
Component Design by Example

... A Step-by-Step Process Using VHDL with UART as Vehicle

Published by:

VhdlCohen Publishing

P.O. 2362

Palos Verdes Peninsula CA 90274-2362

vhdlcohen@aol.com

<http://www.vhdlcohen.com>

Library of Congress Cataloging-in-Publication Data

Library of Congress Card Number: 00-109498

Cohen, Ben

Component Design by Example

ISBN 0-9705394-0-1

Copyright © 2001 by VhdlCohen Publishing

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without the prior written permission from the author, except for the inclusion of brief quotations in a review.

Printed on acid-free paper

Printed in the United States of America

Contents

FOREWORD	IX
PREFACE	XI
ABOUT THE DISK	XV
ACKNOWLEDGEMENTS	XVII
ABOUT THE AUTHOR	XIX
DISCLAIMER	XX
1 OVERVIEW	1
1.1 COMPONENT DESIGN PROCESS.....	2
2 REQUIREMENT SPECIFICATION	7
2.1 LANGUAGE	8
2.2 UART REQUIREMENT SPECIFICATION	11
1.0 SCOPE.....	12
1.1 SCOPE.....	12
1.2 PURPOSE.....	12
1.3 CLASSIFICATION.....	12
2.0 DEFINITIONS.....	12
2.1 ASYNCHRONOUS TRANSMISSION.....	12
2.2 BAUD RATE	12
2.3 DTE.....	12
2.4 DCE.....	12
2.5 FRAMING ERROR.....	12
2.6 OVERRUN ERROR.....	13
2.7 PARITY.....	13
2.8 START BIT	13
2.9 STOP BIT	13
2.10 SYNCHRONOUS TRANSMISSION	13
2.11 UNDERRUN ERROR.....	13
2.12 WORD (WITH UART).....	13
3.0 APPLICABLE DOCUMENTS.....	14

3.1 GOVERNMENT DOCUMENTS	14
3.2 NON-GOVERNMENT DOCUMENTS	14
3.3 EXECUTABLE SPECIFICATIONS.....	14
4.0 ARCHITECTURAL OVERVIEW.....	14
4.1 INTRODUCTION	14
4.2 SYSTEM APPLICATION	15
5.0 PHYSICAL LAYER.....	17
5.1 INTERFACE PORT DESCRIPTION	17
6.0 PROTOCOL LAYER	23
7.0 ROBUSTNESS.....	24
7.1 ERROR DETECTION	24
8.0 HARDWARE AND SOFTWARE.....	24
8.1 FIXED PARAMETERIZATION	24
8.2 SOFTWARE INTERFACES.....	25
8.3 MODES OF OPERATION	30
9.0 PERFORMANCE.....	30
9.1 FREQUENCY.....	30
9.2 POWER DISSIPATION	30
9.3 ENVIRONMENTAL.....	30
9.4 TECHNOLOGY.....	30
10.0 TESTABILITY.....	30
11.0 MECHANICAL	30
3 ARCHITECTURAL PLAN.....	31
1.0 SCOPE	33
1.1 SCOPE	33
1.2 PURPOSE.....	33
1.3 CLASSIFICATION.....	33
2.0 DEFINITIONS.....	33
3.0 APPLICABLE DOCUMENTS	33
4.0 ARCHITECTURAL OVERVIEW.....	34
4.1 CPU SUBBLOCK	34
4.2 RECEIVER SUBBLOCK.....	34
4.3 TRANSMIT SUBBLOCK.....	35
4.4 CLOCK SUBBLOCK.....	35
5.0 PHYSICAL LAYER.....	36
6.0 PROTOCOL LAYER	37
7.0 ROBUSTNESS.....	37
8.0 HARDWARE AND SOFTWARE.....	37
8.1 FIXED PARAMETERIZATION	37
8.2 SOFTWARE INTERFACES.....	37
9.0 PERFORMANCE.....	37
10.0 TESTABILITY.....	37
11.0 DESIGN TOOLS	38

4	VERIFICATION PLAN.....	39
4.1	METHODOLOGIES	40
4.1.1	What is a Verification Plan.....	40
4.1.2	Why a Verification Plan.....	40
4.1.3	Verification Languages	42
4.2	VERIFICATION PLAN.....	46
1.	SCOPE.....	47
1.1	SCOPE	47
1.2	PURPOSE.....	47
1.3	CLASSIFICATION.....	47
2.0	DEFINITIONS	47
3.	APPLICABLE DOCUMENTS	48
3.1	GOVERNEMENT DOCUMENTS.....	48
3.2	NON-GOVERNEMENT DOCUMENTS	48
3.3	EXECUTABLE SPECIFICATIONS.....	48
3.4	REFERENCE SOURCES	48
4.	COMPLIANCE PLAN	49
4.1	FEATURE EXTRACTION AND TEST STRATEGY.....	49
4.2	TESTBENCH ARCHITECTURE.....	60
4.3	VERIFIER	69
5.	DESIGN TOOLS	72
5	DESIGN AND SYNTHESIS.....	73
5.1	RTL DESIGN	73
5.1.1	CPU Interface (CpuIf) Subblock Design	74
	<i>CPUIF.VHD</i>	83
5.1.2	Clock Control	90
	<i>CLKCNTRL.VHD</i>	92
5.1.3	Receiver Subblock (rcvsublk)	94
	<i>RCVSUBLK.VHD</i>	97
	<i>RECEIVER.VHD</i>	100
	<i>FIFO.VHD</i>	103
5.1.4	Transmit Subblock (xmitsublk)	106
5.1.5	UART Model.....	108
	<i>XMITSUBLK.VHD</i>	109
	<i>TRANSMITTER.VHD</i>	112
	<i>UART.VHD</i>	115
5.1.6	Compilation	120
5.1.7	Synthesis	120
5.1.8	Layout	124
5.1.9	Area Statistics	127

6	DESIGN VERIFICATION	129
6.1	OVERVIEW.....	130
6.2	PARSER PACKAGE	130
	<i>PARSER_PB.VHD</i>	133
6.3	CLIENT MODEL.....	142
	<i>UART_CLIENTRNDM.VHD</i>	145
	<i>RCV_CLIENT.VHD</i>	150
6.4	SERVER	152
	<i>UART_SERVER.VHD</i>	153
	<i>RCV_SERVER.VHD</i>	157
	<i>FIFO_SERVER.VHD</i>	160
	<i>FIFO_TB.VHD</i>	162
6.5	VERIFIER.....	165
6.5.1	ISSUES	165
6.5.2	Verifier Design Approach	167
6.5.3	Verifier Design.....	171
6.5.4	Top level Testbench.....	175
6.5.5	Configuration.....	175
	<i>VERIFPEEK.VHD</i>	176
	<i>UART8_TB.VHD</i>	192
	<i>UART_C.VHD</i>	198
6.5.6	Definition of Scenarios (test cases)	202
	<i>COMMAND FILE: INSTR1.TXT</i>	202
	<i>COMMAND FILE: CPU5TO15.TXT</i>	211
	<i>COMMAND FILE: SW_RESET.TXT</i>	211
	<i>COMMAND FILE: RCVINSTR.TXT</i>	214
	<i>COMMAND FILE: RCV11TO15.TXT</i>	216
6.5.7	Compilation Scripts	217
6.5.8	Simulation Results.....	218
6.5.9	Reading Text File into a Linked List	227
7	DOCUMENTATION AND DELIVERY.....	229
2.1	INTRODUCTION	230
2.2	REFERENCE INFORMATION	230
2.2.1	Documented References.....	230
2.2.2	Terminology	230
2.3	DELIVERABLE OVERVIEW	230
2.4	DATA ORGANIZATION FOR THE PACKAGING OF DELIVERABLES.....	233
2.5	DELIVERABLES DESCRIPTIONS.....	238
2.5.1	General Deliverables.....	238
2.5.2	Documentation Deliverables	241
2.5.3	Creation Guide.....	243

2.5.4	Logic Design Deliverables.....	244
2.5.5	Physical Design Deliverables.....	244
2.5.6	Design-for-Test and Manufacturing-Related Test Deliverables	244
2.5.7	Functional Verification Deliverables	245
2.5.8	Design Analysis Deliverables.....	247
2.6	DESIGN STATUS AND RECOMMENDATIONS	247
2.6.1	Status.....	247
2.6.2	Suggested Work	248
2.7	OPENMORE	248
8	INTEGRATION OF COMPONENTS INTO DESIGNS	259
8.1	APPLICATION OF UART INTO HIGHER LEVEL DESIGN.....	260
	UART_LEVEL2.VHD.....	261
8.2	HIGHER LEVEL COMPONENT EXTRACTION AND INTEGRATION	264
8.2.1	Motivation for change	264
8.2.2	Related Industry Trends	265
8.2.3	Types of IP Cores	267
8.2.4	Reuse Automation through High-Level Synthesis	268
8.2.5	IP-Centric Synthesis Methodology	269
8.2.6	Summary and Recommendation.....	270
9	REFLECTIONS.....	273
9.1	REQUIREMENTS.....	273
9.1.1	Realities	273
9.1.2	System implications	275
9.1.3	Consistency.....	276
9.2	DESIGN	277
9.3	VERIFICATION.....	278
9.3.1	Value of verifier.	279
9.3.2	Code coverage	279
9.3.3	Debugger/LINTing	280
9.3.4	When is design fully verified.....	280
9.3.5	Text Command Files	280
9.3.6	Review of testplan against verifier implementation	280
9.4	Summary and Conclusions.....	281
INDEX	283

FOREWORD

When Ben first asked me if I would be interested in reviewing his latest book, I was dually thrilled; once for the opportunity to contribute to the subject matter, second because it meant that Ben was taking on some new issues. In my many years with Synplicity®, I have had the opportunity to read or review many books. Of those, very few I appreciated enough to recommend. Two of Ben's earlier books, *VHDL Coding Styles and Methodologies* and *VHDL Answers to Frequently Asked Questions* are truly the best of the lot. They have long been on my technical recommendation list. Ben has an academic knowledge of the VHDL language, but utilizes that information with a practitioner's sense of reason. Both of these works are targeted toward the designer who utilizes VHDL. He fills these books with tips and recommendations, explanations as to why decisions are made and many references for further reading. What we gain from these books are a practical guide to applying VHDL with consideration for both the circuits to be implemented as well as the tools that you will use to create and verify the designs. I was anticipating that this new work would be similar in approach.

In *Component Design by Example*, Ben attacks the design reuse problem. This topic is timely and important. The Electronic Design Automation community has spent the most of the last decade foreshadowing the emergence and importance of design re-use and "IP" to obtain the next level of productivity gains. It is only in recently that we have seen more frequent occurrences of design reuse. In the past few years, our customers have begun to utilize various sizes and complexities of IP. With our customers, we have discussed, planned, pondered and solved various problems and futures for the development and reuse of design data and modular design flows. Consequently, we have observed that there is much design data that is reused, but only after significant effort. Often this is because the module was not successfully designed with reuse in mind. I suspect that many designers lacked resources and references broad enough to be useful on the topic of design for reuse.

Ben has created a pragmatic and useful book on design for reuse. It is useful because it brings lots of practical information and experience in one place. Useful because he dares to go beyond just the implementation phases of design, (which is more frequently addressed), and takes on the procedures from conception to specification and planning. PLEASE DON'T DISMISS THESE SECTIONS! Too many projects get into too much trouble down the line due to incomplete, ambiguous, or "undocumented" specifications and inadequate planning. It seems obvious, yet so many designers think of it as overhead that impedes progress.

Component Design by Example will be useful to any designer or design team. It may improve efficiency and improve products, or create disagreement in approach. My hope is it will stimulate discussion. I expect it will be the foundation for a future filled with IP. Read this before your next project. Then reread it afterward. You will benefit both times.

Andrew R. Dauman
Vice-President of Corporate Applications
Synplicity, Inc.
Sunnyvale, CA
September 28, 2000

PREFACE

As a VHDL trainer, consultant, and designer I recognized the need to demonstrate how to organize designs from conception to verified products. Many books address the design processes that include reuse methodologies for components, subblocks, and ASIC/FPGA designs. Since 1996, the *Virtual Socket Interface Alliance*™ (VSIA)¹ has played a key role in the design reuse scenario by creating standards for the industry that permits even broader reuse. These books and standards provide guidelines and general recommendations, but lack the provisions of complete design examples (with code) that demonstrate all the front-end phases of a design process. These phases include definition of requirements, architecture, verification approaches, HDL coding, synthesis, verification, and documentation. This book covers this gap and addresses the process of defining requirements and translating these requirements into a verified soft component design.

A component is taken in the sense of a design unit, subblock (or partition of a larger design), and a commercial Intellectual Property (IP). This book recognizes that there are many methodologies adopted by industry to perform front-end designs. This book provides methodologies generally accepted and recommended by many textbooks, including: *Reuse Methodology Manual*, Michael Keating and Pierre Bricaud, *Writing Testbenches*, *Functional verification of HDL Models*, Janick Bergeron, and *Verification Methodology Manual for Code Coverage in HDL Designs* by Michael Stuart and David Dempster. Users can tailor their methodologies to what is required given the constraints of labor force, budgets, and available tools. Even though tools do not represent methodologies, tools are often used to guide methodologies.

This book serves the following goals:

1. It demonstrates, by example, the **processes** involved in specifying, implementing, and verifying a reusable soft component. Most of these front-end processes are independent of the HDL implementation or verification languages. Even though the disciplines involved in every project and company will vary, the presented processes provide a good modeling base that users can modify and build upon. This is the focus and purpose of the book.
2. It demonstrates how to write a **requirement specification** that defines the foundation of the design. Defining a good specification is a difficult task as no single approach represents an exclusive best solution. It discusses potential

¹ *Virtual Socket Interface Alliance*™ <http://www.vsi.org>

specification methodologies, and provides a subset of this domain space as a modeling example to specify a serial interface, which is **UART-like in requirements**. These requirements meet those defined in the EIA (Electronics Industry Association) standard for serial data communication RS-232 UART. To emulate the challenges of real designs, such as parameterization and adaptability to different modes, this design adds additional performance requirements including: word widths, storage depths, and interrupt controller. This UART design is a soft component vehicle, and represents a model of moderate complexity with adaptability to many applications. A UART is a Universal Asynchronous Receiver Transmitter.

3. It demonstrates how to write an **architectural implementation** document, before writing any HDL code. This document defines the architectural approach for analysis and review.
4. It demonstrates how to write reusable and parameterized **VHDL synthesizable RTL code** for designs *using IEEE 1076.6 VHDL RTL for synthesis guidelines*². This parameterization emulates typical components that require such flexibilities.
5. It demonstrates how to write a **verification plan** that defines the foundation of the verification approaches of a design. This document is essential because it guides the design of the testbench and verification models, and provides a forum for scrutinizing the validity and completeness of the tests.
6. It demonstrates how to verify a design using **reusable testbenches in VHDL**.³ The issues of verification techniques are quite controversial,⁴ particularly with the advent of new verification tools and languages. The elementary concepts of verification are independent of tools or languages, even though tools and languages are used to implement the verification. This book concentrates on the strict use of VHDL as the verification language because it is opened and portable. It illustrates advanced VHDL modeling techniques for the generation of stimulus vectors, including the use of text command files, client/server models, and pseudo-random transactions. The text commands for the control of transactions include a rich, but small, instruction set capable to recursively call command files defined as subroutines.
7. It demonstrates the **design and synthesis process, including results of**

² See <http://www.vhdl.org/siwg>

³ Those techniques are referenced in the following books:
VHDL Coding Styles and Methodologies, 2nd Edition, Ben Cohen, KAP, 1999.
Writing Testbenches: Functional Verification of HDL Models, Janick Bergeron, KAP 2000

⁴ See <http://janick.bergeron.com/guild>

synthesis and post-route with Altera tools.

8. It provides, as a by-product of these methodologies, the design of a **high-speed, full-featured UART-like soft component** that can be tailored to applications that require an asynchronous or synchronous serial 8, 16, and 32-bit interface between two equipments.
9. It demonstrates the **integration of the soft component** into a subsystem.
10. It demonstrates the filling of the **OpenMore spreadsheet**. OpenMore is an assessment program developed by Synopsys and Mentor Graphics designed to enable a self-assessment of the reusability of commercial IP offerings.
11. It demonstrates an application of a **Virtual Component Block Deliverables** document, as described by *Motorola's Semiconductor Reuse Standard*⁵.

All VHDL code described in the book is on a companion CD. All code was verified and simulated with *ModelSim version 5.4b*,⁶ and synthesized with *Synplify version 5.3.1.7*. The CD also includes the **GNU toolsuite** with **EMACS** language sensitive editor (with VHDL, Verilog, and other language templates), and **TSHELL** tools that emulate a Unix shell.

This book is intended for:

1. **Engineers.** Book provides examples for the processes involved in defining components from requirements through verification and synthesis. It represents templates for the definition and implementation of a design. Engineers are better at copying and improving upon what is done, than from starting from scratch. This book will provide a head start in these processes.
2. **Application Designers.** Engineers who need a UART can use or modify the models described in this book.
3. **Tool Developers.** This book defines a well-specified and documented model of a common design of medium complexity, with hierarchy. Tool developers may exercise these requirements and HDL designs through new tools to demonstrate enhancements in the design processes.
4. **Trainers.** This book provides the focus of an advanced class for the definition and application of front-end methodologies and processes.

⁵ Motorola: <http://www.mot-sps.com/technology/srs/index.html>

⁶ Model Technology: <http://www.model.com>

⁷ Synplicity: <http://www.synplicity.com>

5. **College students.** Book demonstrates the design processes, from requirements to a verified implementation. It models real working industry design experiences

This book will be helpful as a guide through all the phases of a front-end design. It provides useful document templates for the definition of the requirement, implementation, and test plan documents. It also provides reusable code for the design of testbenches, and demonstrates by example, the application of this code for the synthesis and verification of a UART model.

About The Disk

Table 1 summarizes the contents of the enclosed CD.

Table 1 Contents of Enclosed CD

DIRECTORY NAME	DESCRIPTION
vhdl/rtl	clkcntrl.vhd -- clock controller subblock cpuif.vhd -- CPU Interface subblock fifo.vhd -- FIFO subblock rcvsublk.vhd -- Receiver top-level with receiver subblock and FIFO receiver.vhd -- Receiver subblock transmitter.vhd -- Transmitter subblock xmitsublk.vhd -- top-level with transmitter subblock and FIFO uart.vhd -- UART, top-level uart_level2.vhd -- integration of UART into higher level
vhdl/tb	vsp.vhd -- miscellaneous package lfsrstd.vhd -- Linear feedback shift register package image_pb.vhd -- image package for conversion to strings size_pkg.vhd -- For testbench use, global signals and constants parser_pb.vhd -- Parser package for file I/O and command parsing uart_server.vhd -- UART Server for TB rcv_client.vhd -- Client for receive side of UART rcv_server.vhd -- Server for receive side of UART verifierpeek.vhd -- Verifier with use of global signals for synch verifierblkbox.vhd -- Verifier, black box approach uart_clientrndm.vhd -- Uart client with good random tests uart_client_bad.vhd -- -- Uart client with tests that produce errors uart8_tb.vhd -- Top level testbench for UART uart_c.vhd -- Configuration declarations for UART testbench fifo_server.vhd -- Fifo server for use with uart client fifo_tb.vhd -- fifo testbench filedata.vhd -- Reading data from a file through linked lists
vhdl/gates	uart.vho -- gate level model produced by Altera

Directory Name	Description
uart	cpu5to15.txt -- CPU subroutine command file instr1.txt -- CPU command file rcv11to15.txt -- Receive side subroutine command file rcvinstr.txt -- Receive side command file sw_reset.txt -- Reset subroutine command file
scripts	compile.do -- ModelSim compile scripts compile.log -- ModelSim compile log run.do -- ModelSim run simulation
Altera	Synplify EDIF files, Files produced by Altera
simRuns	Gate_Sim -- gate level simulation run output RTL_BlkJBox -- RTL black box simulation run output RTL_GreyBoxRdmn -- RTL Gray box simulation run output RTL_Grey_OverrunError -- RTL Gray box with errors
IEEE	NUMBIT.VHD, NUMSTD.VHD, STDLOGIC.VHD packages
Synopsys	attribut.vhd, bvarith.vhd, slmisc.vhd, stdarith.vhd, stdxtio.vhd, std_cmpt.vhd, std_sign.vhd, std_unsg.vhd, synopsys.vhd packages
modelsim_ spy	PLI for ModelSim to access signals internal to a design
motorola_ Deliverable	srsmotdeliverable.pdf -- Motorola deleiverable document template
openMore	openmore-uart.xls -- OpenMore spreadsheet for UART openmore.xls -- OpenMore spreadsheet -- unfilled
VHDL_ Syntax	VHDL'87 and VHDL'93 syntax in HTML format VHDL Help: VHDL Language Reference Guide
Verilog	CummingsSNUG2000SJ_NBA_rev1a.pdf, VerilogHDLCoding_Motorola.pdf, verilog_vs_vhdl.PDF, vlog1364- HDLCON-2000.pdf
PDF_Files	ModelSI5_2 reference guide, VHDL and Verilog reference cards, Std_Logic_1164 reference card, and European Space Agency Modeling guidelines
Usr	GNU toolset
man	GNU help files in Windows Help format. Root file is <i>ManPagesDir</i>
Etc	Csh.cshrc and my.cshrc startup files for TSHEL

Acknowledgements

Component Design by Example evolved from the recent recognition in books, technical articles, and presentations on the need to follow a process for large designs. In particular, I thank Michael Keating, author of *Reuse Methodology Manual*, and, Janick Bergeron, author of *Writing Testbenches, Functional verification of HDL Models* and organizer of the *Verification Guild* newsletter for bringing forward those important design issues.

I thank Model Technology for granting me a license of *ModelSim version 5.4b* with the built-in code coverage for the duration of the project. *ModelSim* is an excellent user-friendly HDL toolset that enabled the compilation and verification of this design.

I thank Synplicity for granting a license of *Synplify HDL Analyst version 5.3.1* to synthesize the design and to extract the RTL views and delay paths for visual display and documentation of the logic and critical paths. *Synplify* is a very efficient, user-friendly, and insightful linting FPGA synthesis tool.

Altera's *MAX+PLUS® II ver 9.4* complemented *Synplify's* EDF output because it routed the design and produced an accurate gate level model with routed timing. Altera was kind enough to grant me a tool license for this project.

I thank Novas for granting me a license of *Debussy 5.0* Total Debug (tm) system for complex designs at the gate, RTL and behavioral levels. Even though the license came in late in the project, I was still able to gain insightful views of the design and testbench.

I thank Reto Zimmermann from Synopsys for commenting on the manuscript, and for supporting the community on the excellent upgrades to *vhdl-mode* for *emacs* GNU text editor. The application of the language sensitive *vhdl-mode* significantly helped in the production of VHDL code for design and verification.

I thank YxI for providing me with more insights into advanced synthesis methodologies from higher-level HDL definitions.

I sincerely thank Andrew Dauman from Synplicity and Richard Hall from Cadence Design Systems, Inc for reviewing the book and providing many suggestions.

I especially thank my wife, Gloria Jean, for supporting me in this endeavor.



**Sculpture Created by my Wife Gloria to
Express my Long Hours with a Laptop in the Creation of VHDL Books**

About the Author

Ben Cohen is currently a VHDL language trainer and consultant. He has technical experience in digital and analog hardware design, computer architecture, ASIC design, synthesis, and use of hardware description languages for modeling of statistical simulations, instruction set descriptions, and hardware models. He applied VHDL since 1990 to model various bus functional models of computer interfaces. He authored *VHDL Coding Styles and Methodologies*, first and second editions, and *VHDL Answers to Frequently Asked Questions*, first and second editions. He was one of the pilot team members of the VHDL Synthesis Interoperability Working Group of the Design Automation Standards Committee who authored the *IEEE P1076.6 Standard for VHDL Register Transfer Level Synthesis*. He taught several VHDL training classes, and provided VHDL consulting services on several tasks.

VhdlCohen Training and Consulting
email: VhdlCohen@aol.com
Web page: <http://www.vhdlcohen.com/>
