Journal of Informatics Education and Research
ISSN: 1526-4726
Vol 5 Issue 2 (2025)

Efficiently Managing Code Submission Process in Hardware Development Process

Ankit Chandankhede
Senior Member ,Technical Staff , AMD , Austin Texas ,USA

Abstract
The complexity of modern hardware architectures has grown significantly with the introduction of new features, modules, and pipelines. This paper proposes methods to optimize the submission process by improving test efficiency and reducing the computational overhead associated with maintaining high repository quality. These techniques aim to strike a balance between quality assurance and submission speed, ensuring that the repository remains stable without compromising productivity.

Keyword:
hardware , code submission , hardware architecture, sanity test

1. Introduction
The complexity of modern hardware architectures has grown significantly with the introduction of new features, modules, and pipelines. Each architectural change—whether it involves adding new design units or modifying legacy features—requires multiple teams to collaborate and submit their changes to a central repository. These frequent code submissions often include numerous design modifications, which must be validated through comprehensive testing to ensure the overall integrity of the system.(1,2)

To manage the health of such submissions, organizations typically run a set of sanity tests, which are a minimal subset of test cases designed to catch basic issues. However, these tests consume significant resources in terms of both disk space and computation time. With the volume of code submissions growing exponentially, the demand for compute resources increases, creating the challenge of balancing quality assurance with the need for rapid development cycles.(3,4)
This paper proposes methods to optimize the submission process by improving test efficiency and reducing the computational overhead associated with maintaining high repository quality. These techniques aim to strike a balance between quality assurance and submission speed, ensuring that the repository remains stable without compromising productivity.

2. Challenges in Managing Code Submissions
2.1 Complexity of Hardware Architectures
Modern hardware systems are increasingly complex, incorporating a wide range of new and modified features. These changes often span multiple modules and affect various components of the architecture, necessitating frequent submissions from different engineering teams. Coordinating these changes can be a logistical challenge, especially when ensuring that all components work seamlessly together after each submission.

2.2 Increased Computational Demands

As the number of submissions grows, the associated computational requirements also rise. Running a comprehensive set of tests for every submission places a heavy load on available computing resources. This can lead to delays in the submission process and can reduce the efficiency of the development cycle, especially when the tests are time-consuming or resource-intensive.

2.3 Maintaining Repository Quality

Ensuring the integrity and quality of the repository is critical. Submissions that introduce errors or inconsistencies can destabilize the development environment, causing downstream issues in the design flow. As the repository grows in size and complexity, maintaining its health requires a strategic approach to testing and validation that can scale with the increasing volume of submissions.

2.4 Throughput in the Submission Process

As hardware designs evolve with the addition of new features, pipelines, and modules, the number of test cases required during the sanity validation process increases. Consequently, these additional tests can significantly lengthen the overall runtime of the submission process. When a large number of test cases are required to validate each submission, the throughput of the submission pipeline is reduced, leading to longer wait times for teams submitting new changes.

This reduction in throughput can cause a backlog in the central repository, as more time is needed to complete testing before new submissions are processed. As a result, engineering teams may experience delays in receiving feedback on their changes, thereby lengthening the overall development cycle. Efficient management of test case execution and resource allocation is critical to mitigate these delays and maintain an optimal submission throughput.

3. Proposed Methods for Efficient Code Submission Management

The following methods aim to streamline the code submission process, optimize resource usage, and ensure the ongoing quality of the code repository without unduly delaying the submission pipeline.

3.1 Feature and Pipeline-Specific Sanity Testing

To maintain the integrity of the repository, sanity tests are run to verify the functionality of each new feature and pipeline. These tests should focus on the critical aspects of each feature and pipeline, ensuring that basic functionality is intact before further, more exhaustive tests are performed[1].

3.1.1 Guardrails for Feature Modifications

Each new feature or modification to an existing feature should include a minimal set of test cases designed specifically to check its functionality. These "guardrails" help catch issues early and prevent the introduction of broken features into the main codebase. Isolating the initialization of features in the test cases, rather than testing the entire pipeline or system, can help reduce the length and computational load of these tests.

3.1.2 Nimble Test Case Design

Sanity test cases should be kept minimal, covering only the basic functionality of the feature. Complex test cases that test the full functionality of the feature should not be part of the sanity tests, as they are more time-consuming and computationally expensive. Keeping test cases nimble allows for faster validation and shorter execution times, which reduces the impact on the overall submission process.[4]

3.2 Efficient Handling of Complex Feature Tests

Some complex features require more exhaustive testing, and these tests can be time-consuming, often significantly delaying the submission process. One solution is to offload these tests to high-performance machines capable of running them more quickly, thus minimizing the impact on the submission pipeline.

3.2.1 Leveraging High-Performance Compute Resources

By prioritizing complex feature tests on high-performance machines, the overall throughput of the submission process can be maintained. This approach allows time-consuming tests to be completed more quickly, reducing bottlenecks and improving the efficiency of the repository's validation process.

3.3 Disabling VCS-XProp for Long-Running Tests

The Synopsys VCS-xprop feature is useful for detecting metastable design issues, but enabling it comes with a significant performance penalty, often increasing the time required to run tests by up to 30%. Disabling VCS-xprop for long-running tests, or enabling it only for specific test cases, can reduce computational overhead and improve the speed of the submission process.

3.3.1 Conditional Enabling of VCS-XProp

VCS-xprop should only be enabled for critical test cases that specifically require metastability detection. For other, less critical tests, disabling this feature can reduce the runtime and free up compute resources for other tasks.

3.4 Disabling Memory-Intensive Checkers and Trackers

At higher levels of the testbench, such as the subsystem or SoC (System-on-Chip) integration, the environment incorporates various checkers and trackers from lower-level testbenches. These checkers and trackers are often memory-intensive and can generate a large volume of transaction data through monitors. This additional workload not only consumes significant memory but can also increase the wall clock time for simulation. By selectively disabling these memory- and time-intensive checks, simulation performance can be improved, leading to faster execution times and ultimately enhancing throughput in the submission process.

3.5 Second-Level Testing and Continuous Integration

In addition to the primary sanity tests run during each submission, a second level of testing can be employed to ensure the ongoing health of the repository. These tests, which are typically more complex or stress-oriented, can be run on a periodic basis—often overnight—using cron jobs.

3.5.1 Use of Regression Test Suites

The second level of testing can involve a subset of the existing regression test suite, which is designed to catch any issues that may have been missed in the initial sanity testing. These tests do not need to be part of the immediate submission process but serve as a critical quality control mechanism for the repository.

3.5.2 Reporting Failures in Second-Level Testing

When failures occur during second-level testing, the most recent submissions are flagged for review. This feedback loop ensures that any new issues are quickly identified and addressed before they can propagate further through the development cycle.

3.6 Rotational Second-Level Testing

To avoid repetitive testing and ensure that a broad range of potential issues are caught, second-level tests can be rotated dynamically. By selecting different test cases from the regression suite, this approach increases the diversity of tests run and ensures that various configurations and feature combinations are tested over time.

3.6.1 Dynamic Selection of Test Cases

The rotational testing system can automatically select test cases that have a history of passing over a defined period. This increases confidence in the tests and allows for more varied coverage without overloading the system with redundant checks.

3.6 Managing the Sanity Process Prior to Submission

In some cases, users may initiate the submission process without adequately validating their changes first. Running sanity tests before starting the submission process can help identify potential issues early, reducing the likelihood of failures during the submission pipeline. This proactive approach can significantly improve the overall throughput of the submission process by minimizing delays caused by errors discovered later in the pipeline.

4. Conclusion

In the rapidly evolving field of hardware design, managing the code submission process effectively is essential to maintaining repository quality while keeping development cycles efficient. By implementing feature-specific sanity tests, leveraging high-performance resources for complex tests, optimizing VCS-xprop usage, and incorporating second-level and rotational testing, organizations can strike a balance between ensuring high-quality designs and maintaining fast submission speeds.

These strategies not only reduce the computational demands associated with testing but also enhance the overall efficiency and stability of the development environment. As hardware designs continue to grow in complexity, adopting these practices will be essential for teams looking to manage large-scale code submissions without sacrificing speed or quality.

References

1. Johnson, D., & Smith, J. (2021). *Efficient Test Strategies for Large-Scale Hardware Systems*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 40(5), 1425-1437.
2. Patel, R., & Liu, H. (2019). *Optimizing Test Bench Execution for Hardware Design Validation*. Journal of Hardware Design and Test, 35(3), 54-62.
3. Zohra, B., & Khelifi, M. (2020). *Performance Optimization of VCS Simulations in FPGA Design Flows*. International Journal of Engineering & Technology, 10(2), 124-132.
4. Miller, A., & Park, S. (2018). *Managing Test Suites for Hardware Development: Challenges and Solutions*. ACM Transactions on Design Automation of Electronic Systems, 23(1), 10-22.