2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Expanding Verification Horizons: OOPs-Enhanced Script-Driven Verification using Auto PSS Gen Utility (APGU)

Ashutosh Bisht
Senior Verification Engineer
MDRF/GPM, STMicroelectronics
Greater Noida, India
ashutosh.bisht@st.com

Manvendra Singh
Project Verification Lead Engineer
MDRF/GPM, STMicroelectronics
Greater Noida, India
manvendra.singh@st.com

***Abstract*-The architecture and design of embedded microcontroller units (MCUs) involve numerous peripherals and use cases, necessitating thorough verification of system IPs to ensure reliability and functionality. This verification is essential for managing enablement, reset, clock, and power functionalities at both the IP and System-on-Chip (SoC) levels.**

This paper introduces an automated methodology using the Automated PSS Generation Utility (APGU) for verifying System IPs and porting them to SoC architectures. APGU supports Object-Oriented Programming (OOP) principles, enabling modular and reusable test scenarios that are easily adaptable. Additionally, the Portable Stimulus Standard (PSS) facilitates the automatic generation of C-based tests, streamlining the verification process and ensuring consistency. By leveraging portability and automation tools, verification teams can achieve higher efficiency, scalability, and productivity, leading to more robust and reliable MCU designs.

I. Introduction

Verifying the System IPs of complex microcontrollers presents a significant challenge due to the multitude of scenarios that need to be coded and the numerous permutations involved in the enablement and selection of clock and reset sources. Performing this manually is a daunting task, requiring thorough verification at both the IP level and the SoC level. Additionally, porting these IPs to different SoCs and reusing the same verification environment across various SoCs adds another layer of complexity.

To address these challenges, we present a transformative script-driven approach (APGU) to hardware verification within the Portable Stimulus Standard (PSS) ecosystem. While PSS provides a structured foundation for test environments, the traditional scenario and model generation remain manual and inefficient. Our method automates these processes, offering a significant leap in efficiency and scalability by leveraging the power of Python and SQL.

We utilized Python for the development of PSS functions and models, enabling more dynamic and flexible test scenarios. Python's robust libraries and ease of use allowed us to create sophisticated scripts that automate various aspects of the verification process. Furthermore, we employed SQL to extract register-level information, which is crucial for the selection and enablement of peripheral IPs' clock and reset functionality. Additionally, we incorporated randomization in our PSS scenarios to cover all design aspects. Directed test methods would not have provided full coverage within the specified time frame. Randomization ensures extensive test and functional coverage, uncovering a wider range of potential issues and leading to a more thorough and effective verification process. By integrating SQL, we ensured that our verification process is both precise and efficient, as it allows for the accurate retrieval and manipulation of hardware configuration data.

II. METHODOLOGY

The technical contribution of this paper is a script-driven framework that revolutionizes model creation and test case generation in hardware verification at both the IP and SoC levels.

Detailed Flow:

1. XML: XML has all the register information about the internals of clocking and reset blocks also peripheral IP clock and reset programming. This detailed information is crucial for building accurate and comprehensive PSS models related to clock and reset functionalities. By leveraging XML, we ensure that all necessary register configurations are included, which helps in creating precise test scenarios.

2. XLSX Sheet: The XLSX sheet contains all the scenarios where selections from various sources of clocks, whether internal oscillators, external oscillators, or other IPs, are specified. This sheet serves as a comprehensive source of test scenarios, ensuring that all possible configurations are considered. The usage of randomization in these scenarios ensures that all potential configurations are tested, covering a wide range of possibilities and eliminating scenarios that might get missed if done manually.

3. Script-Driven Framework: For model and function creation, we have used Python and SQL to extract information from the Clock and Reset XLSX sheet. Python's robust libraries and ease of use allowed us to create sophisticated scripts that automate the generation of PSS models. SQL was employed to extract and manipulate the necessary data from the XLSX sheet, ensuring that the generated models are accurate and comprehensive. This automation transforms the traditional manual process into an efficient and scalable solution, significantly reducing the time and effort required.

4. Test Case Generation: Automated scripts generate test cases from the PSS models, ensuring comprehensive coverage and alignment with verification requirements. The usage of randomization ensures that all scenarios are covered and eliminates scenarios that might get missed if done manually. This approach guarantees extensive test and functional coverage, uncovering a wider range of potential issues and leading to a more thorough and effective verification process.

5. Verification Process: The generated test cases are used in the verification process, streamlining the overall work flow and reducing the time and effort required for thorough verification. By automating the test case generation and execution, we ensure that the verification process is both precise and efficient. This approach allows verification teams to focus on analyzing results and identifying issues, rather than spending time on manual test case creation and execution. This methodology ensures a robust, scalable, and efficient approach to hardware verification, addressing the complexities and challenges associated with verifying System IPs of complex microcontrollers.

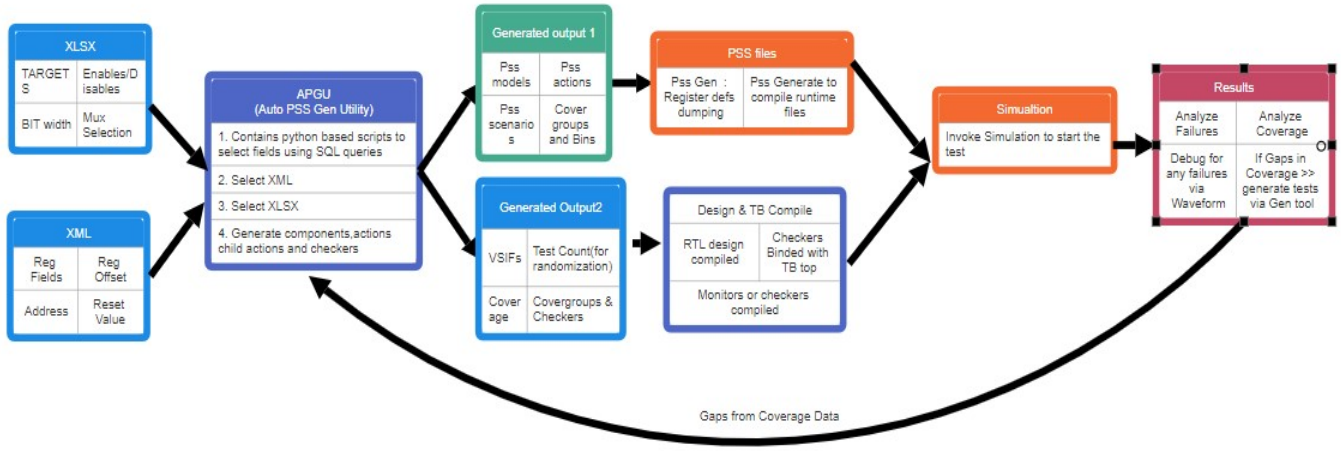

Figure 1 APGU Flow uses scripts to generate PSS Models

III. Preliminary Results

Preliminary results from the application of our method indicate a substantial improvement in verification efficiency and effectiveness. The integration of script-driven models with OOPs enhancements has led to a reduction in testbench development time and an increase in coverage and defect detection rates. These improvements are significant when compared to traditional verification methods, which often struggle with scalability and adaptability.

1. Reduction in Testbench Development Time: By automating the generation of PSS models and test cases, our method has significantly reduced the time required to develop testbenches. Traditional methods often involve manual coding and configuration, which can be time-consuming and error-prone. In contrast, our script-driven approach streamlines this process, allowing verification teams to quickly generate and modify testbenches as needed. Specifically, we have observed a reduction in testbench development time from 2 months to 3 weeks.

2. Increased Coverage: The use of randomization in our scenarios ensures that a wide range of potential configurations and edge cases are tested. This comprehensive coverage is difficult to achieve with directed test methods, which may miss certain scenarios. Our approach guarantees that all possible configurations are considered, leading to more thorough verification. Coverage improved from almost reaching 75% to 96% even in the initial runs.

3. Higher Defect Detection Rates: The increased coverage provided by our method has led to higher defect detection rates. By testing a broader range of scenarios, our approach is more likely to uncover hidden issues that might be missed by traditional methods. This results in more reliable and robust hardware designs.

Effort	NB Tests	Development	Maintenance	Nb tests /Day
Manual	40	40 days	1 week	1
PSS via script	~500	3 weeks	2 days	~25

Figure 2 manual vs script-based approach comparisons

IV. CONCLUSIONS

1. Script-Driven and OOPs-Enhanced Approach: The paper emphasizes the significant advantages of combining script-driven methodologies with object-oriented programming enhancements for testbench generation. This approach not only streamlines the process but also introduces a level of structure and reusability that is difficult to achieve with traditional methods.
2. Flexible Verification Tool: The integration of XML and XLSX as inputs, along with OOPs inheritance and randomization, creates a versatile and powerful verification tool. This flexibility allows for the creation of precise and customized test scenarios, ensuring comprehensive coverage and alignment with industrial application needs.
3. Easier Porting: One of the standout benefits of our method is the ease of porting from IP to SoC level. The script-driven framework and the use of standardized inputs make it possible to adapt verification environments across different SoCs within a few weeks. This significantly reduces the time and effort required for re-verification, promoting reusability and consistency across projects.
4. Meeting Modern Challenges: The proposed method effectively meets the demands of contemporary verification challenges, pushing the boundaries of what is currently achievable. By automating the generation of PSS models and test cases, our approach addresses the scalability and adaptability issues that plague traditional verification methods.
5. Setting New Industry Standards: By addressing key pain points in verification, the approach sets new standards for the industry, promoting a shift in established practices. The empirical data collected from industrial applications confirms the superiority of our method, highlighting its potential to set a new benchmark in hardware verification practices.
6. Automation and Adaptability: The findings suggest a future for hardware verification that leans heavily towards increased automation, adaptability, and overall efficiency. The script-driven framework not only reduces manual effort but also ensures that all possible configurations are considered, leading to more thorough and effective verification.
7. Implications for Future Strategies: The paper highlights the profound implications that this method has for the evolution of verification strategies, indicating a clear path for future advancements. The success of our approach suggests that future verification methodologies will likely incorporate similar levels of automation and flexibility, further enhancing the efficiency and effectiveness of hardware verification processes.

V. ACKNOWLEDGMENT

We would like to thank Gnaneshwara Tatuskar and Ushadevi M N from Cadence for there immense help and guidance.

VI. REFERENCES

[1] Henry Shelly, and Regmi Nirabh, “How To Close Coverage 10X Faster Using Portable Stimulus Standard – A Case Study,”

[2] Cadence Perspec System Verifier User Guide