
QUICK START GUIDE TO VHDL

QUICK START GUIDE TO VHDL

1ST EDITION

Brock J. LaMeres

 Springer

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

ISBN 978-3-030-04515-9 ISBN 978-3-030-04516-6 (eBook)
<https://doi.org/10.1007/978-3-030-04516-6>

Library of Congress Control Number: 2018963722

© 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 illustration: © Carloscastilla | 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 VHDL 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 VHDL 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 VHDL, the language is presented in a manner that builds foundational knowledge first before moving into more complex topics. As such, Chaps. 1–5 only present functionality built into the VHDL standard package. Only after a comprehensive explanation of the most commonly used packages from the IEEE library is presented in Chap. 7, are examples presented that use data types from the widely adopted STD_LOGIC_1164 package. For a reader that is using the book as a reference guide, it may be more practical to pull examples from Chaps. 7–12 as they use the types *std_logic* and *std_logic_vector*. For a VHDL novice, understanding the history and fundamentals of the VHDL base release 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 Alexis. The world is a better place because you are in it.

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: VHDL CONSTRUCTS	13
2.1 DATA TYPES	13
2.1.1 <i>Enumerated Types</i>	13
2.1.2 <i>Range Types</i>	14
2.1.3 <i>Physical Types</i>	14
2.1.4 <i>Vector Types</i>	14
2.1.5 <i>User-Defined Enumerated Types</i>	15
2.1.6 <i>Array Type</i>	15
2.1.7 <i>Subtypes</i>	15
2.2 VHDL MODEL CONSTRUCTION	16
2.2.1 <i>Libraries and Packages</i>	16
2.2.2 <i>The Entity</i>	17
2.2.3 <i>The Architecture</i>	17
3: MODELING CONCURRENT FUNCTIONALITY	21
3.1 VHDL OPERATORS	21
3.1.1 <i>Assignment Operator</i>	21
3.1.2 <i>Logical Operators</i>	22
3.1.3 <i>Numerical Operators</i>	23
3.1.4 <i>Relational Operators</i>	23
3.1.5 <i>Shift Operators</i>	23
3.1.6 <i>Concatenation Operator</i>	24
3.2 CONCURRENT SIGNAL ASSIGNMENTS WITH LOGICAL OPERATORS	24
3.2.1 <i>Logical Operator Example: SOP Circuit</i>	25
3.2.2 <i>Logical Operator Example: One-Hot Decoder</i>	26
3.2.3 <i>Logical Operator Example: 7-Segment Display Decoder</i>	27
3.2.4 <i>Logical Operator Example: One-Hot Encoder</i>	29
3.2.5 <i>Logical Operator Example: Multiplexer</i>	31
3.2.6 <i>Logical Operator Example: Demultiplexer</i>	32
3.3 CONDITIONAL SIGNAL ASSIGNMENTS	34
3.3.1 <i>Conditional Signal Assignment Example: SOP Circuit</i>	34
3.3.2 <i>Conditional Signal Assignment Example: One-Hot Decoder</i>	35
3.3.3 <i>Conditional Signal Assignment Example: 7-Segment Display Decoder</i>	36
3.3.4 <i>Conditional Signal Assignment Example: One-Hot Encoder</i>	37
3.3.5 <i>Conditional Signal Assignment Example: Multiplexer</i>	38
3.3.6 <i>Conditional Signal Assignment Example: Demultiplexer</i>	39

3.4	SELECTED SIGNAL ASSIGNMENTS	41
3.4.1	<i>Selected Signal Assignment Example: SOP Circuit</i>	41
3.4.2	<i>Selected Signal Assignment Example: One-Hot Decoder</i>	42
3.4.3	<i>Selected Signal Assignment Example: 7-Segment Display Decoder</i>	43
3.4.4	<i>Selected Signal Assignment Example: One-Hot Encoder</i>	44
3.4.5	<i>Selected Signal Assignment Example: Multiplexer</i>	45
3.4.6	<i>Selected Signal Assignment Example: Demultiplexer</i>	46
3.5	DELAYED SIGNAL ASSIGNMENTS	48
3.5.1	<i>Inertial Delay</i>	48
3.5.2	<i>Transport Delay</i>	48
4:	STRUCTURAL DESIGN AND HIERARCHY	53
4.1	COMPONENTS	53
4.1.1	<i>Component Instantiation</i>	53
4.1.2	<i>Port Mapping</i>	53
4.2	STRUCTURAL DESIGN EXAMPLES: 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 VHDL</i>	59
5:	MODELING SEQUENTIAL FUNCTIONALITY	65
5.1	THE PROCESS	65
5.1.1	<i>Sensitivity Lists</i>	65
5.1.2	<i>Wait Statements</i>	66
5.1.3	<i>Sequential Signal Assignments</i>	67
5.1.4	<i>Variables</i>	68
5.2	CONDITIONAL PROGRAMMING CONSTRUCTS	70
5.2.1	<i>If/Then Statements</i>	70
5.2.2	<i>Case Statements</i>	71
5.2.3	<i>Infinite Loops</i>	73
5.2.4	<i>While Loops</i>	75
5.2.5	<i>For Loops</i>	75
5.3	SIGNAL ATTRIBUTES	76
6:	PACKAGES	81
6.1	STD_LOGIC_1164	81
6.1.1	<i>STD_LOGIC_1164 Resolution Function</i>	82
6.1.2	<i>STD_LOGIC_1164 Logical Operators</i>	83
6.1.3	<i>STD_LOGIC_1164 Edge Detection Functions</i>	83
6.1.4	<i>STD_LOGIC_1164 Type Conversion Functions</i>	84
6.2	NUMERIC_STD	85
6.2.1	<i>NUMERIC_STD Arithmetic Functions</i>	85
6.2.2	<i>NUMERIC_STD Logical Functions</i>	87
6.2.3	<i>NUMERIC_STD Comparison Functions</i>	87
6.2.4	<i>NUMERIC_STD Edge Detection Functions</i>	87

6.2.5	<i>NUMERIC_STD Conversion Functions</i>	88
6.2.6	<i>NUMERIC_STD Type Casting</i>	88
6.3	<i>TEXTIO AND STD_LOGIC_TEXTIO</i>	89
6.4	<i>OTHER COMMON PACKAGES</i>	92
6.4.1	<i>NUMERIC_STD_UNSIGNED</i>	92
6.4.2	<i>NUMERIC_BIT</i>	92
6.4.3	<i>NUMERIC_BIT_UNSIGNED</i>	93
6.4.4	<i>MATH_REAL</i>	93
6.4.5	<i>MATH_COMPLEX</i>	95
6.4.6	<i>Legacy Packages (STD_LOGIC_ARITH/UNSIGNED/SIGNED)</i>	95
7:	TEST BENCHES	99
7.1	<i>TEST BENCH OVERVIEW</i>	99
7.2	<i>GENERATING STIMULUS VECTORS USING FOR LOOPS</i>	101
7.3	<i>AUTOMATED CHECKING USING REPORT AND ASSERT STATEMENTS</i>	102
7.3.1	<i>Report Statement</i>	102
7.3.2	<i>Assert Statement</i>	103
7.4	<i>USING EXTERNAL I/O IN TEST BENCHES</i>	104
7.4.1	<i>Writing to an External File from a Test Bench</i>	104
7.4.2	<i>Writing to STD_OUTPUT from a Test Bench</i>	107
7.4.3	<i>Reading from an External File in a Test Bench</i>	109
7.4.4	<i>Reading Space-Delimited Data from an External File in a Test Bench</i>	111
8:	MODELING SEQUENTIAL STORAGE AND REGISTERS	117
8.1	<i>MODELING SCALAR STORAGE DEVICES</i>	117
8.1.1	<i>D-Latch</i>	117
8.1.2	<i>D-Flip-Flop</i>	118
8.1.3	<i>D-Flip-Flop with Asynchronous Resets</i>	118
8.1.4	<i>D-Flip-Flop with Asynchronous Reset and Preset</i>	119
8.1.5	<i>D-Flip-Flop with Synchronous Enable</i>	120
8.2	<i>MODELING REGISTERS</i>	121
8.2.1	<i>Registers with Enables</i>	121
8.2.2	<i>Shift Registers</i>	122
8.2.3	<i>Registers as Agents on a Data Bus</i>	123
9:	MODELING FINITE STATE MACHINES	127
9.1	<i>THE FSM DESIGN PROCESS AND A PUSH-BUTTON WINDOW CONTROLLER EXAMPLE</i>	127
9.1.1	<i>Modeling the States with User-Defined, Enumerated Data Types</i>	128
9.1.2	<i>The State Memory Process</i>	129
9.1.3	<i>The Next State Logic Process</i>	129
9.1.4	<i>The Output Logic Process</i>	130
9.1.5	<i>Explicitly Defining State Codes with Subtypes</i>	132
9.2	<i>FSM DESIGN EXAMPLES</i>	133
9.2.1	<i>Serial Bit Sequence Detector in VHDL</i>	133
9.2.2	<i>Vending Machine Controller in VHDL</i>	135
9.2.3	<i>2-Bit, Binary Up/Down Counter in VHDL</i>	137

10: MODELING COUNTERS	143
10.1 MODELING COUNTERS WITH A SINGLE PROCESS	143
10.1.1 Counters in VHDL Using the Type <i>UNSIGNED</i>	143
10.1.2 Counters in VHDL Using the Type <i>INTEGER</i>	144
10.1.3 Counters in VHDL Using the Type <i>STD_LOGIC_VECTOR</i>	145
10.2 COUNTERS WITH ENABLES AND LOADS	148
10.2.1 Modeling Counters with Enables	148
10.2.2 Modeling Counters with Loads	149
11: MODELING MEMORY	153
11.1 MEMORY ARCHITECTURE AND TERMINOLOGY	153
11.1.1 Memory Map Model	153
11.1.2 Volatile vs. Nonvolatile Memory	154
11.1.3 Read-Only vs. Read/Write Memory	154
11.1.4 Random Access vs. Sequential Access	154
11.2 MODELING READ-ONLY MEMORY	155
11.3 MODELING READ/WRITE MEMORY	158
12: COMPUTER SYSTEM DESIGN	163
12.1 COMPUTER HARDWARE	163
12.1.1 Program Memory	164
12.1.2 Data Memory	164
12.1.3 Input/Output Ports	164
12.1.4 Central Processing Unit	164
12.1.5 A Memory-Mapped System	166
12.2 COMPUTER SOFTWARE	168
12.2.1 Opcodes and Operands	169
12.2.2 Addressing Modes	169
12.2.3 Classes of Instructions	170
12.3 COMPUTER IMPLEMENTATION: AN 8-BIT COMPUTER EXAMPLE	177
12.3.1 Top-Level Block Diagram	177
12.3.2 Instruction Set Design	178
12.3.3 Memory System Implementation	179
12.3.4 CPU Implementation	184
APPENDIX A: LIST OF WORKED EXAMPLES	207
INDEX	211