2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

SmartLint Booster: Automation of UVM Testbench Linting with AMIQ Verissimo

Gustavo Guzman Rosales, Gurudatt B M, Richa Gupta and Kanai Ghosh
NXP Semiconductors

***Abstract*- In the evolving landscape of design verification, maintaining code quality and adherence to Universal Verification Methodology (UVM) standards is paramount. AMIQ Verissimo, a powerful linting tool, plays a crucial role in identifying structural issues like coding errors, style violations and non-standard constructs in System Verilog and UVM testbenches. This paper presents an automation framework tailored to AMIQ Verissimo, designed to streamline the linting process, minimize manual effort, and standardize the execution flow. By integrating AMIQ Verissimo into an automated environment the framework enhances verification efficiency, enforces coding consistency, and significantly reduces turnaround time. The proposed solution empowers verification teams with a faster, more reliable, and scalable linting methodology.**

I. INTRODUCTION

The work presents the automation of the linting process for UVM testbench files using the AMIQ Verissimo tool, integrated within an automation framework. The automation is structured into three distinct phases to streamline and standardize the lint flow.

The *define phase*, involves specifying all necessary parameters for the linting process. These parameters are defined based on their nature and role within the flow, setting the foundation for subsequent steps. The *setup phase*, focuses on generating the required configuration, run scripts, and support files. Once these files are prepared, the Verissimo tool will be invoked to initiate the linting operation, ensuring the automation environment is fully configured. The final phase is the *execute phase*, during which setup scripts are executed.

II. METHODOLOGY

The methodology follows three phases of automation framework, and they are namely Define, Setup and Execute phases as shown in Figure 1. In the define phase the parameters are defined along with the type of parameter it is as per functionality. For example, here ruleset is a parameter used in verissimo flow which requires an input file to be given in the CLI, so the ruleset is defined as per its functionality. In the setup phase the necessary files are generated, and the configuration steps are performed. In the execute phase the run scripts or the setup scripts are executed, and the flow will be completed. Here in verissimo linting flow the necessary parameters are defined and the build configuration file, testbench list are generated and the uvm packages are accessed and the lint run is performed.

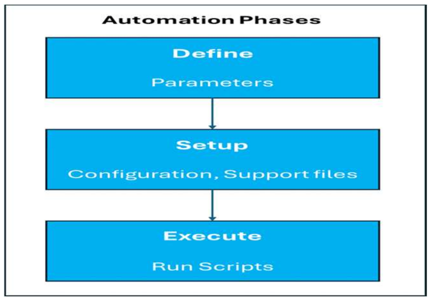

Figure 1. Phases of Automation

The traditional flow of tool flow and the automation framework which is developed can be seen in the Figure 2.

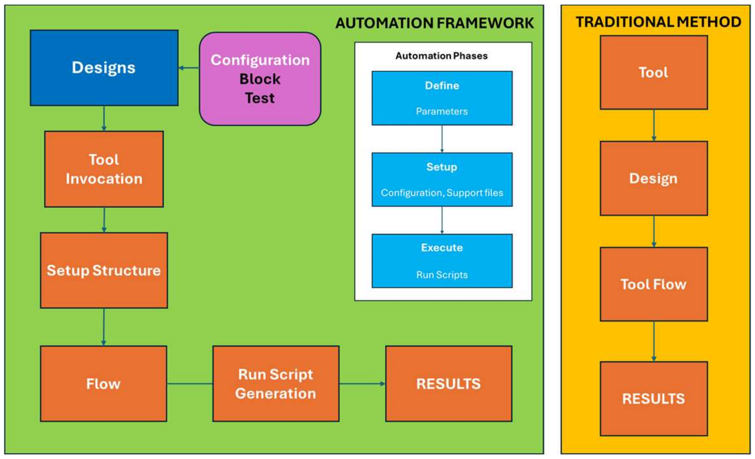

Figure 2. Automation Framework

In the automation framework, the process is divided into three distinct phases such as Define, Setup, and Execute which take inputs from designs and configuration is done, invoke tools, set up the environment, and generate run scripts to produce results efficiently. In contrast, the traditional method follows a linear and less modular path from tool invocation to results, lacking the flexibility and scalability of the automated approach. This comparison underscores how Verissimo automation enhances verification by improving clarity, reusability, and reducing manual effort.

III. IMPLEMENTATION

The verissimo tool implementation has been automated where in it requires the input files of ruleset and waivers in the form of xml files. The intermediate files are generated such as block configuration files which has information about uvm packages and testbench related information. There has been one filelist which is generated that contains list of testbench related files which will be used in lint flow. There is one more file which contains details of all the parameters which are used along with input path and the output at which it is to be stored. The automation has been made in such a way that there is standard directory and location format to store every file which come across the lint run.

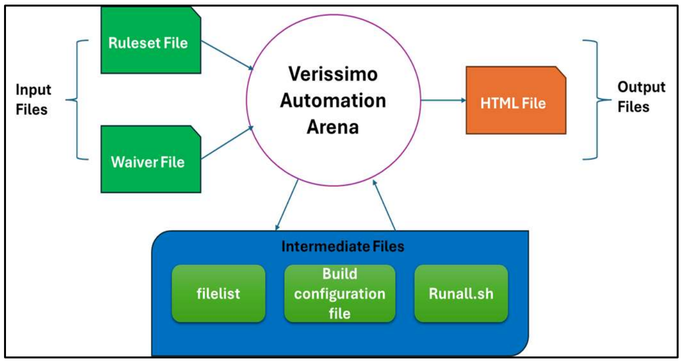

Figure 3. Inputs and outputs of lint flow

The inputs and outputs of the verissimo lint flow is shown in Figure 3 and there are necessary files generated which are required for lint run. The input files ruleset and waivers are fetched from custom or standard directories. A set of intermediate files are created, which includes testbench filelist, build configuration file containing top block and uvm package information, and a run script to track information of all parameters. This setup allows for a flexible and automated verification lint flow tailored for UVM-based environments.

The implementation of verissimo can be given as a flow which is given below. This contains the flow of execution in the automation run.

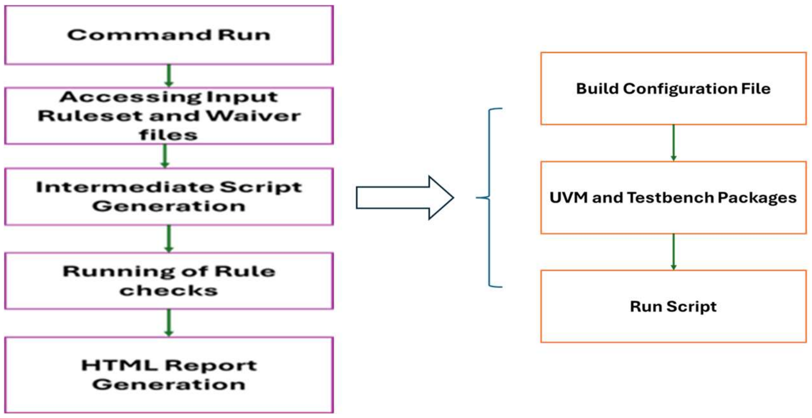
Command Run
Accessing Input Ruleset and Waiver files
Intermediate Script Generation
Running of Rule checks
HTML Report Generation
Build Configuration File
UVM and Testbench Packages
Run Script

Figure 4. Flow Diagram

The command is executed in the Linux shell and the input files of ruleset and waivers are fetched using parameters used in CLI. Then the build configuration file, uvm testbench filelist and the run scripts are generated, and the rule checks are verified on those testbench files and bases on the checks the summary of lint run is obtained, and the HTML Report is generated.

IV. RESULTS

The automation framework wraps the UVM testbench lint flow, providing an interface to user that facilitates the tool setup, execution and report results in standard location, the automation eliminates the manual tool setup and automates 100% the lint execution flow. The automation framework has been implanted in a way that can be escalated from IP to SOC level and used by different projects across design verification teams. The framework supports both command-line interface (CLI) and graphical user interface (GUI)modes, making it flexible and user-friendly. The verissimo tool report is generated in HTML format as the same format as running the verissimo tool standalone. Given the increasing complexity of UVM-based designs, this automated approach ensures consistent enforcement of coding guidelines, thereby enhancing code quality and maintainability.

The results obtained will be pertaining to a particular design and the summary of the checks are generated and the same is plotted in html file in the form of pie chart with detailed information of checks and the errors if any with respect to each rule checks. For any given design with varying complexity of the design and the ruleset provided the html report and summary of the check is obtained in the verissimo lint automation flow.

V. CONCLUSION

The automation framework is developed in a modular way, internally it will run three-phases, from setup to execute the verissimo tool for UVM testbench lint checks. Enabled full integration with CLI and GUI, supporting diverse user needs and streamlines the design verification process. The tool setup has been fully automated to reduce the manual effort and increased consistency of linting across design verification projects to improve code quality by enforcing UVM standard guidelines. It has been demonstrated adaptability across multiple design at different levels without the need for reconfiguration. In the future the tool automation can be integrated in continuous integration and continues delivery pipelines and adding support for additional static analysis tools and advance rule customization.

ACKNOWLEDGEMENT

The authors would like to thank Front End Design Enablement Team and NXP Semiconductors for the support given towards completion of the work.

REFERENCES

[1] https://eda.amiq.com

[2] https://www.chipverify.com

[3] https://vlsiverify.com