

Srivatsa Vasudevan

Practical UVM

Step by Step with IEEE 1800.2

© Srivatsa Vasudevan 2011-2020.

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 or Srivatsa (at) gmail (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 P-2019.06 using Accellera UVM and the IEEE-1800.2 release.

International Edition. February 2020.

ISBN : 978-0-9977896-1-4

The author 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.

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

Please Read! Important Note for the IEEE Edition	xv
UVM Coding Guidelines and UVM Linting	xxv
Glossary	xxix
IEEE 1800.2 - The New UVM Standard	1
Migrating from UVM 1.2	2
Migrating to UVM 1.2 from UVM 1.1* prior to moving to IEEE 1800.2	2
UVM Versioning	3
Porting Hints	4
Removing Dependencies on `uvm_do Macros	4
Future Work on the IEEE Standard	5
Changes Between the First Edition and the IEEE Edition	7
Part I UVM QuickStart	
1 Beginning the UVM Journey	11
1.1 The DUT	11
1.2 A Verilog Testbench	11
1.3 A SystemVerilog Testbench	14
2 The UVM Testbench	19
2.1 How Does UVM Help You?	19
2.2 Some UVM Terminology	21
2.3 Creating a UVM Testbench	21
2.3.1 Create the Testbench Top-Level Module	22
2.3.2 Clocks and Reset	22
2.3.3 Connect the Interfaces	22
2.4 Create UVM Testbench Environment for the DUT	23
2.4.1 Environment Class	24
2.4.2 Master Agent	25
2.4.3 Slave Agent	25
2.4.4 Scoreboard	26
2.4.5 Coverage	26
2.5 Connect Testbench to DUT	27
2.5.1 The Test Class	28
2.6 Starting the Tests From the Command Line	30
2.6.1 How Does the Test Proceed?	30
2.6.1.1 Build Phase	31

2.6.1.2	Connect Phase	33
2.6.1.3	Run Phase	34
2.6.1.4	End of Test	36
2.7	Summary and Coding Guidelines	37
3	Generating Stimulus to Verify The DUT	39
3.1	Creating Your Transaction Stimulus	40
3.1.1	Constraining Data Items	43
3.1.1.1	Valid Value Constraints	44
3.1.1.2	Correctness Constraints	45
3.1.1.3	Reasonability/Sane Constraints	45
3.1.2	Inheritance and Constraint Layering	48
3.2	Creating Sequences	49
3.3	Using the Sequence in a Test Case	51
3.4	Exercises for Further Exploration	52
 Part II UVM Building Blocks		
4	UVM Core Utilities in IEEE 1800.2	55
4.1	Simple Example Classes	56
4.2	Object Creation	58
4.2.1	Create	58
4.2.2	Clone	59
4.3	Common Operations on Objects	60
4.3.1	Copy	61
4.3.2	Compare	62
4.3.3	Print	65
4.3.4	Packing And Unpacking	70
4.4	Core Operations Summary	73
4.5	UVM Macros to Simplify Things	73
4.5.1	Considerations When Using The Macros	77
4.6	A Complete Class in UVM	77
4.7	Coding Guidelines	79
4.8	Before Proceeding Further.	80
5	UVM Factory	81
5.1	Need for a Factory Pattern in UVM	81
5.2	UVM Factory Operation	82
5.3	Factory Behavior for Parameterized and NonParameterized Classes	85
5.4	Using the Factory	86
5.5	Replacing Components Using a Factory.	86
5.5.1	Type Overrides	87
5.5.2	Instance Specific Overrides	88
5.5.3	Rules for Processing Overrides	90
5.6	Debugging	91
5.7	New Features in IEEE 1800.2	91
5.7.1	Abstract Components	91
5.7.1.1	Step 1: Create Abstract Component Class	92
5.7.1.2	Step 2: Instance Abstract Component in your Environment	92
5.7.1.3	Step 3: Create Abstract Component in build_phase()	93
5.7.1.4	Step 4: Connect Abstract Component in connect_phase()	93
5.7.1.5	Step 5. Create the Override Class	93
5.7.1.6	Step 6: Set up Override in Top-level Testbench	94
5.7.2	Abstract Classes	95

5.7.3	Type Aliasing	95
6	Reporting Infrastructure	97
6.1	Elements of a UVM Report	98
6.1.1	Severity Specification	99
6.1.2	Verbosity Settings	99
6.1.3	Handling Specification	100
6.1.4	API Presented By UVM Components to Generate Messages	101
6.2	Reporting Subsystem - Practical Applications	102
6.2.1	Controlling Reporting from the Command Line	103
6.2.2	Controlling Verbosity	103
6.2.3	Logging Messages to a File	104
6.2.4	Demoting Reports From One Level to Another	105
6.2.5	Catching Reports and Changing Them	105
6.2.6	Using Reporting Infrastructure with Assertions and Modules	105
6.3	Behavior Differences between UVM 1.2 and IEEE 1800.2	106
6.4	Exercises for Further Exploration	106
7	Configuration and Resource Databases	107
7.1	Resource Database Infrastructure	107
7.2	Config Database Infrastructure	109
7.3	Priority for Setting Values in the Database	111
7.4	New Features in IEEE 1800.2	112
7.5	Using Regular Expressions in UVM	112
7.6	Debugging	113
7.7	Performance Tuning and Coding Guidelines	113
8	UVM Component Hierarchy	115
8.1	Overview of Capabilities Provided by UVM Component	116
8.1.1	Construction and Hierarchy Management	116
8.1.2	Component Configuration	117
8.1.3	Factory Interface	117
8.1.4	Reporting Interface	117
8.2	Macros Provided for UVM Components	118
8.3	UVM Library Component Classes	118
8.3.1	UVM Driver	119
8.3.1.1	Providing Driver Extensibility	123
8.3.1.2	Other Driver Types - Push Driver	125
8.3.1.3	Other Driver Variants	127
8.3.2	Monitor	127
8.3.3	Sequencer	130
8.3.4	Agent	132
8.3.4.1	Agent Configuration	133
8.3.4.2	Agent Build Phase	134
8.3.4.3	Agent Connect Phase	135
8.3.5	Subscriber	135
8.3.6	Scoreboard	137
8.4	The UVM Environment	139
8.4.1	Environment Class	139
8.4.2	Top Level Test	143
8.5	Component Configuration	145
8.6	New Features in IEEE 1800.2	147
8.7	Creating and Using UVM Components - Guidelines	150

9	Callbacks	153
9.1	Creating Callbacks in Components	153
9.1.1	Step 1: Create Virtual Callback Class	154
9.1.2	Step 2: Register the Callback Class	156
9.1.3	Step 3: Place the Callback Class Hook in Driver Code	156
9.1.4	Step 4: Extend and Use the Callback Class	157
9.2	User Applications of Callbacks	159
9.2.1	Typewide Callbacks	159
9.2.2	Instance Specific Callbacks	160
9.3	Order of Execution of Callbacks	160
9.4	New in IEEE - get_all callbacks	160
9.5	UVM Event Callbacks	161
9.6	Debugging Callbacks	161
9.7	Exercises for Further Exploration	162
10	Transaction Layer Communication	163
10.1	Basics of Transaction Level Communication	164
10.1.1	Put Port	165
10.1.2	Get Ports	166
10.1.3	Understanding Blocking and Non Blocking Methods.	167
10.1.4	Peek	168
10.1.5	Analysis Ports	168
10.2	Connecting Transaction-Level Components	169
10.3	UVM TLM - Ports Summary	171
10.4	UVM TLM 2	172
10.4.1	Blocking Transports	173
10.4.2	NonBlocking Transports	174
10.5	Sockets	177
10.6	Time	179
10.7	Connecting TLM2 Ports and Sockets	179
10.8	Generic Payload	180
10.9	Extensions	182
11	UVM Command Line Processor	183
11.1	Command Line Processor Basics	183
11.2	Built in Arguments	183
11.3	Available Global Settings	183
11.4	Controlling Simulation Behavior	184
11.4.1	Verbosity	185
11.4.2	Severity	185
11.4.3	Configuration	185
11.4.4	Starting Sequences From a Command Line	186
11.4.5	Factory Command Line Interface	186
11.5	Debug Switches	187
11.6	Custom Command Line Processor Examples	187
11.7	Exercises for Further Exploration	189
12	UVM Register Abstraction Layer	191
12.1	Typical UVM Register Use Model	191
12.2	Components of a Register Model	192
12.3	Register Maps	194
12.4	Register Model Architecture	195
12.5	Register Model API	196
12.5.1	Register Model Creation API	196

12.5.2	Register Model Hierarchy Traversal API	197
12.5.3	Register Model Access API	197
12.5.3.1	Reset	198
12.5.3.2	Read/Write	198
12.5.3.3	Get/Set	200
12.5.3.4	Peek/Poke	200
12.5.3.5	Randomize	201
12.5.3.6	Update	202
12.5.3.7	Mirror	203
12.5.4	Front Door Access	203
12.5.5	Back Door Access	204
12.6	Difference Between Backdoor and Peek/Poke Operations	205
12.7	Coverage of Register Models	206
12.8	Customization of Register Models	206
13	Phasing in UVM	207
13.1	Common Domain Pre Run Stages	207
13.1.1	Build_phase	209
13.1.2	connect_phase	209
13.1.3	end_of_elaboration_phase	209
13.1.4	start_of_simulation_phase	209
13.2	Run Time Phases	210
13.3	Clean Up Stages	212
13.4	Preventing Phases from Ending - Objections	213
13.5	Change in IEEE - use of get_objection before drain_time() in uvm_objection	218
13.6	Phase States	219
13.7	Phase Callbacks	219
13.8	Spanning Multiple Phases	222
13.9	Adding a Domain to a Schedule	227
13.10	IEEE UVM New features	227
13.10.1	Run Test Callbacks	227
13.10.1.1	Create the Run Test Callback	228
13.10.1.2	Add Callback to Test	229
13.11	Phase Jumping	230
Part III Advanced Topics in UVM		
14	Advanced Stimulus Generation	233
14.1	The Sequence Library	233
14.2	Sequencer Operation	235
14.2.1	Generating Sequences and Sequence Items on a Sequencer	237
14.2.1.1	Using Sequence Methods	237
14.2.1.2	Using Sequence Macros	239
14.2.2	Configuring the Sequencer's Default Sequence	240
14.3	Sequencer Arbitration	241
14.3.1	Getting a Handle To the Sequencer From a Sequence	242
14.3.2	Sequence Priority	242
14.3.3	Lock/Unlock	244
14.3.4	Grab/Unggrab	245
14.4	Common Sequence Types	247
14.4.1	Flat Sequences	247
14.4.2	Hierarchical Sequences	249
14.4.3	Parallel Sequences	251
14.4.4	Reactive Sequences	252

14.4.5	Virtual Sequences and Sequencers	252
14.4.6	Interrupt Sequences	256
14.4.7	Layered Sequences	256
14.5	Sequence Callbacks	256
14.6	Coding Guidelines	258
14.7	Working around Deprecated Sequence Macros	258
15	Synchronization, Watchdogs and Events	259
15.1	Synchronizing Processes Across Components Using <code>uvm_event</code>	259
15.2	UVM Event Callbacks in Action	264
15.2.1	Step 1: Create Event Callback Class	265
15.2.2	Step 2: Instantiate Callback Class	265
15.2.3	Step 3: Register Callback Class	265
15.3	Using UVM Barrier Classes	266
15.4	Heartbeats From Simulations	270
16	Agents That React To Stimulus	273
16.1	Example of a Reactive Agent	274
16.1.1	Reactive Driver	274
16.1.2	Reactive Sequencer	278
16.1.3	The Creation of a Reactive Sequence	280
16.1.4	The Completed Reactive Agent	281
16.2	Other Approaches to the Reactive Agent	282
17	Advanced Register Concepts	285
17.1	Example DUT	285
17.2	Verification Environment for the DUT	286
17.2.1	Master Agent Sequence	287
17.3	Effect of Register Operations on Various Field Access Policies	291
17.4	Customizing Register Models	294
17.5	Hooks and Callback Facilities in Register Models	295
17.5.1	Factory Replacement for RAL Registers	296
17.5.2	Register Callbacks	297
17.5.2.1	Step 1: Create the Callback with the Appropriate Tasks	297
17.5.2.2	Step 2: Add the Callback to the test	298
17.6	Summary	299
18	A Primer for UVM Config DB Regular Expressions	301
18.1	Component Classes	302
18.2	Config DB Regular Expressions in UVM	305
18.2.1	Case Sensitivity And Other Things To Note	305
18.2.2	WildCards	305
18.2.2.1	'*' Wildcards	306
18.2.2.2	'?' Wildcards	307
18.2.3	'+' Wildcard	308
18.2.4	Anchors	308
18.2.5	Repeat Operators	309
18.2.6	Marked Sub Expressions	310
18.2.7	Alternation	310
18.2.8	Precedence Among Operators	311
19	Advanced Core Utilities in IEEE 1800.2	313
19.1	Infrastructure Additions	313
19.1.1	<code>uvm_field_op</code>	314

19.1.2	do_execute_op	314
19.1.3	How Does This All Work?	314
19.1.4	Additional Policy Callbacks	316
19.1.5	Creating Your Own Behaviour with do_execute_op - A Print Policy Example	316
19.1.6	The Extensions Infrastructure	319
19.2	Introducing Callbacks in the Field Operations	321
19.3	Adding New Policy Classes to UVM	323
19.4	Summary	324

Part IV Practical Verification using UVM - Applications

20	UVM Registers with the VGA LCD Module	327
20.1	Integrating Register Models Into UVM Environments	327
20.1.1	Step 1. Creating the Register Model	330
20.1.2	Step 2. Creating an Adapter to Translate Bus Operations	333
20.1.3	Step 3. Choose a Predictor for the Register Models	335
20.1.3.1	Implicit Prediction	335
20.1.3.2	Explicit Prediction	336
20.1.3.3	The Register Predictor	337
20.1.3.4	Passive Prediction Without Active Components	338
20.1.4	Step 4. Instantiating the Register Model	339
20.1.5	Step 5. Creating the Register Model in the Environment	340
20.1.6	Step 6. Locking the model	341
20.1.7	Step 7. Connecting the Register Model Adapter	341
20.1.8	Step 8. Set the Front-Door Sequencer	342
20.1.9	Step 9. Accessing the Register Model	342
20.1.9.1	Front Door Access	342
20.1.9.2	Back Door Access	343
20.1.10	New in IEEE: Unlocking the model	345
20.2	Burst Read and Write	346
20.3	Various Built-in Tests Available From UVM	346
20.4	Build and Run Instructions	349
20.5	Exercises for Further Exploration	349
21	Factories, Multiple Register Interfaces, Reporting, Callbacks and Command Line Control using the Wishbone Conmax Crossbar	351
21.1	Verification Environment	351
21.1.1	Register Programming Model	353
21.1.2	The Register Model for Multiple Interfaces	353
21.1.3	Using a Scoreboard with Multiple Input Ports and a Simple Comparator	355
21.1.4	Putting it All Together: The Top-level Environment	358
21.1.5	The Build Phase	359
21.1.6	Connect Phase	360
21.1.7	Top-Level Base Test	361
21.1.8	Creating a Specific Test	363
21.1.9	Selecting a Specific Test to Run	364
21.2	Factory Usage for Environment Creation	364
21.2.1	Using a Type Override	364
21.2.2	Using Instance Overrides to Override Specific Instances	366
21.3	Use of Config DB to Apply Various Environment Settings	367
21.4	Use Of a Virtual Sequencer to Direct Traffic to Wishbone Crossbar	369
21.4.1	Creating a Virtual Sequencer	369
21.4.2	Connecting the Virtual Sequence to Subsequencers	370
21.4.3	Creating a Virtual Sequence and Controlling Other Sequencers	370

21.5	Altering Message Verbosity From Some Components	371
21.5.1	Using Methods	372
21.5.2	Using the Command Line	373
21.6	Logging Messages From a Specific Component to a File	373
21.7	An Illustration of Flat Sequences	374
21.8	Using Callbacks	374
21.8.1	Instance Specific Callback Example	375
21.8.2	Type Wide Callback Example	377
21.9	Use Of The Command Line Processor	378
21.9.1	Command Line Options to Affect the Entire Simulation	379
21.9.2	Getting a Command Line Argument Into the Simulation	379
21.9.3	Controlling Sequences Using the Built-In Command Line	381
21.9.4	Using the Command Line Processor to Configure Components	381
21.9.5	Setting Type Override from the Command Line	381
21.9.6	Setting an Instance Override from the Command Line	381
21.9.7	Setting Verbosity for Specific Components on Command Line	382
21.9.8	Setting a Specific Report Action for a Specific Component on the Command Line	384
21.9.9	Setting a Specific Action for a Specific Error for a Component on the Command Line	385
21.10	Build And Run Instructions	385
21.11	Exercises for Further Exploration	385
22	Stimulus Generation with Ethernet	387
22.1	Data Modeling of Ethernet Packets	388
22.2	Developing Interrupt Sequences in UVM	389
22.2.1	Step 1: Adding an Interface to the Environment	389
22.2.2	Step 2: Make the Interface Available Via config_db	390
22.2.3	Step 3: Use the Interface From the config_db in Your Sequence	390
22.3	Developing Layered Sequences in UVM	392
22.3.1	Developing Hierarchical Sequences in UVM	392
22.3.2	Developing Sequences as a Collection of Layers	394
22.4	Exercises for Further Exploration	395
A	UVM Core Utilities in UVM 1.2	397
A.1	Simple Example Classes	397
A.2	Object Creation	400
A.2.1	Create	400
A.2.2	Clone	401
A.3	Common Operations on Objects	401
A.3.1	Copy	401
A.3.2	Compare	403
A.3.3	Print	406
A.3.4	Packing And Unpacking	410
A.3.5	What Does The use_metadata Flag Do?	413
A.4	Core Operations Summary	413
	References	414
	Index	415

Please Read! Important Note for the IEEE Edition

Dear Reader:

Thank you for purchasing this book, created to offer knowledge, theories, and practical examples to advance your knowledge and effective usage of SystemVerilog and UVM. This book is both current and rare; few publications support verification and other advancements related to the Universal Verification Methodology.

This book is the result of "blood, sweat and tears" from many, including the author, graphic designers, editors and many in the UVM community on the inner workings of UVM. It has been prepared with considerable effort for your benefit with an effort spanning multiple years. I would greatly appreciate any feedback, suggestions, or comments; you have about this book - please send me your contribution to me via email, which is provided on the inside of the book cover. It shall be gratefully acknowledged.

The greatest profit to receive from the purchases of this book is donating all funds available after recovering the costs of creating this book and its distribution, to those in real need: *Those who suffer economic hardship with insufficient food for themselves and their children, and often without a place to call home or an opportunity to better themselves.*

Unauthorized copies, such as photocopied, DRM hacked or scanned, do not help us cover the costs, or offer the privilege and compassion enabling others to succeed as well. In many ways, your support of this book gives you an opportunity to help others as you advance your working knowledge of UVM for your own personal achievements.

I hope you will join me in this endeavor. Together, we can all make this world a better place.

Thank You
Srivatsa Vasudevan
Spring 2020

Changes Between the First Edition and the IEEE Edition

The order of chapters varies significantly between the First and IEEE editions. This reorganization was based on feedback from UVM trainers, who indicated that most new users would prefer to come up to speed on UVM and write stimulus quickly. Hopefully, this has been addressed in the UVM Quickstart - Part 1. Readers knowing UVM and wishing to obtain a more in-depth understanding may proceed to Parts II and III to gain in-depth knowledge. Advanced topics have been moved to Part III from Part V of the previous edition. Part IV provides complete practical examples, just like the previous edition. You will find some additional examples in this part as well that address topics in UVM 1800.2.

Readers have offered feedback related to editing and typographical errors in the first edition of this book. Considerable work has been put in to eliminate as many as possible. Many paragraphs were rewritten to clarify the intent and meaning. The book has also been reviewed by a technical writer.

If you work in a company, there is a good chance that you will be working with multiple UVM versions. When you are especially dealing with an SOC, where you inherit multiple environments, each at a different UVM version; things usually get a little interesting. Hence, this book attempts to serve all three different versions of the library UVM 1.1, UVM 1.2, and IEEE 1800.2. Version specific information (where available) is marked using the following icons:



This icon is for features new to UVM 1.2 from UVM 1.1 **ONLY**. If you see an issue between migration from UVM 1.1x to UVM 1.2, watch for this icon. If this feature has changed between the UVM 1.2 and IEEE 1800.2 versions, one of the other icons will also be present in the text adjacent to this icon.



Code/behavior that has changed between UVM 1.2 and IEEE 1800.2 is marked with this icon. Note that this icon is the same one used for the Accellera to IEEE standard implementation differences.



This icon is for features of UVM deprecated between 1.2 and IEEE 1800.2 **only**. It does not cover deprecation between UVM 1.2 and UVM 1.1x. See the migration guide in the previous chapter if you are attempting to migrate to the IEEE 1800.2 version from UVM 1.2.



This icon alerts you to features **new** in IEEE 1800.2 and not present in UVM 1.2. Please do not confuse this with features between UVM 1.2 and UVM 1.1x

Part 1:

This part is specifically geared to engineers wanting to pick up UVM quickly.

- Chapter 1 presents a UVM overview and helps Verilog users understand UVM testbench architecture and philosophy and make a transition to UVM.
- Chapter 2 introduces the UVM environment.
- Chapter 3 provides you with information on creating stimulus for your DUT in UVM quickly.

Part 2:

- Chapter 4 goes over changes to `uvm_object` and the associated core utilities. There are a number of changes in IEEE 1800.2, and you should review this chapter as UVM 1800.2 *is not backward compatible* with earlier versions.
- Chapter 5 goes over the additions to the UVM factory class to support abstract types and aliases. Additional examples are added for abstract classes and factory replacement.
- Chapter 8 on the UVM component hierarchy adds examples explaining the changes in the `build_phase` in UVM 1800.2, allowing you to speed things up.
- Chapter 6 goes over the changes to reporting in IEEE 1800.2. You may see additional failures in your regressions. Please review this chapter as additional changes may be required on your part in your scripts.
- Chapter 9 goes over callbacks and their behavior. Callbacks have been enhanced in IEEE 1800.2.
- The Register Abstraction layer has additional examples to illustrate IEEE capability. Chapter 20 discusses unlocking the register model.
- Chapter 13 on Phasing discusses changes with additional examples due to capabilities added in IEEE version.
- Chapter 14 discusses Advanced Stimulus Generation. The original chapter is partitioned from the earlier book to deal solely with advanced topics. Examples now use the UVM 1800.2 style.

Advanced Topics

- Chapter 15 provides examples of how to add event callbacks to an event, a new feature introduced in IEEE version.
- Chapter 19 has descriptions of the new capabilities in the common operations. Details on how to customize your environment using the capabilities of the IEEE version are included here.

Practical Applications:

- Chapter 21 uses a different comparison function in the scoreboard.
- Content on `uvm_event`, `uvm_barrier` has been relocated from Chapter 22 to Chapter 15.

Deprecated Core Utilities

Appendix A now describes the UVM 1.2 core utilities and behavior for `compare/pack/print` functions which have undergone enhancements in IEEE 1800.2. Some of the API described in this appendix has been deprecated in the IEEE version.