Srivatsa Vasudevan

Practical UVM

Step by Step Examples

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

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 books 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 envison 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.

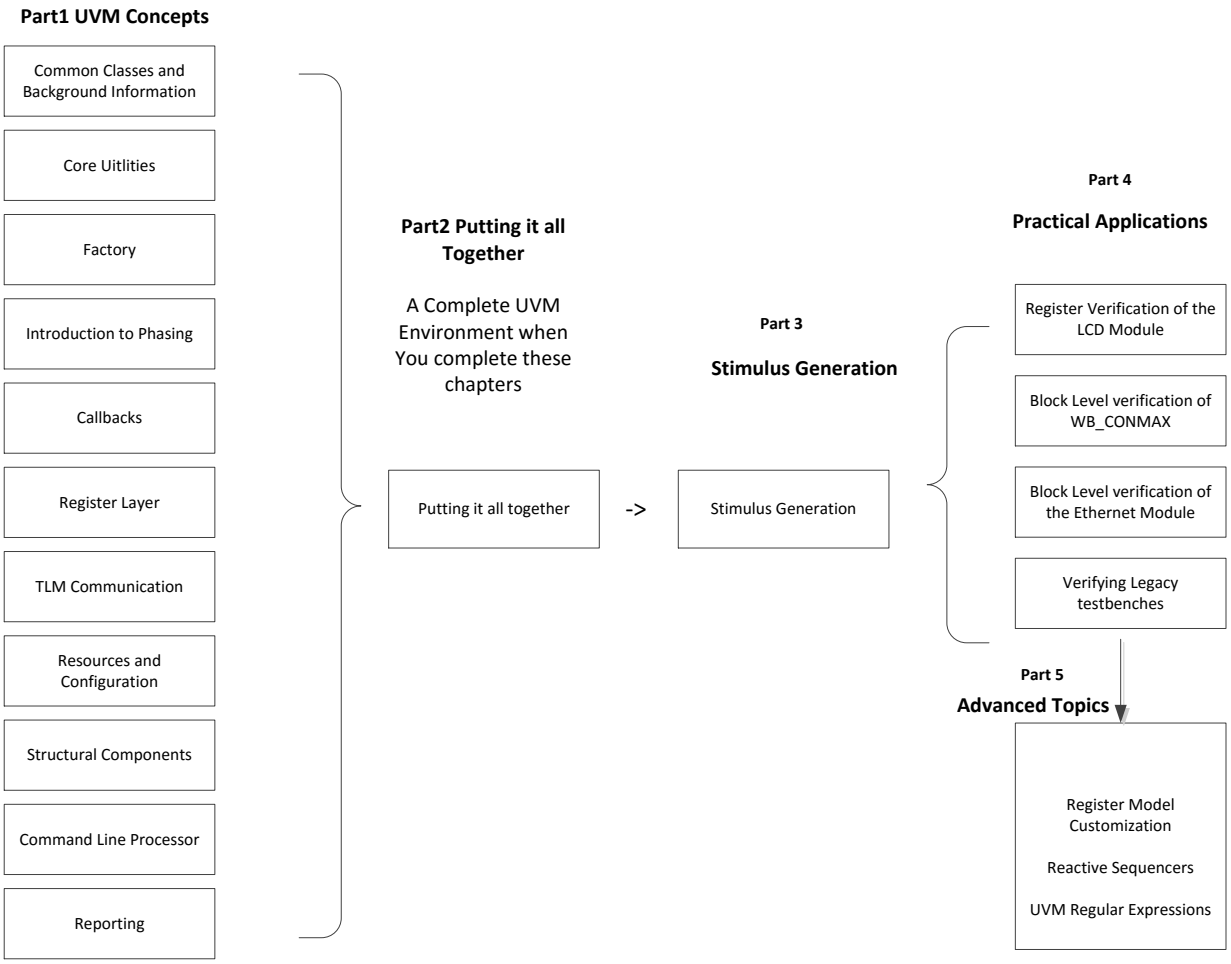

Fig. 0.1: Chapters in this book

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.