iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VIII **Month of publication:** August 2022

DOI: https://doi.org/10.22214/ijraset.2022.46466

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VIII August 2022- Available at www.ijraset.com

Study on Transformation to Universal Verification Methodology

Manjiri Kulkarni[1], Dr. S P Meharunnisa[2]
[1, 2]*Dept of EIE, DSCE Bengaluru*

Abstract: ***There are many approaches to the RTL design Verification. The different types of approach can be software simulation based, hardware accelerated simulation, formal verification etc. It helps to verify the correctness of the design, functional implementation and enhancing the design at every stage. Transition to Systemverilog has been done to cope up with the shrinking size of technology nodes and Time to market. In advanced times, the systems are to be designed in a generic way. The number of registers in the architecture increases as the number of combinations increases. Additionally, memory size is increasing as a result of the increased market need for data storage. An innovative approach is needed to access and validate the large variety of registers and the memory. A SystemVerilog class library called UVM was created specifically to support the creation of modular, reusable verification components and test-benches. Since it is an industry standard, you can use UVM IP that you obtain from other sources in your environment. You'll need to construct everything from scratch if you don't utilise UVM.***

Keywords: UVM, Verification, SystemVerilog, Simulation, Design

I. INTRODUCTION

This document is a template. For questions on paper guidelines, please contact us via e-mail. To make it simpler and more flexible to build reliable test environments, UVM intends to help you validate those challenging ideas with ease. It makes component modification simple. Each component can be overridden in various tests or environments without changing the code by being created using a factory. In precise terms, Universal Verification Methodology is a library of systemverilog code created to aid engineers in achieving efficient way of test creation and verification environment creation. On Accellera's website, you can find the UVM class library code as well as the user manual and reference materials. UVM is being standardised as part of the IEEE 1800.12 standard. In addition to having a class library that makes it simple to create effective constrained random testbenches, UVM has a specified process for architecting testbenches and test cases.

The benefits of UVM can be explained as Reusability and Modularity: The UVM technique is made up of modular components (Drivers, Sequencers, Agent, Env, etc.), which allows components to be reused across projects or from IP to SoC/Chip for IP level to SoC/Subsystem/Chip level verification. Separating Tests from the Test Bench: In terms of stimuli/sequences, the testbench hierarchy is kept distinct from the tests, enabling the reuse of stimulus across projects. Sequence methodology: This provides strong stimulus generation control. There are numerous techniques to build sequences, such as randomization, layered sequences, virtual sequences, etc. This provides a good ability for producing controlled random stimuli. Setting up a configuration: It makes configuring objects with intricate hierarchies simpler. The configuration technique makes it simple to setup various testbench individual components on the verification environment that uses it without having to worry about how deep each component is in the testbench hierarchy.

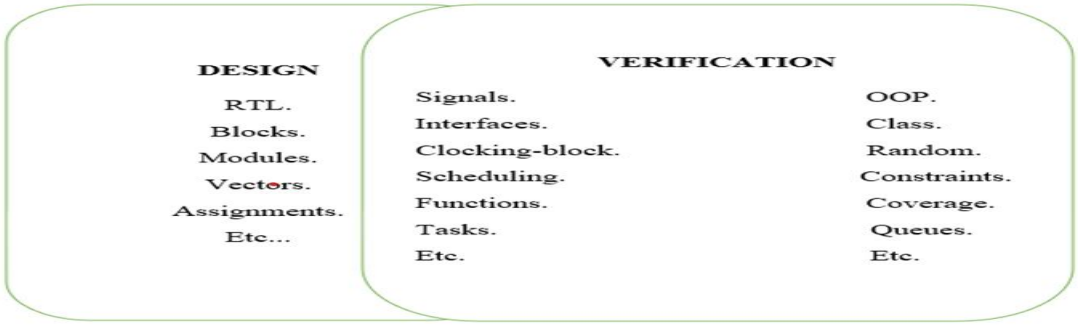


Fig 1 Components of Design and Verificati0n[1]

II. SYSTEM VERILOG AND UVM

1) Hardware verification language- SystemVerilog and UVM are two different things.
2) UVM is a well-defined class library with readily available reusable verification objects and components that allow us to quickly build a verification environment.
3) UVM is not more sophisticated than SystemVerilog; rather, it is a more modern standard library of predefined components and objects.
4) Now, if we created a Verification Environment using simply SystemVerilog, each and every component would need to be generated before being used.
5) It takes a lot of time and is not feasible in current projects because there are many intricate components that need to be built.
6) For instance, it is a difficult effort to design our own Driver, Monitor, and Sequencer.
7) In contrast, UVM allows you to use them directly by extending from a base class that is already there.
8) The execution of Coverage components, which are also available and follow specific phases, is particularly trustworthy for effective verification execution.
9) Systemverilog will be utilised most effectively when using specific techniques like OVM, UVM, VMM, etc.
10) Even with methodology, one HDVL language is required, such as Specman or Systemverilog.
11) UVM verification cannot be performed without familiarity with HVLs, such as systemVerilog.

III. UNIVERSAL WORK-FLOW FOR VERIFICATION

All the resources required to start building an efficient and repeatable test environment are available in the UVM Class Library. Some of the library's classes and methods are depicted in figure 2's picture.

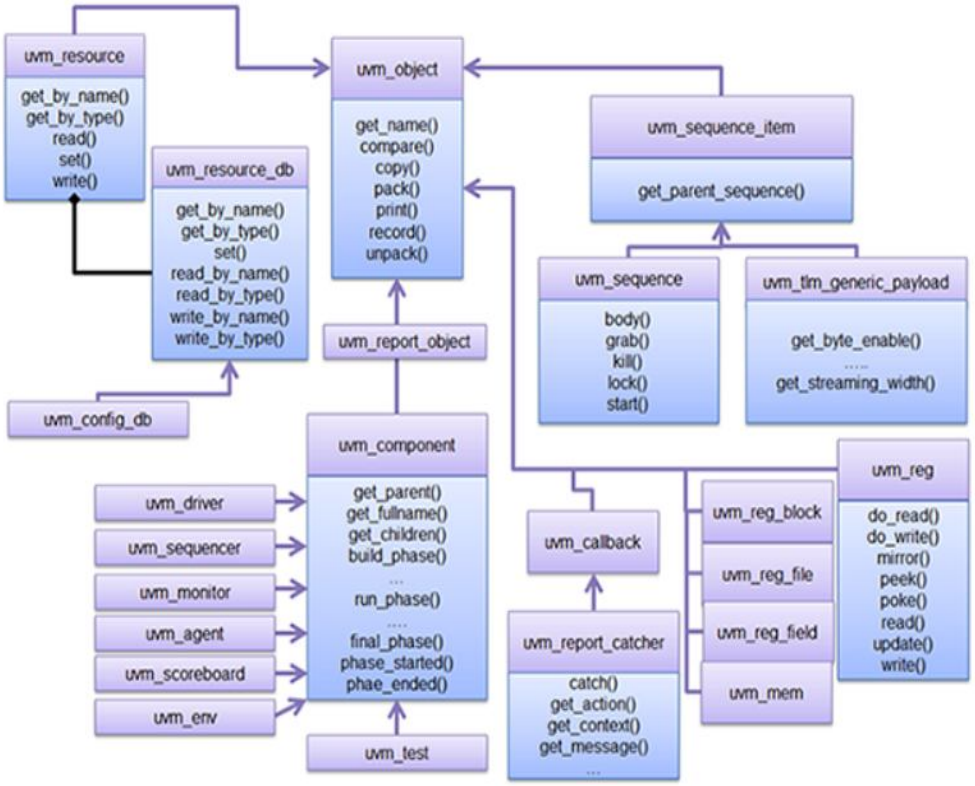


Fig 2. UVM Classes Diagram [1]

A. *Key Pronciples Of UVM*

1) *Constraiined Random Stimulation:* The issue is that simulating all potential signal combinations for a design will take too much of effort. In order to maximise coverage, UVM handles the situation by limiting the stimulus of the code to particular values and randomising particular cases. You have unheard-of control over how your test pattern generation is done. UVM makes it simple to create unique signal sequences that will force your design into desirable corner situations.

2) *Reusing Code:* UVM was created to offer a tonne of reusable "boilerplate" code. For each new design, it is no longer necessary to start from scratch. Simply make small adjustments after pasting a verification environment from another project to make the most of your time.
3) *Typical Verification Techniques:* As the use of UVM spreads, utilising a standardised process to test your designs ensures uniformity among the engineering community. Other engineers, including team members, will be able to understand your verification environment. Without needing to describe your verification processes, send your files to others. You may even troubleshoot your verification environment by finding information on UVM-specific forums.

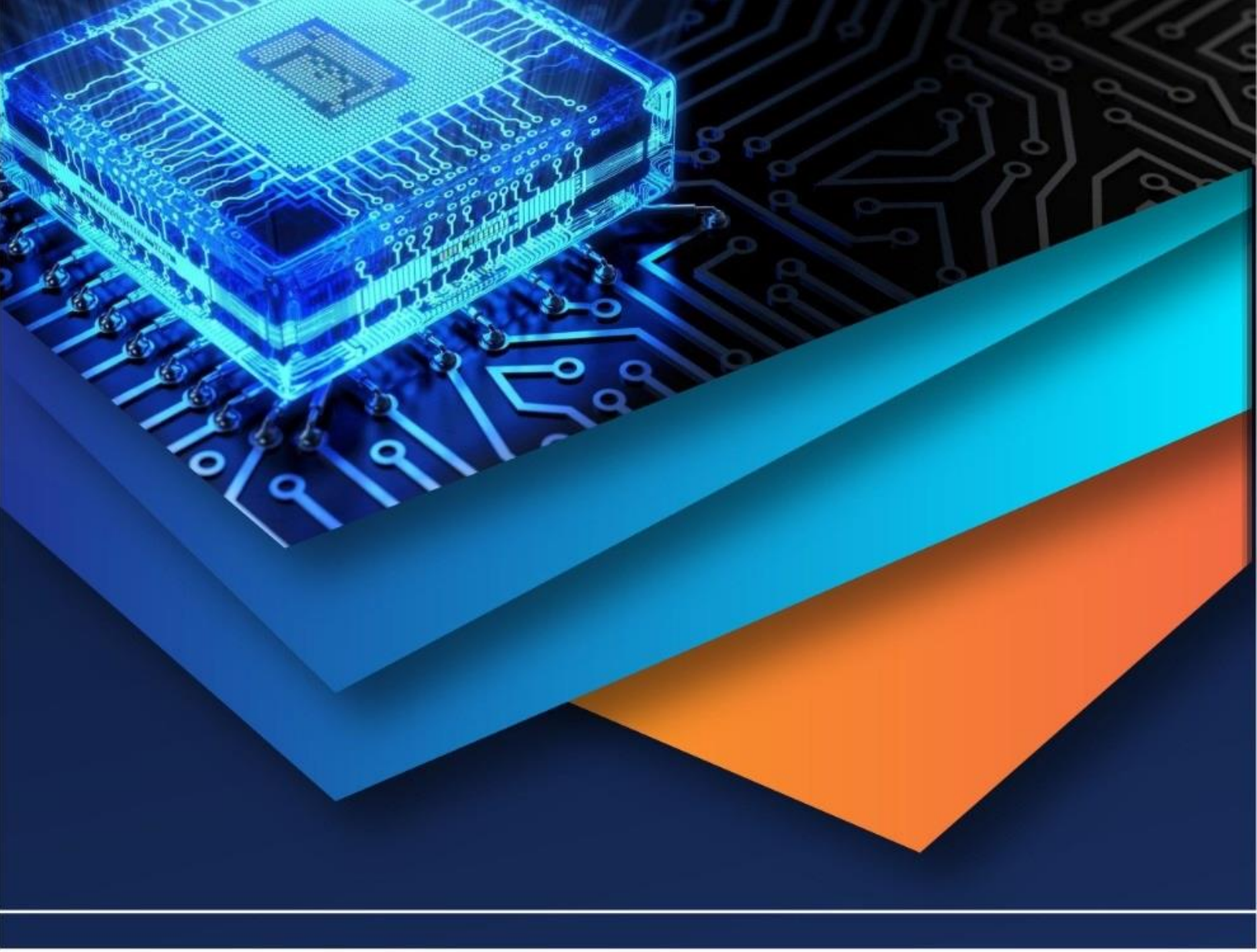
IV. CONCLUSIONS

Based on the tests done in UVM [2], it can be concluded that the transition of design verification from SystemVerilog to using a standard set of methodology- UVM is a more advanced and less complex way of verification in IP and SOC Validation. UVM allows us to develop modular, reusable verification environments, components, and tests, which achieves its objectives. When using UVM, it's crucial to understand that the majority of the library's implementation is devoted to infrastructures and support for the comparatively small set of features that test-case writers, environment(env) writers, and sequence writers will actually utilise.

REFERENCES

[1] https://www.aldec.com/en/company/blog/110--uvm-spells-relief
[2] Vitankar, Pankaj & Kureshi, A.K. (2016). UVM ARCHITECTURE FOR VERIFICATION, International Journal of Electronics and Communication Engineering & Technology. 7. 29-37.
[3] N. B. Harshitha, Y. G. Praveen Kumar and M. Z. Kurian, "An Introduction to Universal Verification Methodology for the digital design of Integrated circuits (IC's): A Review," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 1710-1713, doi: 10.1109/ICAIS50930.2021.9396034
[4] W. Ni and J. Zhang, "Research of reusability based on UVM verification," 2015 IEEE 11th International Conference on ASIC (ASICON), 2015, pp. 1-4, doi: 10.1109/ASICON.2015.7517189.
[5] Universal Verification Methodology (UVM) 1.1 User guide by Accellera
[6] Universal Verification Methodology (UVM) 1.2 Class by Accellera
[7] SystemVerilog 3.1a Language Reference Manual, Accellera's Extensions to Verilog
[8] Bromley, Jonathan. (2013). If SystemVerilog is so good, why do we need the UVM? Sharing responsibilities between libraries and the core language. 1-7.
[9] Incisive Enterprise Specman Products, Verification automation from block to chip to system levels, Cadence

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)