МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ

«____»_____ 2023 р.

На правах рукопису

УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний модуль шифрування даних у багатокористувацькій грі

Виконавець: Світлана РЕПІНА

Керівник: к.т.н., доцент Микола БРАІЛОВСЬКИЙ

Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА

Нормоконтролер: к.т.н., доцент Микола БРАІЛОВСЬКИЙ

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
здобувача вищої освіти Репіної Світлани Олександрівни

1. Тема: *Програмний модуль шифрування даних у багатокористувацькій грі* затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: дослідити основні поняття комп'ютерної гри та дати огляд підходів до їх класифікації; розглянути програмну та апаратну реалізацію на ПЛІС; описати із творення програмного модулю шифрування даних у багатокористувацькій грі; на базі одного з алгоритмів створити модуль шифрування даних, наведено алгоритм його роботи.
4. Зміст пояснювальної записки: загальна характеристика комп'ютерних ігор; аналіз існуючих аналогів та двигунів для багатокористувацької гри; загальна характеристика процесу шифрування даних; розробка програмного модулю

шифрування даних у багатокористувацькій грі; охорона навколишнього середовища.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	17.10.2023-20.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	21.10.2023	*Виконано*
4.	Збір інформації	22.10.2023-31.10.2023	*Виконано*
5.	Аналіз загальної характеристики комп'ютерних ігор	01.11.2023-07.11.2023	*Виконано*
6.	Аналіз існуючих аналогів та двигунів для багатокористувацької гри	08.11.2023-14.11.2023	*Виконано*
7.	Аналіз загальної характеристики процесу шифрування даних	15.11.2023-04.12.2023	*Виконано*
8.	Розробка програмного модулю шифрування даних у багатокористувацькій грі	05.12.2023-11.12.2023	*Виконано*
9.	Апробація роботи на науково-практичній конференції «Прикладні системи та технології в інформаційному суспільстві»	05.10.2023	*Виконано*
10.	Перевірка на антиплагіат	12.12.2023	*Виконано*
11.	Оформлення і друк пояснювальної записки	12.12.2023-12.12.2023	*Виконано*
12.	Оформлення презентації	18.12.2023	*Виконано*
13.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Світлана РЕПІНА

(підпис, дата)

Керівник кваліфікаційної роботи Микола БРАЇЛОВСЬКИЙ

(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота на **тему:** Програмний модуль шифрування даних у багатокористувацькій грі.

Дипломна робота містить: 121 сторінок друкованого тексту, 25 рисунків та нараховує 50 джерел використаної інформації.

Мета роботи – розробити програмний модуль шифрування даних у багатокористувацькій грі.

На поточний момент комп'ютерна техніка досягла такого рівня розвитку, що дозволяє програмістам розробляти дуже реалістичні ігри з хорошим графічним і звуковим оформленням. Зроблено огляд основних класифікацій комп'ютерних ігор (по ігровій платформі, по графіці, за змістом, за видавничими критеріями, за типом їх поширення та кількістю гравців). Встановлено, що блоковий симетричний шифр ДСТУ 7624:2014 «Калина» має доказову квантову захищеність та відповідає вимогам сучасних криптосистем. Перспективністю стандарту ДСТУ 7624:2014 («Калина») є те, що алгоритм ДСТУ 7624:2014 забезпечує вищий рівень криптографічної стійкості і аналогічну або вищу швидкодію.

Розглянуто програмну (бібліотека Cryptopp) та апаратну реалізацію на ПЛІС. Виконана програмна реалізація та симуляція в середовищі Aldec Riviera-Pro, що передбачала перетворення вихідного коду шифрування «Калина» з мови C++ у мову C, подальша її симуляція у середовищі XILINX Vivado HLS та синтез коду VHDL на мові Verilog. Детально описано особливості симулювання коду у середовищі Xilinx Vivado. hls.

Описано із творення програмного модулю шифрування даних у багатокористувацькій грі. Проведено аналіз архітектури програмної реалізації, виконано розробку структури бази даних та use case діаграми модулю шифрування даних.

Здійснено опис цільової аудиторії та функціональність об’єкту, приведено загальні відомості проєкту та способи його реалізації. Розроблено структуру програмного комплексу, визначено його функціональні можливості з наведенням пояснень. Реалізовано алгоритми шифрування / дешифрування з використанням сплайнів. Розроблено модуль шифрування даних, наведено алгоритм його роботи. Наведено детальну, покрокову інструкцію з використання розробленого продукту.

Ключові слова: криптографія, шифрування даних, інтерполяція, інформаційна безпека, багатокористувацька гра, модуль шифрування, xilinx vivado hls, c++, тестування.

ЗМІСТ

ВСТУП

З появою комп'ютерів, на світ з'явилися комп'ютерні ігри, які відразу знайшли своїх прихильників. Гравці комп'ютерів швидко адаптуються до переміщень з одного віртуального світу в інший, реагуючи на нові та незвичайні ситуації. Важливо зауважити, що в комп'ютерних іграх можуть бути моделювані різноманітні обставини, від щоденних прогулянок з собакою до епічних космічних битв. На сьогоднішній день, існує безліч комплектуючих, спеціально розроблених для сучасних ігор. Усі гри розробляються з урахуванням останніх інновацій у сфері комп'ютерної індустрії, постійно вдосконалюючи рівень реалізму графіки та звукового супроводу. [4].

Актуальність дослідження. У сучасний момент рівень розвитку комп'ютерної техніки дозволяє програмістам створювати надзвичайно реалістичні ігри з високоякісною графікою та звуковим супроводом. Комп'ютерні ігри вже так сильно вкоренилися в наше повсякденне життя, що тепер майже неможливо уявити комп'ютер, на якому не було б хоча б однієї гри. Постійне збільшення продуктивності комп'ютерів є мотивацією для цього, завдяки загальній пристрасті до комп'ютерних ігор. Розробники ігор стають все більш вимогливими до ресурсів комп'ютерів та продовжують вдосконалювати їх графіку та звук. Захист інформації від несанкціонованого доступу включає різні аспекти, які не обов'язково пов'язані з кодуванням і криптографією. Проте розробники програмного забезпечення повинні бути свідомі можливих загроз і враховувати їх при розробці програм, зокрема систем криптографічного захисту [1].

На сучасний момент існує різноманітні методи захисту від несанкціонованого доступу, але криптографічні методи відзначаються особливою ефективністю. Вони ґрунтуються на властивостях інформації, а не на фізичних або апаратних параметрах, і тому є дуже універсальними.

Спеціально розроблені стандарти шифрування враховують особливості сучасного середовища, де вони використовуються.

Зокрема, важливо застосовувати криптографічний захист у випадках, коли потрібен безперервний обмін інформацією в реальному часі, наприклад, при керуванні безпілотними апаратами. Сучасні стандарти шифрування розроблені з урахуванням цих вимог і враховують швидкодію та безпеку інформаційних систем.

Для досягнення найвищого рівня криптостійкості і ефективності, важливо аналізувати використовувані алгоритми шифрування і вибирати ті, які найкраще відповідають конкретним потребам. Стандарти блокового шифрування, такі як DES і AES, є поширеними і надійними методами шифрування даних [8].

У даному дослідженні розглядається створення програмного модуля шифрування даних для багатокористувацької гри, що допомагає забезпечити безпеку та конфіденційність інформації в цьому контексті.

Об’єкт дослідження – програмний модуль шифрування.

Предмет дослідження – сукупність необхідних умов, що забезпечують найкращий підхід до вдосконалення засобів шифрування у грі.

Завдання дослідження можна сформулювати так:

1. дослідити основні поняття комп’ютерної гри та дати огляд підходів до їх класифікації (по ігровій платформі та графіці, за змістом та видавничими критеріями, за типом їх поширення та кількістю гравців);

2. розглянути програмну (бібліотека Cryptopp) та апаратну реалізацію на ПЛІС. Описати із творення програмного модулю шифрування даних у багатокористувацькій грі. Провести аналіз архітектури програмної реалізації, виконано розробку структури бази даних та use case діаграми модулю шифрування даних.

3. розробити структуру програмного комплексу, визначити його функціональні можливості з наведенням пояснень. Реалізувати алгоритми

шифрування / дешифрування з використанням сплайнів. На базі одного з алгоритмів створити модуль шифрування даних, наведено алгоритм його роботи. Навести детальну, покрокову інструкцію з використання розробленого продукту.

Методи дослідження в роботі використані такі: пошуковий по наявній методичній та науковій літературі із аналізом знайденого матеріалу, порівняння, з’ясування причинно-наслідкових зв’язків, систематизація, аналіз документації та результатів діяльності дослідників з проблеми проведеного дослідження.

Наукова новизна. Розроблено широкий літературний пошук з детальним аналізом наукової інформації. Проведено систематизацію та адаптацію отриманих результатів. Вдосконалення алгоритму шифрування для програмного модулю для багатокористувацьких ігор.

Джерельна база дослідження включала науково-методичну літературу, методичні посібники, наукові статті, періодичні видання та напрацювання сучасних та попередніх вчених і дослідників в галузі кібербезпеки. Ця різноманітність джерел сприяла об'єктивності та комплексності дослідження.

Теоретична та практична цінність роботи полягає в наявності теоретичного матеріалу по дослідженню, відсіяного з-поміж іншого в процесі пошуку інформації по темі, та в систематизації матеріалу напрямку дослідження. На основі проаналізованих даних розроблено програмний модуль шифрування даних у багатокористувацькій грі за допомогою мови програмування C++, якому немає аналогів.

Апробація була здійснена на науково-практичній конференції «Прикладні системи та технології в інформаційному суспільстві» 2023

РОЗДІЛ 1.
ЗАГАЛЬНА ХАРАКТЕРИСТИКА КОМП’ЮТЕРНИХ ІГОР

1.1. Основні поняття комп’ютерної гри

Комп'ютерна гра є програмним додатком, написаним на одній з мов програмування, призначеним для розваги. З розвитком інформаційних технологій з'явилися нові види розваг. Комп'ютерна гра може шукати партнерів для гри або грати самостійно. Ігри часто базуються на фільмах і книгах, і іноді відбувається оборотне впливання. Деякі комп'ютерні ігри стають предметом аматорських і професійних змагань, які відомі як кіберспорт. Вплив комп'ютерних ігор на сучасне суспільство настільки значущий, що інформаційні технології тенденційно використовуються для геймифікації неігрових застосунків.

Texture (текстура) - це растрове зображення, яке накладається на поверхню полігональної моделі для надання їй кольору, забарвлення або створення ілюзії рельєфу.

2D-гра - це гра, в якій об'єкти ігрового світу рухаються лише в одній площині.

Pixel (піксель) - це найменший елемент двовимірного цифрового зображення в растровій графіці або елемент матриці дисплею, що формує зображення. Піксель може бути прямокутної або круглої форми і має певний колір.

Texture Filtering (фільтрація текстур) - це метод поліпшення якості зображення текстур на поверхнях шляхом зменшення спотворень при накладенні текстур на тривимірний об'єкт.

3D-гра - це гра, в якій простір повністю побудований з тривимірних об'єктів. Гравець зазвичай знаходиться в тривимірному просторі і може мати повну свободу пересування. Для подання об'єктів використовуються полігони, зазвичай у формі трикутників [10].

а)

б)

Рис. 1.1. Зовнішній вигляд головного меню 2D гри – а) та приклад 3D гри Polygone (полігон, багатокутник) – мінімальна поверхня для візуалізації – б)

Camera (камера) - це важливий елемент ігрового світу, який відповідає за проекцію ігрового простору на екран гравця. Камера визначає, який фрагмент ігрового світу буде відображений на екрані, його ширину, висоту та кут повороту. Координати камери в просторі визначають, з якого ракурсу гравець бачить гру.

Однак, якщо весь рівень поміщається на екрані без прокрутки, камера може бути фіктивною, оскільки гра не потребує переміщення видимої області [10].

1.2. Огляд класифікацій комп’ютерних ігор

Класифікація комп'ютерних ігор за різними критеріями є важливою для розуміння різноманітності ігрового світу та вибору гри, яка відповідає особистим вподобанням гравця. Основні критерії класифікації ігор включають:

1. Жанр гри: Це визначає мету гри і її основний стиль геймплею. Декілька прикладів жанрів:

– Пригодницька гра (Adventure): З сильним сюжетом і розв'язуванням головоломок.

– Бойовик (Action): Включає в себе бої та перестрілки.

– Рольова гра (RPG): Гри, де гравці відтворюють ролі персонажів з певними характеристиками та навичками.

– Стратегічна гра (Strategy): Гри, де гравці керують процесами та ресурсами, такі як управління армією або бізнесом.

– Комп'ютерний симулятор (Simulator): Імітує реальні життєві ситуації, такі як авіасимулятори або симулятори виживання.

2. Кількість гравців і спосіб їх взаємодії: Гра може бути призначена для одного гравця (однокористувацька) або дозволяти грі кількох гравців одночасно (багатокористувацька). Багатокористувацькі ігри можуть бути PVP (гравці змагаються один з одним) або PVE (гравці протистоять навколишньому світу).

3. Візуальне уявлення: Гра може використовувати графічне оформлення, текстовий формат або комбінацію обох. Вона також може бути двовимірною або тривимірною.

4. Платформа: Гра може призначатися для однієї платформи (наприклад, PC, консоль, мобільний) або бути мультиплатформенною, що означає, що її можна грати на різних пристроях.

Класифікація грецьких ігор за цими критеріями допомагає гравцям обирати ігри, які відповідають їхнім вподобанням та настроєві, і дає змогу більш детально розібратися в різноманітності ігрового світу. Ці критерії дозволяють розуміти, яку саме гру бажає грати гравець і обирати ігри, які відповідають його смакам і попереднім досвідом гри.

1.2.1 Аналіз класифікацій ігор по ігровій платформі та графіці

Як було вказано раніше, відеогра - це форма інтерактивного розважального взаємодії користувача з комп'ютером, яка імітує реальні та уявні ситуації у віртуальному середовищі. Вона має великий освітній потенціал, оскільки сприяє розвитку пізнавального інтересу. На сьогоднішній день існує безліч відеоігор, і їх кількість продовжує зростати щороку. Цей широкий асортимент відеоігор можна класифікувати за різними критеріями. Комп'ютерні ігри є формою розвивально-розважальної взаємодії користувача і комп'ютера. Вони імітують у віртуальному просторі різноманітні життєві та уявні ситуації та мають великий освітній потенціал. Існує безліч комп'ютерних ігор, і щороку їх кількість зростає. Цю широку різноманітність можна поділити на категорії за різними критеріями.

1. Поділ за платформами:

– Ігрові консолі (Sony PlayStation, Microsoft Xbox, Nintendo 3DS, Wii): Це пристрої, призначені для гри, і на них можна запускати відповідні ігри. Кожне покоління консолей має свої особливості.

– Мобільні пристрої (телефони, планшети, КПК): Мобільні пристрої також можуть служити ігровими платформами. Зазвичай на них популярні міні-ігри через їхню обмежену продуктивність та габарити.

– Планшети та сенсорні мобільні телефони: Ці пристрої володіють сенсорними екранами, що дозволяє створювати ігри з унікальним геймплеєм, таким як малювання на екрані або нахил пристрою для зміни гравітації в грі.

– Ігрові автомати (аркадні автомати): Це пристрої, які мають одну попередньо встановлену гру і вимагають оплату за кожний сеанс гри. Зазвичай їх встановлюють у публічних місцях.

– Браузерні ігри (флеш-гри): Ці ігри можна запускати безпосередньо в браузері і не потребують встановлення. Вони можуть бути запущені на різних пристроях, які мають доступ до Інтернету.

2. Браузерні ігри та флеш-гри: Браузерні ігри запускаються в браузері і часто вимагають створення профілю для гри разом з іншими гравцями. Вони можуть мати високу якість, але, зазвичай, поширюються безкоштовно або за схемою "free-to-play" з мікротранзакціями, де можна купувати віртуальні предмети за реальні гроші.

Ця класифікація допомагає гравцям обирати ігри, які найкраще відповідають їхнім потребам та можливостям пристрою [5].

Наразі існує вже вісім поколінь ігрових консолей, проте через цінову політику найпопулярнішими залишаються консолі 6-го та 7-го поколінь. До найпопулярніших поколінь консолей можна віднести Sony PlayStation (PSP, PSOne, PS2, PS3, PS4), Microsoft Xbox (Xbox, Xbox 360, Xbox One), Nintendo 3DS та Wii.

Мобільні пристрої, такі як телефони, планшети, кишенькові комп'ютери (PDA) та кишенькові персональні комп'ютери (КПК), також використовуються як ігрові платформи. Завдяки їх обмеженій продуктивності та компактним розмірам, на КПК переважно поширені міні-ігри.

Планшети та сенсорні мобільні телефони становлять окрему категорію. Ці пристрої дозволяють здійснювати унікальні ігрові враження, такі як малювання на екрані чи нахилення пристрою для зміни гравітації в грі [20].

Аркадні ігрові автомати - це пристрої у формі шафи з екраном, в яких встановлена передвстановлена гра, і для кожного сеансу гри потрібна оплата. Такі автомати зазвичай розташовані в загальнодоступних місцях. Аркадні

автомати були широко популярні в Америці минулого століття і відіграли важливу роль у початковому розвитку відеоігор.

Браузерні або флеш-ігри - це ігри, які можна запускати у веб-браузері (програма для перегляду сторінок в Інтернеті). Спеціальний засіб для браузерних ігор дозволяє грати в них з будь-якого пристрою, який може підключитися до Інтернету. Найпопулярніші браузери, які підтримують запуск невеликих програм безпосередньо на сторінках в Інтернеті, включають Google Chrome, Opera, Firefox, Internet Explorer та Safari.

Флеш-ігри - це тип програм, створених за допомогою технології Flash (будівництво векторної графіки, яку можна переглядати без використання інших програм). Проста векторна графіка дозволяє створювати невеликі, але захоплюючі міні-ігри. Флеш-ігри не потребують завантаження та встановлення, вони запускаються безпосередньо в браузері.

Браузерні ігри - це ігри, які запускаються в браузері та вимагають створення профілю для спільної гри з іншими гравцями. Це тип ігор, в яких відсутній повноцінний відеорежим, і замість нього використовується графіка, намальована на сторінках в Інтернеті з використанням флеш-відеовставок. Браузерні ігри можуть бути високої якості, але, як правило, їх поширюють безкоштовно. Найчастіше такі ігри організовані за схемою "free-to-play" з мікротранзакціями, що дозволяють гравцям купувати віртуальні предмети за реальні гроші.

Розглянемо класифікацію ігор за положенням ігрової камери. Положення точки, з якої ми спостерігаємо за ігровим світом, тісно пов'язане з жанром гри. Деякі ігри дозволяють змінювати положення ігрової камери навіть під час гри. Існує перший особистий вид (вид з очей), коли ми бачимо віртуальний світ очима головного героя. Також є вид з третьої особи (вид ззаду), коли ми спостерігаємо за віртуальним світом з боку, з головним героєм в центрі екрану. Це дає змогу краще оцінювати ситуацію та детально розглядати оточуючий світ.

Двовимірний вид з боку (2D вид з боку) дозволяє бачити всі висотні різниці на одному рівні, такі як ями, прірви та різні платформи. Відсутність третього виміру спрощує сприйняття ігрового світу і полегшує орієнтацію в ньому. Цей вид використовується в жанрах, таких як платформери, головоломки, файтинги та 2D екшени.

Тривимірний вид з боку (3D вид з боку, псевдотривимірність) також використовується в різних жанрах, таких як платформери, квести, головоломки, файтинги та 2D екшени.

Двовимірний вид зверху (2D TopDown) застосовується в жанрах, таких як стратегії, тактика, головоломки та логічні ігри. Цей вид дозволяє спостерігати за розташуванням багатьох ігрових об'єктів, таких як персонажі, війська, техніка та будівлі.

Тривимірний вид зверху (3D TopDown, ізометрія) також використовується в жанрах, таких як стратегії, тактика, головоломки та логічні ігри.

Далі ми розглянемо класифікацію ігор за технологією графіки. Зовнішній вигляд гри є її головним атрибутом, і багато гравців вибирають гру, орієнтуючись саме на тип і якість графічного відображення [6].

Відсутність графіки (текстові ігри, псевдографіка) - це варіант, коли в грі використовуються текстові описи та мінімальна графіка. Це характерно для деяких старих ігор та сучасних текстових пригодницьких ігор, таких як Draft Fortress.

Двовимірна графіка (векторна, растрова) представляє собою графіку, складену з пікселів, які утворюють зображення. 2D-графіка була дуже популярною в 1990-х роках та повернула свою популярність завдяки інді-іграм.

Тривимірна графіка дає можливість створювати ілюзію тривимірного світу на двовимірному екрані за допомогою обчислень. У цьому випадку графіка більш реалістична і деталізована.

Об'ємне зображення застосовується для створення об'ємних об'єктів у графіці. В цьому випадку можна використовувати стереоокуляри для покращення відчуття об'ємності.

Доповнена реальність доступна на мобільних пристроях з відеокамерами, де реальний світ доповнюється віртуальними об'єктами через екран відеокамери [11].

1.2.2. Огляд класифікацій комп’ютерних ігор за змістом та видавничими критеріями

Давайте розглянемо класифікацію комп'ютерних ігор за їхнім жанром. Жанр гри визначається за схожістю ігрової механіки та правил гри. Існує багато різних ігрових жанрів, кожен з яких має свої особливості. Для визначення жанру гри, можна розглянути її компоненти та взаємозв'язки між ними. Проте важливо зауважити, що критерії віднесення гри до конкретного жанру можуть бути неоднозначними, і різні джерела можуть класифікувати одну і ту ж гру по-різному [18].

Класифікація комп'ютерних ігор за жанром включає наступні категорії:

1. Екшен: ця категорія включає 3D-шутери, ігри в жанрі "бродилки-стрілялки" з видом від першої або третьої особи, "криваві" шутери, тактичні шутери та інші.

2. Симулятори: в цю категорію входять аркадні симулятори, спортивні симулятори, симулятори управління спортивними командами, економічні симулятори та інші.

3. Стратегії: ця категорія охоплює стратегічні ігри, які можуть бути в реальному часі або пошаговими, а також в залежності від масштабу гри.

4. Пригоди: до цієї категорії відносяться текстові пригодницькі ігри, графічні квести з головоломками, пригодницькі бойовики, симулятори побачень, візуальні новели та інші.

5. Музичні ігри: ця категорія включає в себе ігри з музичним акцентом, такі як мовленнєві ігри.

6. Рольові ігри: в цю категорію входять тактичні рольові ігри, а також головоломки, логічні ігри та пазли.

7. Традиційні і настільні ігри: до цієї категорії відносяться текстові ігри та ігри в псевдографіці [3].

Зазначена класифікація допомагає розуміти різноманітність ігрових жанрів та їхні основні характеристики. Однак варто зауважити, що це лише загальний огляд ігрових жанрів, і існують багато гібридних жанрів ігор, які поєднують у собі елементи різних категорій [2].

Класифікація комп'ютерних ігор може бути розширена залежно від різних критеріїв, таких як місце дії, час дії і бюджет розробки.

1.2.3. Опис класифікацій ігор за типом їх поширення та кількістю гравців

Дана класифікація ігор відображає різноманітні способи їхнього поширення та геймплею. Розглянемо ці категорії більш детально [27]:

1. Гра на фізичному носії (диски, картриджі). Це класичний спосіб поширення ігор, який використовувався з моменту зародження ігрової індустрії. Фізичні носії інформації включають картриджі, CD, DVD, BluRay. Ігрові картриджі і диски розповсюджуються через спеціалізовані комп’ютерні та ігрові магазини, а також можуть бути замовлені поштою або через Інтернет-магазини.

2. Цифрова копія гри (продаж ігор через Інтернет). Завдяки розвитку інтернет-технологій і збільшенню пропускної здатності інтернету, гри можна завантажувати з Інтернету. Цифрова копія гри завантажується з інтернет-ресурсу продавця, цей процес називається цифровою дистрибюцією.

3. Оплата за ігровий час. У деяких випадках гра не продається окремо, а оплачується лише час гри. Це типовий підхід для аркадних ігрових автоматів.

4. Умовно безкоштовна гра (shareware). Це гри, в які можна грати безкоштовно протягом певного часу або перших рівнів. Для отримання доступу до повноцінної гри гравцю потрібно її купити.

5. Безкоштовна гра з мікротранзакціями (free to play, free2play). Це гри, які можна безкоштовно завантажити та грати без обмежень. Гра отримує прибуток за рахунок продажу додаткових речей, бонусів, покращень, а також ігрової валюти за реальні гроші.

6. Гра без участі гравців (Zero Player Game).Деякі ігри мають можливість налаштувати комп'ютерних супротивників, які грають між собою, без активної участі гравця.

7. Одиночна гра (Синглплейер). Це тип гри, в якому гравець грає сам, без участі реальних гравців. Всі супротивники і союзники контролюються комп'ютером.

8. Спільна гра на одному пристрої (Hotseat, Splitscreen). Це гра, в якій кілька гравців можуть грати на одному пристрої.

9. Багатокористувацька гра (мультиплеєр, Multiplayer). Гра, в якій багато гравців підключаються з різних пристроїв через Інтернет або локальну мережу. Гравці грають одне проти одного або в команді.

10. Масова онлайн гра (MMO). Це ігри, в яких може брати участь велика кількість гравців (десятки і сотні тисяч). Гра зазвичай базується на сервері та вимагає підключення до Інтернету для гри з іншими гравцями.

Дана класифікація допомагає розуміти різноманітність способів доступу до ігор та їхніх особливостей.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ ТА ДВИГУНІВ ДЛЯ БАГАТОКОРИСТУВАЦЬКОЇ ГРИ

2.1. Загальний алгоритм розробки багатокористувацької гри

Розробка комп'ютерних ігор дуже схожа на процес створення будь-якого іншого програмного продукту і включає три основних етапи: проєктування, розробка та видання з подальшою підтримкою.

На етапі проєктування визначається мета гри та інструменти для її створення. Однією з ключових складових є ідея гри, яка визначає, що робить гру цікавою для гравців і чому вони будуть грати в неї. Жанр гри також вибирається на цьому етапі і відображає основні характеристики геймплею. Наприклад, жанр RPG дозволяє гравцям відчути себе у ролі персонажа та розвивати його, тоді як шутери надають можливість брати участь в стрілянинах.

Після визначення ідеї та жанру, наступним кроком є вибір сеттингу, тобто оточення та обставини, в яких відбуватиметься гра [23].

Сеттинг визначає контекст, в якому відбуватиметься дія гри, включаючи місце, час і умови подій. Вибір правильного сеттингу може значно полегшити розробку сценарію гри, тому важливо обирати його заздалегідь, враховуючи смаки цільової аудиторії.

У розробці комп'ютерних ігор ключовими компонентами є програмний код і ігровий движок. Вибір їх залежить від платформи, для якої створюється гра. Наприклад, для браузерних ігор можна використовувати мови програмування Java або Flash, а для персональних комп'ютерів - мову

програмування C#. Ігровий движок відповідає за фізику об'єктів та рендеринг графіки.

На етапі розробки необхідно визначитися із сюжетом гри і ігровою механікою. Ігрова механіка визначає всі об'єкти та правила взаємодії гравця з ними і базується на цілях гри. Паралельно з розробкою ігрової механіки проводиться написання сюжету гри [26].

При розробці ігрових рівнів спершу створюється спрощений план рівня, на якому схематично зображена сама локація рівня і предмети, з якими гравець буде взаємодіяти. Наступним кроком є створення першої версії рівня, яка зазвичай є мінімальною локацією з обмеженою кількістю предметів, необхідних для проходження. Ця версія рівня використовується для тестування його прохідності. Після успішного тестування рівень поступово заповнюється іншими об'єктами.

Незабаром після створення перших рівнів розробляється альфа-версія гри, яка служить для тестування основної механіки гри та перевірки її відповідності вимогам. У цій версії гри об'єкти можуть навіть не мати текстур або бути представлені у вигляді абстрактних об'єктів.

На цьому етапі проводиться доопрацювання рівнів і механіки гри, додаються сюжетні події, такі як відеоролики, сюжетні діалоги і кат-сцени, і виправляються помилки і несправності в коді гри. Після цього настає етап створення другого прототипу гри, бета-версії, яка є практично готовою грою. У бета-версії можуть бути незначні відсутні елементи, які не впливають на геймплей. Бета-версія піддається комплексному тестуванню, включаючи тестування гравцями, що допомагає виявити будь-які несправності і забезпечити якість гри [24].

Якщо гра проходить бета-тестування, то вона відправляється на остаточне доопрацювання і виправлення критичних помилок, після чого йде збірка фінальної версії гри і потім настає реліз гри. Після релізу гри йде подальша її підтримка, яка полягає у випуску патчів (файлів виправлень помилок в

готовому продукті). Так само для того, щоб продовжити життєвий цикл гри, для неї випускають додатковий контент у вигляді DLC (Downloadable content), в який додають різні предмети або можливості для гравця.

Таким чином, можемо констатувати, що етапи розробки комп’ютерних ігор мало чим відрізняються від етапів розробки будь-якого іншого програмного продукту [27].

Якщо гра успішно проходить бета-тестування, то вона піддається остаточному доопрацюванню і виправленню критичних помилок. Після цього проводиться збірка фінальної версії гри, і гра готова для релізу. Після випуску гри настає етап підтримки, який включає в себе випуск патчів (файлів виправлень помилок у готовому продукті). Також, для продовження життєвого циклу гри, розробники можуть випускати додатковий контент у вигляді DLC (Downloadable Content), який може містити нові предмети або можливості для гравця [27].

Отже, можемо визначити, що етапи розробки комп'ютерних ігор мало відрізняються від етапів розробки будь-якого іншого програмного продукту.

2.2. Аналіз ігрових двигунів

На сьогоднішній день існує значна кількість ігрових двигунів, які призначені для спрощення і прискорення процесу розробки ігор. Ігровий движок включає в себе ігрову логіку та набір систем, які контролюють певні аспекти гри, такі як графіка, звук, мережева взаємодія і багато інших. Один із таких ігрових движків - Unreal Engine 4, який був створений компанією Epic Games в 1998 році. Цей інструмент написаний на мові C++ і підтримує операційні системи, такі як Microsoft Windows, OSX і Linux. Unreal Engine 4 має широкий функціонал, включаючи підтримку різних систем рендеринга,

звукових ефектів та платформ, таких як Mac OS, Microsoft Windows, Linux, Android і iOS. Основними перевагами цього движка є доступ до вихідних кодів, система Blueprint для написання ігрової логіки, можливість компіляції C++ коду під час тестування ігри та велика популярність серед програмістів. Проте він також має високі системні вимоги, є складним для освоєння новачків та має певні проблеми з оптимізацією ігрової фізики і тіней.

Ще одним ігровим движком є Construct 2, який був випущений в 2011 році і орієнтований на створення 2D ігор, зокрема для початківців і розробників. Цей движок базується на HTML5 і JavaScript і дозволяє створювати ігри для різних платформ. Construct 2 відрізняється від інших тим, що не вимагає спеціальних навичок програмування, і має зручний редактор для створення ігор. Ліцензія для нього є платною, але також доступна безкоштовна ознайомча версія.

Обираючи ігровий движок для розробки гри, розробники повинні враховувати особливості кожного з них і вибирати той, який найкраще відповідає їхнім потребам та навичкам [26].

Construct 2 виділяється зручним і простим WYSIWYG інтерфейсом, що робить його доступним для користувачів без навичок програмування. Цей движок також дозволяє створювати ігри для різних платформ, включаючи браузери, і має хорошу документацію, що допомагає розробникам освоїти його.

Blitz3D є іншим інтересним інструментом для швидкого створення тривимірних ігор з використанням мови програмування BlitzBasic. Його синтаксис, що базується на мові Basic, робить його легким у вивченні і використанні. Blitz3D дозволяє створювати графічні об'єкти та виконувати над ними різні дії всього декількома командами, що робить його відмінним інструментом для казуальних і міні-ігор.

MonoGame / КСНК - це ігровий движок, який був створений для Microsoft і отримав продовження від проекту MonoGame. Він відзначається низьким

порогом входження і підтримкою основних платформ, включаючи мобільні пристрої.

Unity - це потужне середовище розробки, яке дозволяє створювати додатки і ігри для різних платформ, включаючи комп'ютери і мобільні пристрої. Unity відзначається великим ком'юніті розробників, розширеним ігровим редактором та наявністю багатьох інструментів для розробки. Внутрішній магазин Unity містить багато корисних ресурсів для розробки проєктів.

Unity - Unity3D підтримує мову програмування C# і має багато відомих ігор, що були створені на цій платформі, такі як Temple Run, Hearthstone: Heroes of Warcraft, і Ori and the Blind Forest. Проте, серед недоліків Unity можна відзначити обмежений доступ до вихідного коду на системному рівні і умовно-безкоштовну ліцензію.

Дефолд-крос - це інструмент для розробки 2D ігор з використанням мови програмування Луа. Інструмент дозволяє створювати ігри для різних платформ, включаючи Windows, Mac OS, Linux, iOS, і Android. Він є мультиплатформеним і повністю безкоштовним, проте має обмежений доступ до вихідного коду та невелике ком'юніті.

Unreal Engine 3 / UDK - це популярний ігровий движок з реалістичною графікою і розширеними можливостями фізики та мережевого режиму. Він використовує мову програмування C++ і доступний для Windows і Mac OS. Unreal Engine 3 має великий потенціал для створення вражаючих ігор.

GameStart - це потужна інструмент для створення ігор з вражаючою графікою і підтримкою фізики. Він має безкоштовну версію для Windows, а також версію з відкритим кодом з іншим цінником. Для новачків цей движок може бути дуже корисним завдяки простоті в освоєнні та гарній підтримці фізики.

ShiVa 3D - це ігрова платформа, яка відзначається підтримкою багатьох платформ і операційних систем, включаючи Windows, Linux, MacOS, Nintendo

Wii, iPhone, Google Android, Palm WebOS і браузерну платформу. Вона також має рендеренгові системи у широкому асортименті. ShiVa 3D може бути хорошим вибором для розробників, які планують створювати ігри для різних платформ і операційних систем.

Leadwerks Engine - ця ігрова платформа базується на графічній бібліотеці OpenGL і використовує фізичну бібліотеку Newton SDK. В Leadwerks Engine можна використовувати кілька мов програмування, таких як Java, C#, VB.NET, Python, C/C++, BlitzMax і Lua. Ця платформа має відмінну графіку, але її фізика ще розвивається, і відсутня підтримка мультиплеєра і штучного інтелекту. Комерційна ліцензія коштує $150.

OGRE - це безкоштовна графічна ігрова платформа, яка підтримує операційні системи Windows, Linux і MacOS. Вона має відмінну графіку і базову фізику. OGRE може бути хорошим вибором для розробників, які фокусуються на графіці і не потребують складної фізики.

GLScene - це безкоштовна ігрова платформа під Delphi, і її використання може бути простим для тих, хто знає мову програмування Delphi. Вона підтримує просту фізику і звук, і фізику можна розширити за допомогою спеціальних компонентів.

Irrlicht - це 3D ігрова платформа з відкритим кодом, яка використовує графічні API OpenGL і DirectX. Вона підтримує багато сучасних ефектів, скелетну і морфанімацію персонажів, лайтмепи, піксельні і вертексні шейдери, системи частинок і багато іншого. Ця платформа підтримує різні операційні системи, включаючи Windows, Linux, Mac OS X і інші.

NeoAxis - ігрова платформа, яка використовує базу OGRE і орієнтована на створення ігор класу AAA. Вона має просунуту фізику від Ageia PhysX і підтримує звук за технологією FMOD. Також є штучний інтелект з пошуком шляху і багато вбудованих редакторів. NeoAxis може бути використаний для створення вражаючих ігор та інших 3D презентацій.

Кожна з цих ігрових платформ має свої переваги і недоліки, і вибір залежить від конкретних потреб розробника та характеристик проекту.

2.3. Вибір мови програмування (JavaScript і C #)

Хоча створення ігрових ресурсів відбувається візуальним редактором, необхідний керуючий код, який забезпечує інтерактивність гри. Unity підтримує лише дві мови програмування: JavaScript і C#.

Мова C# має чимало переваг перед JavaScript і має менше недоліків, особливо з точки зору професійних розробників. Однією з цих переваг є чітка типізація мови C#, що недоступна в JavaScript. Серед досвідчених програмістів існують різні точки зору щодо того, чи є динамічна перевірка типів оптимальним підходом, особливо в галузі веб-розробки. Проте при створенні програм для конкретних ігрових платформ, таких як iOS, часто використовується статична типізація, яка є вигідною або навіть необхідною.

У Unity існує директива #pragma, яка вимагає статичної перевірки типів у мові JavaScript. Незважаючи на те, що технічно це допустимо, це порушує один із основних принципів роботи JavaScript. Тому краще вибирати мову програмування з чіткою типізацією відразу. Це лише один із прикладів того, як мова JavaScript в Unity відрізняється від стандартної JavaScript. В багатьох відношеннях вона схожа на JavaScript, який використовується в веб-браузерах, але є ряд відмінностей, залежних від контексту.

Багато розробників називають версію для Unity "UnityScript", що вказує на схожість, але одночасно й відмінність від JavaScript. Цей "аналогічний" стан може становити проблему для програмістів, які намагаються застосувати свої знання мови JavaScript в контексті Unity, а також для тих, хто намагається

використовувати знання, набуті під час роботи в Unity, у зворотному напрямку. Тому в даній роботі ми будемо використовувати мову C#. Програмування відбувається поза Unity, код існує у вигляді окремих файлів, місце розташування яких ви вказуєте Unity. Файли скриптів можуть бути створені в середовищі Unity, але в будь-якому випадку потрібен текстовий редактор або інтегроване середовище розробки (IDE), де ви будете писати код для цих спочатку порожніх файлів.

Як вже було сказано вище, Unity - це багатоплатформний інструмент для розробки ігор. Редактор працює на платформах Windows і MacOS, а створені ігри можуть запускатися на Windows, MacOS, iPhone, iPad, Android, PS3, Xbox 360, а також через веб-програвач Unity (який може бути підключений до браузера на платформах Windows або MacOS). Unity підтримує DirectX і OpenGL.

РОЗДІЛ 3.
ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПРОЦЕСУ ШИФРУВАННЯ ДАНИХ

3.1. Аналіз стандартів та процесорів шифрування, що діють в Україні

Криптографічна безпека є найбільш надійним та ефективним методом забезпечення захисту інформації, і її основною перевагою полягає в можливості зберігання даних у безпечному режимі, недоступному для прямого доступу. Одним з основних критеріїв при виборі криптосистеми є безпека даних, але це не єдиний аспект. Швидкодія також відіграє ключову роль у сфері криптографії. Незважаючи на широкий вибір сучасних методів та алгоритмів шифрування інформації, не всі з них можуть забезпечити необхідний рівень ефективності (оптимальності) в конкретних умовах [12].

Алгоритм шифрування представляє собою набір обернених перетворень для захисту змісту повідомлення від несанкціонованого доступу. Вихідне повідомлення, яке піддається перетворенню, відоме як відкритий текст, а результат, отриманий за допомогою алгоритму шифрування, називається шифротекстом.

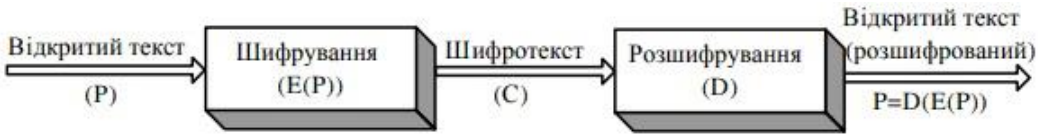


Рис. 3.1. Схематичне зображення процесів шифрування і розшифрування

Додатково, швидкий розвиток обчислювальних засобів та одночасне зниження їх витрат породжують нові вимоги щодо безпеки та продуктивності системи. Це призводить до витіснення старих криптографічних алгоритмів новими, які проходять конкурентні відбори для підтвердження їх здатності

забезпечувати безпеку протягом майбутніх періодів. Перехід від відкритого тексту до шифротексту відомий як шифрування, а зворотній процес - розшифрування. Ключовою вимогою для виконання як шифрування, так і розшифрування є наявність секретного ключа [13].

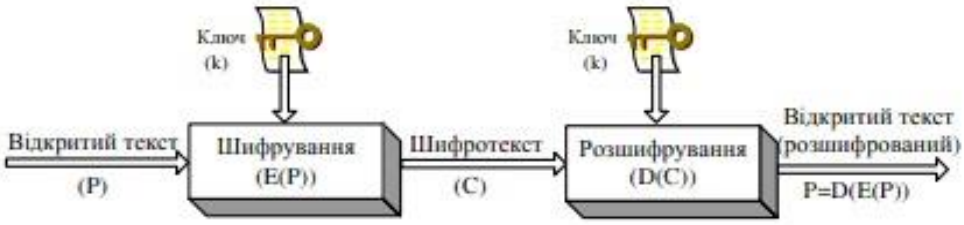


Рис. 3.2. Шифрування і розшифрування за допомогою ключа (симетрична)

Захистом інформації [19] називають сукупність засобів та методів, що дозволяють забезпечити інформації такі основні властивості:

– конфіденційність – дані не повинні бути доступні для сторонніх користувачів;

– цілісність – лише перевірений користувач має право модифікації даних;

– доступність – інформації може бути використана лише авторизованими користувачами, згідно певних вимог, за якими ця інформацію поширюється.

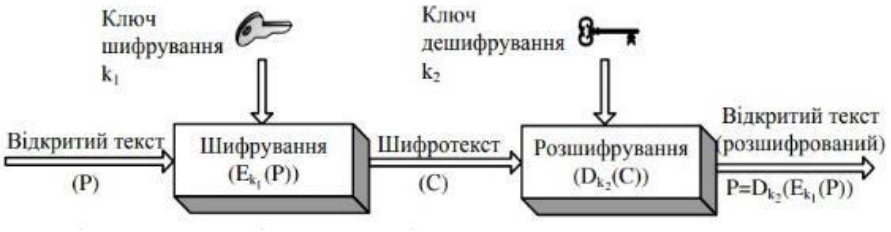


Рис. 3.3. Шифрування і розшифрування за допомогою двох різних ключів (асиметрична)

Відповідно до вищезазначених характеристик, в інформаційній безпеці можуть виникати наступні загрози: порушення цілісності (модифікація,

знищення); розголошення конфіденційності (розголошення, витік); порушення доступності (знищення або блокування).

До 2014 року в Україні існував алгоритм, який був національним стандартом шифрування. Однак він не відповідав сучасним вимогам щодо швидкості, і тому з 2015 року його було замінено новим стандартом. На сьогоднішній день в Україні використовуються такі алгоритми симетричного блокового шифрування: ДСТУ ГОСТ 28147:2009, AES (у складі операційних систем загального призначення), RC4 (у закордонних реалізаціях інструментів для захисту веб-з'єднань відповідно до протоколів SSL / TLS), Triple DES (використовується Національним банком України та у закордонних реалізаціях для захисту мережевого трафіку IPsec), ДСТУ 7624:2014 (алгоритм «Калина»). [20].

Перевагами стандарту ДСТУ 28147:2009 є його загальна відомість, оскільки він був добре досліджений міжнародною спільнотою протягом більше ніж 20 років. Він також має прийнятний рівень швидкодії для 32-бітових платформ і є зручним для апаратної реалізації, зокрема для малоресурсної (lightweight) криптографії. Вузли заміни (S-блоки) мають хороші властивості, що забезпечують практичну стійкість шифру.

Недоліками ДСТУ ГОСТ 28147:2009 є наявність теоретичних атак зі складністю, яка значно менша, ніж повний перебір ключів, великі класи слабких ключів та використання вузлів заміни спеціального типу, що може зменшити рівень стійкості проти практичних атак (особливо на основі шифр-текстів) з використанням одного персонального комп'ютера. Швидкодія на сучасних системах суттєво нижча порівняно з іншими блоковими шифрами [21].

Зазначені характеристики роблять ДСТУ ГОСТ 28147:2009 придатним для застосування на пристроях з обмеженими ресурсами, таких як вбудовані системи і мікроконтролери, де ефективність та надійність шифрування є важливими факторами.

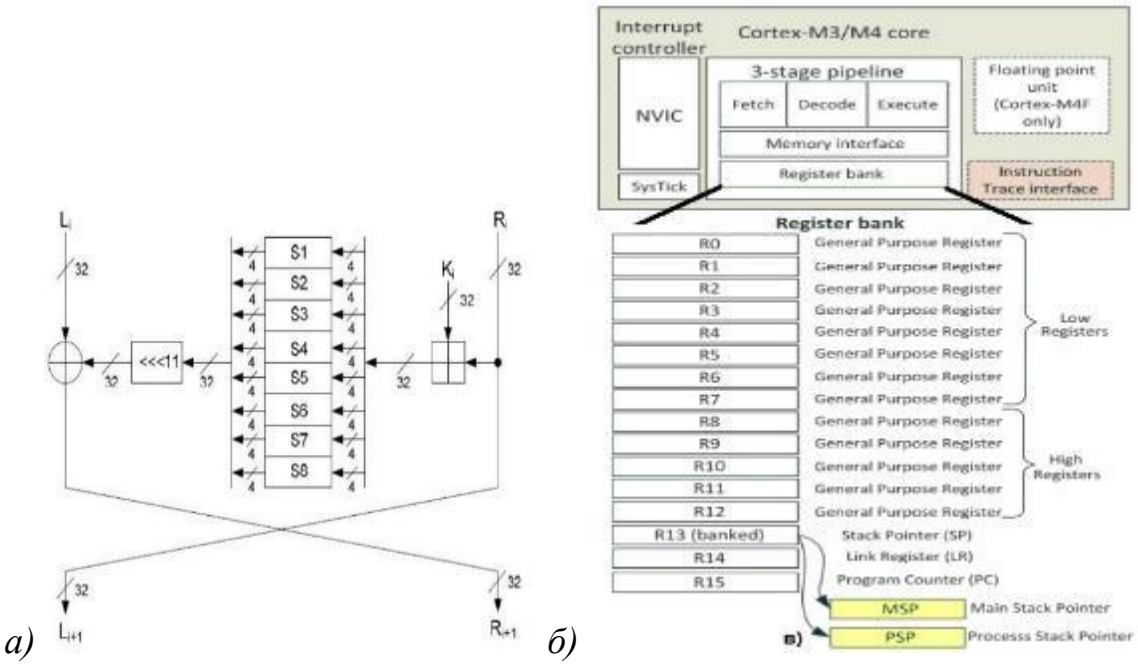


Рис. 3.4. Структурна схема раунду алгоритму ГОСТ 28147 – а) та архітектура центрального процесора ARM Cortex-M³ – б)

AES (Advanced Encryption Standard, також відомий під назвою Rijndael) – це симетричний алгоритм блочного шифрування, який був прийнятий урядом США як національний стандарт шифрування, оскільки став фіналістом конкурсу. Advanced Encryption Standard (AES) [28], спеціально розроблений для ефективної програмної реалізації як на 8-, так і на 32-бітних процесорах. AES демонструє найкраще співвідношення продуктивність / пам’ять порівняно з іншими відомими криптоалгоритмами. Вітчизняний стандарт шифрування ДСТУ ГОСТ 28147:2009 [2], який має важливе значення для національної інформаційної безпеки, також орієнтований на 32-бітні платформи і має невисокі вимоги до складності реалізації у вбудованих системах. З огляду на це, становить інтерес порівняння параметрів вказаних криптоалгоритмів під час їх реалізації на ARM-процесорах.

Низька продуктивність 8-ми та 16-бітних процесорів загального призначення при виконанні криптографічних алгоритмів спонукала виробників до включення в мікроконтролери спеціальних криптомодулів або апаратних

прискорювачів. Наприклад, у 8-бітних мікроконтролерах сімейства AVR XMEGA від фірми Atmel, існує криптомодуль AES-128, який здатний виконувати шифрування/дешифрування одного блока даних за 375 тактів при максимальній тактовій частоті 32 МГц.

В сучасний час розмір блоку даних для шифрування становить 128 біт, а розмір ключа може бути 128, 192 або 256 біт. Підтримка AES була введена фірмою Intel у процесорах x86, включаючи Intel Core i7–980X Extreme Edition і Sandy Bridge.

Щоб досягти подальшого підвищення продуктивності, можна використовувати мікросхеми, в яких на одному кристалі поєднані ядро мікроконтролера і програмована логічна інтегральна схема (ПЛІС). Прикладом таких мікросхем є клас Field Programmable System Level Integrated Circuit (FPSLIC) від фірми Atmel, які об'єднують стандартне 8-бітне ядро мікроконтролера AVR і ПЛІС.

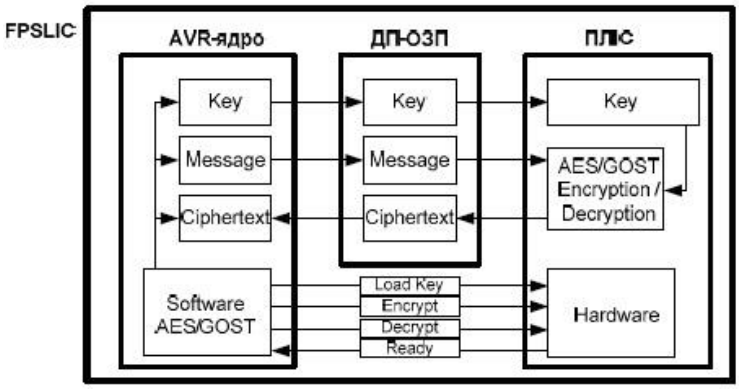


Рис. 3.5. Реалізація апаратного криптопроцесора на базі FPSLIC

Декілька моделей ARM-мікроконтролерів також оснащені апаратними криптомодулями для AES-шифрування. Наприклад, мікроконтролери AT91SAM7X / XC з ARM7TDMI-ядром від фірми Atmel підтримують апаратне шифрування / дешифрування алгоритмом AES-128 за 12 тактів. На апаратному рівні також виконуються заходи для запобігання диференційним атакам через аналіз енергоспоживання (DPA-атаки).

Мікроконтролери сімейства EFM32G від фірми Energy Micro з ядром ARM Cortex-M³ мають внутрішній апаратний акселератор для алгоритму AES з підтримкою ключів довжиною 128 і 256 біт. Для 128-бітного ключа один блок даних шифрується за 54 такти, а для 256-бітного ключа - за 75 тактів.

У мікроконтролерах сімейства STM32F205 / 207 / 215 / 217xx з ядром ARM Cortex-M³ присутній криптографічний процесор, який підтримує AES-шифрування з ключами 128, 192 і 256 біт. Для обробки одного блока потрібно від 14 до 18 тактів, в залежності від довжини ключа, при максимальній тактовій частоті 120 МГц.

Не дивлячись на те, що досліджуваний алгоритм AES має широке використання у вбудованих системах, його вимоги до необхідного обсягу пам’яті є доволі значимими. Зменшення розміру коду приводить до непропорційного зменшення продуктивності, отже, доволі важко здійснювати обмін «продуктивність / розмір коду» залежно від конкретного застосування.

Таблиця 3.1

Результати програмної реалізації алгоритму AES для ядер ARM7TDMI-S та ARM Cortex-M³

Платформа	Шифрування, тактів / блок	Дешифрування, тактів / блок	Шифрування, тактів / байт	Дешифрування, тактів / байт	ROM, байт
AES–128					
ARM7TDMI (unrolling)	1675	2074	105	130	Дані відсутні
ARM7TDMI (on-the-fly)	2074	2378	130	149	Дані відсутні
ARM7TDMI	639	638	40	40	5966
ARM7TDMI	1994	1994	125	125	7944
ARM7TDMI	5542	5542	346	346	2292
ARM Cortex-M³	1329	1347	83	84	5272
AES–192					
ARM7TDMI	747	746	47	47	5966
AES–256					
ARM7TDMI	855	854	53	53	5966

ARM7TDMI-S - це універсальний 32-бітний процесор з Прінстонською архітектурою, який відзначається низьким споживанням енергії і може

працювати на частоті до 133 МГц. Він має трирівневий конвеєр, що дозволяє виконувати прості команди за один такт. У будь-якому з 7 режимів роботи процесору доступні 16 регістрів загального призначення R0-R15. Система команд складається з достатньо повного, для RISC-процесора, набору ортогональних інструкцій з підтримкою префіксів умовного виконання та потужних індексних режимів адресації. Він також оснащений арифметико-логічним пристроєм з блоком зсуву, який дозволяє виконувати зсуви одного з операндів на довільну кількість розрядів під час виконання операції [33].

Cortex-M3 - це ядро, спеціально розроблене для використання в мікроконтролерах. Відмінною особливістю Cortex-M3 порівняно з ARM7TDMI є наявність системних елементів, таких як контролер вкладених переривань, системний таймер і налагоджувальні модулі. Ядро має Гарвардську архітектуру з окремими шинами для даних і команд, що спільно з трирівневим конвеєром покращує продуктивність. Регістровий файл містить 16 регістрів R0-R15, з R0-R12 призначені для користувача, а R13-R15 виконують спеціальні функції. Cortex-M3 також має пристрій для циклічного зсуву, який дозволяє виконувати зсуви на величину до 32 біт одночасно з виконанням інших операцій.

У ядрі Cortex-M3 використовується набір команд THUMB-2, спрямований на компілятори мови C / C++. Набір THUMB-2 включає 16- і 32-бітні команди, що дозволяє поєднувати продуктивність режиму ARM і економічність режиму THUMB. Цей набір команд також розширює можливості і включає нові інструкції, такі як ділення, маніпуляції з бітами і умовні команди. Cortex-M3 може працювати з максимальною тактовою частотою до 264 МГц, а сучасні мікроконтролери на базі Cortex-M3 працюють на частотах до 150 МГц.

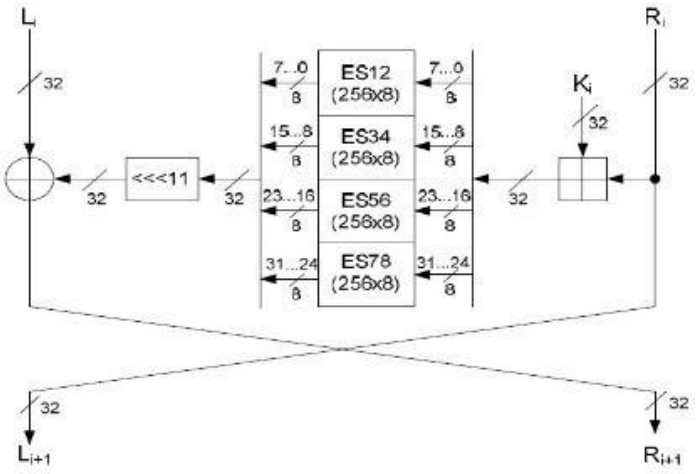


Рис. 3.6. Структурна схема алгоритму ГОСТ 28147 для варіанта максимальної продуктивності

Переваги AES:

1. Досліджений і відкритий: AES є найбільш дослідженим у світі криптографічним алгоритмом і вивчений в численних відкритих публікаціях.

2. Висока стійкість: AES забезпечує високий рівень практичної стійкості і включений до набору Suite-B Агентства національної безпеки США. Це робить його придатним для захисту інформації з обмеженим доступом уряду США.

3. Ефективність на 32-бітних платформах: AES ефективний для реалізації на 32-бітних платформах, включаючи ARM-мікроконтролери.

4. Наявність апаратних акселераторів: Деякі ARM-мікроконтролери оснащені апаратними акселераторами AES, що поліпшують продуктивність обробки даних.

Недоліки AES:

1. Теоретичні атаки: Існують теоретичні атаки зі складністю, меншою, ніж повний перебір. Це може створювати певні обмеження стійкості алгоритму.

2. Обмеженість на 64-бітових платформах: AES не може в повній мірі використовувати можливості 64-бітових платформ, оскільки розроблений для роботи з 128-бітовими блоками даних.

3. Застарілість: AES був розроблений у 1997 році, і в порівнянні з новішими алгоритмами може вважатися застарілим.

4. Довіра до іноземних апаратних реалізацій: Виникають питання щодо довіри до іноземних апаратних реалізацій AES, особливо в контексті розкриття інформації Едвардом Сноуденом. Деякі компанії почали відмовлятися від використання AES на користь інших алгоритмів, таких як ChaCha20.

Отже, AES є добре дослідженим і стійким алгоритмом шифрування, але має свої обмеження і може бути не найкращим вибором у всіх ситуаціях, особливо на платформах з обмеженими ресурсами або з питаннями щодо довіри до апаратних реалізацій.

RC4 (Rivest Cipher 4**)** є потоковим шифром і використовується широко в комп'ютерних мережах і протоколах забезпечення безпеки. Він був розроблений компанією RSA Security.

Переваги RC4:

1. Висока швидкість роботи: RC4 відомий своєю високою швидкістю обробки даних.

2. Змінний розмір ключа: RC4 дозволяє використовувати ключі різних розмірів, що робить його гнучким.

Недоліки RC4:

1. Методи атак: RC4 було піддано атакам, і було виявлено методи успішної атаки на нього. Це підштовхнуло до припинення підтримки RC4 в багатьох криптосистемах і протоколах [34].

Triple DES (3DES) є симетричним блоковим шифром, створеним на основі алгоритму DES (Data Encryption Standard) з метою підвищення його криптостійкості.

Переваги Triple DES:

1. Довгий ключ: 3DES використовує ключі значної довжини, що робить його більш стійким до криптоаналізу.

2. Досліджений і довговічний: 3DES є добре дослідженим і популярним алгоритмом, який був у використанні протягом більше ніж 30 років.

3. Підтримка у банківських системах: 3DES залишається популярним в банківських системах та імпортованих стандартах.

Недоліки Triple DES:

1. Відносно низька швидкодія: 3DES має меншу швидкодію порівняно з більш сучасними алгоритмами шифрування.

2. Класи слабких ключів: Існують класи слабких ключів, які можуть знизити стійкість 3DES.

3. Виходить з ужитку: Замість 3DES сучасні системи поступово переходять на більш швидкий і сучасний AES Rijndael [34].

Узагальнюючи, обидва алгоритми мають свої переваги і недоліки, і вибір між ними залежить від конкретних вимог до безпеки та швидкодії. Проте RC4 не рекомендується для використання через відомі атаки, тоді як 3DES залишається досить достойною альтернативою, особливо для старших систем та програм, які вимагають більшої криптостійкості [45].

Одним із основних алгоритмів симетричного блокового шифрування, що використовуються в Україні, є ДСТУ 7624:2014 («Калина»), який визначає сучасний алгоритм симетричного блокового перетворення для забезпечення конфіденційності й цілісності інформації при її обробці та встановлює режими його роботи.

Таблиця 3.2

Режими роботи алгоритму Калина

Режим	Мета застосування	Збільшення швидкості	Коментар
ECB	конфіденційність	-	Основа для побудови інших режимів. Окреме застосування не передбачене
CTR	конфіденційність	+	Основний режим для шифрування
CFB	конфіденційність	+	
CMAC	цілісність	+	Основний режим для КАП
CBC	конфіденційність	-	Існує для сумісності із іншими міжнародними стандартами

			(використання не рекомендоване)
OFB	конфіденційність	+	
GCM, GMAC	конфіденційність та цілісність	+	Основний режим захисту даних, що передається в мережах
CCM	конфіденційність та цілісність	+	Основний режим, що забезпечує конфіденційність та цілісність
XTS	конфіденційність	-	Призначення режиму – прозоре шифрування носіїв даних
KW	конфіденційність та цілісність	-	Використовується для обробки невеликих об’ємів даних (значно повільніший за інші режими)

Алгоритм шифрування "Калина" є результатом співпраці українських вчених і Держспецзв'язку, яка виникла після проведення національного змагання у галузі криптографії. Основою для "Калина" послужив шифр AES (Advanced Encryption Standard).

Основні характеристики алгоритму "Калина":

1. Структура SPN: Алгоритм "Калина" побудований на базі структури SPN (substitution-permutation network), що є популярною структурою для блокового шифрування.

2. Збільшена MDS-матриця: "Калина" використовує збільшену MDS-матрицю (Multiply–Diffuse–Multiply matrix), що сприяє підвищенню криптографічної стійкості.

3. Підстановку блоків: Алгоритм включає чотири підстановкувальних блоки, які використовуються для обробки даних.

4. Операція додавання по модулю 264: Для попереднього і фінального забілювання використовується операція додавання по модулю 2^64, що додає до безпеки алгоритму.

5. Вищий рівень криптографічної стійкості: Порівняно з міжнародним стандартом AES, алгоритм ДСТУ 7624:2014 забезпечує вищий рівень криптографічної стійкості і може використовуватися з ключами та блоками шифрування розміром до 512 бітів.

6. Режими роботи: Алгоритм ДСТУ 7624:2014 визначає десять різних режимів роботи (застосування), які відповідають міжнародному стандарту ISO/IEC 10116:2006.

Загалом, "Калина" є сучасним і стійким алгоритмом шифрування, який має великий потенціал для застосування в різних програмних і програмно-апаратних платформах і забезпечує високий рівень безпеки для обробки даних.

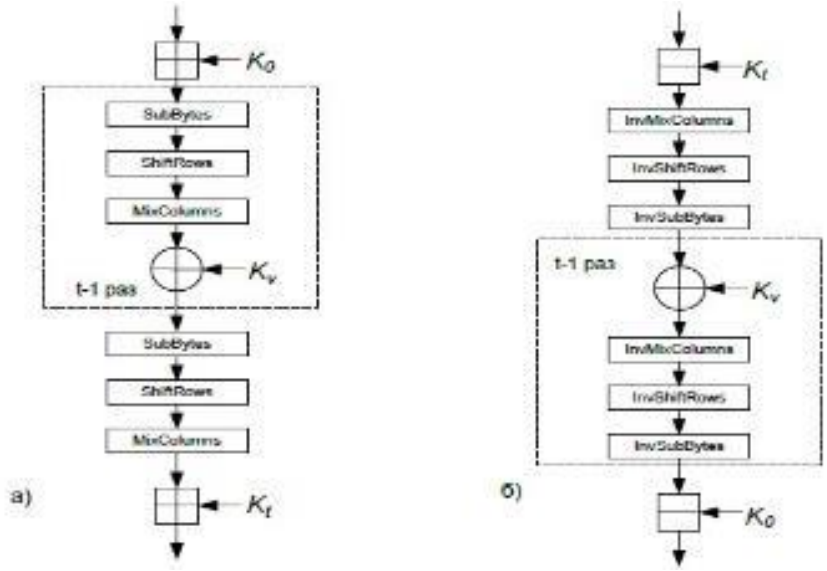


Рис. 3.7. Структурна схема шифру «Калина» в режимах зашифрування (а) і розшифрування (б)

Порівняльна таблиця з перевагами та недоліками апаратних та програмних рішень (Таблиця 3.1).

Таблиця 3.1

Переваги та недоліки апаратної та програмної реалізації шифрування

Апаратна реалізація		Програмна реалізація	
Переваги	Недоліки	Переваги	Недоліки
Високі швидкісні характеристик	Необхідність шифрування усіх каналів зв’язку	Гнучкість	Слабка фізична захищеність
Велика стійкість до сторонніх електромагнітних випромінювань		Переносимість	Можливість втручання в дію алгоритмів

Стійкість від зовнішнього фізичного впливу		Простота реалізації	Не всі задачі шифрування можна реалізувати програмно
Використання у місцях, які не доступні програмним		Більша к-сть алгоритмів доступна для реалізації	

Алгоритм "Калина" має кілька важливих переваг:

1. Надзвичайно висока стійкість: Один із головних аспектів алгоритму "Калина" - це його надзвичайно висока стійкість до криптоаналізу та зламу. Він забезпечує надійний захист даних від несанкціонованого доступу.

2. Висока швидкість реалізації: Алгоритм "Калина" реалізований з урахуванням вимог до високої швидкодії як на програмних, так і на апаратних платформах. Це дозволяє ефективно використовувати його в різних застосунках, включаючи обробку великих обсягів даних.

3. Порівнянна або вища ефективність: У порівнянні з іншими світовими рішеннями в галузі криптозахисту, "Калина" може демонструвати подібну або навіть вищу ефективність. Це робить його конкурентоспроможним у світовому контексті.

4. Різні режими роботи: Алгоритм "Калина" підтримує різні режими роботи, що дозволяє використовувати його для різних сценаріїв криптозахисту та обробки даних. Ця гнучкість важлива для сучасного криптозахисту, де можуть існувати різні вимоги до захисту даних.

5. Зручна реалізація: Алгоритм "Калина" має зручну структуру, що полегшує його реалізацію на різних платформах та в різних програмах.

З урахуванням цих переваг, алгоритм "Калина" може бути ефективним і надійним рішенням для забезпечення криптозахисту даних в різних галузях, включаючи інформаційну безпеку, електронний банкінг, комунікації і багато інших.

Нижче (Таблиця 3.44) наведені усі можливі комбінації роботи алгоритму.

Таблиця 3.4

Стани шифру в алгоритмі шифрування «Калина»

Розмір слова, біт	Розмір блоку, біт	Розмір ключа, біт	Ідентифікатор	Кількість раундів
64	128	1 x 128 = 128	Калина 128 / 128	10
		2 x 128 = 256	Калина 128 / 256	14
	256	1 x 256 = 256	Калина 256 / 256	
		2 x 256 = 512	Калина 256 / 512	18
	512	1 x 512 = 512	Калина 512 / 512	

У даному фрагменті тексту розглядається представлення 128-бітного блоку даних для алгоритму "Калина". Це представлення базується на використанні bitsliced-змінних, які дозволяють ефективно опрацьовувати біти вхідного блоку та забезпечують оптимальне використання ресурсів.

Основні аспекти цього представлення:

1. Bitsliced-змінні: Вхідний 128-бітний блок розбивається на 8 bitsliced-змінних під назвами x0-x7. Кожна з цих змінних містить біти з відповідним номером. Наприклад, в змінній x0 зберігаються нульові біти байтів стану, а в змінній x1 - біти з номером 1 байтів стану і так далі. Це дозволяє ефективно розподіляти та опрацьовувати біти блоку.

2. Розмір блоку: Зазначається, що хоча можна було б розбити кожен біт вхідного блоку на окрему змінну, цей підхід вимагав би надто багато змінних і призводив б до зростання розміру повідомлення, що не завжди доцільно на практиці.

3. Масштабованість: Автори описують, як можна використовувати регістри різних розмірів, такі як xmm, ymm і zmm, для зберігання бітів для відповідно 8, 16 і 32 блоків "Калини". Це робить представлення масштабованим і дозволяє ефективно використовувати ресурси обчислювальних платформ.

4. Позиціонування бітів: Описується, як біти вхідного блоку розташовуються в bitsliced-змінних та як вони групуються побітово, зокрема, як

молодший значущий біт кожного байту попадає в x0, а старший значущий біт - в x7.

Це представлення дозволяє оптимізувати роботу алгоритму "Калина" та ефективно використовувати обчислювальні ресурси для обробки даних у блоках.

Рис. 3.8. Перехід від нормального до bitsliced представлення даних

У даному фрагменті тексту розглядається процес упакування і розпакування даних у алгоритмі "Калина". Цей процес включає в себе відбілювання на базі арифметичних операцій, використання логічних інструкцій та таблиць заміни Sbox0-Sbox3 [42].

Основні аспекти цього процесу:

1. Відбілювання: Процес упакування та розпакування даних включає в себе відбілювання на базі арифметичних операцій додавання і віднімання. Ці операції виконуються на початку і в кінці перетворення. Однак, через те що ці операції в bitsliced-представленні витратні, перехід до bitsliced-формату здійснюється після виконання початкового відбілювання і перед кінцевим відбілюванням.

2. Представлення даних: Вхідні та вихідні дані алгоритму "Калина" представлені у вигляді рядків фіксованої довжини. Вони складаються з 8 × Nb байт (64 × Nb біт). Ключ також представляється у вигляді байтового рядка розміром 8 × Nk байт (64 × Nk біт).

3. Таблиці заміни Sbox0-Sbox3: У шифрі "Калина" використовуються чотири таблиці заміни Sbox0-Sbox3, через які проходять байти стану. Операція SubBytes використовує ці таблиці для обчислення вихідних значень для кожної з таблиць.

4. Упаковка та розпакування: Процедура PACK використовується для упаковки значень s0-s31 у регістри x0-x7 після виконання операції SubBytes [43].

Цей процес дозволяє оптимізувати обробку даних в алгоритмі "Калина" та забезпечує ефективну реалізацію операцій упакування та розпакування.

3.2. Особливості використання алгоритму «Калина»

В даному фрагменті тексту розглядається стандарт блокового симетричного шифрування "Калина" (ДСТУ 7624:2014) та його переваги порівняно з іншими міжнародними аналогами. Основні відомості з цього фрагменту:

1. Мета розробки: Стандарт "Калина" розроблений з урахуванням існуючих та потенційних загроз в галузі криптографії. Він має відповідати сучасним вимогам до рівня криптостійкості та швидкодії та призначений для активного використання протягом кількох наступних десятиліть.

2. Режими роботи: Стандарт ДСТУ 7624:2014 визначає десять різних режимів роботи, які широко поширені відповідно до міжнародних стандартів. Це дозволяє широко використовувати "Калину" для захисту інформації в різних

контекстах, таких як передача даних по мережах, шифрування жорстких дисків і електронних документів.

3. Вимоги до "Калини": Стандарт блокового шифру "Калина" повинен відповідати ряду вимог, таких як висока криптографічна стійкість, висока швидкодія на сучасних платформах, компактність програмної і програмно-апаратної реалізації, можливість інтеграції різних алгоритмів, прозорість проєктування та ефективність порівняно з іншими світовими рішеннями.

4. Параметри "Калини": "Калина" підтримує розмір блоку і довжину ключа від 128 до 512 біт, що дозволяє використовувати різні комбінації для різних завдань криптографічного захисту інформації.

5. Результати роботи: Стандарт "Калина" є результатом співпраці Державної служби спеціального зв'язку та захисту інформації України і вчених. Введення цих стандартів дозволить покращити ефективність систем захисту інформації, які розробляються в Україні, і підняти їх на вищий рівень захисту порівняно зі світовими практиками.

Процес шифрування за допомогою алгоритму «Калина» може бути описаний за допомогою наступної формули:

$$T_{l,k}^{(K)} = \eta_l^{(K_t)} \times \psi_l \times \tau_l \times \acute{\pi}_l \; {}^{*}\prod_{v=1}^{t-1} \left(\eta_l^{(K_v)} \times \psi_l \times \tau_l \times \acute{\pi}_l \right) \times \eta_l^{(K_0)}$$

де $T_{l,k}^{(K)}$ *– ключ шифрування,*

$\eta_l^{(K_v)}$*– функція додавання раундового ключа до матриці станів за модулем* 2^{64}*,*

τ_l*–перестановка елементів матриці станів (циклічний зсув байт вправо),*

$\acute{\pi}_l$*– заміна байтів у матриці станів,*

$k_l^{(K_v)}$*– додавання за mod 2 матриці стану і раундового ключа,*

ψ_l *– лінійне перетворення елементів матриці стану над скінченим полем.*

У алгоритмі блокового шифру "Калина" велика увага приділяється вибору МДВ-матриці, яка впливає на криптографічні характеристики та продуктивність алгоритму. Розмір цієї матриці обрано таким чином, щоб

забезпечити необхідну криптографічну стійкість і спростити реалізацію на сучасних пристроях. Також важливою є кількість циклів в алгоритмі, яка повинна бути достатньою для забезпечення стійкості до різних видів атак.

Для впровадження блоку лінійного розсіювання використовується множення на МДВ-матрицю, яке є ефективним методом. Існують два можливих підходи до реалізації МДВ-перетворення, пов'язані з розміром блоку шифру та вхідних значень.

Збільшення розміру МДВ-матриці призводить до покращення криптографічних властивостей алгоритму, що може підвищити стійкість або ефективність шифрування, в залежності від кількості операцій.

Таблиця 3.5

Залежність кількості активних S-блоків від кількості операцій

Характеристика	Номер S-блоку			
	1	2	3	4
Макс. значення таб. лін. апроксим. (ЛК)	24			
Мінімальна алгебраїчна степінь булевої функції	7			
Макс. значення таб. розпод. різниць (ДК)	8			
Мінімальне значення нелінійності булевої функції	104			
Степінь перевизначеної системи	3 (441 рівняння)			
Кількість циклів	4	4	6	4
Мінімальна довжина циклу	6	8	4	4

Оптимальна реалізація алгоритму "Калина" передбачає використання таблиць передобчислень, які поєднують нелінійні і лінійні перетворення, спрощуючи при цьому реалізацію прямого криптографічного перетворення (зашифрування). Для схеми розгортання ключів встановлено вимоги, що забезпечують необхідну криптографічну і експлуатаційну стійкість. Зокрема, ключові вимоги включають:

1. Нелінійну залежність між цикловими ключами і ключем шифрування.

2. Суттєві відмінності між цикловими ключами з складними нелінійними зв'язками.

3. Захист від відомих криптоаналітичних атак на схему розгортання ключів.

4. Відсутність слабких ключів, які погіршують криптографічні властивості або знижують стійкість перетворення.

5. Обчислювальна складність формування всіх циклових ключів не повинна бути вище, ніж складність зашифрування трьох блоків.

6. Простота реалізації як програмної, так і апаратної версій [30].

Важливою є також можливість зберігати всі таблиці, які використовуються під час шифрування, в кеші L1 сучасних процесорів. Збільшення розміру МДВ-матриці може підвищити криптографічні характеристики, але існує технологічна межа збільшення розміру для забезпечення швидкодії на загальноспоживчих програмних платформах. Новий шифр "Коник" є прикладом такого підходу і має більший розмір МДВ-матриці, але меншу швидкодію на загальноспоживчих платформах через обсяг таблиць, що виходить за межі кешу L1 сучасних процесорів [15].

Результати аналізу стійкості шифру «Калина» із розміром блоку 128 бітів представлено в таб. 3.6.

Таблиця 3.6

Результати аналізу стійкості шифру «Калина» із розміром блоку 128 бітів

Метод криптоаналізу	Найменша кількість циклів, для якої шифр є стійким	Показники атак		
		Макс. кількість циклів	Обчисл. складність, екв. оп. шифрув.	Пам’ять, байтів
Диференційний	5	4	255	
Лінійний	5	3	252,8	
Усіч. диференц.	4	3		
Інтегральний	6	5	297	233+4
Нездійсн. дифер.	6	5	262	266
Бумеранг	5	4	2120	

"Калина" виявляється швидшою за AES на показник від 12,5% до 27%, залежно від розміру блока і довжини ключа. У тих самих умовах "Калина" перевершує ГОСТ 28147-89 від 3,17 до 3,72 разів у швидкості обробки. Блоковий шифр "Калина", який описаний в ДСТУ 7624:2014, забезпечує високий рівень стійкості з різними довжинами блока і ключа, включаючи 128, 256 і 512 бітів.

Таблиця 3.7

Результати аналізу стійкості шифру «Калина» із розміром блоку 256 бітів

Метод криптоаналізу	Найменша кількість циклів, для якої шифр є стійким	Показники атак		
		Макс. кількість циклів	Обчисл. складність, екв. оп. шифрув.	Пам’ять, байтів
Диференційний	7	6	2230	
Лінійний	7	5	2220,8	
Усіч. диференц.	4	3		
Інтегральний	7	6	2145	264+5
Нездійсн. дифер.	6	5	261	266
Бумеранг	6	5	2220	

Новий національний стандарт дозволяє досягти швидкості шифрування на рівні 2611,77 Мбіт/с з 128-бітовим ключем, 2017,97 Мбіт/с з 256-бітовим ключем і 1386,46 Мбіт/с з 512-бітовим ключем при програмній реалізації мовою C++ (gcc v4.9.2) на 64-бітовій платформі Intel Core i5–4670@3.40GHz. Ці показники перевищують швидкість AES на 1–3%, досягаючи до 86 Мбіт/с.

Таблиця 3.8

Результати аналізу стійкості шифру «Калина» із розміром блоку 512 бітів

Метод криптоаналізу	Найменша кількість циклів, для якої шифр є стійким	Показники атак		
		Макс. кілкість циклів	Обчисл. складність, екв. оп. шифрув.	Пам’ять, байтів
Диференційний	9	8	2490	
Лінійний	9	7	2470,4	
Усіч. диференц.	4	3		
Інтегральний	7	6	2137	264+5
Нездійсн. дифер.	6	5	260	266
Бумеранг	7	6	2340	

Таблиця 3.9

Швидкодія використаних алгоритмів і перетворень

Перетворення	Калина – 128	AES–128
XORRoundKey / AddRound Key	0,047 с	0,047 с
Add 32RoundKey	0,125 с	-
Kalina_S_boxes / SubBytes	0,532 с	4,094 с / 0,593 с
ShiftRows / S h i ft Rows	0,063 с	0,078 с
MixColumns / MixC olumns	16,109 с	5,594 с
Kalina_KeyExpansion / Key Expansion	102,688 с	12,265 с

Збільшення кількості операцій множення в полі в алгоритмі «Калина» є основною причиною відставання в швидкості в порівнянні з алгоритмом AES. Результати обчислень наведені в таб. 3.10.

Таблиця 3.10

Розрахункові затрати шифрів

Кількість раундів	1	2	3	4	5	6	7	8	9	10
Калина	125	181	237	293	349	405	461	517	573	629
AES	48	84	120	156	192	228	264	300	336	393

Результати практичних випробувань по оцінці показників швидкодії для алгоритмів в пакеті MAPLE 14, при розмірі ключа і блоку 128 біт, представлені в таб. 3.11.

Коли вимкнена оптимізація компілятора, Калина демонструє перевагу у швидкості порівняно з AES на відстані від 12,5% до 27%, залежно від розміру блока та довжини ключа. Крім того, важливо відзначити, що Калина має суттєво вищий рівень стійкості до криптографічних атак порівняно з AES. Продуктивність нового національного стандарту України на цій платформі приблизно вдвічі вища, ніж у нових стандартах шифрування Білорусії та в 3,16 рази вища, ніж у старого стандарту ДСТУ ГОСТ 28147:2009, при однаковій довжині ключа.

Давайте проведемо порівняння продуктивності шифрів "Калина" та AES, як це робили дослідники раніше. Оскільки алгоритм Калина ґрунтується на ідеях, які також використовуються в алгоритмі AES, ми можемо аналізувати їх продуктивність з точки зору застосованих алгоритмів шифрування.

Якщо говорити про обидва алгоритми AES та Калина, то вони використовують алгебраїчні операції в кінцевих полях, і найбільш обчислювально інтенсивною операцією є множення у GF (2^8). Для оцінки їх продуктивності обидва алгоритми були реалізовані за єдиною методологією в пакеті MAPLE 14. Результати вимірювання швидкості виконання всіх необхідних процедур під час проведення 1000 циклів шифрування наведено в Таблиці 3.

Таблиця 3.11

Показники швидкодії реалізацій, порівнюваних алгоритмів в пакеті MAPLE 14, при розмірі ключа і блоку 128 біт

Розмір файлу	Калина, 10 раундів	AES, 10 раундів	Показники швидкодії
100000 байт	1099,844 с	612,235 с / 396,343 с	1,796 раз / 2,775 раз
50000 байт	540,546 с	302,000 с / 194,047 с	1,789 раз / 2,786 раз
10000 байт	106,563 с	59,594 с / 38,484 с	1,788 раз / 2,769 раз
1000 байт	10,828 с	6,000 с / 3,860 с	1,804 раз / 2,805 раз
100 байт	1,266 с	0,640 с / 0,500 с	1,978 раз / 2,532 раз
10 байт	0,297 с	0,093 с / 0,062 с	3,193 раз / 4,790 раз

На основі обчислень, представлених в Таблицях 3.9–3.11, можна прийти до висновку, що шифр "Калина" відстає в швидкості від AES більш ніж в 1,7 рази, а також більш ніж в 2,7 рази, коли використовується заздалегідь підготовлена таблиця підстановок для алгоритму AES.

Для прикладу, час шифрування файлу розміром 10000 байтів для алгоритму "Калина" склав 106,563 секунди, у порівнянні з 59,594 секундами для алгоритму AES. Але при використанні підготовленої таблиці підстановок для AES цей час скоротився до 38,484 секунд. В такому випадку співвідношення швидкостей обох шифрів складає 1,788 разів і 2,769 разів, відповідно.

3.3. Програмна (бібліотека Cryptopp) та апаратна реалізація на ПЛІС алгоритму «Калина»

Усі найбільш відомі алгоритми шифрування мають різні можливості апаратної реалізації в різних базисах. Один із найпопулярніших базисів - це програмовані логічні інтегральні схеми (FPGA). Використання FPGA дозволяє отримувати ефективні та швидкі рішення з мінімальними витратами на розробку, що робить їх дуже привабливими.

Логічна ємність та продуктивність FPGA в останні роки зростають завдяки кільком факторам, таким як перехід на більш високий технологічний рівень, підвищення ступеня інтеграції кристалів та розвиток більш високих послідовних інтерфейсів і протоколів зв'язку.

Зараз FPGA знаходять широке застосування у вбудованих обчислювальних системах для військового та аерокосмічного використання, де важливі обмеження щодо потужності, розміру та ваги пристроїв. Їх використовують у таких системах, як радіолокаційні установки, системи

радіотехнічної розвідки, системи обробки зображень та обробки сигналів. Вони особливо підходять для пристроїв, які виконують обробку сигналів та векторних або матричних обчислень. У таких застосунках головним критерієм є не вартість, а характеристики пристрою, зокрема, його швидкодія. FPGA такого типу використовуються для реалізації алгоритмів, таких як DES, 3DES, Blowfish, CAST, IDEA і багатьох інших. Для зменшення кількості дефектних чіпів були розроблені оригінальні алгоритми обробки випадкових чисел, які використовуються в FPGA [15], [17].

Ще однією цікавою рисою при розробці пристроїв на програмованих логічних інтегральних схемах (FPGA) є можливість отримання платформо-незалежного опису функціонального блоку. Цей опис може бути модифікований та покращений, перетворений для використання в різних базисах, а також використовуватися для моделювання пристроїв в освітніх та наукових цілях.

Науковці проводили дослідження щодо швидкодії шифру "Калина" з використанням різних криптобібліотек. Наприклад, були проведені вимірювання швидкодії для табличних реалізацій шифру "Калина" у криптобібліотеках "Шифр+" v2 і cppcrypto. У бібліотеці "Шифр+" v2 були досягнуті показники швидкодії, представлені в таблиці 1, і вимірювалися на процесорі Intel Core i7-6700HQ з 16 ГБ оперативної пам'яті під управлінням ОС Windows 10. Шифрування відбувалося для блоку розміром 16 КБ протягом 1 мільйона повторень.

На сьогоднішній день програмна реалізація алгоритму "Калина" доступна в популярній бібліотеці для криптоаналізу з відкритим вихідним кодом - Cryptopp. Ця бібліотека додала підтримку блокових шифрів змінного розміру у 2017 році.

Також в бібліотеці cppcrypto наведено результати вимірювання швидкодії для 64-бітної архітектури. Ця кросплатформена C++ бібліотека спрямована на досягнення максимальної швидкодії. Вимірювання швидкодії проводилися

шифруванням файлу розміром 130 КБ 100000 разів у режимі CBC на процесорі Xeon E5-1650 v3 з тактовою частотою 3,50 ГГц.

Швидкодія табличних реалізацій шифру Калина – 128 / 128 приведена в таб. 3.12.

Таблиця 3.12

Швидкодія табличних реалізацій шифру Калина – 128 / 128

Результати		Бібліотека «Шифр+» v2		Бібліотека cppcrypto					
				VC++ 2015		gcc 5.2		clang 3.7	
Cpb	Mb / s	cpb	MB / s	MB / s	Cpb	MB / s	Cpb	MB / s	cpb
10,41	2611,77	20,28	128,22	179	19,55	236	14,83	206	16,99

Крім того, було проведено дослідження щодо реалізації шифру "Калина" з використанням SIMD-інструкцій AVX-256 і AVX-512. Вимірювання швидкодії виконувалось на процесорі Intel Xeon Skylake-SP з тактовою частотою 2,0 ГГц. Код був написаний на мові C++, використовуючи компілятори з комплекту Microsoft Visual Studio 2019 (MSVC) та gcc 7.3.0 (GCC). Для зменшення впливу переключення контексту процесора було виконано багатократне шифрування (100 мільйонів повторень). Було оцінено як табличні реалізації шифрів на базі SIMD-інструкцій, які є вразливі до кеш-атак (таб. 3.13), так і стійкі до кеш-атак реалізації з покроковим виконанням операцій шифру (таб. 3.14).

Таблиця 3.13

Табличні реалізації шифру «Калина» (Not Constant Time)

К-ть блоків	AVX–256				AVX–512			
	GCC, cpb		MSVC, cpb		GCC, cpb		MSVC, cpb	
	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC
16-way	-	-	-	-	5,61	7,09	5,49	7,78

8-way	9,34	10,6	8,42	10,8	5,64	6,95	5,83	7,30
4-way	9,22	10,4	8,72	9,94	8,25	9,97	8,84	10,5
2-way	10,88	12,6	10,6	12,4	13,6	15,94	14,7	16,8
1-way	19,50	22,3	19,5	22,0	23,6	26,8	25,3	28,2

З представлених результатів слідує, що використання технології AVX–256 для імплементації шифру «Калини» виправдане за умови необхідності гарантування стійкості до кеш-атак. Це особливо актуально для більшості процесорів, які не мають підтримки AVX–512, тим більше, що перехід до AVX–512 у цьому випадку не дає відчутного приросту швидкодії. Через те, що технологія SSE–128 відчутно поступається у швидкодії AVX–256 / AVX–512 ми її не порівнювали.

Таблиця 3.14

Покрокові реалізації шифру «Калина» (Constant Time)

К-ть блоків	AVX–256				AVX–512			
	GCC, cpb		MSVC, cpb		GCC, cpb		MSVC, cpb	
	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC
16-way	-	-	-	-	8,32	15,14	8,78	15,55
8-way	16,42	25,91	10,63	17,19	11,08	16,64	11,73	18,25
4-way	21,59	29,34	15,09	22,50	18,13	24,69	19,72	25,13
2-way	34,75	42,69	24,13	30,94	28,31	43,38	30,00	47,50
1-way	38,13	45,38	35,50	43,88	52,38	74,00	57,25	78,63

Найскладнішим кроком в реалізації bitsliced-шифрування, який має значний вплив на загальну швидкодію, є логічне представлення таблиці нелінійної заміни S-Box. Давайте розглянемо різні підходи до представлення S-Box у вигляді комбінаційної логічної схеми з мінімальною кількістю двовходових вентилів AND, OR, XOR, NOT.

На сьогоднішній день світовим стандартом для мінімізації функцій із великою кількістю змінних є програма Espresso (Espresso logic minimizer), яка використовує евристичний алгоритм. Крім того, існують похідні програми, такі

як BOOM, яка має ще вищу швидкодію, Logic Friday, яка надає графічний інтерфейс до алгоритму Espresso, а також ABC і інші.

Espresso надає результати, які практично дуже близькі до глобального мінімуму, при цьому вона використовує значно менше пам'яті та часу порівняно з іншими методами. Крім того, Espresso майже не має обмежень на кількість вхідних і вихідних змінних. Цей алгоритм широко використовується в багатьох інструментах і CAD-пакетах для синтезу логічних схем під час мінімізації (FPGA, ASIC) [36].

Однак важливо відзначити, що використання Espresso також має свої обмеження. Зокрема, в ньому використовуються лише операції AND, OR, NOT, і операція XOR може бути впроваджена вже після процесу мінімізації, а не в процесі. Крім того, Espresso не враховує обмеження на кількість входів вентилів. Таким чином, вибір між Espresso і іншими методами мінімізації повинен бути зроблений з урахуванням конкретних вимог і обмежень проекту, оскільки кожен із цих методів має свої переваги і недоліки. Результати мінімізації S-Box-ів «Калини» із допомогою Logic Friday представлено в таб. 3.15.

Таблиця 3.15

Мінімізація S-Box шифру «Калина» в програмі Logic Friday

Таблиця	Prime Implicants	Total GE
Sbox3	231	879
Sbox2	234	854
Sboxl	231	857
SboxO	234	841

Варто враховувати, що в число вентилів включаються і елементи NAND, які вимагають дві процесорні інструкції в програмному поданні, що призводить до збільшення загальної кількості операцій, ніж та, яка подана в таблиці. Отримані результати не відповідають вимогам щодо потенційної швидкодії, тому необхідно розглядати альтернативні підходи.

Сучасні мікропроцесори з архітектурою x86–64 підтримують кілька наборів векторних інструкцій, таких як SSE, AVX / AVX2, AVX–512. Давайте коротко проаналізуємо їх можливості та відповідність використанню.

SSE (Streaming SIMD Extensions) - це набір SIMD-інструкцій, які додають в архітектуру процесора 8 регістрів xmm0-xmm7 та понад 70 інструкцій для операцій над 32-бітними даними типу float. SSE пізніше було розширено новими розширеннями, такими як SSE2, SSE3, SSSE3 та SSE4, які додали нові інструкції та покращили ефективність. SSE також підтримується більшістю сучасних процесорів x86–64.

AVX (Advanced Vector Extensions) - це розширення системи команд x86 – мікропроцесорів, доступне в процесорах з 2011 року. AVX включає нові інструкції та розширює розрядність SIMD-регістрів до 256 біт. AVX2 подальше розширення AVX з підтримкою цілочисельних операцій. Обидва ці набори інструкцій підтримуються багатьма процесорами x86–64 і надають високий рівень паралелізму.

AVX–512 є подальшим розширенням 256-бітних інструкцій AVX. Це розширення має 512-бітні регістри та нові інструкції. Проте наразі воно підтримується обмеженою кількістю процесорів.

Щодо ПЛІС (програмована логічна інтегральна схема), вони мають численні переваги, такі як висока швидкість, низька вартість, універсальність та можливість легкої модифікації проєктів. Вони є важливим ресурсом для автоматизованого проєктування і дозволяють працювати з різними параметрами.

Узагальнюючи, є багато можливостей для оптимізації та вибору векторних інструкцій та використання ПЛІС для покращення продуктивності та ефективності операцій. [36].

Для даної роботи вибрано ПЛІС сімейства Xilinx – тип Xilinx Xilinx Zynq – 7000 AP SoC XC7Z020-CLG484.

Загальний вигляд стенду з встановленою ПЛІС (**Ошибка! Источник ссылки не найден.**а). Стенд працює під керуванням персонального комп’ютера. Ресурси, які знаходяться у ПЛІС, і які можна задіяти для створення шифропроцесора (**Ошибка! Источник ссылки не найден.**б).

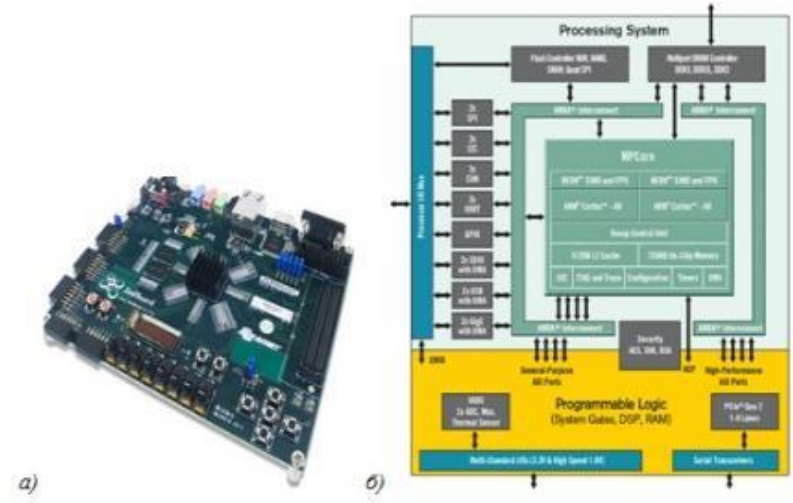


Рис. 3.9. Відлагоджувальний модуль з ПЛІС а) та структурна схема ПЛІС б)

Сучасні ПЛІС - це, по суті, кристали з універсальними мікропроцесорами і логічними елементами, які користувач може програмувати для своїх потреб. Результатом проєктування ПЛІС є файл конфігурації, який містить дані про їхню структуру. Ця інформація записується на постійний запам'ятовуючий пристрій (ПЗП) і використовується для налаштування ПЛІС при кожному включенні. Після вимкнення живлення дані втрачаються, і ПЛІС працює з новими налаштуваннями при наступному запуску.

3.4. Приклад реалізації алгоритму Калина на мікроконтролері

Сучасні стандарти шифрування враховують особливості свого використання та прагнуть забезпечити високий рівень безпеки та продуктивності. Тому важливо аналізувати алгоритми з великими ключами, які

є стійкими до криптоаналізу, або ті, що дозволяють ефективно захищати інформаційні системи.

Таблиця 3.16

Результати аналізу стійкості шифру «Калина» із розміром блоку 512 бітів

Метод криптоаналізу	Найменша кількість циклів, для якої шифр є стійким	Показники атак		
		Максимальна кількість циклів	Обчисл. складність, екв. оп. шифрув.	Пам'ять, байтів
Диференційний	9	8	2490	
Лінійний	9	7	2470,4	
Усіч. диференц.	4	3		
Інтегральний	7	6	2137	264+5
Нездійсн. дифер.	6	5	260	266
Бумеранг	7	6	2340	

Тестування виконувалося на комп'ютері під управлінням 64-бітової ОС Linux (Ubuntu 12.04) з процесором Intel Core i5–4670@3.40GHz. Результати тестування швидкодії програмної реалізації наведено у таб. 3.17.

Таблиця 3.17

Швидкодія програмної реалізації блокових шифрів

№	Intel Core i5–4670@3.40GHz	
	Блоковий шифр	Швидкодія, Мбіт/с
1	Kalyna – 128 / 128	2611.77
2	STB 34.101.31–2011 (BelT)	1055.92
3	Kuznyechik	1081.08
4	Kalyna – 512 / 512	1386.46
5	Kalyna – 256 / 512	1560.89
6	Kalyna – 128 / 256	1779.52
7	AES–256	1993.53
8	Kalyna – 256 / 256	2017.97
9	AES–128	2525.89
10	GOST 28147–89	639.18

Важливо зазначити, що довгий ключ є характеристикою алгоритму. Як вбачається з результатів тестування, він може бути менш продуктивним порівняно з алгоритмами, що використовують короткі ключі. Проте він надає великий резерв стійкості до криптоаналітичних атак.

Тестування швидкодії проводилося на комп'ютері з процесором Intel Core i7–6700HQ 2,6 ГГц та 16 ГБ оперативної пам'яті під управлінням ОС Windows 10. Було шифрування блоку розміром 16 КБ протягом 1 мільйона повторень.

Аналіз швидкодії проводився на процесорі Intel Xeon Skylake-SP 2,0 ГГц з використанням мови програмування C++. Код компілювався з використанням компіляторів Microsoft Visual Studio 2019 (MSVC) та gcc 7.3.0 (GCC). Для зменшення впливу переключення контексту процесора виконувалося багаторазове шифрування (100 мільйонів повторень). Оцінка стійкості до кеш-атак проводилася для реалізацій з послідовним виконанням операцій шифру.

З наведених результатів видно, що використання технології AVX–256 для імплементації шифру "Калина" є обґрунтованим, особливо в разі потреби в гарантованій стійкості до кеш-атак. Це особливо важливо для більшості процесорів, які не підтримують AVX–512, оскільки перехід до AVX–512 не призводить до значного покращення швидкодії. Технологія SSE–128 помітно відстає від AVX–256 / AVX–512 за швидкодією.

Таблиця 3.18

Покрокові реалізації шифру «Калина» з використанням SIMD інструкцій AVX–512

К-ть блоків	AVX–512			
	GCC, cpb		MSVC, cpb	
	ENC	DEC	ENC	DEC
1	52,38	74	57,25	78,63
16	8,32	15,14	8,78	15,55
8	11,08	16,64	11,73	18,25
4	18,13	24,69	19,72	25,13
2	28,31	43,38	30	47,5

Для проведення тестування було використано наступні платформи:

1. Платформа на основі 32-розрядного ядра ARM MCU–STM32G071RBT.
2. Персональний комп'ютер під управлінням 64-бітової операційної системи Linux (Ubuntu 20.04) з процесором Intel Core i7–1185G7 з тактовою частотою 3.00 ГГц.
3. Персональний комп'ютер під управлінням 64-бітової операційної системи Windows 10 з процесором Intel Core i5 6300HQ з тактовою частотою 2.3 ГГц.

Система забезпечує зв'язок з персональним комп'ютером, перетворюючи сигнали USB у телеграми UART. Прошивка буде завантажуватися за допомогою KEIL uVisionIDE з використанням офіційного налагоджувача ST-Link V2 від STMicroelectronics, як показано на схемі на рис. 3.10.

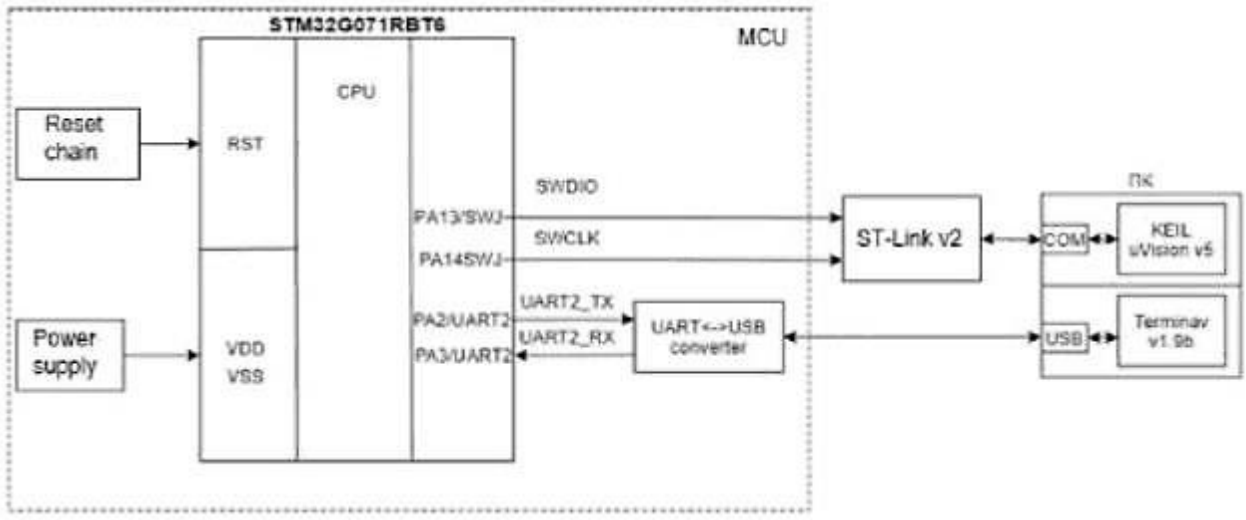


Рис. 3.10. Функціональна схема мікроконтролера

У документації криптобібліотеки представлені результати вимірювання швидкодії, які наведено в таблиці 3.19 для архітектури 64-біт. Ця кросплатформена бібліотека, реалізована мовою програмування C++, спрямована на досягнення максимальної продуктивності. Для вимірювання

швидкодії використовувалось шифрування файлу розміром 130 Кбайт 1 мільйон разів в режимі CBC (режим зчеплення шифроблоків) на процесорі Xeon E5–1650 v3 з тактовою частотою 3,50 ГГц [38]. Результати швидкодії представленні в мегабітах на секунду (Мбіт/с), див. (таб. 3.19).

Таблиця 3.19

Порівняння швидкодія програмної реалізації блокового шифру «Калина» (256, 512) та Kalyna (512, 512)

Модифікація	Швидкодія, Мбіт/с			
	Intel Core i5–4670@3.40GHz (Ubuntu 12.04) [25]	Intel Core i7–1185G7@3GHz (Ubuntu 20.04)	Intel Core i5–6300HQ@2.3GHz (Windows 10)	ARM MCU–STM32G071RBT
Kalyna (512, 512)	1386.46	1787.14	1065.81	24.9
Kalyna (256, 512)	1560.89	1922.83	1220.11	28.7

При порівнянні швидкодії різних варіацій алгоритму Калина (256, 512), було помічено, що алгоритм Калина (256) виявився найшвидшим. Це вкрай очевидно, оскільки він використовує блок та ключ меншого обсягу. З іншого боку, алгоритм Калина (512, 512) був найповільнішим через значний обсяг даних для обробки, але при цьому надавав найвищий рівень захищеності інформації.

USB UART FT232RL, який ми використовували у розробці, має вбудований EEPROM об'ємом 1024 біти. Він також включає приймальний буфер розміром 256 байт, який передає буфер розміром 512 байт. Цей пристрій конфігурується через контакт C BUS I/O та працює за протоколом USB. Плата обладнана світлодіодами rx і Tx, які індикують передачу та прийом даних через USB.

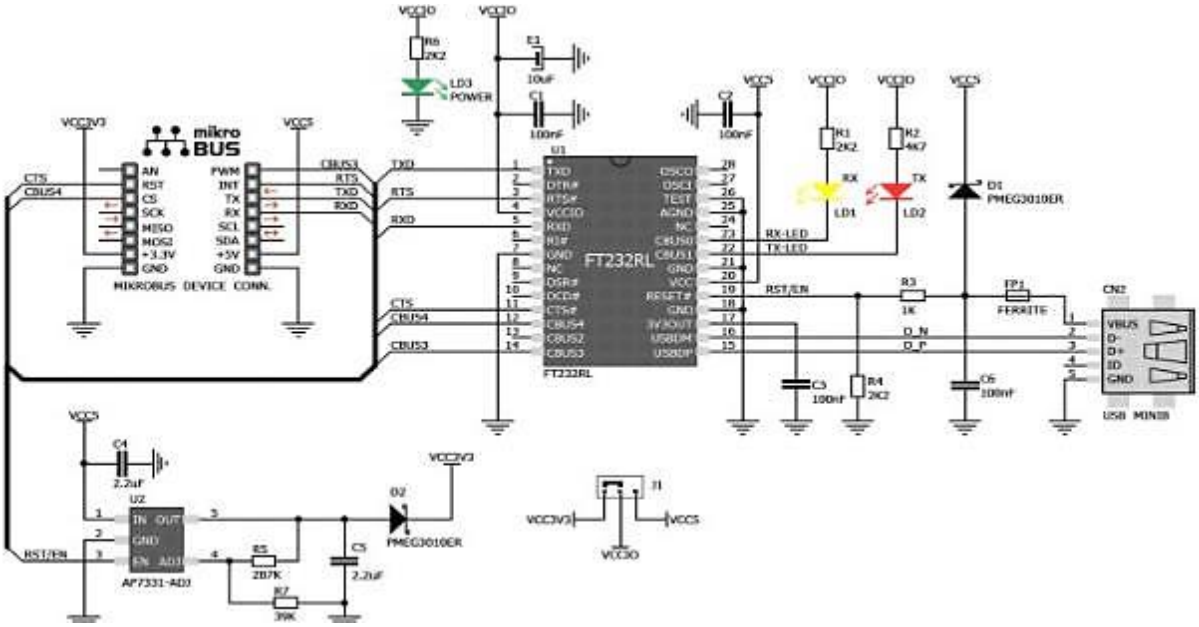


Рис. 3.11. Принципова схема підключення плати USB UART FT232RL у проєкті

Приведемо фрагмент коду для компіляції USB UART FT232RL

```
char uart_rx_data;
void main() {
  UART1_Init(9600); // Ініціалізувати UART модуль зі швидкістю 9600 біт/с
  Delay_ms(100);  // Зачекати, поки UART модуль стабілізується
  UART1_Write_Text("Start");
  UART1_Write(13); // CR (Carriage Return)
  UART1_Write(10); // LF (Line Feed)
  while (1) {
    if (UART1_Data_Ready()) {
      uart_rx_data = UART1_Read(); // Зчитати отримані дані
      UART1_Write(uart_rx_data);  // Відправити дані через UART
    }
  }
}
```

3.5. Програмна реалізація та симуляція в середовищі Aldec Riviera-Pro

На початку програмної реалізації проведено інсталяцію програму Aldec Riviera-PRO та проведемо компіляцію алгоритму «Калина» в її оболонці (рис. 3.13).

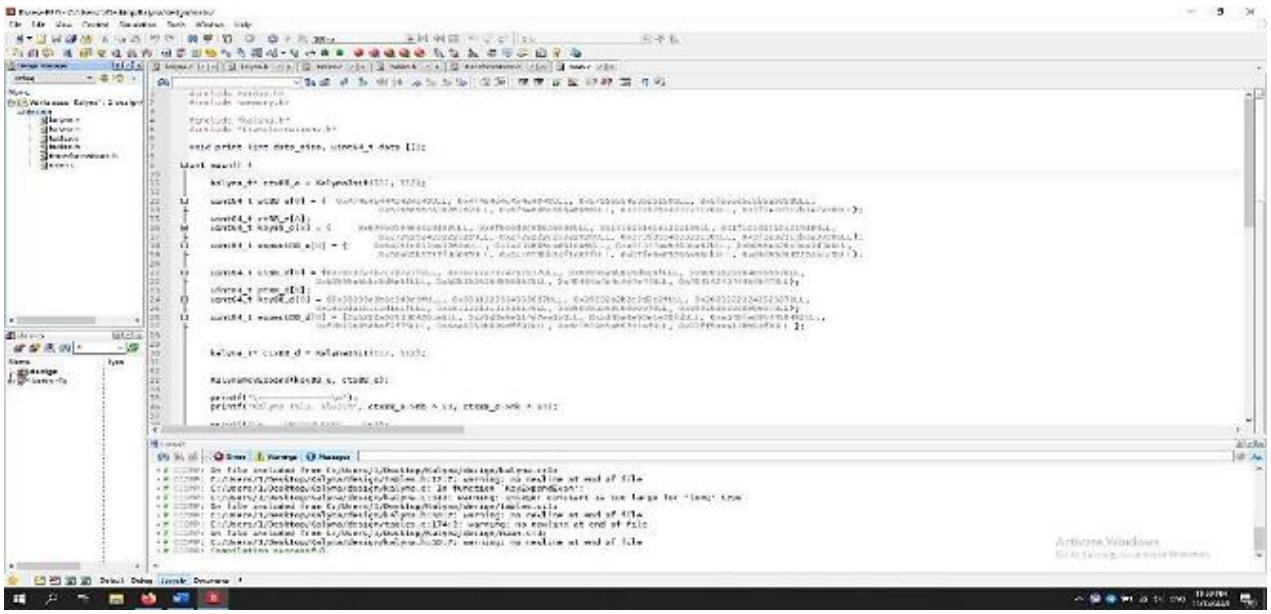


Рис. 3.13. Компіляція програми алгоритму Калина з ключем 512 / 512 в програмі Aldec Riviera-PRO

На рис. 3.14 представлено фрагмент коду алгоритму «Калина» на мові C++

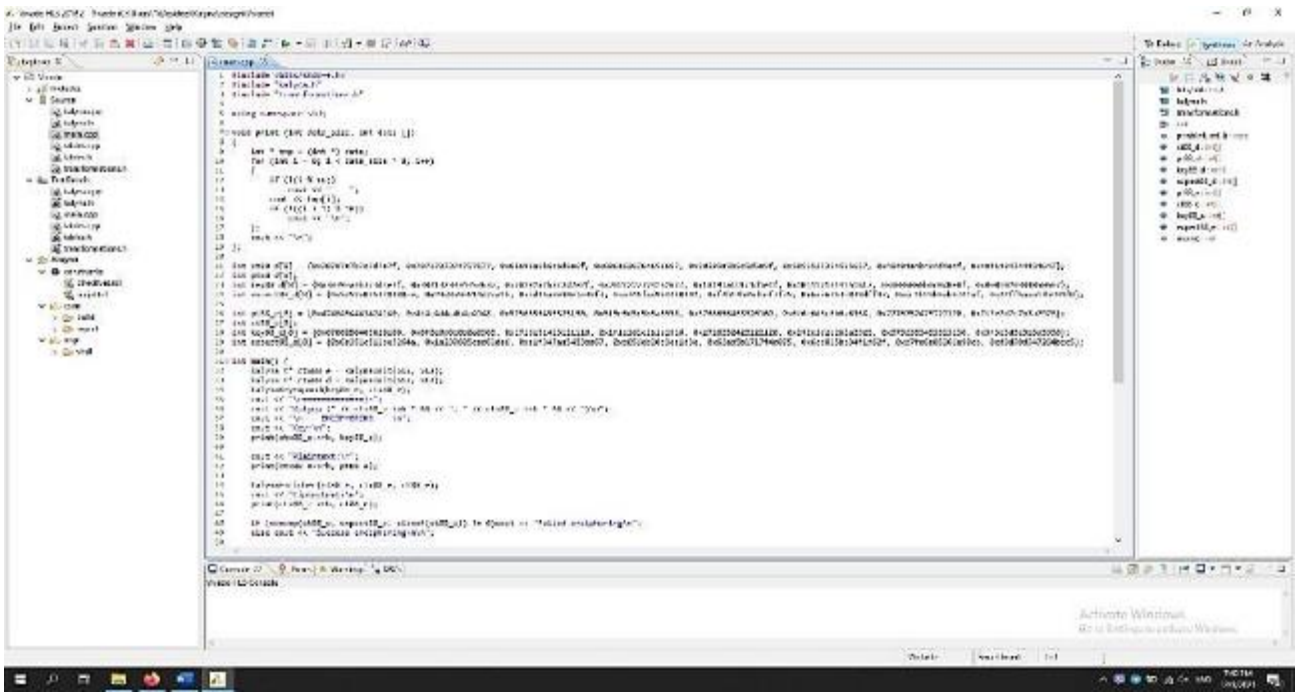


Рис. 3.14. Фрагмент коду алгоритму Калина з ключем 512 / 512 на мові C++

В подальшому здійснено переведення коду алгоритму «Калина» в мову програмування С (рис. 3.15).

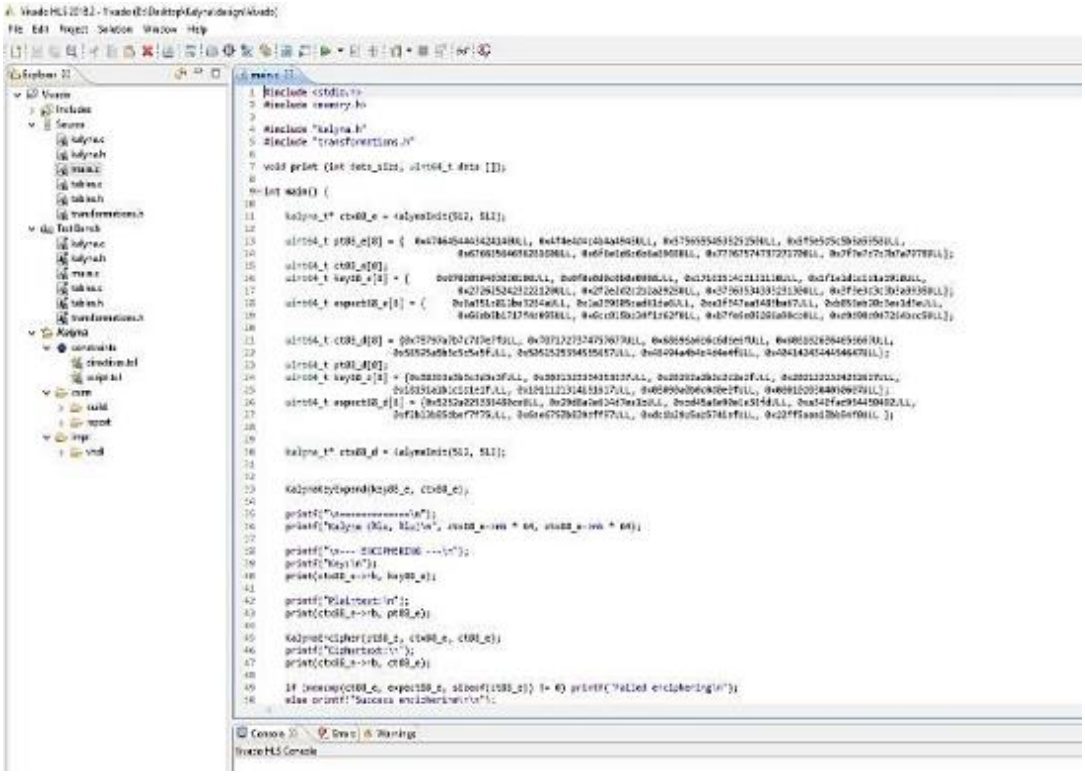


Рис. 3.15. Фрагмент коду алгоритму Калина з ключем 512 / 512 на мові C

Відкриття форматованого коду алгоритму «Калина» в програмі XILINX Vivado HLS (рис. 3.16).

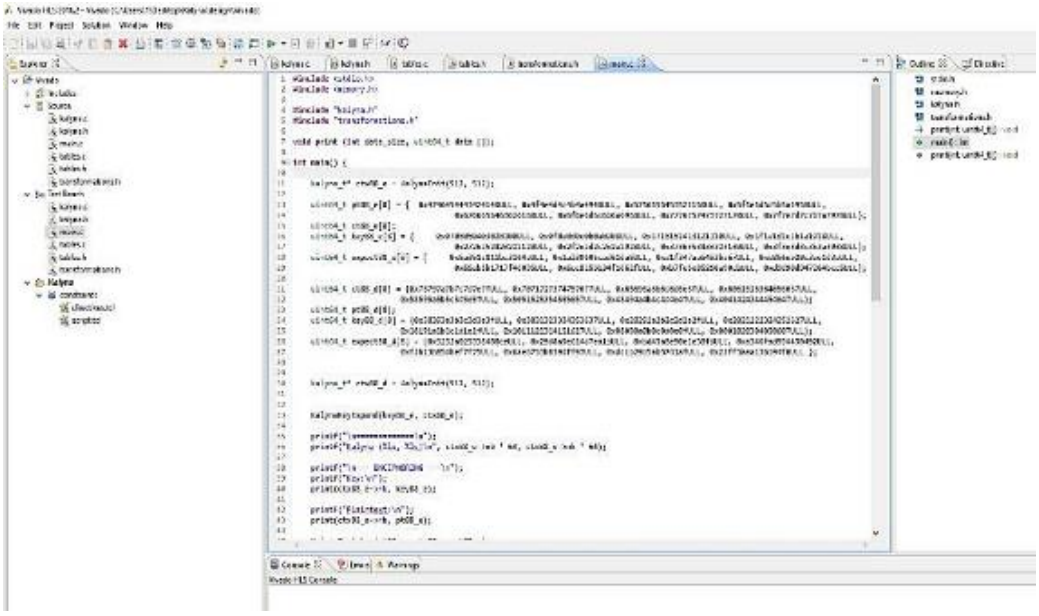


Рис. 3.16. Відкриття програми в середовищі XILINX Vivado HLS для подальшої роботи

Здійснено симуляцію програми алгоритму «Калина» з ключем 512 / 512 в програмі XILINX Vivado HLS (рис. 3.17–3.20).

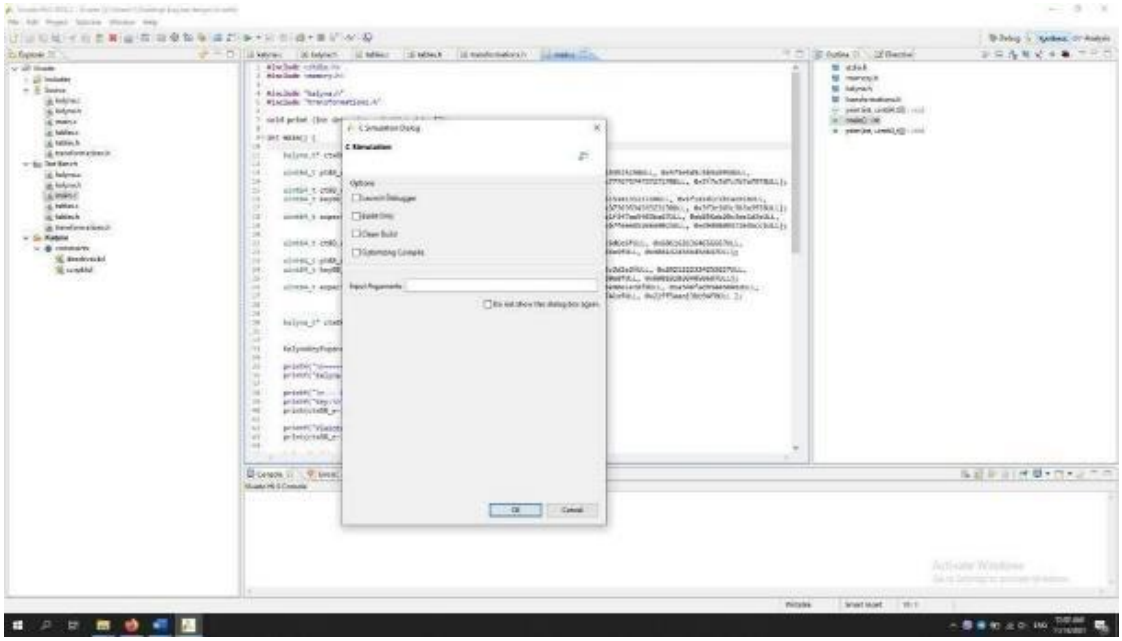


Рис. 3.17. Підготовка симуляції коду алгоритму «Калина» з ключем 512 / 512 в програмі XILINX Vivado HLS

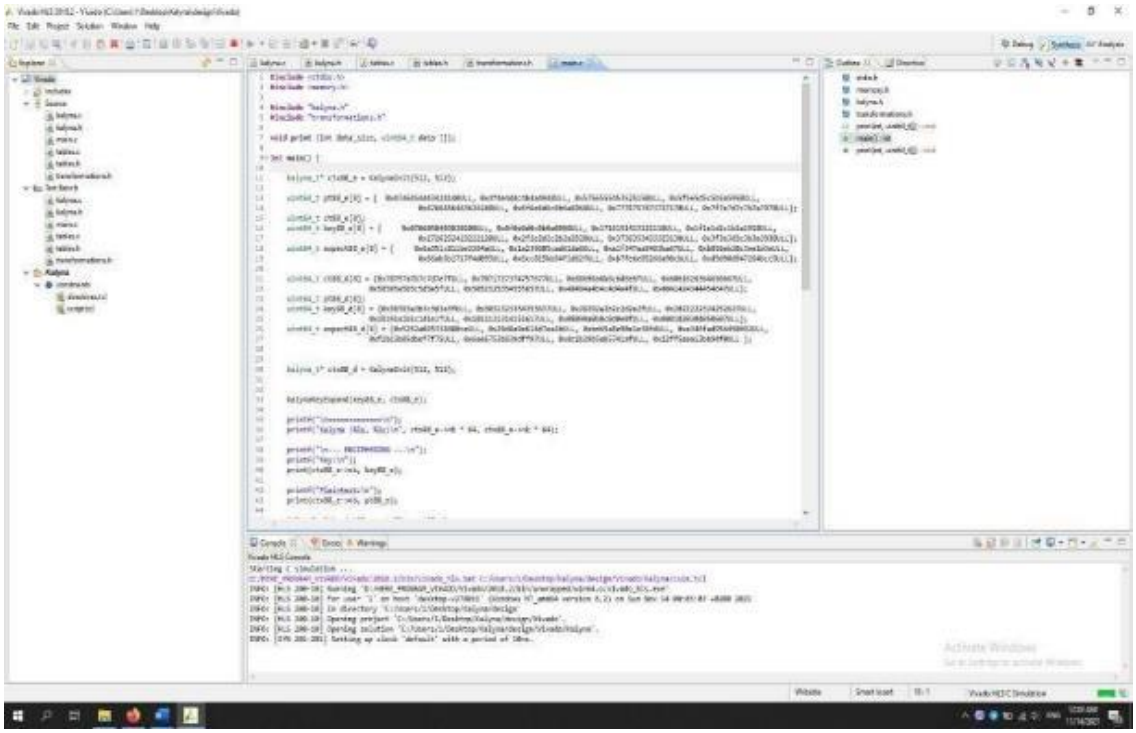


Рис. 3.18. Початок симуляції коду алгоритму «Калина» з ключем 512 / 512 в програмі XILINX Vivado HLS

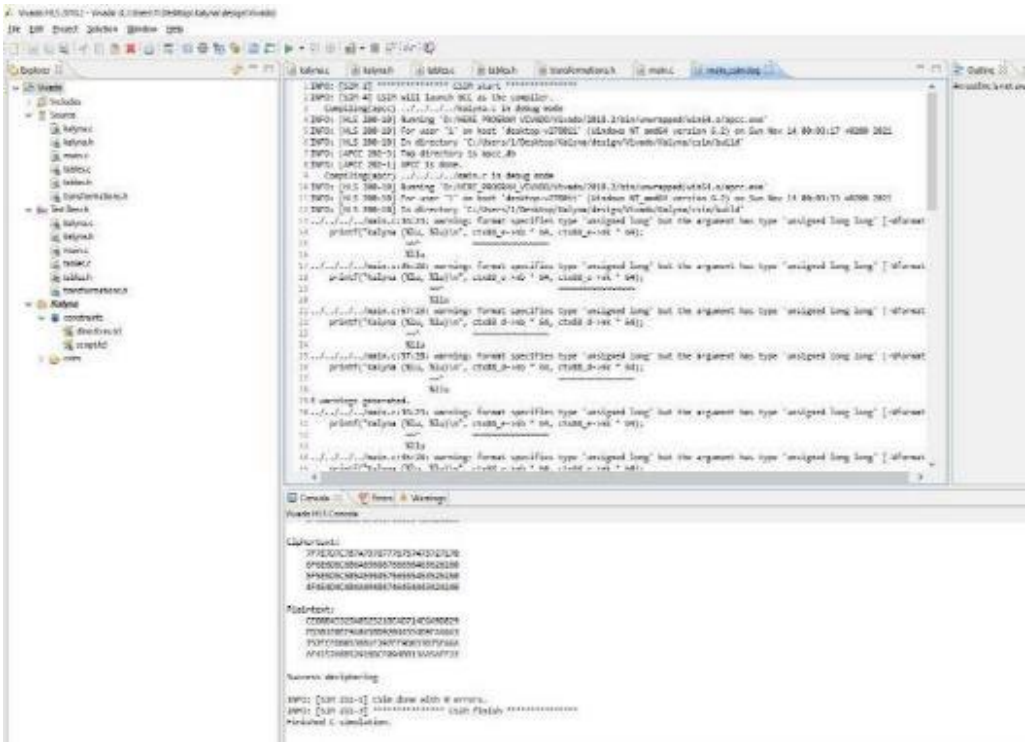


Рис. 3.19. Закінчення симуляції, вікно звіту про операцію успішної симуляції коду алгоритму «Калина» з ключем 512 / 512

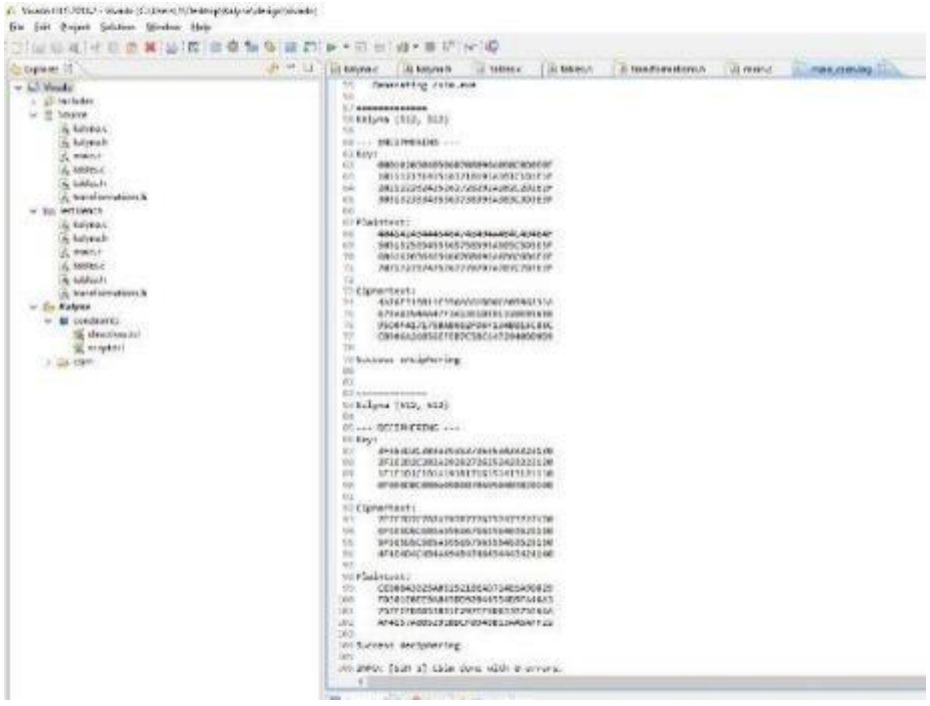


Рис. 3.20. Виведення результату симуляції, шифрований і дешифрований текст

У цьому проекті мікроконтролер виступає як основний компонент, оскільки саме на ньому буде виконуватися алгоритм. Ми використовуватимемо його обчислювальні можливості процесора, а також інтерфейси UART для обміну даними з зовнішнім світом і SDW для відлагодження під час розробки. Приємним аспектом є наявність модуля захисту пам'яті, який запобігає розсинхронізації під час розробки програми.

3.6. Особливості симулювання коду у середовищі Xilinx Vivado. Hls

У нашому дослідженні ми переводимо код алгоритму "Калина", написаний на мові C++, у мову C, і потім компілюємо його в VHDL-файл. Цей VHDL-файл можна використовувати для програмування вибраної ПЛІС. Для цього ми використовуємо програмне забезпечення Xilinx Vivado HLS.

Давайте розглянемо середовища, які ми використовуємо. На сьогоднішній день Xilinx розробляє сучасний продукт - програмне забезпечення для конфігурації кристалів та розробки проєктів. Повний перехід на нове покоління систем автоматизованого проєктування (Integrated Synthesis Environment, ISE) відбувся в 2002 році. Раніше ISE використовувалось як альтернатива Foundation Series (попередня версія САПР). ISE має численні переваги, включаючи значне зменшення часових витрат на розробку, покращені методи проєктування, які підвищують ефективність результату [40].

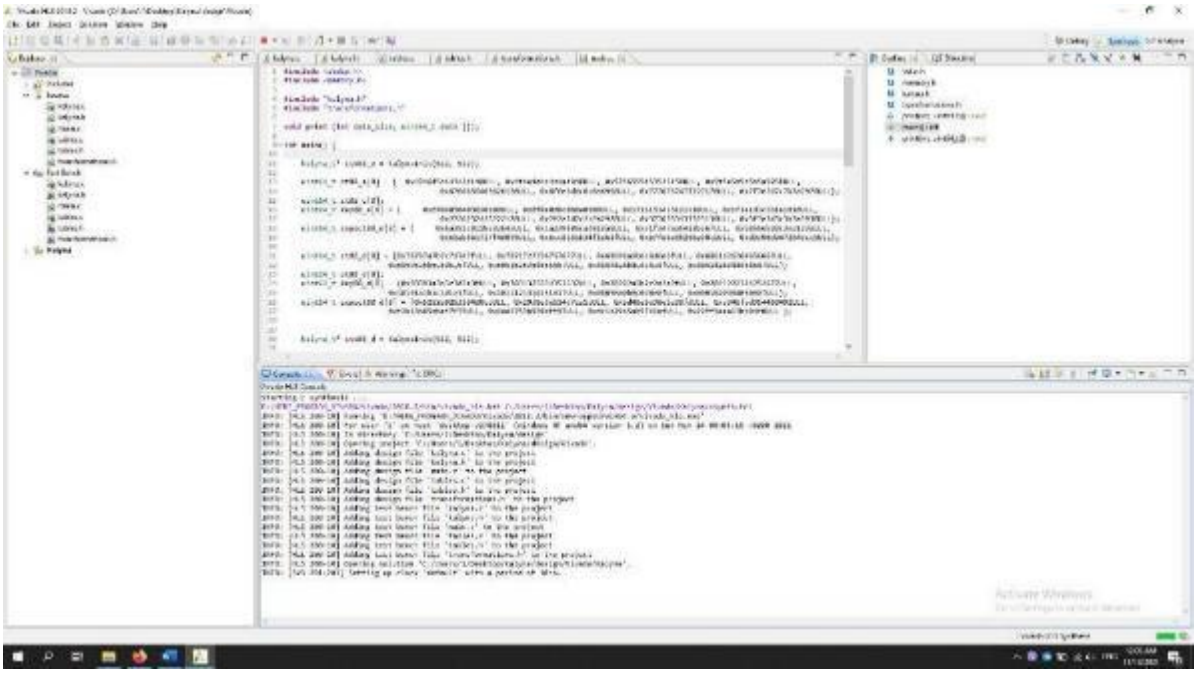


Рис. 3.21. Початок синтезу коду VHDL в програмі XILINX Vivado HLS

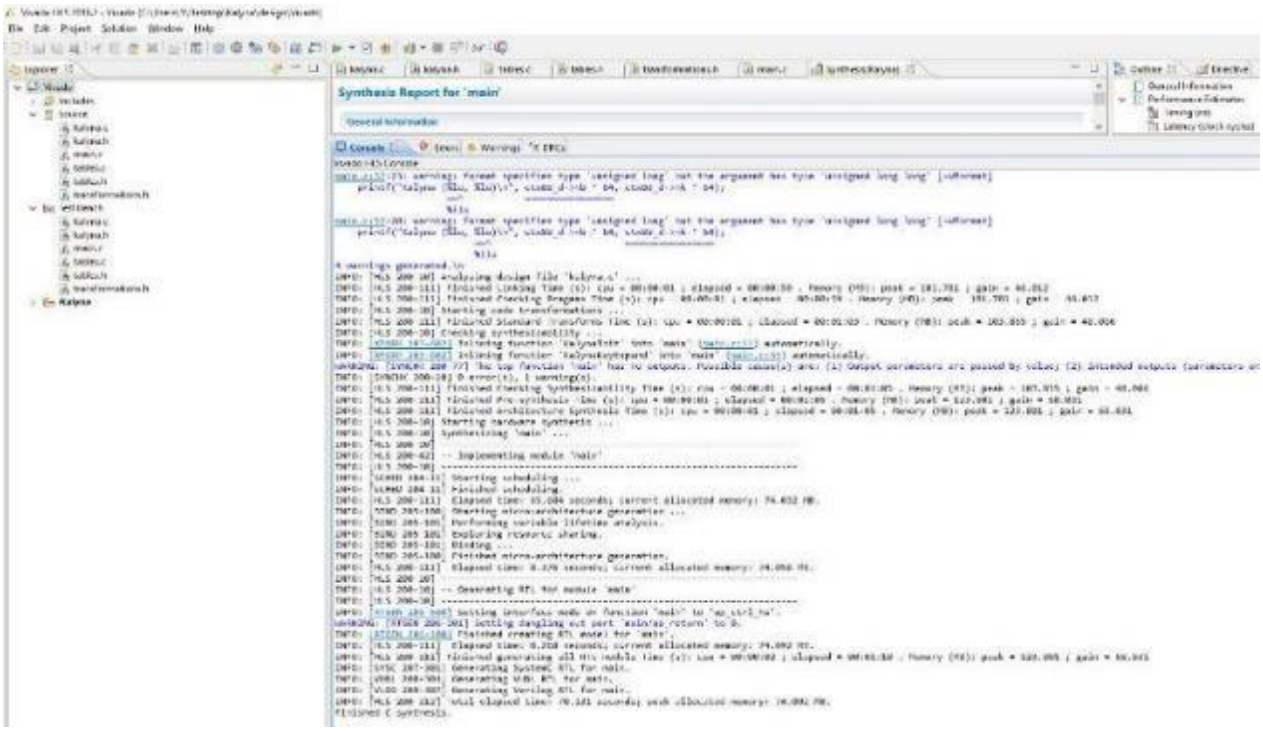


Рис. 3.22. Кінець синтезу програми, всі операції представлені в консольному вікну

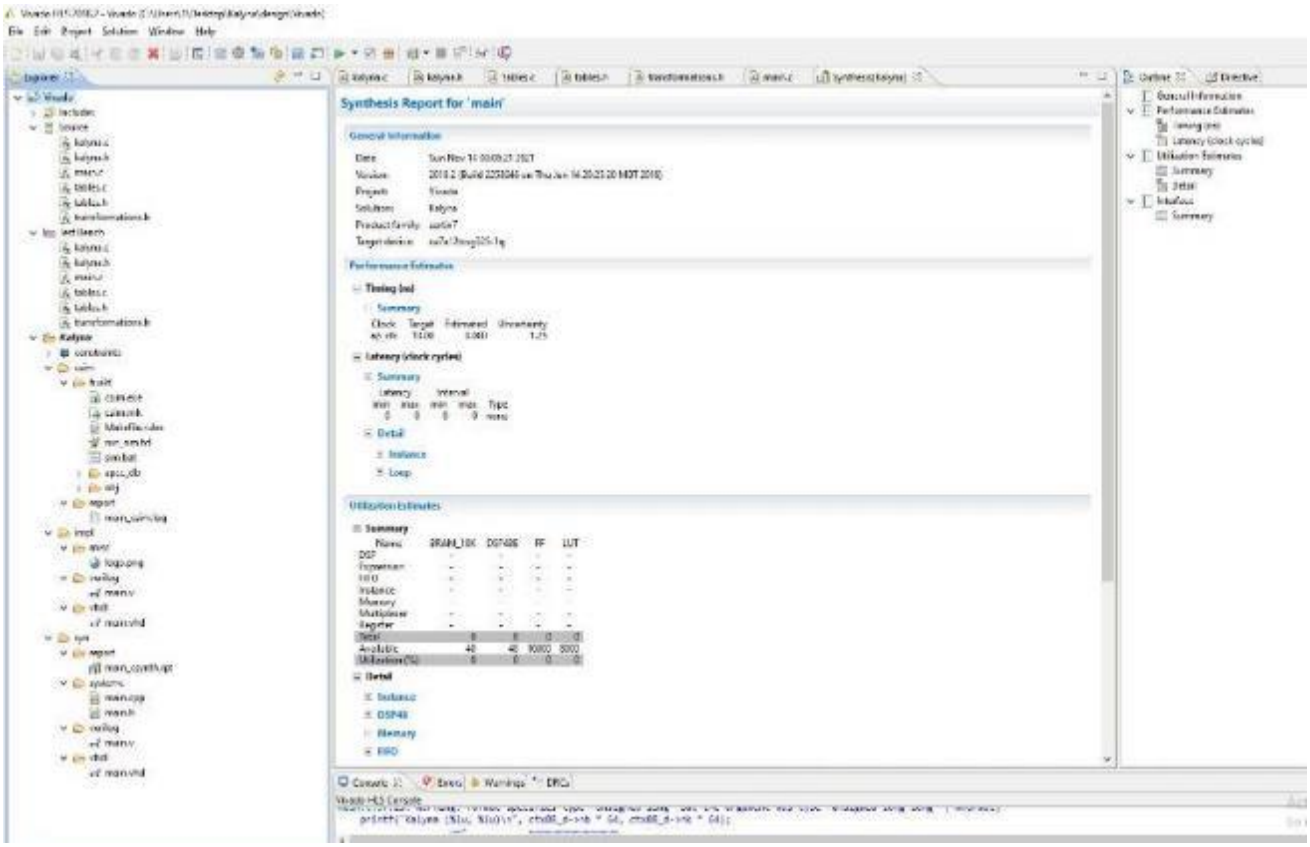


Рис. 3.23. Звіт про успішний синтез програми алгоритму Калина з ключем 512 / 512

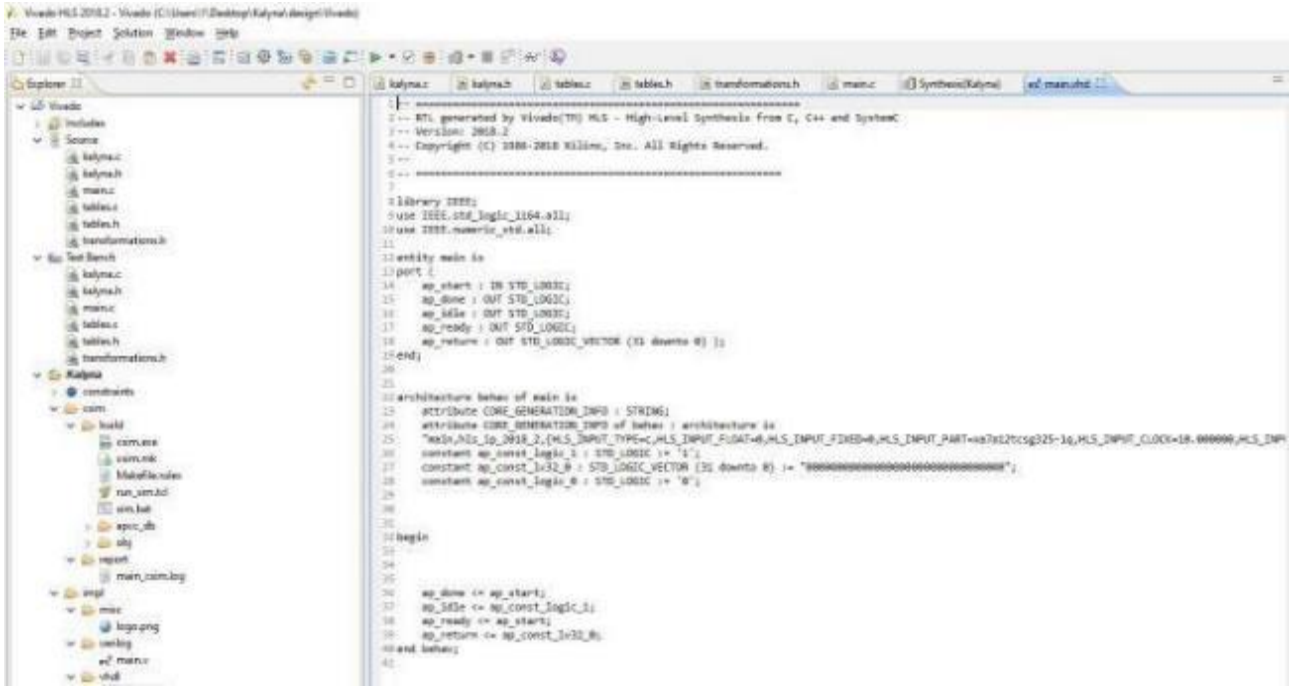


Рис. 3.24. Код на мові VHDL автоматично згенерований програмою XILINX Vivado HLS в результаті синтезу

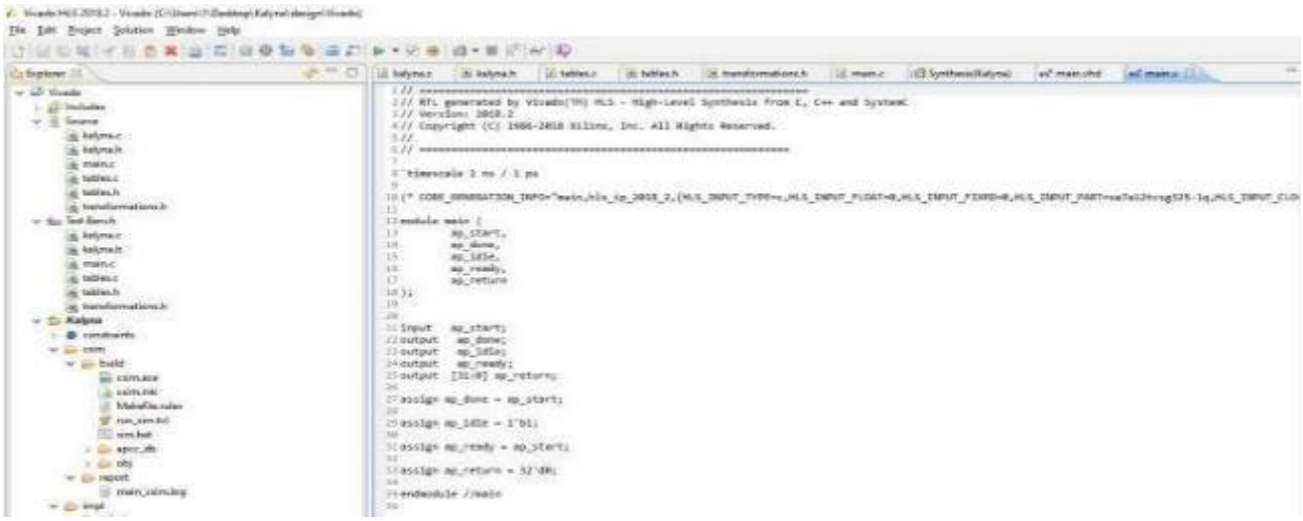


Рис. 3.25. Отриманий код на мові Verilog, що згенерований програмою XILINX Vivado HLS

Серед інструментів компанії Xilinx для розробки FPGA існують наступні варіанти:

1. Foundation ISE - це повноцінний програмний продукт для розробки FPGA, який підтримує всі типи ПЛІС і сімейства. Foundation ISE є найбільш розширеним серед інших CAD-систем від Xilinx.

2. Alliance ISE - ця конфігурація призначена для взаємодії з CAD-системами інших виробників і підтримує всі кристали Xilinx.

3. ISE Design Suite (WebPack) - це безкоштовний інструмент, доступний на офіційному сайті Xilinx. Він дозволяє розробляти проекти для CPLD і FPGA, але має обмеження на логічну ємність - до 300 000 системних вентилів [41].

4. Vivado High Level Synthesis (VHLS) - це нова CAD-система від Xilinx, спрямована на спрощення розробки цифрових пристроїв за допомогою високорівневих мов програмування, таких як C, C++, System C. Основна мета - полегшити процес розробки для програмістів, які володіють високорівневим програмуванням. Проте, при роботі з FPGA, це може створювати виклики для програмістів, які не мають досвіду роботи з апаратурою.

Усі ці CAD-системи відрізняються підтримкою різних типів ПЛІС і сімейств, а також мають різні інструменти для розробки. Однак вони мають спільний інтерфейс, за винятком Alliance ISE.

Для нашого дослідження ми обрали просту у використанні Vivado High Level Synthesis (VHLS), яка підходить для розробки на високорівневих мовах програмування, таких як C, C++, і System C. За допомогою цього середовища ми можемо перетворити прототип програми, написаної на C++, у VHDL або Verilog файл, який можна використовувати для програмування вибраної FPGA, наприклад, моделі XC7Z020-CLG484. На рисунку 3.20 показано інтерфейс середовища Vivado HLS.

Вибір було зроблено на користь середовища Vivado High Level Synthesis компанії Xilinx. Ми використовуємо модель ПЛІС XC7Z020-CLG484 та відладну плату, на якій вже встановлено дану ПЛІС - ZedBoard Zynq-7000 SoC ZC702 Evaluation Board [43].

РОЗДІЛ 4.
РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ ШИФРУВАННЯ ДАНИХ У БАГАТОКОРИСТУВАЦЬКІЙ ГРІ

4.1. Проєктування архітектури програмної реалізації

Відповідно до основної мети цього дослідження, наша робота спрямована на створення програмного модуля для шифрування даних у багатокористувацькій грі. Досягнення цієї мети можливо за допомогою виконання наступних завдань: аналіз предметної області, проєктування програмного модуля для шифрування, реалізація програмної частини, вдосконалення системи та підтримка продукту. Графічне представлення цих цілей можна переглянути на рисунку 4.1.

Рис. 4.1. Зовнішній вигляд дерева цілей згідно досліджуваної проблематики в роботі

У наступному етапі системного аналізу об'єкта дослідження ми створюємо матрицю для порівняння критеріїв з типами систем (таблиця 4.1). Шляхом нормалізації елементів у стовпці та обчисленням середнього значення рядка ми визначаємо вектор пріоритетів.

Таблиця 4.1

Ранжування критеріїв в дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

№	Назва критерію	Порівняння критеріїв						Вектор пріоритету
		1	2	3	4	5	6	
1	Актуальність	1	1 / 4	1 / 3	5	1 / 4	2	0,15
2	Точність	1	3	7	7	1	1	0,26
3	Зручність	1 / 7	4	1	5	1 / 2	1 / 7	0,114
4	Коректність	2	1	4	1 / 6	3	1	0,17
5	Доступність	1 / 3	1 / 7	1 / 5	1 / 5	1 / 8	1	0,04
6	Якість	1 / 2	7	1	6	1	8	0,33

Після чого здійснюється попарне порівняння критерію з типами систем. Таким чином було здійснене попарне порівняння систем для кожного визначеного критерію (таб. 4.2–4.8).

Таблиця 4.2

Матриця попарного порівняння систем за критерієм «Актуальність» в дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

№	Назва системи	Порівняння систем					Вектор пріоритету
		1	2	3	4	5	
1	Інформаційно- консультаційні	1 / 2	1 / 7	6	1 / 6	1 / 7	0,4
2	Інформаційно-керівні	6	1 / 3	5	1 / 4	1 / 4	0,2
3	Інформаційно-пошукові	1 / 5	1 / 6	1 / 2	1 / 4	1 / 6	0,3
4	Інтелектуальні інформаційні	4	1 / 3	5	1	1 / 2	0,5
5	СППР	5	3	5	1 / 2	1	0,45

Таблиця включає в себе матрицю попарного порівняння різних систем за критерієм "Актуальність" у контексті створення програмного модулю для шифрування даних у багатокористувацькій грі. У таблиці наведено назви систем, їх порівняння між собою за важливістю, а також обчислений вектор пріоритетів для кожної системи.

Таблиця 4.3

Матриця попарного порівняння систем за критерієм «Точність» в дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

№	Назва системи	Порівняння систем					Вектор пріоритету
		1	2	3	4	5	
1	Інформаційно- консультаційні	1 /2	6	1/4	1 / 5	7	0,74
2	Інформаційно-керівні	1/ 2	3	1/ 4	1 / 4	6	0,28
3	Інформаційно-пошукові	1/ 5	1 / 7	1/ 5	1 / 3	5	0,22
4	Інтелектуальні інформаційні	1/ 4	3	3	1 / 8	1 / 3	0,27
5	СППР	1/ 6	1 / 2	1/ 4	1 / 5	1 / 4	0,21

Ця таблиця містить матрицю попарного порівняння різних систем за критерієм "Точність" у контексті створення програмного модулю для шифрування даних у багатокористувацькій грі. У таблиці представлені назви систем, їх порівняння за важливістю, а також обчислений вектор пріоритетів для кожної системи.

Таблиця 4.4

Матриця порівняння систем для критерію «Зручність» в дереві цілей при системному аналізі об’єкту дослідження

№	Назва системи	Порівняння систем					Вектор пріоритету
		1	2	3	4	5	
1	Інформаційно-консультаційні	1 / 2	1 / 4	2	1 / 5	5	0,2
2	Інформаційно-керівні	4	1 / 2	5	1 / 4	3	0,5

3	Інформаційно-пошукові	1 / 5	1 / 6	1 / 5	1 / 4	7	0,4
4	Інтелектуальні інформаційні	1 / 7	3	5	2	4	0,38
5	СППР	3	1 / 3	1 / 3	1 / 4	4	0,17

Дана таблиця представляє матрицю порівняння різних систем за критерієм "Зручність" в контексті системного аналізу об'єкта дослідження. У таблиці вказані назви систем, їх порівняння щодо зручності, а також обчислений вектор пріоритетів для кожної системи.

Таблиця 4.5

Матриця порівняння систем для критерію «Коректність» дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

№	Назва системи	Порівняння систем					Вектор пріоритету
		1	2	3	4	5	
1	Інформаційно-консультаційні	1 / 3	2	3	3	7	0,6
2	Інформаційно-керівні	1 / 3	4	1 / 5	4	5	0,24
3	Інформаційно-пошукові	1 / 2	3	1 / 3	3	1 / 2	0,36
4	Інтелектуальні інформаційні	1 / 2	1	1 / 2	1 / 2	3	0,22
5	СППР	1 / 2	3	4	1 / 5	1 / 4	0,28

Ця таблиця представляє матрицю порівняння різних систем за критерієм "Коректність" в контексті створення програмного модулю шифрування даних у багатокористувацькій грі. У таблиці вказані назви систем, їх порівняння щодо коректності, а також обчислений вектор пріоритетів для кожної системи.

Таблиця 4.6

Матриця порівняння систем для критерію «Доступність» в дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

№	Назва системи	Порівняння систем					Вектор пріоритету
		1 / 2	3	4	3	5	

1	Інформаційно-консультаційні	3	1 / 4	½	5	1 / 3	0,24
2	Інформаційно-керівні	1 / 7	1 / 2	1 / 3	2	3	0,22
3	Інформаційно-пошукові	3	6	4	3	4	0,6
4	Інтелектуальні інформаційні	4	3	¼	3	1 / 3	0,6
5	СППР	3	2	1 / 3	1 / 4	1	0,49

У цій таблиці наведена матриця порівняння різних систем за критерієм "Доступність" у контексті створення програмного модулю шифрування даних у багатокористувацькій грі. Таблиця містить назви систем, їх порівняння щодо доступності, а також вектор пріоритетів для кожної системи.

Таблиця 4.7

Матриця порівняння систем для критерію «Якість» в дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

№	Назва системи	Порівняння систем					Вектор пріоритету
		2	3	3	6	3	
1	Інформаційно-консультаційні	3	7	9	1 / 5	2	0,47
2	Інформаційно-керівні	5	3	1 / 8	3	4	0,33
3	Інформаційно-пошукові	1 / 2	1 / 7	1 / 3	1 / 2	3	0,24
4	Інтелектуальні інформаційні	1 / 9	4	1 / 3	1 / 7	½	0,26
5	СППР	1 / 7	6	4	1 / 4	6	0,44

У даній таблиці представлена матриця порівняння різних систем за критерієм "Якість" у контексті створення програмного модулю шифрування даних у багатокористувацькій грі. Таблиця включає назви систем, їх порівняння щодо якості, а також вектор пріоритетів для кожної системи.

Наступним кроком при аналізі дерева цілей було створено таблицю, де зібрано усі коефіцієнти критеріїв кожного типу систем, та було визначено кінцеву пріоритетність інформаційної системи у таб. 4.8.

Таблиця 4.8

Результати методу аналітичної ієрархії в дереві цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

Критерій Тип систем	Актуальність	Точність	Зручність	Коректність	Доступність	Якість	Пріоритет
Інформаційно-консультаційні	0,015	0,034	0,165	0,018	0,086	0,2566	0,49
Інформаційно-керівні	0,15	0,064	0,044	0,029	0,018	0,085	0,28
Інформаційно-пошукові	0,17	0,029	0,023	0,024	0,023	0,036	0,12
Інтелектуальні інформаційні	0,032	0,035	0,042	0,026	0,022	0,033	0,19
СППР	0,033	0,019	0,011	0,028	0,019	0,045	0,29

Порівняння вказує на те, що найвище значення отримано в інформаційно-консультаційній системі. Цей варіант визнано найкращим за критичним показником успішності для подальшого використання у розробці програмного модулю шифрування даних у багатокористувацькій грі.

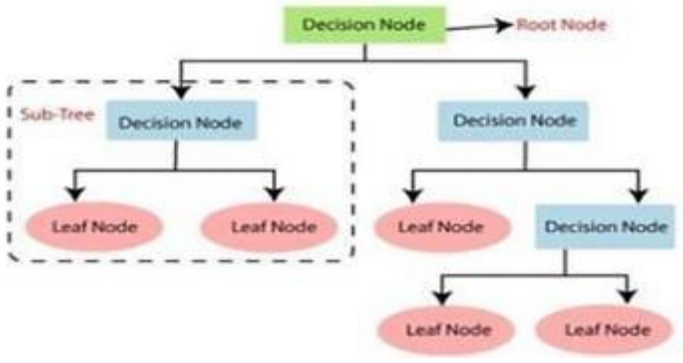


Рис. 4.2. Приклад роботи алгоритму дерева цілей при створенні програмного модулю шифрування даних у багатокористувацькій грі

Цей тип дерева рішень використовується для створення навчальної моделі, яка служить для передбачення класу або значення цільової змінної. Це досягається за допомогою вивчення простих правил прийняття рішень на

основі історичних даних (навчальних даних). У дереві рішень процес передбачення починається з кореня дерева, де значення атрибутів порівнюються з атрибутами запису. Відповідно до результату цього порівняння обирається гілка, яка веде до наступного вузла. Тип дерева рішень зазвичай визначається типом цільової змінної, і існують два основних типи дерев рішень [48].

При проєктуванні програмного модулю шифрування даних у багатокористувацькій грі рекомендується розпочати з побудови контекстної діаграми А–0. Ця діаграма надає стисле та зрозуміле пояснення кожного кроку реалізації, що допомагає оцінити правильність обраного шляху виконання. Основними компонентами цієї діаграми є вхідні дані, вихідні дані, управління та механізми. Зазвичай вказується точка зору (виконавця робіт), предметна область та ціль виконання.

Після проведеного аналізу щодо основних елементів контекстної діаграми "Програмного модулю шифрування даних у багатокористувацькій грі", був сформований наступний перелік даних:

– вхідні дані: завдання на створення гри;

– вихідні дані: реліз apk-версії готової гри;

– управління: вимоги до гри, обмеження ПЗ, планування розробки;

– механізми: команда проєкту, програмні засоби, апаратне забезпечення.

Контекстна діаграма А–0, що була розроблена за допомогою програмного забезпечення Business Process Design Tool на основі цих даних представлена на рис. 4.3.

Рис. 4.3. Контекстна діаграма А-О, що описує черговість дій при створенні програмного модулю шифрування даних у багатокористувацькій грі

Для кращого розуміння та керування процесом розробки програмного модулю шифрування даних у багатокористувацькій грі, діаграма А–0, яка містить загальну інформацію, була поділена на три підрівні. Ця декомпозиція полягає у розподілі складного завдання на менші складові частини або елементи.

Перший етап розробки включає в себе реалізацію ігрової механіки, таку як створення ігрової зони, комірок, пошук можливих ходів і методу автоматичного захоплення. На другому етапі проводиться імітація гри штучним інтелектом на п'яти різних рівнях за допомогою скриптів. Третій етап включає в себе тестування програмного модулю під час гри та відладку у випадку виявлення помилок.

Кожен етап має свої вхідні та вихідні дані, управління, та механізми. Наприклад, на першому етапі вхідними дані є завдання на створення програмного модулю, вихідними - файли з налаштованою механікою гри та моделлю імітації штучного інтелекту. Управління включає в себе вимоги до програми, обмеження програмного забезпечення та планування розробки, а механізми цього етапу включають команду проєкту, програмні засоби та апаратне забезпечення.

Такий підрозділ діаграми A–0 на три етапи допомагає краще розуміти та керувати процесом розробки програмного модулю.

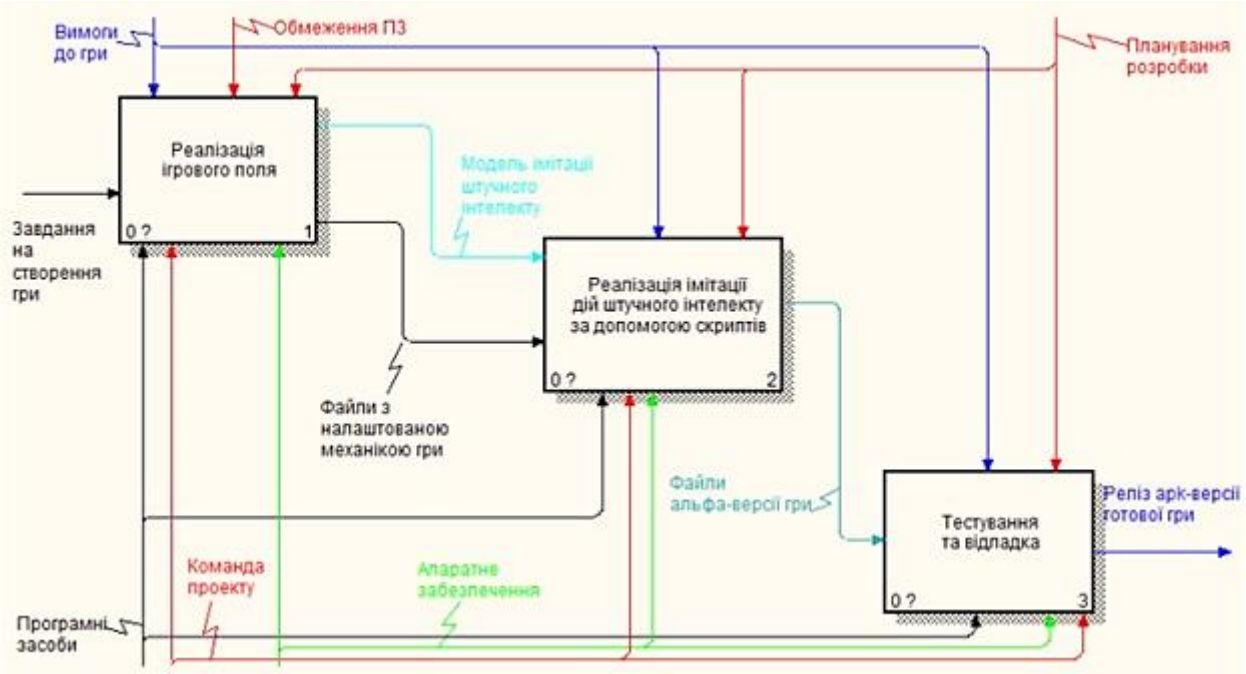


Рис. 4.4. Декомпозиція контекстної діаграми A–0

Після розбиття контекстної діаграми на три процеси, кожен із них було поділено на окремі роботи, при цьому враховуючи, що результати попередніх робіт стають вхідними даними для наступних.

Процес "Реалізація ігрового поля" розкладено на п'ять підрівнів, які включають наступне:

1. Генерація та реалізація ігрової зони, де створюється основний ігровий простір на основі ігрової механіки.

2. Генерація та реалізація ігрових точок, які представляють ігрові комірки та їх можливість захоплення на вже створеній ігровій зоні.

3. Реалізація методу пошуку можливих ходів, який допомагає гравцям та обчисленням штучного інтелекту шукати можливі ходи на ігровому полі.

4. Реалізація методу автоматичного захоплення ігрових комірок, який дозволяє автоматично захоплювати комірки на ігровому полі.

5. Реалізація методів режиму гри "Тактика", де існують перешкоди на ігровому полі.

Кожен етап має свої вхідні та вихідні дані, управління та механізми. Наприклад, на першому етапі вхідними дані є завдання на створення гри та програмного модулю шифрування даних, вихідними - файли ігрової зони. Управління включає в себе вимоги до гри, обмеження програмного забезпечення та планування розробки, а механізми цього етапу включають команду проєкту, програмні засоби та апаратне забезпечення.

Аналогічно, інші етапи розкладаються на підрівні, де кожен має свої вхідні та вихідні дані, управління та механізми, які враховують результати попередніх робіт для подальшого розвитку програмного модулю.

Вигляд процесу «Реалізація ігрового поля» після декомпозиції на 5 етапів наведений на рис. 4.5.

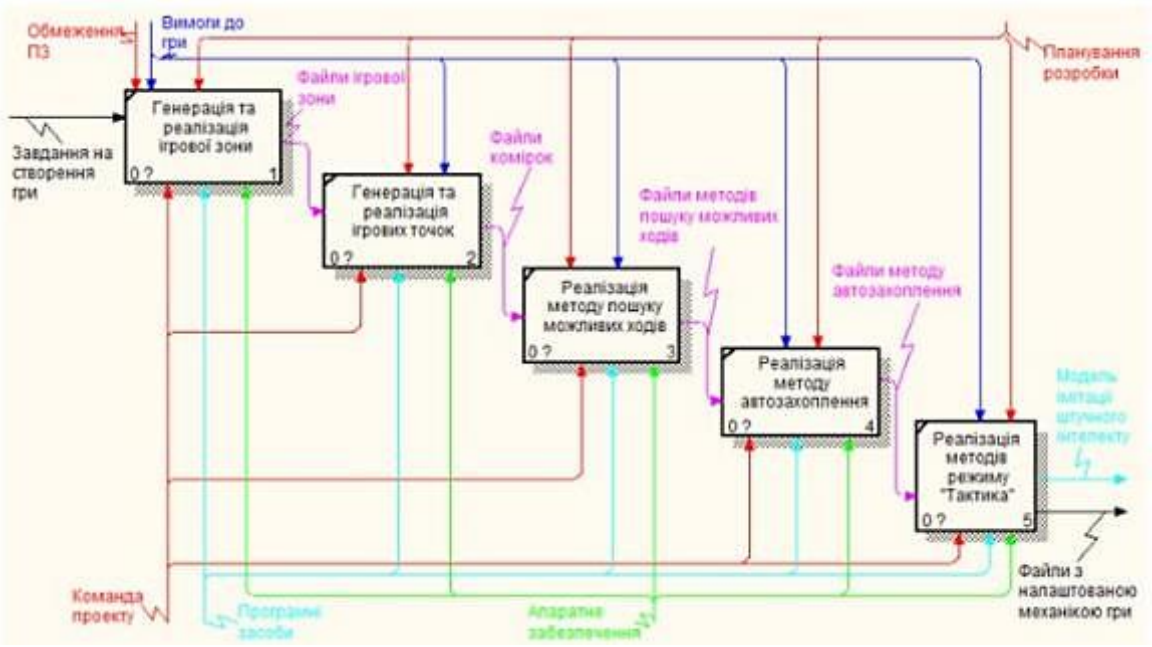


Рис. 4.5. Декомпозиція процесу «Реалізація ігрового поля»

Процес "Реалізація імітації дій штучного інтелекту за допомогою скриптів" був поділений на два підрівні, які включають такі етапи:

1. Реалізація рівнів складності: На цьому етапі реалізуються п'ять рівнів імітації діяльності штучного інтелекту. Вхідними даніми для цього етапу є

файли з налаштованою механікою гри та моделью імітації штучного інтелекту. Механізми включають команду проєкту, програмні засоби та апаратне забезпечення.

2. Реалізація збірки штучного інтелекту та реалізація сцени для вибору одного із рівнів складності: На цьому етапі виконується збірка всіх рівнів складності та створюється сцена для вибору одного із них. Вхідними даніми є файли простого, середнього, легкого, важкого та дуже важкого рівнів складності. Управління включає в себе планування розробки програмного модулю шифрування даних.

Ця декомпозиція дозволяє краще керувати реалізацією імітації дій штучного інтелекту та забезпечує взаємозв'язок між рівнями складності та їх збірку та вибір (Рис. 4.6).

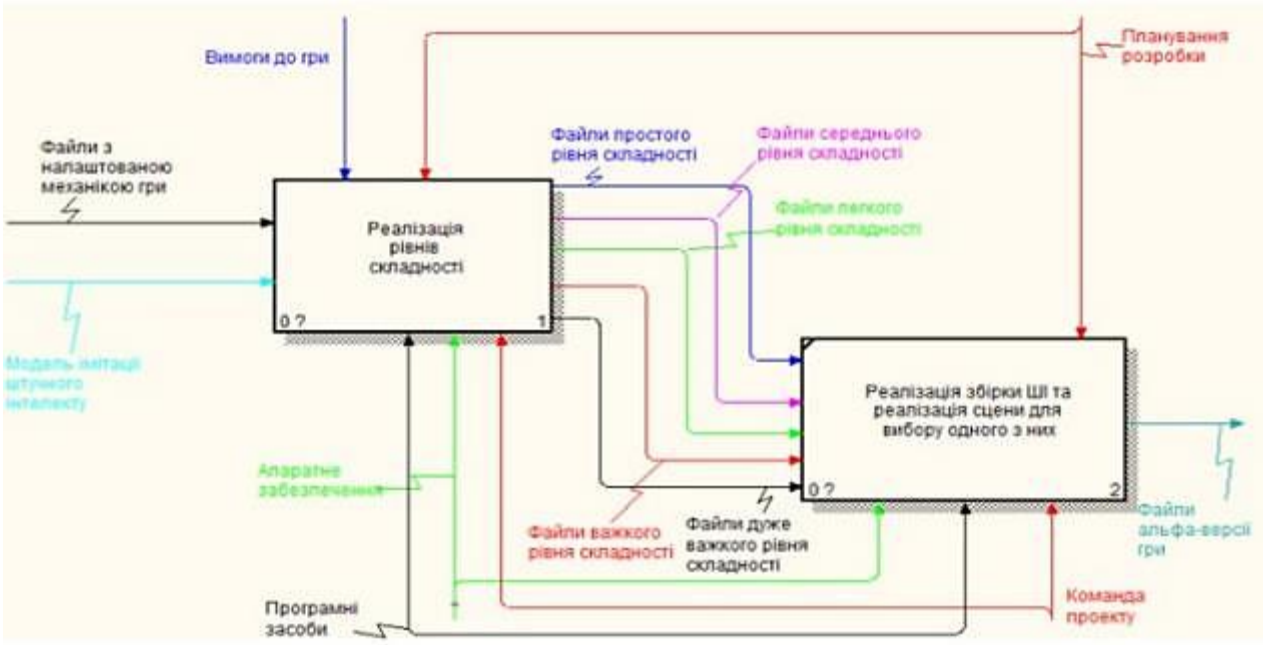


Рис. 4.6. Декомпозиція процесу «Реалізація штучного інтелекту»

Процес "Реалізація рівнів складності" був поділений на п'ять підрівнів, кожен з яких включає реалізацію певного рівня складності для імітації діяльності штучного інтелекту. Ця декомпозиція обумовлена наступними етапами:

1. Реалізація простого рівня складності: На цьому етапі розробляється імітація дій штучного інтелекту на простому рівні складності. Вхідними даніми є файли з налаштованою механікою гри, а вихідними - файли простого рівня складності.

2. Реалізація легкого рівня складності: На цьому етапі створюється імітація дій на легкому рівні складності, використовуючи напрацювання з простого рівня. Вхідні дані включають файли з налаштованою механікою гри та файли простого рівня складності, а вихідні - файли легкого рівня складності.

3. Реалізація середнього рівня складності: На цьому етапі створюється імітація дій на середньому рівні складності, використовуючи напрацювання з попередніх рівнів і модель імітації штучного інтелекту. Вхідні дані включають файли з налаштованою механікою гри, файли легкого рівня складності та модель імітації штучного інтелекту.

4. Реалізація важкого рівня складності: На цьому етапі створюється імітація дій на важкому рівні складності, модернізуючи напрацювання з середнього рівня. У вхідних даних вказані файли з налаштованою механікою гри, файли середнього рівня складності та модель імітації штучного інтелекту.

5. Реалізація дуже важкого рівня складності: На цьому етапі створюється імітація дій на дуже важкому рівні складності, модернізуючи напрацювання з важкого рівня. У вхідних даних вказані файли з налаштованою механікою гри, файли важкого рівня складності та модель імітації штучного інтелекту.

Кожен етап включає в себе власні вхідні та вихідні дані, а також механізми та управління, необхідні для реалізації конкретного рівня складності імітації дій штучного інтелекту.

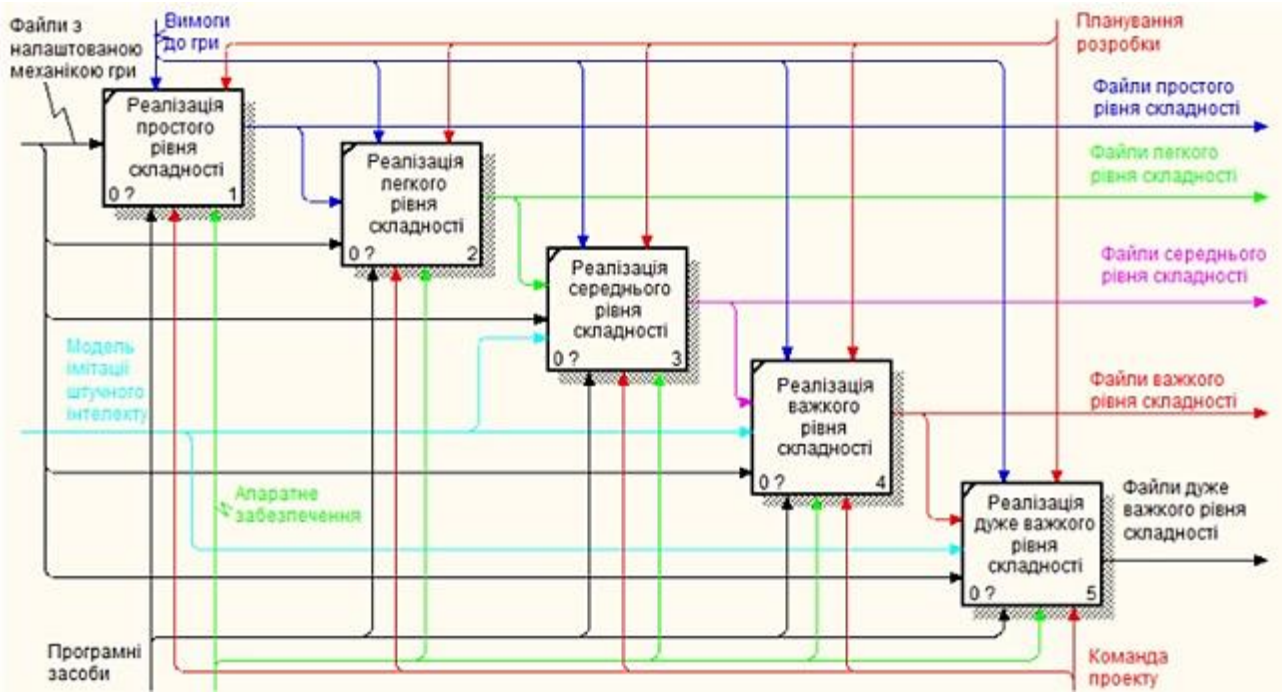


Рис. 4.7. Декомпозиція процесу «Реалізація рівнів складності програмного модулю шифрування даних»

Процес "Тестування та відладка" був розбитий на два підрівні, кожен з яких включає в себе власні етапи:

1. Перший етап: Пошук помилок у програмному модулю шифрування даних. На цьому етапі проводиться тестування програмного модулю з використанням альфа-версії гри. Вхідні дані для цього етапу включають файли альфа-версії гри. Управління включає в себе вимоги до гри та планування розробки. Механізми, які використовуються на цьому етапі, включають команду проєкту, програмні засоби та апаратне забезпечення.

2. Другий етап: Виправлення знайдених помилок, якщо вони виявляються під час тестування. На цьому етапі виявлені помилки у програмному модулі шифрування даних виправляються. Управління на цьому етапі включає в себе планування розробки. Механізми, які використовуються для виправлення помилок, включають команду проєкту, програмні засоби та апаратне забезпечення.

Ця декомпозиція дозволяє систематично провести тестування програмного модулю і, в разі потреби, виправити знайдені помилки для забезпечення якості та надійності програмного продукту.

Вигляд процесу «Тестування та відладка» після декомпозиції на 2 етапи наведений на рис. 4.8.

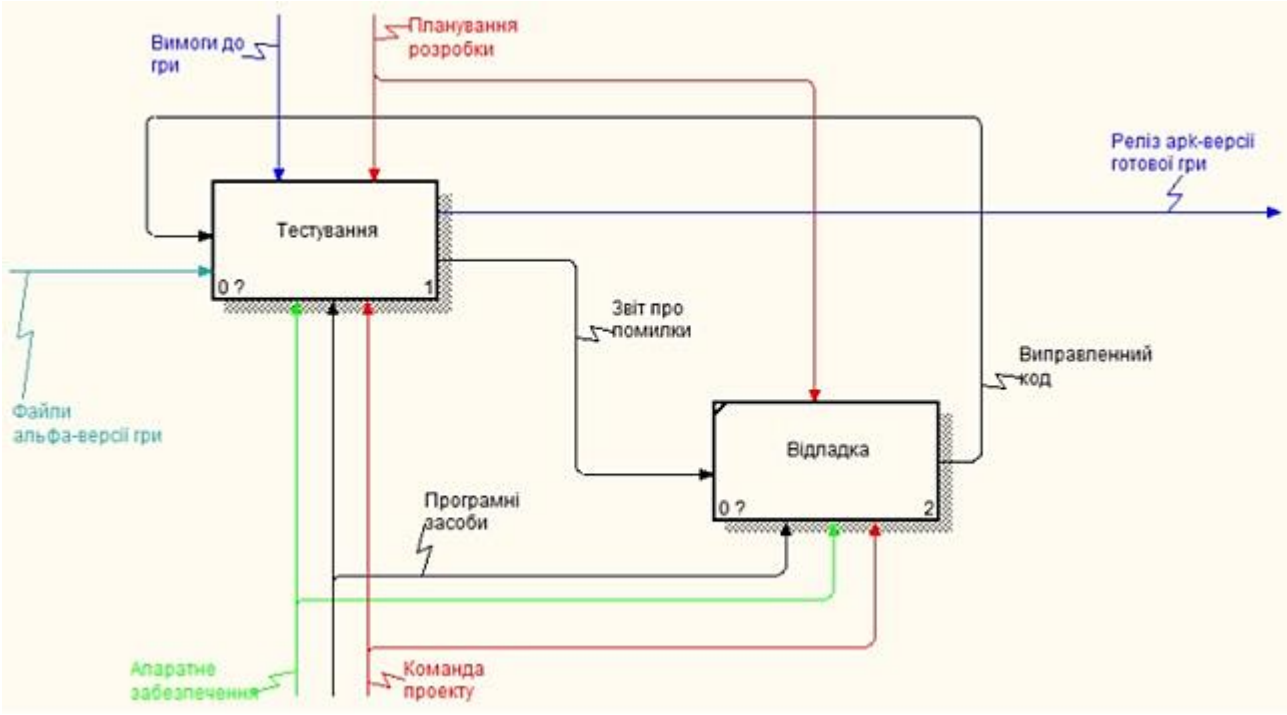


Рис. 4.8. Декомпозиція процесу «Тестування та відладка»

Процес "Тестування" був декомпозований на три підрівні, кожен з яких має свої характеристики:

1. Перший етап: Тестування функціоналу продукту. На цьому етапі перевіряється правильність роботи функціоналу гри. Вхідні дані для цього етапу включають виправлений код та файли альфа-версії гри. Механізми, що використовуються на цьому етапі, включають команду проєкту, програмні засоби та апаратне забезпечення.

2. Другий етап: Тестування режиму гри на двох. На цьому етапі перевіряється правильність роботи режиму гри на двох гравців. Вхідні дані для цього етапу включають відмітку у тест-кейсі про проходження тестування.

Механізми, які використовуються, включають команду проєкту, програмні засоби та апаратне забезпечення.

3. Третій етап: Тестування одиночного режиму гри. На цьому етапі перевіряється правильність роботи одиночного режиму гри. Вхідні дані включають в себе відмітку у тест-кейсі про проходження тестування програмного модулю шифрування даних у багатокористувацькій грі. Механізми, що використовуються на цьому етапі, включають команду проєкту, програмні засоби та апаратне забезпечення.

Ця декомпозиція допомагає систематично провести тестування програмного продукту в різних режимах гри та впевнитися в правильності його роботи перед релізом. Вигляд процесу «Тестування» після декомпозиції на 3 етапи наведений на рис. 4.9.

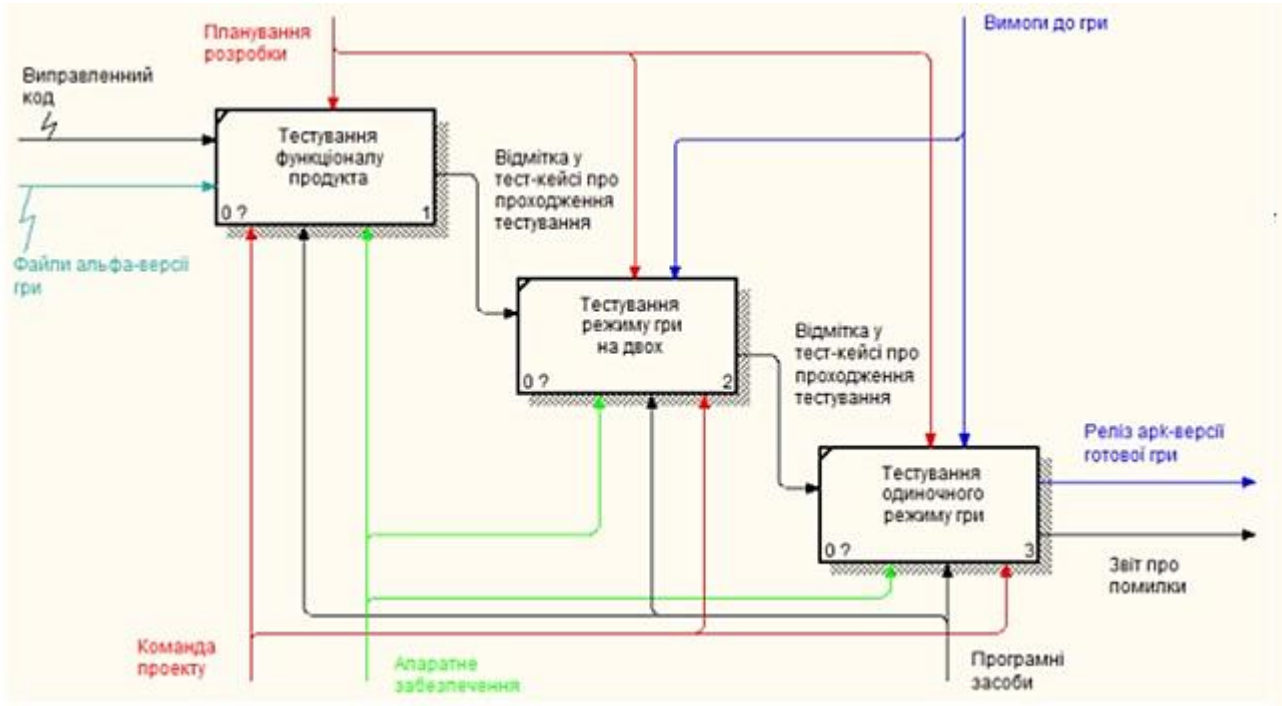


Рис. 4.9. Декомпозиція процесу «Тестування програмного модулю шифрування даних у багатокористувацькій грі»

Дослідження архітектури програмного модулю шифрування даних у багатокористувацькій грі має на меті визначити оптимальний спосіб розбиття системи на складові частини, визначити, як ці частини взаємодіють між собою,

передають інформацію та як кожна з них розвивається окремо. Ця архітектура повинна відповідати вимогам до системи, керуючись принципом "форма відповідає функції". Процес проєктування архітектури відбувається після етапу аналізу і формулювання вимог і передбачає перетворення вимог до модулю у вимоги до програмної системи, а також побудову на їхній основі структури системи.

Побудова архітектури програмного модулю шифрування даних у багатокористувацькій грі включає такі етапи:

1. Визначення цілей системи.

2. Визначення вхідних і вихідних даних системи.

3. Декомпозиція системи на підсистеми, компоненти або модулі.

4. Розроблення загальної структури системи.

Для побудови архітектури можуть використовуватися різні підходи і інструменти. Наприклад, архітектурний патерн Flux, розроблений Facebook, використовується для створення додатків, особливо у випадках, коли необхідно ефективно управляти станом програми. Основними компонентами архітектури Flux є:

1. Дія (Action): Це подія, яка виникає при взаємодії користувача з інтерфейсом і передається від представлення.

2. Відправник (Dispatcher): Відправник зберігає контекст і передає дії з представлення до сховища даних.

3. Сховище (Store): Сховище містить дані і реагує на події оновлення даних від відправника.

4. Представлення (View): Представлення реагує на зміни даних і відображає їх користувачеві.

Архітектурний патерн Flux допомагає створити систему, яка ефективно керує станом програми та забезпечує взаємодію компонентів для досягнення потрібної функціональності.

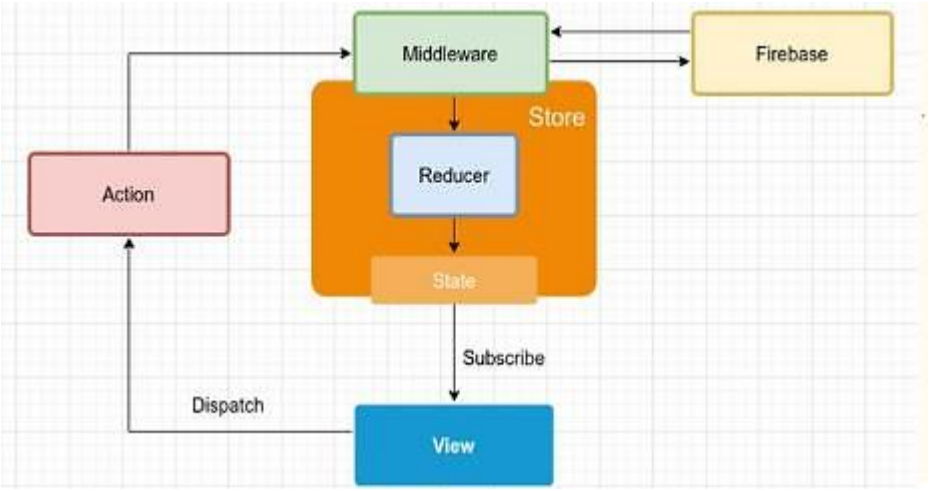


Рис. 4.10. Архітектура проєкту реалізація Flux-архітектури в вигляді Redux

Redux є передбачуваним контейнером стану для додатків JavaScript, який допомагає керувати станом програми і забезпечує передбачувану модель оновлення стану. Основні принципи Redux включають:

1. Централізований стан. У Redux, стан програми зберігається в одному об'єкті JavaScript, що спрощує відображення та передачу даних у всій програмі. Це також полегшує процес тестування і налагодження.

2. Незмінний стан. У Redux, стани не можуть бути змінені безпосередньо. Зміни стану можливі лише за допомогою відправки дій (actions). Дії є незмінними об'єктами JavaScript, які описують зміни стану. Цей підхід сприяє передбачуваній та контрольованій моделі зміни стану.

3. Редуктори вказують, як дія перетворює поточний стан в новий стан. Редуктори є чистими функціями без побічних ефектів, і вони приймають на вхід поточний стан і дію, щоб створити новий стан. Це централізовані функції, які управляють мутаціями даних і можуть впливати на всю або частину стану.

Завдяки цим принципам, Redux надає передбачувану модель управління станом програми. Зі станом і дією, наступний стан програми можна передбачити з впевненістю, що спрощує розробку, налагодження і тестування додатків JavaScript.

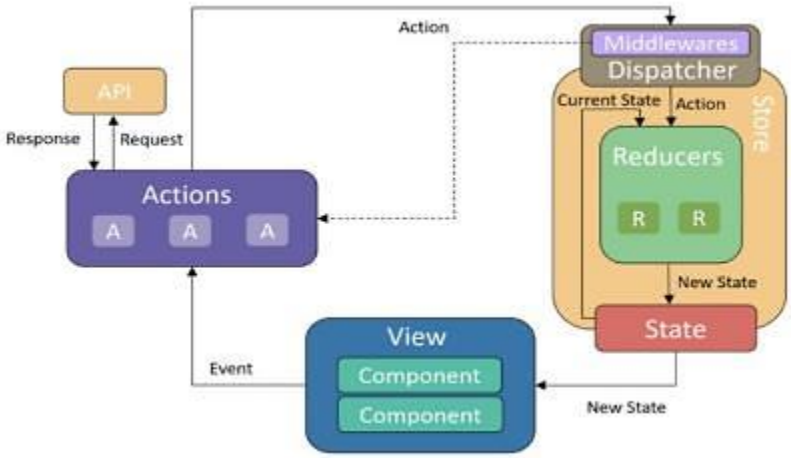


Рис. 4.11. Процес зміни стану системи з Redux

Отже, використання архітектурного підходу Redux має декілька переваг у вашому випадку:

1. Передбачувані оновлення стану: Redux надає передбачувану модель оновлення стану, що полегшує розуміння того, як працює потік даних у вашому додатку. Це робить код більш зрозумілим і підтримуваним.

2. Використання «чистих» редукторних функцій: Redux вимагає, щоб редуктори були "чистими" функціями без побічних ефектів. Це полегшує тестування логіки і робить код більш надійним. Також це дає можливість використовувати корисні функції, такі як "подорож у часі" (time-travel debugging), що допомагає відстежувати стан програми на різних етапах.

3. Централізація стану: Redux централізує стан програми в одному об'єкті, що спрощує реалізацію подій, таких як внесення змін до даних або збереження даних між оновленнями сторінок. Це дозволяє легко взаємодіяти зі станом та виконувати операції з ним.

4. Підтримка для React: Redux ідеально поєднується з бібліотекою React, що робить його відмінним вибором для розробки веб-додатків на React. Він надає інтеграцію з React і допомагає зберігати стан додатку та взаємодіяти з компонентами React.

Отже, на основі цих переваг обрання архітектурного підходу Redux для розробки програмного модулю шифрування даних у багатокористувацькій грі є логічним та обґрунтованим рішенням. Він сприяє покращенню структури, надійності і продуктивності вашого додатку.

4.2. Розробка структури бази даних та use case діаграми

Firebase Realtime Database є потужним інструментом для розробки спільних додатків, і він має кілька ключових переваг:

1. Безпечний доступ до бази даних: Firebase Realtime Database надає безпечний доступ до бази даних безпосередньо з клієнтського коду. Це означає, що можна налаштувати правила доступу, які контролюють, хто і як може звертатися до вашої бази даних. Це дозволяє вам забезпечити конфіденційність та безпеку даних.

2. Локальне зберігання даних: Дані зберігаються локально на пристрої, що дозволяє працювати з ними навіть в автономному режимі. Це особливо корисно для додатків, які мають працювати без доступу до Інтернету.

3. Реальний час: Firebase Realtime Database надає можливість взаємодіяти з даними в режимі реального часу. Це означає, що події та зміни в базі даних відображаються миттєво, що дозволяє створювати додатки з живою взаємодією та сповіщеннями.

4. Синхронізація при відновленні з'єднання: Коли пристрій відновлює з'єднання з Інтернетом, Firebase Realtime Database автоматично синхронізує локальні зміни з віддаленими оновленнями. Це допомагає уникнути втрати даних та конфліктів при відновленні з'єднання.

5. Гнучкі правила доступу* Firebase Realtime Database надає гнучкі правила доступу на основі виразів. Можна точно визначити, які дані можуть

бути прочитані або записані і хто має до них доступ. Це дозволяє забезпечити дотримання прав доступу та безпеки даних.

6. Інтеграція з перевіркою справжності Firebase: можна інтегрувати Firebase Realtime Database з перевіркою справжності Firebase для більш точного контролю доступу до даних. Можна визначити, хто має доступ до яких даних і як вони можуть отримати до них доступ.

Усі ці переваги роблять Firebase Realtime Database потужним інструментом для створення спільних та взаємодіючих додатків з надійним доступом до даних і можливістю реагувати на зміни в режимі реального часу.

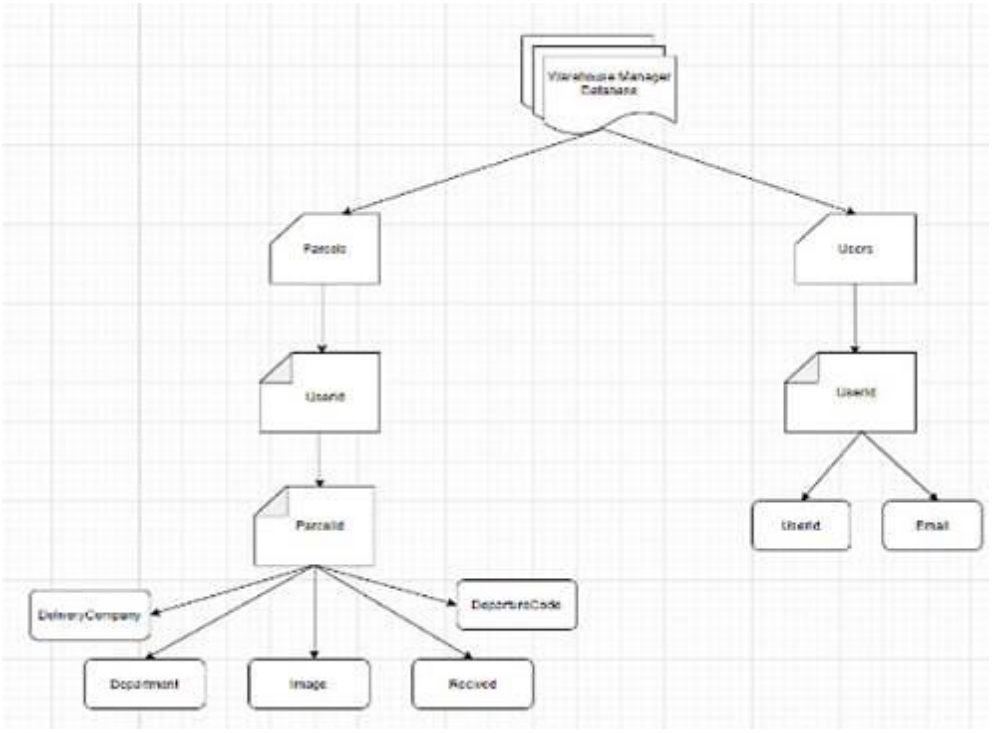


Рис. 4.12. База даних платформа Firebase Realtime Database

Опишемо переваги використання бази даних NoSQL та важливості використання Redux для створення програмного модулю шифрування даних у багатокористувацькій грі. Розглянемо основні переваги обох цих підходів:

Переваги бази даних NoSQL:

1. Ефективність розробки: Використання бази даних NoSQL дозволяє спростити взаємодію з даними і зменшити кількість коду, що потрібно написати, налагодити і розвивати. Це робить процес розробки більш продуктивним.

2. Великомасштабні дані: Бази даних NoSQL спрямовані на роботу з великими обсягами даних і можуть бути легко масштабовані на кластерах серверів. Це дозволяє обробляти великі обсяги даних ефективно.

3. Гнучкість структури даних: NoSQL дозволяє зберігати дані без фіксованої схеми, що дає більшу гнучкість у роботі з даними, особливо коли структура даних змінюється з часом.

Переваги використання Redux:

1. Централізована структура стану: Redux забезпечує централізовану структуру стану програми, що полегшує розуміння та управління станом. Весь стан зберігається в одному об'єкті, що спрощує відображення та передачу даних.

2. Реалізація Flux-підходу: Redux базується на архітектурі Flux, яка полегшує управління потоком даних у додатку. Він допомагає уникнути складних зв'язків між компонентами та забезпечити передбачувані оновлення стану.

3. Незмінність стану: В Redux стан є незмінним, тобто його не можна змінити безпосередньо. Зміни в стані відбуваються за допомогою дій (actions), що дозволяє контролювати зміни та створювати реактивні додатки.

4. Ефективне керування станом: За допомогою Redux можна ефективно керувати станом програми, що сприяє покращенню продуктивності та зручності тестування.

Отже, використання бази даних NoSQL дозволяє ефективно зберігати та обробляти дані великих обсягів, а використання Redux спрощує управління станом програми і забезпечує передбачуваність реакції на зміни. Обидва ці

підходи можуть бути важливими для розробки програмного модулю шифрування даних у багатокористувацькій грі.

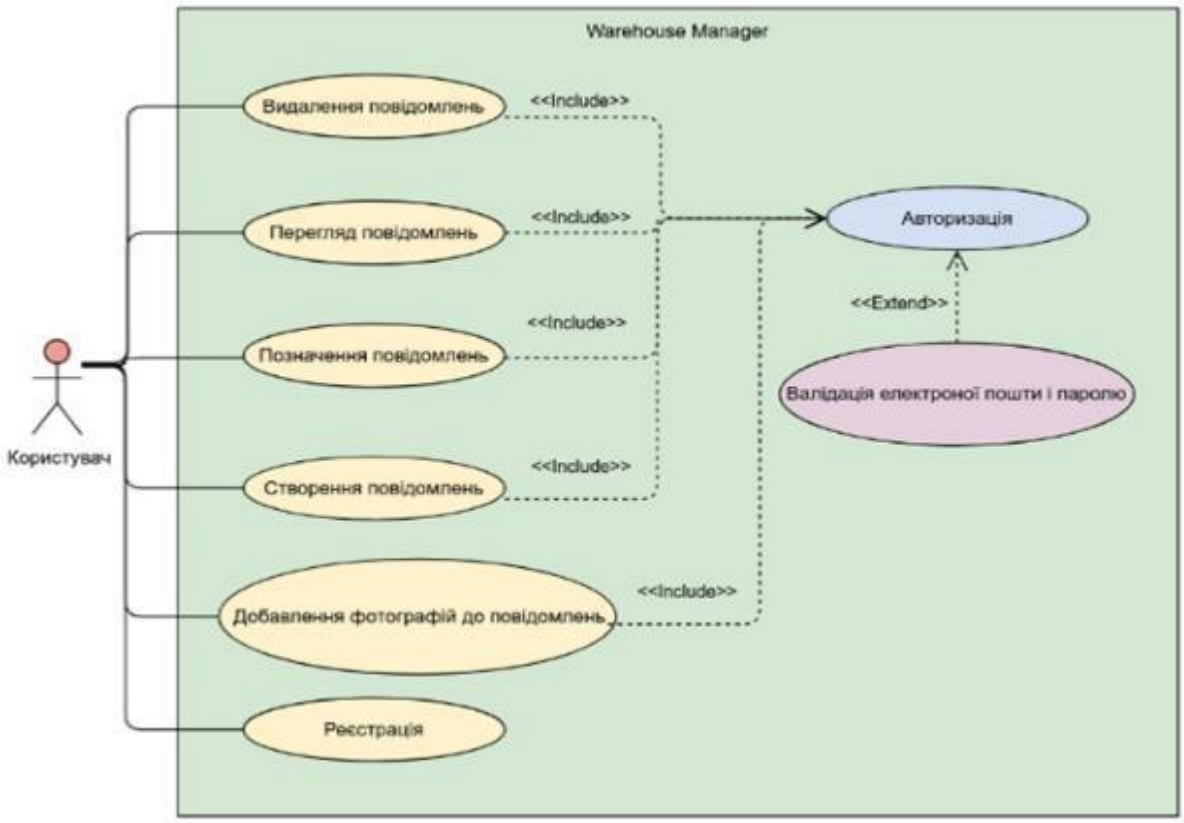


Рис. 4.13. Зовнішній вигляд use case діаграми

Use case діаграми використовуються для обліку функціональних вимог програмного модулю шифрування даних у багатокористувацькій грі. Після визначення вищевказаних пунктів ми повинні використовувати такі рекомендації для побудови діаграми ефективного використання:

– Ім’я прецеденту дуже важливо – виберіть його таким чином, щоб воно могло ідентифікувати їх функції;

– Необхідно дати відповідне ім’я для акторів;

– Показати на діаграмі відносини і залежності;

– Не потрібно включати всі типи відносин, оскільки основна мета діаграми полягає у визначенні вимог;

– При необхідності можна використовувати пояснення, щоб прояснити деякі важливі моменти.

Таким чином, use case діаграми використовуються для збору вимог до системи, включаючи внутрішні і зовнішні впливи (як правило, це вимоги до дизайну). Отже, коли система аналізується для збору її функціональних можливостей, розробляються приклади використання і ідентифікуються учасники.

Приклад use case діаграми – графічне зображення взаємодій між елементами програмного модулю шифрування даних у багатокористувацькій грі. Це методологія, яка використовується в системному аналізі для виявлення, уточнення і організації системних вимог. У цьому контексті термін «система» відноситься до того, що розробляється або експлуатується, наприклад до веб-сайту з продажу та обслуговування товарів поштою. Кроки використання більш високого рівня розглядаються як цілі для більш низького рівня. Використання діаграми варіантів use case diagram в середовищі продажів включає в себе упорядкування товарів, оновлення каталогу, обробку платежів і відносини з клієнтами. Діаграма використання виглядає як блок-схема. Інтуїтивні символи представляють собою елементи системи.

Діаграма може підсумувати відомості про користувачів (також відомих як суб’єкти) та їх взаємодію з системою. Щоб побудувати один об’єкт, буде використовуватися набір спеціалізованих символів і конекторів.

Для того щоб користувач зробив будь яку дію крім реєстрації йому обов’язково потрібно авторизуватись. Авторизація також включає в себе валідацію даних.

Use case діаграма не має великого значення при відсутності чіткого розуміння процесу – вона не буде моделювати порядок виконання кроків, якщо не закладено чіткий алгоритм. Експерти рекомендують використовувати дані діаграми для доповнення текстового варіанта. У діаграмі на високому рівні демонструється взаємозв’язок між варіантами використання, суб’єктами і системами.

Аналізуючи структуру програмного модулю шифрування даних у багатокористувацькій грі, можна виділити, що він складається із двох колекцій: "Дані" та "Користувачі". Основні особливості цього модулю включають:

1. Колекція "Дані " та "Користувачі" містять документи із унікальними ідентифікаторами.
2. Кількість документів в базі даних не обмежена.
3. Документи користувачів мають ідентифікатор користувача, який не повторюється.
4. Документи посилок містять інформацію про стан даних.
5. У документі "Користувачі" присутні два поля: "UserId" та "Email".
6. Дані зберігаються у форматі JSON та оновлюються в режимі реального часу за допомогою механізму push / sub.
7. Доступ до даних можливий як на клієнтській стороні (веб-браузер, Android / iOS додаток), так і через REST API з можливістю контролю доступу до конфіденційних даних.

У програмній та системній інженерії діаграма використання (use case) представляє собою список дій або кроків, які описують взаємодію між акторами (учасниками) та системою з метою досягнення певної цілі. Акторами можуть бути як люди, так і зовнішні системи. Діаграма використання допомагає визначити функціональні вимоги та взаємодію учасників системи.

Характеристики методики включають:

1. Організацію функціональних вимог.
2. Моделювання цілей взаємодії користувачів системи.
3. Запис сценаріїв відпрацювання подій для досягнення цілей.
4. Опис основного та виняткового ходу подій.
5. Управління доступом до функцій інших подій.

Діаграма використання є важливою частиною розробки програмного модулю шифрування даних у багатокористувацькій грі, оскільки вона

допомагає розробникам, замовникам і користувачам спілкуватися і розуміти функціональні можливості системи.

4.3. Опис цільової аудиторії та функціональність об’єкту

Аудиторія, до якої спрямований створений додаток, включає дітей віком старше 12 років, які перебувають під наглядом дорослих, а також дорослих віком від 18 років і старше, які виявляють інтерес до даного розробленого продукту. Важливо зазначити, що даний додаток є безкоштовним, тому ним можуть користуватись люди з різними ігровими інтересами.

Функціональні можливості створеного програмного модулю шифрування даних у багатокористувацькій грі включають:

1. Шифрування даних в грі.
2. Зберігання інформації про обрані типи персонажів у грі, бонуси та штрафи.
3. Допомога гравцю у введенні результатів дій суперників та оцінка позиції гравців перед закінченням гри.

Важливим аспектом є наявність інформації про розробника, що дозволяє користувачам знати особу, відповідальну за продукт. Це сприяє підвищенню довіри користувачів, покращує впізнаваність розробки та надає можливість отримати підтримку в разі потреби.

Для програмного модулю шифрування даних є можливість синхронізації з іншими засобами комунікації, такими як ноутбук, стаціонарний комп’ютер, планшет. Це полегшує роботу гравців та дозволяє їм легко оновлювати інформацію.

Додаток також підтримує push-сповіщення для зручної комунікації з гравцями та інтеграцію із соціальними мережами для швидкої реєстрації та

авторизації, обміну інформацією з друзями та створення галереї, яка підвищує залученість та довіру користувачів.

Усі ці функції спрямовані на поліпшення взаємодії гравців та забезпечення їм зручності під час гри.

4.4. Загальні відомості проєкту та реалізація

Алгоритм роботи програмного модулю шифрування даних у багатокористувацькій грі є простим і включає наступні кроки:

1. Користувач взаємодіє з базами записника і планувальника, вводячи дані.

2. Введені дані, при умові коректного вводу, зберігаються в базі даних.

3. Користувач має можливість обрати опцію з переліку доступних:

   – Операція над окремим елементом.

   – Маніпуляція всією базою.

   – Вибір опції виходу.

4. Після вибору опції, виконуються відповідні дії.

5. Графічне відображення алгоритму надає користувачу можливість візуально спостерігати за виконанням кожного кроку в грі.

Цей алгоритм спрощує взаємодію користувача з програмним модулем і забезпечує зручність у введенні та обробці даних у багатокористувацькій грі. (рис. 4.14):

Рис. 4.14. Загальний алгоритм роботи програмного модулю у багатокористувацькій грі

У цій роботі використовуються програмовані користувачем вентильні матриці або FPGA. Весь дизайн інтерфейсу програмного модулю виконаний в FPGA, і для цього використовується програмний пакет Xilinx з родини Virtex. Результат будівництва схеми наведено в додатку до цієї роботи.

На другому етапі розробки необхідно промоделювати отриману схему, щоб переконатися в правильності її функціонування. Для цього інтерпретуємо отриману схему, запускаємо логічне моделювання схеми за допомогою засобів Xilinx і виводимо всі основні вхідні та вихідні сигнали.

Цей підхід дозволяє забезпечити перевірку правильності роботи схеми та впевнитися в її коректності перед впровадженням у практичну діяльність. Результат моделювання наведено на рис. 4.15.

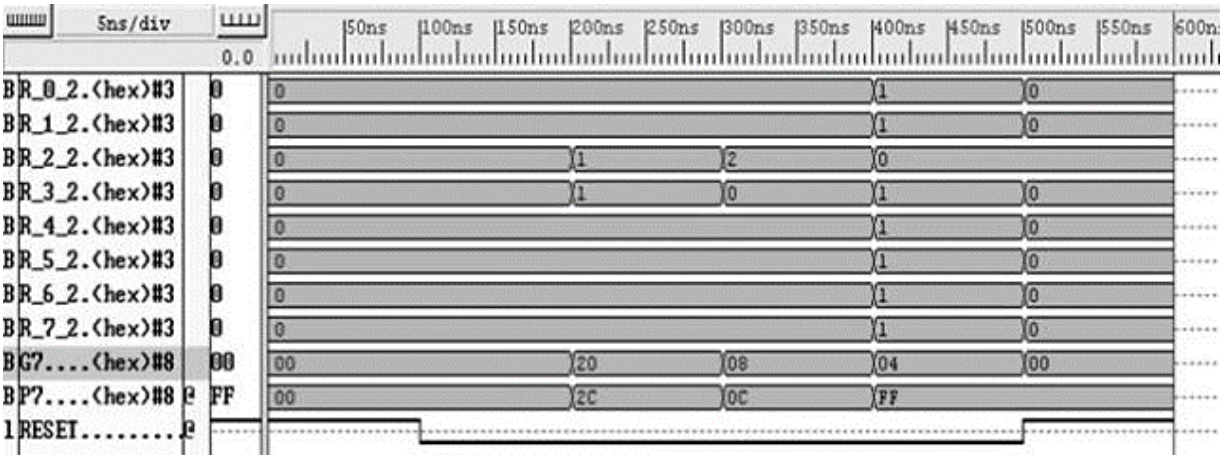


Рис. 4.15. Тимчасова діаграма роботи в циклі запису інформації

На діаграмі (рис. 4.15 і 4.16) представлені тести для перевірки роботи схеми читання і запису. Після успішної перевірки функціонування розроблена схема впроваджується в програмовану логічну матрицю (ПЛІС). У відміну від звичайних цифрових мікросхем, ПЛІС програмується для визначення її логіки роботи, а не виготовляється заздалегідь.

Для прошивки ПЛІС використовуються налагоджувальні середовища та програматори. Це дозволяє задати необхідну структуру цифрового пристрою у вигляді програми, написаної на спеціальних мовах опису апаратури, таких як Verilog, VHDL, AHDL, або у формі принципової електричної схеми.

Оскільки ПЛІС є перепрограмованим пристроєм, це надає можливість швидко модернізувати та додавати до нього додаткові функціональні вузли. Наприклад, вузли опрацювання цифрових підписів або алгоритми опрацювання елементів розширених та простих полів Галуа для захисту інформації від квантових комп'ютерів.

Переваги ПЛІС включають високий попит, масове виробництво, низьку вартість, універсальність, високу швидкодію та надійність. Вони також надають різноманітність у виборі напруги живлення та параметрів сигналів введення / виведення, а також підтримку зручних програмних засобів автоматизованого проєктування. Для цієї роботи було вибрано ПЛІС з сімейства Xilinx, зокрема, тип Xilinx Zynq-7000 AP SoC XC7Z020-CLG484. Результат занурення представлений на рис. 4.16.

Рис. 4.16. Тимчасова діаграма роботи програмного модулю шифрування даних в циклі читання інформації

Результатом проєктування ПЛІС є файл конфігурації, який містить дані про внутрішню схему ПЛІС, спроєктовану користувачем, в даному випадку - схему процесора. Інформація з цього файлу записується до спеціалізованої мікросхеми постійної пам'яті (ПЗП). Після подачі живлення ПЗП переписує цю інформацію до ПЛІС, і після цього ПЛІС починає працювати за цією схемою. При вимкненні живлення опис схеми у ПЛІС зникає.

Процес створення файлу конфігурації включає в себе наступні кроки:

1. Опис апаратної частини проєкту, що може бути в схемах або С-описах.
2. Функціональне моделювання опису апаратної частини.
3. Створення програмного забезпечення для універсального мікроконтролера.
4. Моделювання програмного забезпечення.
5. Комплексне функціональне моделювання опису апаратної частини та програмного забезпечення мікроконтролера.
6. Проєктування топології кристалу ПЛІС.

7. Уточнене логічне моделювання опису апаратної частини та програмного забезпечення мікроконтролера.

8. Визначення технічних характеристик ПЛІС, таких як максимальна робоча частота, споживана потужність, максимальна температура корпусу, час виконання криптографічних перетворень тощо.

9. Створення файлу конфігурації.

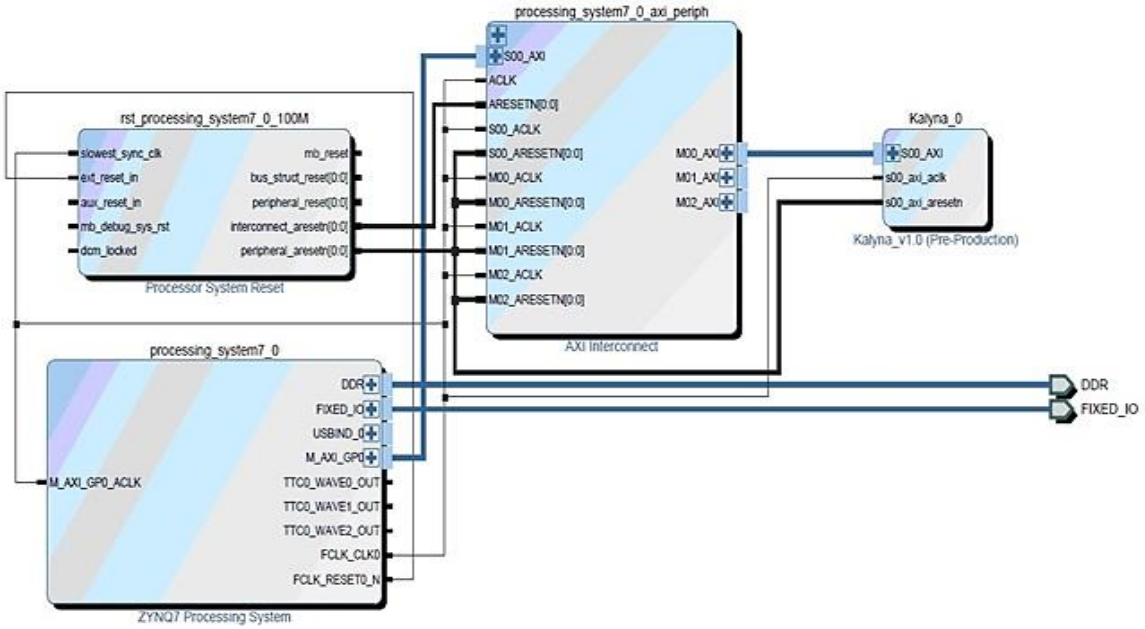


Рис. 4.17. Функціональна схема сконфігурованої ПЛІС

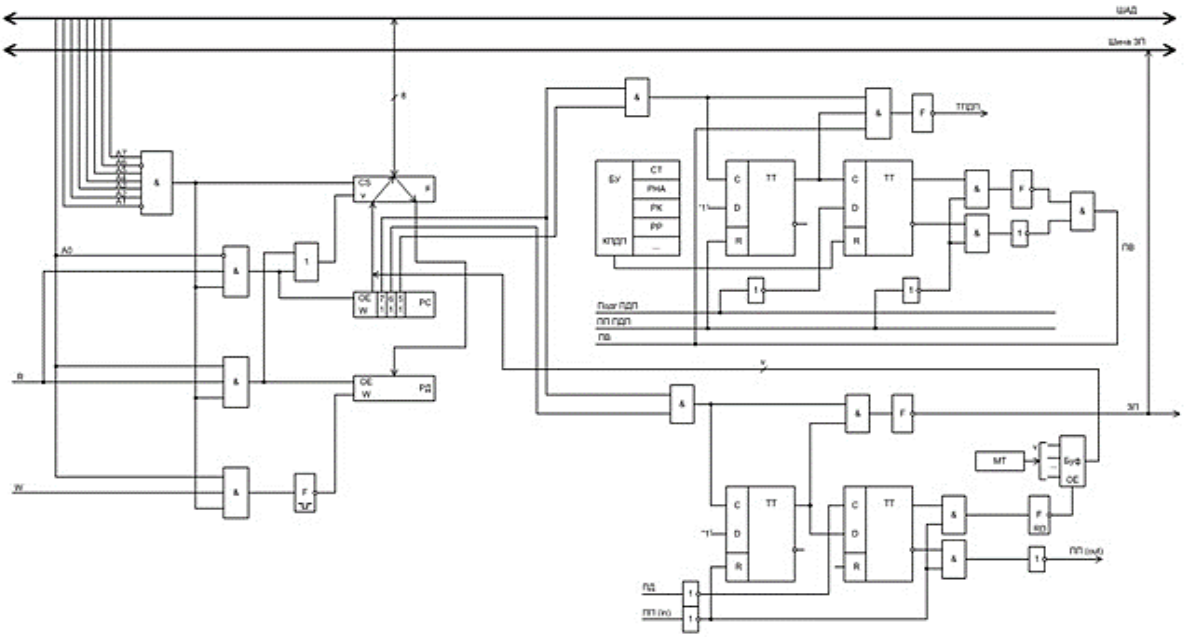


Рис. 4.18. Схематичне зображення функціональної схеми розробки програмного модулю шифрування даних

Створення прототипу алгоритму проведемо у середовищі Visual Studio. Успішна компіляція програмної моделі зображено в додатку даної роботи.

4.5 Тестування додатку

Для реалізації програмного модуля шифрування даних у багатокористувацькій грі використовувалася мова програмування C++.

Для досягнення поставленої мети створення програмного модулю була розроблена структурна схема (див. рис. 4.19). Ця схема показує систему, яка передбачає взаємодію з персональним комп'ютером, перетворюючи USB сигнали в UART телеграми. Прошивка програмного модулю буде завантажуватися через середовище KEIL uVision з використанням офіційного відлагоджувача ST-Link V2 від фірми STMicroelectronics. У схемі також передбачені вузли для скидання та стабілізації живлення.

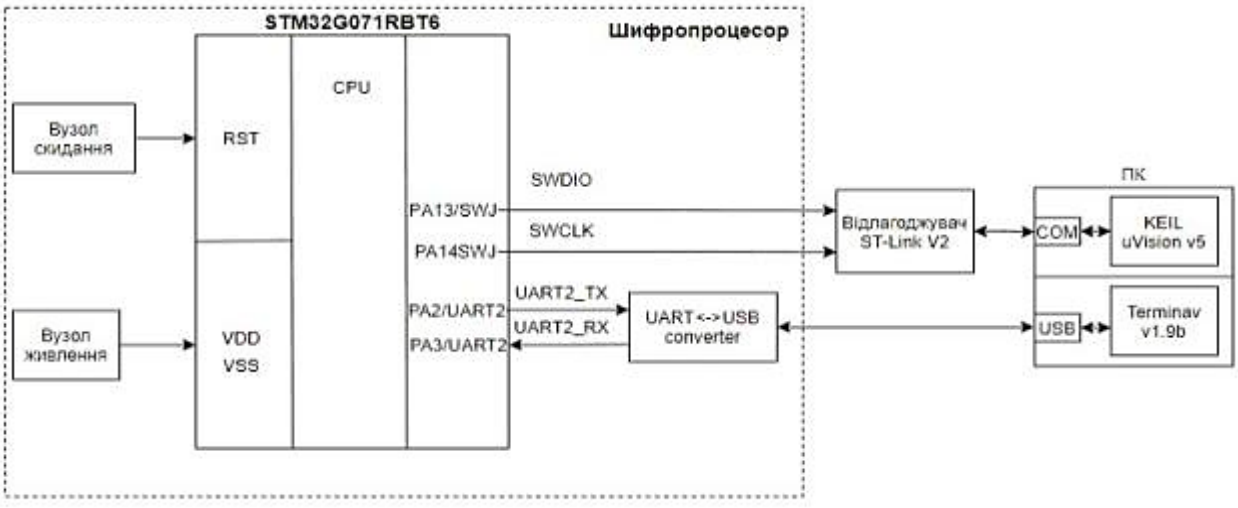


Рис. 4.19. Схема підключення модулю шифрування даних у багатокористувацькій грі

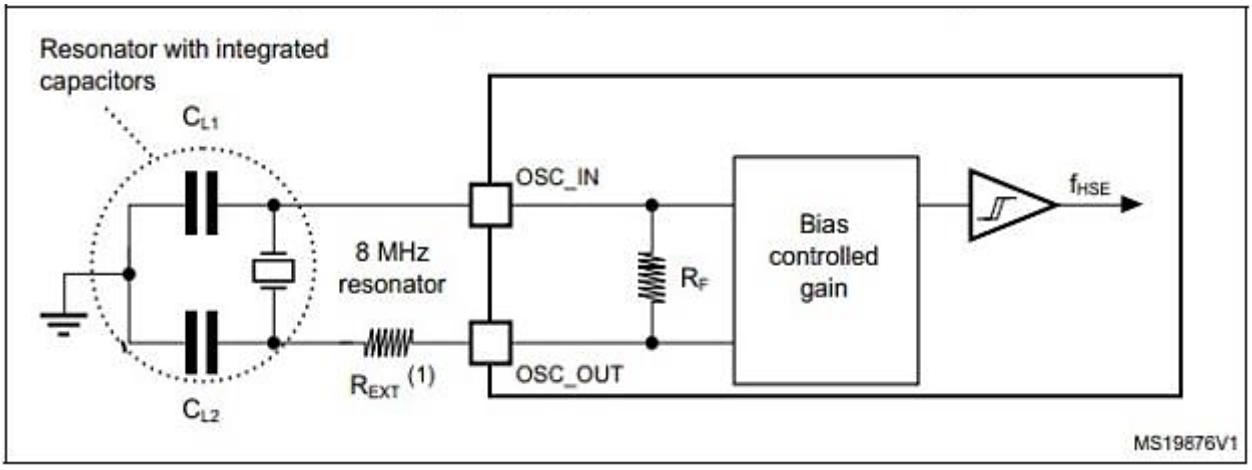


Рис. 4.20. Схема підключення мікроконтролера в проєкті

При підключенні мікроконтролера до виводів CL1 і CL2 рекомендується використовувати високоякісні зовнішні керамічні конденсатори у діапазоні від 5 ПФ до 20 ПФ, які призначені для високочастотних застосувань і обираються відповідно до вимог мікроконтролера або резонатора (див. рис. 4.21). Зазвичай конденсатори CL1 і CL2 мають однаковий розмір. При визначенні розмірів цих конденсаторів необхідно враховувати ємність виводів печатної плати і мікроконтролера (можна використовувати 10 ПФ як приблизну оцінку комбінованої ємності виводів і плати) CL1 і CL2.

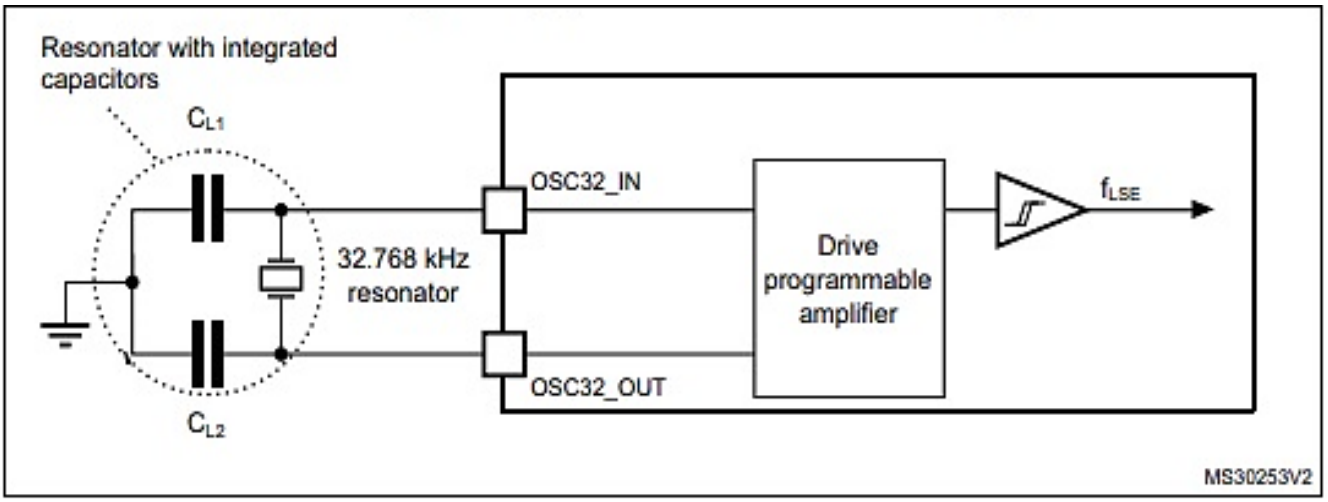


Рис. 4.21. Порядок підключення мікроконтролера в проєкті при частоті 32,768 кГц

У цьому проєкті, пристрій піддається навантаженню за допомогою подачі струму на контакти введення-виведення, які запрограмовані в режимі плаваючого введення. Під час поетапної подачі струму на виводи введення-виведення, в проєкті проводиться перевірка на наявність функціональних збоїв.

Функціональний збій визначається за наступними параметрами:

1. Помилка АЦП, коли значення перевищує певну межу (більше 5 BT LSB).

2. Індукований струм витоку на сусідніх виводах виходить за звичайні межі (діапазон – 5 мкА / +0 мкА).

3. Інші функціональні збої, такі як виникнення скидання або відхилення частоти генератора.

Порядок роботи мікропроцесора, який виконує ці перевірки, представлений в додатку даної роботи. Типова схема підключення компонентів проєкту з використанням АЦП представлена на рис. 4.22.

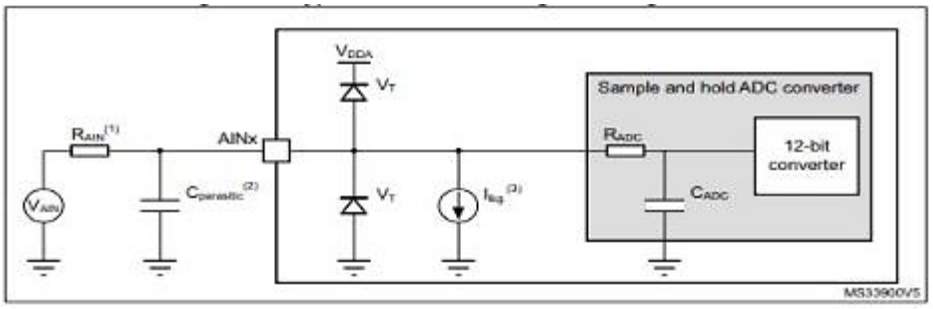


Рис. 4.22. Типова схема підключення компонентів проєкту з використанням АЦП

Порядок установки драйверів для мікропроцесора PL–2303 Windows Driver Installer v1.5.0 представлено на рис. 4.23.

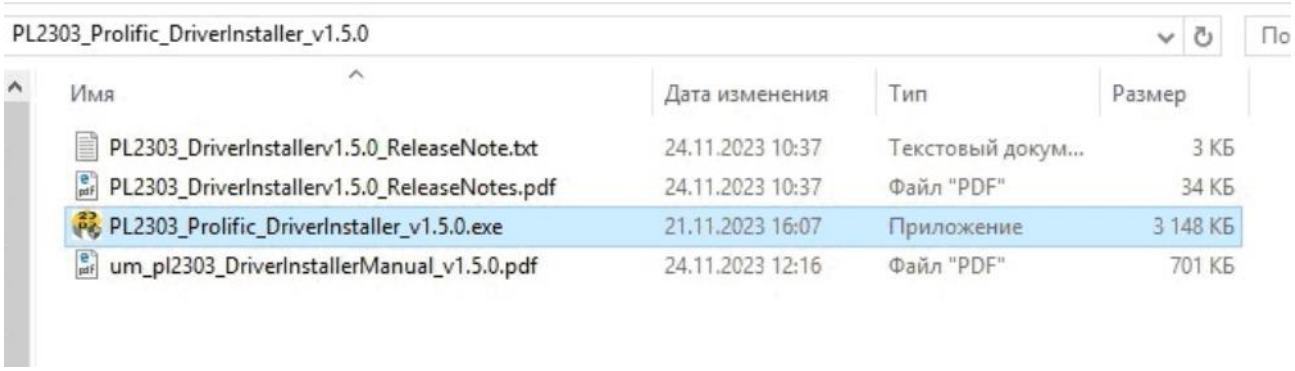


Рис. 4.23. Порядок установки драйверів для мікропроцесора PL–2303

Далі потрібно перевірити у вікні «Установка програм» появу запису «PL–2303 USB-to-Serial» (v1.5.0).

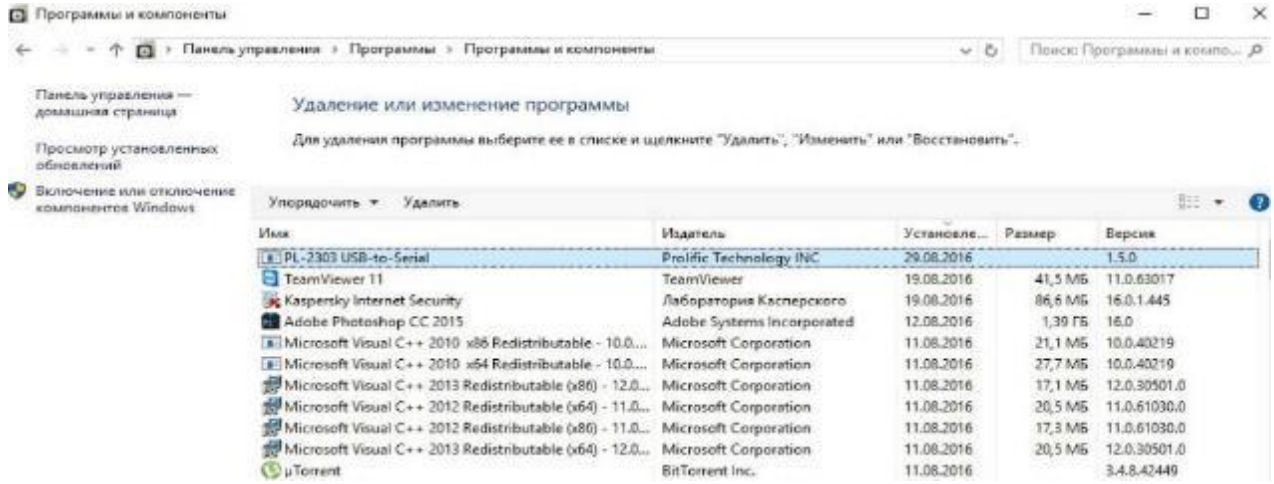


Рис. 4.24. Перевірка у вікні «Установка програм» появу запису «PL–2303 USB-to-Serial»

Далі, для підключення перетворювача до ПК, слід виконати наступні кроки:

1. Підключіть перетворювач до ПК за допомогою відповідного кабелю.

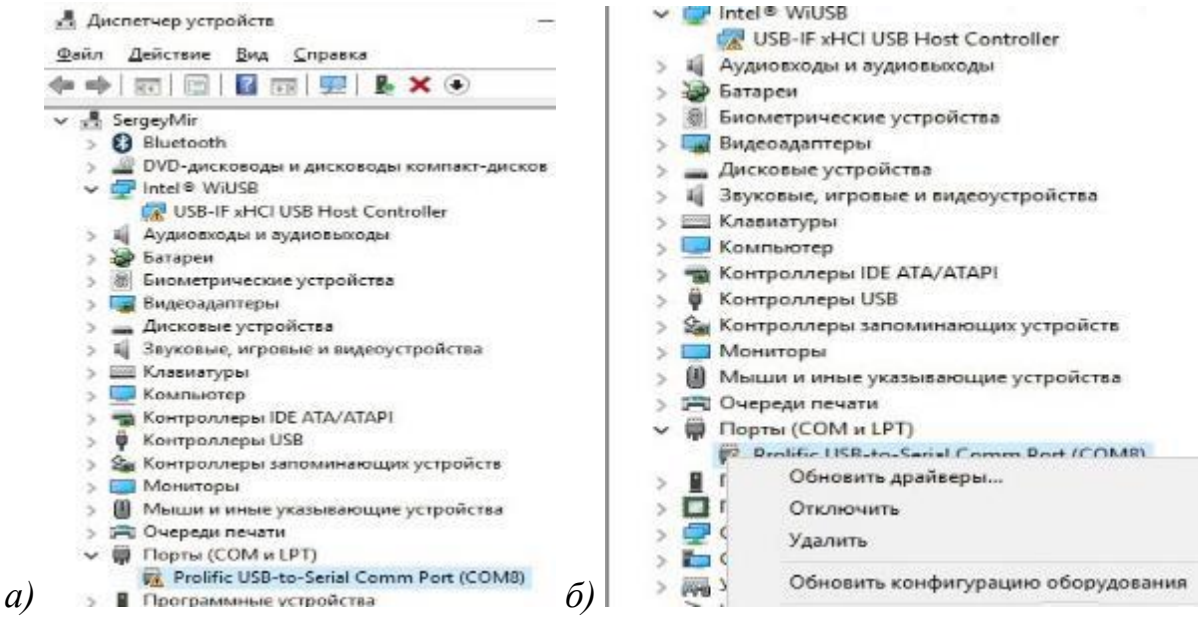
2. Натисніть «Пуск» на вашому комп'ютері і виконайте пошук за допомогою фрази «Диспетчер пристроїв».

3. Оберіть «Диспетчер пристроїв» з результатів пошуку і відкрийте його.

4. У вікні «Диспетчера пристроїв» ви знайдете розділ «Порти (COM та LPT)». Розгорніть цей розділ.

5. Якщо ваш комп'ютер підключений до Інтернету, Windows спробує автоматично оновити драйвер для вашого перетворювача. У цьому випадку, він може завантажити останній драйвер, який може бути несумісний з вашим пристроєм і призвести до помилки 10 для "Prolific USB-to-Serial Comm Port".

Для уникнення цієї проблеми, вам може знадобитися встановити власний драйвер, який сумісний з вашим перетворювачем. Цей драйвер зазвичай надається виробником пристрою або на їхньому веб-сайті. Встановлення правильного драйвера допоможе уникнути помилки 10 і забезпечить правильну роботу вашого перетворювача.



а) *б)*

Рис. 4.25. Підключення перетворювача до ПК – а) та оновлення драйверів на ПК – б)

Для переустановлення драйвера, потрібно зробити наступне:

1. Правою кнопкою миші клацніть на «Prolific USB-до-Serial Comm Port».

2. Виберіть опцію «Оновити драйвер».

3. Переконайтеся, що «Prolific USB-до-Serial Comm Port» встановлено правильно.

4. Клацніть правою кнопкою миші ще раз на пристрої і перевірте версію драйвера в властивостях пристрою.

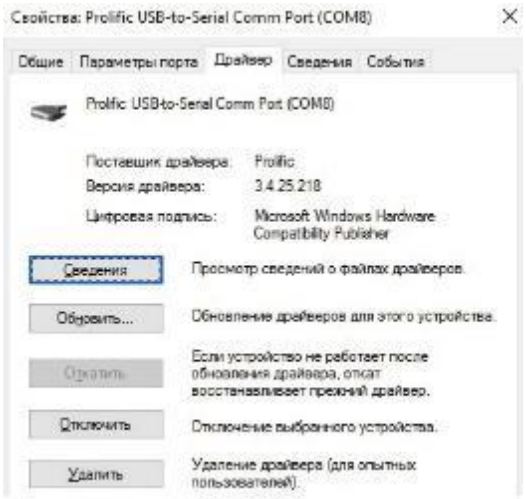


Рис. 4.26. Підтвердження версії драйвера після установки на ПК

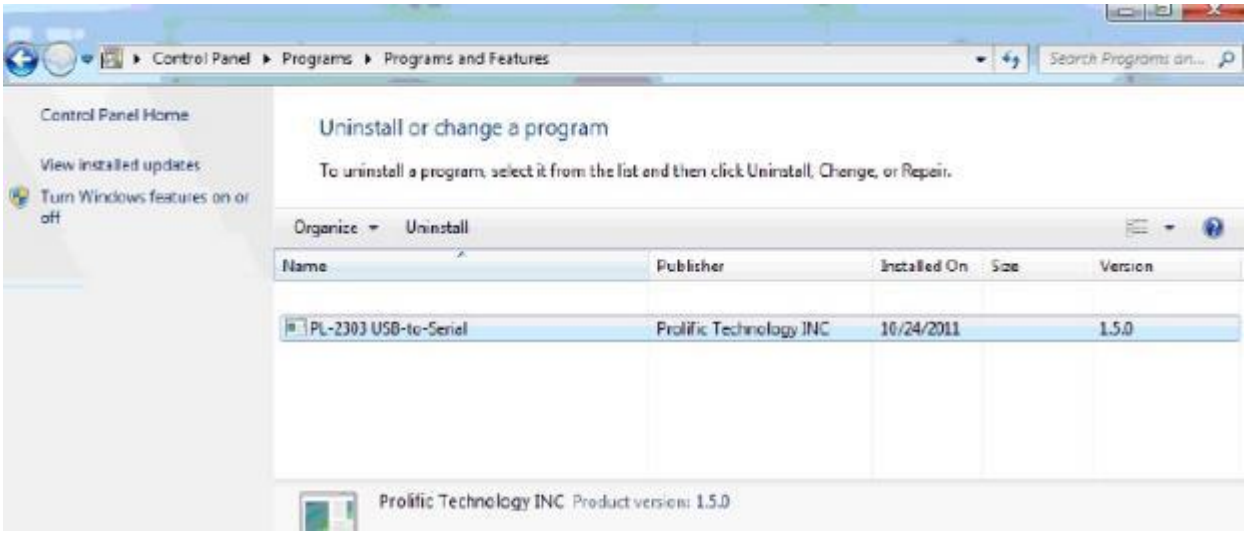


Рис. 4.27. Видалення старих драйверів з ПК

Для початку роботи проєкту встановлюємо необхідний драйвер для конвертера USB-UART на основі чіпа FT232. Запускаємо «Terminal1_9_b» від імені адміністратор. У верхньому кутку, вибираємо «COM Port «(можна подивитися в диспетчері пристроїв) і натискаємо «Connect».

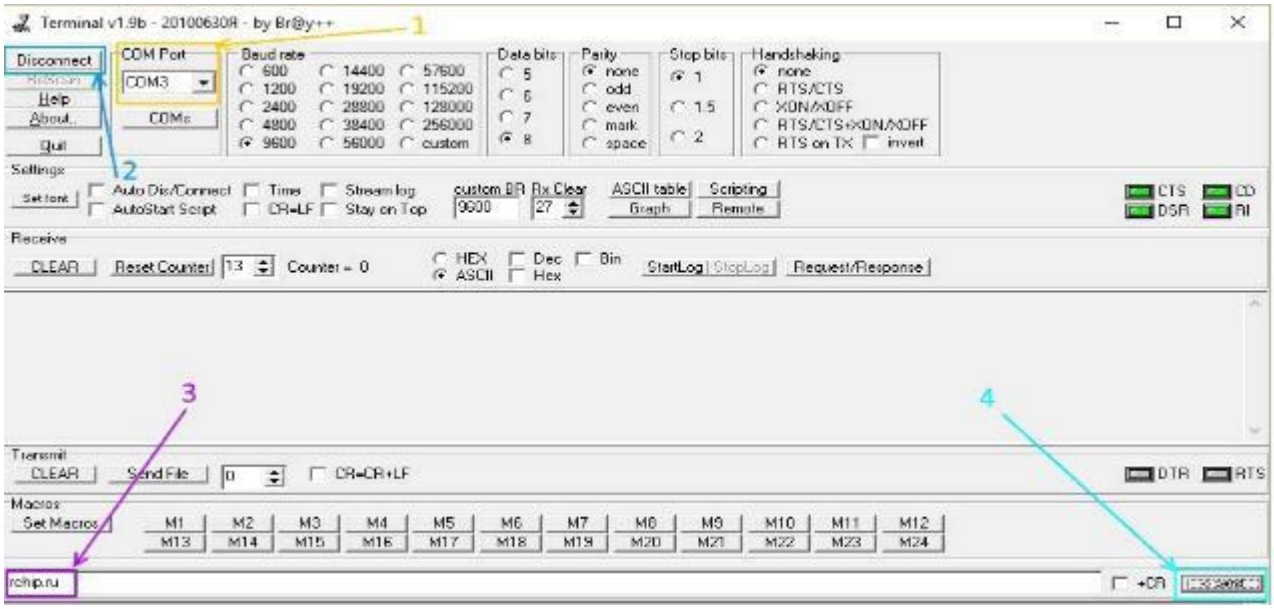


Рис. 4.28. Робоче вікно програми по інсталяції проєкту

Внизу вводимо довільне значення і натискаємо «-> Send», короткочасно загориться світлодіод TxD при кожному натисканні.

Замкніть виведення TxD і RxD між собою і натисніть «- > Send», короткочасно загоряться два світлодіоди, TxD і RxD при кожному натисканні, так само в програмі відобразиться послана команда.

Рис. 4.29. Робота в середовищі проєкту

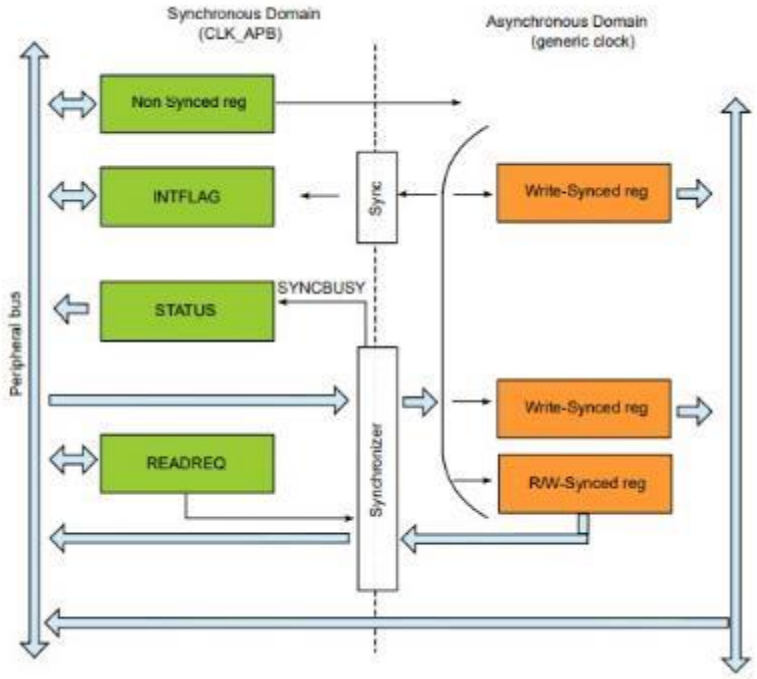


Рис. 4.30. Алгоритм синхронізації компонентів проєкту

Класи SplineFirst, SplineSecond і SplineThird є нащадками класу Spline і мають подібну структуру, але відрізняються виконуваними діями у своїх методах.

Клас Spline відповідає за операції в кінцевих полях і включає наступні елементи:

1. Поле modDefault, яке зберігає число, за модулем якого проводяться розрахунки.

2. Метод Mod, який виконує операцію ділення за модулем.

Програмний модуль складається з таких компонентів:

1. Модуль шифрування даних, який отримує вхідні повідомлення по блокам та виконує операції шифрування чи дешифрування, повертаючи результати.

2. Модуль обробки даних, який отримує вхідний потік, розбиває його на блоки, доповнює, якщо необхідно, та надсилає на шифрування / дешифрування в "Модуль шифрування даних", а також генерує та перевіряє ключі шифрування.

3. Модуль керування, який обробляє запити з графічного інтерфейсу, перевіряє вхідні дані на правильність, комутує дані та команди між іншими модулями.

Основні функціональні можливості розробленого модуля включають:

1. Зашифровку та розшифровку даних з файлу в файл.
2. Зашифровку даних з текстового поля в файл.
3. Розшифровку даних з файлу в файл.
4. Розшифровку даних з файлу в текстове поле.
5. Шифрування / дешифрування даних трьома алгоритмами на основі сплайн-вейвлетових перетворень першого, другого та третього порядку.
6. Генерацію ключів шифрування.
7. Зберігання ключів в файлі.
8. Відкриття ключів з файлу.

Для зашифровки вибирається алгоритм шифрування та розмір блоку шифрування. Також обирається, чи потрібно шифрувати чи розшифровувати дані, а також роботу з файлами чи текстом. Для зашифровки тексту в файл потрібно ввести текст у текстове поле і вказати шлях до вихідного файлу.

Система також перевіряє правильність ключа та довжину блоку шифрування.

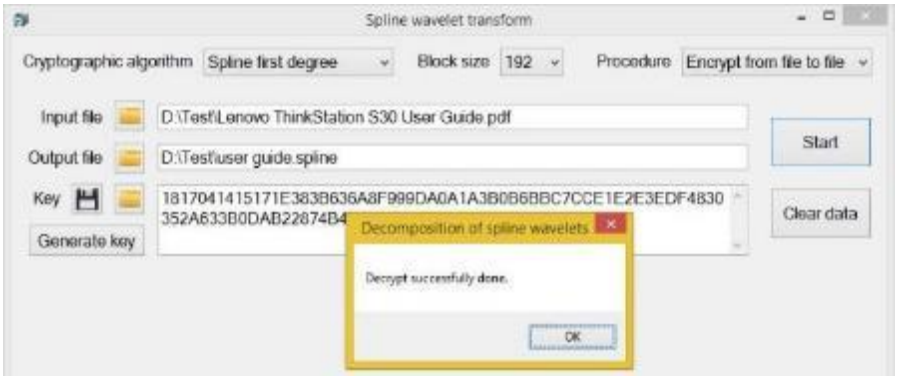


Рис. 4.31. Приклад успішного завершення процесу шифрування

Файл із зашифрованим вмістом рекомендується зберігати у форматі ".spline". Під час дешифрування у текстовий вигляд, результат буде

відображено у нижньому полі. Інформація про вхідний файл вводиться так само, як і для вихідного файлу. Значення ключа відображається в шістнадцятирічному форматі і може бути редаговано вручну. Ключ також можна зберігати в файлі та відкривати його з файлу. При генерації ключа враховуються встановлені параметри алгоритму та розміру блоку, які служать вхідними даними для його створення.

РОЗДІЛ 5.
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Екологічна безпека - це концепція, що стосується збереження природних ресурсів та екосистем для забезпечення здоров'я людей і довкілля в цілому. Це не лише захист від екологічних катастроф, таких як забруднення атмосфери, знищення лісів, вимирання видів, а й врахування сталого використання ресурсів та планування дій з метою збереження природи на майбутнє.

Ця тема охоплює багато аспектів, включаючи:

1. Збереження ресурсів: Мінімізація використання необхідних природних ресурсів, зокрема води, повітря, лісів, ґрунту та інших природних складових середовища.

2. Зменшення забруднення: Зменшення викидів токсичних речовин, виробничого сміття та інших факторів, що негативно впливають на довкілля.

3. Біорізноманіття: Збереження та захист різноманітності життя, зокрема рослин, тварин та мікроорганізмів.

4. Стале використання ресурсів: Розвиток технологій та практик, які дозволяють ефективно використовувати природні ресурси без їх вичерпання на майбутнє.

5. Захист екосистем: Діяльність з метою підтримки та відновлення природних екосистем, таких як ліси, водні басейни, морські та океанські екосистеми.

6. Свідоме споживання: Підвищення свідомості громадян про вплив їхніх дій на довкілля та сприяння усвідомленому споживанню та поведінці.

7. Міжнародне співробітництво: Спільна робота країн та міжнародних організацій для вирішення екологічних проблем, які перетинають кордони.

Екологічна безпека ґрунтується на:

- усвідомленні того, що людство — невід'ємна частина природи, повністю залежна від навколишнього його середовища;
- визнанні обмеженості і конечності природно-ресурсного (екологічного) потенціалу Землі і окремих її регіонів, необхідності його якісної та кількісної інвентаризації;
- неможливості штучного розширення природно-ресурсного (екологічного) потенціалу понад природно-системні обмеження;
- визначенні допустимого максимуму вилучення природних ресурсів і зміни екосистем як середовища життя;
- необхідності вироблення превентивних екологічних заборон задовго до економічного вичерпання природних ресурсів або їх непрямого руйнування;
- обов'язковості створення соціально-економічного механізму гомеостазу в системі «людина — природа» типу «природа — товар — гроші — природа» (аналогічно механізму «товар — гроші — товар»);
- нагальної і обов'язкової необхідності регулювання чисельності людей, їх тиску на природне середовище на локальному, регіональному та глобальному рівнях;
- прийнятності тільки «екологосумісних» технологій і техніки в усіх галузях господарювання;
- переході до ресурсоекономних технологій і мініатюризації виробів, до безпечних для природи і людей господарських прийомів;
- визнанні закону оптимальності, а в господарюванні — принципу розумної достатності у використанні способів отримання життєвих благ в просторових і часових конкретних рамках (обмеження по факторах екологічного, соціального і економічного ризику);

- розумінні, що без адекватного середовища життя (цілісності екосистем) неможливе збереження нічого живого, в тому числі його видів (включаючи людину) і природних систем більш низького рівня ієрархії.

Екологічна безпека складається з:

- екологічного аудиту,
- моніторингу,
- прогнозу розвитку екологічної ситуації
- екологічного менеджменту
- Екологічна проблема

Над проблемами забезпечення екологічної безпеки господарських процесів в період кінця XX — на початку XXI ст. плідно працюють багато зарубіжних і українських вчених, зокрема: Ансоф І., Балацький О. Ф., Борщевський П. П., Буркінський Б. В., Веклич О. П., Вишняков Я. Д., Волошин В. В., Галушко О. С., Герасимчук З. В., Гірусов Е. В., Горлачук В. В., Прусов Є. В., Джігірей В.С., Дорогунцов С. І., Єрмаков В.М., Кредісов А. П., Ляшенко І., Лебединський Ю. П., Ландар Г. І., Луньова О.В., Мельник Л. Г., Міщенко В. С., Паламарчук В. О., Путілов А. В., Саллі В. І., Сахаєв В. Г., Сокур М. І., Стадницький Ю. І., Степанов В. Н., Тимченко О. Г., Трегобчук В. М., Туниця Ю. Ю., Турило А.М., Федорищева А. М., Чумаченко М. Г., Чухно А., Шевчук В. Я., Шмандій В. М., Бондар О.І., Горобей М.С., а також Фелікс Доддс, Норман Майєрс, Джесіка Тачман Метьюз, Майкл Реннер, Річард Уллман, Артур Вестінг, Майкл Клейр, Томас Гомер Діксон, Джеффрі Дабелко, Пітер Глейк, Ріта Флойд і Джозеф Ромм та ін.

Досягнення екологічної безпеки вимагає поєднання наукових знань, технологічного прогресу, політичної волі та активної участі суспільства на всіх рівнях, щоб забезпечити збалансоване використання ресурсів і збереження природи для майбутніх поколінь.

ВИСНОВКИ

У роботі були розглянуті основні поняття комп’ютерної гри та був представлений огляд підходів до їх класифікації (по ігровій платформі та графіці, за змістом та видавничими критеріями, за типом їх поширення та кількістю гравців), що дало змогу сформулювати напрямок методики шифрування даних в них.

Розглянуто програмну (бібліотека Cryptopp) та апаратну реалізацію на ПЛІС. Дало змогу описати підхід до створення програмного модулю шифрування даних у багатокористувацькій грі, а також проаналізовано архітектуру програмної реалізації, виконано розробку структури бази даних та use case діаграми модулю шифрування даних

Була розроблена структура програмного комплексу, визначено його функціональні можливості з наведенням пояснень. Реалізовані алгоритми шифрування/дешифрування з використанням сплайнів. На базі одного із алгоритмів був створений модуль шифрування даних, наведено алгоритм його роботи. Наведено детальну, покрокову інструкцію з використання розробленого продукту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 7624–2014 Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення «Калина (з внесеними змінами в 2015 р.)». Київ, 2014. С. 20–23. (дата звернення: 10.12.2023).

2. ДСТУ ГОСТ 28147:2009 Система обробки інформації. Криптографічний захист. Алгоритм криптографічного перетворення. Київ, 2009. С. 10–15. (дата звернення: 10.12.2023).

3. ДСТУ 8845:2019 Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного потокового перетворення. – Режим доступу: http: // online. budstandart. com / ua / catalog / doc-page. html? id_doc=82494. (дата звернення: 10.12.2023).

4. Алгоритми та структури даних / уклад. О. В. Щербаков, Ю. Е. Парфьонов, В. М. Федорченко. Харків: ХНЕУ ім. С. Кузнеця, 2017. 58 с.

5. Алієва О. Віртуальні ігри як феномен сучасної культури. *Схід.* 2016. № 5. С. 64–67.

6. Біленко В. М., Глухов В. С. Принципи реалізації ядра криптопроцесора, що реалізує алгоритм Калина для мікроконтролера. *Переваги кіберфізичних систем.* 2021. №6. С. 57–64.

7. Безчотнікова А. О. Гра, та її вплив на поведінку людини. Інформаційне суспільство: науковий збірник КНУ імені Тараса Шевченка. К.: КНУ, 2016. Вип. 23 (січень – червень). С. 102–107.

8. Бучик С. С. Методика оцінювання інформаційних ризиків в автоматизованій системі. *Проблеми створення, випробування, застосування та експлуатації складних інформаційних систем: зб. наук. праць*. Житомир: ЖВІ ДУТ, 2015. Вип. 11. С. 33–43.

9. Васильцов І. В. Методи захисту проти атак спеціального виду. *Вісник Хмельницького національного університету.* 2007. №5. С. 174–182.

10. Вовк О. Б. Комп'ютерні ігри як форма інтерактивної інформаційної системи. *Математичні машини і системи,* 2017, № 4. С. 98–103.

11. Гаврилова Н. В., Москальов Т. К. Потенціал комп'ютерних ігор у зміні політики національної безпеки держави. *Вісник Маріупольського державного університету. Серія: Історія*, 2017, Вип. 1. С. 103–110.

12. Горбенко І. Д. Перспективний блоковий симетричний шифр «Калина»: основні положення та специфікації. *Прикладна радіоелектроніка*. 2007. Т. 6, № 2. С. 195–208.

13. Горбенко Ю. І. Методи побудування та аналізу криптографічних систем. Х.: Форт, 2015. 959 с.

14. Гордієнко А. В. Комп'ютерні ігри та їхні позитивні психологічні ефекти. *Наукові записки НаУКМА*. 2017. Том 199. С. 58–62.

15. Джамал Х. М. Алгоритм і структура модуля для обчислення квадратного кореня в ПЛІС. *Праці міжнародної конференції «Безпека, відмовостійкість, інтелект»*, 2018. С. 74–77.

16. Дьоміна М., Біленко В., Глухов В. Валідація реалізації блокового шифру Калина за допомогою тестових наборів. *Матеріали конференції: «Litterisetartibus»*. Львів: «Львівська політехніка». 2019, С. 62–64.

17. Заболоцький Т. Кіберпростір як інструмент соціального впливу на сучасну молодь. *Ввічливість. Humanitas,* 2021. № (1). С. 22–27.

18. Єфіменко А. А., Байлюк Є. М. Порівняльний аналіз алгоритму симетричного блокового перетворення Калина з іншими міжнародними стандартами шифрування даних. *Зібрання наукових робіт Житомирського військового інституту*. 2019, №18, С. 124–142.

19. Калінін Д. О., Козіна Г. Л. Швидкодія шифрів «Калина» й AES. *Математичне та комп'ютерне моделювання. Радіоелектроніка, інформатика, управління*. 2013. № 1. С. 62–65.

20. Кісільчук Б. Я., Тесленко О. К. Ключі алгоритму шифрування на базі підстановок довільної розрядності. *Системний аналіз та інформаційні технології*, 2018. С. 230–231.

21. Корченко О. Прикладні системи оцінювання ризиків інформаційної безпеки. К., 2017. 435 с.

22. Кузнецов О. О., Олійников Р. В. Обґрунтування вимог, побудування та аналіз перспективних симетричних криптоперетворень на основі блочних шифрів. *Вісник Нац. ун-ту «Львів. політехніка»,* 2014. № 806, С. 124–140.

23. Кузь М. В., Соловко Я. Т. Методологія формування узагальненого критерію якості програмного забезпечення в умовах невизначеності. *Вісник Вінницького політехнічного інституту*. 2015. №5. С. 104–107.

24. Лисицький К. Є. Оптимізація перспективних алгоритмів блокового симетричного перетворення по критеріям швидкодії і стійкості. *Математичне та комп'ютерне моделювання: зб. наук. праць*. 2017. Вип. 15. С. 115–119.

25. Луцків А. М. Архітектури високопродуктивних систем опрацювання великих даних. *Збірник тез доповідей Науково-технічна VI Науково-технічна конференція «Інформаційні моделі, системи та технології»,* 12–13 грудня 2018 року. Т.: ТНТУ, 2018. С. 75.

26. Мальцева І. Р. Аналіз деяких кіберзагроз в умовах війни. *Кібербезпека: освіта, наука, техніка.* 2022. № 4 (16). С. 37–43. DOI10.28925 / 2663–4023.2022.16.3744.

27. Мудрий Я. П. Комп'ютерні ігри та їх класифікація. *Актуальні питання сучасної інформатики*, 2016. № 2. С. 133–138.

28. Олійников Р., Горбенко І. Принципи побудови і основні властивості нового національного стандарту блокового шифрування України. *Захист інформації*, 2015. Том 17, № 2, С. 142–157.

29. Пантєлєєва Н. М. Кіберзагрози в умовах цифрової економіки. *Фінансовий простір.* 2019. № 1. С. 130–139.

30. Поворозник В. С. Сучасні методи підвищення стійкості криптоалгоритмів. *Інтелектуальні комп'ютерні системи та мережі.* Тернопіль, 2020. С. 55.

31. Пузиренко О. Г. Аналіз процесу управління ризиками інформаційної безпеки в забезпеченні живучості інформаційно-телекомунікаційних систем. *Системи обробки інформації*. Л.: Академія сухопутних військ імені гетьмана Петра Сагайдачного, 2014. Вип. 8 (124). С. 128–134.

32. Пузиренко О. Г. Застосування моделей оцінювання ризиків інформаційної безпеки в інформаційно-телекомунікаційних системах. *Системи обробки інформації.* Л.: Академія сухопутних військ імені гетьмана Петра Сагайдачного, 2015. Вип. 3 (128). С. 75–79.

33. Ревак І. О. Особливості формування безпечного кіберпростору в умовах розвитку цифрової економіки. *Інформаційні технології та економічна безпека.* 2021. № 3–4. С. 164–169.

34. Северина С. Інформаційна безпека та методи захисту інформації. *Вісник Запорізького національного університету.* 2016. № 1. С. 155–161.

35. Совин Я. Р. Ефективна реалізація алгоритм блокового симетричного шифрування ДСТУ 7624:2014 для 8 / 16 / 32 бітових вбудованих систем. Київ, Україна: «Сучасний захист інформації». 2017. С. 6–16.

36. Совин Я. Р. Ефективна імплементація та порівняння швидкодії шифрів «КАЛИНА» та ГОСТ 28147–89 за використання векторних розширень SSE, AVX та AVX–512. *Захист інформації*. 2019. Том 21, № 4, С. 207–223.

37. Стець В. Кіберпростір як об'єкт публічного управління в сфері забезпечення кібербезпеки. *Публічне управління: традиції, інновації, глобальні тренди: матеріали Всеукраїнської наук. – практ. конф. за міжнар. участю*. Одеса: ОРІДУ НАДУ, 2021. С. 290–292.

38. Стець В. В. Кіберпростір як основний об’єкт публічного управління у сфері забезпечення кібербезпеки. *Наукові перспективи.* 2022. № 8 (26). С. 98–115.

39. Тесленко О. К., Кісільчук Б. Я. Формування ключів алгоритму шифрування на базі лінійних структур. *Прикладна математика та комп’ютинг*. Київ, НТУУ «КПІ», 14–16 листопада 2018 р.

40. Ткач Ю. Концептуальна модель безпеки кіберпростору. *Технічні науки та технології.* 2021. № 4 (22). С. 96–108. https: // doi. org / 10.25140 / 2411–5363–2020–4 (22) – 96–108.

41. Ткач Ю. М. Тенденції розвитку сучасного кіберпростору та його захищеності в умовах інформаційного протиборства. *Безпека інформації.* 2020. Т. 26 (2). С. 74–80.

42. Фролова О. Міжнародне співробітництво в галузі забезпечення інформаційної безпеки. *Вісник Львівського університету. Серія: Міжнародні відносини*. 2019. Вип. 46. С. 123–136.

43. Фурсова Н. А. Розробка мережевої комп’ютерної гри з використанням Unity Engine. Тези 70-ої ювілейної наук. конф. проф., викл., наук. прац., аспір. та студ. університету. Том 2. (Полтава, 23 квітня – 18 травня 2018 р.). Полтава: ПолтНТУ, 2018. С. 244–245.

44. Чунарьова А. В. Аналіз підходів та програмних рішень оцінки і контролю інформаційних ризиків в комп’ютеризованих системах. *Вісник Інженерної академії України*. Х. 2014. Вип. 2. С. 138–142.

45. Шеломовська О. М. Особливості сучасних ігрових комп’ютерних практик. *Вісник Харківського національного університету імені В. Н. Каразіна,* 2017 р. С. 253–262.

46. Iryna Svyd, Oleksandr Maltsev. Review of Seventh Series FPGA Xilinx. Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGA. Kharkiv, Ukraine, July 26–27, 2019. Kharkiv: 2019. P. 25–26.

47. Iryna Svyd. Special Features of the Educational Component «Design of Devices on Microcontrollers and FPGA». Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs. Kharkiv, Ukraine, 2020, pp. 55–57.

48. Бібліотека алгоритмів шифрування. [Електронний ресурс] – Режим доступу до ресурсу: https: // www. cryptopp. com/ (дата звернення: 03.12.2023).

49. Інформаційний лист STM32G071. [Електронний ресурс] – Режим доступу до ресурсу: https: // www. st. com / resource / en / datasheet / stm32g071cb. pdf. (дата звернення: 03.12.2023).

50. Інформаційний лист LPC11U6X [Електронний ресурс] – Режим доступу до ресурсу: https: // www. nxp. com / docs / en / data-sheet / LPC11U6X. pdf. (дата звернення: 03.12.2023).

ДОДАТКИ

Додаток А

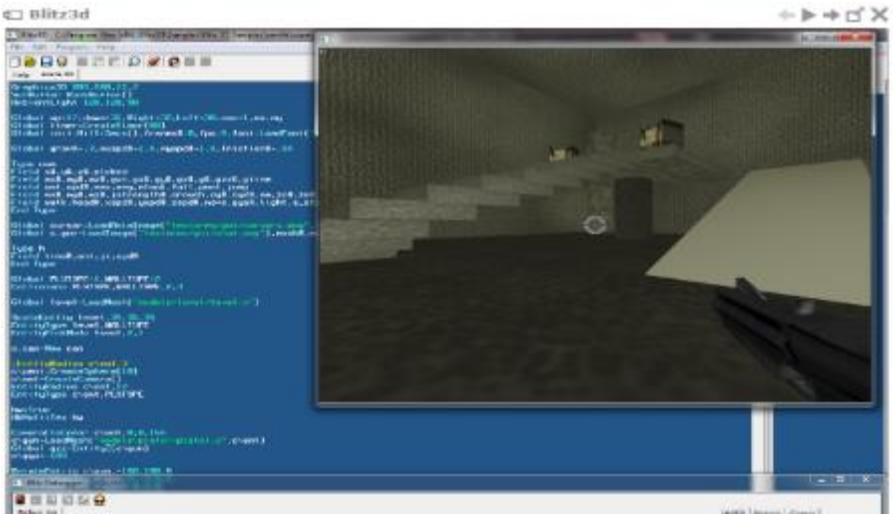
Головне меню програми Unreal Engine 4

Додаток Б

Головне меню програми Construct 2

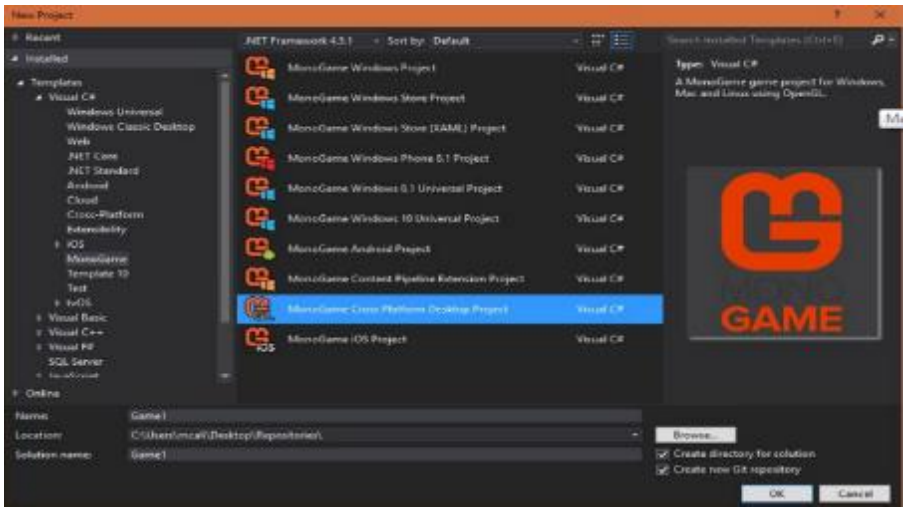
Додаток В

Головне меню програми Blitz3D



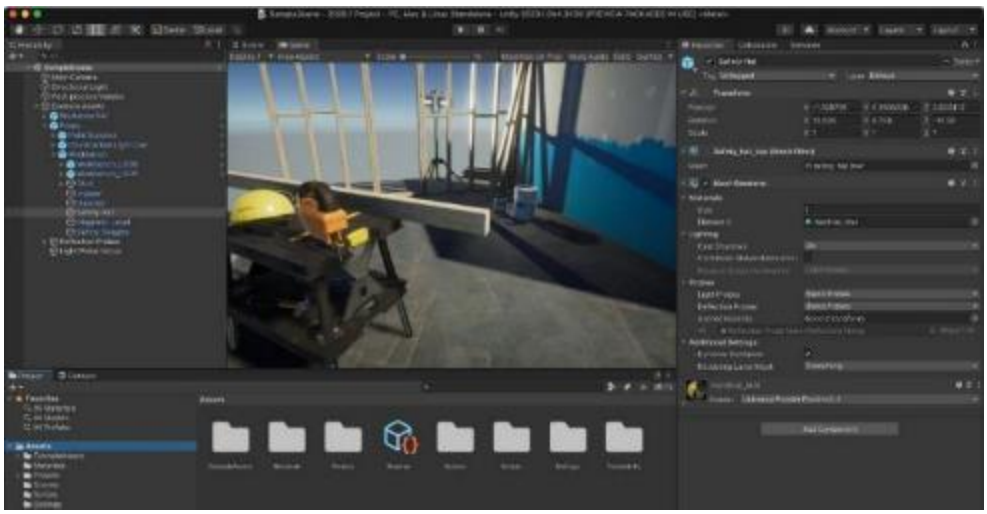
Додаток Г

Головне меню програми MoneGame



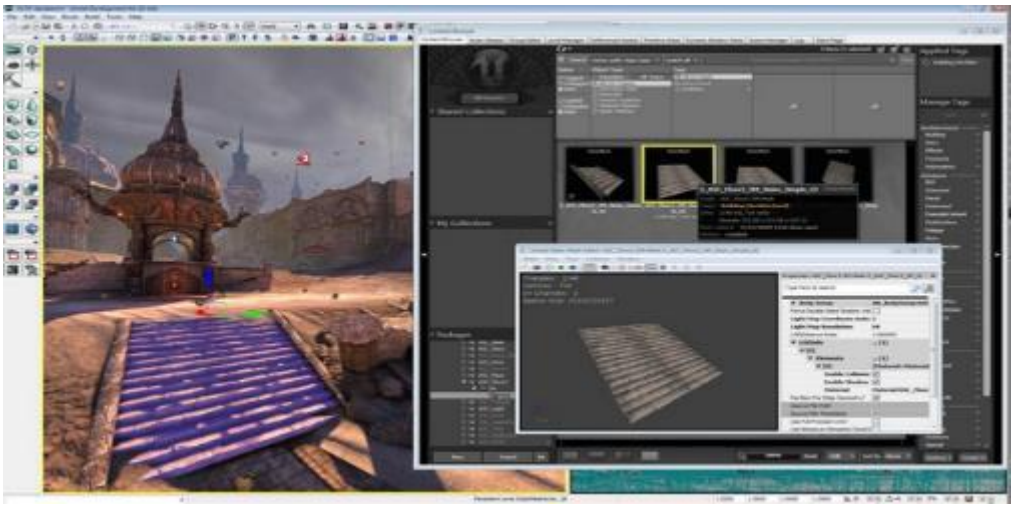
Додаток Д

Головне меню програми Unity



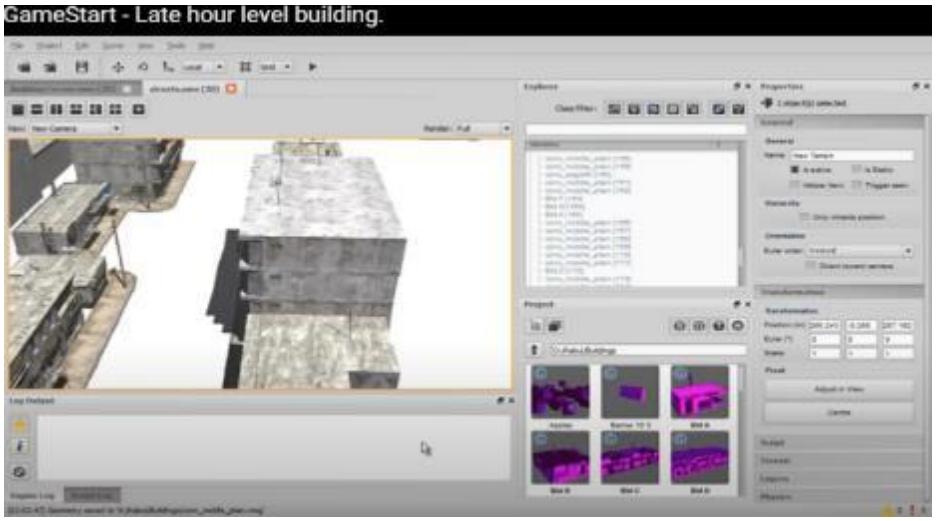
Додаток Е

Головне меню програми Unreal Engine 3



Додаток Є

Головне меню програми GameStart

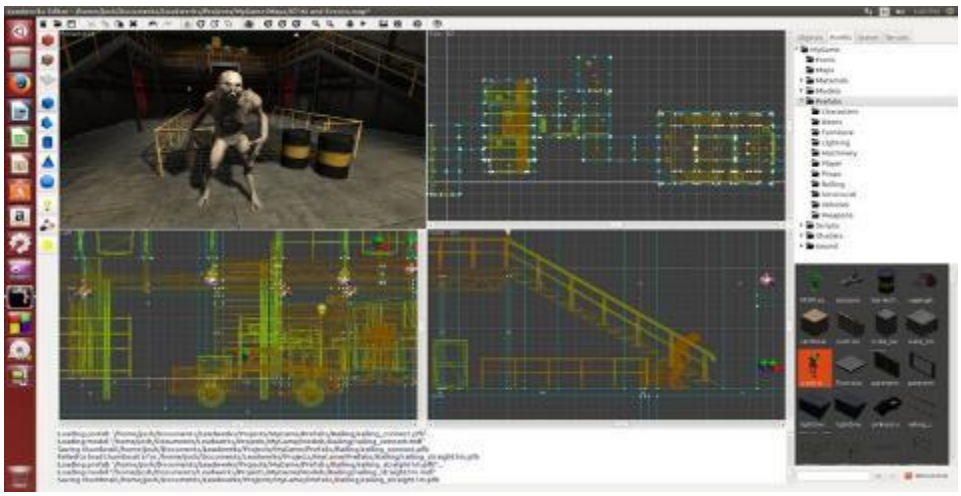


Додаток Ж

Головне меню програми ShiVa 3D

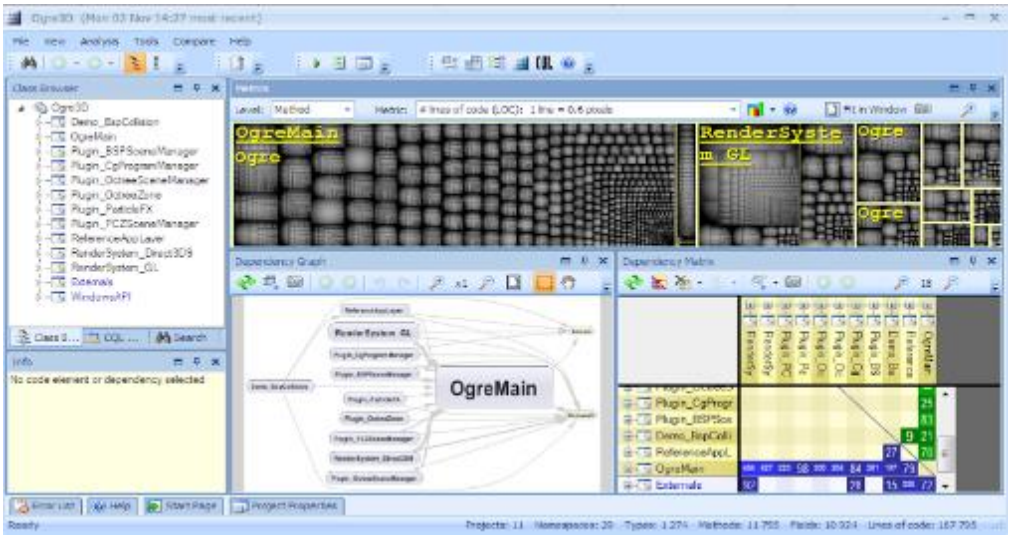
Додаток З

Головне меню програми Leadwerks Engine



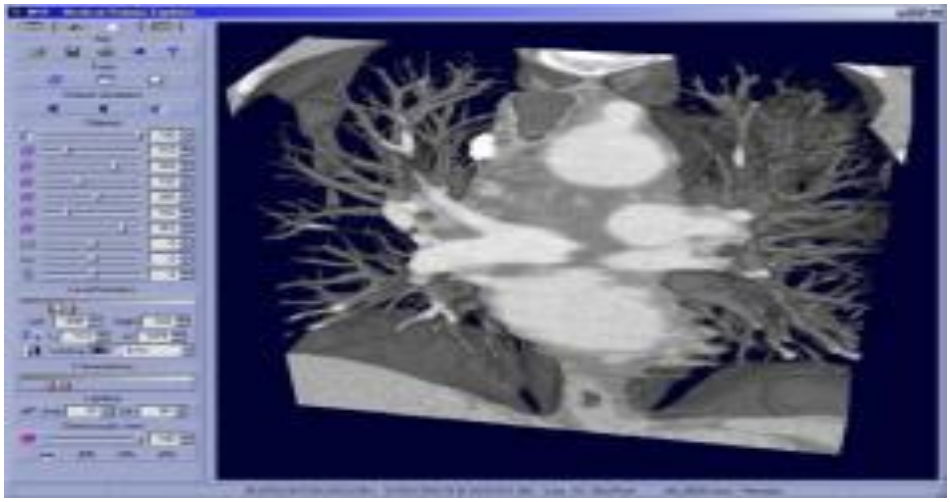
Додаток И

Головне меню програми OGRE



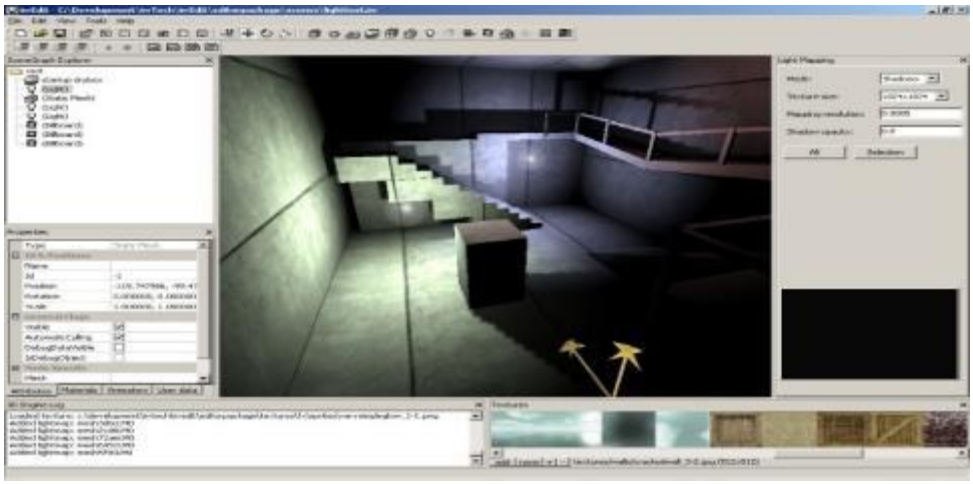
Додаток І

Головне меню програми GLScene



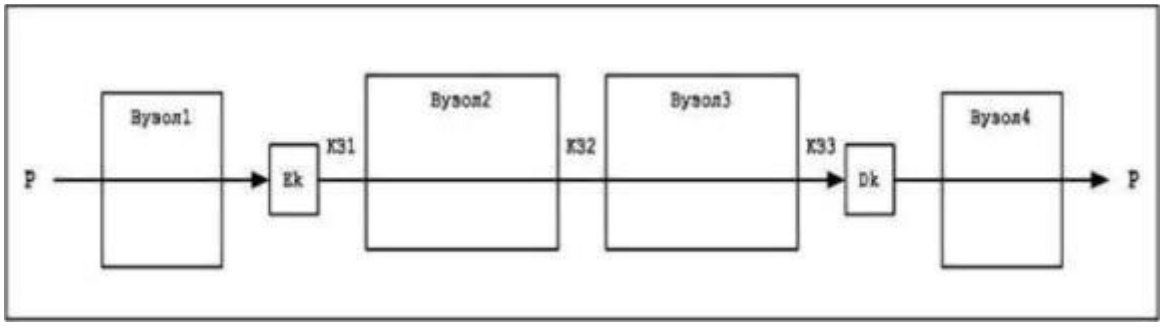
Додаток Ї

Головне меню програми Irrlicht



Додаток Й

Загальна схема альтернативного рішення шифрування



де P – вхідне повідомлення, Вузол1-Вузол4 – вузли мережі, К31-К33 – канали зв'язку, Ek – зашифрування повідомлення за допомогою ключа k, Dk – розшифрування повідомлення за допомогою ключа k.

Додаток К

Заповнення поточного стану для довжини блока 512 біт

Вхідна послідовність

in_0	in_8	in_{16}	in_{24}	in_{32}	in_{40}	in_{48}	in_{56}
in_1	in_9	in_{17}	in_{25}	in_{33}	in_{41}	in_{49}	in_{57}
in_2	in_{10}	in_{18}	in_{26}	in_{34}	in_{42}	in_{50}	in_{58}
in_3	in_{11}	in_{19}	in_{27}	in_{35}	in_{43}	in_{51}	in_{59}
in_4	in_{12}	in_{20}	in_{28}	in_{36}	in_{44}	in_{52}	in_{60}
in_5	in_{13}	in_{21}	in_{29}	in_{37}	in_{45}	in_{53}	in_{61}
in_6	in_{14}	in_{22}	in_{30}	in_{38}	in_{46}	in_{54}	in_{62}
in_7	in_{15}	in_{23}	in_{31}	in_{39}	in_{47}	in_{55}	in_{63}

Поточний стан

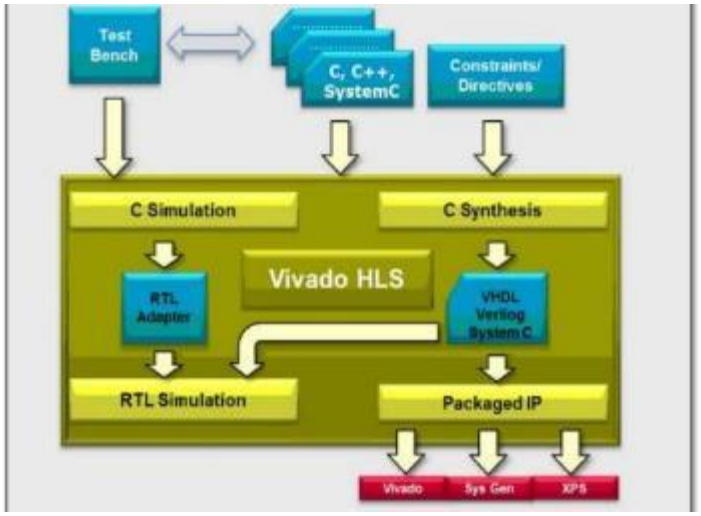
$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$	$s_{0,4}$	$s_{0,5}$	$s_{0,6}$	$s_{0,7}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,4}$	$s_{1,5}$	$s_{1,6}$	$s_{1,7}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$	$s_{2,4}$	$s_{2,5}$	$s_{2,6}$	$s_{2,7}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$	$s_{3,4}$	$s_{3,5}$	$s_{3,6}$	$s_{3,7}$
$s_{4,0}$	$s_{4,1}$	$s_{4,2}$	$s_{4,3}$	$s_{4,4}$	$s_{4,5}$	$s_{4,6}$	$s_{4,7}$
$s_{5,0}$	$s_{5,1}$	$s_{5,2}$	$s_{5,3}$	$s_{5,4}$	$s_{5,5}$	$s_{5,6}$	$s_{5,7}$
$s_{6,0}$	$s_{6,1}$	$s_{6,2}$	$s_{6,3}$	$s_{6,4}$	$s_{6,5}$	$s_{6,6}$	$s_{6,7}$
$s_{7,0}$	$s_{7,1}$	$s_{7,2}$	$s_{7,3}$	$s_{7,4}$	$s_{7,5}$	$s_{7,6}$	$s_{7,7}$

Вихідна послідовність

out_0	out_8	out_{16}	out_{24}	out_{32}	out_{40}	out_{48}	out_{56}
out_1	out_9	out_{17}	out_{25}	out_{33}	out_{41}	out_{49}	out_{57}
out_2	out_{10}	out_{18}	out_{26}	out_{34}	out_{42}	out_{50}	out_{58}
out_3	out_{11}	out_{19}	out_{27}	out_{35}	out_{43}	out_{51}	out_{59}
out_4	out_{12}	out_{20}	out_{28}	out_{36}	out_{44}	out_{52}	out_{60}
out_5	out_{13}	out_{21}	out_{29}	out_{37}	out_{45}	out_{53}	out_{61}
out_6	out_{14}	out_{22}	out_{30}	out_{38}	out_{46}	out_{54}	out_{62}
out_7	out_{15}	out_{23}	out_{31}	out_{39}	out_{47}	out_{55}	out_{63}

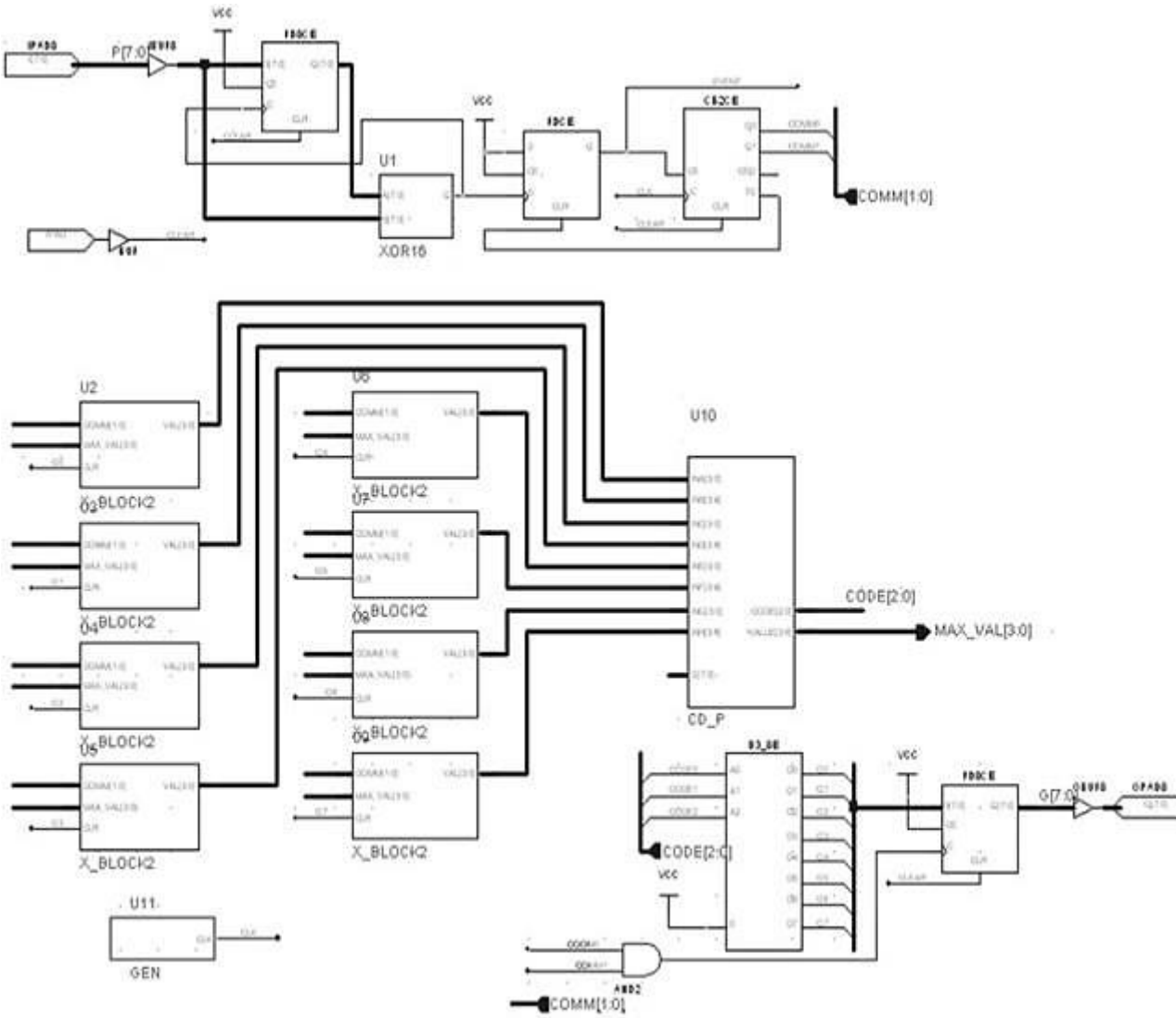
Додаток Л

Середовище розробки VHLS та схема проєктування в ViVado



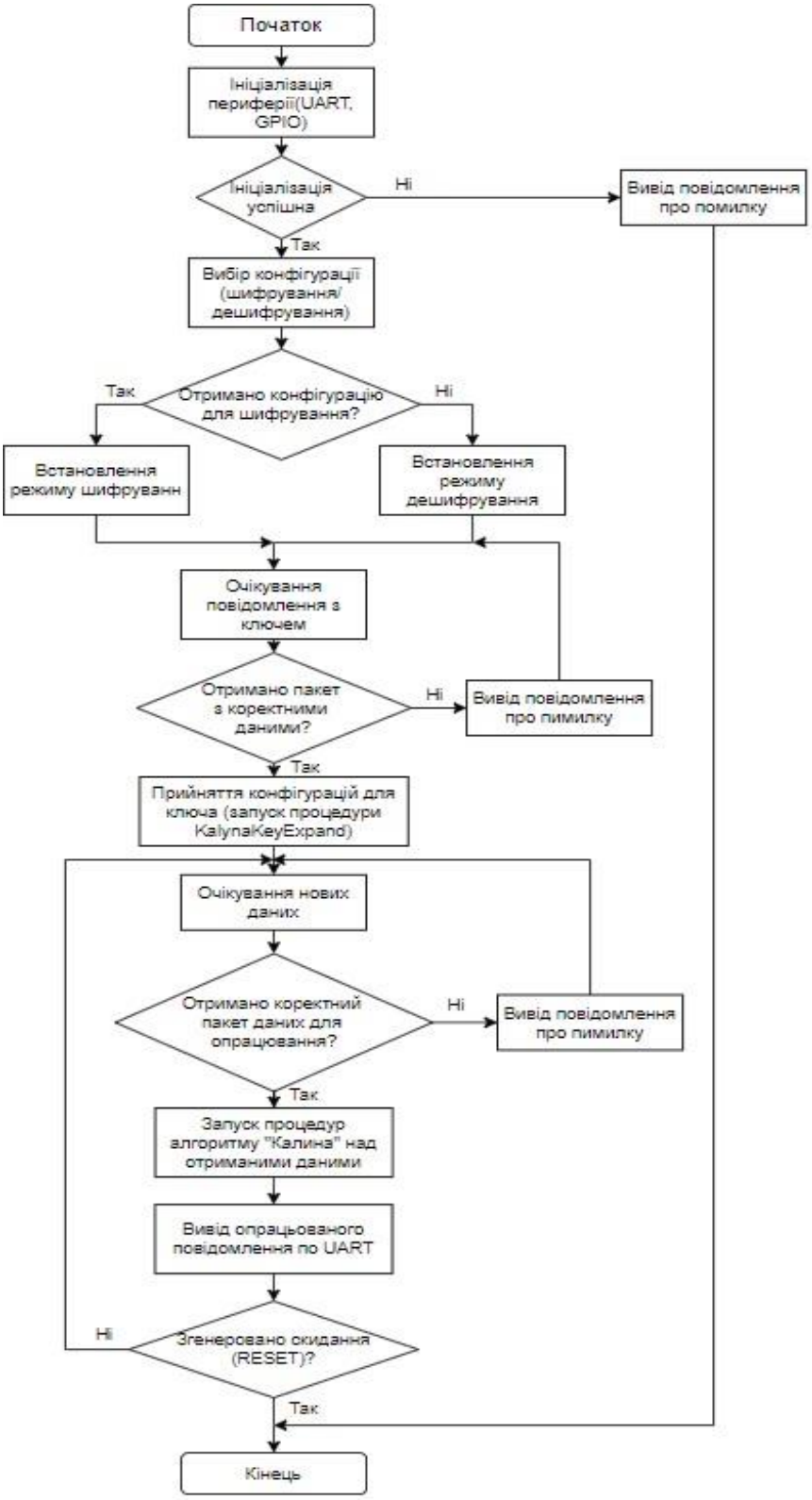
Додаток М

Схема КПП в Xilinx



Додаток Н

Блок схема алгоритму «Калина» для шифропроцесора даних у багатокористувацькій грі



Додаток О

Код на мові VHDL, згенерований програмою при синтезуванні

– RTL generated by Vivado (TM) HLS–High-Level Synthesis from C, C++ and SystemC

– Version: 2018.2

```
– Copyright (C) 1986–2018 Xilinx, Inc. All Rights Reserved.
– ==========================================================
library IEEE;
use IEEE. std_logic_1164. all;
use IEEE. numeric_std. all;
entity main is
port (
ap_start: IN STD_LOGIC;
ap_done: OUT STD_LOGIC;
ap_idle: OUT STD_LOGIC;
ap_ready: OUT STD_LOGIC;
ap_return: OUT STD_LOGIC_VECTOR (31 downto 0) );
end;
architecture behav of main is
attribute CORE_GENERATION_INFO: STRING;
attribute CORE_GENERATION_INFO of behav: architecture is
«main, hls_ip_2018_2,{HLS_INPUT_TYPE=c, HLS_INPUT_FLOAT=0, HLS_INPUT_FIXED=0, HLS_INPUT_PART=xa7a12tcsg325–1q, HLS_INPUT_CLOCK=10.000000, HLS_INPUT_ARCH=others, HLS_SYN_CLOCK=0.000000, HLS_SYN_LAT=0, HLS_SYN_TPT=none, HLS_SYN_MEM=0, HLS_SYN_DSP=0, HLS_SYN_FF=0, HLS_SYN_LUT=0, HLS_VERSION=2018_2}»;
constant ap_const_logic_1: STD_LOGIC := '1';
constant ap_const_lv32_0: STD_LOGIC_VECTOR (31 downto 0):= «00000000000000000000000000000000»;
constant ap_const_logic_0: STD_LOGIC := '0';
begin
ap_done <= ap_start;
ap_idle <= ap_const_logic_1;
ap_ready <= ap_start;
ap_return <= ap_const_lv32_0;
end behav;
```

Додаток П

Код на мові Verilog, згенерований програмою при синтезуванні

```
// ==========================================================
// RTL generated by Vivado (TM) HLS–High-Level Synthesis from C, C++ and SystemC
// Version: 2018.2
// Copyright (C) 1986–2018 Xilinx, Inc. All Rights Reserved.
// ==========================================================
`timescale 1 ns / 1 ps
(* CORE_GENERATION_INFO=«main, hls_ip_2018_2,{HLS_INPUT_TYPE=c, HLS_INPUT_FLOAT=0, HLS_INPUT_FIXED=0, HLS_INPUT_PART=xa7a12tcsg325–1q, HLS_INPUT_CLOCK=10.000000,
```

```
HLS_INPUT_ARCH=others, HLS_SYN_CLOCK=0.000000, HLS_SYN_LAT=0, HLS_SYN_TPT=none,
HLS_SYN_MEM=0, HLS_SYN_DSP=0, HLS_SYN_FF=0, HLS_SYN_LUT=0, HLS_VERSION=2018_2}» *)
        module main (
        ap_start,                ap_done,
        ap_idle,          ap_ready,
        ap_return                );
        input ap_start;          output ap_done;
              output ap_idle;          output ap_ready;
        output [31:0] ap_return; assign ap_done = ap_start;
        assign ap_idle = 1'b1;   assign ap_ready = ap_start;
        assign ap_return = 32'd0;          endmodule // main
```