

Srivatsa Vasudevan

Practical UVM

Step by Step Examples

© Srivatsa Vasudevan 2011-2016.

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed “Attention: Book Permissions,” at the address below or via email to Srivatsa (at) uvmbook (dot) com.

The programs and code in this book have been included for their instructional value. While care was taken during the preparation of this book, no warranty or fitness is implied. The author assumes no liability for errors, omissions, or for damages resulting from the use of the information contained herein. All references to known as trademarks or service marks have been appropriately indicated.

All code within this text was compiled and simulated with Synopsys® VCS J-2015.09 using Accellera UVM versions 1.1d and 1.2. Any information on the proposed IEEE 1800.2 standard is current at of the time of the book publishing.

First Edition. July 2016.

ISBN : 9780997789607

The publisher offers discounts when ordered in quantity. Special discounts are available on quantity purchases by corporations, associations, and others. For more information, please contact:

Srivatsa Vasudevan
5567 Kimberly Street,
San Jose CA 95129
United States of America.

Printed in the United States of America Summer 2016.

Cover Design: Fourth Dimension Inc.

*This book is dedicated to:
The ONE by whose Grace,
A mute can speak eloquently,
A lame person can climb a mountain,
And for whom, nothing is impossible.
Our families and friends for supporting us
on this incredible journey.*

Contents

Part I UVM Building Blocks - A Tutorial

1	UVM Library Introduction	3
1.1	A Basic Verilog Testbench	3
1.2	A Simple SystemVerilog Testbench	4
1.3	How Does UVM Improve a Basic SystemVerilog Testbench?	6
1.4	Some UVM Terminology to Begin our UVM Journey	8
2	UVM Core Utilities	9
2.1	Essential Core Utilities	11
2.1.1	Create	11
2.1.2	Copy	12
2.1.3	Clone	13
2.2	Common Operations	14
2.2.1	Compare	14
2.2.2	Print	17
2.2.3	Packing And Unpacking	18
2.3	Core Operations Summary	23
2.4	UVM Macros to Simplify Things	23
2.5	A Complete Class in UVM	26
2.6	Coding Guidelines	29
2.7	Before Proceeding Further.	29
3	UVM Factory	31
3.1	Need for a Factory Pattern	31
3.2	UVM Factory Operation	32
3.3	Factory Behaviour for Parameterized and NonParameterized Classes	35
3.4	Using the Factory	36
3.5	Replacing Components Using a Factory.	36
3.5.1	Type Overrides	37
3.5.2	Instance Specific Overrides.	38
3.5.3	Rules for Processing Overrides	40
3.6	Debugging	40
3.7	Coding Guidelines	41
4	Transaction Layer Communication	43
4.1	Basics of Transaction Level Communication	44
4.1.1	Put Port	45
4.1.2	Get Ports	46
4.1.3	Understanding Blocking and Non Blocking Methods.	47

4.1.4	Peek	48
4.1.5	Analysis Ports	48
4.2	Connecting Transaction-Level Components	49
4.3	UVM TLM - Ports Summary	51
4.4	UVM TLM 2	52
4.4.1	Blocking Transports	53
4.4.2	NonBlocking Transports	55
4.5	Sockets	58
4.6	Time	60
4.7	Connecting TLM2 Ports and Sockets	60
4.8	Generic Payload	61
4.9	Extensions	63
5	Reporting Infrastructure	65
5.1	Elements of a UVM Report	66
5.1.1	Severity Specification	67
5.1.2	Verbosity Settings	67
5.1.3	Handling Specification	68
5.1.4	API Presented By UVM Components to Generate Messages	69
5.2	Reporting Subsystem - Practical Applications	70
5.2.1	Controlling Reporting from the Command Line	71
5.2.2	Controlling Verbosity	71
5.2.3	Logging Messages to a File	72
5.2.4	Demoting Reports From One Level to Another	72
5.2.5	Catching Reports and Changing Them	73
5.2.6	Using Reporting Infrastructure with Assertions and Modules	73
6	Phasing	75
6.1	Common Domain Pre Run Stages	75
6.1.1	Build_phase	75
6.1.2	connect_phase	77
6.1.3	end_of_elaboration_phase	77
6.1.4	start_of_simulation_phase	77
6.2	Run Time Phases	78
6.3	Clean Up Stages	80
6.4	Preventing Phases from Ending - Objections	81
6.5	Phase States	85
6.6	Phase Callbacks	86
6.7	Spanning Multiple Phases	89
6.8	Adding a Domain to a Schedule	93
7	UVM Command Line Processor	95
7.1	Command Line Processor Basics	95
7.2	Built in Arguments	96
7.3	Available Global Settings	96
7.4	Controlling Simulation Behavior	97
7.4.1	Verbosity	97
7.4.2	Severity	97
7.4.3	Configuration	98
7.4.4	Starting Sequences From a Command Line	98
7.4.5	Factory Command Line Interface	99
7.5	Debug Switches	99
7.6	Custom Command Line Processor Examples	100

8	Configuration and Resource Databases	103
8.1	Resource Database Infrastructure	103
8.2	Config Database Infrastructure	105
8.3	Priority for Setting Values in the Database	106
8.4	Using Regular Expressions in UVM	107
8.5	Debugging	108
8.6	Performance Tuning and Coding Guidelines	108
9	UVM Component Hierarchy	111
9.1	Overview of Capabilities Provided by UVM Component	112
9.1.1	Construction and Hierarchy Management	112
9.1.2	Component Configuration	113
9.1.3	Factory Interface	113
9.1.4	Reporting Interface	113
9.2	Macros Provided for UVM Components	114
9.3	UVM Library Component Classes	114
9.3.1	UVM Driver	115
9.3.1.1	Providing Driver Extensibility	119
9.3.1.2	Different Driver modes	121
9.3.1.3	Other Driver Variants	123
9.3.2	Monitor	123
9.3.3	Sequencer	126
9.3.4	Agent	128
9.3.4.1	Agent Configuration	128
9.3.4.2	Agent Build Phase	130
9.3.4.3	Agent Connect Phase	130
9.3.5	Subscriber	131
9.3.6	Scoreboard	132
9.3.7	Environment Class	135
9.3.8	Top Level Test	139
9.4	Component Configuration	141
9.5	Creating and Using UVM Components - Guidelines	143
10	Callbacks	145
10.1	Creating Callbacks in Components	145
10.1.1	Step 1: Create Virtual Callback Class	146
10.1.2	Step 2: Register the Callback Class	148
10.1.3	Step 3: Place the Callback Class Hook in Driver Code	148
10.1.4	Step 4: Extend and Use the Callback Class	149
10.2	User Applications of Callbacks	151
10.2.1	Typewide Callbacks	151
10.2.2	Instance Specific Callbacks	152
10.3	Order of Execution of Callbacks	152
10.4	Coding Guidelines	152
10.5	Exercises for Further Exploration	153
11	UVM Register Abstraction Layer	155
11.1	Typical UVM Register Use Model	155
11.2	Components of a Register Model	156
11.3	Register Maps	158
11.4	Register Model Architecture	159
11.5	Register Model API	159
11.5.1	Register Model Creation API	160
11.5.2	Register Model Hierarchy Traversal API	160

11.5.3	Register Model Access API	162
11.5.3.1	Reset	162
11.5.3.2	Read/Write	163
11.5.3.3	Get/Set	164
11.5.3.4	Peek/Poke	164
11.5.3.5	Randomize	166
11.5.3.6	Update	166
11.5.3.7	Mirror	167
11.5.4	Front Door Access	167
11.5.5	Back Door Access	168
11.6	Difference Between Backdoor and Peek/Poke Operations	169
11.7	Coverage of Register Models	169
11.8	Customization of Register Models	170

Part II Using UVM Building Blocks

12	Creating a Complete UVM Environment	173
12.1	Verification Environment	174
12.2	Top level Integration	175
12.2.1	Step 1: Create the Testbench Top Level Module	175
12.2.2	Step 2: Clocks and Reset	175
12.2.3	Step 3: Connect the Interfaces	176
12.2.4	Step 4: Start UVM Test	176
12.2.5	The Complete Top Level Module	177
12.3	The UVM Class Environment	178
12.3.1	Master Agent	179
12.3.2	Slave Agent	179
12.3.3	Scoreboard	179
12.3.4	Coverage	181
12.3.5	Environment Class	182
12.3.6	Build Phase	183
12.3.7	Connect Phase	184
12.4	Test Class	185
12.5	How Does the Test Proceed?	187
12.6	End of Test	191
12.7	Starting the Test From the Command Line	192
12.8	Summary and Coding guidelines	193
12.9	Exercises for Further Exploration	193

Part III Stimulus Generation in UVM

13	Generating Stimulus to Verify The DUT	197
13.1	Creating Your Transaction Stimulus	199
13.1.1	Constraining Data Items	201
13.1.1.1	Valid Value Constraints	202
13.1.1.2	Reasonability/Sane Constraints	202
13.1.1.3	Correctness Constraints	203
13.1.2	Inheritance and Constraint Layering	205
13.2	Creating Sequences	206
13.3	The Sequence Library	208
13.3.1	Selection Modes in a Sequence Library	209
13.4	Sequencer Operation	211
13.4.1	Generating Sequences and Sequence Items on a Sequencer	212
13.4.1.1	Using Sequence Methods	212

13.4.1.2	Using Sequence Macros	214
13.4.2	Configuring the Sequencer's Default Sequence	215
13.5	Sequencer Arbitration	216
13.5.1	Getting a Handle To the Sequencer From a Sequence	217
13.5.2	Sequence Priority	217
13.5.3	Lock/Unlock	219
13.5.4	Grab/Ungrab	220
13.6	Common Sequence Types	222
13.6.1	Flat Sequences	222
13.6.2	Hierarchical Sequences	224
13.6.3	Parallel Sequences	226
13.6.4	Reactive Sequences	227
13.6.5	Virtual Sequences and Sequencers	227
13.6.6	Interrupt Sequences	230
13.6.7	Layered Sequences	231
13.7	Sequence Callbacks	231
13.8	Coding Guidelines	233

Part IV Practical Verification using UVM - Applications

14	Example SOC Used for the Examples	237
14.1	The Example SOC	237
14.2	Memory Map	237
14.3	Directory Structure	239
14.4	Building and Running the Examples	240
15	UVM Registers with the VGA LCD Module	241
15.1	Integrating Register Models Into UVM Environments	242
15.1.1	Step 1. Creating the Register Model	244
15.1.2	Step 2. Creating an Adapter to Translate Bus Operations	248
15.1.3	Step 3. Choose a Predictor for the Register Models	250
15.1.3.1	Implicit Prediction	250
15.1.3.2	Explicit Prediction	251
15.1.3.3	The Register Predictor	252
15.1.3.4	Passive Prediction Without Active Components	254
15.1.4	Step 4. Instantiating the Register Model	255
15.1.5	Step 5. Creating the Register Model in the Environment	256
15.1.6	Step 6. Locking the model	256
15.1.7	Step 7. Connecting the Register Model Adapter	257
15.1.8	Step 8. Set the Front-Door Sequencer	258
15.1.9	Step 9. Accessing the Register Model	258
15.1.9.1	Front Door Access	258
15.1.9.2	Back Door Access	259
15.2	Various Built-in Tests Available From UVM	261
15.3	Build and Run Instructions	264
15.4	Exercises for Further Exploration	264
16	Factories, Multiple Register Interfaces, Reporting, Callbacks and Command Line Control using the Wishbone Conmax Crossbar	265
16.1	Verification Environment	265
16.1.1	Register Programming Model	267
16.1.2	The Register Model for Multiple Interfaces	268
16.1.3	Using a Scoreboard with Multiple Input Ports and a Simple Comparator	270
16.1.4	Putting it All Together: The Top Level Environment	272

16.1.5	The Build Phase	273
16.1.6	Connect Phase	275
16.1.7	Top Level Base Test	276
16.1.8	Creating a Specific Test	278
16.1.9	Selecting a Specific Test to Run	279
16.2	Factory Usage for Environment Creation	279
16.2.1	Using a Type Override	280
16.2.2	Using Instance Overrides to Override Specific Instances	282
16.3	Use of Config DB to Apply Various Environment Settings	283
16.4	Use Of a Virtual Sequencer to Direct Traffic to Wishbone Crossbar	286
16.4.1	Creating a Virtual Sequencer	286
16.4.2	Connecting the Virtual Sequence to Subsequencers	287
16.4.3	Creating a Virtual Sequence and Controlling Other Sequencers	287
16.5	Altering Message Verbosity From Some Components	288
16.5.1	Using Methods	288
16.5.2	Using the Command Line	290
16.6	Logging Messages From a Specific Component to a File	290
16.7	An Illustration of Flat Sequences	291
16.8	Using Callbacks	292
16.8.1	Instance Specific Callback Example	292
16.8.2	Type Wide Callback Example	294
16.9	Use Of The Command Line Processor	296
16.9.1	Command Line Options to Affect the Entire Simulation	296
16.9.2	Getting a Command Line Argument Into the Simulation	297
16.9.3	Controlling Sequences Using the Built In Command Line	298
16.9.4	Using the Command Line Processor to Configure Components	299
16.9.5	Setting Type Override from the Command Line	299
16.9.6	Setting an Instance Override from the Command Line	299
16.9.7	Setting Verbosity for Specific Components on Command Line	300
16.9.8	Setting a Specific Report Action for a Specific Component on the Command Line	302
16.9.9	Setting a Specific Action for a Specific Error for a Component on the Command Line	302
16.10	Build And Run Instructions	303
16.11	Exercises for Further Exploration	303
17	Stimulus Generation with Ethernet	305
17.1	Data Modeling of Ethernet Packets	306
17.2	Developing Interrupt Sequences in UVM	307
17.2.1	Step 1: Adding an Interface to the Environment	307
17.2.2	Step 2: Make the Interface Available Via config_db	308
17.2.3	Step 3: Use the Interface From the config_db in Your Sequence	308
17.3	Developing Layered Sequences in UVM	310
17.3.1	Developing Hierarchical Sequences in UVM	310
17.3.2	Developing Sequences as a Collection of Layers	313
17.4	Exercises for Further Exploration	313
18	Using UVM for Functional Verification Closure of NON-UVM Testbenches	315
18.1	Legacy Environment Description	316
18.2	Adding UVM Environment to Legacy Testbench	318
18.2.1	Changes to the Top Level Testbench	318
18.2.2	Connecting UVM Testbench to the Top Level	320
18.3	Coverage Model	321
18.3.1	Register Layer Based Coverage Model	321
18.3.2	The Functional Coverage Model	323
18.4	Taking Care of Phasing	324

18.4.1	The Reset Agent	326
18.4.2	Synchronizing the Phases in Legacy and UVM Testbenches	326
18.5	UVM Environment	328
18.5.1	Environment Block Diagram	328
18.5.2	The Register Model	328
18.5.3	The External Predictor Model	328
18.5.4	The Coverage Models	328
18.5.5	The Build Phase	329
18.5.6	The Connect Phase	330
18.6	Passive UVM Test Class	331
18.7	Observing Generated Coverage Results	333
18.8	Cover Additional Scenarios by Tweaking Additional Scenarios	334
18.9	Troubleshooting	335
18.10	Exercises for Further Exploration	335

Part V Advanced Topics in UVM

19	Synchronization, Watchdogs, Heartbeats and Events	339
19.1	Synchronizing Processes Across Components Using <code>uvm_event</code>	339
19.2	Using UVM Barrier Classes	345
19.3	Heartbeats From Simulations	349
20	Agents That React To Stimulus	353
20.1	Example of a Reactive Agent	354
20.1.1	Reactive Driver	355
20.1.2	Reactive Sequencer	358
20.1.3	The Creation of a Reactive Sequence	360
20.1.4	The Completed Reactive Agent	361
20.2	Other Approaches to the Reactive Agent	362
21	Advanced Register Concepts	365
21.1	Example DUT	365
21.2	Verification Environment for the DUT	366
21.2.1	Master Agent Sequence	367
21.3	Effect of Register Operations on Various Field Access Policies	370
21.4	Customizing Register Models	373
21.5	Hooks and Callback Facilities in Register Models	374
21.5.1	Factory Replacement for RAL Registers	375
21.5.2	Register Callbacks	376
21.5.2.1	Step 1: Create the Callback with the Appropriate Tasks	376
21.5.2.2	Step 2: Add the Callback to the Test	377
22	A Primer for UVM Config DB Regular Expressions	379
22.1	Component Classes	380
22.2	Config DB Regular Expressions in UVM	383
22.2.1	Case Sensitivity And Other Things To Note	383
22.2.2	WildCards	383
22.2.2.1	'*' Wildcards	384
22.2.2.2	'?' Wildcards	385
22.2.3	'+' Wildcard	386
22.2.4	Anchors	386
22.2.5	Repeat Operators	387
22.2.6	Marked Sub Expressions	388
22.2.7	Alternation	388

22.2.8 Precedence Among Operators	389
A Simple RAM slave	391
References	402
Glossary	403
Index	405

Foreword

As a person that has been involved from day one on the UVM effort, serving as co-chair of the Accellera UVM WG, I am glad to embrace Srivatsa's book, which will serve as an additional means to expanding the UVM community. As a frequent user and support person for UVM, I am glad to see a book with many important details that can teach one UVM quickly. Coding up a UVM testbench rapidly and correctly is growing required need in the Semiconductor industry. As designs are becoming more complex and schedule times are reducing, the pressure on the UVM testbench implementers is getting tougher and tougher. With this book, one can effectively learn practical aspects of UVM and can use it as a reference to get detailed information quickly.

A notable chapter is the configuration database; regular expressions are used to enable assignment of configurations to a desired set of components. The book provides detailed information about how the regular expressions are applied to achieve this.

Another notable chapter is about the register library; the book goes to great effort to point out the different register/field databases with details on how they are referenced and updated.

The author is deeply involved in the UVM IEEE efforts (P1800.2). I am also looking forward towards the first book that will have details on P1800.2 as it plans to come out early next year.

To summarize, the book has truly recognized the complexities of UVM and what is needed to bring new users on board and what is needed to enable existing users better.

Austin TX, March 2016

Hillel Miller
Accellera UVM Committee Co-Chair.

Preface

SystemVerilog allows you to use a single language for both design and verification of ASIC designs. The Universal Verification Methodology (UVM) allows verification engineers to leverage the SystemVerilog language with a standardized methodology. UVM is now widely supported by EDA vendors and used by many design teams worldwide to verify complex ASIC devices. Many new designs use UVM as a methodology for their projects moving forward. This book is a product of knowledge accumulated from many sources (UVM Users Guide and Class Reference, code comments, various DVCON presentations and papers, UVM/OVM books, UVM committee discussions and my own experience to name a few). Many examples illustrate key concepts and show how UVM may be used to verify complex ASIC devices. The provided examples combined with a practical hands-on approach with actual RTL cores should enable the verification engineer to explore UVM at their own pace.

Before you get started

The subject of UVM is vast. There are new developments often concerning this class library. To keep the discussion in this book focused on UVM, knowledge of SystemVerilog and object-oriented concepts are prerequisites and are not covered as a part of this book. Please visit our website www.uvmbook.com for links to enhance your knowledge of SystemVerilog if needed.

How is this book structured?

The focus of this book is practical learning. There are a series of examples, each illustrating a concept with insight into UVM. The examples in Part 1 progress from relatively trivial examples to elaborate full chip environments as the book progresses in Part 4. The examples are complete so that you can run them in a simulator. Part of this book's approach is that you execute/change the examples provided and work with them to learn UVM better.

During book development, a decision was made to use real world examples. The book structure takes the form of a practical DIY (Do It Yourself) format. I envision that you will download the examples to your computer and study UVM through this book as you execute the examples. The examples progress from simple to complex concepts while being complete in illustrating specific aspects of UVM, I have used many cores from opencores.org to help illustrate key concepts along the way. *The Verilog testbenches which were originally part of the opencores offering have also been included. This way, you can compare and contrast approaches, and make transition from Verilog to UVM + SystemVerilog.* If you are new to UVM, you have a reference design with testcases that you can relate to as a platform.

There are five parts to this collection. After you download and install the examples, the provided makefiles will build and run these examples. Feel free to examine and modify them as you see fit. The figure below illustrates the flow of UVM concepts through this book.

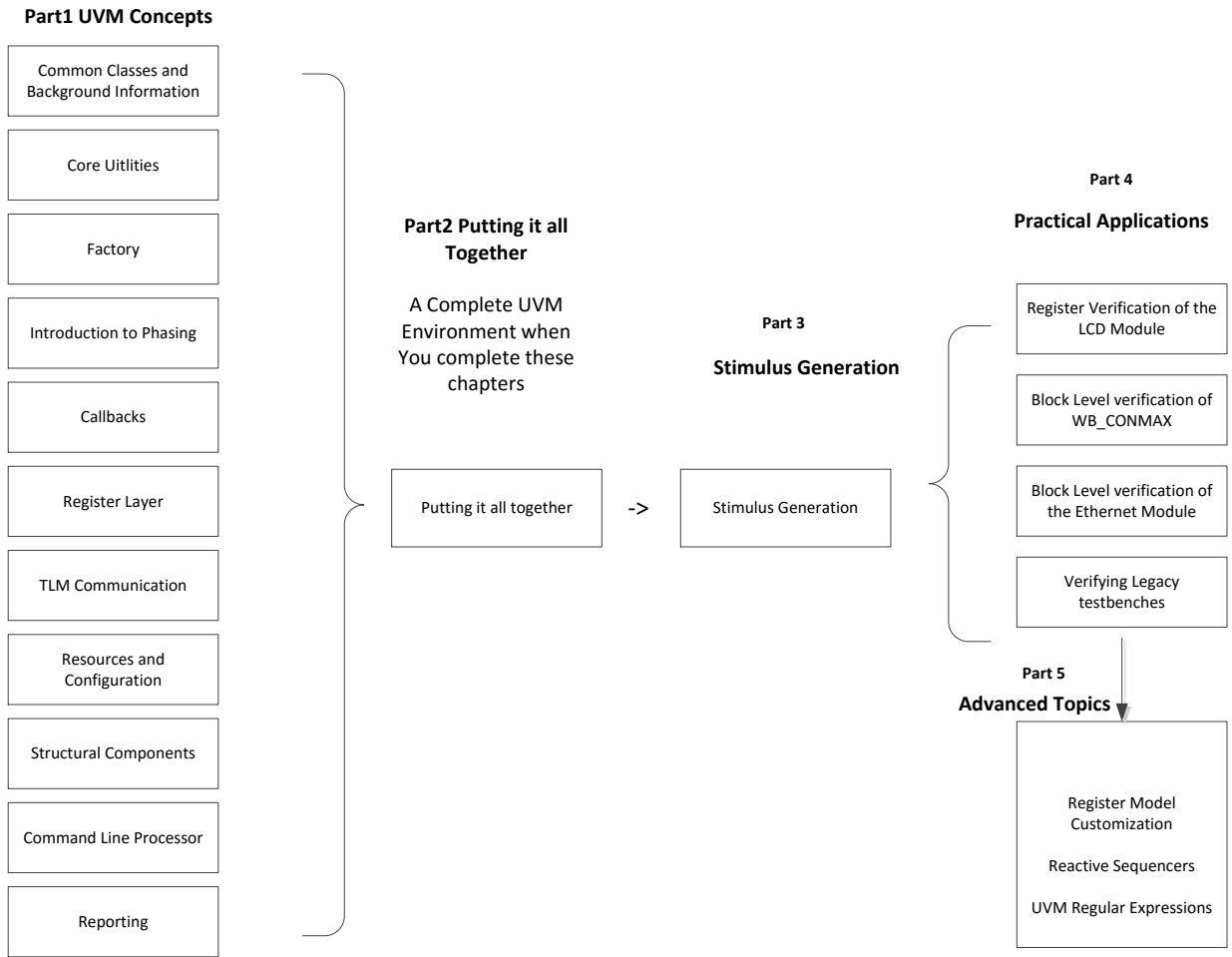


Fig. 0.1: Chapters in this book

PART 1: Overview of the UVM Class Library

This part is a UVM Tutorial. It begins with a UVM Library overview introducing some concepts and classes that you will need to complete your journey through in Parts 2,3,4. Completing this Part should give you a good idea of what the UVM library contains. If you know UVM, you can look into the content of this Part for any details.

Part 2: Stitching it all together

This part attempts to bring together concepts presented in Part 1 to put together a complete verification environment. It discusses in practical detail various aspects of a UVM environment and connects master and slave verification components through a simple pass through DUT. The components and concepts developed in this chapter are reused in Part IV. The wishbone protocol is chosen instead of the UBUS protocol described in the UVM Users Guide. While it would have been possible to extend the UBUS protocol description to allow some other transfers and also create examples using actual RTL with this protocol, such an effort would be rather significant. This book instead leverages RTL IP that has been proven elsewhere to teach UVM so that you can continue learning and exploring well documented IP.

Part 3: Stimulus Generation

Creation and driving of stimuli form a crucial activity in any verification effort. The UVM class library allows the user to

write reusable code and also provides some guidelines for how to structure this code. This part covers various concepts and considerations for stimulus generation.

Part 4: Block level verification environments

Part 4 now moves on to block level verification environments. Since it was not possible to address aspects of UVM in verification in a fair amount of detail in Parts 2 and 3, Part 4 provides practical examples of integrated UVM environments while highlighting aspects of UVM.

As some RTL cores with verilog environments are available from www.opencores.org, it makes it possible for readers who are coming into ASIC verification from a design world to relate more quickly to the concepts in UVM using these cores. Hence, all environments are built around these available cores merged into an example platform. You can also choose to add other RTL cores to this design or delve into a variety of other topics in verification using these examples as a platform.

Note: While complete environments have indeed been provided to you to learn and extend from, *it has not been my intention to either verify the core provided nor delve into details of each core in this book*. Such an exercise is left to you hoping that you will undertake it with the intention of exploring deeper into UVM at your own pace.

The main areas of UVM covered in Part IV are:

- UVM Register modeling in context of a Video Display Unit
- The cross bar random environment in UVM is used to explain many UVM concepts with multiple components in the environment.
- Data and stimulus considerations and various sequence types for a random verification of the Ethernet MAC in UVM
- Using UVM to enhance functional coverage of a DUT in a legacy environment.

Part 5: Advanced Topics

Part V now goes into some more advanced topics in UVM that are not covered in earlier sections. Content in this part assumes you have studied the earlier four parts. Some of the main areas covered are:

- Synchronization among processes in UVM
- Heartbeat applications to ensure your testbench is running
- Reactive sequencer applications in UVM
- Advanced Register topics
- Regular expressions in the UVM configuration database.

Hence on completion of this part of the book, you should be able to use UVM to verify your designs and tweak your environments to accomplish your specific goals.