УДК 004.415.5

МЕТОДИ ЗАСТОСУВАННЯ БЕЗПЕРЕРВНОЇ ІНТЕГРАЦІЇ У ПРОЦЕСІ РОЗРОБКИ ТА ВЕРИФІКАЦІЇ RTL-ОПИСІВ

Васильєв О.Ю., Філіппенко О.І.
oleksandr.vasyliev1@nure.ua, oleh.filippenko@nure.ua
Науковий керівник – доц., к.т.н., Філіппенко І.В.
Харківський національний університет радіоелектроніки, каф. АПОТ
м. Харків, Україна

The purpose of this work is to investigate the use of continuous integration infrastructure and tools in ASIC and FPGA development and verification. The study considers the integration of version control systems, automated testing pipelines, and static analysis tools to improve the reliability of RTL code. Particular attention is given to the Jenkins and GitHub platforms for automated execution of simulation tests, lint checks, and project consistency verification. The results demonstrate that combining continuous integration infrastructure with AI-assisted development tools can significantly improve design and verification efficiency and reduce the time required to detect errors.

Сучасна розробка цифрових інтегральних схем характеризується високим рівнем складності проєктів, значною кількістю взаємопов’язаних модулів та необхідністю постійного контролю коректності змін у коді. У таких умовах автоматизація процесів перевірки та інтеграції змін відіграє важливу роль. Одним із підходів, що широко застосовується у промисловій практиці, є безперервна інтеграція. Цей підхід передбачає автоматичне виконання перевірок після внесення змін у код, що дозволяє своєчасно виявляти помилки та запобігати їх накопиченню у проєкті. Метою роботи є дослідження застосування безперервної інтеграції у сфері розробки апаратного забезпечення. Такий підхід поєднує системи контролю версій, інструменти автоматизованого тестування та інфраструктуру запуску перевірок. Важливими компонентами такого середовища є платформа керування кодом GitHub, система автоматизації Jenkins та допоміжні інструменти аналізу коду.

Сучасний процес розробки RTL-описів, що виконуються мовами SystemVerilog або VHDL, можна організувати із використанням системи контролю версій Git. Репозиторій проєкту розміщується на платформі GitHub, що забезпечує централізоване зберігання коду та підтримку колективної роботи. Нові функціональні можливості або виправлення помилок розробляються у окремих гілках репозиторію. Після завершення роботи над змінами створюється запит на інтеграцію змін (pull request), який використовується для перевірки та обговорення запропонованих модифікацій перед їх інтеграцією у основну гілку проєкту. Такий механізм дозволяє виконати попередній аналіз змін, провести перевірку коду та оцінити відповідність реалізації вимогам проєкту.

Після створення або оновлення запиту на інтеграцію змін система GitHub ініціює запуск автоматичних перевірок у середовищі безперервної інтеграції Jenkins. У межах цього процесу формується послідовність автоматизованих перевірок. На початковому етапі виконується компіляція RTL-коду, що дозволяє перевірити синтаксичну коректність опису модулів та правильність структури проєкту. Далі запускаються тести моделювання, які виконуються у середовищі безперервної інтеграції з використанням testbench. Такі тести дозволяють перевірити функціональну поведінку модулів та переконатися, що внесені зміни не призводять до появи регресійних помилок. Використання автоматичних регресійних тестів є важливою складовою процесу верифікації, оскільки дозволяє повторно перевіряти значну кількість сценаріїв після кожної зміни коду.

Крім функціонального тестування, важливу роль у процесі перевірки відіграє статичний аналіз RTL-описів. Для цього використовуються лінтери, які аналізують код без виконання симуляції та перевіряють його відповідність прийнятим стандартам розробки. Статичний аналіз дозволяє виявляти потенційно небезпечні конструкції, некоректне використання сигналів або неповні описання комбінаційної логіки. Подібні перевірки сприяють підвищенню якості коду та зменшують імовірність появи помилок на пізніх етапах розробки. Додатковим елементом автоматизованої перевірки є контроль цілісності структури проєкту. Такі перевірки спрямовані на забезпечення узгодженості між різними компонентами та рівнями системи, а також коректності конфігурації проєкту. Зокрема перевіряється відповідність назв модулів назвам файлів, правильність структури каталогів та наявність необхідних конфігураційних даних. Подібні механізми дозволяють уникнути ситуацій, коли проєкт компілюється лише у певному середовищі або залежить від локальних змін розробника. У великих проєктах із великою кількістю учасників такі перевірки є важливою складовою підтримання стабільності системи.

Результати виконання всіх перевірок автоматично передаються у систему керування змінами. У випадку виникнення помилки результати перевірок автоматично відображаються у запиті на інтеграцію змін, що дозволяє розробнику оперативно усунути проблему. Лише після успішного завершення всіх етапів перевірки зміни можуть бути інтегровані у основну гілку репозиторію. Такий підхід забезпечує контроль якості коду та дозволяє підтримувати стабільність проєкту протягом усього циклу розробки.

У сучасних проєктах розробки апаратного забезпечення також активно використовуються інструменти штучного інтелекту (ШІ), які допомагають розробникам аналізувати код та прискорювати виконання типових задач. Такі інструменти можуть застосовуватися для генерації фрагментів RTL-описів, пояснення складних ділянок коду або формування додаткових тестових сценаріїв. Використання ШІ-інструментів дозволяє зменшити час розробки та підвищити продуктивність інженерів. При цьому ШІ-

інструменти виступають допоміжним засобом і не замінюють традиційні методи верифікації, які базуються на автоматизованому тестуванні та моделюванні поведінки системи.

Застосування інтегрованого підходу, що поєднує систему контролю версій GitHub, інфраструктуру безперервної інтеграції Jenkins та сучасні ШІ-інструменти, дозволяє значно підвищити ефективність процесу розробки цифрових інтегральних схем. Автоматизація перевірок забезпечує раннє виявлення помилок, підвищує якість RTL-описів та зменшує витрати часу на ручний аналіз коду. У результаті формується стабільне середовище розробки, яке дозволяє підтримувати надійність і масштабованість складних апаратних систем. Це стає особливо важливим при збільшенні кількості розробників, які одночасно працюють над проєктом. Саме такий підхід дозволяє поєднати надійність та швидкість розробки в сучасній передовій технологічній галузі апаратної розробки.

Таким чином, автоматизація процесів розробки та верифікації цифрових інтегральних схем є важливим чинником підвищення ефективності сучасних інженерних команд. Використання систем безперервної інтеграції дозволяє виконувати автоматичну перевірку змін у коді, включаючи компіляцію, моделювання, статичний аналіз та перевірки цілісності проєкту. Інтеграція таких інструментів, як GitHub та Jenkins, забезпечує раннє виявлення помилок та підтримує стабільність проєкту протягом усього циклу розробки. Додаткове застосування ШІ-інструментів сприяє прискоренню аналізу коду та формуванню тестових сценаріїв. Поєднання інфраструктури безперервної інтеграції та ШІ-інструментів дозволяє підвищити якість RTL-описів і скоротити час верифікації складних цифрових систем.

Список використаних джерел:

1. Duvall P., Matyas S., Glover A. Continuous Integration: Improving Software Quality and Reducing Risk. Boston : Addison-Wesley Professional, 2007. 336 p.
2. Dranga D., Dumitrescu C. Artificial Intelligence Application in the Field of Functional Verification // Electronics. 2024. Vol. 13, № 12. P. 2361. DOI: https://doi.org/10.3390/electronics13122361
3. Revolutionizing Semiconductor Design and Manufacturing with AI // Journal of Knowledge Learning and Science Technology. 2024. Vol. 3, № 3. P. 272–277. DOI: https://doi.org/10.60087/jklst.vol3.n3.p.272-277
4. Generative AI for Semiconductor Design // Amazon Web Services. 2024. URL: https://aws.amazon.com/blogs/industries/generative-ai-for-semiconductor-design/ (дата звернення: 06.03.2026).
5. Wilson G. et al. Best Practices for Scientific Computing // PLOS Biology. 2014. Vol. 12, № 1. DOI: https://doi.org/10.1371/journal.pbio.1001745