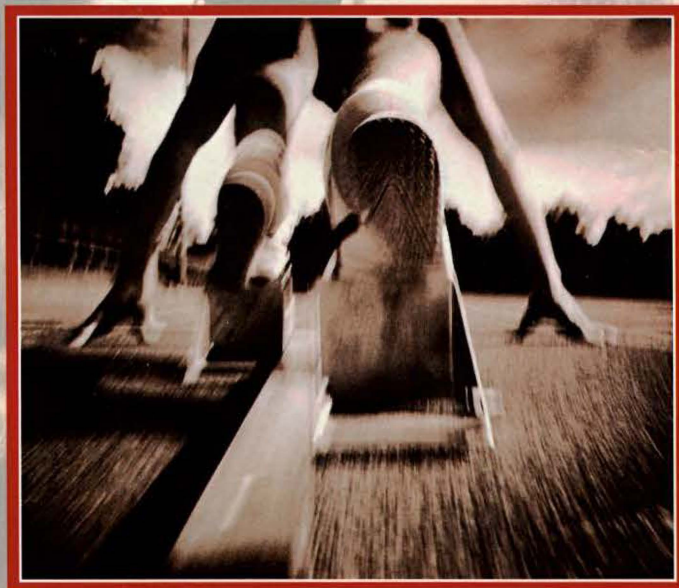


VHDL

Starter's Guide



Sudhakar Yalamanchili

JAMES
HURT

VHDL Starter's Guide

Sudhakar Yalamanchili

*School of Electrical and Computer Engineering
Georgia Institute of Technology*



PRENTICE HALL

Upper Saddle River, New Jersey 07458

Cadence Design Systems, Inc. - Ex. 1035, Page 00002

Library of Congress Cataloging-in-Publication Data

Yalamanchili, Sudhakar

VHDL starter's guide / Sudhakar Yalamanchili

p. cm.

Includes bibliographical references and index.

ISBN: 0-13-519802-X (pbk.)

1. VHDL (Hardware description language) 2. Very high speed integrated circuits—Design and construction—Data processing.

I. Title.

TK7885.7.Y35 1998

821.395—DC21

97-18139

CIP

Acquisitions editor: *TOM ROBBINS*

Editor-in-chief: *MARCIA HORTON*

Managing editor: *BAYANI MENDOZA DE LEON*

Director of production and manufacturing: *DAVID W. RICCARDI*

Production editor: *KATHARITA LAMOZA*

Cover design: *BRUCE KENSELAAR*

Manufacturing buyer: *DONNA SULLIVAN*

Editorial assistant: *NANCY GARCIA*



©1998 by Prentice-Hall, Inc.

Simon & Schuster / A Viacom Company

Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-519802-X

Prentice-Hall International (UK) Limited, London

Prentice-Hall of Australia Pty. Limited, Sydney

Prentice-Hall Canada Inc., Toronto

Prentice-Hall Hispanoamericana, S.A., Mexico

Prentice-Hall of India Private Limited, New Delhi

Prentice-Hall of Japan, Inc., Tokyo

Simon & Schuster Asia Pte. Ltd., Singapore

Editora Prentice-Hall do Brasil, Ltda., Rio de Janeiro

Contents

Preface	ix
1 Introduction	1
1.1 What is VHDL?	1
1.2 Digital System Design	2
1.3 The Marketplace	4
1.4 The Role of Hardware Description Languages	5
1.5 Chapter Summary	8
2 Simulation of Digital Systems	11
2.1 Describing Systems	12
2.2 Events, Propagation Delays, and Concurrency	14
2.3 Waveforms and Timing	17
2.4 Signal Values	18
2.5 Shared Signals	19
2.6 A Discrete Event Simulation Model	20
2.7 Chapter Summary	23

3	Basic Language Concepts	25
3.1	Signals	25
3.2	Entity–Architecture	26
3.3	Concurrent Statements	30
3.4	Constructing VHDL Models Using CSAs	42
3.5	Understanding Delays	47
3.6	Chapter Summary	54
4	Modeling Behavior	57
4.1	The Process Construct	57
4.2	Programming Constructs	61
4.3	More on Processes	63
4.4	The Wait Statement	68
4.5	Attributes	75
4.6	Generating Clocks and Periodic Waveforms	77
4.7	Using Signals in a Process	79
4.8	Modeling State Machines	82
4.9	Constructing VHDL Models Using Processes	85
4.10	Common Programming Errors	91
4.11	Chapter Summary	92
5	Modeling Structure	95
5.1	Describing Structure	95
5.2	Constructing Structural VHDL Models	100
5.3	Hierarchy, Abstraction, and Accuracy	105
5.4	Generics	109
5.5	Configurations	116
5.6	Common Programming Errors	123
5.7	Chapter Summary	123
6	Subprograms, Packages, and Libraries	127
6.1	Essentials of Functions	128
6.2	Essentials of Procedures	137
6.3	Subprogram and Operator Overloading	143
6.4	Essentials of Packages	144
6.5	Essentials of Libraries	147
6.6	Chapter Summary	149
7	Basic Input/Output	153
7.1	Basic Input/Output Operations	154
7.2	The Package TEXTIO	161

7.3	Testbenches in VHDL	162
7.4	ASSERT Statement	168
7.5	A Testbench Template	168
7.6	Chapter Summary	172
8	Programming Mechanics	173
8.1	Analyzing VHDL Programs	173
8.2	Order of Analysis	176
8.3	Elaboration of VHDL Programs	178
8.4	Initialization of VHDL Programs	179
8.5	Simulation of VHDL Programs	179
8.6	Chapter Summary	183
9	Identifiers, Data Types, and Operators	185
9.1	Identifiers	185
9.2	Data Objects	186
9.3	Data Types	187
9.4	Operators	191
9.5	Chapter Summary	193
	References	195
	Appendices	
A	Simulator Tutorial - Workview Office	197
B	Simulator Tutorial - Cypress WARP	211
B.1	Simulation vs. Synthesis	212
B.2	Using the WARP Tools	214
C	The SPIM Model: WVOffice	227
C.1	Model Overview	227
C.2	Sample Execution	229
C.3	Sample Execution in Workview Office	230
C.4	Loading a New Program into Memory	232
C.5	Extending the Datapath	232
C.6	Some Helpful Pointers	232
C.7	Datapath	233
C.8	VHDL Source for the SPIM Model	233

D	Standard VHDL Packages	251
D.1	Package STANDARD	251
D.2	Package TEXTIO	253
D.3	The Standard Logic Package	255
D.4	Other Useful Packages	259
E	A Starting Program Template	261
	Index	267

```
assert Q = check(1) and Qbar = check(0)
report "Test Vector Failed"
severity error;
```

FIGURE 7-8 An example of the use of the **assert** statement in a testbench

with a periodic signal. The module can apply the input vectors, wait for a period equal to the propagation delay of the longest path through the circuit, and then read the output signal values of the module under test.

Example End: Writing a Testbench

7.4 ASSERT Statement

In the prior example, the testbench model recorded errors or failures to pass a test vector by writing the test vector and a brief message to the file `outfile.txt`. Alternatively, we could use the **assert** statement. The **assert** statement is a general mechanism for detecting and reporting incorrect conditions during a simulation. We can use this statement for detecting failed test vectors, as shown in Figure 7-8. This statement would replace the **if** statement in the `srtester` module in Figure 7-6 that checks the output signal values and causes error messages to be written to a file. If we use the **assert** statement, rather than having an error message and the offending test vector being written to a file, the message regarding a violation and the string provided with the **report** clause would be sent to the simulation output. This output is generally the simulator console window unless you have redirected this to a file. The designer can report messages at one of several predefined severity levels: `NOTE`, `WARNING`, `ERROR`, and `FAILURE`. This provides a clean mechanism for the designer to classify the levels of information conveyed during simulation. For example, the `NOTE` category can be used to provide information about the progress of the simulation while a severity level of `ERROR` may cause the simulation to be aborted.

7.5 A Testbench Template

We are now ready to outline a few basic steps toward constructing a testbench for testing a VHDL model. It should be clear from the above examples that the testbench is a structural model with two components: a tester and the model under test. This is not the only way to structure a testbench, but is intuitive and simple to construct. Toward the end of the chapter we will describe an alternative structure for a testbench module.