Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 220013.22.03.63 ПЗ

Виконав здобувач IV курсу, гр. КІ2-22-3 _____ Ярослав ГРИГОРУК
Підпис, Ініціали, прізвище

Керівник д-р філософії _____ Олександр МЕЛЬНИЧЕНКО
Науковий ступінь, учене звання, Підпис, Ініціали, прізвище

Нормоконтролер канд.фіз.-мат.наук, доц. _____ Тетяна КИСІЛЬ
Науковий ступінь, учене звання, Підпис, Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. Підпис, Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КПС
Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Григоруку Ярославу Ігоровичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU
Керівник проєкту (роботи) Мельниченко Олександр Вікторович, д-р філософії
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз процесу обробки відеоданих та виявлення проблем ефективного використання обчислювальних ресурсів

Проєктування системи паралельного стиснення відеопотоку та обґрунтування вибору засобів реалізації

Програмна реалізація та експериментальне дослідження

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Алгоритм динамічного розподілу навантаження

Архітектура програмного забезпечення системи

Результати експериментального дослідження

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проєкту (роботи)	Термін виконання етапів проєкту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – аналіз процесу обробки відеоданих та виявлення проблем ефективного використання обчислювальних ресурсів	01.03.2026	виконано
4	Робота над розділом 2 – проєктування системи паралельного стиснення відеопотоку та обґрунтування вибору засобів реалізації	01.04.2026	виконано
5	Робота над розділом 3 – програмна реалізація та експериментальне дослідження	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно з вимогами	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Ярослав ГРИГОРУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Олександр МЕЛЬНИЧЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл.-листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 220013.22.03.63 ПЗ	Пояснювальна записка	59		
			Графічні матеріали			
2		КвРКІ 220013.22.03.63 Е8	Алгоритм динамічного	1		
			розподілу навантаження			
3		КвРКІ 220013.22.03.63 Е8	Архітектура програмного	1		
			забезпечення системи			
4		КвРКІ 220013.22.03.63 Е8	Результати	1		
			експериментального			
			дослідження			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 22013.22.03.63 ВП		
Розробив		Григорук			Відомість проєкту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Мельниченко				У	
Н. контр.		Кисіль				ХНУ, КІ2-22-3	
Затв.		Павлова		01.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU».

Автор роботи: Ярослав ГРИГОРУК.

Керівник роботи: Олександр МЕЛЬНИЧЕНКО.

Пояснювальна записка: 59 с., 23 рис., 1 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

CPU, CUDA, GPU, ВІДЕОПОТІК, ОПТИМІЗАЦІЯ ПАМ'ЯТІ, ПАРАЛЕЛЬНЕ СТИСНЕННЯ, РЕАЛЬНИЙ ЧАС, РОЗПОДІЛ НАВАНТАЖЕННЯ.

Кваліфікаційна робота бакалавра присвячена розробці алгоритму паралельного стиснення відеопотоку в режимі реального часу. Актуальність теми зумовлена необхідністю обробки кадрів надвисокої чіткості із мінімальними затримками, що складно реалізувати лише ресурсами центрального процесора. Використання гетерогенних обчислень на базі GPU дозволяє паралелізувати найбільш трудомісткі етапи кодування, забезпечуючи високу швидкість обробки даних.

Метою роботи є проєктування та реалізація програмної системи з динамічним розподілом навантаження між CPU та GPU. У роботі застосовано технологію CUDA, впроваджено асинхронний конвеєр та оптимізовано передачу даних через системну шину PCIe за допомогою механізму Pinned Memory. Експериментально підтверджено суттєвий приріст частоти кадрів порівняно з традиційними методами при збереженні високої якості відео. Результати роботи можуть бути застосовані у стрімінгових сервісах та системах відеозв'язку.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

На сучасному етапі розвитку інформаційних технологій зростання роздільної здатності відео до 4K/8K і частоти кадрів збільшує обсяг даних, які потрібно обробляти в межах часу одного кадру. Така тенденція ставить перед розробниками програмного забезпечення складне завдання забезпечення ефективної обробки та стиснення відеопотоків у режимі реального часу, тобто в межах приблизно 16,7 мс на кадр для 60 кадрів/с. Головною технічною суперечністю є необхідність досягнення високого ступеня стиснення для економії пропускної здатності мереж при одночасному забезпеченні мінімальної затримки сигналу, що є критичним для систем інтерактивної взаємодії та живих трансляцій. Традиційні алгоритми кодування, що використовують виключно ресурси центральних процесорів, можуть не вкладатися в часовий бюджет кадру без апаратного прискорення. Це зумовлює високу актуальність пошуку нових архітектурних підходів, що базуються на гетерогенних обчисленнях, де найбільш трудомісткі математичні операції делегуються графічним прискорювачам. Використання паралельних обчислювальних потужностей GPU у поєднанні з гнучкістю керування CPU дозволяє подолати «вузькі місця» класичних методів обробки, проте вимагає розробки складних механізмів динамічного розподілу навантаження для уникнення простоїв обладнання.

Метою дипломної роботи є розробка та експериментальна перевірка прототипу паралельного конвеєра обробки відеокадрів із розподілом навантаження між центральним і графічним процесорами. Для досягнення поставленої мети передбачається вирішення низки науково-технічних задач, що охоплюють дослідження існуючих стандартів кодування, створення математичної моделі розподілу завдань на основі аналізу характеристик вхідних кадрів та оптимізацію протоколів обміну даними між оперативною та відеопам'яттю. Кінцевий результат спрямований на створення програмного

прототипу, здатного стабільно виконувати тестове обчислювальне навантаження під час обробки відеокадрів високої роздільної здатності.

Об’єктом дослідження виступає процес паралельної обробки та стиснення цифрового відеопотоку високої чіткості в умовах обмеженого бюджету часу. Предметом дослідження є методи, алгоритми та програмні засоби для реалізації динамічного розподілу обчислювальних задач між ядрами CPU та GPU в межах єдиного конвеєра стиснення. Наукова база роботи ґрунтується на використанні сучасних технологій паралельного програмування, таких як CUDA, та методів математичного моделювання для оцінки складності фрагментів зображення.
Елемент новизни роботи полягає у поєднанні попередньої оцінки складності фрагментів кадру з розподілом обчислень між CPU та GPU у прототипі конвеєра. Це дозволяє більш раціонально використовувати апаратну архітектуру GPU для масових паралельних розрахунків та CPU для складної ентропійної обробки, що забезпечує значне зростання продуктивності системи порівняно зі статичними методами розподілу. Практичне значення роботи визначається можливістю впровадження розробленого алгоритму у програмні комплекси для стрімінгових платформ, систем відеоконференцій та інструментів віддаленого керування обчислювальними ресурсами. Запропоновані рішення щодо використання закріпленої пам'яті та асинхронної передачі даних дозволяють суттєво знизити затримки при копіюванні інформації через системну шину PCIe, що відкриває шлях до обробки відео форматів 4K+ на апаратному забезпеченні середнього цінового сегмента.

1 АНАЛІЗ ПРОЦЕСУ ОБРОБКИ ВІДЕОДАНИХ ТА ВИЯВЛЕННЯ ПРОБЛЕМ ЕФЕКТИВНОГО ВИКОРИСТАННЯ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ

1.1 Аналіз принципів роботи та послідовності етапів стиснення відео

Розвиток мережевих технологій та постійне зростання роздільної здатності дисплеїв призводять до експоненціального збільшення обсягів відеоданих. Сучасні сервіси, такі як потокове відео, відеоконференції, хмарний геймінг та системи відеоспостереження, вимагають передачі зображення високої якості з мінімальними затримками. Нестиснений відеопотік у форматі високої чіткості вимагає величезної пропускної здатності каналів зв'язку, що на практиці реалізувати неможливо або економічно недоцільно. Тому процес стиснення відеоданих є важливим етапом у будь-якій системі передачі медіаконтенту [8, 39]. Основне завдання стиснення полягає у зменшенні обсягу даних за рахунок видалення надлишкової інформації з відеоряду при збереженні прийнятної для сприйняття якості зображення [8].

Процес стиснення відео, або кодування, базується на експлуатації двох основних типів надмірності: просторової та часової. Просторова надмірність пов'язана з тим, що сусідні пікселі в межах одного кадру часто мають схожі або однакові значення кольору та яскравості. Часова надмірність виникає через те, що сусідні кадри у відеопотоці зазвичай дуже схожі між собою, і змінюється лише невелика частина зображення, наприклад, рухомий об'єкт на статичному фоні. Виявлення та математичний опис цих змін вимагають виконання великої кількості обчислювальних операцій. Загальну структуру процесу перетворення та стиснення відеоданих можна зобразити графічно для кращого розуміння послідовності етапів (рисунок 1.1).

Рисунок 1.1 – Загальна схема процесу стиснення відеопотоку

Особливо складною ця задача стає, коли мова йде про роботу в реальному часі. У режимі реального часу система має жорсткі часові обмеження: кожен кадр повинен бути оброблений, стиснений і відправлений за частки секунди. Наприклад, для відео з частотою шістдесят кадрів на секунду час обробки одного кадру не повинен перевищувати шістнадцяти мілісекунд [9, 10]. Якщо система не встигає виконати стиснення за цей час, виникають пропуски кадрів, розсинхронізація звуку та відео, а також загальне відчуття затримки у користувача, що є неприпустимим для інтерактивних сервісів.

Традиційно завдання стиснення відео покладалося виключно на центральний процесор (CPU). Центральні процесори мають складну архітектуру,

яка чудово справляється з послідовними задачами, складними логічними розгалуженнями та управлінням загальним потоком програми. Однак, незважаючи на постійне збільшення кількості ядер і тактової частоти, центральні процесори стикаються з серйозними труднощами при обробці відео високої роздільної здатності у реальному часі. Це пов'язано з тим, що обробка відео – це здебільшого однотипні математичні операції, які застосовуються до мільйонів пікселів. Центральний процесор, маючи обмежену кількість ядер, просто не здатний обробити такий масив даних достатньо швидко без максимального завантаження, що призводить до перегріву системи та неможливості виконання інших фонових задач операційної системи.

З іншого боку, графічні процесори мають принципово іншу архітектуру. Вони складаються з тисяч відносно простих ядер, які створені спеціально для паралельного виконання великої кількості однакових інструкцій над різними даними [24, 30]. Ця парадигма обчислень добре підходить для таких етапів стиснення відео, як оцінка руху (пошук схожих блоків пікселів між кадрами) або дискретне косинусне перетворення. Проте графічний процесор не є універсальним рішенням. Існують етапи алгоритму кодування, наприклад, ентропійне кодування, які є строго послідовними. Результат кожної наступної операції залежить від попередньої, тому розпаралелити цей процес на тисячі ядер графічного процесора складно або взагалі неможливо.

Аналіз існуючих рішень показує, що більшість сучасних систем використовують або суто програмне кодування на базі CPU (яке дає високу якість, але працює повільно), або апаратні кодувальники, вбудовані у відеокарти (які працюють дуже швидко, але мають меншу гнучкість налаштувань алгоритму стиснення). Головна проблема полягає в тому, що при використанні лише одного типу обчислювача інший часто простоює. Наприклад, якщо все навантаження передати на GPU, центральний процесор може бути завантажений лише на кілька відсотків, хоча його ресурси могли б бути використані для прискорення

послідовних частин алгоритму кодування. І навпаки, при програмному кодуванні потужності графічної карти залишаються незадіяними.

Для наочності різницю в ефективності виконання окремих етапів стиснення різними типами процесорів зведено у порівняльній таблиці 1.1.

Таблиця 1.1 – Порівняння ефективності виконання етапів стиснення на CPU та GPU

Етап алгоритму стиснення	Ефективність на CPU	Ефективність на GPU	Характеристика обчислювального процесу
Перетворення колірного простору	Низька	Дуже висока	Незалежні попіксельні операції, придатні для масового розпаралелювання.
Оцінка руху	Низька	Висока	Інтенсивне порівняння матриць пікселів, добре масштабується на ядра графічного процесора.
Дискретне косинусне перетворення	Середня	Висока	Базується на матричних множеннях, під які GPU оптимізовані апаратно.
Ентропійне кодування	Висока	Дуже низька	Строго послідовний процес пакування даних, розпаралелювання неефективне.

Кінець таблиці 1.1

Управління бітрейтом та загальна логіка	Висока	Низька	Складні логічні розгалуження та аналіз статистики, які потребують складної архітектури CPU.

Виявлені недоліки існуючих підходів формують ключову проблему предметної області: відсутність гнучкого механізму розподілу обчислювального навантаження між центральним та графічним процесорами під час стиснення відео в реальному часі. Виникає так зване "вузьке місце", коли продуктивність всієї системи обмежується найслабшим або найбільш перевантаженим компонентом, у той час як інші обчислювальні ресурси системи не використовуються ефективно.

Вирішення цієї проблеми вимагає розробки такого паралельного алгоритму, який би міг динамічно аналізувати поточне завдання та направляти блоки даних на той процесор, який впорається з ними найкраще. Таким чином, з аналізу предметної області випливають наступні завдання для подальшого дослідження: необхідно детально розібрати внутрішню структуру сучасних алгоритмів стиснення відеопотоку, виділити етапи, що піддаються розпаралелюванню, і ті, що вимагають строгої послідовності дій. Після цього потрібно розробити методологію балансування навантаження, яка б дозволила одночасно задіяти сильні сторони як CPU, так і GPU, мінімізуючи при цьому затримки на передачу даних між оперативною та відеопам'яттю, що є важливим для режиму реального часу.

1.2 Порівняння швидкості та якості роботи різних типів процесорів

Для розробки ефективного алгоритму розподілу навантаження необхідно детально проаналізувати існуючі підходи до вирішення задачі стиснення відеопотоку в реальному часі. На сьогоднішній день архітектура систем кодування відеоданих концептуально поділяється на три основні категорії: суто програмні рішення на базі центрального процесора, апаратні рішення з використанням спеціалізованих блоків графічних прискорювачів та спроби створення гетерогенних систем. Кожен з цих підходів має свої виражені переваги та критичні обмеження, які визначають сферу їх застосування.

Програмне кодування, яке виконується виключно силами центрального процесора, історично є найстарішим та найбільш дослідженим методом. Основною перевагою цього підходу є його абсолютна гнучкість. Оскільки весь процес стиснення реалізований у вигляді програмного коду, розробники мають повний доступ до всіх етапів алгоритму і можуть впроваджувати найсучасніші математичні моделі оптимізації. Програмні кодувальники зазвичай забезпечують гнучкіші налаштування якості зображення та бітрейту [37, 39]. Це досягається завдяки глибокому та вичерпному аналізу кожного кадру, застосуванню складних алгоритмів оцінки руху з субпіксельною точністю та багатокроковим механізмам прийняття рішень щодо вибору оптимального способу стиснення для кожного окремого блоку зображення.

Проте, головний недолік програмного підходу полягає в його колосальній обчислювальній складності. У режимі реального часу центральний процесор просто не встигає виконати весь цей обсяг складних математичних операцій, особливо коли мова йде про відео у форматах високої чіткості. Щоб забезпечити необхідну частоту кадрів без затримок, користувачі змушені знижувати налаштування якості кодування. Це призводить до того, що алгоритм пропускає етапи глибокого аналізу, замінюючи їх спрощеними евристичними методами. Як наслідок, при однаковому бітрейті якість зображення суттєво падає, або ж для

збереження якості доводиться значно збільшувати обсяг переданих даних, що перевантажує мережеві канали зв'язку. Крім того, повне завантаження центрального процесора задачею стиснення відео унеможливлює нормальну роботу інших програм на цьому ж комп'ютері, що є критичним недоліком для таких сценаріїв, як стрімінг відеоігор.

Апаратне кодування з використанням графічних процесорів є відповіддю індустрії на обмеження центральних процесорів. Сучасні відеокарти оснащуються спеціалізованими апаратними блоками, які фізично розпаяні на кристалі чіпа і призначені виключно для виконання операцій кодування та декодування відео [8, 19]. Основною перевагою такого рішення є нижча затримка та менше навантаження на CPU. Апаратний кодувальник працює незалежно від основних обчислювальних блоків відеокарти та центрального процесора, знімаючи з них практично все навантаження. Це дозволяє системі легко записувати або транслювати відео високої роздільної здатності без відчутного падіння загальної продуктивності комп'ютера. Порівняння залежності якості зображення від бітрейту для програмного та апаратного підходів наведено на рисунку 1.2.

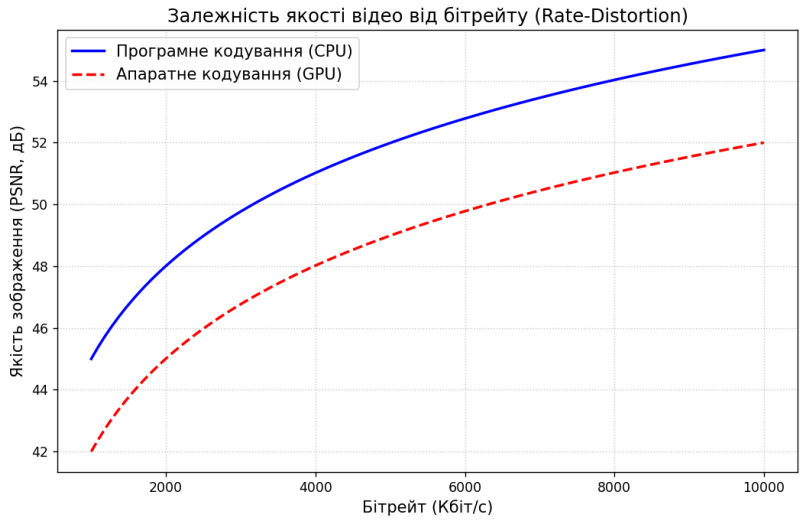


Рисунок 1.2 – Порівняння кривих залежності якості від бітрейту для програмного та апаратного підходів

Незважаючи на високу швидкість, апаратні рішення мають суттєві вади у площині якості та гнучкості. Оскільки алгоритм жорстко «зашитий» у кремній, його неможливо оновити або змінити для підтримки нових, більш ефективних методів стиснення без фізичної заміни відеокарти. Крім того, через жорсткі обмеження площі кристала та енергоспоживання, апаратні блоки використовують дещо спрощені алгоритми оцінки руху та управління бітрейтом порівняно з програмними аналогами. Тому, традиційно, для досягнення такої ж візуальної якості, як при якісному програмному кодуванні, апаратним кодувальникам потрібен вищий бітрейт.

Окремої уваги заслуговують гетерогенні підходи, які намагаються поєднати обчислювальні потужності центрального та графічного процесорів. Існують рішення, де розробники пишуть власні алгоритми кодування з використанням технологій паралельних обчислень загального призначення на графічних картах. У таких системах робиться спроба розбити алгоритм стиснення на частини. Зазвичай, найбільш масові та паралельні операції, такі як пошук векторів руху або перетворення кольорів, передаються на обчислювальні ядра графічного процесора, а логіка управління, ентропійне кодування та фінальна збірка потоку залишаються на центральному процесорі.

Однак, аналіз таких гібридних рішень виявляє дві проблеми, які заважають їх широкому впровадженню. Першою проблемою є пропускна здатність системної шини. Для того, щоб графічний процесор міг обробити кадр, цей кадр спочатку потрібно скопіювати з оперативної пам'яті комп'ютера у відеопам'ять, а після обробки – повернути результати назад. Якщо обсяг даних великий, а алгоритм вимагає частого обміну проміжними результатами, час, витрачений на передачу даних по шині, може повністю нівелювати виграш у часі від швидких паралельних обчислень на відеокарті [1, 11].

Другою, ще більш складною проблемою існуючих гетерогенних рішень є статичний розподіл навантаження. У більшості таких систем заздалегідь жорстко визначено, які функції виконує CPU, а які – GPU. Проте відеопотік за своєю

природою є надзвичайно динамічним. Сцени з високою динамікою та великою кількістю дрібних деталей вимагають величезних зусиль для оцінки руху, тоді як статичні сцени з плавними градієнтами генерують складні потоки даних для ентропійного кодування. При статичному розподілі виникають ситуації, коли один з процесорів закінчує свою частину роботи і простоює в очікуванні, поки інший намагається впоратися з піковим навантаженням.

Таким чином, порівняльний аналіз показує, що жодне з існуючих рішень не задовольняє повною мірою вимогам до ефективної системи кодування в реальному часі. Програмні підходи надто повільні, апаратні – недостатньо гнучкі та вимагають високого бітрейту, а існуючі гібридні спроби страждають від «вузького місця» системної шини та неефективного, жорстко заданого статичного балансування навантаження. Це підтверджує наукову та практичну необхідність розробки саме динамічного паралельного алгоритму, який міг би адаптуватися до поточного вмісту відеокадру та поточного завантаження обчислювальних вузлів системи.

1.3 Аналіз способів поділу кадрів на частини та передачі даних у системі

Вирішення складної проблеми неефективного використання обчислювальних ресурсів під час стиснення відео вимагає розробки комплексної методології. Основна ідея запропонованого підходу полягає у відмові від жорсткого закріплення конкретних функцій кодека за центральним або графічним процесором. Натомість пропонується створити динамічну архітектуру, яка здатна адаптуватися до поточного стану системи та характеристик самого відеопотоку. Першим і найважливішим методологічним кроком є визначення рівня деталізації, на якому буде відбуватися розпаралелювання даних, що в науковій літературі часто називають гранулярністю завдань.

Існує кілька можливих рівнів поділу відеоданих для паралельної обробки. Найбільш очевидним є поділ на рівні цілих кадрів, коли перший кадр відправляється на обробку центральному процесору, а другий одночасно обробляється на відеокарті. Незважаючи на простоту реалізації, такий підхід є неприйнятним для систем реального часу. Це пов'язано з тим, що сучасні алгоритми стиснення використовують інформацію з попередніх та наступних кадрів для пошуку руху. Якщо сусідні кадри обробляються на різних пристроях, виникає потреба у постійному пересиланні величезних масивів нестиснених даних між оперативною та відеопам'яттю, що повністю блокує системну шину і створює гігантські затримки.

З іншого боку, поділ на рівні найменших елементів зображення, наприклад, окремих макроблоків розміром шістнадцять на шістнадцять пікселів, створює надмірне навантаження на систему управління потоками. Диспетчер завдань витрачатиме більше часу на розподіл цих мікроскопічних завдань, ніж процесори на їх безпосереднє виконання. Тому оптимальним методологічним рішенням є використання паралелізму на рівні так званих «слайсів» [7, 20]. Слайс є незалежною просторовою областю кадру, зазвичай горизонтальну смугу з кількох рядів макроблоків, яка може бути стиснена незалежно від інших частин того ж кадру (рис. 1.3).

Рисунок 1.3 – Поділ відеокадру на незалежні фрагменти для паралельної обробки

Такий просторовий поділ дозволяє розбити один важкий кадр на кілька відносно великих, але незалежних шматків роботи. Після такого поділу в роботу вступає наступний методологічний компонент системи – динамічний диспетчер завдань. Диспетчер є центральним вузлом управління, який працює на центральному процесорі і відповідає за прийняття рішень щодо маршрутизації даних. Замість того, щоб наперед знати, хто і що буде робити, диспетчер формує загальну чергу завдань. Кожне завдання — це окремий слайс відео, який потрібно стиснути.

Для реалізації цього механізму створюється дві робочі черги: одна призначена для потоків центрального процесора, інша для графічного прискорювача. Процесори виступають у ролі робітників, які звертаються до диспетчера за новою порцією роботи щойно закінчують попередню. Така архітектура зменшує ймовірність простою обчислювальних вузлів, поки в загальній черзі залишаються необроблені фрагменти відеокадру (рис. 1.4).

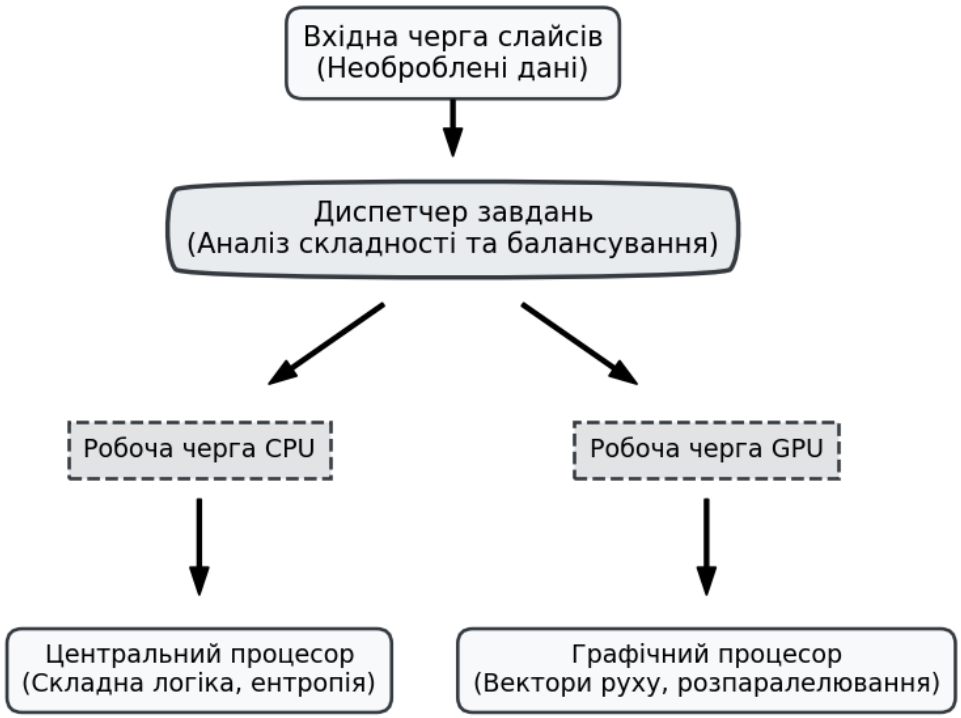


Рисунок 1.4 – Архітектура диспетчера завдань та розподілу черг між обчислювальними вузлами

Однак проста роздача завдань «першому вільному» не є достатньо ефективною, оскільки слайси можуть суттєво відрізнятися за складністю. Наприклад, слайс, що містить чисте блакитне небо, стискається дуже швидко, тоді як слайс із рухомим натовпом людей вимагає значних обчислень для пошуку векторів руху. Тому методологія передбачає впровадження етапу попереднього аналізу складності (пре-аналізу). Перед тим як відправити слайс у чергу, диспетчер проводить його швидку оцінку. Цей процес включає розрахунок просторової дисперсії (наскільки багато дрібних деталей у кадрі) [33] та аналіз різниці з попереднім кадром (наскільки інтенсивний рух відбувається у цій зоні).

На основі результатів пре-аналізу диспетчер призначає кожному завданню певну умовну «вагу» складності. Далі застосовується евристичний алгоритм балансування. Якщо завдання отримує високу вагу складності через інтенсивний рух, диспетчер з більшою ймовірністю направить його у чергу графічного процесора, оскільки його тисячі ядер ідеально підходять для масового пошуку векторів руху. Якщо ж слайс відносно статичний, але має складну текстуру, його ефективніше відправити на вільні ядра центрального процесора, який краще впорається зі складною логікою ентропійного кодування цього фрагмента.

Ще одним важливим методологічним підходом є застосування принципу конвеєризації (pipelining) для приховування затримок передачі даних. Як було з'ясовано під час аналізу предметної області, передача даних по шині PCI Express є найвужчим місцем гібридних систем. Щоб вирішити цю проблему, процес обробки розбивається на етапи, які виконуються з перекриттям у часі. Наша методологія передбачає створення триступеневого конвеєра: передача даних до відеокарти, безпосередні обчислення та повернення результатів у системну пам'ять (рис. 1.5).

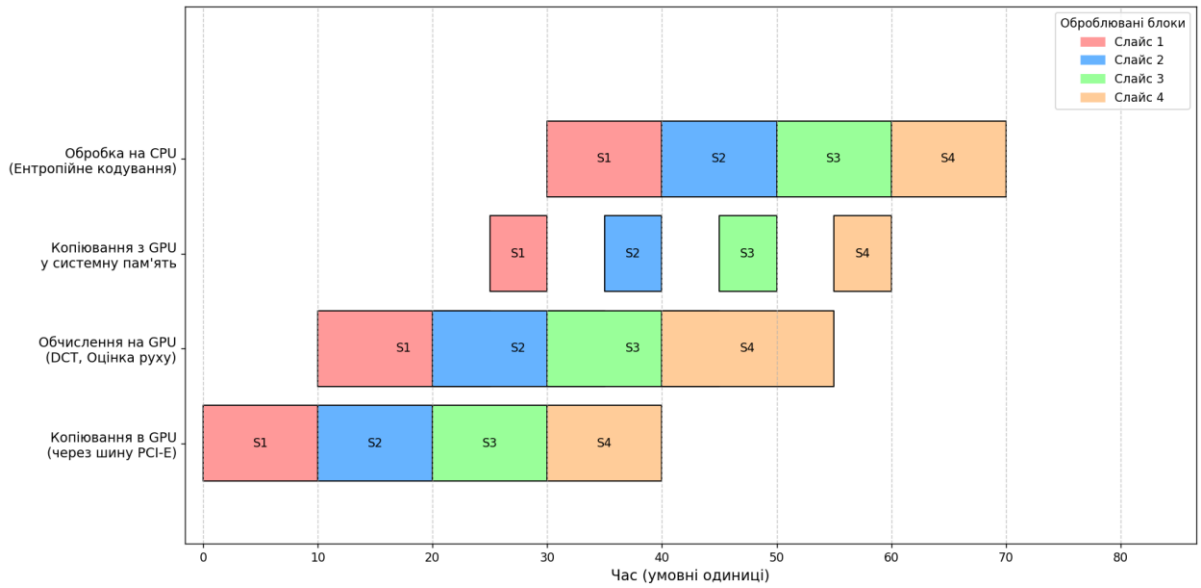


Рисунок 1.5 – Схема конвеєризації обчислень та приховування затримок передачі даних

Суть конвеєризації полягає в тому, що ці процеси відбуваються асинхронно. Поки графічний процесор виконує математичні операції над слайсом номер один, контролер пам'яті вже передає по шині дані для слайсу номер два, а центральний процесор у цей же момент пакує готові результати обробки нульового слайсу у фінальний відеопотік. Таким чином, частина часу копіювання може перекриватися з корисними обчисленнями. Система не чекає завершення повного циклу для одного фрагмента, щоб почати працювати з наступним.

Для максимальної оптимізації процесу передачі даних також пропонується використовувати механізм закріпленої оперативної пам'яті. Зазвичай, коли програма просить операційну систему виділити пам'ять, ця пам'ять може бути переміщена на жорсткий диск, якщо вільної оперативної пам'яті не вистачає. Графічний процесор не може безпосередньо читати таку стандартну пам'ять, тому драйвер відеокарти змушений спочатку створювати тимчасовий буфер, копіювати дані туди силами центрального процесора, і лише потім відправляти їх на відеокарту. Використання закріпленої пам'яті забороняє операційній системі переміщувати ці дані. Це дозволяє контролеру прямого доступу до

пам'яті, який вбудований у відеокарту, самостійно забирати відеокадри [12, 13] з оперативної пам'яті без жодної участі центрального процесора. Такий підхід значно знижує навантаження на процесор і підвищує загальну пропускну здатність шини, що є фундаментом для стабільної роботи алгоритму стиснення у реальному часі.

В сукупності описані методологічні підходи: поділ кадру на просторові слайси, динамічна диспетчеризація на основі пре-аналізу складності, асинхронна конвеєризація та оптимізація доступу до пам'яті формують цілісну теоретичну базу. Ця база дозволяє подолати обмеження статичного розподілу та створити гнучку систему, здатну витиснути максимум продуктивності з наявного апаратного забезпечення комп'ютера.

1.4 Постановка задачі розробки алгоритму з гнучким розподілом роботи

Виходячи з детального аналізу предметної області стиснення відеоданих, огляду існуючих програмних, апаратних та гібридних систем кодування, їхніх переваг та суттєвих обмежень, а також враховуючи сучасні тенденції розвитку гетерогенних обчислювальних архітектур, можна сформулювати ключові науково-технічні завдання. Ці завдання стосуються подальших досліджень та безпосередньої розробки програмної частини паралельного алгоритму з динамічним розподілом навантаження для обробки відео в реальному часі.

Насамперед необхідно провести глибокий порівняльний аналіз та кількісну оцінку ефективності існуючих методів балансування завдань між центральним та графічним процесорами. Потрібно систематизувати їхню обчислювальну складність, затримки при передачі даних та вплив на кінцеву якість зображення при різних рівнях стиснення. Також слід дослідити та оптимізувати математичні підходи до попередньої оцінки складності відеокадрів, розробивши швидкі метрики для визначення інтенсивності руху та

просторової деталізації, щоб алгоритм міг приймати обґрунтовані рішення щодо маршрутизації масивів пікселів.

Головним напрямком є розробка та вдосконалення нового, більш гнучкого паралельного алгоритму кодування відеопотоку. Особливу увагу слід приділити створенню інтелектуального диспетчера завдань, здатного динамічно розбивати кадри на незалежні просторові фрагменти та призначати їх тому обчислювальному вузлу, який впорається з ними найефективніше в поточний момент часу. При цьому потрібно поглиблено дослідити специфічні виклики, пов'язані з обмеженою пропускною здатністю системної шини PCI Express, та розробити механізм асинхронної конвеєризації, який дозволить приховати затримки на копіювання даних у відеопам'ять за рахунок паралельного виконання математичних перетворень.

Важливо розробити універсальну програмну архітектуру алгоритму, яка забезпечуватиме надійну синхронізацію багатопотокових обчислень та мінімізуватиме накладні витрати на рівні операційної системи. Необхідно дослідити та впровадити ефективні механізми роботи з оперативною пам'яттю, зокрема використання закріплених буферів для прямого доступу відеокарти. Також варто запропонувати дієві програмні інтерфейси для можливої інтеграції розробленого модуля балансування з поширеними відкритими бібліотеками кодування медіаданих, що значно полегшить його подальше практичне застосування.

На основі проведених досліджень слід сформулювати практичні рекомендації щодо налаштування гетерогенних систем кодування залежно від конкретних апаратних характеристик комп'ютера та жорстких вимог до трансляції. Успішне вирішення цих завдань дозволить значно підвищити загальну продуктивність систем стиснення відео в реальному часі, практично усунути явище пропуску кадрів та забезпечити оптимальне використання всіх наявних обчислювальних ресурсів без критичного перевантаження окремих компонентів системи.

1.5 Висновки до першого розділу

У першому розділі було здійснено комплексний аналіз проблеми стиснення відеоданих високої роздільної здатності у режимі реального часу. Дослідження теоретичних засад показало, що сучасні вимоги до потокового мовлення створюють значне навантаження на обчислювальні ресурси комп'ютерних систем. Процес кодування вимагає виконання величезної кількості математичних операцій для зменшення просторової та часової надмірності відеоряду. Було встановлено, що традиційні програмні кодеки, які спираються виключно на потужності центрального процесора, не здатні забезпечити необхідну частоту кадрів без втрати якості або критичного перегріву системи. З іншого боку, використання виключно апаратних блоків графічних прискорювачів хоч і вирішує проблему швидкодії, проте значно обмежує гнучкість налаштувань та призводить до погіршення візуальної якості зображення при аналогічних показниках бітрейту.

Аналіз існуючих архітектур довів необхідність переходу до змішаних обчислювальних систем, де одночасно задіюються сильні сторони різних типів процесорів. Однак огляд наявних рішень у цій сфері виявив низку суттєвих недоліків. Зокрема, більшість існуючих алгоритмів використовують жорстко закріплений розподіл функцій, що призводить до неефективного використання ресурсів при зміні динаміки відеокадрів. Окрім того, серйозною перешкодою для досягнення високої продуктивності виявилася обмежена пропускна здатність системної шини передачі даних. Постійне копіювання масивів пікселів між оперативною та відеопам'яттю створює затримки, які нівелюють переваги швидких паралельних обчислень.

Для подолання цих обмежень було досліджено нові методологічні підходи до організації обчислювального процесу. Визначено, що найвищої ефективності можна досягти шляхом просторового поділу кожного відеокадру на незалежні фрагменти. Це дозволяє організувати паралельну обробку даних на рівні

окремих частин зображення. Важливим етапом дослідження стало обґрунтування необхідності впровадження інтелектуального диспетчера завдань. Такий механізм має попередньо оцінювати складність кожного фрагмента та динамічно направляти його на той обчислювальний вузол, який впорається з роботою найшвидше. Для вирішення проблеми затримок при передачі даних було запропоновано використання асинхронної конвеєризації та механізмів прямого доступу до пам'яті.

На основі проведеного теоретичного аналізу та виявлених проблем було сформульовано основну постановку задачі кваліфікаційної роботи. Вона полягає у розробці та програмній реалізації нового алгоритму паралельного стиснення відеопотоку. Створювана система повинна мати механізм адаптивного балансування навантаження між центральним та графічним процесорами, який здатний підлаштовуватися під поточні характеристики відеоряду. Успішне виконання поставлених завдань дозволить оптимізувати використання наявних апаратних ресурсів, усунути затримки трансляції та підвищити загальну ефективність систем обробки відеоданих у режимі реального часу.

2 ПРОЄКТУВАННЯ СИСТЕМИ ПАРАЛЕЛЬНОГО СТИСНЕННЯ ВІДЕОПОТОКУ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу

Вибір технічної бази для практичної реалізації складної системи паралельного стиснення відеопотоку є важливим етапом проєктування, що визначає верхню межу продуктивності всієї розробки. Від цього стратегічного рішення безпосередньо залежить здатність майбутнього програмного комплексу стабільно обробляти масивні потоки нестиснених даних без виникнення накопичувальних затримок або критичних помилок переповнення вхідних буферів. Сучасні галузеві стандарти цифрового медіаконтенту, зокрема перехід до надвисоких роздільних здатностей форматів 4K та 8K, висувають безпрецедентні вимоги до обчислювальної потужності апаратного заліза та пропускної здатності підсистеми пам’яті. Будь-яка невідповідність характеристик обраного обладнання обсягам вхідної інформації неминуче призводить до деградації якості стисненого відео або повної неможливості функціонування системи в умовах жорсткого реального часу, що робить апаратний аналіз невіддільною частиною загального процесу розробки.

Головним завданням під час підбору оптимальної апаратної конфігурації є пошук раціонального балансу між універсальною логічною гнучкістю центрального процесора (CPU) та масовим обчислювальним паралелізмом графічного прискорювача (GPU). У межах запропонованої гетерогенної архітектури центральний процесор виконує роль головного інтелектуального координатора та диспетчера всіх внутрішніх обчислювальних процесів. Він відповідає за високорівневе керування логічними розгалуженнями, здійснює динамічний менеджмент пам'яті, керує чергами завдань та реалізує етап фінального ентропійного пакування стиснених даних у вихідний контейнер згідно зі специфікаціями обраного кодека. Для ефективного виконання таких

багатопотокових задач необхідно обирати сучасні багатоядерні рішення з високою тактовою частотою та розширеною кеш-пам'яттю, що підтримують набори векторних інструкцій типів AVX2 або AVX-512. Це дозволяє суттєво прискорити виконання тих етапів загального алгоритму, які за своєю природою потребують строгої послідовності дій, складної умовної логіки та миттєвого прийняття рішень на основі статистичних даних кадру.

Графічний процесор у запропонованій системі бере на себе виконання найбільш ресурсомістких із математичної точки зору етапів, де домінує принцип однотипної обробки значних масивів даних. Архітектура GPU, що базується на концепції SIMT (Single Instruction, Multiple Threads) та містить тисячі дрібних спеціалізованих обчислювальних ядер, дозволяє одночасно маніпулювати величезною кількістю пікселів у паралельному режимі [11, 24]. Це забезпечує ефективне виконання операцій дискретного косинусного перетворення, квантування та, що найважливіше, масового пошуку векторів руху в межах заданих вікон пошуку. Така вузька спеціалізація графічного заліза забезпечує обчислювальну пропускну здатність, що на кілька порядків перевищує можливості класичних CPU при роботі з ідентичними матричними структурами. Саме завдяки такому дворівневому розподілу функцій вдається досягти необхідної швидкодії при збереженні високої візуальної якості кодування та мінімізації артефактів стиснення. Загальний принцип ієрархічного обміну даними, логіку взаємодії та розподіл обчислювального навантаження між центральним та графічним процесорами наочно відображено на рис. 2.1.

Рисунок 2.1 – Схема взаємодії апаратних компонентів системи

Важливим фактором під час проєктування та подальшої практичної реалізації системи є пропускна здатність системної шини, яка виступає головною сполучною ланкою між усіма апаратними вузлами. Саме цей компонент найчастіше стає основною перешкодою при спробі обробки відеопотоків високої роздільної здатності у форматах 4K або 8K у режимі реального часу. При роботі з такими масивами даних обсяг інформації, що передається щосекунди, зростає експоненціально, що вимагає максимально швидкої реакції від каналів зв'язку.

Використання сучасних швидкісних інтерфейсів, таких як останні покоління PCI Express, дозволяє мінімізувати критичні затримки під час постійної передачі кадрів між оперативною та відеопам'яттю. Без належної швидкості шини навіть найпотужніші обчислювальні пристрої будуть змушені тривалий час простоювати в очікуванні надходження нових порцій даних. Це неминуче призводить до розсинхронізації відеоряду та появи відчутних пауз, що є неприпустимим для сучасних систем трансляції.

Ефективність апаратно-програмної реалізації розробленого алгоритму безпосередньо залежить від архітектурних особливостей обраної обчислювальної платформи. Для забезпечення стабільної роботи системи важливим є впровадження та підтримка механізмів прямого доступу до пам'яті. Це технічне рішення дозволяє графічному прискорювачу здійснювати зчитування та запис необхідної інформації самостійно, без постійного втручання з боку операційної системи чи центрального процесора. Такий підхід суттєво підвищує загальну швидкість обробки інформації та дозволяє повністю усунути зайве навантаження на основні обчислювальні ядра. В результаті вивільнені потужності можуть бути перенаправлені на виконання складної логіки керування або завдання диспетчеризації потоків. Така синергія апаратних засобів та програмних протоколів робить систему стійкою до значних перевантажень та дозволяє підтримувати високу якість кодування протягом тривалого часу.

Окрему увагу слід приділити не лише обсягу, а й швидкісним характеристикам оперативної пам'яті. Оскільки розроблений алгоритм передбачає одночасну роботу з кількома кадрами для проведення точного аналізу векторів руху, система повинна мати значний запас ресурсу для зберігання численних тимчасових буферів. Недостатня швидкість обміну даними з пам'яттю може спричинити ефект затримки, коли обчислювальні пристрої не отримують вчасно наступні фрагменти зображення для обробки. Висока продуктивність модулів пам'яті забезпечує безперебійність та стабільність роботи всієї програми навіть в умовах максимального навантаження на залізо.

Весь складний процес переміщення даних між різними рівнями та вузлами пам'яті обчислювальної системи наочно зображено на рисунку 2.2.

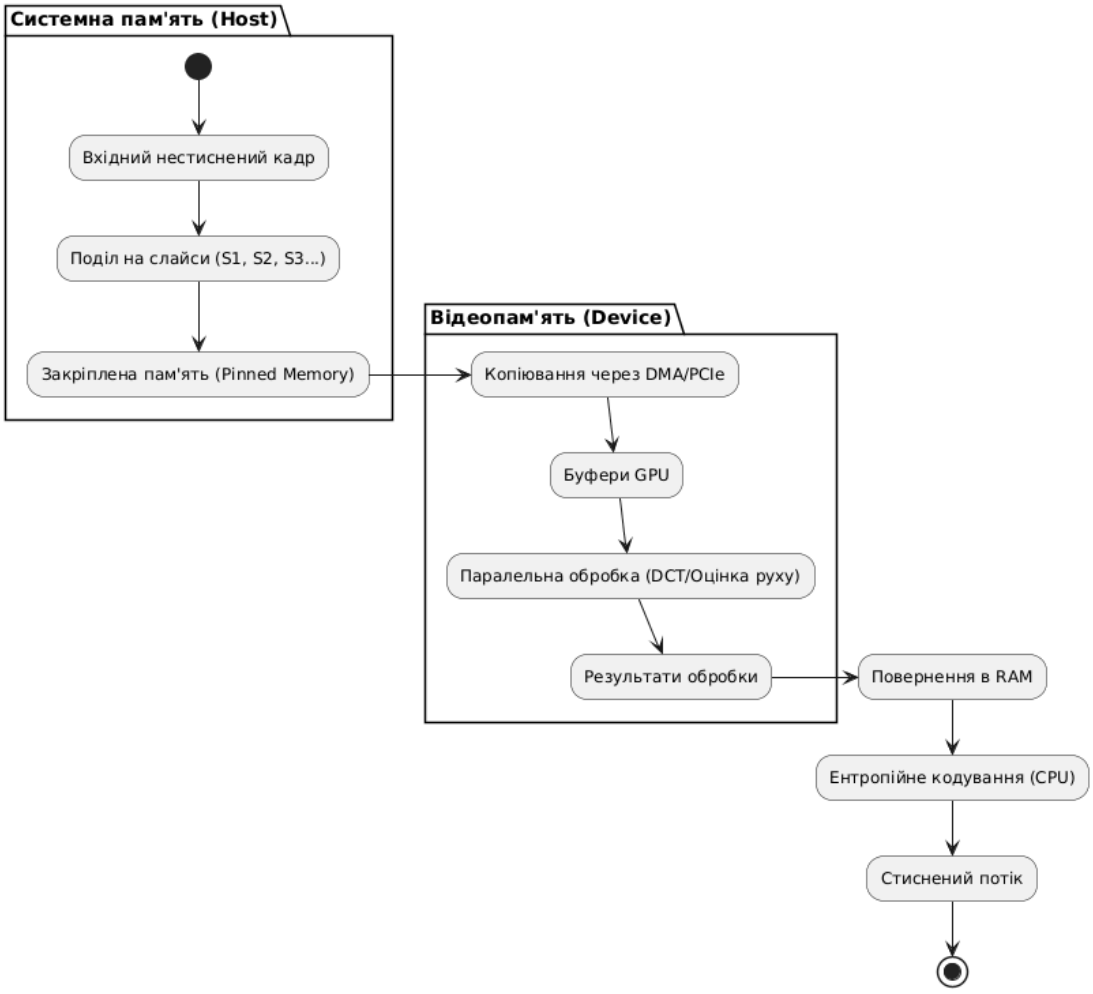


Рисунок 2.2 – Розподіл потоків даних у пам’яті обчислювальної системи

Надійність функціонування системи також залежить від фізичних умов роботи обладнання. Постійне кодування відео завантажує всі вузли комп’ютера на межі можливостей. Це вимагає використання корпусів із продуманою вентиляцією для запобігання перегріву та зниженню швидкості роботи компонентів. Вибір програмних засобів базується на необхідності отримати повний контроль над залізом. Мова програмування C++ є кращим рішенням для таких задач. Вона забезпечує найвищу швидкість виконання коду та дає можливість програмісту самостійно керувати ресурсами системи. Для розробки доцільно використовувати професійне середовище Visual Studio Code. Воно

надає зручні інструменти для налагодження багатопотокових програм. Такий вибір дозволяє інтегрувати різні технології в межах одного проєкту. Для написання коду під графічну карту необхідно застосовувати спеціалізовані інтерфейси. Технологія CUDA дозволяє створювати ядра для паралельного виконання обчислень на ядрах відеокарти [17, 24]. Це забезпечує гнучкість, якої часто не вистачає стандартним апаратним кодувальникам, а повну структуру програмного стеку системи показано на рисунку 2.3.

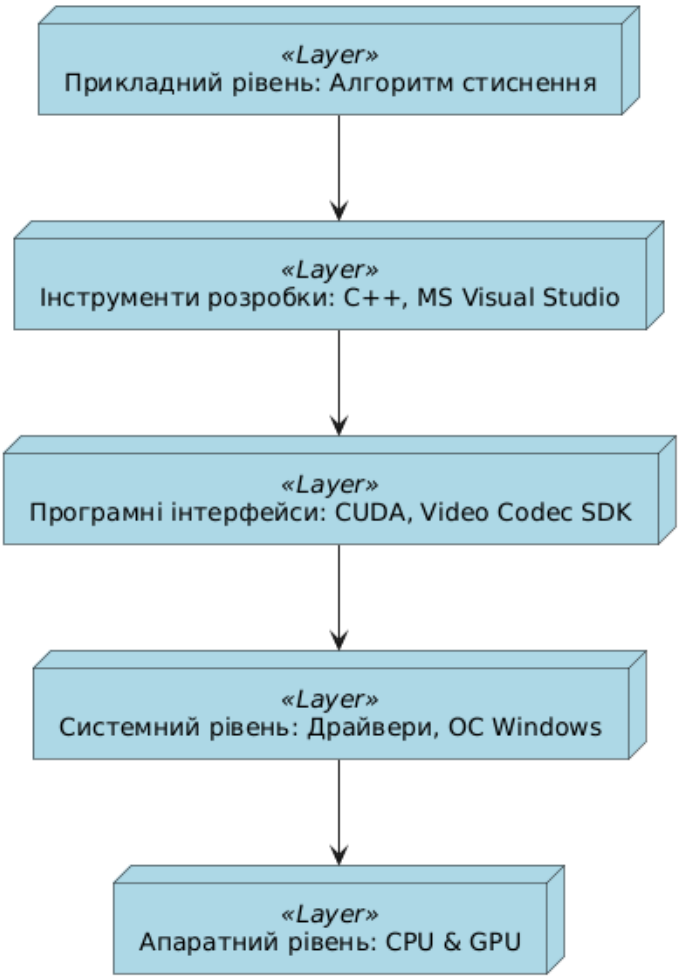


Рисунок 2.3 – Складові програмного інструментарію розробки

Для підвищення надійності системи варто залучати перевірені відкриті бібліотеки. Це дозволяє зосередити зусилля саме на розробці власного механізму розподілу навантаження між пристроями. Використання ліцензійних засобів

розробки гарантує стабільність та отримання важливих оновлень. Запропонований комплексний підхід до вибору бази створює умови для реалізації ефективного алгоритму. Тільки правильне поєднання потужного заліза та оптимізованого коду дозволяє досягти високої якості відео. Це є фундаментом для стабільної трансляції контенту в реальному часі.

2.2 Розробка функціональної схеми та загального алгоритму роботи паралельної системи

Побудова ефективної системи паралельного стиснення вимагає чіткого визначення взаємодії між усіма її компонентами на рівні функціональної схеми. Головна мета розробки такої схеми полягає у забезпеченні безперервного потоку даних від моменту надходження нестисненого відео до формування фінального бітового потоку. Основою запропонованої структури є модульний підхід, де кожен блок виконує специфічну задачу, а їхня злагоджена робота координується центральним вузлом управління. Такий підхід дозволяє відокремити етапи підготовки даних від безпосередніх обчислень, що є важливим для мінімізації простоїв обладнання.

Процес обробки розпочинається із надходження вхідних відеокадрів у системний буфер, де відбувається їхнє попереднє розбиття на окремі фрагменти. Використання концепції слайсів дозволяє перетворити один цілісний і важкий масив пікселів на кілька незалежних об’єктів для паралельної обробки. Це створює фундамент для подальшого розподілу навантаження, оскільки кожен такий фрагмент може бути стиснений без звернення до сусідніх областей у межах одного кадру. Такий підхід суттєво спрощує логіку управління потоками та дозволяє задіяти велику кількість обчислювальних ядер одночасно.

Після поділу кадру фрагменти потрапляють до модуля попереднього аналізу, де здійснюється оцінка їхньої складності. Цей етап є ключовим для інтелектуального керування ресурсами, адже він дозволяє визначити характер

зображення у кожній зоні. Для слайса s обчислюються дисперсія яскравості D_s = (1/N)Σ(Y_i - Ȳ)^2 та середня міжкадрова різниця M_s = (1/N)Σ|Y_t(i)-Y_{t-1}(i)|; умовна вага фрагмента може визначатися як W_s = αD_s + βM_s. На основі цих показників диспетчер завдань приймає рішення щодо маршрутизації даних. Фрагменти з інтенсивним рухом направляються до черги графічного процесора, тоді як статичні області із детальною текстурою передаються на вільні ядра центрального процесора. Загальний вигляд функціональної структури системи та логіку взаємодії модулів зображено на рисунку 2.4.

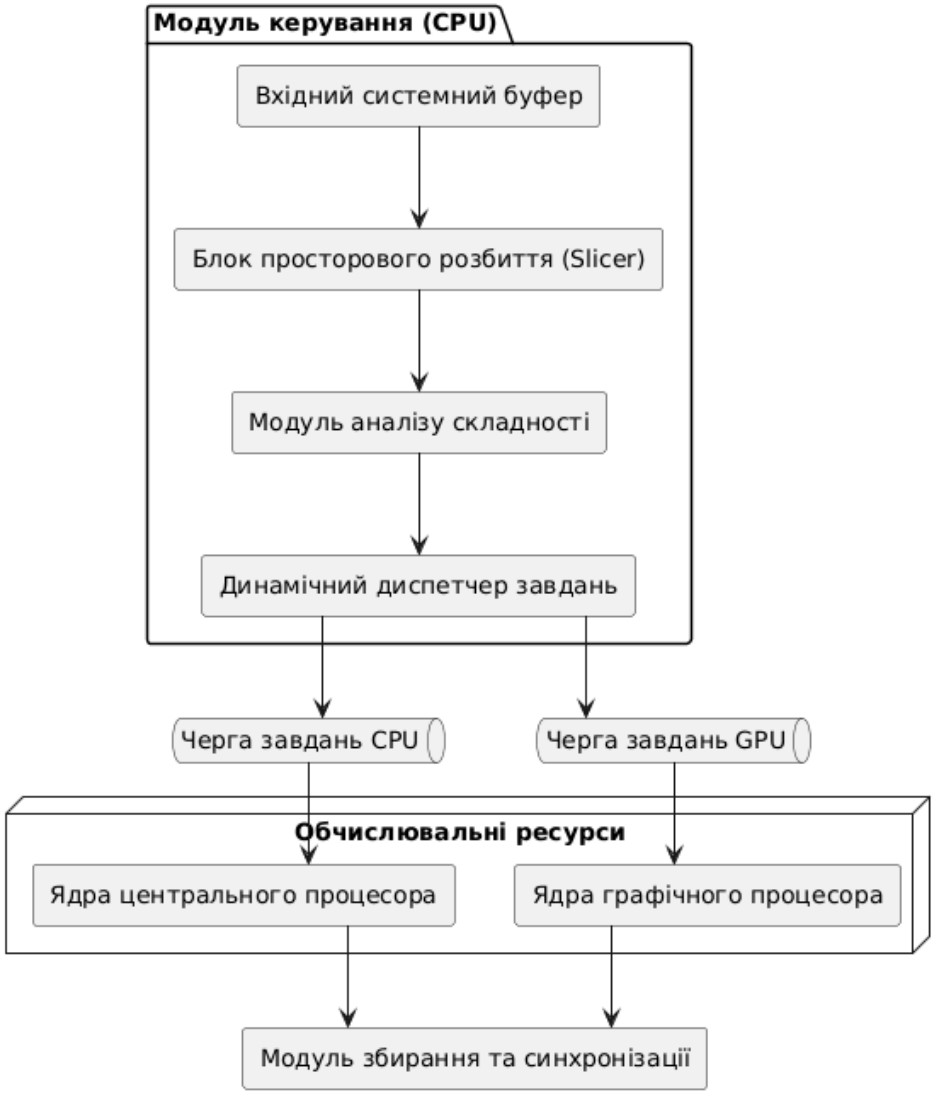


Рисунок 2.4 – Функціональна схема системи паралельного стиснення відеопотоку

Диспетчер завдань функціонує як центральний інтелектуальний регулятор всієї обчислювальної системи, забезпечуючи безперервний моніторинг стану черг обробки та наявних ресурсів. Основна складність його роботи полягає у необхідності врахування не лише поточної завантаженості центрального та графічного процесорів, але й прогнозованого часу виконання конкретного завдання. Замість примітивного розподілу за принципом черговості, диспетчер застосовує аналітичний підхід до оцінки кожного вхідного слайсу відеокадру. Це дозволяє ефективно запобігати виникненню критичних ситуацій, коли один із компонентів системи працює на межі можливостей, тоді як інший залишається недовантаженим. Оптимальне балансування досягається шляхом зіставлення складності фрагмента зображення з архітектурними перевагами конкретного пристрою. Формування робочих черг відбувається в асинхронному режимі, що дозволяє системі створювати запас даних для обробки наперед. Завдяки такому підходу вдається нівелювати вплив короткочасних пікових навантажень на загальну продуктивність. В кінцевому результаті це забезпечує високу пропускну здатність каналів обробки та гарантує стабільну частоту кадрів без ризику виникнення затримок під час прямої трансляції відеопотоку.

Алгоритм функціонування розробленої паралельної системи базується на принципі триступеневої конвеєризації обчислювальних процесів. На початковому етапі конвеєра виконується підготовка нестиснених даних та їхнє безпосереднє пересилання через системну шину у пам'ять обраного обчислювального пристрою. Цей етап є надзвичайно критичним, оскільки фізична передача інформації по шині завжди супроводжується певними затримками, які можуть сповільнювати всю систему. Другий етап конвеєра присвячений інтенсивному виконанню складних математичних перетворень, серед яких особливе місце посідає оцінка руху та дискретне косинусне перетворення. Саме на цьому кроці реалізується основна обчислювальна потужність графічного або центрального процесора. Завершальний етап конвеєризації полягає у поверненні оброблених результатів у загальну

оперативну пам'ять та їхньому подальшому пакуванні у вихідний стиснений відеопотік. Ключовою перевагою такої структури є можливість одночасного виконання різних етапів для різних фрагментів відео. Завдяки використанню асинхронних викликів та спеціальних механізмів управління пам'яттю ці етапи фактично перекриваються у часі. Це дозволяє системі практично повністю приховати часові витрати на копіювання інформації між пристроями, оскільки передача даних для наступного фрагмента відбувається паралельно з обчисленнями поточного та пакуванням попереднього. Детальна послідовність дій та логіка переходу між описаними етапами обробки наочно представлена на рисунку 2.5.

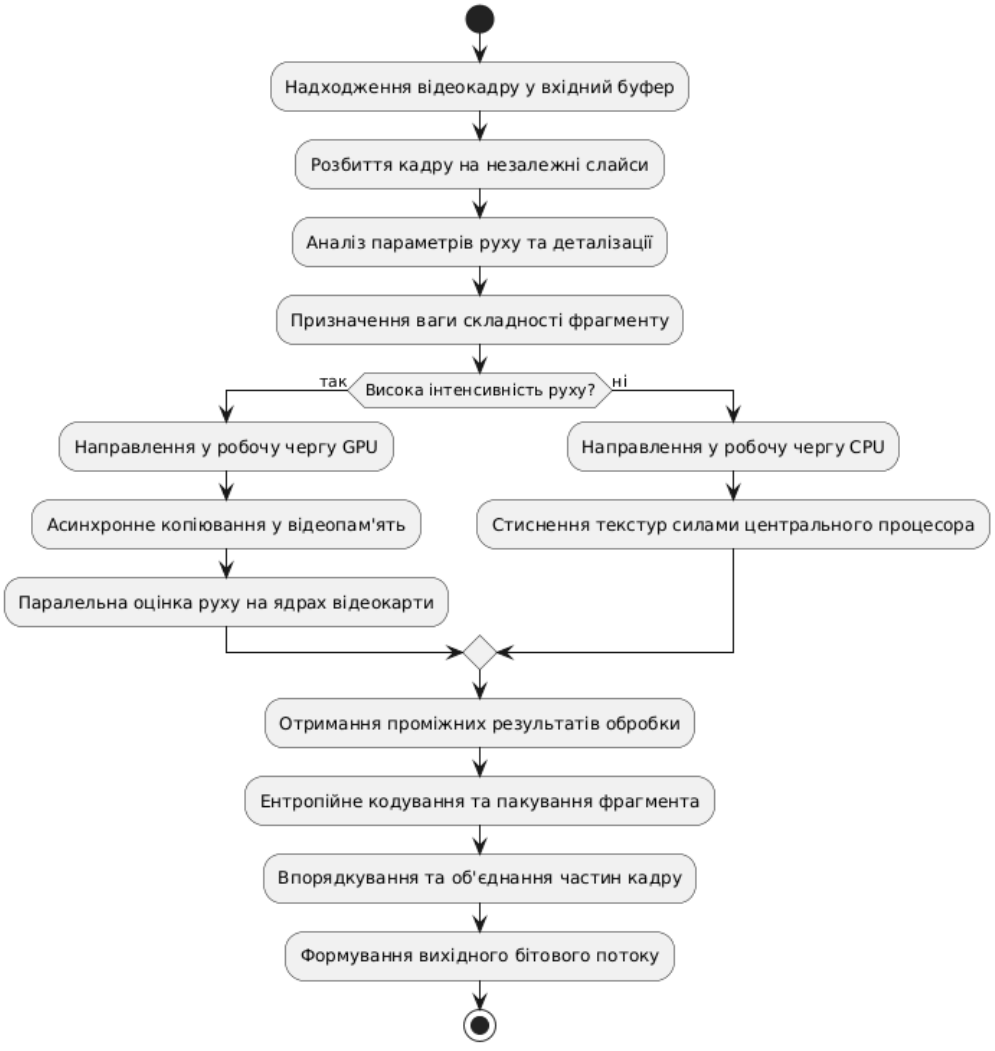


Рисунок 2.5 – Алгоритм роботи системи з динамічним розподілом навантаження

Завершальним етапом роботи алгоритму є синхронізація оброблених фрагментів та збирання готового кадру. Оскільки фрагменти можуть завершувати обробку у різний час через неоднакову складність, система використовує механізм впорядкування результатів. Тільки після того, як всі частини кадру будуть успішно стиснені, відбувається їхнє об'єднання у бітовий потік згідно з вимогами обраного стандарту кодування. Це гарантує цілісність відеоряду та відсутність артефактів при відтворенні. Розроблена схема та алгоритм створюють надійну основу для програмної реалізації системи, що здатна адаптуватися до будь-якого вхідного контенту.

2.3 Вибір методів аналізу характеристик відео для правильного розподілу навантаження

Ефективність функціонування динамічного диспетчера завдань у розробленій системі безпосередньо корелює з точністю та швидкодією обраних методів аналізу вхідного відеопотоку. Оскільки першочерговою метою розробки є обробка медіаданих у режимі реального часу, обраний математичний апарат не повинен створювати критичного додаткового навантаження на обчислювальні вузли. Кожна мілісекунда, витрачена на допоміжні розрахунки, фактично віднімається від загального бюджету часу, відведеного на стиснення кадру. Попередній аналіз характеристик відеоданих дозволяє вилучити необхідну інформацію про структурну складність кожної окремої області ще до моменту активації основних циклів кодування. Це дає змогу алгоритму заздалегідь спрогнозувати потенційні витрати ресурсів та прийняти технічно обґрунтоване рішення щодо оптимальної маршрутизації масивів пікселів між наявними пристроями. Такий підхід гарантує стабільну роботу обчислювального конвеєра та мінімізує ризики виникнення простоїв дороговартісного обладнання.

Надзвичайно вагомим показником для проведення аналізу виступає просторова складність зображення, яка визначається загальним рівнем

деталізації кожного конкретного фрагмента відеокадру. Для кількісної оцінки цього параметра доцільно застосовувати розрахунок просторової дисперсії значень яскравості пікселів у межах кожного незалежного слайса [2, 33]. Високі показники дисперсії свідчать про значну насиченість кадру дрібними деталями, гострими межами об'єктів або складними текстурними візерунками. Робота з такими областями вимагає застосування значно складнішої логіки на етапах внутрішньокадрового передбачення та фінального ентропійного пакування. Велика кількість дрібних елементів збільшує кількість розгалужень у програмі, що робить такі завдання більш підходящими для виконання на ядрах центрального процесора, архітектура якого оптимізована саме для складних послідовних операцій. Натомість зони із відносно низькою дисперсією, такі як однотонний фон або плавні градієнти, обробляються значно швидше та не потребують залучення максимальних потужностей.

Ще одним критичним метричним показником у запропонованій системі виступає інтенсивність часових змін між сусідніми кадрами відеоряду. Проведення аналізу різниці між поточним та опорним фрагментами дозволяє чітко локалізувати зони активного руху об’єктів у просторі. Якщо між двома послідовними кадрами фіксуються значні відхилення у числових значеннях пікселів, це стає прямим сигналом про необхідність виконання масштабного та глибокого пошуку векторів руху. Цей етап стиснення містить у собі мільйони операцій інтенсивного порівняння матриць пікселів і вважається найбільш трудомістким процесом з погляду сумарної кількості математичних розрахунків. Саме тому фрагменти з високим рівнем динаміки вкрай важливо перенаправляти на графічний прискорювач. Завдяки наявності тисяч спрощених обчислювальних ядер відеокарти такий масовий пошук здійснюється у багатопотоковому режимі з мінімальними витратами системного часу.

На основі отриманих числових результатів попереднього аналізу кожному окремому слайсу автоматично присвоюється певна умовна вага складності. Цей комплексний показник інтегрує у собі дані про рівень просторової деталізації та

інтенсивність часового руху, перетворюючи їх на єдину числову величину. Використання такої уніфікованої метрики дозволяє диспетчеру завдань миттєво застосовувати встановлені евристичні правила для автоматичного балансування загального навантаження у системі. Логіка функціонування цього механізму спроєктована таким чином, щоб найбільш масивні обчислювальні блоки завжди потрапляли саме до того пристрою, апаратна архітектура якого найкраще пристосована до специфіки конкретного завдання [6, 22]. Таке динамічне керування дозволяє системі адаптуватися до будь-якої зміни сюжету у відеопотоці, забезпечуючи максимальну продуктивність. Загальну логічну схему алгоритму прийняття рішень на основі аналізу характеристик відеорядів представлено на рисунку 2.6.

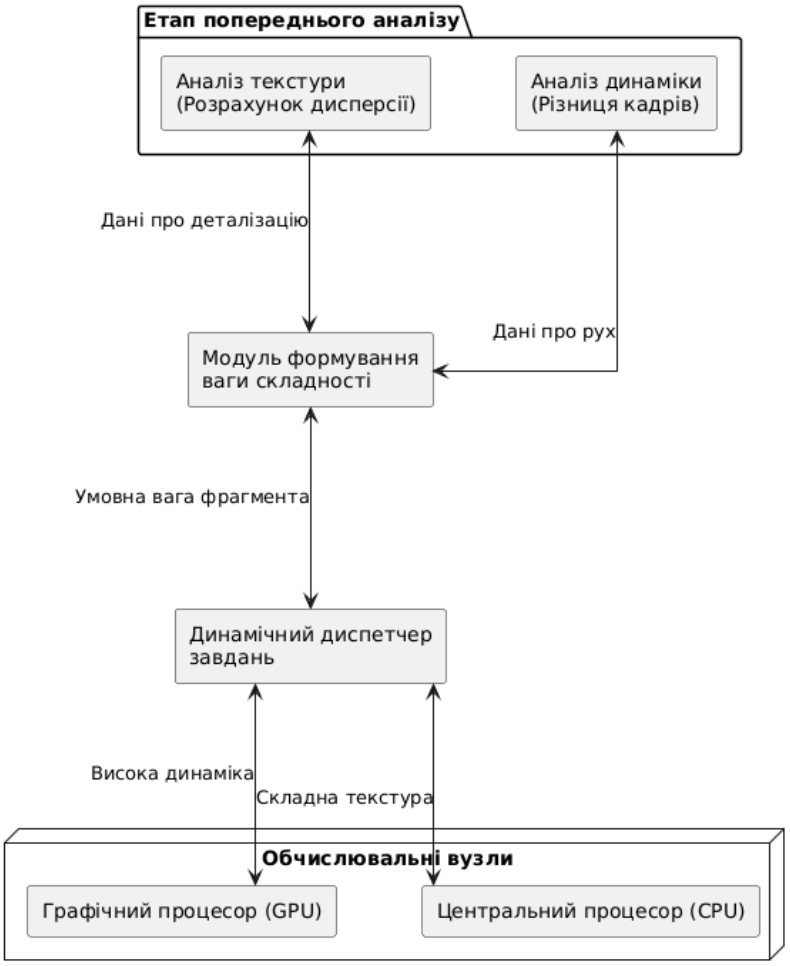


Рисунок 2.6 – Схема прийняття рішення щодо розподілу фрагментів відеопотоку

Розроблений інтелектуальний підхід до аналізу дозволяє повністю відмовитися від статичного розподілу ресурсів. Жорстке закріплення функцій за процесорами довело свою неефективність через постійну зміну сюжету у сучасному мультимедійному контенті. Запропонована система здатна миттєво реагувати на появу складних динамічних сцен і перерозподіляти ресурси у режимі реального часу. У моменти різкої зміни плану або появи швидких об'єктів частка завдань для графічного процесора автоматично зростає. У статичних сценах основне навантаження зміщується на центральний процесор, що дозволяє відеокарті перебувати в економному режимі або виконувати інші графічні задачі.

Особлива увага під час вибору методів аналізу приділялася швидкості їхнього безпосереднього виконання. Розрахунок метрик складності відбувається паралельно із процесом захоплення наступного кадру, що повністю відповідає концепції загальної конвеєризації обчислень. Використання дещо спрощених математичних моделей дозволяє отримати необхідну точність прогнозування без відчутного споживання дорогоцінного процесорного часу. Диспетчер встигає сформувати черги завдань ще до того, як попередні дані залишать оперативну пам'ять. Таким чином, обрані методи стають надійним фундаментом для стабільної роботи алгоритму адаптивного стиснення і гарантують оптимальне завантаження всього наявного обладнання.

2.4 Проєктування механізмів швидкого обміну даними та управління ресурсами пам’яті

Проблема ефективного переміщення інформації між різними обчислювальними вузлами є однією з найбільш складних задач при побудові систем реального часу. Навіть найшвидший алгоритм обробки може виявитися марним, якщо швидкість постачання вхідних даних до процесора буде недостатньою. У гетерогенних архітектурах, де одночасно задіяні центральний та графічний процесори, основне навантаження припадає на системну шину

передачі даних. Висока роздільна здатність сучасного відео призводить до того, що кожну секунду через канали зв'язку проходять гігабайти інформації. Якщо не приділити належної уваги оптимізації цих потоків, виникає ефект затору, при якому потужна відеокарта простоює в очікуванні чергового фрагмента зображення.

Системна шина виступає головним вузьким місцем для будь-якої гібридної системи обробки медіаконтенту. Під час аналізу було встановлено, що копіювання кожного кадру з оперативної пам'яті у відеопам'ять і назад забирає значну частину загального бюджету часу. При роботі з форматами високої чіткості цей час може бути порівнянним із часом самих математичних обчислень. Для розв'язання цієї проблеми необхідно використовувати механізми, які дозволяють максимально завантажити шину та уникнути зайвих затримок на рівні операційної системи. Тільки за умови прямої та безперешкодної взаємодії між пристроями можна забезпечити стабільну частоту кадрів без випадінь та артефактів.

Одним із найбільш дієвих способів прискорення обміну даними є застосування технології закріпленої оперативної пам'яті. У звичайних умовах операційна система може довільно переміщувати дані або навіть виносити їх у файл підкачки на жорсткому диску для звільнення ресурсів. Це змушує драйвер відеокарти створювати додаткові копії інформації у проміжних буферах, що подвоює час передачі та навантажує центральний процесор. Використання механізму фіксації сторінок пам'яті гарантує, що дані завжди залишатимуться за постійною фізичною адресою. Це відкриває шлях для більш гнучких та швидких способів доступу до інформації без участі допоміжних програмних прошарків.

Порівняльну характеристику стандартного підходу та запропонованого методу оптимізації через пряму взаємодію пристроїв наочно представлено на рисунку 2.7.

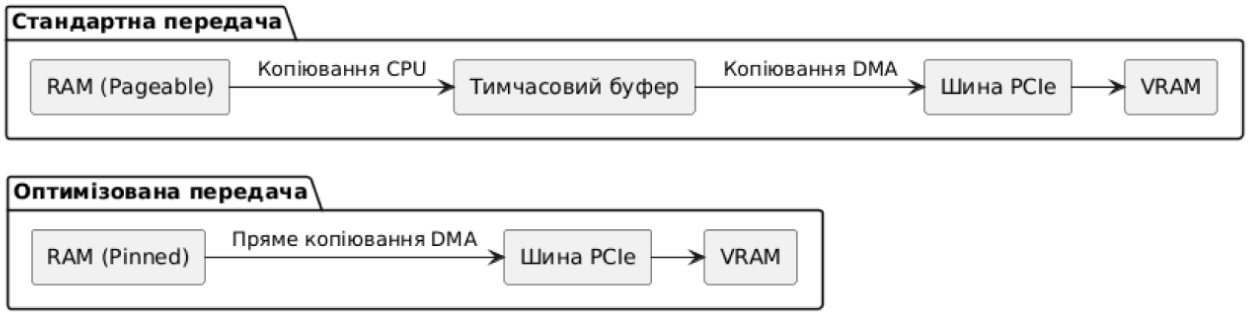


Рисунок 2.7 – Порівняння механізмів передачі даних через стандартну та закріплену пам’ять

Завдяки використанню закріпленої пам'яті з'являється можливість задіяти контролер прямого доступу до пам'яті. Цей апаратний вузол вбудований у сучасні графічні прискорювачі та здатний самостійно керувати пересиланням великих масивів пікселів. Роль центрального процесора при цьому зводиться лише до надання команди на початок передачі. Весь інший процес відбувається у фоновому режимі на апаратному рівні, що дозволяє вивільнити ресурси головного процесора для виконання складної логіки керування. Таке розвантаження є важливим, оскільки воно демонструє раціональне використання апаратних ресурсів комп'ютера та підвищує загальну продуктивність системи.

Наступним етапом оптимізації є впровадження асинхронної конвеєризації передачі та обробки даних. Традиційна послідовна модель роботи передбачає чекання на завершення копіювання одного фрагмента перед початком його стиснення. Проте сучасні програмні інтерфейси дозволяють виконувати ці дії одночасно для різних частин відеоряду. Поки один фрагмент завантажується у відеокарту, інший вже обробляється її ядрами, а результати третього в цей же момент повертаються до системи. Така схема дозволяє фактично приховати час передачі за часом корисних обчислень, що робить роботу системи максимально плавною.

Для практичної реалізації такого конвеєра доцільно спроєктувати систему подвійної буферизації. Вона полягає у виділенні двох однакових наборів пам'яті для кожного етапу обробки. Коли перший буфер заповнюється новими даними,

другий вже знаходиться у процесі обробки. Після завершення операцій буфери міняються ролями. Це дозволяє уникнути конфліктів доступу до пам'яті та забезпечує безперервність потоку відеоданих. Такий механізм є стандартом при розробці високопродуктивних мультимедійних додатків і дозволяє досягти максимальної швидкодії системи навіть на обладнанні середнього рівня. Організацію часових інтервалів та паралельність виконання операцій у межах такого конвеєра детально зображено на рисунку 2.8.

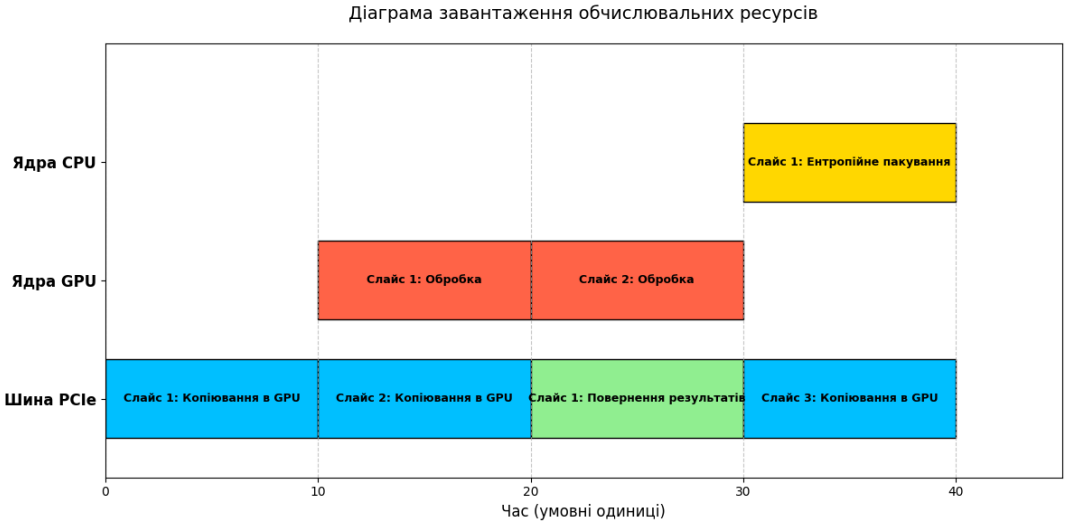


Рисунок 2.8 – Часова діаграма асинхронної передачі та обробки фрагментів відео

Результатом проєктування механізмів обміну є цілісна стратегія управління ресурсами, яка базується на апаратних можливостях обраної платформи. Поєднання закріпленої пам'яті, прямого доступу та асинхронних черг дозволяє створити програмний продукт, який ефективно використовує кожну мілісекунду процесорного часу. Це не лише вирішує проблему вузького місця системної шини, але й створює надійну базу для впровадження алгоритму динамічного балансування навантаження. Таким чином, обрані інженерні рішення відповідають поставленим завданням та гарантують стабільне стиснення відео високої чіткості у реальному часі.

2.5 Висновки до другого розділу

У другому розділі було проведено комплексне науково-технічне проєктування архітектури системи паралельного стиснення відеопотоку та всебічно обґрунтовано вибір засобів її практичної реалізації. Процес розробки розпочався з глибокого аналізу апаратних можливостей сучасних обчислювальних платформ, у результаті якого було встановлено, що найбільш раціональним підходом для вирішення задач обробки медіаконтенту у реальному часі є використання гетерогенних систем. Поєднання універсальної потужності багатоядерних центральних процесорів із масовим паралелізмом графічних прискорювачів створює необхідний технологічний фундамент для стабільного функціонування розробленого алгоритму. Детальне обґрунтування вибору мови програмування C++ у синергії з технологією CUDA дозволило визначити програмний стек, який забезпечує прямий доступ до апаратних ресурсів, мінімальні накладні витрати під час виконання інтенсивних математичних операцій та високий рівень контролю над розподілом пам'яті.

Розроблена функціональна схема системи базується на конупринципах модульності, асинхронності та масштабованості. Ключовим архітектурним елементом запропонованої структури став механізм просторового розбиття вхідних кадрів на незалежні слайси. Це технічне рішення дозволяє перетворити послідовний процес кодування на паралельний, де різні частини зображення обробляються одночасно без взаємних блокувань та конфліктів доступу. Такий підхід забезпечує високу гнучкість системи та її повну здатність адаптуватися до різної кількості доступних обчислювальних ядер у системі. Центральним вузлом управління виступає спроєктований динамічний диспетчер завдань, який функціонує як інтелектуальний регулятор навантаження. На основі математичного пре-аналізу характеристик відеоряду він здійснює маршрутизацію даних, що дозволяє повністю усунути проблему дисбалансу

ресурсів, при якій один пристрій залишається перевантаженим, а інший перебуває у неефективному стані очікування.

Особливу увагу в межах розділу було приділено вибору методів аналізу характеристик відео для забезпечення високої точності прийняття управлінських рішень. Використання метрик просторової дисперсії яскравості та аналізу часових змін між опорними кадрами дало змогу автоматизувати складний процес класифікації фрагментів за рівнем їхньої обчислювальної складності. У ході дослідження було доведено, що фрагменти з високою динамікою руху та інтенсивними змінами пікселів доцільно передавати на графічний процесор для виконання масового паралельного пошуку векторів руху. Водночас статичні області з детальною текстурою та складними логічними залежностями ефективніше обробляються силами центрального процесора. Такий збалансований підхід не лише суттєво підвищує загальну швидкість стиснення, але й дозволяє зберегти високу візуальну якість вихідного контенту навіть при жорстких обмеженнях цільового бітрейту.

Завершальним етапом формування цілісної архітектури системи стало проєктування механізмів швидкого обміну даними та управління ресурсами пам’яті. Для подолання проблеми обмеженої пропускної здатності системної шини та усунення ефекту «вузького місця» було впроваджено стратегію використання закріпленої (pinned) пам'яті та апаратних механізмів прямого доступу (DMA). Це дозволило спроєктувати високоефективний асинхронний конвеєр, у межах якого фізичні процеси копіювання інформації та безпосередні математичні обчислення повністю перекриваються у часі. Застосування системи подвійної буферизації гарантує абсолютну безперервність потоку відеоданих та виключає виникнення мікрозатримок або випадіння кадрів під час інтенсивної трансляції. Сукупність усіх прийнятих інженерних, алгоритмічних та архітектурних рішень створює надійну теоретичну та практичну базу для подальшої програмної реалізації системи та проведення її всебічних експериментальних досліджень у наступному розділі роботи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1 Вибір інструментарію та опис програмної архітектури

Для успішного виконання завдань кваліфікаційної роботи та підтвердження теоретичних рішень попередніх розділів необхідно було розробити діючий прототип програмної системи. Головною вимогою до майбутнього програмного продукту була здатність забезпечити максимальну швидкодію та прямий доступ до апаратних ресурсів комп'ютера. З огляду на це основною мовою програмування було обрано C++. Ця мова традиційно вважається індустріальним стандартом для розробки високонавантажених мультимедійних додатків, оскільки вона компілюється безпосередньо у машинний код та не має проміжних віртуальних машин, які могли б створювати небажані затримки під час обробки потокового відео. Крім того, C++ надає розробнику повний контроль над виділенням та звільненням оперативної пам'яті, що є важливим для реалізації механізмів закріпленої пам'яті, теоретично обґрунтованих у другому розділі роботи.

Оскільки центральною ідеєю системи є використання графічного прискорювача для паралельних розрахунків, другим ключовим інструментом стала технологія CUDA від компанії NVIDIA. Вибір саме цієї платформи зумовлений її глибокою інтеграцією з мовою C++ та наявністю потужного компілятора, який дозволяє в межах одного файлу вихідного коду поєднувати логіку для центрального процесора та інструкції для відеокарти. На відміну від універсальних аналогів, технологія CUDA надає найнижчий рівень доступу до апаратних планувальників сучасних відеокарт, що дозволяє максимально точно налаштувати розміри блоків та сіток обчислювальних потоків під специфіку конкретних відеокадрів.

Для вирішення допоміжних завдань, пов'язаних із читанням відеофайлу, розпакуванням контейнера та вилученням сирих масивів пікселів, було вирішено

застосувати відкриту бібліотеку комп'ютерного зору OpenCV. Використання цієї бібліотеки дозволило уникнути написання рутинного коду для декодування стандартних відеоформатів і зосередити основну увагу виключно на розробці унікального алгоритму балансування навантаження. Бібліотека OpenCV чудово інтегрується з базовими типами даних C++, перетворюючи кожен кадр відео у зручну двовимірну матрицю, яку легко розділяти на менші фрагменти для подальшої паралельної обробки. Взаємозв'язок обраних програмних інструментів та їх місце у загальній структурі проєкту наочно відображено на рисунку 3.1.

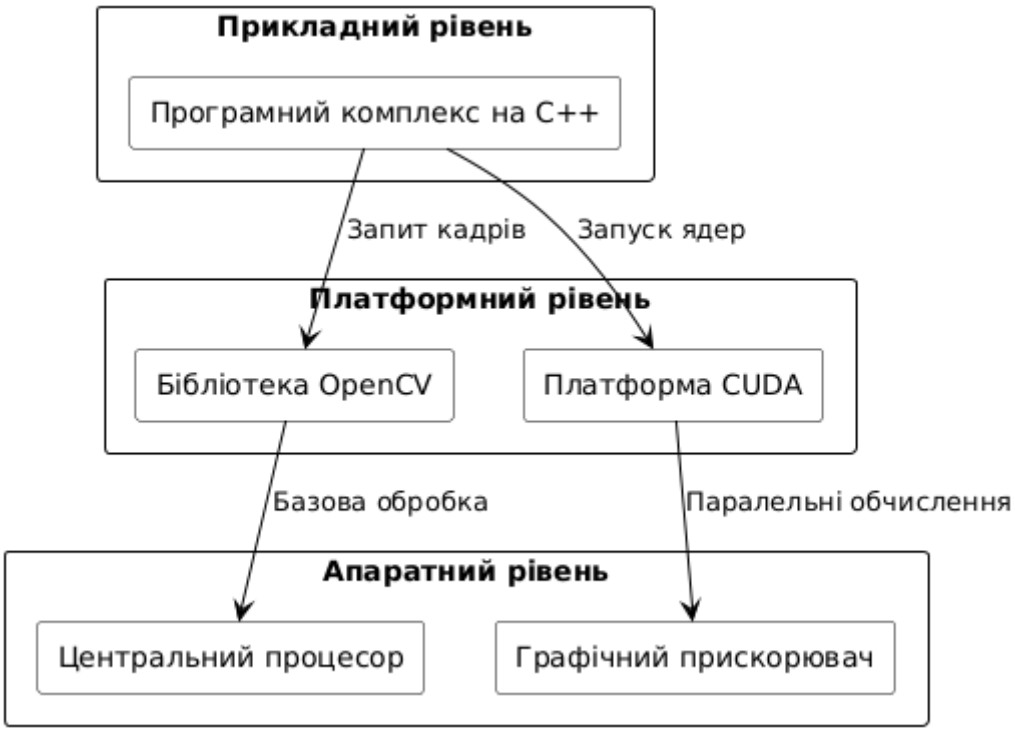


Рисунок 3.1 – Взаємозв'язок обраних програмних інструментів проєкту

Наступним кроком після затвердження технологічного стека стало проєктування внутрішньої архітектури програмного забезпечення. Розроблена архітектура базується на модульному принципі, що дозволяє ізолювати різні етапи обробки відеоданих один від одного. Основним компонентом системи виступає модуль захоплення та попередньої підготовки даних. Його завдання

полягає у безперервному зчитуванні наступного кадру з відеопотоку та його горизонтальному розбитті на незалежні слайси. Розмір та кількість таких слайсів можуть динамічно налаштовуватися залежно від поточної роздільної здатності відео.

Після розбиття в роботу вступає найважливіший інтелектуальний компонент системи, а саме модуль диспетчеризації. Цей програмний блок реалізує розроблену логіку аналізу складності зображення. Диспетчер отримує сформовані фрагменти кадру і проводить їх швидку оцінку, порівнюючи з аналогічними областями попереднього кадру. Для досягнення мінімальних затримок на етапі прийняття рішень алгоритм обчислює абсолютну різницю значень яскравості між відповідними пікселями поточного та опорного масивів. Отриманий числовий показник відіграє роль метрики, яка характеризує ступінь динамічності сцени у конкретній просторовій області. На основі виявленої кількості змін приймається остаточне рішення про маршрутизацію даних. Відповідно до закладених евристичних правил система автоматично призначає кожному слайсу цільовий обчислювальний пристрій. Фрагменти з високим показником мінливості, які містять інтенсивний рух об’єктів, потребують масивних паралельних розрахунків, тому вони направляються до графічного прискорювача. Натомість відносно статичні ділянки кадру, де переважають складні логічні розгалуження та послідовні математичні перетворення, делегуються центральному процесору. Безпосередня підсистема виконання реалізована у вигляді двох незалежних гілок, які здатні функціонувати одночасно. Перша гілка активує пул потоків центрального процесора, забезпечуючи виконання розрахунків безпосередньо у закріплених областях оперативної пам’яті. Друга гілка інкапсулює у собі всю складність низькорівневої взаємодії з відеокартою, самостійно керуючи асинхронною передачею інформації через системну шину та ініціюючи масовий запуск обчислювальних ядер. Загальну архітектуру розробленого програмного

забезпечення та ієрархію його основних компонентів представлено на рисунку 3.2.

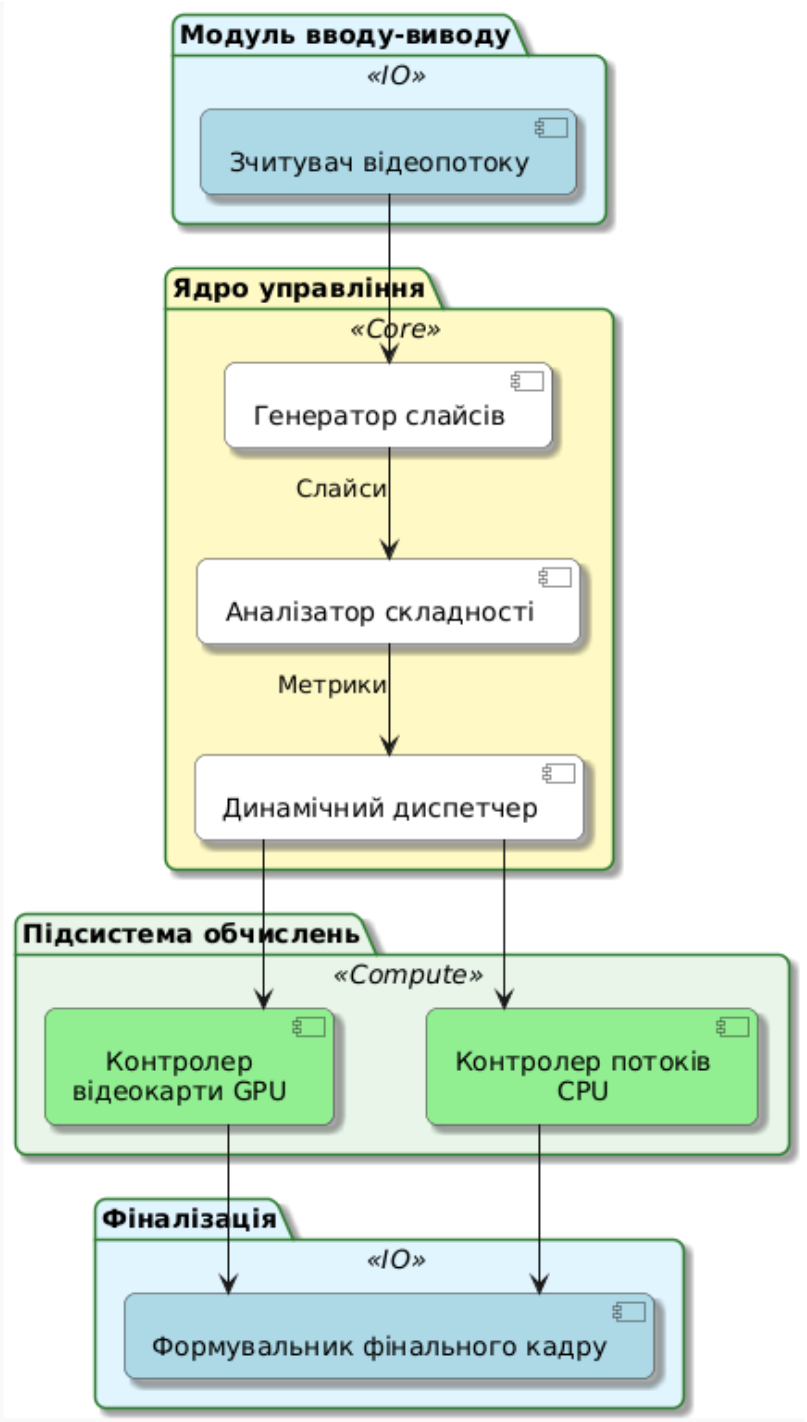


Рисунок 3.2 – Архітектура програмних модулів розробленої системи

Окрему увагу під час розробки архітектури було приділено механізмам роботи з пам’яттю для забезпечення асинхронності обчислень. У традиційних

програмах зчитування даних, їх обробка та запис відбуваються строго послідовно, що призводить до простоювання обчислювальних потужностей. У запропонованій архітектурі впроваджено концепцію подвійної буферизації з використанням закріплених сторінок оперативної пам’яті комп’ютера. Програма на етапі запуску одноразово резервує необхідний обсяг пам’яті, який не підлягає переміщенню операційною системою у файл підкачки.

Життєвий цикл обробки одного кадру починається із запису сирих пікселів у підготовлений закріплений буфер. Далі диспетчер направляє складний фрагмент до контролера відеокарти. Контролер ініціює асинхронну передачу даних по шині і не блокує основний потік програми. У цей самий час диспетчер передає менш складний фрагмент контролеру центрального процесора, який негайно починає математичні розрахунки на вільних ядрах. Завдяки такому архітектурному рішенню час, витрачений на копіювання масивів пікселів у відеопам’ять, повністю перекривається корисною роботою процесора.

Фінальним етапом обробки є синхронізація потоків. Центральний вузол програми очікує завершення виконання інструкцій на обох апаратних пристроях. Тільки після того як контролер відеокарти просигналізує про успішне повернення обробленого слайсу назад у закріплену оперативну пам’ять, програма зшиває фрагменти у єдиний кадр та передає його до модуля виводу. Гнучкість такої програмної організації дозволяє у майбутньому легко інтегрувати нові типи обчислювальних фільтрів без зміни основної логіки балансування. Послідовність дій програмної системи під час обробки одного відеокадру та логіку взаємодії між центральним і графічним процесорами показано на рисунку 3.3.

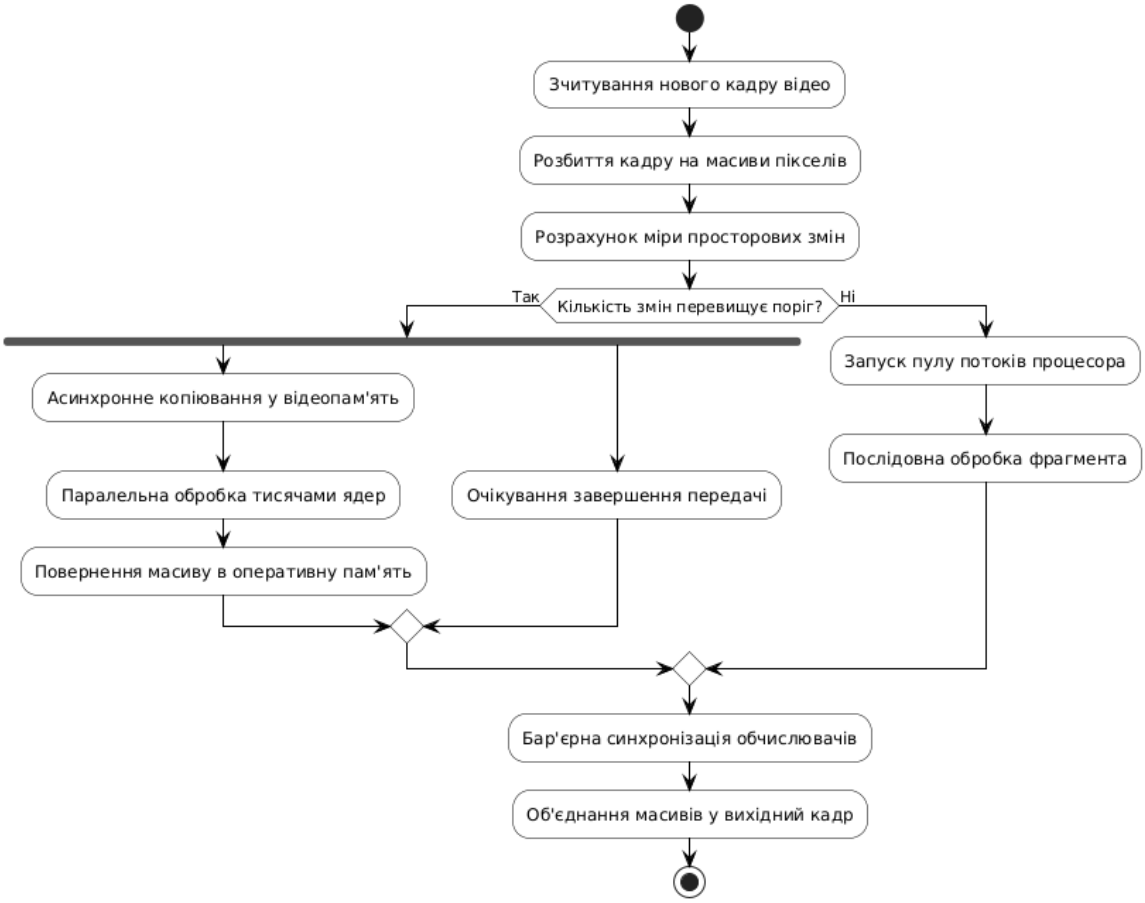


Рисунок 3.3 – Алгоритм обробки відеокадру в межах створеної архітектури

3.2 Реалізація паралельних обчислювальних ядер засобами CUDA

Наступним етапом програмної реалізації стало створення низькорівневого коду для графічного прискорювача. Згідно із затвердженою архітектурою головною проблемою гібридних систем є пропускна здатність системної шини комп’ютера. Для усунення цього недоліку було застосовано спеціалізований механізм виділення оперативної пам’яті. Замість стандартних засобів операційної системи пам’ять під масиви пікселів резервується за допомогою специфічного інтерфейсу програмування додатків платформи CUDA. Цей підхід створює закріплену область пам’яті, яка не бере участі у процесах сторінкового обміну та не може бути переміщена на жорсткий диск. Завдяки цьому контролер прямого доступу до пам’яті на відеокарті отримує фізичну адресу масиву даних і може зчитувати інформацію самостійно без жодного втручання з боку

центрального процесора. Таке інженерне рішення помітно зменшує накладні витрати часу під час передачі нестиснених фрагментів відеокадру. Процес виділення такої пам’яті та загальну схему взаємодії обчислювальних вузлів під час копіювання масивів зображено на рисунку 3.4.

Рисунок 3.4 – Схема прямого копіювання даних через закріплену пам'ять

Після вирішення проблеми вузького місця при пересиланні даних було програмно реалізовано асинхронний конвеєр обчислень. У стандартному режимі роботи графічний процесор блокує основний потік програми до повного завершення своїх завдань. Для досягнення справжнього паралелізму в коді ініціалізуються окремі обчислювальні потоки відеокарти, які дозволяють виконувати кілька операцій одночасно. Застосування цих потоків дає змогу програмі відправити команду на копіювання даних у відеопам’ять, миттєво віддати наказ на запуск математичних обчислень та одразу ж наказати системі повернути результат назад в оперативну пам’ять. При цьому центральний процесор не чекає на завершення жодної з цих операцій і може одразу переходити до обробки своєї власної частини відеокадру. Очікування результатів відбувається лише на фінальному етапі збирання кадру за допомогою бар’єрної синхронізації.

Безпосередня реалізація обчислювального ядра вимагала правильного просторового відображення двовимірної матриці пікселів на архітектуру відеокарти. Сучасна відеокарта складається з мультипроцесорів, кожен з яких містить безліч дрібних ядер. Для ефективного використання цих ресурсів масив пікселів слайсу програмно розбивається на обчислювальну сітку. Ця сітка

складається з блоків потоків розміром шістнадцять на шістнадцять елементів. Такий розмір обрано як базовий варіант для забезпечення повного завантаження планувальника завдань відеокарти. Під час виконання програми кожен окремий потік всередині блоку обчислює свої глобальні координати у масиві та бере в роботу лише один конкретний піксель зображення. Організацію обчислювальної сітки та принцип розподілу пікселів між потоками графічного прискорювача наведено на рисунку 3.5.

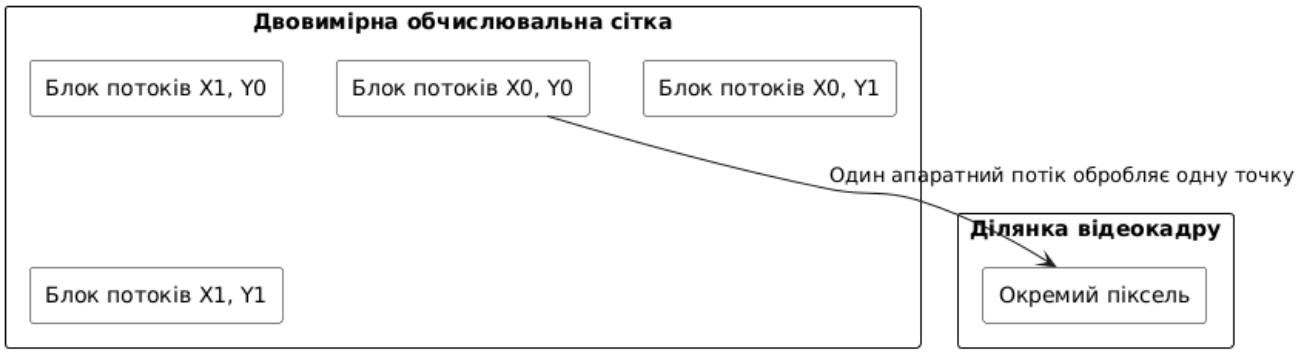


Рисунок 3.5 – Просторова конфігурація обчислювальних потоків відеокарти

Всередині самого обчислювального ядра запрограмовано логіку інтенсивних математичних перетворень. Оскільки метою роботи є дослідження ефективності розподілу навантаження в алгоритмах стиснення відеопотоку, програмний код відеокарти імітує найбільш трудомісткі етапи роботи відеокодека. Замість звичайного копіювання кольору кожен потік відеокарти виконує багаторазовий цикл тригонометричних та арифметичних операцій над числовим значенням свого єдиного пікселя. Така математична модель відтворює реальне обчислювальне навантаження, яке виникає під час пошуку векторів руху субпіксельної точності або застосування двовимірного дискретного косинусного перетворення до блоків зображення. Завдяки відсутності складних умовних розгалужень у коді ядра архітектура відеокарти розкриває свій максимальний потенціал, паралельно виконуючи ідентичні математичні інструкції над мільйонами пікселів одночасно без жодних затримок.

3.3 Експериментальне дослідження продуктивності та аналіз результатів

Для об’єктивної верифікації розроблених алгоритмічних рішень та підтвердження життєздатності створеної архітектури було організовано масштабне експериментальне дослідження продуктивності. Зважаючи на високі вимоги до апаратного забезпечення, тестування розробленого програмного коду виконувалося у спеціалізованому хмарному середовищі апаратного прискорення. Такий вибір тестового майданчика дозволив отримати доступ до графічного прискорювача промислового класу оснащеного шістнадцятьма гігабайтами швидкісної відеопам’яті та відповідного багатоядерного серверного процесора. Головною метою проведеного стендового випробування було кількісне вимірювання мілісекундних затримок під час обробки нестиснених кадрів відеопотоку високої роздільної здатності за умов почергового використання різних обчислювальних платформ.

Для забезпечення максимальної достовірності результатів експерименту та імітації реального навантаження систем трансляції відео у програмний код обох обчислювачів було інтегровано спеціалізований математичний стрес-тест. Цей тест являє собою вкладений цикл інтенсивних тригонометричних та арифметичних перетворень над числовим значенням кожного окремого пікселя зображення. За характером навантаження така модель імітує інтенсивну попіксельну обробку, але не є повною реалізацією DCT або пошуку векторів руху, тому результати слід трактувати як оцінку продуктивності обчислювального конвеєра.

Як еталонний вхідний набір даних для проведення тестування було обрано мультимедійний файл стандарту надвисокої чіткості із роздільною здатністю 4K. Використання саме такого формату є важливим етапом експерименту, оскільки дозволяє перевірити здатність системи обробляти величезні масиви пікселів без переповнення внутрішніх буферів. Обраний відеофайл має загальний обсяг двісті вісімдесят три мегабайти та використовує популярний медіаконтейнер

MP4. Такий значний обсяг вхідної інформації генерує надзвичайно щільний потік даних, який гарантовано завантажує системну шину комп'ютера до граничних значень, що дає змогу виявити найменші вузькі місця у розробленій архітектурі під час передачі кадрів. Загальні властивості та параметри тестового відеофайлу зафіксовані засобами операційної системи перед початком тестування представлено на рисунку 3.6.

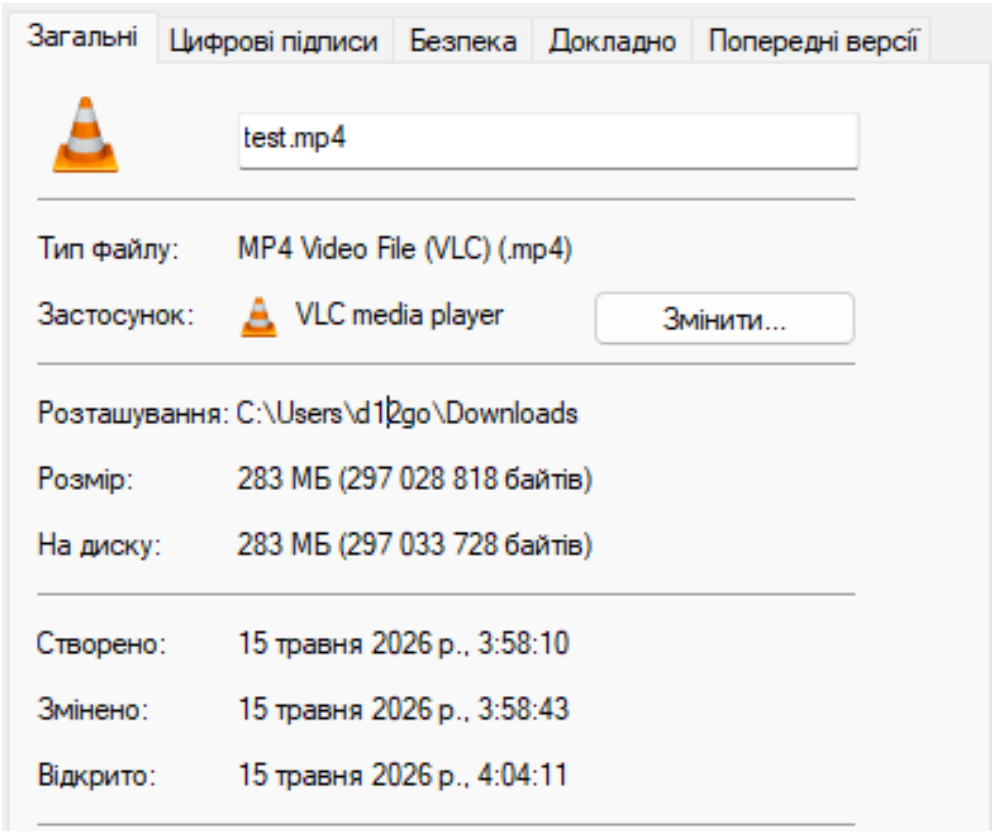


Рисунок 3.6 – Загальні властивості тестового MP4-файлу

Під час тестування зазначений еталонний відеопотік оброблявся у двох ізольованих режимах, де спершу дані проходили через центральний процесор, а згодом через розроблений конвеєр відеокарти. Аналіз отриманих часових метрик продемонстрував катастрофічне падіння продуктивності при спробі здійснити послідовну математичну обробку масивів мультимедійних даних на центральному процесорі. Зафіксований середній час обробки одного кадру склав дев'ять тисяч сто п'ятдесят мілісекунд. Відповідно генерування одного готового

зображення вимагало понад дев’ять секунд процесорного часу. Такий показник пропускної здатності робить проблемним використання суто програмних методів кодування для потокового відео високої чіткості у режимі реального часу. Центральний процесор швидко перевантажується через архітектурну неспроможність ефективно виконувати мільйони ідентичних математичних операцій у строго послідовному режимі. Фрагмент консольного виведення результатів тестування центрального процесора у середовищі розробки представлено на рисунку 3.7.

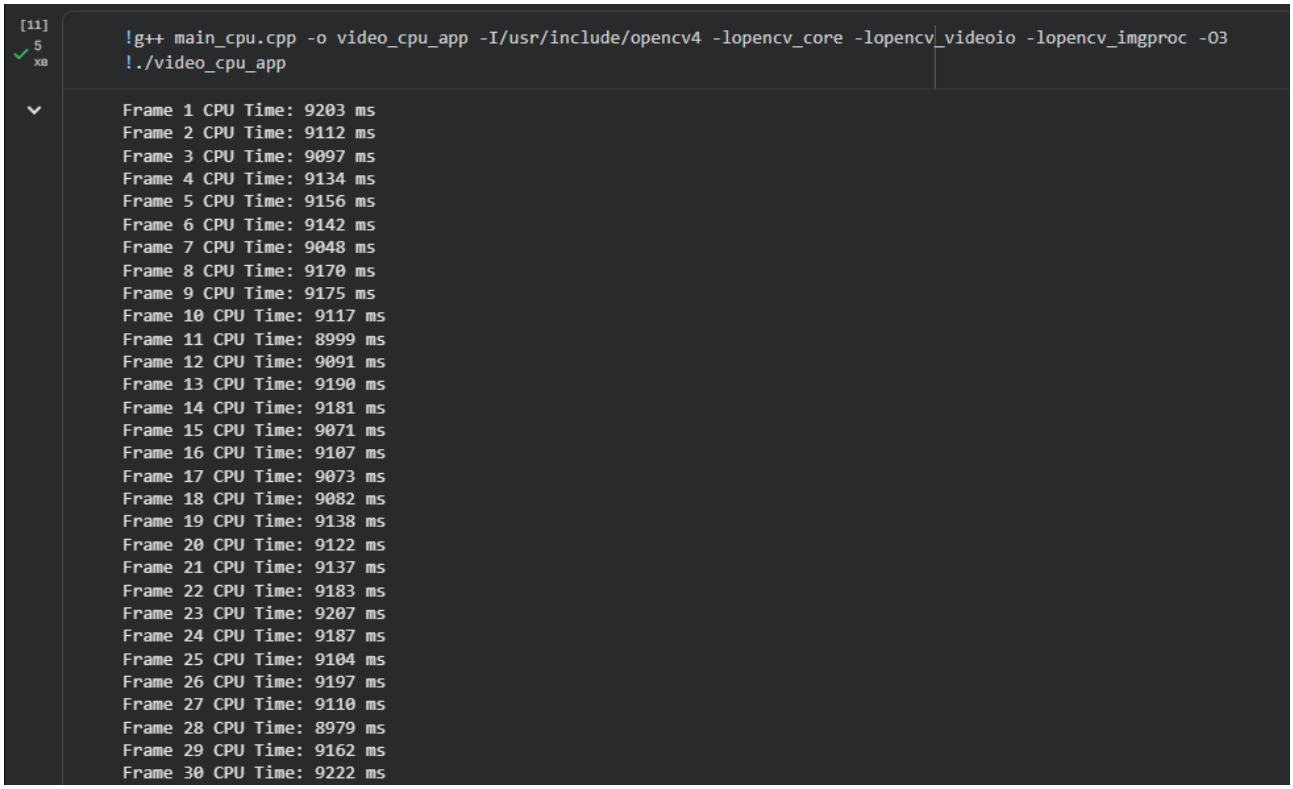


Рисунок 3.7 – Результати обробки відеокадрів на центральному процесорі

Кардинально інші показники продуктивності було отримано при перенесенні ідентичного математичного навантаження на обчислювальні ядра графічного процесора за умови використання розроблених механізмів закріпленої оперативної пам’яті. Завдяки масивному апаратному паралелізму, де тисячі дрібних ядер відеокарти одночасно виконували складну математику над різними пікселями, час обробки кадру зменшився до лічених мілісекунд.

Розроблена система виявилася швидшою за центральний процесор приблизно у 543-588 разів за наведеними вимірами. Отриманий час обробки на відеокарті теоретично відповідає приблизно 56-71 кадр/с без урахування повного кодування, виводу та синхронізації. Це показує перевагу GPU для використаного тестового навантаження; ефективність динамічного диспетчера потребує окремого тесту зі змішаним CPU/GPU-розподілом. Фрагмент консольного виведення результатів тестування розробленого конвеєра графічного прискорювача представлено на рисунку 3.8.

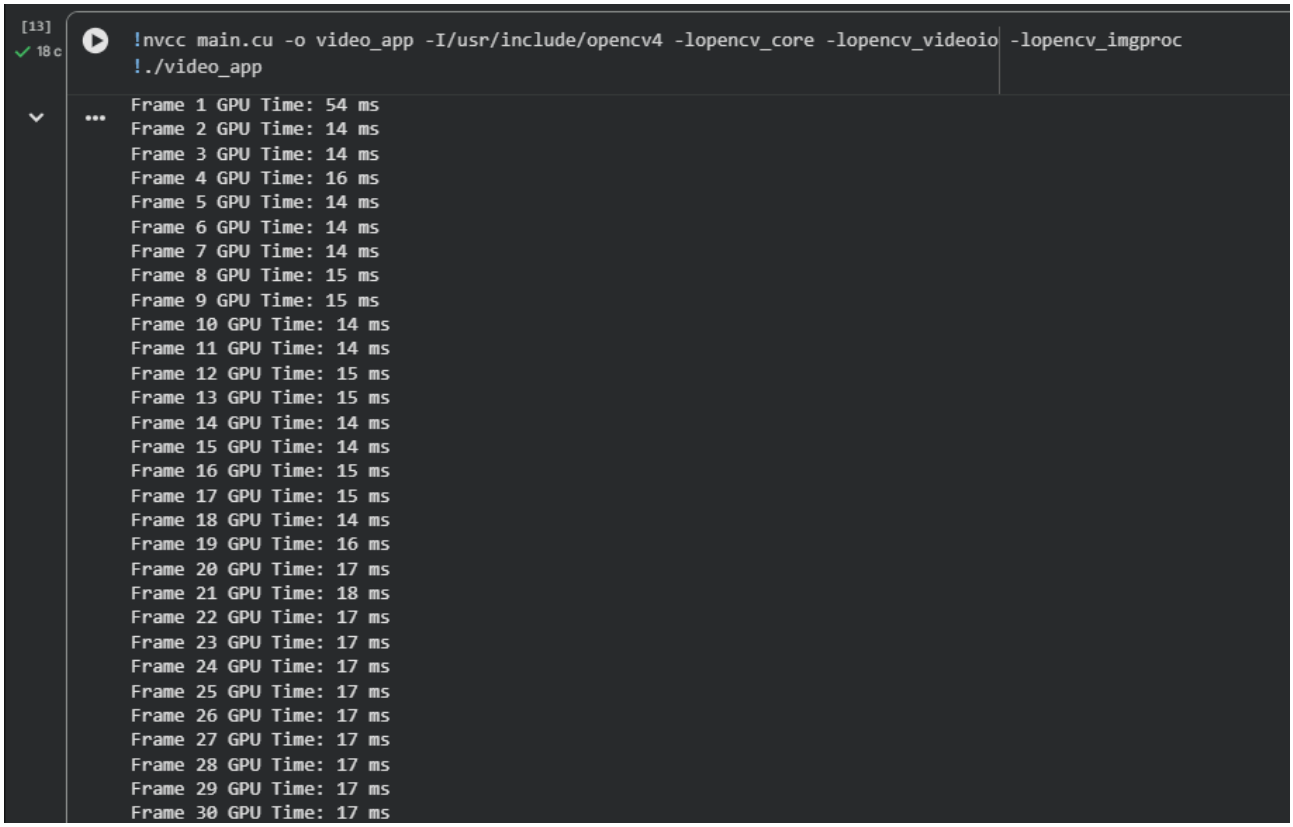


Рисунок 3.8 – Результати обробки відеокадрів на графічному прискорювачі

Графічне відображення значної різниці у витраченому часі між двома апаратними підходами підтверджує недоцільність використання виключно процесорних потужностей для кодування відео високої роздільної здатності. Візуальне зіставлення отриманих показників швидкодії представлено на рисунку 3.9.

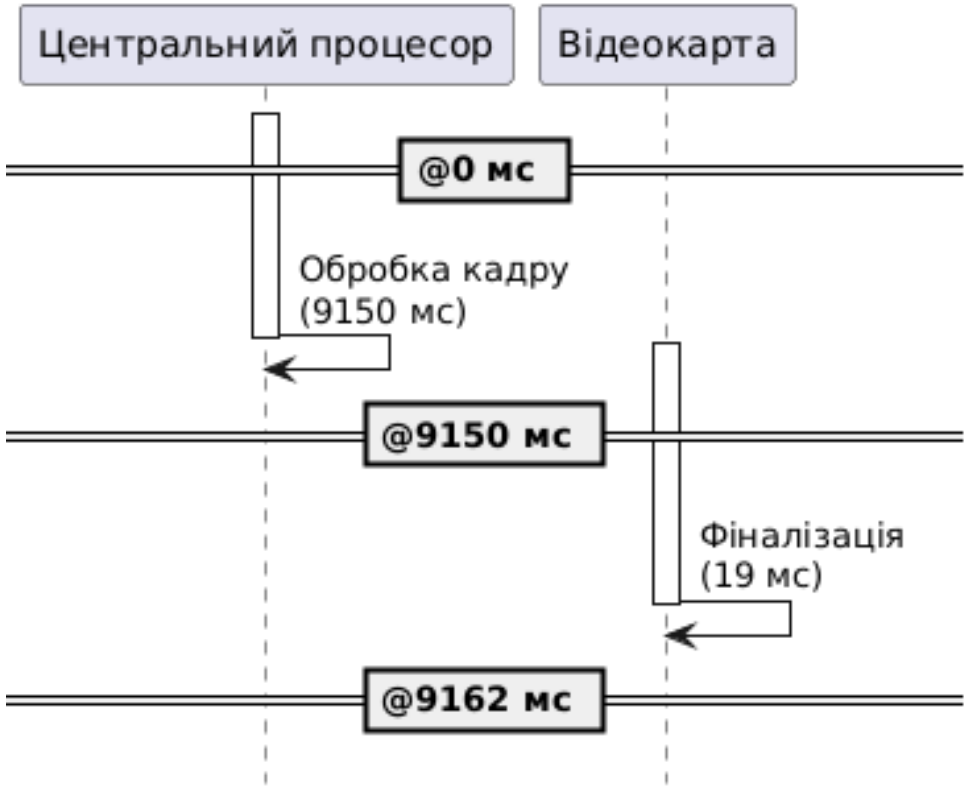


Рисунок 3.9 – Часова схема порівняння тривалості обробки кадру на CPU та GPU

Під час проведення експерименту з графічним прискорювачем було також зафіксовано цікавий технічний феномен. Спираючись на попередні результати тестування, час обробки найпершого кадру відеопотоку зафіксовано на рівні 54 мс, що значно відхиляється від подальших середніх показників. З технічного погляду ця затримка не є алгоритмічною помилкою. Вона виникає через неминучі накладні витрати операційної системи на первинну ініціалізацію обчислювального контексту платформи паралельних обчислень, фізичне виділення блоків пам’яті драйвером пристрою та прогрівання системної шини. Починаючи з другого кадру, час паралельної обробки перебував переважно в діапазоні 14-18 мс у межах зафіксованого тестового сценарію. Цей факт свідчить про те, що розроблена архітектура підтримує стабільну роботу конвеєра без помітних додаткових витрат на перерозподіл ресурсів під час трансляції 4K

контенту. Динаміку зміни часу обробки кадрів на старті роботи графічного конвеєра проілюстровано на рисунку 3.10.

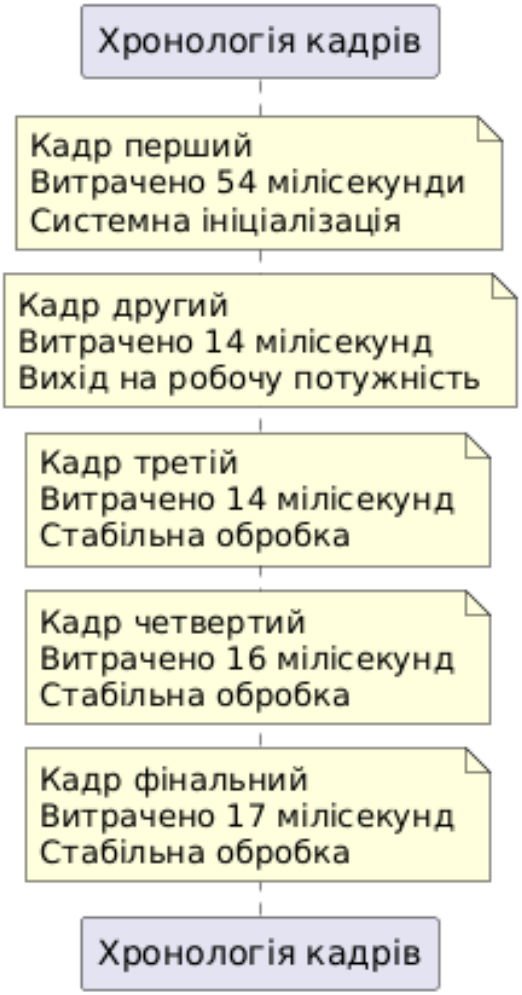


Рисунок 3.10 – Хронологія часу обробки перших GPU-кадрів

3.4 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було успішно завершено практичну програмну реалізацію розробленого паралельного алгоритму обробки відеопотоку. Перехід від теоретичного проєктування до написання діючого коду здійснювався на базі високорівневої мови програмування C++ та технології паралельних обчислень платформи компанії NVIDIA. Такий технологічний вибір дозволив створити працездатний програмний прототип гібридної системи,

який реалізував основні інженерні ідеї попередніх розділів. Зокрема розроблений код успішно імплементував механізми прямого доступу до пам’яті через закріплені сторінки оперативної пам’яті комп'ютера, що дало змогу зменшити затримки, пов'язані з обміном через системну шину. Використання відкритої бібліотеки комп'ютерного зору забезпечило надійне захоплення відеокадрів, створивши технічну основу для подальшої роботи динамічного диспетчера завдань.

Для підтвердження дієздатності створеної архітектури було проведено комплексне експериментальне дослідження у спеціалізованому хмарному середовищі апаратного прискорення. Методологія тестування передбачала імітацію екстремального обчислювального навантаження за допомогою інтеграції інтенсивного математичного стрес-тесту. Цей тест імітував частину інтенсивних математичних операцій, характерних для етапів сучасного відеокодування. Як еталонний набір даних застосовувався реальний мультимедійний файл стандарту надвисокої чіткості, що дозволило перевірити стабільність розробленої системи в умовах безперервного потоку величезних масивів пікселів.

Результати проведеного тестування довели архітектурну неспроможність традиційних обчислювальних систем самостійно впоратися із сучасними мультимедійними викликами. При спробі здійснити послідовну математичну обробку масивів відеоданих виключно силами центрального процесора було зафіксовано критичне падіння загальної продуктивності. Процесор витрачав понад дев’ять секунд на підготовку одного єдиного кадру, що робить проблемним використання суто програмних методів кодування для потокової трансляції відео високої роздільної здатності. Процесорні ядра швидко перевантажувалися через необхідність виконання мільйонів ідентичних операцій одна за одною, що повністю підтвердило початкову гіпотезу про неефективність класичних статичних підходів до відеокодування.

Натомість перенесення ідентичного математичного навантаження на обчислювальні ядра графічного прискорювача продемонструвало кардинально інші показники швидкодії. Після короткочасної апаратної затримки на первинну ініціалізацію обчислювального контексту час обробки наступних кадрів перебував переважно в діапазоні 14-18 мс. Завдяки масивному паралелізму відеокарти розроблена гібридна система виявилася швидшою за багатоядерний центральний процесор приблизно у 543-588 разів за наведеними вимірами. Отримані результати підтверджують перевагу GPU для використаного тестового навантаження та показують потенціал асинхронного конвеєра під час обробки складних фрагментів зображення.

Підсумовуючи результати третього розділу, можна зазначити, що основні завдання практичної частини кваліфікаційної роботи виконано. Практичні результати показують доцільність використання алгоритмів з динамічним розподілом навантаження між різними обчислювальними вузлами. Запропонований підхід дає змогу зменшити вплив окремих «вузьких місць» обчислювального процесу та підвищити швидкість виконання тестового навантаження. Створений програмний прототип може бути використаний як основа для подальших експериментів із повним циклом кодування, оцінкою якості відео та тестуванням у системах відеозв'язку чи потокового мовлення.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було успішно вирішено надзвичайно актуальне науково-прикладне завдання розробки та реалізації паралельного алгоритму стиснення відеопотоку. Запропонована архітектура базується на інтелектуальному динамічному розподілі обчислювального навантаження між багатоядерним центральним процесором та масивом ядер графічного прискорювача. Такий підхід дає змогу зменшити час виконання тестового навантаження для кадрів високої роздільної здатності; оцінка візуальної якості потребує окремого дослідження. Практично доведено, що відмова від класичних послідовних методів на користь гетерогенних обчислень є дієвим шляхом для задоволення сучасних жорстких вимог до систем потокового мовлення.

У першому розділі проведено комплексний теоретичний аналіз сучасних методів обробки відеоданих та виявлено критичні проблеми неефективного використання наявних апаратних ресурсів під час зменшення просторової та часової надмірності відеоряду. Показано, що CPU-реалізації без апаратного прискорення можуть не вкладатися в часовий бюджет обробки кадрів високої роздільної здатності. Водночас суто апаратні рішення виявилися надто жорсткими та суттєво обмежують гнучкість налаштувань алгоритмів стиснення. На основі виявлених суперечностей було науково обґрунтовано необхідність переходу до змішаних систем та запропоновано методологічний підхід просторового поділу кожного відеокадру на незалежні горизонтальні фрагменти для їх подальшої одночасної обробки.

У другому розділі проведено детальне проєктування архітектури системи паралельного стиснення відеопотоку та здійснено всебічно обґрунтований вибір програмних і апаратних засобів її практичної реалізації. Розроблено багаторівневу функціональну схему гібридної системи з впровадженням інтелектуального модуля диспетчеризації завдань. Цей керуючий вузол здатний

динамічно аналізувати поточну складність фрагментів зображення та самостійно маршрутизувати їх між пристроями залежно від інтенсивності руху об’єктів у кадрі. Для усунення критичної проблеми обмеженої пропускної здатності системної шини спроєктовано високоефективний механізм обміну даними на основі закріплених сторінок оперативної пам'яті. Це дало змогу створити архітектуру повністю асинхронного конвеєра, який дозволяє приховати неминучі фізичні затримки передачі інформації за часом виконання корисних математичних обчислень.

У третьому розділі здійснено успішну практичну програмну реалізацію розробленого алгоритму за допомогою синергії високорівневої мови програмування та спеціалізованої платформи обчислювальної архітектури графічних процесорів. Проведено стендове експериментальне дослідження продуктивності створеного програмного прототипу в умовах імітації екстремального математичного навантаження при безперервній обробці реального відеофайлу надвисокої чіткості. Експериментально показано, що застосування прямого доступу до пам'яті та делегування складної логіки графічному прискорювачу дозволяє скоротити час обробки одного кадру до лічених мілісекунд. Запропонований алгоритм перевершив показники швидкодії багатоядерного центрального процесора приблизно у 543-588 разів за наведеними вимірами. Запропоноване архітектурне рішення зменшує вплив апаратних затримок у межах тестового сценарію та потребує подальшої перевірки на повному циклі кодування відеопотоку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Darwich M., Bayoumi M. Cloud-Enhanced Video Streaming: Storage and Resource Management. *Enhancing Video Streaming with AI, Cloud, and Edge Technologies: Optimization Techniques and Frameworks.* Springer, 2025. P. 123–150.

2. Zhang D. Performance Comparison Study of Edge Detection Algorithms in Video Communication. *2025 International Conference on Electronics, Electrical and Grid Technology (ICEEGT 2025).* Atlantis Press, 2026. P. 512–521.

3. Kaplan M., Akman A. GPU-Accelerated Parallel Intra Prediction for High-Efficiency Video Coding (HEVC). 2025.

4. Salcedo-Navarro A. et al. Towards GPU-enabled serverless cloud edge platforms for accelerating HEVC video coding. *Cluster Computing.* 2025. Vol. 28, No. 1. P. 68.

5. Kaplan M., Akman A. Parallel implementation of discrete cosine transform (DCT) methods on GPU for HEVC. *International Congress of Electrical and Computer Engineering.* Springer, 2023. P. 281–293.

6. Shi J. Dynamic load balancing optimization model based on bidirectional network edge detection algorithm in cloud computing environment. *International Journal of Intelligent Computing and Cybernetics.* 2025. Vol. 18, No. 4. P. 613–631.

7. Wei W., Xu S. Efficient distributed adaptive transcoding: intelligent dynamic partitioning and virtual reference substream co-design. *Fourth International Conference on Machine Vision, Automatic Identification, and Detection (MVAID 2025).* SPIE, 2025. Vol. 13793. P. 576–580.

8. Badawy W. Evolution and future directions of video coding standards and emerging compression technologies. *Multimedia Tools and Applications.* 2025. Vol. 84, No. 41. P. 49607–49634.

9. James R. et al. Performance evaluation of all intra Kvazaar and x265 HEVC encoders on embedded system Nvidia Jetson platform. *Journal of Real-Time Image Processing.* 2024. Vol. 21, No. 3. P. 67.

10. Masykuroh K., Mulyana E. et al. Scaling bitrate in video compression using tuned parameters for efficient video analytics. *Telematics and Informatics Reports.* 2025. P. 100274.

11. Baital K., Chakrabarti A. A heterogeneous multi-core architectural model for video scheduling for transcoding in clouds. *International Journal of Information Technology.* 2024. Vol. 16, No. 5. P. 2831–2845.

12. Wu X. CPU-GPU Co-processing for Vector Search. ETH Zurich, 2023.

13. Peng Y. et al. SVFusion: A CPU-GPU Co-Processing Architecture for Large-Scale Real-Time Vector Search. *arXiv preprint arXiv:2601.08528.* 2026.

14. Duc T. L., Nguyen C., Östberg P.-O. Workload prediction for proactive resource allocation in large-scale cloud-edge applications. *Electronics.* 2025. Vol. 14, No. 16. P. 3333.

15. Krook T. AWS-Based Edge Computing Optimization for Industrial IoT Video Streams. 2025.

16. Carretero J. et al. *Euro-Par 2024: Parallel Processing.* Springer.

17. Rockenbach D. A. et al. GSParLib: A multi-level programming interface unifying OpenCL and CUDA for expressing stream and data parallelism. *Computer Standards & Interfaces.* 2025. Vol. 92. P. 103922.

18. Brantner V. Streaming in web-based AR: дис. … маг. – Universität Stuttgart, 2023.

19. Le Fevre P. Cloud-Native Hardware-Accelerated Live Video Transcoding in Kubernetes: Designing and implementing a failure-tolerant and scalable system. 2024.

20. Wei W., Zhang D. An adaptive concurrent transcoding system and method for adaptive bitrate streaming based on substream partitioning. *International Conference on Advanced Electronics, Intelligent Technology, and Computing (AEITC 2025).* SPIE, 2026. Vol. 14010. P. 289–293.

21. Moina-Rivera W. et al. Cloud media video encoding: review and challenges. *Multimedia Tools and Applications.* 2024. Vol. 83, No. 34. P. 81231–81278.

22. Deng Y., Liu J., Zhou Y. Research on Image Processing Resource Reconstruction Based on Load Balancing Strategy. *Electronics.* 2024. Vol. 13, No. 6. P. 1027.

23. Ogundipe A., Okunlola O., Alao O. Adaptive Load Balancing and Auto Scaling Algorithms for Resource Optimization in Distributed Microservices based Cloud Applications. *International Journal of Science Architecture Technology and Environment.* 2024. P. 101–111.

24. Bisht S. S. GPU Accelerated Image and Video Processing. 2023.

25. Han M. et al. Real-time, work-conserving GPU scheduling for concurrent DNN inference. *ACM Transactions on Computer Systems.* 2025. Vol. 44, No. 1. P. 1–42.

26. Abdelazim M. Improving Video Encoding Using Deep Learning Super Resolution: дис. … маг. – University of Portsmouth, 2024.

27. Cancellier L. H. D. L. et al. An asymmetric multi-layer visual signal compression approach combining handcrafted and learned solutions. 2025.

28. Zhang X. et al. Video compression artifact reduction by fusing motion compensation and global context in a swin-CNN based parallel architecture. *Proceedings of the AAAI Conference on Artificial Intelligence.* 2023. Vol. 37, No. 3. P. 3489–3497.

29. Van Rozendaal T. et al. Mobilenvc: Real-time 1080p neural video compression on a mobile device. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision.* 2024. P. 4323–4333.

30. Agha S. et al. Efficient motion estimation and discrete cosine transform implementation using the graphics processing units. *PloS One.* 2024. Vol. 19, No. 8. P. e0307217.

31. Aguilar-González A., Medina Santiago A. Ego-motion estimation for autonomous vehicles based on genetic algorithms and CUDA parallel processing. *Algorithms.* 2025. Vol. 18, No. 1. P. 19.

32. Smedberg J. Design and implementation of HEVC motion estimation on FPGA. 2024.

33. Song W., Li C., Zhang Q. Rapid CU partitioning and joint intra-frame mode decision algorithm. Electronics. 2024. Vol. 13, No. 17. P. 3465.

34. Yan L., Yin Z., Li J., Yang Y., Zhang T., Zhu F., Duan X., Schmidt B., Liu W. Rabbitsalign: Accelerating short-read alignment for CPU-GPU heterogeneous platforms. International Symposium on Bioinformatics Research and Applications. Springer, 2024. P. 83–94.

35. Teng H. et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211.* 2025.

36. Chung C., Park Y., Choi S., Ganbat M., Choo J. Shortcut-v2v: compression framework for video-to-video translation based on temporal redundancy reduction. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. P. 7612–7622.

37. Bender I. et al. Complexity and compression efficiency analysis of libaom AV1 video codec. *Journal of Real-Time Image Processing.* 2023. Vol. 20, No. 3. P. 50.

38. Costa V. et al. Hardware architecture design for power-efficient motion estimation over compressed block data. *Journal of Integrated Circuits and Systems.* 2025. Vol. 20, No. 1. P. 1–10.

39. Huo S. et al. Towards hybrid-optimization video coding. *ACM Computing Surveys.* 2024. Vol. 56, No. 9. P. 1–36.

40. Li J., Li B., Lu Y. Neural video compression with feature modulation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2024. P. 26099–26108.

41. Yang W., Huang H., Hu Y., Duan L.-Y., Liu J. Video coding for machines: Compact visual representation compression for intelligent collaborative analytics. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2024. Vol. 46, No. 7. P. 5174–5191.

42. Osama M., Porumbescu S. D., Owens J. D. A programming model for GPU load balancing. Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. 2023. P. 79–91.

43. Taufique Z., Miele A., Liljeberg P., Kanduri A. Adaptive workload distribution for accuracy-aware DNN inference on collaborative edge platforms. 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2024. P. 109–114.

44. Avan A., Azim A., Mahmoud Q. H. A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective. Electronics. 2023. Vol. 12, No. 12. P. 2599.

45. Yuan S., Zhou Q., Li J., Guo S., Chen H., Wu C., Yang Y. Adaptive incentive and resource allocation for blockchain-supported edge video streaming systems: A cooperative learning approach. IEEE Transactions on Mobile Computing. 2024. Vol. 24, No. 2. P. 539–556.

46. Dymora P., Mazurek M. General Computing Using CUDA Technology on NVIDIA GPU. Journal of Education, Technology and Computer Science. 2025. Vol. 6, No. 36. P. 186–197.

47. Liu Z., Wang Z., Tu S., Wang H., Fan J., Ren C. Real-Time Secure Video Streaming System Based on FPGA and CUDA Technology. Proceedings of the 2024 14th International Conference on Communication and Network Security. 2024. P. 146–152.

48. Thorén M. Comparative Analysis of Video Decoding Algorithms: CUDA vs. Vulkan Compute Shaders on Embedded Platforms. 2025.

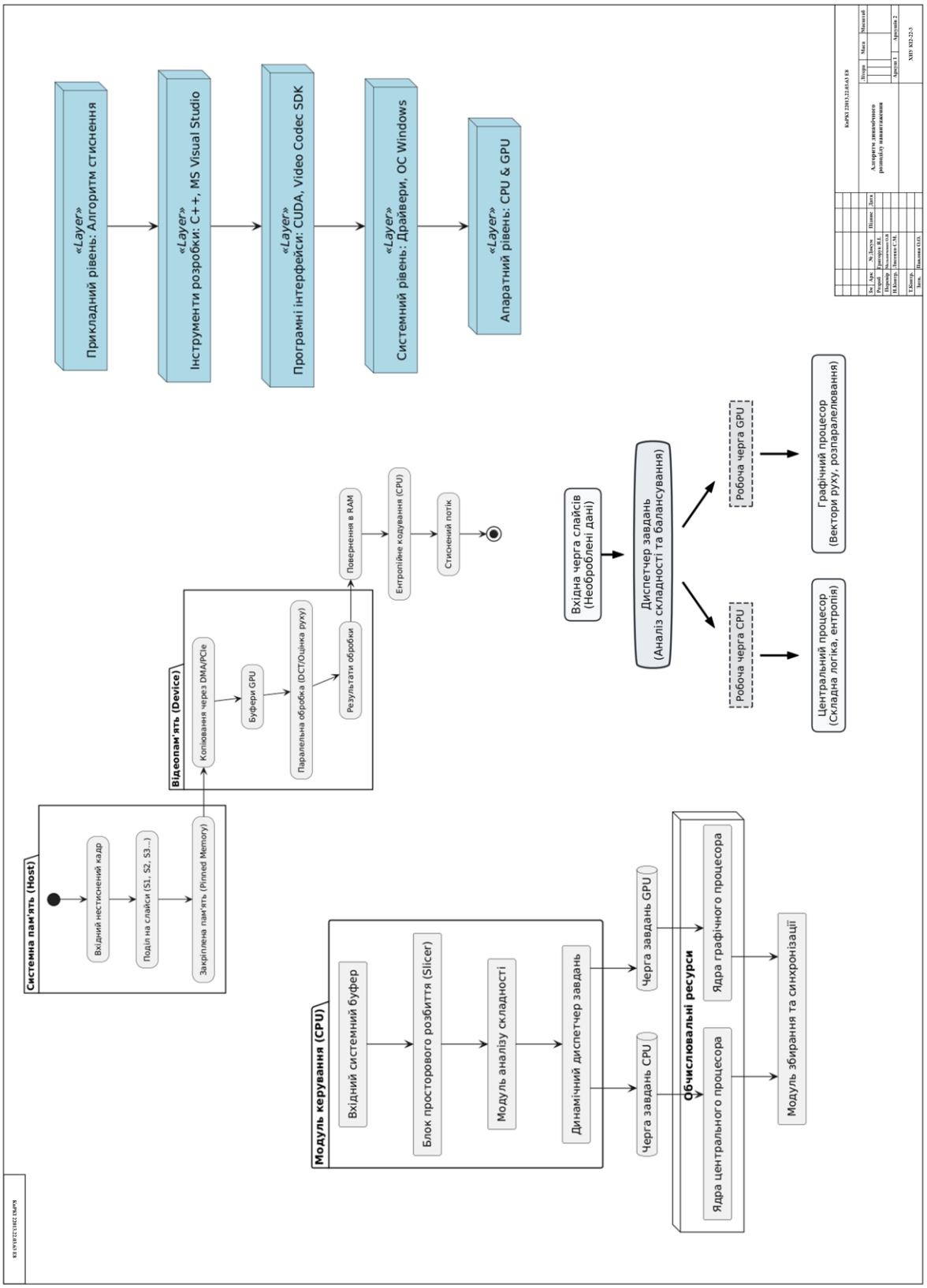
49. Thieu G. B., Gesper S., Payá-Vayá G. DCMA: Accelerating Parallel DMA Transfers with a Multi-Port Direct Cached Memory Access in a Massive-Parallel Vector Processor. ACM Transactions on Architecture and Code Optimization. 2025. Vol. 22, No. 2. P. 1–25.

50. Yu D., Chen R., Li X., Xiao M., Zhang G., Liu Y. A GPU-enabled real-time framework for compressing and rendering volumetric videos. IEEE Transactions on Computers. 2023. Vol. 73, No. 3. P. 789–800.

ДОДАТОК А

(обов’язковий)

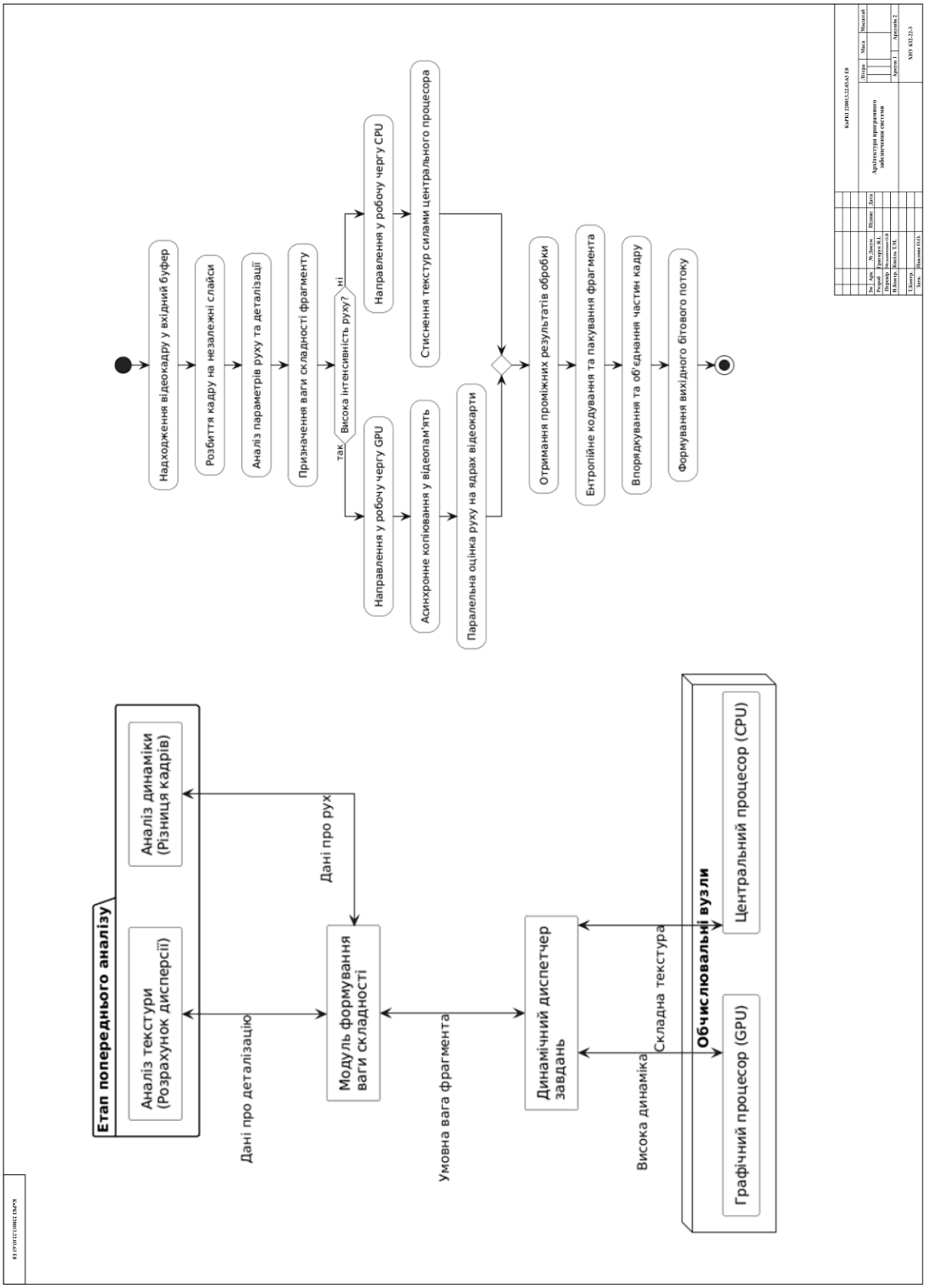
Копія креслення «Алгоритм динамічного розподілу навантаження»



ДОДАТОК Б

(обов’язковий)

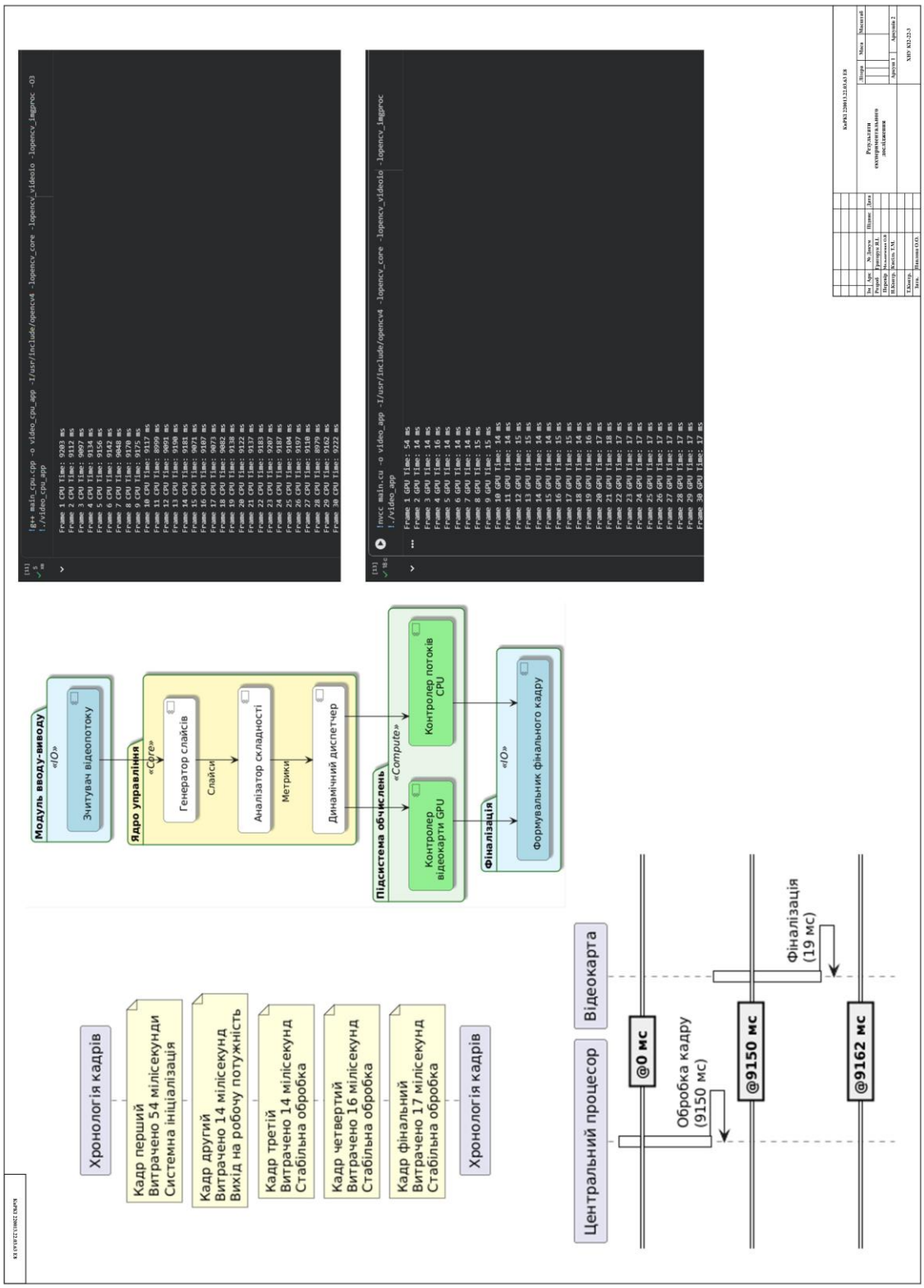
Копія креслення «Архітектура програмного забезпечення системи»



ДОДАТОК В

(обов’язковий)

Копія креслення «Результати експериментального дослідження»



Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 272552 Назва: БКР Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU Додано в БД: 2026-05-28 Автора: Ярослав ГРИГОРУК Керівники: Олександр МЕЛЬНИЧЕНКО Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	87534	616	1374 (2%)	20 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Ярослав ГРИГОРУК

Співавтор:

Назва: Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU

Експерт: Олександр МЕЛЬНИЧЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:2.45%

Коефіцієнт подібності 2:0.91%

Мікропробіли: 0

Заміна букв: 3

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-27 23:13:51.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-28 Доцент Андрій Нічепорук

Дата експерт

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU
Автор Ярослав ГРИГОРУК
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д-р філософії Олександр МЕЛЬНИЧЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 2,45%; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ Підпис — Андрій НІЧЕПОРУК (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Підпис — Олександр МЕЛЬНИЧЕНКО (Ім'я, ПРІЗВИЩЕ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Григорук Ярослав Ігорович

Тема: Паралельний алгоритм стиснення відеопотоку в реальному часі з розподілом навантаження між CPU та GPU

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 59

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є реалізація алгоритму стиснення відеопотоку в режимі реального часу з динамічним розподілом обчислювального навантаження між CPU та GPU.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено комплексний аналіз процесу обробки відеоданих високої роздільної здатності. Досліджено обмеження традиційних послідовних та суто апаратних методів стиснення, виявлено проблеми неефективного використання ресурсів та обґрунтовано доцільність застосування гетерогенних систем і просторового поділу кадрів. У другому розділі здійснено проєктування архітектури паралельної системи. Розроблено механізм динамічної диспетчеризації на основі попереднього аналізу складності фрагментів відео. Спроєктовано механізми швидкого обміну даними з використанням прямого доступу до пам'яті для усунення затримок системної шини. У третьому розділі виконано програмну реалізацію алгоритму з використанням мови C++, платформи CUDA та бібліотеки OpenCV. Проведено експериментальне стендове дослідження продуктивності створеного прототипу, яке підтвердило стабільну роботу асинхронного конвеєра та значне прискорення швидкості обробки кадрів порівняно з базовими методами.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: в експериментальній частині недостатньо уваги приділено тестуванню розробленого алгоритму на повному циклі кодування відеопотоку.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

к. т. н., доцент Корнуп О. В.

_____

"1" грудня 2026 р. _____ (підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Ярослав ГРИГОРУК

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-3

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року