VHDL ANSWERS TO FREQUENTLY
ASKED QUESTIONS
Second Edition

VHDL ANSWERS TO FREQUENTLY ASKED QUESTIONS Second Edition

by

Ben Cohen

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

ISBN 978-1-4613-7581-4 ISBN 978-1-4615-5641-1 (eBook)

DOI 10.1007/978-1-4615-5641-1

Additional material to this book can be downloaded from <http://extras.springer.com>.

(1) Reprinted from IEEE Std 1076-1993 IEEE Standard VHDL Language Reference Manual, Copyright© 1994 by the Institute of Electrical and Electronics Engineers, Inc.

The IEEE disclaims any responsibility or liability resulting from the placement (in hard copy or electronic format) and use in this publication of the following documents and packages:

- IEEE Std 1076-1993 IEEE Standard VHDL Language Reference Manual, Copyright© 1994 by the Institute of Electrical and Electronics Engineers, Inc.
- Package STANDARD and package TEXTIO from the IEEE Std 1076-1993 IEEE Standard VHDL Language Reference Manual.
- IEEE Std 1164-1993 IEEE Standard Multivalued Logic System for VHDL Model Interoperability (Std Logic 1164), Copyright© 1993 by the Institute of Electrical and Electronics Engineers, Inc.
- IEEE Draft Standard P1076.3, Version: 2.4, Dated 12 April 1995, Copyright© 1995 by the Institute of Electrical and Electronics Engineers, Inc.. This is an unapproved draft of a proposed IEEE standard, subject to change. Use of information contained in the unapproved draft is at your own risk.

Information is reprinted with the permission of the IEEE.

Copyright © 1998 Springer Science+Business Media New York

Originally published by Kluwer Academic Publishers, New York in 1998

Softcover reprint of the hardcover 2nd edition 1998

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, mechanical, photo-copying, recording, or otherwise, without the prior written permission of the publisher, Springer Science+Business Media, LLC.

Printed on acid-free paper.

I dedicate this book to my wife, Gloria Jean. Her creative and organizational contributions proved to be invaluable in the design of the cover, and the format and title of the book. Her support throughout this project is deeply appreciated.

CONTENTS

PREFACE	xv
ABOUT THE DISK	xix
NOTATION CONVENTIONS	
Symbols	xxv
Syntactic Description	xxvi
ACKNOWLEDGMENTS	xxvii
ABOUT THE AUTHOR	xxviii
1. LANGUAGE ELEMENTS	1
1.1 WHY VHDL FOR DIGITAL DESIGNS	4
1.2 SALIENT POINTS OF CONCURRENT STATEMENTS	6
1.2.1 Process	6
1.2.2 Concurrent Signal Assignment Statements	7
1.2.3 Component Instantiation	8
1.2.4 Concurrent Procedure	8
1.2.5 Generate	9
1.2.6 Concurrent Assertion	9
1.2.7 Block statement	10
1.3 GUARDED SIGNAL ASSIGNMENTS	11
1.4 SIGNALS AND PORTS	17
1.4.1 Disconnecting a Signal	17
1.4.2 Difference between <i>inout</i> and <i>buffer</i> ports	17
1.4.3 Port Association Rules – the Open	20
1.4.4 Type Conversion in Port Associations	22
1.5 CONFIGURATIONS	24
1.5.1 Configuration Requirements	24
1.5.2 Configuration Specification/Declaration	25
1.5.2.1 Configuration Specification	25
1.5.2.2 Configuration Declaration	28
1.5.2.3 Binding with configured components	31
1.5.2.4 Removing a Component from an Architecture through Configuration	31
1.5.3 Configuration Statement for a Specific Generic	32
1.5.4 CONFIGURATION OF GENERATE STATEMENTS	33
1.6 ARITHMETIC ISSUES AND OPERATORS	38
1.6.1 Defining 2'S Complement Value $-(2^{**31})$ in VHDL	38
1.6.2 Value Casting	39
1.6.3 REM/MOD Difference	40
1.6.4 Non-Associative Operators	41

1.6.5 Analysis Error with the "&" Operator	42
1.6.6 Specifying a Multi-Line String.	43
1.7 PACKAGE STD_LOGIC_1164	43
1.7.1 Differences Std_Logic_1164'1987 and '1993	43
1.7.2 Accessing Multiple Fields of Std_Logic_Vector	44
1.7.3 Aliases and Constants	44
1.7.4 Enumeration Type	46
1.7.4.1 Separate conversion functions.	46
1.7.4.2 Use of Integer Subtypes	47
1.7.4.3 Use of 'val attributes	47
1.7.5 Testbench for Demonstration Models	48
1.8 RANGE CONSTRAINT IN TYPE DEFINITION	50
1.9 SHARED VARIABLES	50
2. ARRAYS	55
2.1 ARRAY STRUCTURE REPRESENTATIONS	56
2.1.1 One Dimensional Arrays	56
2.1.2 Multi-Dimentional Arrays	56
2.2 ARRAYS -- LEGAL OPERATIONS	58
2.2.1 Overloaded Operators on Arrays	60
2.3 ARRAY SLICES AND RANGES	65
2.4 ARRAY INITIALIZATION	67
2.5 CONSTANT ARRAYS IN CASE	70
2.6 CONSTRAINED AND UNCONSTRAINED ARRAYS	71
2.6.1 Constraining Methods	72
2.6.2 Allowed Constrained and Unconstrained Objects	73
2.7 MAPPING ARRAYS OF DIFFERENT SIZES	74
2.8 UNCONSTRAINED AGGREGATE WITH "OTHERS "	75
2.9 ILLEGAL ARRAY TYPES	78
2.9.1 UNCONSTRAINED ARRAY OF AN UNCONSTRAINED ARRAY	78
3. DRIVERS	81
3.1 MULTIPLE DRIVERS - CASE 1	82
3.1.1 Std_Logic_Vector Solution	83
3.1.2 Separate Signals Solution	83
3.1.3 Atomicity	84

Table of Contents	ix
3.2 MULTIPLE DRIVERS - CASE 2	91
3.3 MULTIPLE DRIVERS ERROR – CASE 3	92
3.4 MULTIPLE DRIVERS ERROR – COMPONENT	93
3.5 CODING STYLE FOR DETECTION OF MULTIPLE DRIVERS	95
4. SUBPROGRAMS	99
4.1 SIDE EFFECTS FROM A PROCEDURE	100
4.1.1 Language View	100
4.1.2 STYLE VIEW	103
4.2 GARBAGE COLLECTION OF DYNAMICALLY CREATED OBJECTS	106
4.3 ACCEPTABLE TYPES IN PARAMETER LISTS FOR FUNCTION CALLS	110
4.4 FILES DECLARATIONS IN PROCEDURES	111
4.5 MULTIPLE ACCESSES OF SAME FILE	113
4.6 FILE ARRAY	114
4.7 CONVERSION FUNCTION FROM INTEGER TO TIME	116
4.8 NORMALIZATION IN SUBPROGRAMS	118
5. PACKAGES	121
5.1 CONVERTING TYPED OBJECTS TO STRINGS	122
5.2 PRINTING OBJECTS FROM VHDL	126
5.2.1 Writing to One file from Multiple Processes	129
5.3 MULTIPLE INPUT SIGNATURE REGISTER (MISR)	130
5.4 DESIGN OF A LINEAR FEEDBACK SHIFT REGISTER (LFSR)	132
5.5 RANDOM NUMBER GENERATION	137
5.6 DEFERRED CONSTANT IN PACKAGE DECLARATION	138
5.7 COMPLEX NUMBERS AND OVERLOADED OPERATORS	138
5.8 IEEE STANDARDS	143
6. MODELS	145

6.1 LARGE RAM MODEL FOR SIMULATION.	146
6.1.1 Traditional Memory Modeling	146
6.1.2 Efficiency Memory Modeling	149
6.1.3 Fixed Array Caching	149
6.1.4 Fixed Page Caching	152
6.1.5 Dynamic Page Caching	154
6.1.6 Disk paging with swap files	156
6.1.7 C Language Interface	157
6.1.8 RAM Testbench and Configurations	157
6.2 ZERO OHM RESISTOR (WIRE, BRIDGE) MODEL	159
6.2.1 Zero Ohm Resistor Model Restrictions	162
6.2.2 Architecture Variation Method for Resistor	163
6.3 ERROR INJECTOR MODEL	163
6.4 TRANSFER GATE (SWITCH)	178
7. SYNTHESIS	183
7.1 SUPPORTED/UNSUPPORTED CONSTRUCTS FOR SYNTHESIS	184
7.2 SYNTHESIS SENSITIVITY RULES	186
7.3 LATCH/REGISTER/COMBINATIONAL LOGIC	186
7.4 LATCH INFERRANCE IN FUNCTIONS	193
7.5 VARIABLE INITIALIZATION AND LIFETIME	193
7.5.1 Variable Initialization in Processes	193
7.5.2 Variable Initialization for Subprograms (Called from Processes)	195
7.6 WAIT STATEMENT	195
7.7 DEFINING SHIFT REGISTERS IN SYNTHESIS	197
7.7.1 Resolution -- Code for Shift Register	197
7.8 REGISTER FILE	199
7.8.1 Register File -- with signals	199
7.8.2 Register File -- with variables	200
7.9 MULTIPLEXER MODEL	202
7.10 DEMULTIPLEXER MODEL	203
7.11 BARREL SHIFTER	205
7.12 USE OF "DON'T CARE" IN CASE STATEMENT	207
7.12.1 Case Statement with No "Don't Care"	208
7.12.2 IF Statement	210

7.12.3 Loop Statement	210
7.12.4 IEEE Std_Synth	211
7.13 PARAMETERIZED PRIORITY ENCODER	212
7.13.1 Straight Encoding Priority Encoder	213
7.13.2 Two level encoding	215
7.14 GENERATING A SYNCHRONOUS PRECHARGE	220
7.15 TECHNOLOGY AND VHDL CODE DESIGN	223
7.15.1 Basic architecture of a logic cell.	223
7.15.2 Synthesis tool technology	226
7.15.3 Supporting component libraries	226
7.16 SYNTHESIZING TRI-STATES	226
7.16.1 Latches in Tri-State Controls	227
7.17 SUBELEMENT ASSOCIATION	230
7.17.1 Subelement Association for Ports	231
7.18 FINITE STATE MACHINE (FSM)	232
7.19 ONE-HOT ENCODING	233
7.20 INSTANTIATING SYNOPSYS® DESIGNWARE COMPONENTS	234
7.21 RESOURCE SHARING	235
7.22 APPLYING DIGITAL DESIGN EXPERIENCES	236
7.23 ADDRESS RANGE IDENTIFICATION VIA INFERRED COMPARATOR	238
7.24 PORT MAPPING TO GROUND OR VCC	239
7.25 BIT REVERSAL	240
7.26 HOW TO DESIGN A TIMER IN VHDL	241
7.27 SPECIFYING A MULTIPLIER	242
7.28 BEHAVIORAL SYNTHESIS	243
8. DESIGN VERIFICATION AND TESTBENCH	245
8.1 VERIFICATION PROCESSES	246
8.2 FUNCTIONAL VERIFICATION	247
8.3 REGRESSION TESTS	250
8.3.1 File Compare Method	250

8.3.2 Design Verifier Method	251
8.3.3 MISR Method	251
8.3.4 Formal Verification Method	252
8.4 FORMAL VERIFICATION	252
8.4.1 How Formal Verification Works	252
8.4.2 How Formal Verification Fits in the Design Cycle	253
8.4.3 Formal Verification Advantages	253
8.5 BUS FUNCTIONAL MODEL (BFM) MODELING	254
8.6 APPLICATION OF MISR, RANDOM, LFSR PACKAGES FOR AUTO-REGRESSION	257
8.7 STRENGTH STRIPPER	267
9. POTPOURRI	269
9.1 METHODS TO ENHANCE SIMULATION SPEED	270
9.1.1 Signals	270
9.1.1.1 Use Signals for Inter-Process Communication Only	271
9.1.1.2 Avoid Signal Reassignment for Same Value	271
9.1.1.3 Avoid Individual Assignment of Individual Elements of a Composite	271
9.1.1.4 Reduce Number of Signals	272
9.1.1.5 Avoid resolved signals	272
9.1.2 Concurrent Statements	273
9.1.2.1 Number of Processes/Concurrent Statements	273
9.1.2.2 Concurrent Signal Assignment, Sensitivity list or Wait Statements ^[6]	273
9.1.2.3 Processes with Multiple Clocks	275
9.1.2.4 Disabling Non-Necessary Concurrent Statements	275
9.1.2.5 Avoid Repeated Code or Function Calls	277
9.1.2.6 Avoid Unnecessary Processes	278
9.1.3 Types	278
9.1.4 Constants	279
9.1.5 Files	280
9.1.6 Subprograms	284
9.1.7 Memory	284
9.1.8 Packages	285
9.1.9 VITAL	285
9.2 ACCESSING SIGNALS INTERNAL TO COMPONENTS	285
9.2.1 Global Signal Method	285
9.2.2 PORT Method	289
9.3 TRANSFERRING A LINE ONTO A SIGNAL	291
9.4 TYPE DECLARATION IN MULTIPLE PACKAGES	293
9.5 INTERNET - FREQUENTLY ASKED QUESTIONS	294

9.6 VHDL TEXT EDITOR?	294
9.7 VITAL	295
9.7.1 VITAL Features	295
9.8 BEHAVIORAL MODELING	296
9.9 FINAL VHDL EXAM	297
9.9.1 Design Units	297
9.9.2 Compilation Order	298
9.9.3 VHDL Types	300
9.9.4 Attributes	302
9.9.5 “WAIT” Statement	302
9.9.6 Control Structure	303
9.9.7 Signals versus Variables	304
9.9.8 Operator Overloading	306
9.9.9 Model	306
9.9.10 Concurrent Statements	309
9.9.11 Drivers and Resolution Functions	309
9.9.12 Subprogram	311
10. DESIGN FOR REUSE	313
10.1 DESIGN PROCESSES FOR REUSABILITY	314
10.2 PARAMETERIZED, REUSABLE AND READABLE CODE	316
10.2.1 Notation	316
10.2.2 Path Definition	317
10.2.3 “Accordion” Parameterization	318
10.2.4 “Control” Parameterization	319
10.2.5 <i>Block</i> Statements for Partition Separations	323
10.2.6 <i>Generate</i> Statement for Multiple Instantiations of Components	328
10.2.7 Subprograms	328
10.2.8 Use of VHDL Attributes	328
10.2.9 Core Models	329
10.2.10 Bus Functional Model (BFM) and Testbench Architecture	330
10.3 DOCUMENTATION OF VHDL DESIGNS	331
10.3.1 Summary	331
10.3.2 Applicable Documents/References	331
10.3.3 Definitions	331
10.3.4 Design Team	331
10.3.5 Requirement Versus Design Matrix	332
10.3.6 Design Description	332
10.3.6.1 Interface Definitions	332
10.3.6.2 Block Diagrams and Registers	333
10.3.6.3 State Diagrams	333
10.3.6.4 Design Rationale	333
10.3.6.5 Operation Modes	333
10.3.6.6 Instruction Set Architecture (<i>ISA</i>)	334

10.3.6.7 Software Interface (Hardware/Firmware) Model	334
10.3.6.8 Design Restrictions and Ambiguities	334
10.3.6.9 Model Scalability	334
10.3.6.10 Design Testability	334
10.3.6.11 Hardware Initialization	335
10.3.6.12 Electrical Specification	335
10.3.6.13 Timing specifications	335
10.3.6.14 Design Files and compilation order	335
10.3.6.15 Application notes	335
10.3.7 Synthesis	336
10.3.7.1 Synthesis tools	336
10.3.7.2 Synthesis Library	336
10.3.7.3 Design Constraints	336
10.3.7.4 Compilation Scripts	336
10.3.7.5 Performance	336
10.3.8 Technology Mapping and Routing	336
10.3.9 Verification	337
10.3.9.1 Validation plan [9].	337
10.3.9.2 Testbench and Verifier.	338
10.3.10 Simulation	338
10.3.10.1 Platforms	338
10.3.10.2 Design Files and compilation order	339
10.3.10.3 Simulation Results	339
10.3.11 Backup Media	339
10.3.12 Recommendations	339
10.3.13 Index	339
APPENDIX A: VHDL'93 AND VHDL'87 SYNTAX SUMMARY	341
APPENDIX B: PACKAGE STANDARD	351
APPENDIX C: PACKAGE TEXTIO	353
APPENDIX D: PACKAGE STD_LOGIC_1164	355
APPENDIX E: PACKAGE STD_LOGIC_ARITH	359
APPENDIX F: VHDL PREDEFINED ATTRIBUTES	365
BIBLIOGRAPHY	369
INDEX	371

PREFACE

VHDL Answers to Frequently asked Questions is a follow-up to the author's book *VHDL Coding Styles and Methodologies* (ISBN 0-7923-9598-0). On completion of his first book, the author continued teaching VHDL and actively participated in the *comp.lang.vhdl* newsgroup. During his experiences, he was enlightened by the many interesting issues and questions relating to VHDL and synthesis. These pertained to: misinterpretations in the use of the language; methods for writing error free, and simulation efficient, code for testbench designs and for synthesis; and general principles and guidelines for design verification. As a result of this wealth of public knowledge contributed by a large VHDL community, the author decided to act as a facilitator of this information by collecting different classes of VHDL issues, and by elaborating on these topics through complete simulatable examples.

This book is intended for those who are seeking an enhanced proficiency in VHDL. Its target audience includes:

1. **Engineers.** The book addresses a set of problems commonly experienced by real users of VHDL. It provides practical explanations to the questions, and suggests practical solutions to the raised issues. It also includes packages of common utilities that are useful in the generation of debug code and testbench designs. These packages include conversions to strings (the IMAGE package), generation of Linear Feedback Shift Registers (LFSR), Multiple Input Shift Register (MISR), and random number generators. It also defines models of useful components including large, but memory-on-demand Random Access Memory (RAM), zero ohm resistor, and error injector. It includes synthesis guidelines and rules, and demonstrates these concepts through the modeling of several primitives. It provides a discussion and guidelines on design verification techniques, and describes methods to enhance simulation speed. A methodology for testbench designs is also defined. All VHDL code described in the book is on a companion 3.5" PC disk.
2. **College students.** This book enhances a student's comprehension of the language by addressing the user's concerns. It furnishes the students with useful examples that can be compiled and simulated.

This book differs from other VHDL books in the following respects:

1. Emphasizes real VHDL problems, rather than philosophical or introductory type of information.
2. Emphasizes application of VHDL for synthesis.
3. Uses complete examples to demonstrate problems and solutions.

4. Provides a disk that includes all the book examples and other useful reference VHDL material.
5. Uses easy to remember symbology notation to emphasize language rules, good and poor methodology and coding styles.
6. Identifies obsolete VHDL constructs that must be avoided.
7. Identifies synthesizable/non-synthesizable structures.
8. Uses a question and answer format to clarify and emphasize the concerns of VHDL users.

The book is organized in nine sections, starting with language related issues and progressing to real applications of VHDL. These sections include:

1. **LANGUAGE ELEMENTS.** Addressed are the issues related to the basic language elements of VHDL, including the salient points of concurrent statements, configurations, ports, arithmetic issues, and package Std_Logic_1164.
2. **ARRAYS.** Addressed are array operations, array initialization, use of constrained and unconstrained arrays, and mapping of arrays of different sizes. The application of arrays in synthesis is also addressed
3. **DRIVERS.** Errors in the creation of unexpected multiple drivers are examined.
4. **SUBPROGRAMS.** The issues of side effects, garbage collection, files, conversion functions, and normalization of parameters are discussed.
5. **PACKAGES.** Useful packages are examined. These include conversion of various types to binary, integer and hex strings; Multiple Input Signature Register (MISR); and Linear Feedback Shift Register (LFSR) definitions and applications; and random number generation and applications.
6. **MODELS.** Several classes of VHDL concepts are demonstrated through the definition of models. These models demonstrate various methods to implement a RAM, including a memory efficient model using a linked list and paging approach. It also includes a model of a zero ohm resistor and its applications. A model of an error injector and its applications is also provided.
7. **SYNTHESIS.** This section identifies the supported and unsupported constructs for synthesis. It also identifies a class of synthesis problems and solutions using a variety of primitive models. It identifies when synthesis interprets certain objects as registers, latches, or combinational logic. Synthesis of tri-states and design optimization is also discussed.
8. **DESIGN VERIFICATION AND TESTBENCH.** Various design verification approaches are examined, including: functional verification, auto-regression, and formal verification. Design method for testbench modeling is extensively addressed.
9. **POTPOURRI.** This section is a collection of topics that did not fit into a specific set of classification. It particularly emphasizes the methods to enhance simulation speed.

10. **DESIGN FOR REUSE.** This section discusses the disciplines in design processes, VHDL coding styles and documentation to achieve optimum design reuse. The disciplines in design processes address the issues of integrating pre-designed models and of entering into the reuse data base newly defined items. The VHDL Coding style discipline addresses the issues of partitioning, parameterization, and model design for readability, flexibility and reuse. The documentation discipline addresses the issues of good documentation to enable design reuse and better communication.

About The Disk

This disk contains all of the VHDL code presented in the chapters. Table 1 summarizes the contents of the disk.

Table 1. Contents of Enclosed Disk

Directory Name	File Name	Figure #	Description
root	disclaim.txt	- - -	Disclaimer
element	alias1.vhd	- - -	Aliases in an entity
element	alist_ea.vhd	1.5.4-1	Port association list
element	alistb.vhd	1.5.4-1	Examples of port association conversion rules, VHDL'87
element	alisttb2.vhd	1.5.4-2	Examples of port association conversion rules, VHDL'93
element	attribte.vhd	1.8.4.3	Use of 'val attribute to enhance code readability
element	block.vhd	1.3.7-1	Block example in synthesis
element	blockh.vhd	1.3.7-2	Block example with port header
element	buffer.vhd	1.5.2-4	Alternative use of buffer ports with <i>out</i> ports
element	buffer2.vhd	1.5.2-1	Component with buffer port
element	bufftop2.vhd	1.5.2-2	Instantiation of a component with <i>buffer</i> error
element	bufftop3.vhd	1.5.2-3	Instantiation of a component with <i>buffer</i> error
element	cfgdecl.vhd	1.6.2.2-1	Architecture and configuration declaration example
element	cfgspec.vhd	1.6.2.1	Configuration specification example
element	cntrl_c.vhd	1.8.5-2	Configuration declaration for controller model
element	cntrl_e.vhd	1.8.3-1	Aliases to enhance readability
element	cntrtb.vhd	1.8.5-1	Testbench for controller model
element	concat.vhd	1.7.5-2	Error in case expression
element	concat2.vhd	- - -	Same as concat.vhd, but error code commented out
element	confused.vhd	1.8.1	Differences in <i>Std_Logic_1164'87</i> and <i>Std_Logic_1164'93</i>
element	constant.vhd	1.8.3-2	Aliases and constants to enhance readability
element	convert.vhd	- - -	non-synthesizable version of <i>consynt.vhd</i> using aliasing
element	convn.vhd	1.7.2	Examples of VHDL type conversions
element	convsynt.vhd	1.8.4.1-1	Conversion function to enhance code readability
element	count_ea.vhd	1.7.3-2	<i>Mod</i> operator for a count-down counter
element	cproc.vhd	1.3.4	Concurrent procedure with side effect
element	deferd_c.vhd	1.6.2.4	Configuration to remove a component from an architecture
element	filter2.vhd	1.6.3	Setting generics with configuration declarations
element	genrat_c.vhd	1.6.4-1	Configuration of generate statement
element	hierarch.vhd	1.6.2.2-3	Binding of components of a hierarchical design with a configuration declaration
element	left2r.vhd	1.7.4	Non-associative operators – errors
element	open_ea.vhd	1.5.3	Port association examples
element	process.vhd	1.3.1	<i>Wait</i> statements in process
element	shared.vhd	1.10	Two processes exclusively accessing a shared resource
element	ssasgnt.vhd	1.3.2-1	Selected signal assignment
element	struct3.vhd	1.6.2.3	Binding with configured component
element	T4guard.vhd	1.4-4	Application of guards

Directory Name	File Name	Figure #	Description
element	Tguard.vhd	1.4-1	Application of guards -- testbench
element	topcfg.vhd	1.6.4-1	Configuration of a hierarchical design with <i>generate</i>
element	static.vhd	1.9	Subtype definition
element	wave.vhd	1.3.2-2	Waveform generation
arrays	anony_ea.vhd	2.9	Illegal anonymous array
arrays	array1.vhd	2.2	Implicit operators on arrays
arrays	ary.vhd	2.6	Unconstrained and constrained array restrictions
arrays	aryinit.vhd	2.4-1	Array initialization error
arrays	casecst.vhd	2.5-2	Case statement with locally static expression
arrays	casecst2.vhd	2.5-1	Case statement with error
arrays	generic.vhd	2.4-2	Illegal referencing of an identifier
arrays	multidm.vhd	2.7	Mapping of arrays of different sizes
arrays	signed.vhd	2.2.1-2	Overloaded "+" operator on signed and unsigned objects
arrays	test.vhd	2.9.1-1	Constraining array size based on size of other types, generics, and deferred constants
arrays	tmspec.vhd	2.1.2-2	Time parameters with two-dimensional arrays
arrays	uncons.vhd	2.8-1	Unconstrained aggregate with <i>others</i> -- errors
arrays	unconsok.vhd	2.8-2	Constrained aggregate with <i>others</i>
arrays	SynopOk.vhd	2.8-3	Assignment of a Slice with Synopsys (\
drivers	atmndrv5.vhd	3.1.3-2	Process sensitivity on resolved signal
drivers	atombit5.vhd	---	Process sensitivity on unresolved signal
drivers	atomic_pb.vhd	3.1.3-1	Package demonstrating resolved array type
drivers	atomic.vhd	---	Process sensitivity on resolved and unresolved signal
drivers	atomic2.vhd	---	Process sensitivity on resolved and unresolved signal
drivers	atomic3.vhd	---	Process sensitivity on resolved and unresolved signal
drivers	atomic5.vhd	3.1.3-4	Sensitivity on processes with resolved signal
drivers	atomstd5.vhd	---	Process sensitivity on Std_Logic_Vector
drivers	bug.vhd	3.3-1	Multiple drivers error
drivers	bugfix.vhd	3.3-2	fix of bug for file <i>bug.vhd</i>
drivers	drvstd5.vhd	---	Drivers and sensitivity on Std_Logic_Vector
drivers	drvbit.vhd	3.1	Problem with multiple drivers
drivers	drvstd.vhd	3.1.1	Problem with drivers
drivers	drvstd2.vhd	3.1.2	Separate signals resolution to avoid multiple drivers
drivers	recrd.vhd	3.2-1	Multiple drivers -- errors
drivers	recrdok.vhd	3.2-2	Multiple drivers resolution with static indices
drivers	test.vhd	3.4-1	Multiple drivers -- components
drivers	testok.vhd	3.3-4	Resolution of multiple drivers for file <i>test.vhd</i>
drivers	uresolve2.vhd	3.5-2	Levels of Complexity when using Unresolved signal and Arithmetic Operations
drivers	uresolve.vhd	3.5-1	Instantiating a Component with Unresolved Ports
subprgm	access.vhd	4.2-2	Aggravated case of a leaky subprogram
subprgm	accessok.vhd	4.2-3	Non-Leaky subprogram
subprgm	bitrfnct.vhd	4.8-1, -2	Normalized and Non-normalized variables in subprogram
subprgm	datain.txt	---	support file
subprgm	drvproc.vhd	4.1.2-2	Client/server approach to avoid side effects
subprgm	f_arry87.vhd	4.6-1	Indexed array model (VHDL'87)
subprgm	\wrlin87.vhd	4.2-4	Code to Test for a Memory Leak Condition
subprgm	wrifix87.vhd		Fix for readline memory leak
subprgm	fcopy87.vhd	4.4-1	Procedure to copy a file (VHDL'87)
subprgm	fcopy93.vhd	4.4-2	Procedure to copy a file (VHDL'93)

Directory Name	File Name	Figure #	Description
subprgm	file1.txt, file2.txt, file3.txt, fileout.txt	---	Files used for f_arry87.vhd file
subprgm	fline87.vhd	4.5	Multiple access of a file in VHDL'87
subprgm	image.vhd	4.2-1	Non-leaky dynamically linked object in a subprogram
subprgm	nrmaliases.vhd	4.8-3	Normalization using aliases
subprgm	proc87.vhd	4.3.1	Corrected subprogram for VHDL'87
subprgm	proc93.vhd	4.3.2	Corrected subprogram for VHDL'93
subprgm	sidefct.vhd	4.1.1-2	Architecture with side effects
subprgm	timecnvt.vhd	4.7	Conversion package for type time
package	arithstd.vhd	---	Sample application of Std_Logic_Arith package
package	complex.vhd		Application examample of complex package
package	complexp.vhd		Complex package
package	image_pb.vhd	5.1	Image package
package	lfsr4.vhd	5.4-4	Design of a four bit LFSR register
package	lfsr_tb.vhd	5.4-6	Testbench for he LFSR function
package	lfsrarth.vhd	---	LFSR package for <i>Std_Logic_Arith Signed and Unsigned</i>
package	lfsrbit.vhd	---	LFSR Bit
package	lfsrnbit.vhd	---	LFSR Numeric_Bit
package	lfsrnstd.vhd	---	LFSR package - Numeric_Std
package	lfsrstd.vhd	5.4-2	LFSR package
package	misr_pb.vhd	5.3	Multiple Input Signature Register (MISR) package
package	misr_tst.vhd	---	Test bench for MISR package
package	numstd.vhd	---	Operations using Numeric_Std (with error)
package	numstdok.vhd	---	Operations using Numeric_Std
package	rand_ea.vhd	---	application of integer random number package
package	randm.vhd	---	Random integer package
package	randm_ea.vhd	5.5-2	Application of integer random package
package	randm_pb.vhd	4.5-3	RND (random) package
package	rndm_int.vhd	5.5-1	Integer random package
package	strgimg.vhd	5.2-2	Use of 'image in VHDL'93
package	strngout.txt	---	simulation results
package	strngout.vhd	5.2-1	Strings to screen
package	uniform.vhd	---	Integer random number procedure
package	vs_util.vhd	---	Conversion routines
package	xyz.in	---	Initialization file
package	file2.vhd	5.2.1	Writing to One file from Multiple Processes
package	deferpb.vhd	5.6	Referencing Deferred Constant
package	complexp.vhd	5.7-1	Complex Package
models	ej_cfg.vhd	6.3-8	configuration declarations for error injector test
models	ej_cpu.vhd	6.3-5	Simple CPU BFM
models	ej_order.txt	---	Error injector compilation order
models	ej_reg.vhd	6.3-5	Register file BFM
models	ej_tb.vhd	6.3.7	Error injector testbench
models	erinj_tb.vhd	---	Error injector testbench
models	errinj.vhd	6.3-4	Error injector model
models	memory.vhd	6.1.1	Memory model, initialized at start-up
models	memry0.vhd	---	Simple memory model, no file preload
models	memrytb.vhd	---	memory test bench

Directory Name	File Name	Figure #	Description
models	ramfix.vhd	6.1.3	Fixed array caching RAM model
models	ramfix_c.vhd	6.1.8-2	Ramfix configuration
models	switch1.vhd	6.4-1	One Switch Model
models	\switch2.vhd	6.4-2	Two Switches in Series Model
models	ramlink.vhd	6.1.5-1	Dynamic page caching
models	ramlnk_c.vhd	6.1.8-2	Ramlink configuration
models	rampag_c.vhd	6.1.8-2	Rampage configuration
models	rampage.vhd	6.1.4	Fixed page caching
models	ram_tb.vhd	6.1.8-1	Memory testbench
models	setholdp.vhd	---	setup and hold package
models	switch1.vhd		
models	switch2.vhd		
models	z0quiet.vhd	6.2-3	Resistor model
models	zohm0_ea.vhd	6.2-2	Resistor model
models	zohm_ea.vhd	6.2-4	Resistor bank model
synths	agggreg.vhd	---	Application of aggregates
synths	areaop1.vhd	7.22-1	Model for optimization
synths	areaop2.vhd	7.22-3	Optimized model for area
synths	areaopt.vhd	---	Area optimization model
synths	array1.vhd	---	Enumerated data type index for synthesis
synths	asoc_bad.vhd	---	Subelement association
synths	asoc_err.vhd	7.17-1	Subelement association
synths	asoc_sy2.vhd	7.17-3	Legal VHDL'93 interface association
synths	asoc_syn.vhd	7.17-2	In-whole subelement association
synths	barrel.vhd	7.11-3	Synthesizable barrel shifter with case statement
synths	barrel2.vhd, barrel3.vhd, barrel4.vhd		
synths	barreln.vhd	7.11-1	Non-synthesizable barrel shifter
synths	barlsyn1	7.11-2	synthesizable barrel shifter
synths	bitrevp.vhd	7.25-1	Bit reversal process
synths	bitrevs.vhd	7.25-2	Bit reversal with a function
synths	casecst1.vhd	---	Case statement with constant
synths	compart1.vhd	7.23-1	Model of a comparator
synths	compart2.vhd	7.23-2	Address range identification without inference of a comparator
synths	count.vhd	7.3-2	One-dimensional array declaration and synthesis
synths	countvar.vhd	7.3-3	One-dimensional array declaration with variables
synths	d2ff_asn.vhd	7.3-6	Two sequential Flip-flop with asynchronous reset on one FF
synths	demuxp1.vhd	7.10.2	Demultiplexer Model with a Process
synths	demux1.vhd	---	Demultiplexer model
synths	demuxc.vhd	7.10-1	Demultiplexer model
synths	demuxp2.vhd	7.10-3	Demultiplexer model with a process
synths	dff.vhd	7.6-1	Flip-flop model
synths	dffasync.vhd	7.3-5	D flip-flop with asynchronous reset
synths	dffsync.vhd	7.3-4	D flip-flop with synchronous reset
synths	initvc1.vhd	7.5.1-1	Non-synthesizable VHDL code
synths	initvc2.vhd	7.5.1-1	Synthesizable VHDL code
synths	invert.vhd	---	Inverter model
synths	inverttb.vhd	---	Test bench for inverter
synths	ivtrol.vhd	7.17.1-1	Model of component with unconstrained interfaces
synths	ivtroltb.vhd	7.17.1-2	Subelement association of a component

Directory Name	File Name	Figure #	Description
synths	lead1.vhd	7.12.1-1	Using case statement
synths	lead2.vhd	7.12.1-2	Nested case statement
synths	lead3.vhd	7.12.1-3	Case statement with type "unsigned"
synths	lead4.vhd	7.12.2	Using If instead of case statement
synths	lead5.vhd	7.12.3-1	Using for..loop in a function
synths	lead6.vhd	7.12.3-2	Using for..loop in a process
synths	lead7.vhd	7.12.4	Application of Std_Match function
synths	lead8.vhd	---	Leading one
synths	mc14532b.vhd	7.13.1-2	Parameterized Priority Encoder
synths	mc14532q.vhd	7.13.1-2	Parameterized Priority Encoder
synths	multpy.vhd	7.27	Multiplier model
synths	mux.vhd	7.9	Multiplexer model
synths	prech1.vhd		
synths	prechrg.vhd	7.14-4	Precharge Model
synths	prechtb.vhd		
synths	priority.vhd	7.13.2-2	Two-Level Encoding Architecture for Priority Encoder
synths	ppty_tb.vhd	7.13.2-3	Priority Encoder Testbench
synths	reg9.vhd	7.7	Synthesis error
synths	regfile.vhd	7.8.1	Register file model
synths	regfilev.vhd	7.8-3	Register File with Variable
synths	regf_bad.vhd	7.8.2-1	Improper Register File with Variable
synths	regok.vhd	7.7.1-2	Corrected shift register model
synths	regok2.vhd	7.7.1-3	Improved shift register model
synths	sharing.vhd	7.21-2	Automatic resource sharing
synths	sharingf.vhd	7.21-3	Resource sharing by design
synths	timer_ea.vhd	7.26-1	Synthesizable timer design – integer method
synths	timr3_ea.vhd	7.26-2	Synthesizable timer design – Unsigned method
synths	togndvcc.vhd	7.24-1	Element association to Ground and Vcc
synths	triasync.vhd	---	Asynchronous tri-state
synths	trierr.vhd	7.16-1	Incorrect implementation of tri-state
synths	triok.vhd	7.16-2	Corrected model of tri-state with if
synths	triok2.vhd	7.16-3	Corrected model of tri-state with case
synths	trisync.vhd	7.16-4	Tri-state enable synchronous control
synths	trisync2.vhd	7.16-6	Method to avoid registering of tri-state control
verific	alu_ea.vhd	8.6-5	ALU model (for test)
verific	alu_tb.vhd	8.6-7	testbench
verific	proctcl.vhd	8.6-6	Protocol component
verific	types_p.vhd	8.6-3	Package for definition of a type
verific	uniform.vhd	8.6-4	Integer based random number package
potpouri	cntrcfg.vhd	9.9.9-3	Counter Configuration
potpouri	cntrtb.vhd	9.9.9-2	Counter Testbench
potpouri	cntr_ea.vhd	9.9.9-1	Counter architectures
potpouri	const.vhd	9.1.4	Use of constant for type conversion
potpouri	countdd.vhd	---	Vital 2.2 counter
potpouri	counter.vhd	---	Vital 2.2 counter
potpouri	crash.vhd	---	Model to test host memory capabilities
potpouri	crystal.vhd	9.1.2.2-2	Acceleratable process
potpouri	datain.txt	---	Text files for test
potpouri	dataout.txt	---	Text files for test
potpouri	dataout0.txt	---	Text files for test
potpouri	faq1.txt	---	Internet FAQ1 – general

Directory Name	File Name	Figure #	Description
potpouri	faq2.txt	---	Internet FAQ2 – books
potpouri	faq3.txt	---	Internet FAQ1 - Products and services
potpouri	faq4.txt	---	Internet FAQ1 - VHDL Glossary
potpouri	finta87.vhd	---	Write binary file, VHDL'87
potpouri	finta93.vhd	9.1.5-1	Write binary file, VHDL'93
potpouri	fintb87.vhd	---	Read binary file, VHDL'87
potpouri	fintb93.vhd	9.1.5-2	Read binary file, VHDL'93
potpouri	fline87.vhd	9.3-2	Transfer a line with a signal
potpouri	fline93.vhd	9.3-1	Transfer a line with shared variables VHDL'93
potpouri	global.vhd	---	Use of global signals
potpouri	hierglob.vhd	9.2.1-2	Global signals to access signals internal to component
potpouri	hierport.vhd	9.2.2-1	Use of ports to access signals internal to component
potpouri	rbool.vhd	9.9.11	Resolved Boolean Function and Example
potpouri	redund.vhd	9.1.2.5-1	Example of redundant code
potpouri	redund0.vhd	9.1.2.5-2	Redundant code avoidance
potpouri	redund2.vhd	---	Redundant code avoidance
potpouri	simout.txt	---	Text file for test
potpouri	speed.vhd	9.1.2.2-1	Sensitivity list versus wait statement
potpouri	subprgm.vhd	9.9.12	Subprogram Example
potpouri	types_p.vhd	9.9.3	Example of Type and Constant Declarations
potpouri	wait.vhd	9.1.2.2-2	acceleratable process with wait statement
reuse	aggr.vhd	10.2.4	Defining and Using Parameters for Design Parameterization
reuse	global.vhd	10.2.3	Limitations of Globally Static Parameters
reuse	param3.vhd	10.2.5-2	Use of Subtypes and constants
ieee	numbit.vhd	---	Numeric_Bit package DECLARATION
ieee	numstd.vhd	---	Numeric_Std package DECLARATION
ieee	stdlogic.vhd	---	Std_Logic_1164 package
std	standard.vhd	---	Package standard
std	textio.vhd	---	TextIO package
std	vhdl_87.txt	---	Syntax VHDL'87
std	vhdl_93.txt	---	Syntax VHDL'93
synopsys	attribut.vhd	---	ATTRIBUTES package
synopsys	bvarith.vhd	---	BV_ARITHMETIC package
synopsys	slmisc.vhd	---	std_logic_misc package
synopsys	std_sign.vhd	---	STD_LOGIC_SIGNED package
synopsys	std_unsg.vhd	---	STD_LOGIC_UNSIGNED package
synopsys	stdarith.vhd	---	STD_LOGIC_ARITH package
synopsys	stdxtio.vhd	---	STD_LOGIC_TEXTIO package
synopsys	synopsys.vhd	---	SYNOPSYS package
util	random_p.vhd	---	Random package
util	vs_util.vhd	---	Various utilities package
util	vsp.vhd	---	String operations package

NOTATION CONVENTIONS

The following symbols and syntactic description are used to facilitate the learning of VHDL.

SYMBOLS

Methodology and guideline.

Two thumbs up. Good methodology or approach.

Two thumbs down. Poor methodology or approach.

Disagreement in community on methodology or approach.

Legal or OK code

Coding Error

Synthesizable

Non-Synthesizable

...

Ellipsis points in code: Source code not relevant to discussion.

[1]

Quotations reprinted from IEEE Std 1076-1993 IEEE Standard VHDL Language Reference Manual (LRM). Quotations printed in *"italic and in this font"*.

Boldface

Boldface in text: Emphasizes important points.

Boldface in syntax and sample code: Emphasizes VHDL reserved words.

Italic font

Italic in text emphasizes VHDL word versus English text. For example, Objects of type *Signed* or *Unsigned* can be converted to type *Std_Logic_Vector* with the VHDL provided type conversion feature.

Italic

Italic wide font used in chapter 7 to identify material, extracted by permission, from the "1996 Synopsys Users Group Conference."

SYNTACTIC DESCRIPTION

left_hand_side ::= right_hand side

left_hand_side is the syntactic category

right_hand_side is a replacement rule

::= (read as "can be replaced by")

Vertical bar separates alternative items

Example: letter_or_digit ::= letter | digit

Square brackets [] enclose optional items

*Example: return_statement ::= **return** [expression]*

Braces {} enclose a repeated item (zero or more times).

Example: index_constraint ::= (discrete_range, {discrete_rang})

Underlined identifies that the notation is applicable for VHDL'93 ONLY

*Example: ... end **[configuration]** [configuration_simple_name]*

Acknowledgments

VHDL Answers to Frequently Asked Questions evolved from discussions on *comp.lang.vhdl* newsgroup, European Space Agency (ESA) reports, discussions with Model Technology Inc., Synopsys® presentation, and class discussions.

I especially thank the engineers, consultants, and educators of the *comp.lang.vhdl* public domain newsgroup who have addressed some very interesting questions and furnished challenging answers. These newsgroup contributors supplied a wealth of experiences and wisdom in the use of the language and its practical applications. Their conversations provided the stimulus that became the basis for this book.

I thank the Institute of Electrical and Electronics Engineers, Inc. for granting me permission to extract some material from the IEEE Std 1076-1993 Standard VHDL Language Reference Manual.

I thank Model Technology for the use of their friendly VHDL and Verilog PC and workstation simulation environment, for the loan of their VHDL/Verilog PC compiler/simulator, and for their excellent and competent product support group. Vince Corbin of Model Technology Inc. was particularly helpful in reviewing the book and giving excellent suggestions.

I thank Synopsys, Inc. for the release of their VHDL packages and for granting me permission to incorporate *VHDL Synthesis Techniques and Recommendations*, presented at the 1996 Synopsys Users Group Conference. Joseph Pick of Synopsys, Inc., author of that excellent tutorial presentation, reviewed this book and furnished many technical and editorial suggestions.

Sandi Habinc² from the European Space Agency published *VHDL Models for Board-Level Simulation* that was used as a seed in the discussions on verification and simulation speed. I thank Sandi for granting me permission to extract some valuable information from this document, and for supplying important comments.

I thank Dino Caporossi, *VFormal* Product Manager at COMPASS Design Automation, for contributing the article on formal verification.

I thank the following reviewers of this book who have provided valuable comments and useful suggestions in the first edition of this book: Paul Menchini of *Menchini & Associates*, Richard Hall of *Cadence Design Systems, Inc*³, Jake Karrfalt of *Alternative System Concepts, Inc.*,⁴ Greg Peterson, LT USAF under the direction of John Hines from USAF, and Johan Sandstrom and Stephen Schoessow, modeling and synthesis consultants.

I thank Carl Harris from Kluwer Academic Publishers for supporting me in this book endeavor. I acknowledge my daughter Lori Hillary, and my son Michael Lloyd for inspiring me to teach.

About the Author

Ben Cohen has an MSEE from USC and is a Scientist engineer at Hughes Aircraft Company. He has technical experience in digital and analog hardware design, computer architecture, ASIC design, and synthesis. He used hardware description languages for the modeling of systems at various modeling levels including, statistical, instruction set architecture, and synthesis. He applied VHDL, since 1990, to model various bus functional models of computer interfaces. He was also involved on several hardware, software, and firmware tasks for several microprocessor applications. He taught classes at USC and at Hughes on the application of microprocessors, Pascal, and VHDL. In 1995, he authored the book *VHDL Coding Styles and Methodologies*, ISBN 0-7923-9598-0 Kluwer Academic Publishers. In 1996, he authored the book *VHDL Answers to Frequently Asked Questions* (1st edition), ISBN 0-7923-9791-6 Kluwer Academic Publishers.

He recently studied and applied SES Workbench⁵, a statistical and transaction based system level simulator, for the analysis of systems when exposed to application workloads.

email: VhdlCohen@aol.com
ftp: users.aol.com/vhdlcohen/vhdl
Web: <http://members.aol.com/vhdlcohen/vhdl>

VHDL ANSWERS TO FREQUENTLY
ASKED QUESTIONS
Second Edition