**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Національний університет «Запорізька політехніка»**

Факультет комп'ютерних наук та технологій
Кафедра «Комп'ютерні системи та мережі»

**КОТЕЛЕВСЬКИЙ ДАНИЛО СЕРГІЙОВИЧ**
Група КНТ-514м

# КОНСУЛЬТАТИВНА СИСТЕМА ДЛЯ ВЕРИФІКАЦІЇ КОДУ НА VHDL ІЗ ВИКОРИСТАННЯМ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ

**АВТОРЕФЕРАТ**

дипломної роботи на здобуття освітньо-кваліфікаційного рівня
«магістр» 123 Комп'ютерна інженерія
освітньої програми Комп'ютерні системи та мережі

2025 р.

Дипломна робота є рукопис.

Робота виконана в національному університеті «Запорізька політехніка», на кафедрі комп’ютерних систем та мереж

**Керівник** кандидат технічних наук, доцент
**Зеленьова Ірина Яківна,**
Національний університет «Запорізька політехніка», доцент кафедри комп'ютерних систем та мереж

**Офіційний рецензент:** **Щербаков Володимир Іванович,**
Директор
НВФ «Бізнес Комп’ютер Сервіс»

Захист відбудеться "26" грудня 2025 р.

Секретар екзаменаційної комісії, доцент кафедри комп’ютерних систем та мереж Т.В. Голуб

# ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

**Актуальність теми.** Сучасний етап розвитку цифрової електроніки характеризується стрімким зростанням складності проєктів на базі програмованих логічних інтегральних схем (FPGA). Перехід до архітектур типу "система на кристалі" (SoC), що інтегрують мільйони логічних елементів, створив фундаментальну проблему галузі – так званий "розрив у верифікації" (verification gap). Суть цього явища полягає у тому, що логічна ємність мікросхем зростає значно швидше, ніж продуктивність праці розробників та ефективність традиційних засобів перевірки коду.

Згідно з дослідженням Wilson Research Group Functional Verification Study, проведеним компанією Siemens EDA у 2022 році, лише 16% усіх FPGA-проєктів змогли досягти стадії виробництва без жодної пропущеної помилки. Це свідчить про те, що традиційні методи ручної перевірки та написання тестбенчів часто виявляються недостатньо ефективними для покриття всіх можливих сценаріїв роботи пристрою.

Особливої актуальності ця проблема набуває в контексті використання мови опису апаратури VHDL. Будучи мовою із суворою типізацією та складним синтаксисом, VHDL створює високий поріг входження для початківців та студентів: значна частина часу витрачається не на реалізацію алгоритмів, а на боротьбу із синтаксичними конструкціями, типами даних та сигналами.

Паралельно з цим, останні роки характеризуються революційним проривом у галузі штучного інтелекту. Великі мовні моделі (Large Language Models, LLM), такі як GPT, Claude та Gemini, продемонстрували вражаючі здібності до аналізу програмного коду та генерації технічних рішень. Ці моделі, навчені на мільйонах репозиторіїв коду різними мовами програмування, засвоїли синтаксичні правила та семантичні патерни, включаючи мови опису апаратури.

Однак, попри значний потенціал LLM як інтелектуальних асистентів для верифікації коду, наразі відсутні систематичні дослідження їх ефективності саме для задач аналізу VHDL. Ринок пропонує широкий спектр моделей, що відрізняються точністю,

швидкістю та вартістю використання, але немає чітких даних щодо того, яка саме модель є найбільш придатною для специфічної задачі виявлення помилок апаратної логіки.

Таким чином, актуальною і важливою є розробка спеціалізованої консультативної системи на базі штучного інтелекту, яка б не лише автоматизувала процес верифікації VHDL-коду, але й базувалася на науково обґрунтованому виборі оптимальної AI-моделі. Створення такої системи дозволить знизити бар'єр входу для нових спеціалістів, підвищити якість навчального процесу та скоротити час розробки FPGA-проєктів.

**Мета і завдання дослідження.** Метою магістерської роботи є підвищення якості та прискорення процесу верифікації FPGA-проєктів шляхом розробки та дослідження спеціалізованої консультативної системи на базі засобів штучного інтелекту.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Розробити комплексну методику порівняльного аналізу ефективності сучасних великих мовних моделей (LLM) у задачах ідентифікації синтаксичних, логічних та стилістичних дефектів у VHDL-коді;

2. Провести експериментальне дослідження дев'яти сучасних AI-моделей (сімейств GPT, Claude та Gemini) для визначення оптимального інструменту за критеріями точності, швидкодії та вартості експлуатації;

3. Спроєктувати та програмно реалізувати клієнт-серверну архітектуру вебсистеми, що забезпечує стабільну взаємодію користувача з обраною AI-моделлю;

4. Розробити функціональні модулі статичного аналізу коду з інтелектуальним поясненням помилок та автоматичної генерації верифікаційного оточення;

5. Провести тестування розробленої системи на реальних сценаріях та надати практичні рекомендації щодо її впровадження в освітній та виробничий процеси.

**Об’єкт дослідження** – процес автоматизованої верифікації коду на мові опису апаратури VHDL.

**Предмет дослідження** – методи та засоби підвищення ефективності аналізу VHDL-коду з використанням великих мовних моделей.

**Методи дослідження.** Для вирішення поставлених завдань використано: методи теоретичного аналізу архітектур трансформерних нейронних мереж; порівняльний експеримент для оцінки ефективності LLM; методи об'єктно-орієнтованого проєктування та розробки сучасних вебсистем (SPA).

**Наукова новизна отриманих результатів** полягає у розробці методики автоматизованого порівняльного аналізу ефективності великих мовних моделей для задач верифікації апаратної логіки, яка, на відміну від існуючих підходів, застосовує семантичну нормалізацію відповідей за допомогою моделі-арбітра. Це дозволило вперше систематизувати кількісні показники точності та надійності сучасних LLM при роботі з VHDL-кодом та експериментально обґрунтувати доцільність використання гібридного підходу в консультативних системах.

**Практичне значення отриманих результатів.** Розроблено повнофункціональну вебсистему, яка дозволяє суттєво скоротити час на налагодження проєктів та написання тестбенчів. Система забезпечує візуалізацію помилок та надання навчальних порад, що дозволяє використовувати її як ефективний допоміжний інструмент у підготовці фахівців з комп'ютерної інженерії.

**Апробація результатів дипломної роботи.** Основні положення дипломної роботи та результати досліджень подано до участі на конференції:

– Modern Trends In Information Technology Development у секції «Artificial Intelligence» (19 листопада, Дніпро).

**Структура та обсяг роботи.** Робота складається зі вступу, п'яти розділів, висновків, переліку джерел та додатків. Пояснювальна записка містить 95 сторінок тексту, 27 рисунків, 5 лістингів та 3 таблиці.

## ОСНОВНИЙ ЗМІСТ РОБОТИ

**У першому розділі** проведено аналіз сучасного стану верифікації FPGA-проєктів. Виявлено проблему «розриву у верифікації» (verification gap), за якої зростання складності мікросхем випереджає можливості традиційних засобів перевірки. Встановлено, що існуючі інструменти мають високий поріг

входження, а ручний пошук помилок у VHDL-коді є трудомістким через специфіку мови.

Обґрунтовано перспективність використання великих мовних моделей (LLM) як інтелектуальних асистентів. Проаналізовано роль промпт-інжинірингу для підвищення точності відповідей та нівелювання «галюцинацій» моделей. Запропоновано власну класифікацію LLM за критерієм продуктивності: флагманські («важкі»), збалансовані та високоефективні («легкі») .

Сформульовано вимоги до консультативної системи: реалізація у вигляді вебзастосунку (SPA), підтримка інтерактивного аналізу з поясненнями та генерація верифікаційного оточення. Розроблено структурну схему функціональних модулів системи (рис. 1).

Обґрунтовано вибір стека технологій для розробки (Next.js, TypeScript, Material UI, Zustand), що забезпечує надійність, масштабованість та відповідність сучасним стандартам вебпроєктування.

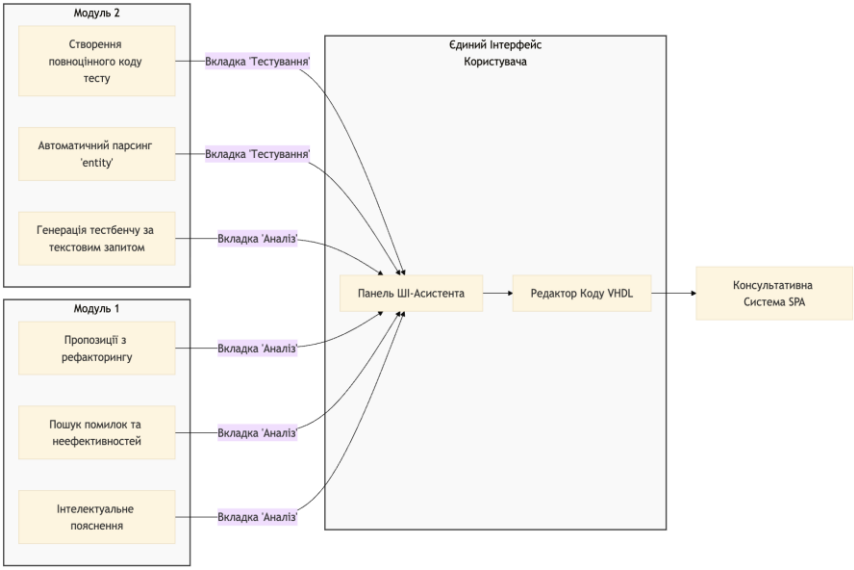


Рисунок 1 – Функціональні модулі системи

**У другому розділі** розроблено та апробовано комплексну методику порівняльного аналізу дев'яти сучасних великих мовних моделей (LLM) сімейств GPT, Claude та Gemini. Метою дослідження стало обґрунтування вибору оптимальної моделі для інтеграції у консультативну систему.

Для забезпечення об'єктивності експерименту сформовано спеціалізований датасет із 25 тестових кейсів, що включає файли з синтаксичними, логічними, стилістичними помилками, проблемами синтезу, а також еталонні («чисті») файли для перевірки на хибні спрацювання.

Наукова новизна підходу полягає у впровадженні етапу семантичної нормалізації результатів із використанням «моделі-арбітра» (Claude Sonnet 4.5). Це дозволило перетворити різнорідні текстові відповіді нейромереж на структуровані дані для кількісного порівняння з еталоном (Ground Truth). Алгоритм методики представлено на рисунку 2.

Рисунок 2 – Блок-схема методики порівняльного аналізу

Експериментально встановлено:

- Claude Opus 4.1 є лідером за загальною ефективністю (F1-Score 73.81%) та надійністю (0 хибних спрацювань);
- Claude Haiku 4.5 показала найкращі результати у виявленні складних логічних помилок (F1 89.33%) при найвищій швидкодії (6.40 с), що робить її оптимальною для інтерактивного режиму;
- моделі Gemini є стратегічно важливими для реалізації безкоштовного доступу завдяки наявності Free Tier API;
- моделі сімейства GPT продемонстрували значно вищі показники часу обробки запитів порівняно з конкурентами (рис. 3), що є критичним показником для користувацького інтерфейсу та робить їх малопридатними для ролі оперативного інтерактивного асистента.

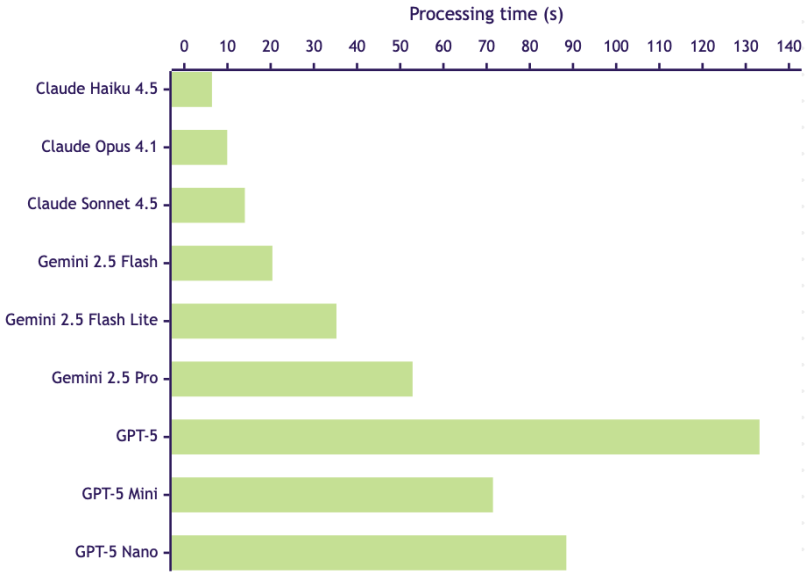


Рисунок 3 – Порівняння моделей за середнім часом обробки (с)

На основі отриманих даних обґрунтовано гібридний підхід до побудови системи: використання Claude Haiku 4.5 як основного рушія для швидкого інтерактивного аналізу та Claude Opus 4.1 для

режиму поглибленої фінальної верифікації. Також відзначено важливість моделей Gemini (Flash та Pro) для реалізації безкоштовного доступу до системи.

**У третьому розділі** обґрунтовано архітектуру та виконано програмну реалізацію консультативної системи. Для забезпечення гнучкості та масштабованості обрано класичну клієнт-серверну архітектуру на базі стека Next.js, TypeScript та бібліотеки Material UI.

Клієнтська частина реалізована як односторінковий застосунок (SPA), що забезпечує реактивність інтерфейсу та зручність роботи з редактором коду.

Серверна частина виконує роль захищеного API-шлюзу, приховуючи ключі доступу до зовнішніх сервісів. Архітектуру системи наведено на рисунку 4.

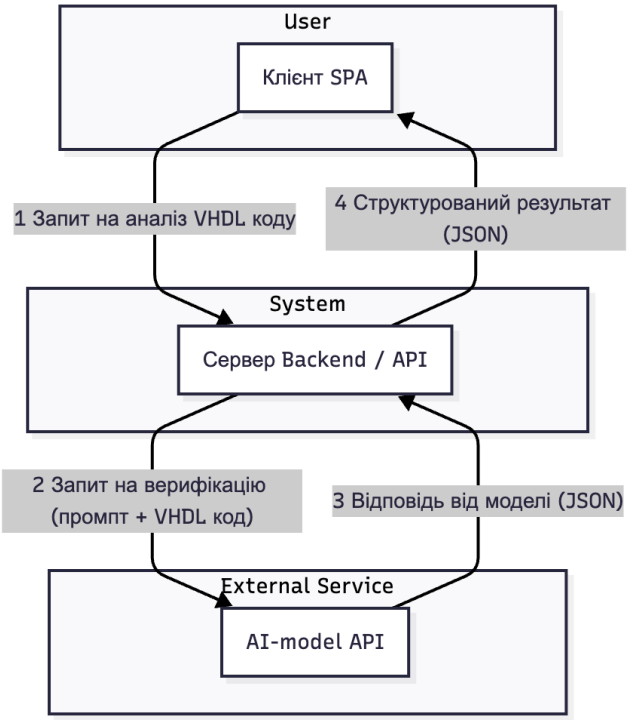

Рисунок 4 – Архітектура розроблюваної консультативної системи

Для уніфікації взаємодії з різними провайдерами штучного інтелекту (Anthropic, Google) у системі застосовано патерн проєктування «Абстрактна Фабрика» (Abstract Factory). Це дозволяє динамічно змінювати активну модель без втручання в основну бізнес-логіку застосунку.

Особливу увагу приділено стратегії промпт-інжинірингу. Для забезпечення стабільної роботи системи впроваджено сувору вимогу (JSON Enforcement) повертати відповідь виключно у структурованому форматі, що усуває необхідність складного парсингу тексту. Алгоритм обробки запиту та валідації даних зображено на рисунку 5.

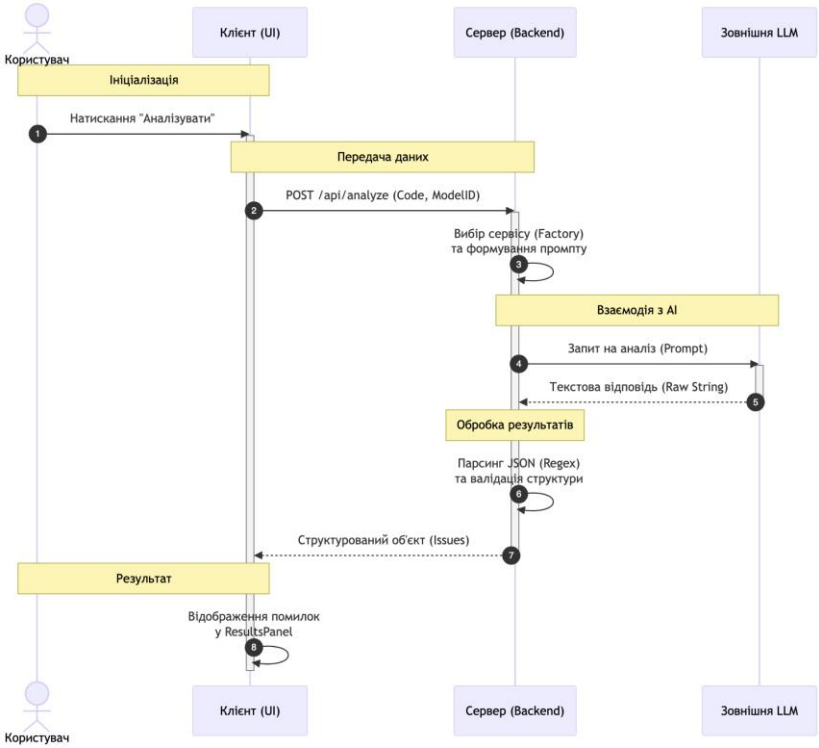


Рисунок 5 – Діаграма послідовності процесу аналізу коду

Важливим науково-технічним рішенням стало визначення англійської мови як єдиного стандарту для системних промптів. Аналіз ефективності токенізації показав, що кириличні символи кодуються менш ефективно, що призводить до зростання вартості запиту втричі (коефіцієнт витрат для української мови становить 3.0 відносно англійської). Використання англійської мови дозволило оптимізувати витрати ресурсів та підвищити точність термінології VHDL.

Функціонально система розділена на два модулі:

1. Модуль аналізу коду, який використовує регулярні вирази для екстракції JSON-об'єкта з відповіді нейромережі та групує помилки за рівнями критичності;

2. Модуль генерації верифікаційного оточення, який на основі текстового опису сценарію автоматично створює VHDL-код тестбенчу, готовий до симуляції.

**У четвертому розділі** викладено результати експериментальної перевірки функціональних можливостей та надійності розробленої системи. Тестування наскрізного потоку даних підтвердило коректність взаємодії клієнтської частини, серверного шлюзу та зовнішніх API.

Для верифікації базового функціоналу проведено модельний експеримент із виявлення синтаксичних дефектів у VHDL-коді. Система успішно ідентифікувала пропущені оператори та незакриті конструкції if, класифікувавши їх як критичні помилки, що унеможливлюють компіляцію. Результат роботи модуля аналізу наведено на рис. 6.

Особливу увагу приділено тестуванню механізмів багаторівневої валідації даних. Реалізовано стратегію захисту ресурсів API, яка блокує доступ до генерації тестбенчів до моменту успішного проходження аналізу. Це гарантує, що система не витрачатиме токени на генерацію тестів для непрацездатного коду. Логіку активації функціоналу продемонстровано на рис. 7.

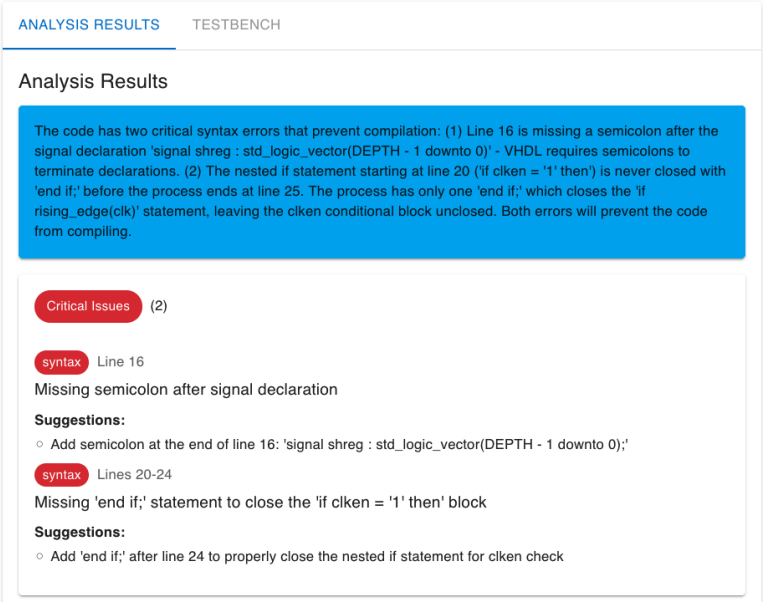


Рисунок 6 – Результат аналізу коду з синтаксичною помилкою

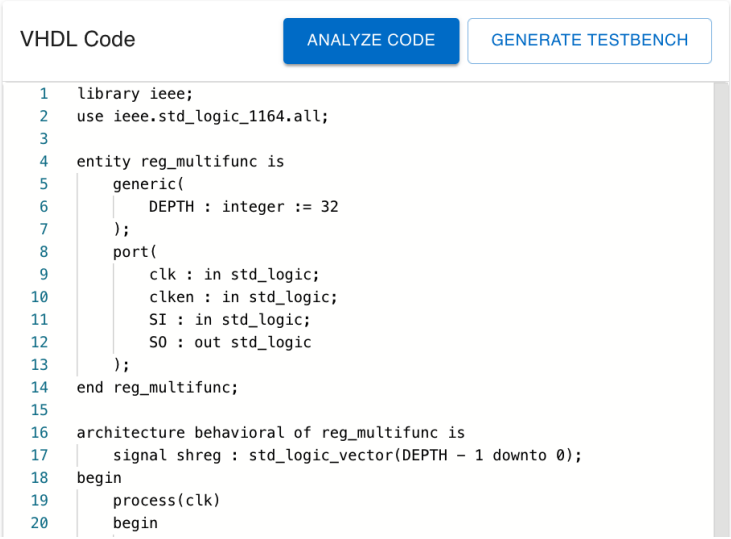


Рисунок 7 – Активація функції генерації тестбенчу після успішного аналізу коду

Перевірка інтерфейсу підтвердила адаптивність верстки та коректність роботи механізмів зворотного зв'язку (Loading states) під час тривалих запитів до LLM.

Окремим етапом стало тестування модуля автоматичної генерації верифікаційного оточення. Перевірено зручність та функціональність налаштування параметрів симуляції через спеціалізоване діалогове вікно. Експеримент показав, що інтерфейс коректно обробляє введення часових обмежень та текстового опису сценарію, передаючи їх на сервер для обробки. Зовнішній вигляд вікна налаштувань наведено на рисунку 8.

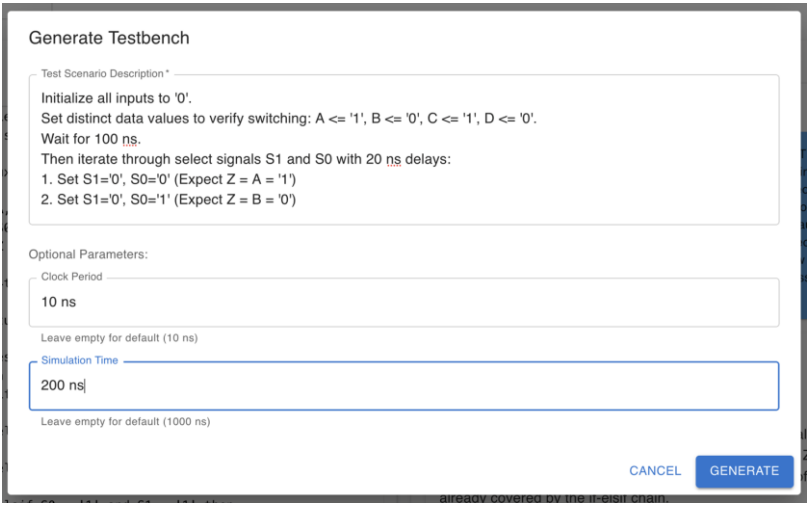


Рисунок 8 – Діалогове вікно генерації тестового оточення

Тестування генеративних можливостей підтвердило, що застосування стратегії «Text-to-Code» дозволяє отримувати синтаксично коректні тестбенчі на основі опису сценарію природною мовою. Система коректно інтерпретує інструкції користувача щодо часових параметрів симуляції та послідовності вхідних впливів, формуючи готовий до використання VHDL-файл. Приклад згенерованого коду наведено на рис. 9.

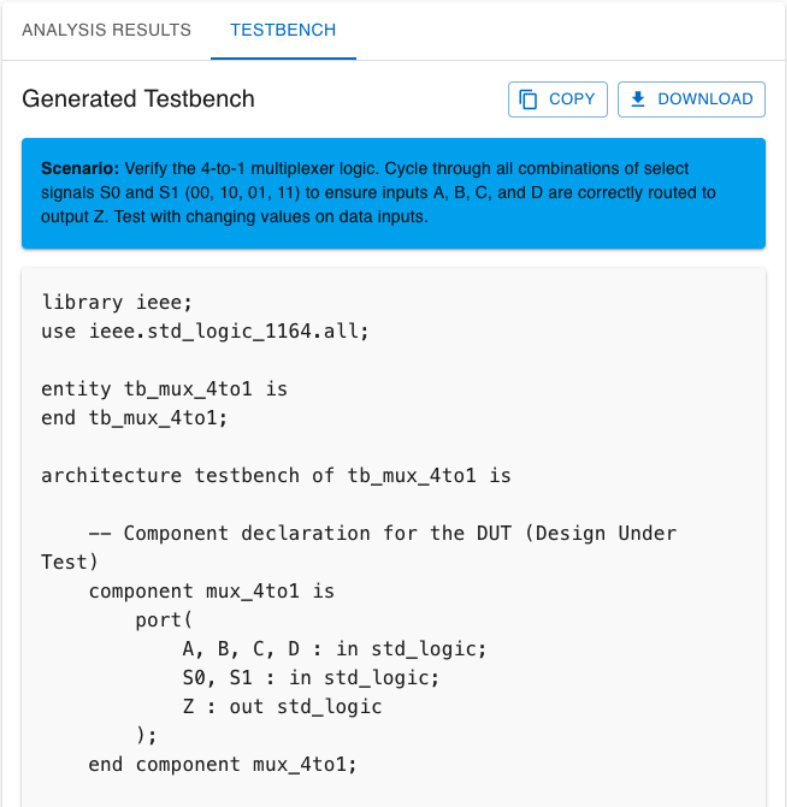


Рисунок 9 – Результат роботи генеративного модуля

Підсумовуючи результати тестування, можна стверджувати, що розроблена система демонструє високу стабільність роботи алгоритмів валідації та точність генерації коду. Комплексна перевірка підтвердила відповідність програмного продукту функціональним вимогам, що дозволяє рекомендувати його до впровадження у навчальний процес як ефективний інструмент для автоматизації верифікації FPGA-проєктів.

**У п'ятому розділі** сформульовано практичні рекомендації щодо впровадження та ефективної експлуатації системи у навчальному та виробничому процесах. Визначено, що завдяки реалізації у вигляді вебзастосунку, система не висуває високих вимог до апаратного забезпечення користувача, однак вимагає

базового розуміння принципів цифрової схемотехніки для коректної інтерпретації результатів.

Розроблено методику диференційованого вибору AI-моделі залежно від етапу роботи над проєктом:

- для поточної перевірки під час написання коду рекомендовано використовувати модель Claude Haiku 4.5, яка забезпечує миттєвий зворотний зв'язок;
- для фінальної верифікації перед синтезом доцільно застосовувати Claude Opus 4.1, що мінімізує ризики пропуску помилок;
- для навчальних цілей в умовах обмеженого бюджету обґрунтовано використання Gemini 2.5 Flash (Free Tier).

Особливу увагу приділено проблемі хибної інтерпретації свідомих інженерних рішень. Наголошено, що користувач повинен виступати фінальним арбітром при аналізі попереджень про створення засувок (latches) або асинхронної логіки, які модель може класифікувати як помилки через відсутність контексту фізичної реалізації схеми.

Для підвищення якості роботи модуля генерації тестбенчів формалізовано структуру ефективного користувацького запиту. Встановлено, що для отримання коректного VHDL-коду тесту, вхідний опис має містити три обов'язкові компоненти: об'єкт тестування, чіткий сценарій зміни сигналів у часі та критерій успіху (очікуваний стан виходів).

## ВИСНОВКИ

У магістерській роботі вирішено науково-прикладну задачу підвищення ефективності верифікації VHDL-коду шляхом розробки спеціалізованої консультативної системи на базі технологій штучного інтелекту. Отримані теоретичні та практичні результати дозволяють зробити наступні висновки:

1. На основі аналізу предметної області обґрунтовано доцільність інтеграції великих мовних моделей як інтелектуальних асистентів для вирішення проблеми високого порогу входження та автоматизації рутинних процесів верифікації FPGA-проєктів;

2. Експериментальне дослідження дев'яти AI-моделей дозволило визначити Claude Haiku 4.5 як оптимальний вибір для інтерактивного режиму завдяки високій точності виявлення логічних помилок (89,33%), тоді як Claude Opus 4.1 рекомендовано для фінальної верифікації через повну відсутність хибних спрацювань;

3. Спроєктовано та реалізовано гнучку клієнт-серверну архітектуру на базі стека Next.js та TypeScript із використанням патерну «Абстрактна Фабрика» для уніфікованої взаємодії з різними провайдерами штучного інтелекту;

4. Розроблено модуль статичного аналізу коду, який завдяки застосуванню стратегій промпт-інжинірингу та валідації JSON-структури забезпечує детальну класифікацію синтаксичних, логічних та стилістичних помилок із наданням рекомендацій щодо їх виправлення;

5. Реалізовано підсистему автоматичної генерації верифікаційного оточення, що дозволяє створювати синтаксично коректні VHDL-тестбенчі на основі текстового опису сценаріїв тестування природною мовою;

6. Підтверджено надійність та безпеку розробленої системи шляхом впровадження багаторівневої валідації вхідних даних, захисту від ін'єкцій у промпти та реалізації механізму обмеження частоти запитів.

Результати проведеного комплексного тестування підтверджують працездатність запропонованих рішень та досягнення мети роботи у повному обсязі.