2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

ChipGuard: A Robust Automated System to Streamline Design Verification Quality Parameters

Jitender Dahiya, Giridhar Rangarajan, Sriram Kazhiyur Soundarrajan, Albin P Jacob

Samsung Semiconductors India Research, Bangalore, India

j.dahiya@partner.samsung.com

g.rangarajan@samsung.com

sriram.k.s@samsung.com

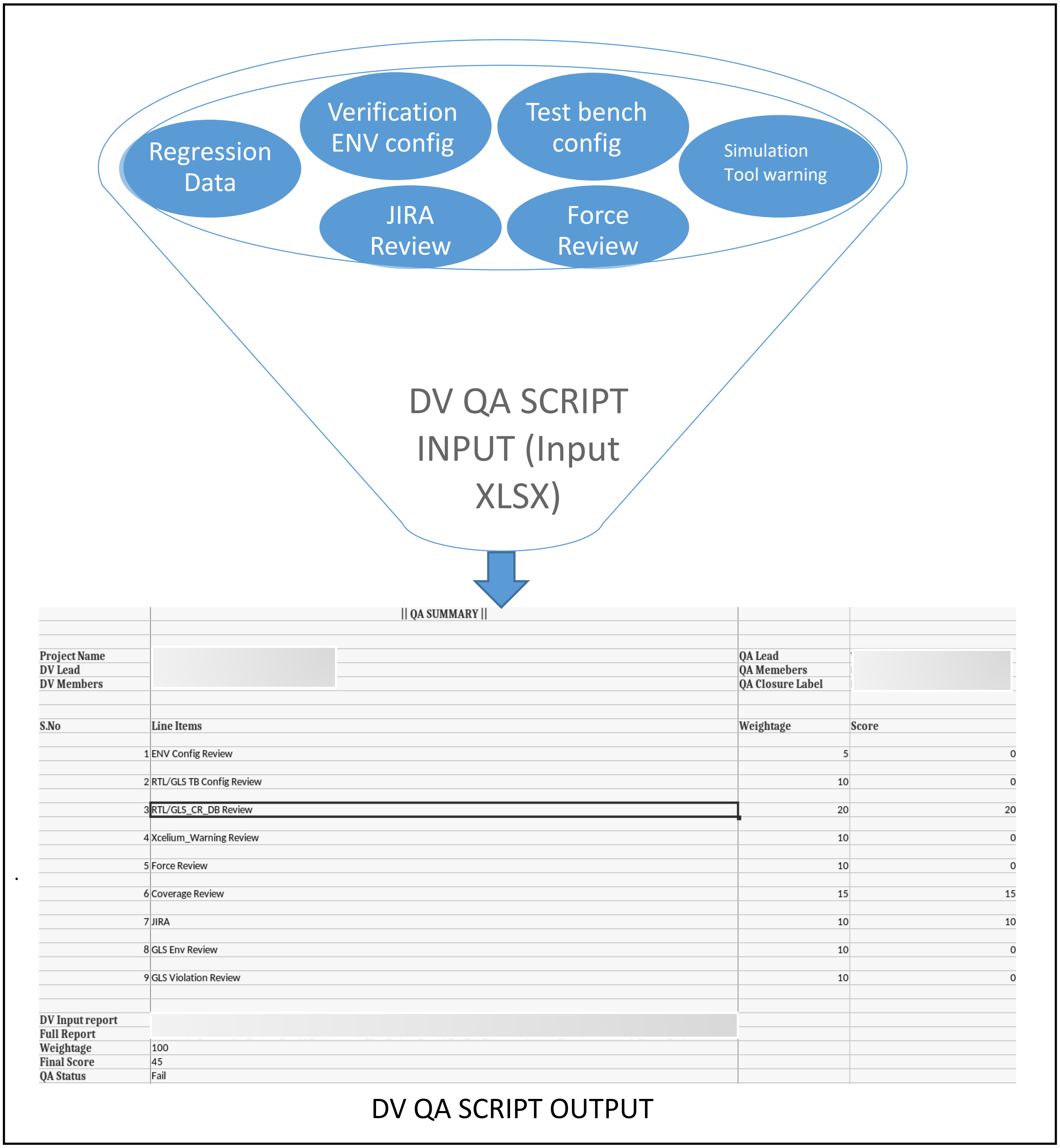
ap.jacob@samsung.com

***Abstract*- This document specifies the automation of processes involved in chip design verification represents a significant step forward. This advancement entails the utilization of scripting techniques to streamline the verification process, thereby reducing human intervention and potential errors. Rather than relying on manual verification methods that are time-consuming and prone to human error, automation offers a solution that ensures accuracy and reliability in chip design verification(DV). This approach not only optimizes resource utilization but also enhances the overall reliability and robustness of chip design verification procedures.**

I. Introduction

In the realm of semiconductor design and verification, the process of ensuring the correctness and functionality of integrated circuits (ICs) is paramount. DV Engineers are responsible for this, but we also need to think about the quality of DV. Traditionally, the process of validating the quality of DV has been a labour-intensive task. In response to these challenges, there has been a growing emphasis on the development of automated quality assurance (QA) systems aimed at simplifying and streamlining the process. The overarching goal of such systems is to alleviate the burden placed on DV engineers by automating repetitive tasks and minimizing the need for manual intervention. By leveraging automation technologies, these systems aim to enhance the efficiency, accuracy, and reliability of the verification process while simultaneously reducing the time and effort required to achieve comprehensive verification coverage. Moreover, by providing comprehensive coverage of the design space and identifying potential issues early in the development cycle, automated QA systems help mitigate the risk of costly design errors and delays.

Furthermore, an automated QA system serves as a critical component of the broader quality assurance framework implemented by semiconductor companies to ensure the integrity and reliability of their intellectual property (IP). Through rigorous QA audit processes, these systems validate the correctness and functionality of IP blocks, subsystems, and full-chip designs, thereby instilling confidence in the quality of the final product delivered to clients. By leveraging automated verification methodologies and QA processes, semiconductor companies can demonstrate their commitment to delivering high-quality, error-free IP to their clients while minimizing the risk of costly design re-spins and delays.



|| QA SUMMARY ||

Project Name		QA Lead	
DV Lead		QA Memebers	
DV Members		QA Closure Label	

S.No	Line Items	Weightage	Score
1	ENV Config Review	5	0
2	RTL/GLS TB Config Review	10	0
3	RTL/GLS_CR_DB Review	20	20
4	Xcelium_Warning Review	10	0
5	Force Review	10	0
6	Coverage Review	15	15
7	JIRA	10	10
8	GLS Env Review	10	0
9	GLS Violation Review	10	0

DV Input report	
Full Report	
Weightage	100
Final Score	45
QA Status	Fail

Fig 1: QA Inputs and Outputs

When the script starts running, it prompts the user to choose from one of the four modes from “Fig. 2”:

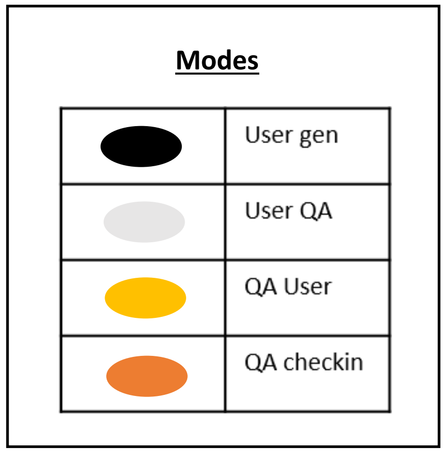


Fig 2: Different QA modes

A. *User Generation Mode*

The script begins by prompting the user to select one of four modes, with the first mode being user generation mode. This initial mode is crucial as it generates essential files required for subsequent modes. The generated files include simulation tool warning and error files, mconf warning files, GLS violation files, and Jira details files.

In the user generation mode, the script facilitates the creation of these files, which serve as inputs for the other three modes. Among these, the jira details file is particularly significant, as it compiles information on Jira IDs utilized throughout the project, accompanied by their respective states (e.g., open, closed, or deferred).
Upon completion of the user generation mode, the generated files are typically updated by the user in the input Excel file, often named QA_Input.xls. This Excel file contains paths referencing the corrected generated files, which are then utilized as input for modes 2, 3, and 4.

Subsequently, modes 2, 3, and 4 operate based on the updated file paths within the Excel file, leveraging the information provided in the corrected generated files. This ensures that the quality assurance (QA) process progresses efficiently, utilizing accurate and up-to-date data to inform decision-making and facilitate further analysis.

In summary, the user generation mode serves as the initial step in the QA process, generating essential files and laying the groundwork for subsequent modes. By allowing users to update and refine the generated files as necessary, it ensures that the QA process operates smoothly and effectively, promoting thorough analysis and informed decision-making throughout the project lifecycle.

B. *User QA Mode*

In this mode, the script begins by accepting input from an Excel file, which is then converted into a YAML file for subsequent processing. The primary output of this mode is the QA audit script's results, which are presented in the form of output files such as QA_Audit_Report.html and QA_Audit_Report.xlsx.

Additionally, users receive the output of the terminal as the user_output.log file, providing a comprehensive overview of the script's execution. Furthermore, all other generated files, including logs and reports, are stored in a designated directory labeled "1_All_logs." This centralized storage ensures that DV engineers can easily access and review the generated data at their convenience.

Overall, this mode streamlines the QA audit process by efficiently processing input data, generating comprehensive reports, and organizing outputs for easy review and analysis by stakeholders.

C. QA Mode

In the third mode, the script functions similarly to the second mode, processing input from the Excel file and generating output files such as QA_Audit_Report.html and QA_Audit_Report.xlsx. However, this mode emphasizes detailed logging for quality assurance (QA) engineers, with comprehensive script descriptions and logging available in the QA_Audit_Top.log file. While the core functionality remains consistent with the second mode, the emphasis shifts to providing QA engineers with detailed insights into the script's execution process. This includes comprehensive logging of script actions, errors, and outputs, facilitating thorough analysis and troubleshooting by QA personnel.

Furthermore, the organizational structure of output files and directories remains consistent, ensuring easy access and review of generated data by QA engineers. The user_output.log file continues to capture terminal output, while all other generated files are stored in the designated directory "1_All_logs."

Overall, the third mode enhances the QA audit process by prioritizing detailed logging and script description, empowering QA engineers with the information needed to conduct comprehensive analysis and ensure the quality of software products.

D. QA Check-in Mode

In the fourth mode, the script maintains its core functionality, similar to the previous modes, with input processing from the Excel file and generation of output files like QA_Audit_Report.html and QA_Audit_Report.xlsx. However, in this mode, the emphasis shifts towards version control integration.

The script continues to provide detailed logging in the QA_Audit_Top.log file, ensuring QA engineers have comprehensive insights into script execution. Moreover, all files within the "1_All_logs" folder undergo scrutiny for version control integration. Specifically, the script checks all files within the "1_All_logs" directory into the Perforce version control system. This process ensures that all generated logs, reports, and outputs are securely stored and versioned within the project's version control repository. Additionally, the script creates a new label in Perforce, marking the current state of the project's files. This label serves as a snapshot of the project at a specific point in time, providing a reference for future development and audit purposes.

Overall, the fourth mode enhances the QA audit process by integrating version control functionality, ensuring that all generated files are securely stored, versioned, and labeled within the project's version control repository.

The script operates intelligently as shown in “Fig. 3” by soliciting input from the user for the desired tool version. It then retrieves the tool version from the ENV TB CONFIG file and compares it with the actual tool version launched via the run command. This multifaceted comparison ensures a thorough review of the environment configuration.

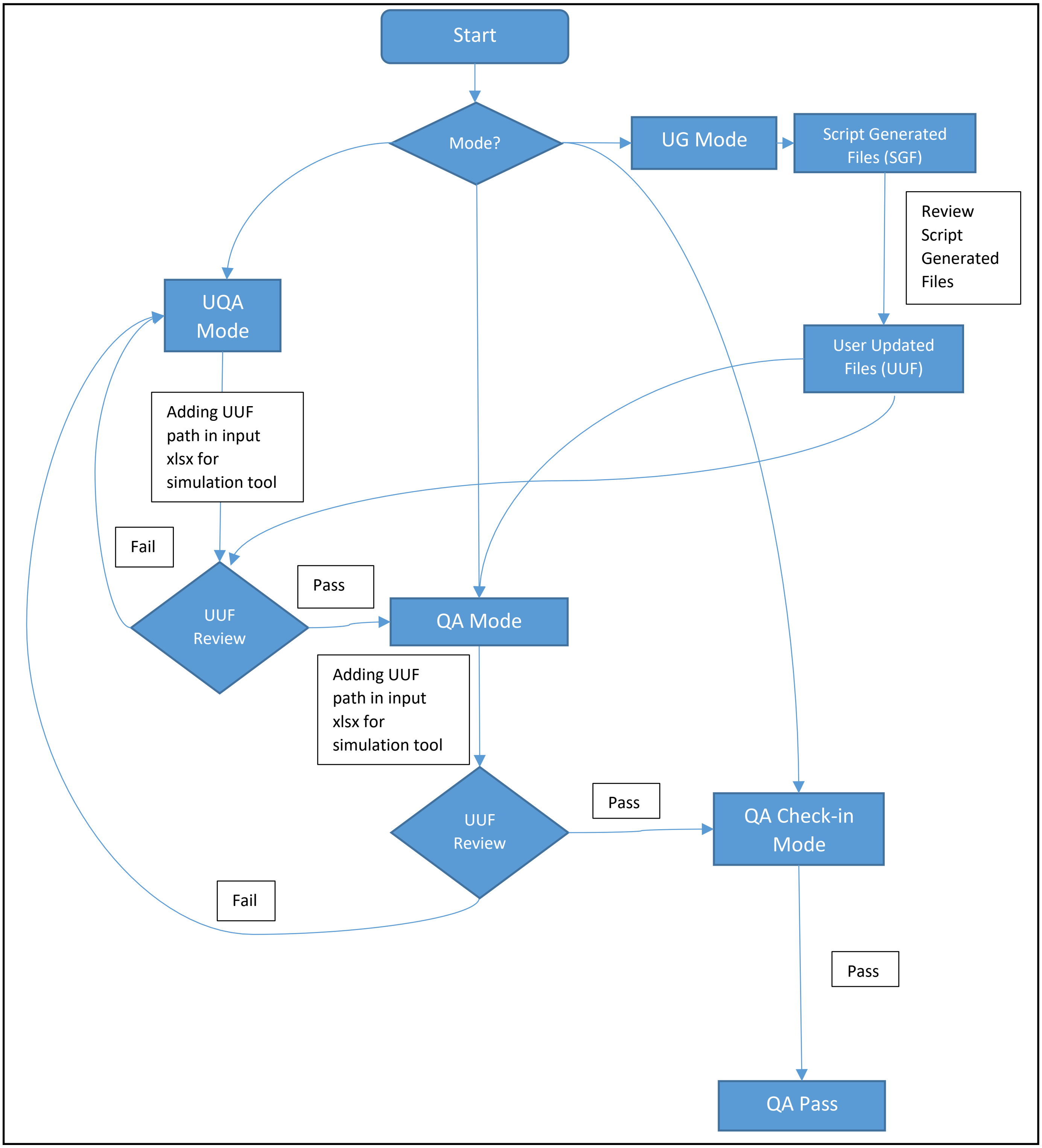


Fig 3: DV QA flow diagram

Upon completion of the environment review, the script evaluates the consistency of three critical aspects: the user-provided tool version, the version retrieved from the ENV TB CONFIG file, and the actual tool version launched. A successful review results in a pass, indicating alignment across all three aspects. The pass condition is met only when all three aspects exhibit congruence, indicating a harmonized environment configuration. If any discrepancy is detected among the tool versions, the review results in a fail status.

The script's validation process serves as a robust quality assurance measure, ensuring the integrity and consistency of the environment configuration. By meticulously comparing user input, environment configuration data, and actual runtime parameters, the script provides valuable insights into the environment's state. Passing the environment configuration review signifies that the environment is correctly configured, with all components aligned to the specified tool version. This ensures optimal functionality and reliability for subsequent processes and operations reliant on the environment configuration.

In summary, the script's intelligent evaluation of tool versions from multiple sources, coupled with its stringent pass/fail criteria based on alignment across all aspects, underscores its effectiveness as a quality assurance tool. By providing clear pass/fail outcomes, the script enables users to confidently proceed with operations, secure in the knowledge that the environment is accurately configured and ready for use as shown in “Fig. 4”.

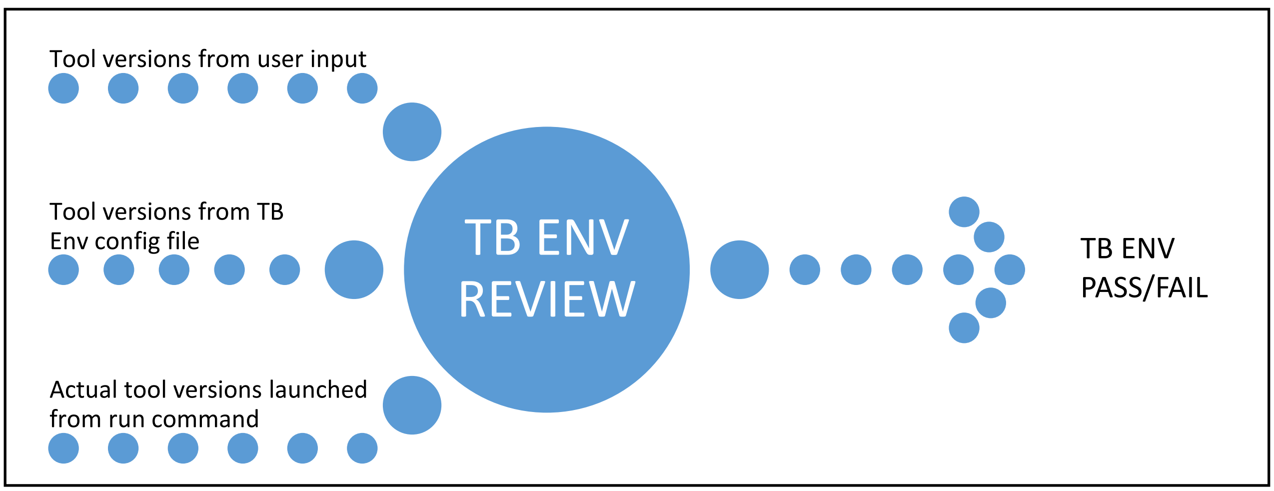


Fig 4: I/O’s of DV environment review

As soon as the compile and run command runs the label will be configured and once its configured then make a DV label and all of this will happen with the reproducibility of regression sanity as shown in “Fig. 5”.

Upon execution of the compile and run commands, the script initiates the labeling process, ensuring the reproducibility of regression sanity. Initially, the script configures the label, marking the current state of the project. This label encompasses all relevant files and configurations associated with the compile and run processes. Once the label is configured, signifying the completion of the setup phase, the script proceeds to create a DV label. This DV label serves as a snapshot of the project's state post-configuration, specifically tailored for regression sanity testing purposes.

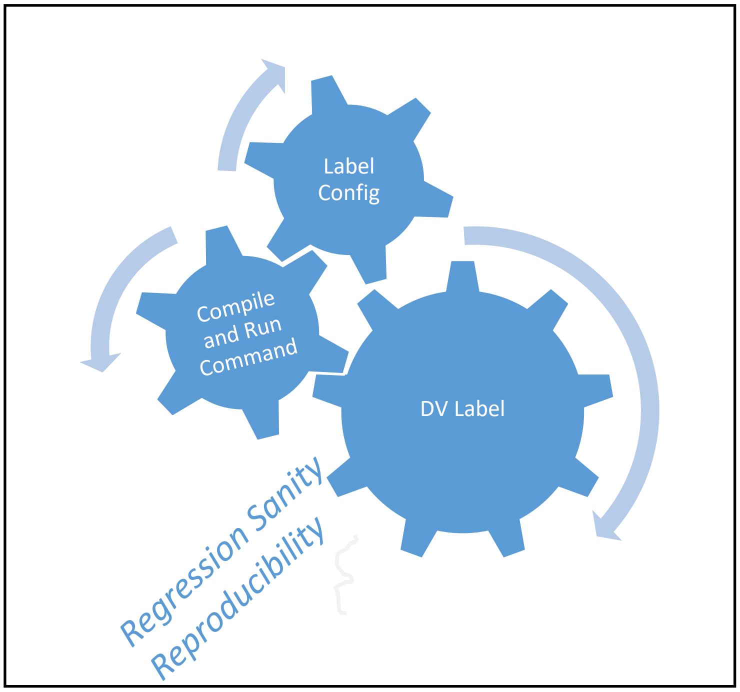


Fig 5: Importance of Labelling

The labeling process guarantees the reproducibility of regression sanity, as it captures the exact state of the project at key stages: pre and post-configuration. By encapsulating all pertinent files and configurations within the labels, the script facilitates consistent and reliable testing across regression iterations. Moreover, the labeling process ensures traceability and accountability, as each label represents a distinct point in the project's lifecycle. This enables seamless tracking of changes and facilitates debugging and analysis in the event of issues or discrepancies during regression testing.

Overall, by integrating labeling functionality into the compile and run commands, the script enhances the reproducibility and reliability of regression sanity testing. Through meticulous configuration and labeling, it provides a robust framework for consistently evaluating project integrity and functionality across multiple iterations.

II. RESULTS

The automation of quality assurance (QA) processes in semiconductor design projects yields outputs in the form of XLS and HTML files, providing users with comprehensive insights into the verification outcomes. These files encapsulate project information, review details, and assessments of various aspects of the design. Each aspect undergoes scrutiny based on predefined criteria, with scores and weightages assigned according to the pass or fail status and the availability of Register Transfer Level (RTL) or Gate Level Simulation (GLS) data.

The final report, along with the input XLS path, offers users an overall status overview. Should users opt to skip certain reviews or checks, the output files highlight the omissions and warn users about the skipped checks' implications. Priority names of the reviews skipped are also provided for reference.

It is crucial to ensure that the QA weightage aligns with the scope defined for the project. Any disparities between the two would result in a failed QA status, signifying that some checks did not meet the specified criteria. In such instances, users are prompted to review the other output sheets or pages containing the reviews considered for QA checks.

Review checks encompass various aspects, including environment configuration, RTL/GLS Test bench (TB) configuration, RTL/GLS Central Regression (CR) database, Simulation tool warnings and errors, Jira reviews, Force reviews, Coverage reviews, GLS Environment reviews, and GLS violation reviews. Each review meticulously examines version matching, total test runs, the presence of warnings and errors in tests, any violations detected during testing, and any pending Jira issues relevant to the project. These reviews ensure thorough coverage of the design and help identify any areas requiring further attention or refinement.

Overall, the automated QA process enhances efficiency, accuracy, and reliability in semiconductor design projects. By providing detailed insights into the verification outcomes and highlighting areas of concern or improvement, these automated systems empower users to make informed decisions and optimize the quality and reliability of semiconductor designs. Through rigorous review processes and comprehensive assessments, automated QA systems play a pivotal role in ensuring the integrity and success of semiconductor design projects.

III. CONCLUSION

The advent of automation has revolutionized the semiconductor industry, particularly in the domain of quality assurance (QA) processes. Traditionally, the verification of chip designs was a labour-intensive and time-consuming endeavour, often prone to human error. However, with the introduction of automated QA systems, this landscape has undergone a significant transformation.

These automated QA systems produce XLS and HTML files as output, providing users with detailed insights into the verification outcomes. Through meticulous scrutiny of various aspects of the design, including environment configuration, test bench setup, error detection, and coverage analysis, these systems ensure comprehensive assessment and validation of semiconductor designs. Each aspect undergoes rigorous evaluation based on predefined criteria, with scores and weightages assigned accordingly.

The generated output files not only summarize the verification results but also flag any skipped reviews or checks, alerting users to potential gaps in the assessment process. Moreover, discrepancies between the QA weightage and the project scope are promptly identified, ensuring alignment between evaluation criteria and project objectives.

In conclusion, the automation of QA processes in semiconductor design projects has significantly enhanced efficiency, accuracy, and reliability. By providing users with comprehensive insights into the verification outcomes and highlighting areas of concern or improvement, these automated systems empower stakeholders to make informed decisions and optimize the quality and reliability of semiconductor designs. Through robust review processes and comprehensive assessments, automated QA systems play a pivotal role in ensuring the integrity and success of semiconductor design projects in today's fast-paced and competitive semiconductor industry landscape.

ACKNOWLEDGMENT

We would like to express our deepest gratitude to all those who have contributed to the successful completion of this research paper. We would like to thank our mentors Avneesh Singh Verma and Sriram Kazhiyur Soundarrajan for their invaluable guidance, support, and encouragement throughout the process. Their insights and expertise have greatly enhanced the quality of our work. We are also thankful to our colleagues and friends who provided constructive feedback and assistance during the various stages of this research. Their suggestions have been instrumental in refining our ideas and improving the overall quality of the paper.

REFERENCES

[1] Cadence Design Systems, Inc., "Xcelium Parallel Logic Simulation," Xcelium User Guide, San Jose, CA, 2020.

[2] Synopsys, Inc., "Verdi Automated Debug System," Verdi User Manual, Mountain View, CA, 2021.

[3] Comprehensive Perl Archive Network (CPAN), "Data::Dumper," version 2.174, [Online]. Available: https://metacpan.org/pod/Data::Dumper. Accessed: May. 17, 2024.