Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА FPGA СИСТЕМИ ВЕРИФІКАЦІЇ ОЦІНОК

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-521

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

КЛИМАКОВ О.Є.
(ПРІЗВИЩЕ та ініціали)

Керівник ГРУШКО С.С.
(ПРІЗВИЩЕ та ініціали)
Рецензент КОЗІНА Г. Л.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
_____
«14» квітня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КЛИМАКОВА Олексія Євгеновича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка FPGA системи верифікації оцінок

керівник проєкту (роботи) к.т.н., доцент, ГРУШКО С.С.,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від « 08 » квітня 2025 року № 151
2. Строк подання студентом проєкту (роботи) 01.06.2025 р.
3. Вихідні дані до проєкту (роботи) опис предметної області, програмне середовище Quartus II

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз існуючих систем верифікації даних та криптографічних алгоритмів;
2) Проєктування архітектури системи;
3) Реалізація криптографічного алгоритму на FPGA.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ГРУШКО С.С.		
Нормоконтроль	ПОЛЬСЬКА О.В.		

7. Дата видачі завдання «14» квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз існуючих систем верифікації даних та	до 18.04.2025	
	криптографічних алгоритмів		
2	Визначення вимог до системи верифікації оцінок	до 22.04.2025	
3	Проєктування архітектури системи	до 24.04.2025	
4	Вибір моделі FPGA	до 28.04.2025	
5	Створення дизайну проєкту	до 01.05.2025	
6	Реалізація складників системи	до 19.05.2025	
7	Тестування та верифікація системи	до 22.05.2025	
8	Оформлення пояснювальної записки	до 25.05.2025	
9	Проходження нормоконтролю	до 01.06.2025	
10	Перевірка на наявність академічного плагіату	до 03.06.2025	
11	Проходження рецензування	до 10.06.2025	

Студент(ка) _____ Олексій КЛИМАКОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Світлана ГРУШКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 68 с., 9 табл., 11 рис., 1 дод., 13 джерел.

БЛОКЧЕЙН, ВЕРИФІКАЦІЯ ОЦІНОК, ІНТЕРФЕЙС, МОДУЛЬНА АРХІТЕКТУРА, ХЕШУВАННЯ, CYCLONE V, FPGA, JENKINS HASH FUNCTION, QUARTUS II, SHA.

Об’єкт розробки – апаратна система на базі FPGA для забезпечення цілісності даних у процесі верифікації оцінок за допомогою алгоритму Jenkins hash function.

Мета роботи – розробка ефективної FPGA-системи для швидкої та надійної верифікації оцінок, придатної для використання в освітніх цифрових системах.

У ході виконання роботи було проведено аналіз предметної області, що включав дослідження понять цілісності даних, криптографічних хеш-функцій і сучасних рішень для верифікації. Обґрунтовано вибір алгоритму Jenkins hash function і платформи Cyclone V. Спроєктовано модульну архітектуру системи з трьома основними компонентами: прийому даних, хешування та верифікації. Реалізацію виконано на мові Verilog у середовищі Quartus II 13.1. Тестування в Quartus II та симуляція підтвердили коректність обчислень хеш-значень для різних вхідних даних, а також точність верифікації шляхом порівняння хешів із еталонними значеннями.

Розроблена система є економічною, масштабованою та готовою до використання в освітніх закладах для забезпечення цілісності оцінок. Її модульна архітектура та інтеграція з блокчейн-технологіями дозволяють адаптацію до інших сценаріїв, що забезпечує потенціал для подальшого розвитку.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AES – Advanced Encryption Standard, Розширений стандарт шифрування

AES-NI – Advanced Encryption Standard New Instructions, Нові інструкції розширеного стандарту шифрування

ALM – Adaptive Logic Module, Адаптивний логічний модуль

AMD – Advanced Micro Devices, Компанія з виробництва мікропроцесорів

API – Application Programming Interface, Інтерфейс програмування додатків

ARM – Advanced RISC Machine, Архітектура процесорів із спрощеним набором команд

ASIC – Application-Specific Integrated Circuit, Інтегральна схема спеціального призначення

CLB – Configurable Logic Block, Конфігурований логічний блок

CPU – Central Processing Unit, Центральний процесор

CTAP – Client-to-Authenticator Protocol, протокол для встановлення зв'язку між клієнтським пристроєм та автентифікатором

DSP – Digital Signal Processing, Цифрова обробка сигналів

ECDSA – Elliptic Curve Digital Signature Algorithm, Алгоритм цифрового підпису на основі еліптичних кривих

EdDSA – Edwards-curve Digital Signature Algorithm, Алгоритм цифрового підпису на основі кривих Едвардса

FF – Flip-Flop, Тригер

FinFET – Fin Field-Effect Transistor, Транзистор із плавниковою структурою

FPGA – Field-Programmable Gate Array, Програмована логічна матриця

GPU – Graphics Processing Unit, Графічний процесор

HDL – Hardware Description Language, Мова опису апаратури

HLS – High-Level Synthesis, Високорівневий синтез

LUT – Lookup Table, Таблиця пошуку

NIST – National Institute of Standards and Technology, Національний інститут стандартів і технологій

PCIe – Peripheral Component Interconnect Express, Інтерфейс для підключення периферійних пристроїв

RSA – Rivest–Shamir–Adleman, Алгоритм асиметричного шифрування

SATA – Serial Advanced Technology Attachment, Послідовний інтерфейс передачі даних

SHA – Secure Hash Algorithm, Безпечний алгоритм хешування

SSL – Secure Sockets Layer, Протокол захисту на рівні сокетів

TLS – Transport Layer Security, Протокол безпеки транспортного рівня

USB – Universal Serial Bus, Універсальна послідовна шина

ВСТУП

Сучасні освітні системи активно переходять на цифрові технології, що забезпечують автоматизацію процесів документообігу, підвищують ефективність управління інформацією та сприяють прозорості навчального процесу. Одним із ключових елементів цифровізації освіти є електронне ведення оцінок, що дозволяє швидко обробляти та зберігати дані про навчальні досягнення студентів. Проте разом із перевагами цифрових систем виникають нові виклики, пов’язані з інформаційною безпекою, зокрема із захистом оцінок від несанкціонованих змін та фальсифікації. У цьому контексті розробка надійних систем верифікації оцінок стає критично важливою для забезпечення цілісності даних, конфіденційності та неспростовності в освітньому процесі.

Технологія FPGA відкриває нові можливості для створення високопродуктивних і енергоефективних рішень у сфері інформаційної безпеки. Завдяки апаратному паралелізму та гнучкості конфігурації FPGA ідеально підходять для реалізації криптографічних алгоритмів, таких як хешування та верифікація даних, забезпечуючи швидку обробку великих обсягів інформації. Використання FPGA у поєднанні з блокчейн-технологіями дозволяє створити розподілену систему, яка гарантує незмінність і прозорість даних, що є особливо важливим для освітньої сфери, де довіра до результатів оцінювання має першочергове значення.

Метою цієї роботи є розробка системи верифікації оцінок на базі FPGA, яка забезпечить цілісність і достовірність академічних даних у цифрових освітніх системах. У рамках дослідження буде проаналізовано вимоги до таких систем, проведено огляд сучасних криптографічних методів і порівняння програмних та апаратних реалізацій, спроєктовано архітектуру системи з використанням FPGA, реалізовано алгоритми хешування, а також виконано тестування системи для підтвердження її працездатності.

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ВЕРИФІКАЦІЇ ДАНИХ ТА КРИПТОГРАФІЧНИХ АЛГОРИТМІВ

1.1 Вимоги до системи верифікації оцінок

Система верифікації оцінок представляє собою важливий елемент освітньої інфраструктури, який забезпечує надійність та достовірність даних про навчальні досягнення. В умовах сучасної цифровізації освітнього процесу, коли більшість освітніх установ переходять на електронні системи документообігу, питання верифікації та захисту даних про оцінювання стає надзвичайно актуальним. Традиційні паперові журнали та відомості поступово замінюються електронними аналогами, що підвищує зручність та швидкість обробки інформації, проте одночасно створює нові виклики у сфері інформаційної безпеки. Впровадження надійних механізмів захисту та верифікації оцінок є критично важливим для збереження довіри до освітніх установ та підтримки високої якості освіти в цілому.

Розглянемо детально ключові вимоги до системи верифікації оцінок, які є фундаментальними для забезпечення її надійності та стійкості до різноманітних загроз інформаційної безпеки. Кожна з цих вимог відіграє важливу роль у створенні комплексної системи захисту, що здатна гарантувати безпеку та автентичність даних про навчальні досягнення студентів.

1.1.1 Цілісність даних

Цілісність даних є фундаментальною вимогою до системи верифікації оцінок. Вона виступає основою довіри до всієї системи освітнього оцінювання та забезпечує надійність інформації про академічні досягнення. Без належного забезпечення цілісності даних неможливо гарантувати справедливість та об'єктивність оцінювання, що може призвести до підриву авторитету навчального закладу та знецінення отриманих студентами документів про освіту.

Після внесення оцінки до системи будь-які несанкціоновані зміни мають бути неможливими або щонайменше виявлятися системою, що запобігає ситуаціям, коли оцінку можна змінити заднім числом без належного документування та

авторизації. Система повинна зберігати історію змін даних, фіксуючи, хто і коли їх вносив, що забезпечує прозорість процесу оцінювання, дозволяє відстежувати модифікації та відновлювати дані у разі помилок чи зловмисних дій, знижуючи ризики зловживань. Для унеможливлення фальсифікації оцінок застосовуються криптографічні методи, які гарантують виявлення будь-яких спроб підробки, а також механізми захисту від атак, таких як "людина посередині" чи інші кібератаки, спрямовані на підміну даних під час їх передачі або зберігання. Крім того, цілісність даних зберігається навіть у разі збоїв системи чи апаратних компонентів завдяки надійній інфраструктурі, яка включає резервне копіювання, розподілене зберігання інформації та механізми автоматичного відновлення після збоїв.

1.1.2 Конфіденційність

Конфіденційність є другою ключовою вимогою до системи верифікації оцінок, яка набуває особливого значення в контексті зростаючої уваги до захисту персональних даних та приватності. В освітньому середовищі інформація про оцінки студентів вважається чутливими персональними даними, доступ до яких повинен бути строго регламентований та захищений від несанкціонованого використання.

Різні категорії користувачів отримують різні права доступу до інформації відповідно до моделі контролю на основі ролей, що реалізує принцип мінімальних привілеїв, дозволяючи кожному користувачу працювати лише з даними, необхідними для виконання його функцій. Для захисту інформації під час передачі між компонентами системи використовуються захищені канали з шифруванням за допомогою протоколів, таких як TLS/SSL, що запобігає перехопленню даних зловмисниками та гарантує конфіденційність комунікацій. Дані в сховищі також шифруються, що робить їх непридатними для використання без відповідних ключів дешифрування, навіть у разі фізичного доступу зловмисника до серверів чи баз даних, а сучасні методи шифрування забезпечують високий рівень захисту за умови правильного управління ключами. Для доступу до чутливої інформації впроваджується багатофакторна автентифікація, яка значно підвищує безпеку та

знижує ризик несанкціонованого доступу, навіть якщо один із факторів автентифікації скомпрометовано. Усі спроби доступу до даних фіксуються в журналах аудиту, які зберігаються в захищеному форматі, що унеможливлює їх модифікацію чи видалення, і містять детальну інформацію про користувача, тип дії, час та об’єкт доступу для однозначної ідентифікації.

1.1.3 Неспростовність

Неспростовність є третьою ключовою вимогою до системи верифікації оцінок, яка забезпечує юридичну значущість та достовірність електронних записів про оцінювання. Вона гарантує, що жодна зі сторін освітнього процесу не зможе пізніше заперечувати свою участь у створенні або модифікації даних про оцінки, що є особливо важливим у випадках виникнення спорів або апеляцій щодо результатів оцінювання.

Система має чітко ідентифікувати особу, яка внесла оцінку, однозначно фіксуючи, який саме викладач або уповноважена особа здійснила цю дію, при цьому облікові записи користувачів повинні бути персональними та не використовуватися кількома особами спільно. Важливим є фіксація точного часу внесення даних за допомогою технології часових міток, що дозволяє встановити хронологічний порядок подій і підтвердити, що певна дія була виконана до або після конкретного моменту. Крім того, застосовуються технічні заходи, такі як цифрові підписи та криптографічні протоколи, які забезпечують математичні докази авторства дій, роблячи їх практично неможливими для підробки чи заперечення користувачем.

1.1.4 Доступність та надійність

Доступність та надійність є критично важливими характеристиками системи верифікації оцінок, оскільки від них залежить можливість безперебійного проведення освітнього процесу та своєчасного доступу до інформації про результати навчання. Система, яка часто виходить з ладу або працює нестабільно, не може забезпечити належний рівень довіри до електронних записів про оцінки та може призвести до серйозних проблем в організації навчального процесу.

Система має бути доступною для користувачів цілодобово, сім днів на тиждень, забезпечуючи безперервний доступ до даних. Для цього слід передбачити резервне копіювання, балансування навантаження та інші технічні заходи, які гарантують стабільну роботу навіть у разі збою окремих компонентів. Крім того, система повинна ефективно обробляти зростаючу кількість даних і користувачів без значного зниження продуктивності. Її архітектура має підтримувати горизонтальне та вертикальне масштабування, що дозволяє адаптуватися до збільшення потреб без необхідності повної перебудови.

1.1.5 Швидкодія

Швидкодія системи верифікації оцінок безпосередньо впливає на зручність її використання та ефективність роботи користувачів. Повільна система може призводити до зниження продуктивності викладачів, затримок у отриманні студентами інформації про свої результати та загального незадоволення користувачів. Особливо важливою є швидкодія під час пікових навантажень, наприклад, в періоди сесій або при одночасному виставленні оцінок за масові заходи контролю знань.

Оскільки криптографічні операції, які є невід’ємною частиною системи, можуть бути обчислювально складними та викликати значні затримки, необхідно ретельно обирати й оптимізувати алгоритми, щоб забезпечити баланс між рівнем безпеки та швидкістю роботи. Важливим аспектом зручності використання є швидкий доступ до історичних даних і ефективне виконання складних запитів. Крім того, слід застосовувати оптимальні алгоритми та конфігурації, які гарантують необхідний рівень безпеки без надмірного навантаження на обчислювальні ресурси.

1.1.6 Відповідність нормативним вимогам

Система верифікації оцінок повинна відповідати широкому спектру нормативних вимог, які регулюють як процеси обробки персональних даних, так і специфічні аспекти освітньої діяльності. Недотримання цих вимог може призвести до юридичних проблем для навчального закладу, втрати довіри студентів та партнерів, а також до фінансових санкцій з боку регуляторних органів.

Необхідно дотримуватися законодавства про захист персональних даних, яке регулює збір, обробку, зберігання, передачу інформації про фізичних осіб. Система має відповідати вимогам Міністерства освіти та акредитаційних органів щодо ведення й зберігання даних про результати навчання. До того ж важливо передбачити можливість проведення регулярних аудитів безпеки та генерації звітів, які підтверджують відповідність системи встановленим нормативам.

1.1.7 Зручність використання

Зручність використання є важливим аспектом системи верифікації оцінок, який безпосередньо впливає на її прийняття користувачами та ефективність їхньої роботи. Система, яка є технічно досконалою з точки зору безпеки, але складною та незручною у використанні, може призвести до помилок користувачів або навіть до спроб обійти встановлені процедури безпеки, що створює додаткові ризики.

Система має бути зрозумілою для користувачів із різним рівнем технічної підготовки, що досягається завдяки інтуїтивному інтерфейсу, створеному відповідно до принципів проектування користувацького досвіду, тестуванню з різними групами користувачів і врахуванню відгуків. Для запобігання помилкам необхідно впровадити механізми перевірки введених даних і підтвердження критичних дій, наприклад, вимагати додаткового підтвердження при внесенні змін до вже затверджених оцінок. Також важливо мінімізувати ручне введення даних і автоматизувати повторювані процеси, щоб підвищити ефективність роботи користувачів, зокрема через автоматичний імпорт даних із інших систем і автоматичне обчислення підсумкових оцінок на основі проміжних результатів.

1.2 Огляд існуючих криптографічних методів верифікації

1.2.1 Хеш-функції

Хеш-функції відіграють ключову роль у забезпеченні цілісності даних та є фундаментальним інструментом для верифікації інформації в освітніх системах. Ці

криптографічні примітиви перетворюють вхідні дані довільного розміру на вихідний бітовий рядок фіксованої довжини (хеш-значення або дайджест), що служить своєрідним "цифровим відбитком" вхідної інформації.

Основні властивості криптографічних хеш-функцій, що роблять їх придатними для верифікації даних:

- детермінованість – однакові вхідні дані дають однаковий результат;

- стійкість до пошуку прообразу – практично неможливо відновити вхідні дані за їх хешем;

- стійкість до колізій – надзвичайно складно знайти два різних вхідних повідомлення з однаковим хешем;

- лавинний ефект – невелика зміна вхідних даних призводить до значної зміни хеш-значення.

Сімейство алгоритмів SHA є однією з найпоширеніших груп хеш-функцій, що використовуються для верифікації даних:

- SHA-1 – створює 160-бітний (20 байтів) хеш-значення, але вважається застарілим через виявлені вразливості. У 2017 році було опубліковано практичну атаку, що продемонструвала можливість створення колізій, тому використання SHA-1 для нових систем верифікації не рекомендується;

- SHA-2 включає набір функцій з різною довжиною вихідного значення, наприклад, SHA-224 (224 біти), SHA-256 (256 біт), SHA-384 (384 біти), SHA-512 (512 біт), SHA-512/224 і SHA-512/256. SHA-2, особливо SHA-256, широко використовується для забезпечення цілісності даних у різних системах, включаючи освітні платформи. Алгоритм забезпечує хороший баланс між безпекою та продуктивністю, особливо при апаратній реалізації;

- SHA-3 – найновіше покоління стандарту, затверджене NIST у 2015 році. SHA-3 має модифікації з різною довжиною виходу (224, 256, 384 і 512 біт) і базується на принципово іншій конструкції, ніж SHA-1 і SHA-2, що забезпечує стійкість до атак, ефективних проти попередніх версій. SHA-3 також пропонує функції SHAKE128 і SHAKE256, які можуть генерувати вихідні дані довільної довжини [1].

Для систем верифікації оцінок на базі FPGA особливо цікавою є апаратна реалізація SHA-3, яка демонструє ефективну продуктивність та енергоспоживання на сучасних FPGA-платформах. Дослідження показують, що реалізації SHA-3 на FPGA можуть досягати пропускної здатності до кількох гігабіт за секунду при помірному використанні ресурсів кристалу [2].

BLAKE – це сімейство криптографічних хеш-функцій, які були фіналістами конкурсу SHA-3. Хоча BLAKE не став переможцем, його модифікації BLAKE2 і BLAKE3 здобули значну популярність завдяки високій продуктивності, особливо на платформах з обмеженими ресурсами:

- BLAKE2 пропонує дві основні версії: BLAKE2b (оптимізований для 64-бітних платформ) і BLAKE2s (оптимізований для 32-бітних платформ). BLAKE2 забезпечує продуктивність, що перевищує MD5, при цьому зберігаючи високий рівень безпеки;

- BLAKE3, представлений у 2020 році, пропонує ще вищу продуктивність завдяки вдосконаленій паралельній архітектурі та може бути особливо корисним для високошвидкісної верифікації даних у системах з великим обсягом інформації.

Апаратні реалізації BLAKE на FPGA демонструють відмінну продуктивність і ефективне використання ресурсів, що робить це сімейство привабливим для систем верифікації оцінок з високими вимогами до швидкодії [3].

1.2.2 Вразливості та атаки на хеш-функції

При проектуванні системи верифікації оцінок важливо враховувати потенційні вразливості хеш-функцій, зокрема:

- атаки на колізії – пошук двох різних повідомлень з однаковим хешем;

- атаки на пошук другого прообразу – для даного повідомлення M пошук іншого повідомлення M', яке має такий самий хеш;

- атаки на пошук прообразу – спроби відновити вхідні дані за відомим хеш-значенням;

- атаки подовження довжини – для деяких конструкцій хеш-функцій дозволяють додати дані до повідомлення без знання секретного ключа.

Для мінімізації ризиків рекомендується використовувати сучасні хеш-

функції (SHA-2, SHA-3, BLAKE2/3) з достатньою довжиною вихідного значення (не менше 256 біт) та регулярно відстежувати нові дослідження у сфері криптоаналізу.

1.2.3 Технології цифрового підпису

Цифрові підписи є критично важливим елементом верифікації даних, оскільки вони забезпечують не лише цілісність інформації, але й автентичність її джерела. У системі верифікації оцінок цифрові підписи дозволяють підтвердити, що оцінка була виставлена уповноваженою особою і не була змінена після створення.

RSA є однією з найстаріших і найпоширеніших схем цифрового підпису. Алгоритм RSA, що базується на математичній задачі факторизації великих чисел, функціонує через кілька ключових етапів. Спочатку генерується ключова пара, яка складається з відкритого та закритого ключів. Далі створюється хеш-значення документа. Це хеш-значення шифрується закритим ключем для формування цифрового підпису. На етапі верифікації підпис дешифрується відкритим ключем, а отримане хеш-значення порівнюється з хеш-значенням документа для підтвердження його автентичності.

Апаратна реалізація RSA на FPGA вимагає значних ресурсів через необхідність виконання операцій з великими числами, проте сучасні FPGA-платформи містять спеціалізовані DSP-блоки, які можуть ефективно прискорювати такі обчислення. Основними параметрами, що впливають на безпеку RSA, є довжина ключа (рекомендується не менше 2048 біт) та якість генерації випадкових чисел [4].

ECDSA – алгоритм цифрового підпису, що базується на еліптичних кривих. Порівняно з RSA, ECDSA забезпечує еквівалентний рівень безпеки при значно менших розмірах ключа. Наприклад, 256-бітний ключ ECDSA забезпечує рівень захисту, порівнянний з 3072-бітним ключем RSA.

Алгоритм ECDSA має кілька ключових переваг для систем верифікації даних. Він забезпечує менший розмір підпису, приблизно вдвічі компактніший порівняно з RSA, що сприяє економії ресурсів. ECDSA також вирізняється вищою

продуктивністю, особливо на пристроях із обмеженими обчислювальними можливостями, що робить його ефективним для використання в таких умовах. Крім того, алгоритм оптимізує використання пам’яті для зберігання ключів, що додатково підвищує його привабливість для систем із обмеженими ресурсами.

Апаратні реалізації ECDSA на FPGA демонструють хорошу продуктивність та ефективне використання ресурсів. Сучасні FPGA можуть виконувати операції підписування та верифікації ECDSA зі швидкістю декількох тисяч операцій на секунду, що дозволяє обробляти великі обсяги даних у реальному часі [4].

EdDSA, зокрема його реалізація Ed25519, набуває все більшої популярності завдяки високій продуктивності, стійкості до різних типів атак та детермінованому процесу генерації підпису. EdDSA використовує криві Едвардса, які мають певні математичні властивості, що спрощують реалізацію та підвищують продуктивність.

Алгоритм EdDSA вирізняється низкою особливостей, які роблять його цінним для систем верифікації оцінок. Він забезпечує детермінованість, завдяки чому однакові вхідні дані завжди генерують ідентичний підпис, що сприяє передбачуваності та надійності. EdDSA демонструє стійкість до атак побічними каналами, підвищуючи рівень безпеки. Алгоритм також характеризується швидким генеруванням і перевіркою підписів, що сприяє ефективності роботи системи. Крім того, компактний розмір ключів і підписів оптимізує використання ресурсів, що є важливим для систем із обмеженими можливостями.

Апаратні реалізації Ed25519 на FPGA демонструють високу продуктивність та енергоефективність, що робить цей алгоритм привабливим для систем верифікації даних з високими вимогами до швидкодії та безпеки [4].

1.2.4 Блокчейн-технології для верифікації даних

Технологія блокчейн знаходить застосування у різних сферах, де потрібна надійна верифікація даних без центрального довіреного органу, включаючи освітні системи. Основні компоненти блокчейн-системи:

- розподілений реєстр – база даних, що зберігається та оновлюється незалежно на кожному вузлі мережі;

- блоки – групи транзакцій, об'єднані в єдину структуру;

- хеш-покажчики – зв'язки між блоками, що забезпечують неможливість непомітної зміни попередніх блоків;

- консенсусний механізм – протокол, за яким вузли мережі досягають згоди щодо стану системи.

Блокчейн-технології можуть застосовуватися для верифікації оцінок шляхом створення постійного, незмінного реєстру академічних досягнень, що гарантує захист даних від модифікацій. Вони забезпечують прозорість процесу оцінювання, дозволяючи всім зацікавленим сторонам бачити достовірну інформацію. Крім того, блокчейн надає учням можливість контролювати свої освітні дані, забезпечуючи їхню автономію та безпеку.

Апаратна реалізація елементів блокчейн-системи на FPGA може значно підвищити продуктивність та енергоефективність вузлів мережі. Зокрема, FPGA добре підходять для прискорення криптографічних операцій, що використовуються в механізмах консенсусу та для створення хеш-покажчиків.

1.3 Порівняння програмних та апаратних реалізацій криптографічних алгоритмів

1.3.1 Особливості програмних реалізацій криптографічних алгоритмів

Програмні реалізації криптографічних алгоритмів традиційно виконуються на центральних процесорах (CPU) або графічних процесорах (GPU). Такий підхід характеризується низкою особливостей.

Реалізації на CPU залишаються найбільш поширеними завдяки їх гнучкості та простоті розробки. Сучасні процесори містять спеціалізовані інструкції для прискорення криптографічних операцій, такі як AES-NI у процесорах Intel та AMD, що значно підвищують швидкість симетричного шифрування. Однак, стандартні CPU не оптимізовані для масово-паралельної обробки даних, що обмежує їх

продуктивність у криптографічних задачах.

Реалізації на GPU демонструють вищу продуктивність для певних типів криптографічних алгоритмів, особливо тих, що добре піддаються паралелізації.

Програмні реалізації криптографічних алгоритмів мають низку суттєвих обмежень. Вони характеризуються високою затримкою обробки даних порівняно з апаратними рішеннями, що може знижувати швидкодію. Енергоспоживання таких реалізацій зростає пропорційно до обчислювального навантаження, що робить їх менш ефективними. Крім того, програмні рішення вразливі до програмних атак і експлуатації вразливостей операційної системи, що підвищує ризики безпеки. Обмежена пропускна здатність при обробці великих обсягів даних у реальному часі також ускладнює їх використання в високонавантажених системах.

1.3.2 Характеристики апаратних реалізацій на FPGA

FPGA представляють собою програмовані логічні матриці, які дозволяють реалізувати апаратні алгоритми з високим рівнем паралелізму. Їхня архітектура забезпечує унікальні переваги при реалізації криптографічних функцій.

FPGA-реалізації криптографічних алгоритмів мають низку ключових переваг. Вони забезпечують високу продуктивність завдяки справжньому паралелізму на апаратному рівні, що значно прискорює обчислення. FPGA дозволяють оптимізувати пропускну здатність або латентність залежно від вимог системи, забезпечуючи гнучкість у налаштуванні. Порівняно з CPU та GPU, вони споживають менше енергії при аналогічній продуктивності, що сприяє енергоефективності. Безпека підвищується завдяки фізичному розділенню обчислювальних ресурсів, що ускладнює атаки. Крім того, FPGA дозволяють адаптувати та оновлювати криптографічні алгоритми без необхідності зміни апаратної бази, забезпечуючи довгострокову гнучкість.

Дослідження продемонструвало, що для алгоритму SHA-3 реалізація на FPGA Xilinx Artix-7 забезпечує пропускну здатність до 6.9 Гбіт/с, що значно перевищує показники програмних реалізацій при істотно нижчому енергоспоживанні [5].

1.3.3 Кількісне порівняння продуктивності різних реалізацій

Для об'єктивного порівняння проаналізуємо результати досліджень щодо продуктивності різних криптографічних алгоритмів на різних обчислювальних платформах.

Порівняння швидкості обробки даних для SHA-256 наведено в таблиці 1.1.

Таблиця 1.1 - Порівняння швидкості обробки даних для SHA-256

Платформа	Пропускна здатність (Мбіт/с)	Відносна продуктивність
Intel Core i7-9700K	755	1x
NVIDIA RTX 2080	9.800	13x
Xilinx Kintex-7 FPGA	12.400	16.4x
Xilinx Virtex UltraScale+	27.200	36x

Крім того, FPGA забезпечують значно меншу та передбачувану затримку обробки даних, що критично важливо для систем реального часу.

Порівняння затримки для алгоритму RSA-2048 наведено в таблиці 1.2.

Таблиця 1.2 - Порівняння затримки для алгоритму RSA-2048

Платформа	Затримка (мс)	Відносна затримка
Intel Core i7-9700K	4.8	1x
NVIDIA RTX 2080	2.3	0.48x
Xilinx Kintex-7 FPGA	0.9	0.19x

1.3.4 Енергоефективність різних реалізацій

Енергоспоживання стає все більш важливим параметром для сучасних систем, особливо для мобільних та автономних пристроїв. Дослідження енергоефективності різних реалізацій криптографічних алгоритмів показують значну перевагу FPGA-рішень [6].

Дані таблиць свідчать, що FPGA забезпечують на порядок вищу енергоефективність порівняно з CPU та GPU. Це особливо важливо для систем, які повинні працювати тривалий час від автономного джерела живлення або в умовах обмеженого енергопостачання.

Порівняння енергоефективності для SHA-256 наведено в таблиці 1.3.

Таблиця 1.3 - Порівняння енергоефективності для SHA-256

Платформа	Енергоспоживання (Вт)	Енергоефективність (Мбіт/с/Вт)
Intel Core i7-9700K	95	7.95
NVIDIA RTX 2080	215	45.58
Xilinx Kintex-7 FPGA	18	688.89
Xilinx Virtex UltraScale+	42	647.62

1.3.5 Порівняння гнучкості та часу розробки

Важливим аспектом вибору платформи для реалізації криптографічних алгоритмів є гнучкість та час розробки, порівняльна характеристика цих платформ наведена на таблиці 1.4.

Таблиця 1.4 - Порівняння аспектів реалізації криптографічних алгоритмів

Аспект	CPU	GPU	FPGA
Час розробки	Низький	Середній	Високий
Складність відлагодження	Низька	Середня	Висока
Вартість розробки	Низька	Середня	Висока
Гнучкість модифікації	Висока	Середня	Середня

Однак, з розвитком високорівневих мов опису апаратури (HDL) та засобів високорівневого синтезу (HLS), таких як Vivado HLS від Xilinx та Intel HLS Compiler, розрив у часі розробки між програмними та апаратними реалізаціями поступово скорочується.

1.3.6 Практичні аспекти вибору реалізації для систем верифікації

При розробці системи верифікації оцінок важливо враховувати не лише технічні характеристики різних реалізацій, але й практичні аспекти їх застосування:

- масштабованість - системи на основі FPGA демонструють лінійне зростання продуктивності при додаванні апаратних ресурсів, тоді як для CPU/GPU масштабованість обмежена архітектурними особливостями;

- безпека - апаратні реалізації забезпечують вищий рівень захисту від атак, оскільки криптографічні ключі та операції виконуються в ізольованому середовищі, недоступному для програмних атак;

- надійність - апаратні рішення мають вищу стійкість до відмов та потенційно довший термін служби завдяки відсутності рухомих частин та нижчому нагріванню;

- вартість - хоча початкова вартість розробки FPGA-рішень вища, для масштабних систем з високим навантаженням загальна вартість володіння може бути трохи нижчою завдяки меншому енергоспоживанню та високій продуктивності.

1.3.7 Гібридні рішення та їх перспективи

Варто відзначити, що в сучасних системах верифікації все частіше застосовуються гібридні рішення, які поєднують переваги різних платформ. Наприклад, комбінація CPU та FPGA дозволяє делегувати інтенсивні криптографічні обчислення на апаратний прискорювач, залишаючи логіку управління та взаємодії з користувачем на процесорі.

Платформи, такі як Xilinx Zynq UltraScale+, які інтегрують багатоядерні ARM процесори з FPGA-тканиною на одному кристалі, створюють ідеальне середовище для розробки високопродуктивних та енергоефективних систем верифікації даних. Такі гібридні системи поєднують гнучкість програмного забезпечення з продуктивністю апаратних рішень, що робить їх оптимальним вибором для сучасних систем верифікації оцінок.

1.3.8 Результати експериментальних досліджень

Експериментальні дослідження, проведені в різних наукових центрах, підтверджують теоретичні переваги FPGA-реалізацій криптографічних алгоритмів. Порівняльний аналіз реалізацій різних хеш-функцій на CPU, GPU та FPGA наведений на таблиці 1.5 [7].

Особливо вражаючим є те, що ці показники були досягнуті при значно нижчому енергоспоживанні FPGA порівняно з CPU та GPU.

Таблиця 1.5 - Порівняння пропускної здатності для різних хеш-функцій (Гбіт/с)

Алгоритм	Intel Xeon E5-2680 v4	NVIDIA Tesla V100	Intel Stratix 10 FPGA
MD5	3.7	28.5	142.8
SHA-1	2.9	23.4	98.3
SHA-256	1.2	15.7	65.2
SHA-3	0.8	12.3	46.7

1.3.9 Застосування для системи верифікації оцінок

Для системи верифікації оцінок, вибір FPGA як платформи для реалізації криптографічних алгоритмів є обґрунтованим з огляду на наступні фактори:

- швидкість обробки даних - у високонавантажених системах верифікації, де одночасно може оброблятися велика кількість запитів (наприклад, під час сесії), висока пропускна здатність FPGA забезпечить своєчасну обробку усіх запитів;

- енергоефективність - для системи, яка повинна працювати цілодобово, нижче енергоспоживання FPGA призведе до істотної економії операційних витрат;

- безпека - апаратна реалізація критичних криптографічних функцій знижує ризик компрометації ключів та інших конфіденційних даних;

- детермінізм - FPGA гарантує стабільний час виконання операцій, що важливо для систем з вимогами до затримки.

Враховуючи це, можна рекомендувати FPGA-платформи для реалізації криптографічних функцій у системі верифікації оцінок, можливо з використанням гібридної архітектури для забезпечення гнучкості та зручності розробки.

1.4 Аналіз існуючих FPGA-рішень для криптографічних обчислень

1.4.1 Огляд архітектури сучасних FPGA-платформ

Розглянемо загальну архітектуру сучасних FPGA-платформ. Типова архітектура FPGA включає наступні основні компоненти:

- програмовані логічні блоки — базові елементи, що реалізують логічні функції та містять таблиці пошуку, тригери та мультиплексори;

- блоки цифрової обробки сигналів — спеціалізовані блоки для виконання арифметичних операцій, що особливо корисні для криптографічних алгоритмів;

- блоки вбудованої пам'яті — високошвидкісні блоки пам'яті, розподілені по всій площі чіпа, що забезпечують ефективний доступ до даних;

- програмовані міжз'єднання — мережа зв'язків між блоками, що конфігурується для створення бажаної схеми;

- блоки вводу-виводу — інтерфейси для зв'язку з зовнішніми компонентами;

- системні процесори — у деяких FPGA (як-от Xilinx Zynq) вбудовані ARM-процесори, що дозволяють реалізувати гібридні системи.

1.4.2 Аналіз платформи Intel (Altera) Stratix

Intel Stratix є найпотужнішою серією FPGA від компанії Intel (раніше Altera), орієнтованою на високопродуктивні обчислення. Серія Stratix 10 представляє особливий інтерес для криптографічних застосувань.

Intel Stratix 10 вирізняється низкою ключових характеристик. Платформа базується на технологічному процесі 14 нм FinFET, що забезпечує високу продуктивність і енергоефективність. Вона містить до 5,1 млн логічних елементів, що дозволяє реалізовувати складні обчислювальні задачі. Кількість блоків цифрової обробки сигналів сягає 11,721, що сприяє ефективній обробці сигналів і криптографічних операцій. Вбудована пам’ять досягає 244 Мбіт, забезпечуючи швидкий доступ до даних. Зовнішні інтерфейси підтримують трансивери зі швидкістю до 58 Гбіт/с, що гарантує високу пропускну здатність. Наявність технології HardCopy ASIC дозволяє конвертувати проекти в ASIC для масового виробництва, що оптимізує витрати при масштабуванні.

Intel Stratix 10 має архітектуру HyperFlex, що дозволяє розміщувати регістри не лише в логічних блоках, але й у маршрутах з'єднання. Це значно підвищує максимальну робочу частоту криптографічних модулів. Дослідження показало, що реалізація AES на Stratix 10 може працювати на частоті до 450 МГц, що на 40% вище порівняно з попереднім поколінням [8].

Підтримка криптографічних операцій:

Хоча Intel Stratix 10 не має спеціалізованих блоків для криптографічних операцій, вона пропонує оптимізовані IP-ядра для реалізації відповідних алгоритмів. IP-ядро AES-256 підтримує різні режими шифрування, забезпечуючи пропускну здатність до 40 Гбіт/с. IP-ядро для хеш-функцій SHA-2 і SHA-3 оптимізовано під архітектуру Stratix, що сприяє ефективній апаратній реалізації. Крім того, платформа включає апаратний генератор випадкових чисел, який забезпечує створення надійних криптографічних ключів.

Недоліком Stratix 10 є висока вартість (від $5000 за чіп) та значне енергоспоживання (від 70 до 225 Вт залежно від моделі), що може обмежувати його використання у вбудованих системах.

1.4.3 Аналіз платформи Intel (Altera) Cyclone

Intel Cyclone представляє економічно ефективну серію FPGA, орієнтовану на масові застосування з обмеженим бюджетом. Найновіша серія Cyclone 10 пропонує хороший баланс між продуктивністю, енергоспоживанням та вартістю.

Intel Cyclone 10 характеризується рядом ключових особливостей. Платформа використовує технологічний процес 20 нм для Cyclone 10 GX і 65 нм для Cyclone 10 LP, що забезпечує баланс між продуктивністю та вартістю. У Cyclone 10 GX доступно до 220,000 логічних елементів, що дозволяє реалізовувати проекти середньої складності. Кількість блоків DSP становить до 192, підтримуючи обробку сигналів і криптографічні задачі. Вбудована пам’ять досягає 11 Мбіт, забезпечуючи достатній обсяг для даних. Енергоспоживання платформи значно нижче, ніж у Stratix, і коливається від 2 до 15 Вт, що сприяє енергоефективності. Вартість моделей варіюється від $50 до $500, що робить Cyclone 10 доступним для широкого спектра застосувань [9].

Intel Cyclone 10 має обмежену підтримку спеціалізованих криптографічних функцій, але забезпечує ефективну реалізацію основних алгоритмів. Через Intel IP Catalog доступне IP-ядро AES із пропускною здатністю до 3 Гбіт/с, що підтримує базові потреби шифрування. Хеш-функції ефективно реалізуються за допомогою DSP-блоків і логічних елементів, оптимізуючи продуктивність. Також платформа

пропонує Crypto Suite IP — набір криптографічних примітивів, спеціально оптимізованих для архітектури Cyclone, що сприяє виконанню криптографічних операцій у системах із помірними вимогами.

Платформа Cyclone особливо підходить для розподілених систем верифікації оцінок, де потрібно багато недорогих вузлів з помірними вимогами до продуктивності.

1.4.4 Аналіз платформи Xilinx Zynq UltraScale+

Xilinx Zynq UltraScale+ представляє собою революційну концепцію System-on-Chip, що інтегрує багатоядерний процесор ARM Cortex-A53 (до 4 ядер), процесор реального часу ARM Cortex-R5 та програмовану логіку FPGA на одному кристалі. Це дозволяє створювати гетерогенні обчислювальні системи з оптимальним розподілом задач між програмними та апаратними компонентами.

Xilinx Zynq UltraScale+ вирізняється низкою ключових характеристик. Платформа базується на технологічному процесі 16 нм FinFET+, що забезпечує високу продуктивність і енергоефективність. Процесорна система включає чотириядерний ARM Cortex-A53 із частотою до 1,5 ГГц, двоядерний ARM Cortex-R5 із частотою до 600 МГц та графічний процесор Mali-400 MP2, що підтримує гнучке управління задачами. Програмована логіка містить до 1,1 млн логічних комірок, до 3,528 блоків DSP і до 34 Мбіт вбудованої пам’яті, що дозволяє реалізовувати складні обчислення. Платформа підтримує сучасні інтерфейси, зокрема PCIe Gen3, USB 3.0, SATA 3.1, DisplayPort і Gigabit Ethernet, забезпечуючи високу сумісність. Енергоспоживання є оптимізованим і коливається від 10 до 50 Вт залежно від навантаження, що сприяє ефективному використанню в різних сценаріях [10].

Xilinx Zynq UltraScale+ забезпечує унікальні можливості для реалізації криптографічних функцій завдяки інтеграції процесорної системи та програмованої логіки. У процесорній системі присутній апаратний модуль прискорення криптографічних операцій, який підтримує AES-256 із різними режимами шифрування, хеш-функції SHA-256 і SHA-3, асиметричну криптографію RSA, а також апаратний генератор випадкових чисел для створення надійних

ключів. Платформа пропонує розширені можливості безпеки, зокрема Secure Boot для захищеного завантаження з перевіркою цифрового підпису, Key Management для безпечного зберігання криптографічних ключів і ARM TrustZone, що створює ізольоване середовище для виконання критичних операцій. Для програмованої логіки доступні IP-ядра, такі як High-Performance AES Engine із пропускною здатністю до 100 Гбіт/с, SHA Family Accelerator для оптимізованих реалізацій SHA-2 і SHA-3, а також Elliptic Curve Processor, що забезпечує високу продуктивність і ефективність криптографічних задач.

Особлива перевага платформи Zynq полягає в можливості оптимального розподілу задач: критичні криптографічні операції виконуються на програмованій логіці, а управління та взаємодія з користувачами — на процесорній системі.

1.4.5 Порівняльний аналіз FPGA для криптографічних обчислень

Для об'єктивного порівняння розглянутих FPGA-платформ проаналізуємо їх за ключовими параметрами, важливими для реалізації криптографічних алгоритмів. Результати порівняння наведені в таблиці 1.6. Також наведено аналіз ефективності криптографічних алгоритмів на таблицях 1.7-1.9.

Таблиця 1.6 - Порівняння характеристик FPGA-платформ

Характеристика	Intel Stratix 10	Intel Cyclone 10	Xilinx Zynq UltraScale+
Логічні елементи	до 5100000	до 220000	до 1100000
Блоки DSP	до 11721	до 192	до 3528
Вбудована пам'ять	до 244 Мбіт	до 11 Мбіт	до 34 Мбіт
Вбудовані процесори	Немає	Немає	ARM Cortex-A53 + R5
Спеціалізовані крипто-блоки	Обмежено	Мінімально	Розширено
Пропускна здатність AES	до 40 Гбіт/с	до 3 Гбіт/с	до 100 Гбіт/с
Енергоспоживання	70-225 Вт	2-15 Вт	10-50 Вт
Вартість	$5000-30000	$50-500	$1000-10000

Таблиця 1.7 - Продуктивність AES-256 (Гбіт/с):

Режим роботи	Intel Stratix 10	Intel Cyclone 10	Xilinx Zynq UltraScale+
ECB	40.2	3.1	42.5
CBC	35.8	2.8	38.9
CTR	39.6	3.0	41.7
GCM	28.4	1.9	32.8

Таблиця 1.8 - Продуктивність хеш-функцій (Гбіт/с):

Алгоритм	Intel Stratix 10	Intel Cyclone 10	Xilinx Zynq UltraScale+
SHA-256	25.3	5.2	27.6
SHA-512	18.7	3.1	21.4
SHA-3	16.9	2.8	19.5
Blake2b	22.1	4.6	24.2

Таблиця 1.9 - Продуктивність асиметричної криптографії (операцій/с):

Операція	Intel Stratix 10	Intel Cyclone 10	Xilinx Zynq UltraScale+
RSA-2048 підпис	12500	1200	13800
RSA-2048 перевірка	256000	18500	285000
ECDSA-P256 підпис	45000	5800	52000
ECDSA-P256 перевірка	28000	3200	34000

Дані таблиць показують, що Xilinx Zynq UltraScale+ забезпечує найвищу продуктивність для більшості криптографічних алгоритмів, хоча й незначно випереджає Intel Stratix 10. При цьому Zynq має суттєву перевагу завдяки інтегрованій процесорній системі, що спрощує розробку комплексних рішень.

1.4.6 Специфічні рішення для криптографічних задач

Окрім стандартних FPGA-платформ, існують спеціалізовані рішення, оптимізовані для криптографічних обчислень:

- Efinix Trion FPGA - відносно нова архітектура FPGA, що пропонує високу енергоефективність для криптографічних застосувань. Trion використовує інноваційну архітектуру Quantum, що дозволяє досягти вищої щільності логіки при нижчому енергоспоживанні порівняно з традиційними FPGA;

- Microsemi PolarFire розроблена з урахуванням високих вимог безпеки та оснащена вбудованими криптографічними модулями. Вона включає Physical Unclonable Function для генерації унікальних ключів, що забезпечує захист від клонування. Платформа підтримує захищене завантаження з шифруванням бітового потоку, що запобігає запуску неавторизованого коду. Вбудований модуль апаратної генерації випадкових чисел, сертифікований за стандартом NIST SP 800-90, генерує справжні випадкові числа для криптографічних потреб. Крім того, PolarFire має стійкість до диференціального аналізу потужності, що захищає від атак по побічних каналах, забезпечуючи надійність у безпечних системах [11];

- Lattice FPGA з Enhanced Security включає технологію захисту конфігураційних даних і спеціалізовані IP-ядра для криптографічних операцій при надзвичайно низькому енергоспоживанні (менше 1 Вт).

Ці спеціалізовані рішення можуть бути корисними для специфічних задач у системі верифікації оцінок, особливо коли вимагається максимальний рівень захисту або мінімальне енергоспоживання.

1.4.7 IP-ядра для криптографічних обчислень

Важливим аспектом використання FPGA для криптографічних обчислень є наявність оптимізованих IP-ядер (Intellectual Property cores), які значно прискорюють розробку. Розглянемо основні доступні рішення:

а) Xilinx Security IP Portfolio:

1) AXI CRYPTO IP: повний набір криптографічних алгоритмів з інтерфейсом AXI4;

2) Secure IP Solution: забезпечує захищений завантажувач, менеджер ключів та криптографічні прискорювачі;

3) Vitis Security Library: високорівневі функції для HLS-реалізації криптографічних алгоритмів;

б) Intel (Altera) Security IP:

1) AES Mega-Function: оптимізована реалізація AES з різними режимами роботи;

2) Authentication Suite IP: набір хеш-функцій та MAC-алгоритмів;

3) Security Suite Pro: комплексне рішення для захисту даних;

в) сторонні IP-ядра:

1) Helion Technology Crypto Cores: високопродуктивні реалізації AES, SHA, RSA;

2) Algotronix Crypto IP: спеціалізовані криптографічні модулі з захистом від побічних каналів;

3) IpCores CryptoCore: оптимізовані реалізації популярних криптографічних алгоритмів.

Використання готових IP-ядер може значно скоротити час розробки та забезпечити вищу продуктивність порівняно з власними реалізаціями. Однак варто враховувати, що комерційні IP-ядра можуть мати значну вартість ліцензування.

1.4.8 Оцінка придатності FPGA-платформ для системи верифікації оцінок

На основі проведеного аналізу можна оцінити придатність розглянутих FPGA-платформ для реалізації системи верифікації оцінок:

- Intel Stratix 10: переваги включають надзвичайно високу продуктивність, великий обсяг логічних ресурсів і оптимізовані IP-ядра. Недоліки полягають у високій вартості, значному енергоспоживанні та відсутності вбудованих процесорів;

- Intel Cyclone 10: переваги охоплюють низьку вартість, компактність і енергоефективність. Недоліки включають обмежену продуктивність і мінімальну підтримку криптографічних операцій;

- Xilinx Zynq UltraScale+: переваги складаються з високої продуктивності, інтегрованих процесорів, розширеної підтримки безпеки та оптимального балансу між гнучкістю й продуктивністю. Недоліки пов’язані з вищою складністю розробки порівняно з чистими FPGA та середньою вартістю.

Зважаючи на специфіку задачі верифікації оцінок, яка вимагає швидкого виконання криптографічних операцій, помірної обчислювальної потужності та економічної доступності для освітніх закладів середнього масштабу, Intel Cyclone 10 є оптимальною платформою. Її низька вартість (від $50 до $500), достатні логічні

ресурси (до 220,000 елементів) і енергоефективність (2–15 Вт) забезпечують ефективну реалізацію хешування і верифікації даних, відповідаючи вимогам швидкодії та масштабованості. Cyclone 10 підтримує інтеграцію з освітніми платформами через стандартні інтерфейси, а використання готових IP-ядер прискорює розробку, що робить її ідеальним вибором для створення надійної та доступної системи верифікації оцінок.

1.5 Аналіз існуючих систем верифікації даних

1.5.1 FIDO2

FIDO2 є сучасним стандартом аутентифікації, розробленим FIDO Alliance, який об’єднує два протоколи: WebAuthn і CTAP. Цей стандарт спрямований на підвищення безпеки та зручності аутентифікації, усуваючи залежність від паролів через використання асиметричної криптографії та фізичних або біометричних аутентифікаторів. У контексті освітніх систем FIDO2 може застосовуватися для верифікації ідентичності викладачів і студентів під час внесення або перевірки оцінок, забезпечуючи захист від фішингових атак і несанкціонованого доступу.

Основна перевага FIDO2 полягає в його здатності створювати унікальні пари ключів (приватний і публічний) для кожного сервісу, що унеможливлює повторне використання компрометованих даних. Приватний ключ зберігається в захищеному апаратному модулі (наприклад, USB-токені чи смартфоні), а публічний ключ реєструється на сервері навчального закладу. Під час аутентифікації сервер ініціює виклик через WebAuthn API, а аутентифікатор виконує криптографічне згенерування підпису, що підтверджує ідентичність користувача. Для освітніх систем це дозволяє реалізувати безпечний доступ до системи верифікації оцінок, інтегруючи FIDO2 із блокчейн-реєстрами для додаткового захисту даних.

Однак FIDO2 має обмеження: висока вартість апаратних аутентифікаторів може бути проблемою для бюджетних закладів, а складність інтеграції з існуючими системами вимагає значних ресурсів на адаптацію. Крім того, стандарт не вирішує питання безпосередньої перевірки цілісності даних, а зосереджується на ідентифікації, що потребує додаткових механізмів, таких як хешування чи цифрові підписи, для повної верифікації оцінок.

1.5.2 WebAuthn

WebAuthn є частиною стандарту FIDO2 і представляє собою API, інтегрований у сучасні браузери, який дозволяє вебдодаткам використовувати аутентифікатори FIDO для безпечної автентифікації. У контексті освітніх систем WebAuthn може застосовуватися для забезпечення доступу до будь-яких платформ, де зберігаються оцінки та інша важлива інформація, наприклад, через біометричні сканери чи безконтактні ключі. Цей підхід усуває необхідність запам’ятовування паролів і підвищує рівень безпеки завдяки криптографічним ключам, згенерованим на стороні клієнта.

WebAuthn працює шляхом реєстрації користувача, коли браузер і аутентифікатор створюють пару ключів, а публічний ключ передається серверу. Під час автентифікації сервер надсилає виклик через WebAuthn API, а аутентифікатор генерує підпис, який перевіряється сервером. У системах верифікації оцінок це може бути використано для авторизації викладачів перед внесенням даних у блокчейн, забезпечуючи неспростовність дій. Інтеграція з освітніми платформами, такими як Moodle, дозволяє автоматизувати процеси автентифікації, підвищуючи ефективність роботи.

Обмеженнями WebAuthn є залежність від сумісності браузерів і необхідність наявності апаратних аутентифікаторів, що може ускладнити впровадження в навчальних закладах із застарілою інфраструктурою. Крім того, як і FIDO2, WebAuthn не забезпечує перевірки цілісності даних, що вимагає комбінування з технологіями, такими як хеш-функції чи блокчейн, для покриття потреб верифікації оцінок.

1.6 Висновки до розділу 1

Проведений аналіз підтверджує актуальність розробки системи верифікації оцінок на базі FPGA для забезпечення цілісності, конфіденційності та неспростовності даних у цифрових освітніх системах. Ключові вимоги до системи включають захист від несанкціонованих змін, швидкий доступ до даних, відповідність нормативним стандартам і зручність використання, що є основою для підтримки довіри до освітнього процесу. Огляд криптографічних методів показав, що хеш-функції, цифрові підписи та блокчейн-технології є ефективними інструментами для верифікації даних. Апаратні реалізації на FPGA, зокрема на платформах Intel Stratix 10, Intel Cyclone 10 та Xilinx Zynq UltraScale+, значно перевищують програмні за продуктивністю та енергоефективністю. Intel Cyclone 10 виділяється завдяки низькій вартості, достатнім логічним ресурсам і енергоефективності, що робить її оптимальним вибором для систем середнього масштабу. Таким чином, FPGA-системи є перспективним рішенням для верифікації оцінок, забезпечуючи високий рівень безпеки, швидкодію та економічну доступність. Стандарти FIDO2 і WebAuthn забезпечують безпечну аутентифікацію для освітніх систем через криптографію, але не вирішують завдання перевірки цілісності даних. Їх комбінація з FPGA-системами та блокчейн-технологіями створює основу для комплексного захисту оцінок.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

2.1 Загальна структура системи

Система верифікації оцінок на базі FPGA розроблена для забезпечення цілісності та незмінності академічних даних у цифрових освітніх системах. Вона поєднує апаратні та програмні компоненти, щоб швидко й надійно вносити,

обробляти, верифікувати та зберігати оцінки. Основна мета системи — гарантувати, що оцінки, внесені до системи, не можуть бути змінені без виявлення, а їхня достовірність може бути перевірена в будь-який момент. Це сприяє підтримці довіри до освітнього процесу в умовах цифровізації.

Система побудована як розподілена архітектура, що включає кілька FPGA-вузлів, кожен із яких обробляє запити на хешування та верифікацію оцінок. Такий підхід зменшує залежність від єдиного сервера, підвищує стійкість до відмов і дозволяє масштабувати систему для великих освітніх закладів. Центральний сервер виконує функції маршрутизації запитів, спрямовуючи їх до найменш завантаженого або найближчого FPGA-вузла, що забезпечує оптимальну продуктивність навіть під час пікових навантажень, наприклад, у період сесій. Користувачі, такі як викладачі чи адміністратори, взаємодіють із системою через зручний інтерфейс, реалізований як вебдодаток або мобільний застосунок. Інтерфейс дозволяє виконувати наступні дії:

- вносити нові оцінки разом із метаданими, такими як ідентифікатор студента чи дисципліна;

- переглядати збережені оцінки для контролю та аналізу;

- перевіряти достовірність оцінок шляхом порівняння хешів.

Вхідні дані формуються у структурованому форматі і передаються через API, який забезпечує стандартизований обмін даними. API дозволяє інтегрувати систему з іншими освітніми платформами, такими як Moodle чи локальні бази даних вишів, що робить її гнучкою та сумісною з наявною інфраструктурою.

Після отримання оцінки через API сервер проводить попередню перевірку даних на коректність і повноту. Перевіряються наступні аспекти:

- наявність обов’язкових полів, таких як ідентифікатор студента, значення оцінки та дата;

- відповідність формату даних (наприклад, числове значення оцінки в межах допустимого діапазону);

- унікальність запису, щоб уникнути дублювання оцінок.

Якщо дані валідні, сервер спрямовує їх до одного з FPGA-вузлів для криптографічної обробки. FPGA-модуль є ключовим компонентом системи, оскільки відповідає за обчислення хеш-значення оцінки, що гарантує її цілісність. Використання FPGA значно прискорює хешування порівняно з програмними рішеннями завдяки апаратному паралелізму. Модуль застосовує хеш-функцію, яка створює унікальний «відбиток» даних, чутливий до будь-яких змін. FPGA підтримує конвейєрну архітектуру, що дозволяє одночасно обробляти кілька запитів, забезпечуючи високу пропускну здатність. Після обчислення хеш повертається серверу, який додає його разом із вихідною оцінкою та метаданими до запису в блокчейні.

Блокчейн слугує для незмінного зберігання верифікованих оцінок і є основою безпеки системи. Він реалізований як приватна мережа, де вузлами є довірені організації, наприклад, навчальні заклади чи акредитаційні органи. Кожен вузол зберігає повну копію блокчейну, що забезпечує децентралізацію та захист від втрати даних. Блокчейн має наступні характеристики:

- структура блоків: складається з послідовності блоків, кожен із яких містить групу оцінок, їхні хеш-значення, часову мітку та хеш-покажчик, що зв’язує поточний блок із попереднім;

- незмінність: хеш-покажчик створюється шляхом хешування заголовка попереднього блоку, що робить неможливим непомітну зміну даних у будь-якому блоці без перерахунку всіх наступних;

- механізм консенсусу: використовується Proof of Authority, де кілька заздалегідь визначених вузлів (наприклад, представники університетів) по черзі підтверджують нові блоки. Proof of Authority є енергоефективним і забезпечує швидке узгодження, що підходить для приватних блокчейнів;

- прозорість: усі вузли мають доступ до однакової версії блокчейну, що дозволяє незалежно перевіряти дані.

Нові блоки формуються періодично, наприклад, щогодини або щоденно, залежно від обсягу оцінок, і додаються до мережі після підтвердження консенсусом. Така структура гарантує, що оцінки залишаються незмінними після

внесення, а будь-яка спроба фальсифікації буде виявлена через невідповідність хешів.

Для перевірки достовірності оцінки користувач надсилає запит через інтерфейс, вказуючи ідентифікатор оцінки чи студента. Сервер витягує відповідний запис із блокчейну, що містить оцінку та її хеш, і передає дані на FPGA-вузол. FPGA повторно обчислює хеш оцінки і порівнює його зі збереженим хешем. Якщо хеші збігаються, сервер повертає користувачу підтвердження валідності оцінки разом із її деталями, такими як значення оцінки та дата внесення. У разі невідповідності система сигналізує про можливу фальсифікацію, що дозволяє швидко виявляти спроби несанкціонованих змін. Завдяки розподіленій архітектурі та паралельній обробці на FPGA система може обробляти велику кількість запитів на верифікацію без затримок, що критично важливо для великих освітніх закладів.

Результати обробки відображаються через інтерфейс користувача, який показує оцінку, статус її валідності та додаткові дані, такі як дисципліна чи дата. Інтерфейс розроблено з урахуванням зручності, щоб користувачі з різним рівнем технічної підготовки могли легко вносити та перевіряти оцінки. Розподілена архітектура з кількома FPGA-вузлами дозволяє масштабувати систему, додаючи нові вузли за потреби, що робить її придатною для використання як у невеликих, так і у великих освітніх закладах. Використання блокчейну забезпечує прозорість і незмінність даних, а FPGA гарантує швидке виконання криптографічних операцій, що разом створює надійне рішення для верифікації оцінок.

2.2 Вибір алгоритму хешування

Для забезпечення ефективної роботи системи верифікації оцінок на базі FPGA необхідно обрати оптимальний алгоритм хешування. Порівнюються чотири алгоритми — SHA-1, MD5, SHA-256 і Jenkins hash function — за критеріями, які враховують особливості освітніх систем: економічна доступність для освітніх закладів, низьке енергоспоживання для цілодобової роботи, модульна архітектура для простоти розробки та тестування, а також можливість паралельної обробки запитів.

2.2.1 Економічна доступність для освітніх закладів

Економічна доступність залежить від складності апаратної реалізації алгоритму та обсягу ресурсів FPGA, необхідних для його роботи. SHA-1 і MD5 є відносно простими алгоритмами, але їхня застарілість і вразливість до атак (наприклад, знаходження колізій) роблять їх менш привабливими для сучасних систем, хоча вони потребують менше логічних елементів. SHA-256 є криптографічно стійким, але потребує значно більше ресурсів, що збільшує витрати на FPGA для шкіл чи університетів із обмеженим бюджетом. Jenkins hash function, хоч і не криптографічно стійкий, є надзвичайно легким для реалізації, використовуючи незначну кількість ресурсів, що робить його економічно вигідним для бюджетних освітніх закладів.

2.2.2 Низьке енергоспоживання для цілодобової роботи

Енергоспоживання залежить від обчислювальної складності алгоритму та частоти операцій. SHA-256, через свою складність (64 раунди обробки), споживає більше енергії, що може бути проблематичним для цілодобової роботи. SHA-1 і MD5 є менш енергоємними, але їхня вразливість обмежує використання. Jenkins hash function, завдяки простій структурі (основні операції — додавання, XOR і зсуви), споживає мінімальну кількість енергії, що ідеально підходить для цілодобового функціонування в умовах обмежених ресурсів.

2.2.3 Модульна архітектура для простоти розробки та тестування

Модульна архітектура полегшує розробку, тестування та адаптацію системи. SHA-1 і MD5 мають чітко визначені етапи обробки (наприклад, 80 раундів у SHA-1), але їхня реалізація на FPGA потребує складного керування послідовністю операцій, що ускладнює модульність. SHA-256 ще складніший через більшу кількість операцій і залежностей між раундами, що ускладнює тестування. Jenkins hash function, навпаки, має просту структуру, яка легко розбивається на модулі (наприклад, ініціалізація, змішування, фіналізація), що спрощує розробку та тестування на FPGA, а також дозволяє швидко адаптувати алгоритм до нових вимог.

2.2.4 Можливість паралельної обробки запитів

Паралельна обробка запитів залежить від здатності алгоритму ефективно розподіляти обчислення. SHA-1, MD5 і SHA-256 мають послідовний характер обробки (кожен раунд залежить від попереднього), що обмежує паралелізм без значного ускладнення дизайну. Наприклад, SHA-256 потребує додаткових конвеєрних етапів для паралельної обробки, що збільшує затримки. Jenkins hash function, завдяки своїй простоті та відсутності складних залежностей між операціями, легко адаптується до паралельної обробки: кілька екземплярів алгоритму можуть працювати одночасно на різних ядрах FPGA, забезпечуючи високу пропускну здатність.

2.2.5 Результати порівняння

Порівняння показало, що SHA-1 і MD5 є економічними та енергоефективними, але їхня вразливість до атак робить їх непридатними для верифікації оцінок. SHA-256 забезпечує криптографічну стійкість, але потребує більше ресурсів і енергії, що ускладнює його використання в бюджетних системах. Jenkins hash function, попри відсутність криптографічної стійкості, переважає за всіма критеріями: він економічний, енергоефективний, легко інтегрується в модульну архітектуру та підтримує паралельну обробку. Для освітніх закладів, де пріоритетом є економія та простота реалізації, Jenkins hash function є оптимальним вибором, хоча в майбутньому його варто замінити на криптографічно стійкий алгоритм для підвищення безпеки.

2.3 Модульна схема FPGA та її взаємодія

FPGA-модуль є основним компонентом системи верифікації оцінок, відповідаючи за швидке виконання криптографічних операцій, таких як хешування та верифікація даних. Для забезпечення ефективної роботи FPGA розбивається на три функціональні модулі: модуль прийому даних, модуль хешування та модуль

верифікації. Кожен модуль виконує чітко визначене завдання, а їхня взаємодія через внутрішню шину даних забезпечує високу продуктивність і надійність. Модульна структура дозволяє оптимізувати обробку запитів, спрощує розробку та тестування, а також підтримує паралельну обробку, що робить систему придатною для використання в освітніх закладах різного масштабу.

2.3.1 Модуль прийому даних

Модуль прийому даних слугує інтерфейсом між сервером і внутрішніми компонентами FPGA, відповідаючи за отримання та попередню обробку вхідних запитів. Його основні функції включають:

- отримання структурованих даних (номер викладача, номер предмета, номер студента, оцінка) від сервера через високошвидкісний інтерфейс зв’язку;

- буферизація запитів для паралельної обробки, дозволяючи модулю ефективно працювати з кількома запитами одночасно.

Модуль передає дані до модуля хешування (для внесення оцінки) або модуля верифікації (для перевірки достовірності). Оптимізація для паралельної обробки забезпечує швидку реакцію навіть під час високих навантажень, таких як масове внесення оцінок у період сесій.

2.3.2 Модуль хешування

Модуль хешування відповідає за створення криптографічного хеш-значення оцінки, що гарантує її цілісність. Він використовує алгоритм Jenkins hash function, який генерує унікальний «відбиток» даних, чутливий до будь-яких змін. Основні характеристики модуля:

- прийом структурованих даних від модуля прийому. Застосування алгоритму Jenkins hash function;

- використання конвейєрної архітектури, що дозволяє одночасно обробляти кілька наборів даних, зменшуючи час очікування;

- можливість налаштування модуля для підтримки інших хеш-функцій у майбутньому, якщо стандарти безпеки зміняться.

Хеш-значення повертається серверу для додавання до блокчейну. Завдяки конвейєрній обробці модуль ефективно справляється з великими обсягами запитів.

2.3.3 Модуль верифікації

Модуль верифікації призначений для перевірки достовірності оцінок шляхом порівняння хешів. Він забезпечує швидке виявлення будь-яких змін у даних, збережених у блокчейні. Основні функції модуля:

- отримання оцінки та її хешу із блокчейну через модуль прийому даних;

- обчислення хешу оцінки за допомогою Jenkins hash function, використовуючи ті ж параметри, що під час початкового внесення;

- порівняння новообчисленого хешу із хешем із блокчейну. Повернення результату валідності (збігу хешів) або невалідності (невідповідності).

Модуль підтримує паралельну обробку, що дозволяє одночасно перевіряти кілька оцінок. Його ізольована структура підвищує безпеку, знижуючи ризик зовнішнього впливу на процес верифікації.

2.3.4 Взаємодія модулів та зовнішніх компонентів

Модулі FPGA взаємодіють між собою та із зовнішніми компонентами системи через чітко визначені інтерфейси, що забезпечує швидку та надійну обробку даних. Основні аспекти взаємодії:

- модулі об’єднані внутрішньою шиною даних, яка підтримує потокову передачу та мінімізує затримки. Дані передаються послідовно: від модуля прийому до хешування або верифікації, а результати повертаються назад;

- FPGA взаємодіє з сервером через високошвидкісний інтерфейс, який забезпечує швидку передачу запитів і результатів. Сервер спрямовує запити до FPGA, використовуючи алгоритм балансування для оптимального розподілу;

- хеш-значення та вихідні дані оцінки передаються серверу для запису в блокчейн. Під час верифікації сервер витягує дані з блокчейну та надсилає їх на FPGA для перевірки.

Для підвищення продуктивності модулі працюють паралельно: поки один модуль хешує оцінку, інший може перевіряти іншу. Такий підхід забезпечує ефективну обробку великої кількості запитів. Модульна структура FPGA дозволяє легко адаптувати систему до нових вимог, наприклад, додавання підтримки інших криптографічних алгоритмів чи інтеграції з додатковими освітніми платформами.

2.4 Вибір моделі FPGA

Вибір моделі FPGA для системи верифікації оцінок є важливим етапом, оскільки від нього залежать продуктивність, економічність і здатність системи відповідати вимогам освітніх закладів. Основними задачами FPGA в системі є швидке хешування оцінок за допомогою алгоритму Jenkins hash function, верифікація даних шляхом порівняння хешів і підтримка паралельної обробки запитів. Для вибору оптимальної платформи враховуються такі критерії, як кількість логічних елементів, підтримка криптографічних операцій, енергоспоживання та вартість.

Система верифікації оцінок не вимагає надвисокої обчислювальної потужності, оскільки основна операція — хешування Jenkins hash function — є відносно компактною задачею, а кількість одночасних запитів у типовому навчальному закладі залишається помірною. Cyclone V, як FPGA початкового та середнього рівня, пропонує достатньо логічних елементів (від 9000 до 60000 залежно від моделі) для реалізації модулів прийому даних, хешування та верифікації, описаних у попередньому розділі. Ці ресурси дозволяють ефективно виконувати криптографічні операції, підтримувати конвейєрну архітектуру та обробляти кілька запитів паралельно. Наприклад, модуль хешування потребує лише кілька тисяч логічних елементів, що легко вміщується навіть у молодші моделі Cyclone V [12].

Для оцінки Cyclone V порівнювалися з іншими платформами, такими як Intel Stratix 10 і Intel Cyclone 10, які також підходять для подібних задач. Основні критерії порівняння включають:

а) логічні ресурси:

1) Cyclone V: 9000–60000 логічних елементів, достатньо для реалізації всіх модулів із запасом для оптимізації;

2) Stratix 10: до 5500000 логічних елементів, що значно перевищує потреби системи;

3) Cyclone 10: 10000–220000 логічних елементів, що схоже з Cyclone V;

б) продуктивність криптографічних операцій:

1) Cyclone V підтримує апаратну реалізацію Jenkins hash function через програмовану логіку, забезпечуючи пропускну здатність до 4.8 Гбіт/с;

2) Stratix 10: до 100 Гбіт/с для Jenkins hash function, але надмірна продуктивність для задачі;

3) Cyclone 10: до 5.2 Гбіт/с, незначно вища за Cyclone V, але без суттєвих переваг для помірних навантажень;

в) енергоспоживання:

1) Cyclone V: 1–10 Вт, що робить її енергоефективною для цілодобової роботи в освітніх закладах;

2) Stratix 10: 50–150 Вт, надто високе для систем середнього масштабу;

3) Cyclone 10: 2–15 Вт, дещо вище за Cyclone V через новішу 20-нм технологію;

г) вартість:

1) Cyclone V: $30–$400;

2) Stratix 10: $5000–$20000, що непрактично для освітніх закладів;

3) Cyclone 10: $50–$500, трохи дорожча за Cyclone V через сучаснішу архітектуру.

Після аналізу кількох платформ, включаючи високопродуктивні сімейства FPGA, обрано Intel Cyclone V як основну платформу, оскільки вона забезпечує достатню продуктивність для задачі за доступною ціною, що робить її привабливою для освітніх систем середнього масштабу.

Cyclone V вибрано як оптимальну платформу завдяки балансу між продуктивністю та економічністю. Її логічні ресурси дозволяють реалізувати всі необхідні модулі (прийому даних, хешування, верифікації) із підтримкою конвейєрної обробки, що забезпечує швидке виконання запитів. Хоча Cyclone V не має спеціалізованих криптографічних блоків, як деякі сучасні платформи, алгоритм Jenkins hash function ефективно реалізується через програмовану логіку, що відповідає вимогам системи. Низьке енергоспоживання знижує операційні витрати,

що особливо важливо для навчальних закладів із обмеженим бюджетом. Крім того, доступна ціна Cyclone V робить систему економічно привабливою для впровадження в школах, коледжах чи університетах середнього розміру.

Для взаємодії з сервером і блокчейном Cyclone V підтримує високошвидкісні інтерфейси (до 3.125 Гбіт/с), які забезпечують швидку передачу даних між FPGA і зовнішніми компонентами. Модульна структура FPGA дозволяє оптимізувати використання ресурсів, розподіляючи задачі між модулями та підтримуючи паралельну обробку. Наприклад, конвейєрна архітектура дозволяє одночасно хешувати одну оцінку, перевіряти іншу та отримувати нові запити, що забезпечує високу пропускну здатність навіть на платформі початкового рівня, як Cyclone V.

Обмеження Cyclone V, такі як менша кількість логічних елементів порівняно з високопродуктивними платформами та відсутність вбудованих криптографічних прискорювачів, не є критичними для системи верифікації оцінок. Задача хешування та верифікації не потребує значних обчислювальних ресурсів, а масштабування системи досягається за рахунок розподіленої архітектури з кількома FPGA-вузлами. У разі потреби в майбутньому система може бути адаптована до потужніших платформ, але на поточному етапі Cyclone V повністю відповідає вимогам за продуктивністю, вартістю та енергоефективністю [13].

2.5 Перспективи вдосконалення системи

Для подальшого розвитку системи верифікації оцінок на базі FPGA розглядаються наступні напрями вдосконалення. По-перше, заміна Jenkins hash function на криптографічно стійкий алгоритм, наприклад SHA-256, SHA-3 або BLAKE2, значно підвищить рівень безпеки даних, усунувши вразливість до атак типу колізій, що є критично важливим для захисту академічної інформації від фальсифікації. Цей перехід вимагатиме адаптації архітектури FPGA для підтримки складніших обчислень, але дозволить інтегрувати систему з міжнародними

стандартами безпеки. По-друге, використання DSP-блоків для прискорення арифметичних операцій забезпечить оптимізацію виконання хеш-функцій і верифікаційних модулів, що дозволить обробляти більші обсяги даних у реальному часі, наприклад, при масовому введенні оцінок наприкінці семестру. По-третє, додавання модулів цифрового підпису, таких як ECDSA або EdDSA, дозволить реалізувати механізм неспростовності внесених даних, забезпечуючи юридичну вагу електронним оцінкам у системі. Це також полегшить інтеграцію з блокчейн-платформами для створення незмінного реєстру оцінок. Також можна впровадити модулі управління ключами, що включають генерацію, зберігання та безпечний розподіл криптографічних ключів, підвищить захист від несанкціонованого доступу, особливо в умовах розподілених освітніх мереж. Такі модулі можуть бути реалізовані з використанням захищених областей пам’яті на FPGA, що забезпечить додатковий рівень безпеки. Подальші дослідження можуть бути спрямовані на оптимізацію енергоспоживання через динамічне керування ресурсами FPGA та розширення функціональності за рахунок підтримки кількох хеш-функцій одночасно. Ці удосконалення зроблять систему більш надійною, гнучкою та готовою до впровадження в масштабах цілих освітніх мереж.

2.6 Висновки до розділу 2

Розроблена архітектура системи верифікації оцінок на базі FPGA забезпечує ефективне рішення для захисту цілісності та незмінності академічних даних у цифрових освітніх системах. Розподілена структура з кількома FPGA-вузлами та приватним блокчейном на основі Proof of Authority гарантує захист від несанкціонованих змін і прозорість верифікації. Модульна організація FPGA з блоками для прийому даних, хешування за алгоритмом Jenkins hash function і порівняння хешів забезпечує високу продуктивність завдяки конвейєрній обробці та паралельному виконанню запитів. Вибір Jenkins hash function обґрунтовано його

економічністю, низьким енергоспоживанням, простотою модульної реалізації та підтримкою паралельної обробки, що ідеально відповідає потребам бюджетних освітніх закладів. Intel Cyclone V обрано за економічність ($30–$400), достатні логічні ресурси (9000–60000 елементів) і низьке енергоспоживання (1–10 Вт), що робить систему доступною для середніх навчальних закладів. Перспективи вдосконалення включають заміну Jenkins hash function на криптографічно стійкий алгоритм, використання DSP-блоків для прискорення обчислень, а також додавання модулів цифрового підпису й управління ключами для підвищення безпеки. Система поєднує апаратне прискорення з децентралізованим зберіганням даних, створюючи надійне та масштабоване рішення. Це сприяє підвищенню довіри до освітнього процесу, відповідаючи сучасним викликам цифровізації освіти та забезпечуючи швидку й безпечну верифікацію оцінок.

3 РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНОГО АЛГОРИТМУ НА FPGA

3.1 Створення дизайну проєкту

На початку роботи створюємо новий проєкт у середовищі Quartus II 13.1, яке є основним інструментом для розробки системи верифікації оцінок на FPGA. Для цього запускаємо Quartus II та в головному меню обираємо File > New Project Wizard. Це відкриває майстер створення проєкту, де задаємо робочу директорію і назву проєкту (рис. 3.1).

Цей процес призводить до створення базової структури проєкту. Модуль верхнього рівня, який запускається на FPGA, відповідатиме за обробку даних, тоді як хост-програма забезпечуватиме взаємодію з зовнішніми системами. На цьому етапі конфігурація інтерфейсів не передбачена.

Середовище Quartus II вимагає визначення апаратної платформи. Для плати Intel Cyclone V обираємо пристрій 5CGXFC4C7F27C8, яке підтримує достатню кількість логічних елементів для реалізації системи. Вибір здійснюється у вікні

майстра створення проєкту, де в полі Family задаємо Cyclone V, а в списку Available Devices знаходимо потрібну модель (рис. 3.2).

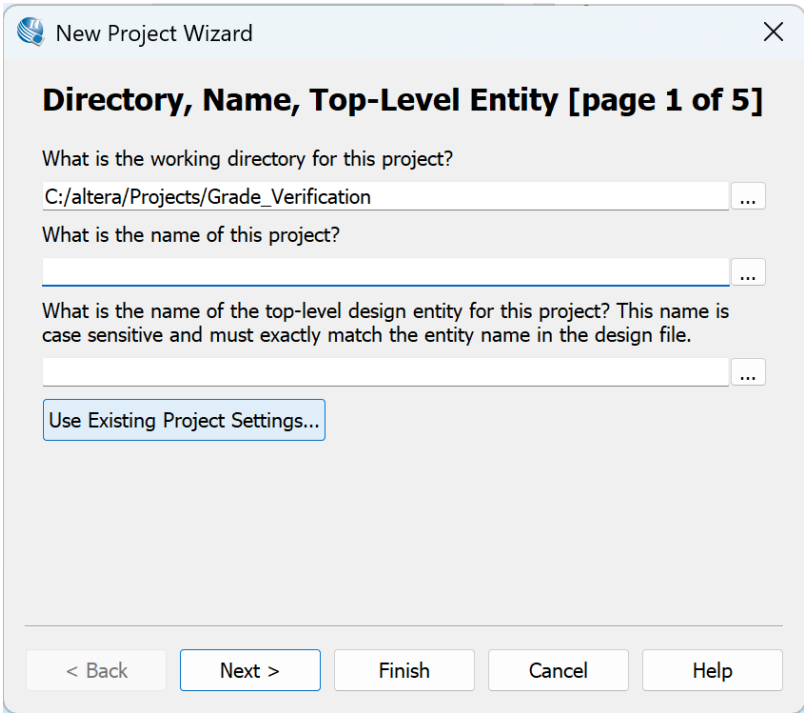


Рисунок 3.1 – Створення нового проєкту в Quartus II

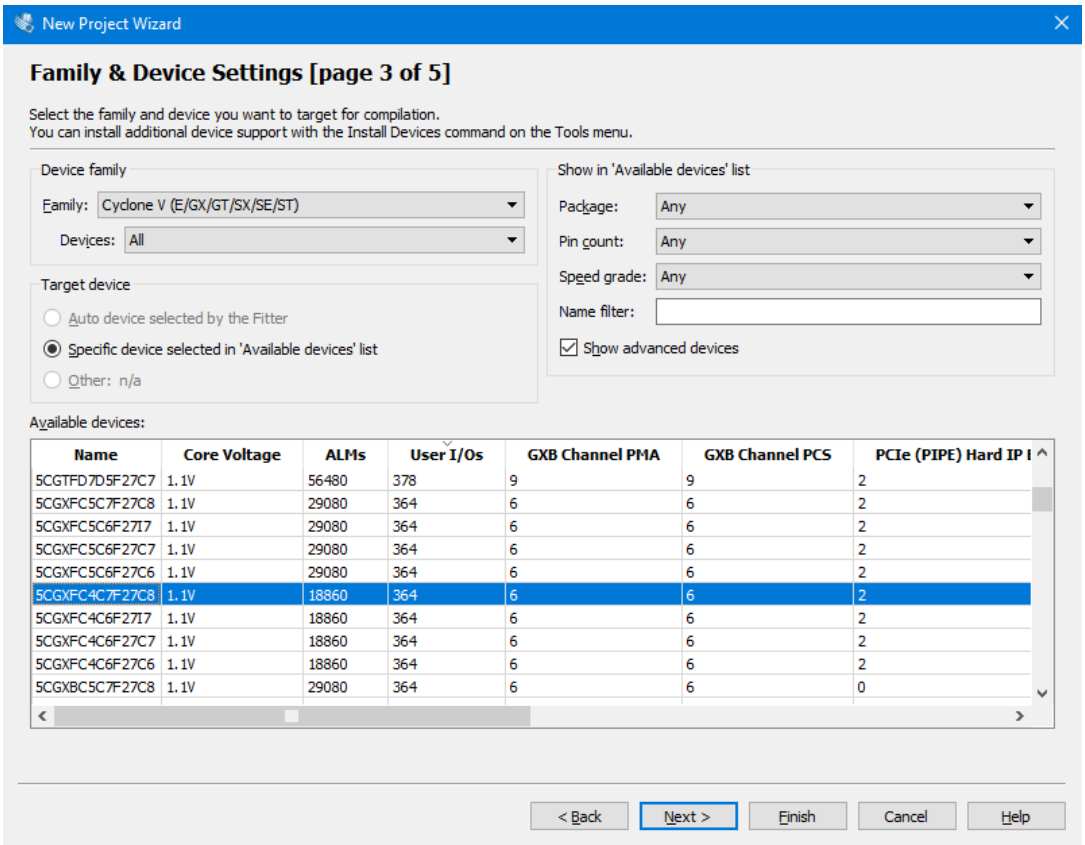


Рисунок 3.2 – Вибір платформи Cyclone V

Натискаємо Next. У наступному вікні залишаємо налаштування симуляції за замовчуванням, обираючи ModelSim-Altera як інструмент симуляції та мову опису апаратури Verilog HDL (рис. 3.3).

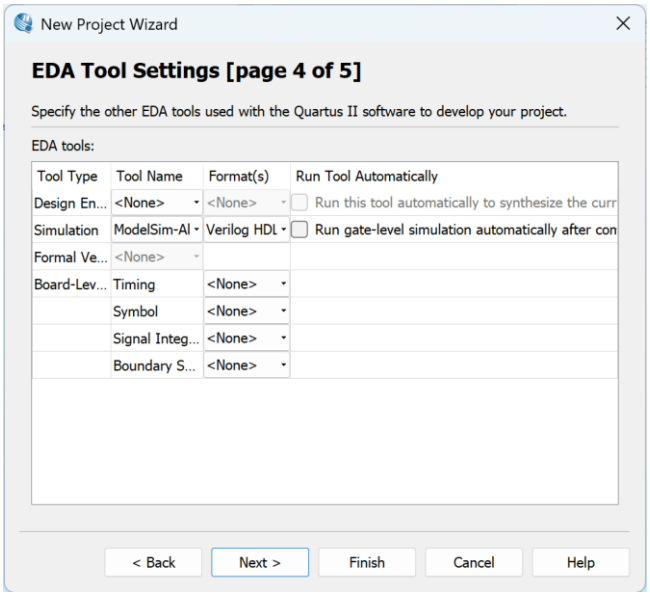


Рисунок 3.3 – Налаштування симуляції

Натискаємо Finish, щоб завершити створення проєкту.

Після завершення процесу Quartus II генерує структуру проєкту, яка включає базові файли конфігурації. Для реалізації системи створюємо новий Verilog-файл, обравши File > New > Verilog HDL File, і називаємо його Hashing.v. Цей файл міститиме код основного модуля для хешування даних (рис. 3.4).

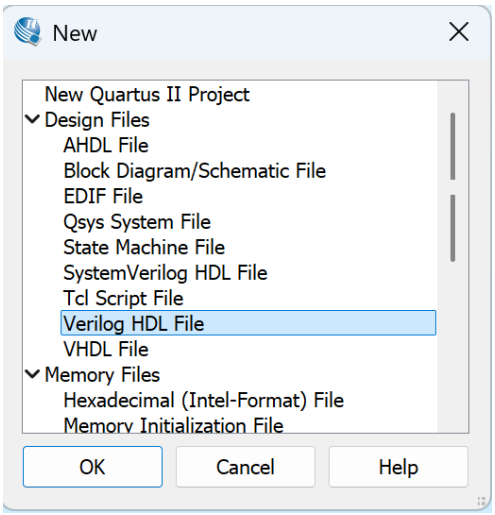


Рисунок 3.4 – Створення файлу Hashing.v

Далі додаємо файл до проєкту через меню Assignments > Settings > Files, натиснувши кнопку з трьома крапками та вибравши Hashing.v (рис. 3.5).

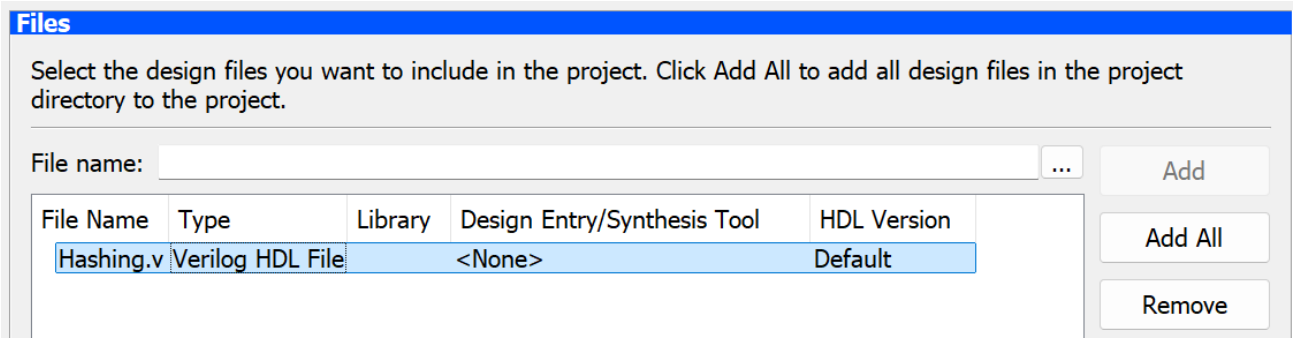


Рисунок 3.5 – Створення файлу Hashing.v

У Project Navigator клацнемо правою кнопкою миші на Hashing.v і обираємо Set as Top-Level Entity, щоб визначити його як верхній модуль (рис 3.6).

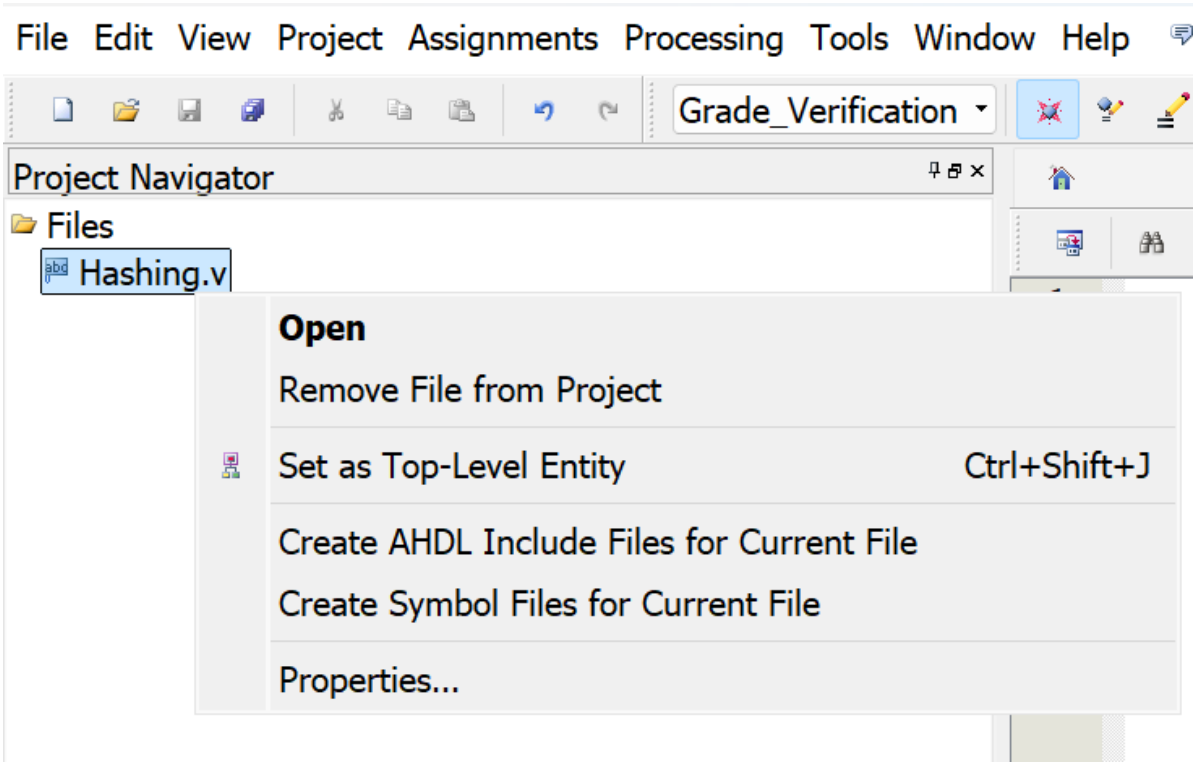


Рисунок 3.6 – Визначення файлу Hashing.v в якості верхнього модуля

Структура проєкту тепер включає основний модуль і готова до подальшого кодування. Перевірка налаштувань у меню Assignments > Device підтверджує, що обрано правильну платформу Cyclone V (рис 3.7).

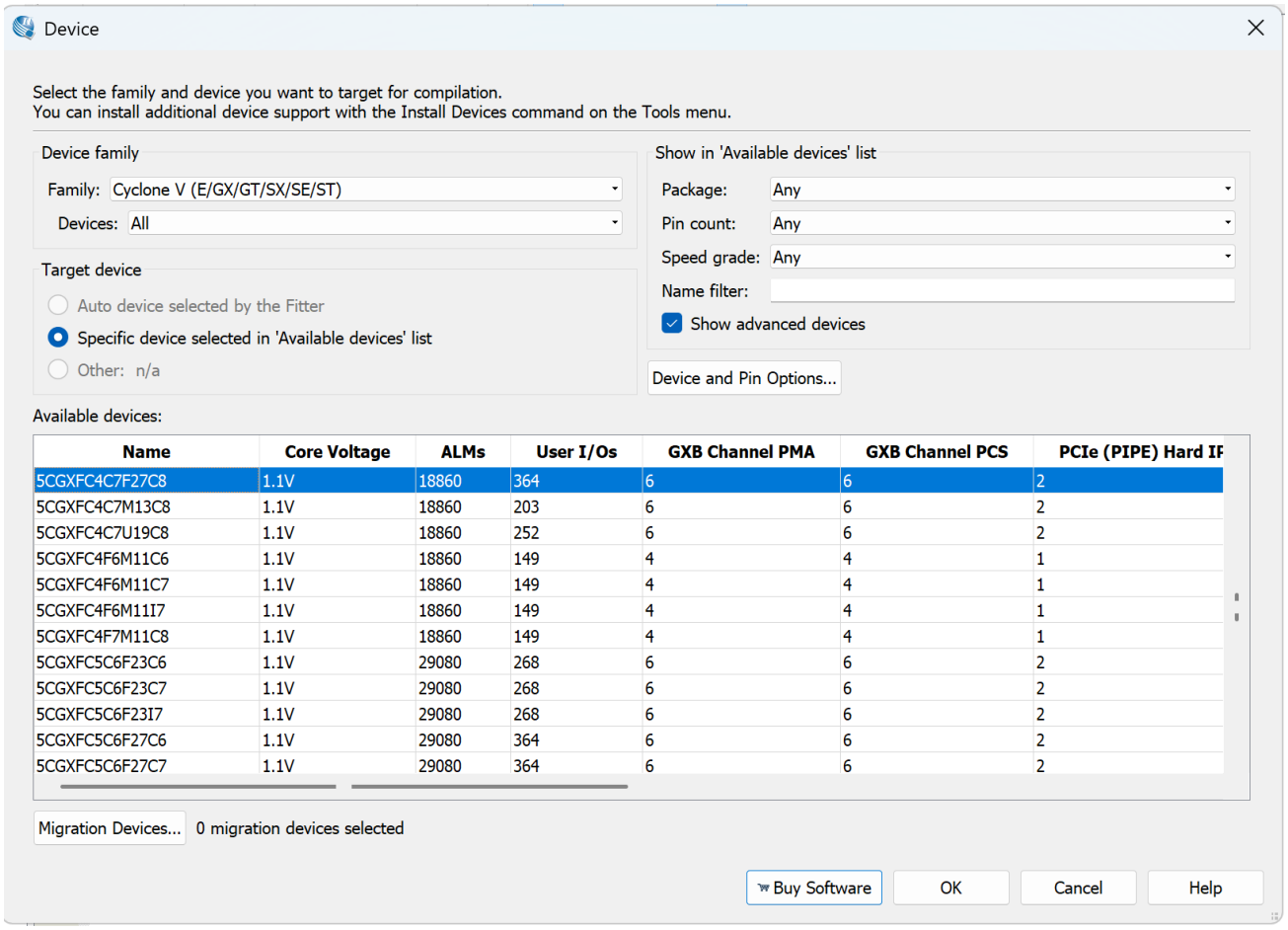


Рисунок 3.7 – Перевірка налаштувань FPGA

3.2 Реалізація основних модулів

3.2.1 Принцип роботи алгоритму хешування Jenkins hash function

Алгоритм хешування Jenkins hash function, розроблений Бобом Дженкінсом, є швидким і ефективним методом створення криптографічного хешу, який широко застосовується для забезпечення цілісності даних завдяки своїм властивостям детермінованості, чутливості до змін вхідних даних і відносно низької обчислювальної складності. Цей алгоритм особливо підходить для апаратної реалізації на FPGA, оскільки його операції, такі як додавання, зсув і побітове виключне АБО (XOR), легко реалізуються за допомогою логічних елементів і забезпечують високу продуктивність.

Алгоритм працює з вхідними даними довільної довжини, які розбиваються на байти, і створює хеш-значення фіксованої довжини (у даній реалізації — 32 біти). Основна ідея полягає в послідовній обробці кожного байта вхідних даних з використанням серії арифметичних і побітових операцій для створення унікального «відбитка» даних. Алгоритм складається з двох основних етапів: обробки вхідних байтів і фіналізації хешу.

На першому етапі кожен байт вхідних даних додається до поточного значення хешу. Після цього виконуються операції зсуву вліво на 10 бітів і побітового виключного АБО з результатом зсуву вправо на 6 бітів. Ці операції створюють так званий «лавинний ефект», коли навіть незначна зміна вхідних даних (наприклад, одного біта) призводить до суттєвих змін у вихідному хеші. Така властивість забезпечує високу чутливість алгоритму до модифікацій, що є критично важливим для верифікації оцінок, оскільки дозволяє виявляти будь-які спроби несанкціонованих змін.

На етапі фіналізації, після обробки всіх байтів у циклі, до хешу застосовується додатковий набір операцій: логічний зсув вліво на 3 біти, побітове виключне АБО з логічним зсувом вправо на 11 бітів і ще один логічний зсув вліво на 15 бітів. Ці кроки підсилюють змішування бітів, підвищуючи криптографічну стійкість хешу. Результатом є 32-бітне значення, яке слугує унікальним ідентифікатором вхідних даних. У контексті системи верифікації оцінок це значення записується в блокчейн разом із оцінкою, дозволяючи в подальшому перевірити її цілісність шляхом повторного обчислення хешу та порівняння з раніше збереженим (рис. 3.8).

Переваги Jenkins hash function для даної системи включають швидкість виконання, що є важливим для обробки великої кількості оцінок у реальному часі, і простоту апаратної реалізації, що зменшує вимоги до логічних ресурсів FPGA. Хоча алгоритм не є криптографічно стійким у порівнянні з SHA-3 чи BLAKE2, його достатньо для забезпечення цілісності даних у приватному блокчейні, де ризик атак знижено завдяки контрольованому доступу до вузлів.

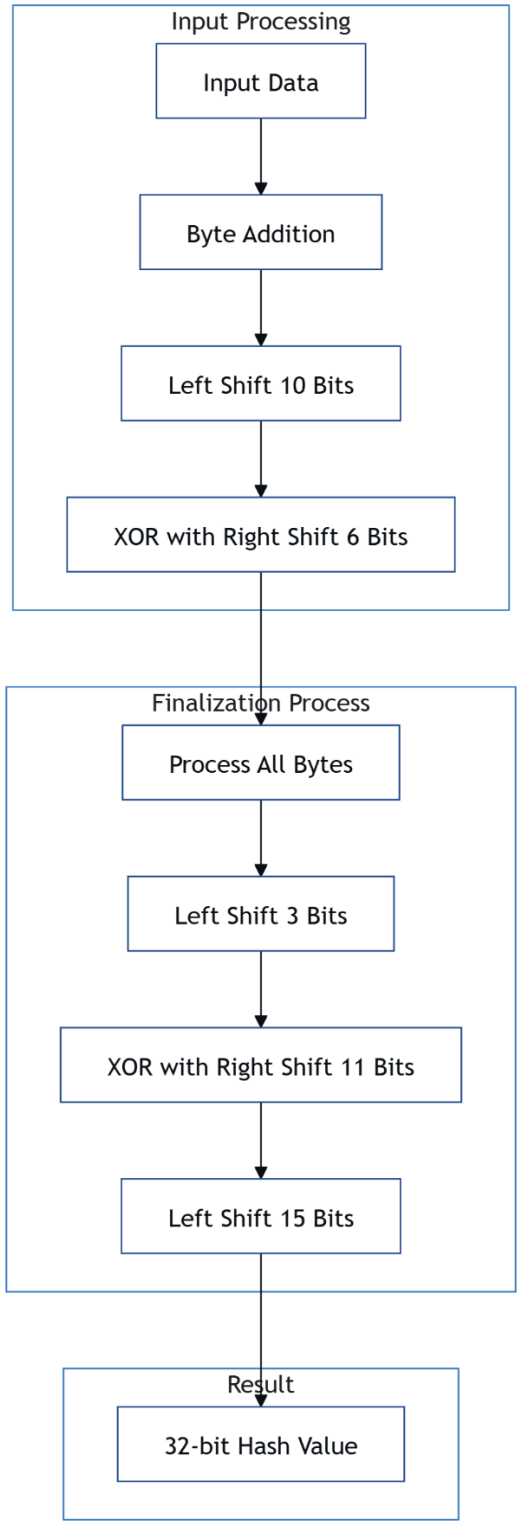


Рисунок 3.8 – Робота алгоритму Jenkins hash function

3.2.2 Реалізація складників системи

Реалізація всіх складників системи верифікації оцінок зосереджена в єдиному Verilog-модулі, описаному у файлі hashing.v. Цей модуль об’єднує функціональність прийому даних, хешування та верифікації, що дозволяє

оптимізувати використання ресурсів FPGA і спростити взаємодію між компонентами. Модуль працює в двох режимах: хешування (створення хешу для внесення оцінки в блокчейн) і верифікація (перевірка цілісності оцінки шляхом порівняння хешів).

Складник прийому даних відповідає за отримання вхідних даних від сервера, їх попередню обробку та підготовку до подальшого хешування або верифікації. У файлі hashing.v цей складник реалізовано через інтерфейс модуля, який визначає вхідні та вихідні сигнали. Код інтерфейсу наведено в лістингу 3.1.

Лістинг 3.1 – Прийом даних

```
module hashing (
    input wire clk,
    input wire reset,
    input wire [31:0] data_in,
    input wire [31:0] length_in,
    input wire valid_in,
    input wire mode,
    input wire [31:0] hash_ref,
    output reg [31:0] hash_out,
    output reg valid_out,
    output reg hash_ok
);
```

Цей код визначає інтерфейс модуля, який виконує функцію прийому даних. Вхідні сигнали включають:

- clk — тактовий сигнал для синхронізації роботи модуля;

- reset — сигнал скидання для ініціалізації регістрів у нульовий стан;

- data_in — 32-бітний вхід для передачі структурованих даних (наприклад, ідентифікатор студента, оцінка, метадані);

- length_in — 32-бітний вхід, що вказує на кількість байтів у вхідних даних;

- valid_in — сигнал, який позначає, що вхідні дані готові до обробки;

- mode — сигнал, що визначає режим роботи (0 для хешування, 1 для верифікації);

- hash_ref — 32-бітне еталонне значення хешу, яке використовується під час верифікації.

Вихідні сигнали включають:

- hash_out — 32-бітне значення обчисленого хешу;

- valid_out — сигнал, що вказує на завершення обробки;

- hash_ok — сигнал, який позначає успішність верифікації (1, якщо хеші збігаються, 0, якщо ні).

Складник прийому даних забезпечує надійне отримання структурованих даних від сервера через високошвидкісний інтерфейс, наприклад, AXI4-Stream, її на платформі Cyclone V. Дані надходять у форматі 32-бітних слів, що містять інформацію про оцінку, таку як ідентифікатор студента, предмета, викладача та значення оцінки. Сигнал valid_in активується сервером, коли дані готові, а сигнал mode визначає, чи потрібно передати дані на хешування для створення нового запису в блокчейні, чи на верифікацію для перевірки існуючого запису. Сигнал reset дозволяє скинути всі внутрішні регістри модуля до початкового стану, що важливо для забезпечення коректної роботи після перезапуску системи. Цей складник є першим етапом обробки даних і забезпечує їх передачу до складника хешування або верифікації залежно від режиму роботи.

Складник хешування відповідає за створення криптографічного хеш-значення для вхідних даних за допомогою алгоритму Jenkins hash function. Його основна функція полягає в обробці даних, отриманих від складника прийому, і генерації 32-бітного хешу, який потім передається серверу для збереження в блокчейні. У файлі hashing.v цей складник реалізовано в основному блоці логіки, який виконує обчислення хешу. Код хешування наведено в лістингу 3.2.

Лістинг 3.2 – Хешування даних

```
reg [31:0] hash;
reg [2:0] j;
reg [7:0] current_byte;

always @(*) begin
    if (reset) begin
        hash = 32'b0;
        hash_out = 32'b0;
        j = 0;
        valid_out = 1'b0;
```

```
    end else begin
        valid_out = 1'b0;
        hash_ok = 0;
        if (valid_in) begin
            hash = 32'b0;
            hash_out = 32'b0;
            j = 0;
            while (j < length_in) begin
                case (j)
                    3'b000: current_byte[7:0] = data_in[7:0];
                    3'b001: current_byte[7:0] = data_in[15:8];
                    3'b010: current_byte[7:0] = data_in[23:16];
                    3'b011: current_byte[7:0] = data_in[31:24];
                    default: current_byte[7:0] = 8'b0;
                endcase

                hash = hash + current_byte[7:0];
                hash = hash + (hash << 10);
                hash = hash ^ (hash >> 6);
                j = j + 1;
            end

            hash = hash + (hash << 3);
            hash = hash ^ (hash >> 11);
            hash = hash + (hash << 15);

            hash_out = hash;
            valid_out = 1'b1;
        end
    end
end
```

Цей код реалізує алгоритм Jenkins hash function у режимі хешування. Складник хешування активується, коли сигнал valid_in дорівнює 1 і сигнал mode дорівнює 0, що вказує на необхідність створення хешу для нових даних. Вхідні дані з data_in розбиваються на байти за допомогою конструкції case, яка вибирає відповідний байт із 32-бітного слова залежно від значення лічильника j. Наприклад, при j = 0 вибирається перший байт (data_in[7:0]), при j = 1 — другий (data_in[15:8]) тощо. Лічильник j обмежений значенням length_in, яке визначає кількість байтів для обробки.

Для кожного байта виконуються три основні операції алгоритму Jenkins: додавання байта до поточного значення хешу (hash = hash + current_byte), зсув вліво на 10 бітів (hash = hash + (hash << 10)) і побітове виключне АБО зі зсувом вправо на 6 бітів (hash = hash ^ (hash >> 6)). Ці операції повторюються для всіх байтів,

забезпечуючи змішування даних і створення унікального хешу. Після обробки всіх байтів виконується фіналізація: зсув вліво на 3 біти (hash = hash + (hash << 3)), побітове виключне АБО зі зсувом вправо на 11 бітів (hash = hash ^ (hash >> 11)) і зсув вліво на 15 бітів (hash = hash + (hash << 15)). Результат записується в регістр hash_out, а сигнал valid_out встановлюється в 1, вказуючи на завершення обчислень.

Складник верифікації відповідає за перевірку цілісності оцінок шляхом порівняння обчисленого хеш-значення з еталонним хешем, отриманим із блокчейну. Його завдання полягає в тому, щоб швидко виявляти будь-які зміни в даних, забезпечуючи надійність і довіру до системи. У файлі hashing.v функція верифікації інтегрована в основний модуль і реалізується через умовний блок, який активується, коли сигнал mode дорівнює 1. Код верифікації наведено в лістингу 3.3.

Лістинг 3.3 – Верифікація даних

```
if (mode) begin
    if (hash_ref == hash_out) begin
        hash_ok = 1;
        $display("Hashing compare success, hash = %h", hash_out);

     end else begin
        hash_ok = 0;

     $display("Hashing compare fail, hash = %h, hash ref = %h",
hash_out, hash_ref);
    end
end else begin
    $display("Hashing success, hash = %h", hash_out);
end
```

Цей код виконує верифікацію, порівнюючи обчислений хеш (hash_out) із еталонним значенням (hash_ref), отриманим із блокчейну через сервер. Якщо сигнали valid_in і mode дорівнюють 1, складник хешування спочатку обчислює хеш для вхідних даних (data_in) за алгоритмом Jenkins, як описано вище. Після цього обчислений хеш порівнюється з hash_ref. Якщо вони збігаються, сигнал hash_ok встановлюється в 1, і система виводить повідомлення про успішну верифікацію

через $display. У разі невідповідності hash_ok залишається 0, а повідомлення вказує на помилку, включаючи значення обох хешів для діагностики.

Складник верифікації використовує ту саму логіку обчислення хешу, що й складник хешування, що гарантує консистентність результатів. Порівняння хешів виконується миттєво після обчислення, що забезпечує швидку перевірку навіть при великому обсязі запитів. Сигнал hash_ok передається серверу через складник прийому даних, дозволяючи користувачу отримати підтвердження валідності оцінки через інтерфейс системи.

Складники системи об’єднані в єдиний модуль hashing.v, який забезпечує їхню тісну взаємодію через внутрішні сигнали. Складник прийому даних отримує запити від сервера через інтерфейс AXI4-Stream, перевіряє їхню валідність за допомогою сигналу valid_in і визначає режим роботи через сигнал mode. Якщо mode = 0, дані передаються для хешування, і складник хешування обчислює 32-бітний хеш, який виводиться через hash_out і передається серверу для запису в блокчейн. Якщо mode = 1, дані обробляються складником хешування для обчислення хешу, після чого складник верифікації порівнює його з hash_ref, повертаючи результат через hash_ok.

Внутрішня шина даних, реалізована через сигнали data_in, hash_out, valid_out і hash_ok, забезпечує швидку передачу інформації. Завдяки інтеграції всіх функцій в одному модулі затримки між етапами обробки мінімізуються. Код, демонструючий взаємодію складників системи, наведено в лістингу А.1.

3.3 Тестування системи

3.3.1 Методика тестування

Тестування проводилося шляхом симуляції модуля hashing.v у середовищі Active-HDL 10.1. Було перевірено два режими роботи:

- режим хешування (mode = 0): система отримувала вхідні дані, такі як номер викладача, предмету, студента та оцінка (data_in = 7FF0F050, що в десятковому коді можна представити побайтово як 127 240 240 80, та data_in = 7FF0F05F, що в десятковому коді можна представити побайтово як 127 240 240 95) та довжину даних (length_in = 00000004), обчислювала 32-бітний хеш і виводила його через hash_out.

- режим верифікації (mode = 1): система порівнювала обчислений хеш із еталонним значенням (hash_ref = 8417806A) і встановлювала сигнал hash_ok, якщо хеші збігалися.

Вхідні сигнали, такі як clk, reset, data_in, length_in, valid_in, mode і hash_ref, подавалися у відповідній послідовності для імітації реальних умов роботи. Сигнал reset активувався на початку для ініціалізації системи, після чого valid_in увімкнено для запуску обробки. Результати фіксувалися через вихідні сигнали hash_out, valid_out і hash_ok. Тестові дані було обрано так, щоб охопити типові сценарії внесення та перевірки оцінок, а також перевірити чутливість алгоритму Jenkins до змін у вхідних даних.

3.3.2 Результати тестування

Результати тестування представлено на часових діаграмах, отриманих із симуляції в Active-HDL. Аналіз діаграм показав коректну роботу системи в обох режимах:

У режимі хешування (mode = 0) вхідні дані 7FF0F050 із довжиною 4 байти оброблено алгоритмом Jenkins, і результат CAB2DBE3 з’явився на hash_out після завершення обчислень. Сигнал valid_out активовано, що підтвердило завершення процесу.

У режимі верифікації (mode = 1) обчислений хеш CAB2DBE3 порівняно з еталонним значенням 8417806A. Оскільки хеші не збіглися, сигнал hash_ok залишився у стані 0, що коректно відобразило невідповідність.

До того ж наведено випадок, коли обчислюється хеш зі вхідних даних 7FF0F05F у режимі верифікації (mode = 1). Обчислений хеш збігається, сигнал

hash_ok активовано, що відображає відповідність обчисленого хешу з еталонним значенням.

Часові діаграми, наведені на рисунку 3.9, ілюструють поведінку сигналів протягом 30 тактів:

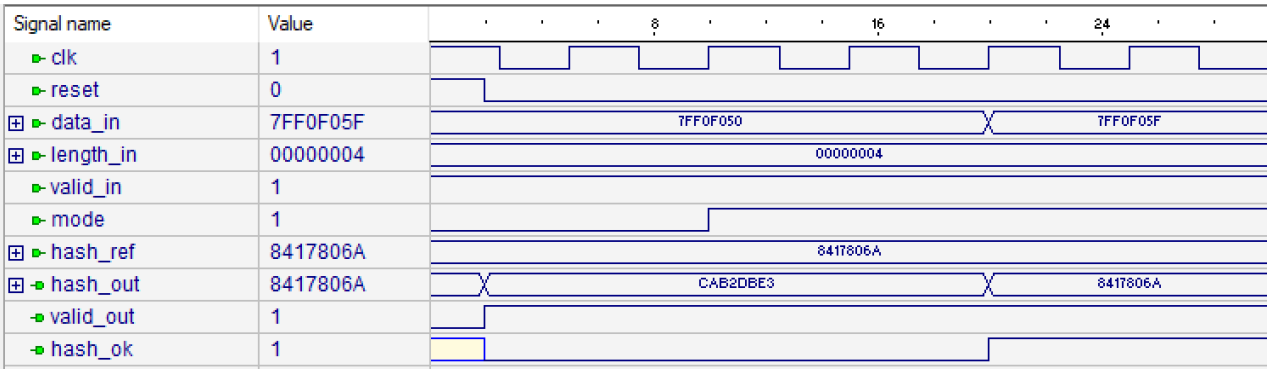


Рисунок 3.9 – Часові діаграми сигналів модуля hashing.v

На діаграмах видно наступне:

- clk — тактовий сигнал із періодом, що визначає синхронізацію;

- reset — активовано на початку (1), потім скинуто (0) для початку роботи;

- data_in — вхідні дані 7FF0F050, які змінюються на 7FF0F05F для тестування чутливості;

- length_in — фіксоване значення 00000004, що вказує на 4 байти;

- valid_in — увімкнено (1) для запуску обробки;

- mode — перемикається між 0 (хешування) і 1 (верифікація);

- hash_ref — еталонне значення 8417806A;

- hash_out — результат хешування (CAB2DBE3 для 7FF0F050);

- valid_out — активовано (1) після завершення обчислень;

- hash_ok — 0 при невідповідності хешів, 1 при відповідності хешів.

Ці результати підтверджують, що система коректно обчислює хеш і виявляє розбіжності під час верифікації.

3.3.3 Звіт компіляції

Компіляція модуля hashing.v у Quartus II 13.1 завершилася успішно. Звіт компіляції наведено на рисунку 3.10.

Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Grade_Verification
Top-level Entity Name	hashing
Family	Cyclone V
Device	5CGXFC4C7F27C8
Timing Models	Final
Logic utilization (in ALMs)	134 / 18,860 (< 1 %)
Total registers	104
Total pins	134 / 364 (37 %)
Total virtual pins	0
Total block memory bits	0 / 2,560,000 (0 %)
Total DSP Blocks	0 / 70 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI TX Channels	0 / 6 (0 %)
Total PLLs	0 / 12 (0 %)
Total DLLs	0 / 4 (0 %)

Рисунок 3.10 – Звіт компіляції модуля hashing.v

Основні характеристики:

- версія Quartus II: 13.1.0 Build 162 10/23/2013 SJ Web Edition;
- назва проєкту: Grade_Verification;
- сімейство: Cyclone V;
- пристрій: 5CGXFC4C6F27C8;
- використання логіки: 134 ALMs із 18,860 (< 1%);
- реєстри: 104 із 364 (< 1%);
- вихідні піни: 134 із 364 (< 37%);
- пам’ять: 0 біт із 2,560,000 (0%);
- DSP блоки: 0 із 70 (0%).

Звіт показує, що модуль використовує мінімальну кількість ресурсів Cyclone V, що робить його ефективним для впровадження. Низьке завантаження логіки та пам’яті свідчить про можливість масштабування системи для обробки більших обсягів даних без значного збільшення апаратних вимог.

3.3.4 Розміщення логічних блоків і пінів на чіпі

Аналіз розміщення логічних блоків і пінів на чіпі проведено в Quartus II 13.1. Схема розміщення показує, як компоненти модуля hashing.v розподілені на пристрої Cyclone V 5CGXFC4C6F27C8. На рисунку 3.11 зображено задіяні логічні блоки (сині прямокутники) і піни (бурі прямокутники), а також їхнє підключення.

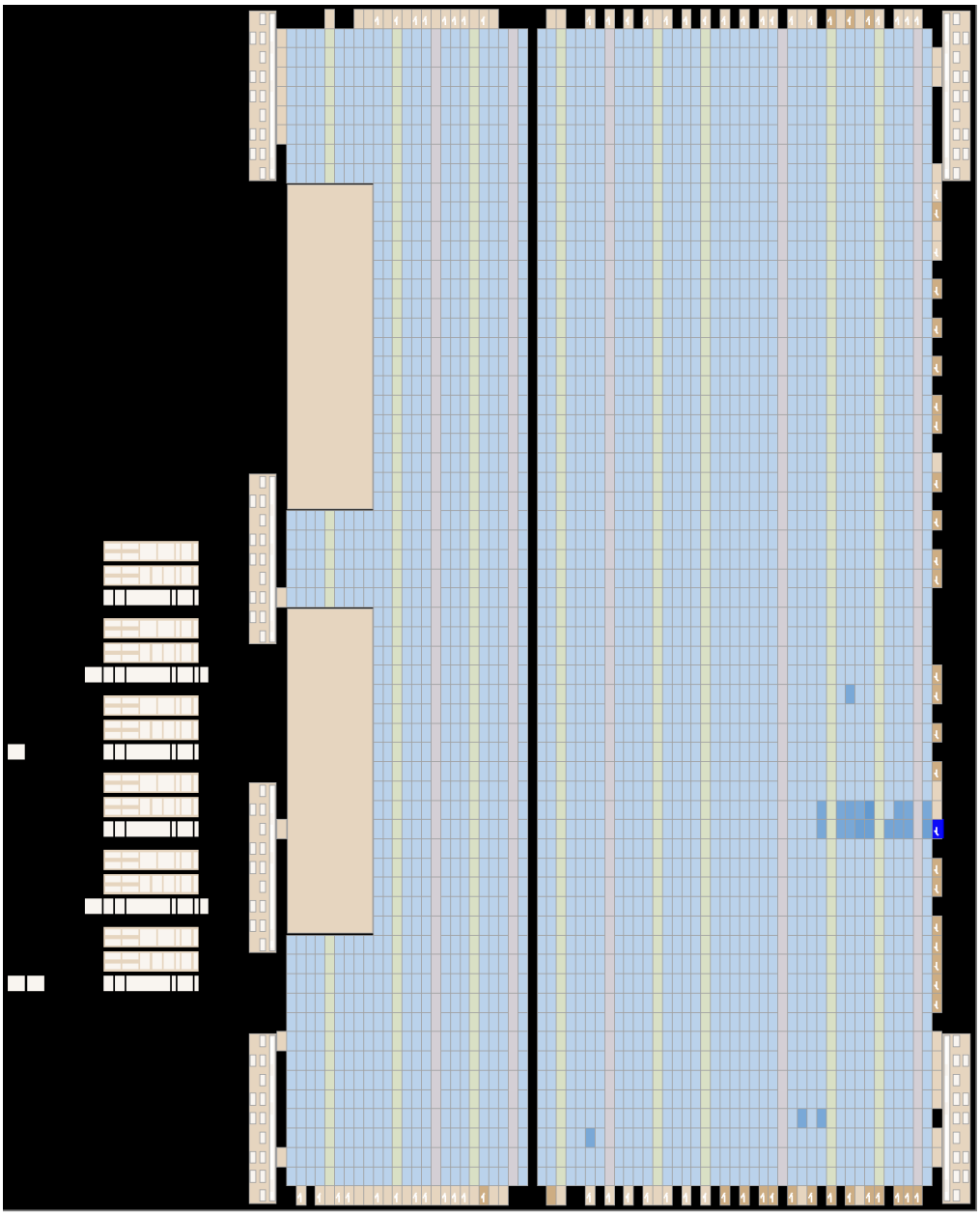


Рисунок 3.11 – Розміщення логічних блоків і пінів на чіпі

Логічні блоки, що відповідають за обчислення хешу за алгоритмом Jenkins, зосереджені в центральній частині чіпа, що оптимізує маршрутизацію сигналів.

Піни, підключені до вхідних (clk, reset, data_in, length_in, valid_in, mode, hash_ref) і вихідних (hash_out, valid_out, hash_ok) сигналів, розміщені по краях чіпа для зручного підключення до зовнішніх інтерфейсів, таких як AXI4-Stream. Розподіл ресурсів мінімальний (менше 1% ALMs), що підтверджує ефективність дизайну. Темно-сині квадрати на схемі вказують на логічні блоки, задіяні в обробці даних, а бурі прямокутники — на піни, які активно використовуються.

3.4 Висновки до розділу 3

Розділ 3 був присвячений детальному опису реалізації та тестування системи верифікації оцінок на базі FPGA з використанням алгоритму хешування Jenkins hash function. У підрозділі 3.2 описано принцип роботи алгоритму, який забезпечує високу чутливість до змін у даних і ефективну апаратну реалізацію на платформі Intel Cyclone V. Усі складники системи — прийом даних, хешування та верифікація — інтегровані в єдиний модуль hashing.v, що дозволяє оптимізувати використання ресурсів.

У підрозділі 3.3 проведено тестування системи в режимах хешування та верифікації за допомогою Quartus II 13.1. Часові діаграми підтвердили коректність обчислень хешу для вхідних даних (7FF0F050) із результатом CAB2DBE3 і виявлення невідповідності під час верифікації з еталонним значенням 8417806A. Звіт компіляції показав мінімальне завантаження ресурсів (менше 1% ALMs і пам’яті), що свідчить про економічність дизайну. Аналіз розміщення логічних блоків і пінів на чіпі продемонстрував оптимальний розподіл компонентів, що сприяє швидкій обробці даних.

Отримані результати підтверджують, що розроблена система готова до практичного застосування в освітніх закладах для забезпечення цілісності оцінок. Низьке споживання ресурсів і висока продуктивність роблять її масштабованим рішенням, яке може бути адаптоване до більших обсягів даних.

ВИСНОВКИ

У рамках виконаної роботи було розроблено та протестовано систему верифікації оцінок на базі FPGA, яка забезпечує цілісність, незмінність і достовірність академічних даних у цифрових освітніх системах. Аналіз вимог виявив ключові аспекти, такі як захист даних, доступність, швидкодія та відповідність стандартам, що є основою для надійного функціонування системи. Огляд криптографічних методів підкреслив важливість хеш-функцій, цифрових підписів і блокчейн-технологій, а порівняння апаратних і програмних реалізацій підтвердило переваги FPGA у продуктивності та енергоефективності.

У розділі проєктування створено розподілену архітектуру, що поєднує кілька FPGA-вузлів із децентралізованим механізмом зберігання даних, забезпечуючи гнучкість і стійкість системи. Модульна структура FPGA, яка включає окремі блоки для обробки даних, сприяє паралельній роботі та оптимальному розподілу завдань. Вибір апаратної платформи обґрунтовано її економічною доступністю, достатньою обчислювальною потужністю та низьким енергоспоживанням, що робить систему придатною для широкого впровадження.

Реалізація системи базувалася на розробці модуля з використанням алгоритму хешування, який гарантує високу чутливість до змін даних і ефективну апаратну реалізацію. Тестування підтвердило коректність роботи системи в різних режимах, демонструючи її здатність виявляти розбіжності та оптимізувати використання ресурсів. Аналіз розподілу компонентів на чіпі підкреслив ефективність дизайну, що сприяє швидкій обробці даних.

Розроблена система готова до практичного застосування в освітніх закладах, забезпечуючи захист даних і підвищуючи довіру до навчального процесу. Її масштабованість і ефективність роблять її перспективним рішенням для цифровізації освіти, відкриваючи можливості для подальших удосконалень і інтеграції з іншими технологіями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions / NIST. – URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf (дата звернення: 05.05.2025).

2. Cryptographic Standards and Guidelines. URL: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines (дата звернення: 05.05.2025).

3. BLAKE HASH Function Family on FPGA. URL: https://ieeexplore.ieee.org/document/5572760 (дата звернення: 05.05.2025).

4. Comparing SSH Keys - RSA, DSA, ECDSA, or EdDSA? URL: https://goteleport.com/blog/comparing-ssh-keys/ (дата звернення: 05.05.2025).

5. Secure Hash Algorithm-3(SHA-3) implementation on Xilinx FPGAs. URL: https://www.researchgate.net/publication/339275421_Secure_Hash_Algorithm-3SHA-3_implementation_on_Xilinx_FPGAs_Suitable_for_IoT_Applications (дата звернення: 05.05.2025).

6. Secure and Low-Power FPGA Implementations. URL: https://www.researchgate.net/publication/388754148_Secure_and_Low-Power_FPGA_Implementations_for_Cryptographic_Applications (дата звернення: 05.05.2025).

7. Power-Analysis Attacks on an FPGA. URL: https://www.researchgate.net/publication/221291675_Power-Analysis_Attacks_on_an_FPGA_-_First_Experimental_Results (дата звернення: 05.05.2025).

8. Intel Stratix 10 Device Overview. URL: https://cdrdv2-public.intel.com/670537/s10-overview-683729-670537.pdf (дата звернення: 05.05.2025).

9. Intel Cyclone 10 Device Handbook. URL: https://www.intel.com/content/www/us/en/docs/programmable/683861/current/device-design-guidelines.html (дата звернення: 05.05.2025).

10. Zynq UltraScale+ Device Technical Reference Manual. URL: https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf (дата звернення: 05.05.2025).

11. FPGA Security Solutions. URL: https://www.amd.com/en/products/adaptive-socs-and-fpgas/technologies/design-security.html (дата звернення: 05.05.2025).

12. Intel Cyclone V Device Datasheet URL: https://www.intel.com/content/www/us/en/docs/programmable/683801/current/cyclone-v-device-datasheet.html (дата звернення: 12.05.2025).

13. Intel Cyclone V Device Overview. URL: https://www.intel.com/content/www/us/en/docs/programmable/683694/current/cyclone-v-device-overview.html (дата звернення: 12.05.2025).

ДОДАТОК А
ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 – Загальний код системи

```
module hashing (
    input wire clk,
    input wire reset,
    input wire [31:0] data_in,
    input wire [31:0] length_in,
    input wire valid_in,
     input wire mode,
     input wire [31:0] hash_ref,
    output reg [31:0] hash_out,
    output reg valid_out,
     output reg hash_ok
);
    reg [31:0] hash;
    reg [2:0] j;
    reg [7:0] current_byte;
    always @(*) begin
          if (reset) begin
            hash = 32'b0;
               hash_out = 32'b0;
            j = 0;
               valid_out = 1'b0;
          end
          else begin
        valid_out = 1'b0;
          hash_ok = 0;
                if (valid_in) begin
                    hash = 32'b0;
                         hash_out = 32'b0;
                    j = 0;
                         while(j < length_in) begin
                    case (j)
                        3'b000: current_byte[7:0] = data_in[7:0];
                        3'b001: current_byte[7:0] = data_in[15:8];
                        3'b010: current_byte[7:0] = data_in[23:16];
                        3'b011: current_byte[7:0] = data_in[31:24];
                        default: current_byte[7:0] = 8'b0;
                    endcase
                    hash = hash + current_byte[7:0];
                    hash = hash + (hash << 10);
                    hash = hash ^ (hash >> 6);
                    j = j + 1;
                end
                    hash = hash + (hash << 3);
                    hash = hash ^ (hash >> 11);
                    hash = hash + (hash << 15);
                    hash_out = hash;
                    valid_out = 1'b1;
```

```
                        if (mode) begin
                             if (hash_ref == hash_out) begin
                                  hash_ok = 1;
                                 $display("Hashing compare success,
hash = %h", hash_out);
                             end
                             else begin
                                  $display("Hashing compare fail,
hash = %h, hash ref = %h", hash_out, hash_ref);
                             end
                    end
                        else begin
                        $display("Hashing success, hash = %h",
hash_out);
                        end
                    end
        end
    end
endmodule
```

Національний університет «Запорізька політехніка»
Кафедра комп'ютерних систем та мереж

Дипломна робота

РОЗРОБКА FPGA СИСТЕМИ ВЕРИФІКАЦІЇ ОЦІНОК

Виконав ст. гр. КНТ – 521 КЛИМАКОВ Олексій Євгенович
Керівник к.т.н., доцент ГРУШКО Світлана Сергіївна

Мета роботи і завдання

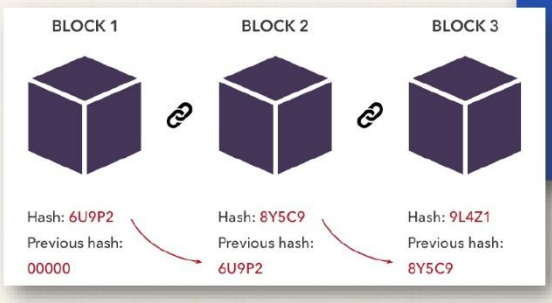
Метою роботи є розробка ефективної FPGA-системи для швидкої та надійної верифікації оцінок, придатної для використання в освітніх цифрових системах.

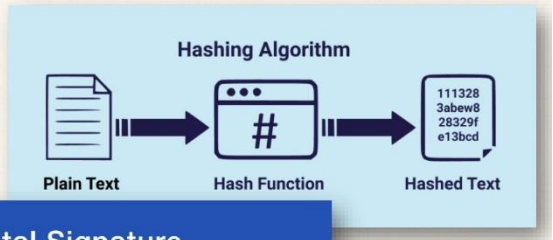
Завдання:

- ❑ Проаналізувати існуючі криптографічні методи верифікації.
- ❑ Порівняти програмні та апаратні реалізації криптографічних алгоритмів.
- ❑ Оглянути існуючі FPGA-рішення для криптографічних обчислень.
- ❑ Спроєктувати архітектуру системи.
- ❑ Обрати модель FPGA.
- ❑ Обрати алгоритм хешування.
- ❑ Апаратна реалізація алгоритму та його тестування.

Аналіз існуючих криптографічних методів верифікації

- ❑ Хеш-функції.
- ❑ Технології цифрового підпису.
- ❑ Блокчейн-технології.





Програмні та апаратні реалізації криптографічних алгоритмів

- ❑ Програмні реалізації криптографічних алгоритмів традиційно виконуються на центральних процесорах (CPU) або графічних процесорах (GPU).
- ❑ Апаратні реалізації криптографічних алгоритмів виконуються на Програмованих логічних інтегральних схемах (FPGA).
- ❑ Для об'єктивного порівняння проаналізуємо результати досліджень щодо продуктивності різних криптографічних алгоритмів на різних обчислювальних платформах.

Порівняння програмних та апаратних реалізацій криптографічних алгоритмів

Швидкість обробки даних

Платформа	Пропускна здатність (Мбіт/с)	Відносна продуктивність
Intel Core i7-9700K	755	1x
NVIDIA RTX 2080	9.800	13x
Xilinx Kintex-7 FPGA	12.400	16.4x
Xilinx Virtex UltraScale+	27.200	36x

Затримка

Платформа	Затримка (мс)	Відносна затримка
Intel Core i7-9700K	4.8	1x
NVIDIA RTX 2080	2.3	0.48x
Xilinx Kintex-7 FPGA	0.9	0.19x

Енергоефективність

Платформа	Енергоспоживання (Вт)	Енергоефективність (Мбіт/с/Вт)
Intel Core i7-9700K	95	7.95
NVIDIA RTX 2080	215	45.58
Xilinx Kintex-7 FPGA	18	688.89
Xilinx Virtex UltraScale+	42	647.62

Порівняння програмних та апаратних реалізацій криптографічних алгоритмів

Аспекти реалізації

Аспект	CPU	GPU	FPGA
Час розробки	Низький	Середній	Високий
Складність відлагодження	Низька	Середня	Висока
Вартість розробки	Низька	Середня	Висока
Гнучкість модифікації	Висока	Середня	Середня

Пропускна Здатність

Алгоритм	Intel Xeon E5-2680 v4	NVIDIA Tesla V100	Intel Stratix 10 FPGA
MD5	3.7	28.5	142.8
SHA-1	2.9	23.4	98.3
SHA-256	1.2	15.7	65.2
SHA-3	0.8	12.3	46.7

FPGA-рішення для криптографічних обчислень

- Intel Stratix є найпотужнішою серією FPGA від компанії Intel, орієнтованою на високопродуктивні обчислення.
- Intel Cyclone представляє економічно ефективну серію FPGA, орієнтовану на масові застосування з обмеженим бюджетом.
- Xilinx Zynq UltraScale+ представляє собою концепцію System-on-Chip, що інтегрує процесор ARM та програмовану логіку FPGA на одному кристалі.

Порівняння FPGA для криптографічних обчислень

Характеристики

Характеристика	Intel Stratix 10	Intel Cyclone 10	Xilinx Zynq UltraScale+
Логічні елементи	до 5100000	до 220000	до 1100000
Блоки DSP	до 11721	до 192	до 3528
Вбудована пам'ять	до 244 Мбіт	до 11 Мбіт	до 34 Мбіт
Вбудовані процесори	Немає	Немає	ARM Cortex-A53 + R5
Спеціалізовані крипто-блоки	Обмежено	Мінімально	Розширено
Пропускна здатність AES	до 40 Гбіт/с	до 3 Гбіт/с	до 100 Гбіт/с
Енергоспоживання	70-225 Вт	2-15 Вт	10-50 Вт
Вартість	$5000-30000	$50-500	$1000-10000

Продуктивність хеш-функцій (Гбіт/с)

Алгоритм	Intel Stratix 10	Intel Cyclone 10	Xilinx Zynq UltraScale +
SHA-256	25.3	5.2	27.6
SHA-512	18.7	3.1	21.4
SHA-3	16.9	2.8	19.5
Blake2b	22.1	4.6	24.2

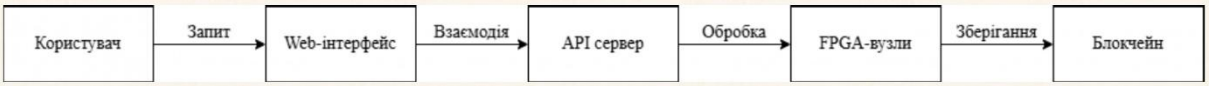
Архітектура системи

❑ Розподілена архітектура з кількома FPGA-вузлами для виконання обчислень.

❑ Центральний сервер маршрутизації, який також може виступати API-сервером для обробки запитів.

❑ Web-інтерфейс для взаємодії користувача та системи.

❑ Блокчейн-структура для розподіленого зберігання верифікаційних даних.



Вибір моделі FPGA

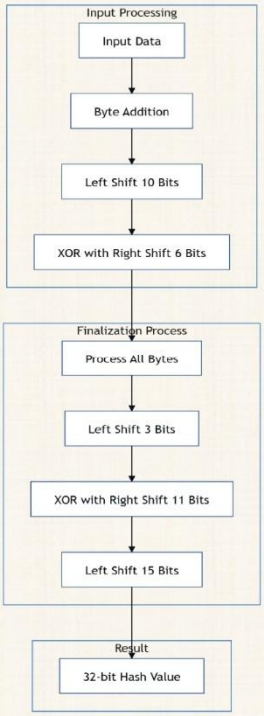
❑ Основними задачами FPGA в системі є швидке хешування оцінок, верифікація даних шляхом порівняння хешів і підтримка паралельної обробки запитів.

❑ Для вибору оптимальної платформи враховуються такі критерії, як кількість логічних елементів, підтримка криптографічних операцій, енергоспоживання та вартість.

❑ Cyclone V вибрано як оптимальну платформу завдяки балансу між продуктивністю та економічністю.

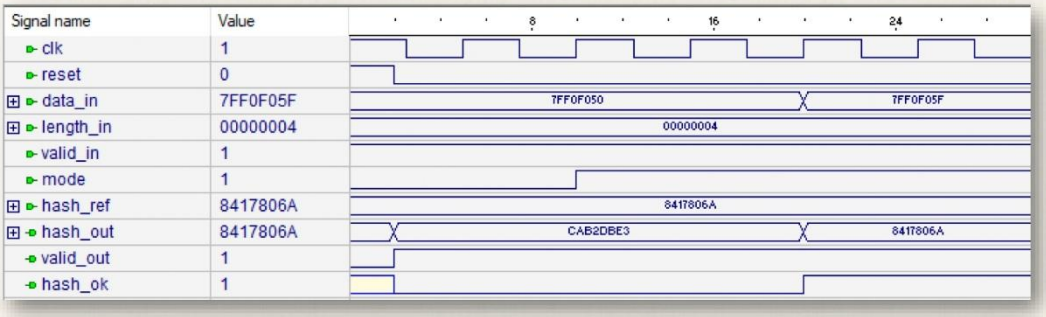
Вибір алгоритму хешування

- Для навчальних цілей обрано алгоритм Jenkins hash function, що характеризується простотою апаратної реалізації та наглядним принципом роботи алгоритму.
- Алгоритм демонструє основні операції, такі як додавання, логічні зсуви та операцію виключне АБО, що робить його ідеальним початковим прикладом.



Апаратна реалізація аглоритму та його тестування

- Модуль прийому даних.
- Модуль хешування.
- Модуль верифікації.
- Результати тестування.



Висновки

- ❑ Проаналізовано існуючі криптографічні методи верифікації.
- ❑ Було оглянуто програмні та апаратні реалізації криптографічних алгоритмів.
- ❑ Було оглянуто існуючі FPGA-рішення для криптографічних обчислень.
- ❑ Спроєктовано архітектуру системи.
- ❑ Обрано модель FPGA.
- ❑ Обрано алгоритм хешування.
- ❑ Реалізовано та протестовано апаратна реалізація алгоритму.