNMENT WITH DELAY	43
4:	STRUCTURAL DESIGN AND HIERARCHY	51
4.1	STRUCTURAL DESIGN CONSTRUCTS	51
4.1.1	<i>Lower-Level Module Instantiation</i>	51
4.1.2	<i>Port Mapping</i>	51
4.1.3	<i>Gate-Level Primitives</i>	53
4.1.4	<i>User-Defined Primitives</i>	54
4.1.5	<i>Adding Delay to Primitives</i>	55
4.2	STRUCTURAL DESIGN EXAMPLE: RIPPLE CARRY ADDER	56
4.2.1	<i>Half Adders</i>	56
4.2.2	<i>Full Adders</i>	56
4.2.3	<i>Ripple Carry Adder (RCA)</i>	58
4.2.4	<i>Structural Model of a Ripple Carry Adder in Verilog</i>	59
5:	MODELING SEQUENTIAL FUNCTIONALITY	65
5.1	PROCEDURAL ASSIGNMENT	65
5.1.1	<i>Procedural Blocks</i>	65
5.1.2	<i>Procedural Statements</i>	68
5.1.3	<i>Statement Groups</i>	73
5.1.4	<i>Local Variables</i>	73
5.2	CONDITIONAL PROGRAMMING CONSTRUCTS	74
5.2.1	<i>if-else Statements</i>	74
5.2.2	<i>case Statements</i>	75
5.2.3	<i>casez and casex Statements</i>	77
5.2.4	<i>forever Loops</i>	77
5.2.5	<i>while Loops</i>	77
5.2.6	<i>repeat Loops</i>	78
5.2.7	<i>for Loops</i>	78
5.2.8	<i>disable</i>	79
5.3	SYSTEM TASKS	80
5.3.1	<i>Text Output</i>	80
5.3.2	<i>File Input/Output</i>	81
5.3.3	<i>Simulation Control and Monitoring</i>	83
6:	TEST BENCHES	89
6.1	TEST BENCH OVERVIEW	89
6.1.1	<i>Generating Manual Stimulus</i>	89
6.1.2	<i>Printing Results to the Simulator Transcript</i>	91

6.2 USING LOOPS TO GENERATE STIMULUS	93
6.3 AUTOMATIC RESULT CHECKING	95
6.4 USING EXTERNAL FILES IN TEST BENCHES	96
7: MODELING SEQUENTIAL STORAGE AND REGISTERS	103
7.1 MODELING SCALAR STORAGE DEVICES	103
7.1.1 <i>D-Latch</i>	103
7.1.2 <i>D-Flip-Flop</i>	103
7.1.3 <i>D-Flip-Flop with Asynchronous Reset</i>	104
7.1.4 <i>D-Flip-Flop with Asynchronous Reset and Preset</i>	105
7.1.5 <i>D-Flip-Flop with Synchronous Enable</i>	106
7.2 MODELING REGISTERS	107
7.2.1 <i>Registers with Enables</i>	107
7.2.2 <i>Shift Registers</i>	108
7.2.3 <i>Registers as Agents on a Data Bus</i>	109
8: MODELING FINITE STATE MACHINES	113
8.1 THE FSM DESIGN PROCESS AND A PUSH-BUTTON WINDOW CONTROLLER EXAMPLE	113
8.1.1 <i>Modeling the States</i>	114
8.1.2 <i>The State Memory Block</i>	115
8.1.3 <i>The Next State Logic Block</i>	115
8.1.4 <i>The Output Logic Block</i>	116
8.1.5 <i>Changing the State Encoding Approach</i>	118
8.2 FSM DESIGN EXAMPLES	119
8.2.1 <i>Serial Bit Sequence Detector in Verilog</i>	119
8.2.2 <i>Vending Machine Controller in Verilog</i>	121
8.2.3 <i>2-Bit, Binary Up/Down Counter in Verilog</i>	123
9: MODELING COUNTERS	129
9.1 MODELING COUNTERS WITH A SINGLE PROCEDURAL BLOCK	129
9.1.1 <i>Counters in Verilog Using the Type reg</i>	129
9.1.2 <i>Counters with Range Checking</i>	130
9.2 COUNTER WITH ENABLES AND LOADS	131
9.2.1 <i>Modeling Counters with Enables</i>	131
9.2.2 <i>Modeling Counters with Loads</i>	131
10: MODELING MEMORY	135
10.1 MEMORY ARCHITECTURE AND TERMINOLOGY	135
10.1.1 <i>Memory Map Model</i>	135
10.1.2 <i>Volatile vs. Non-volatile Memory</i>	136
10.1.3 <i>Read-Only vs. Read/Write Memory</i>	136
10.1.4 <i>Random Access vs. Sequential Access</i>	136
10.2 MODELING READ-ONLY MEMORY	137
10.3 MODELING READ/WRITE MEMORY	139

11: COMPUTER SYSTEM DESIGN	143
11.1 COMPUTER HARDWARE	143
11.1.1 <i>Program Memory</i>	144
11.1.2 <i>Data Memory</i>	144
11.1.3 <i>Input/Output Ports</i>	144
11.1.4 <i>Central Processing Unit</i>	144
11.1.5 <i>A Memory Mapped System</i>	146
11.2 COMPUTER SOFTWARE	148
11.2.1 <i>Opcodes and Operands</i>	149
11.2.2 <i>Addressing Modes</i>	149
11.2.3 <i>Classes of Instructions</i>	150
11.3 COMPUTER IMPLEMENTATION: AN 8-BIT COMPUTER EXAMPLE	157
11.3.1 <i>Top-Level Block Diagram</i>	157
11.3.2 <i>Instruction Set Design</i>	158
11.3.3 <i>Memory System Implementation</i>	159
11.3.4 <i>CPU Implementation</i>	163
APPENDIX A: LIST OF WORKED EXAMPLES	187
INDEX	189