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

КВАЛІФІКАЦІЙНА РОБОТА

Спеціалізований багатоядерний процесор підвищеної продуктивності
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 2301119.23.25.46 ПЗ

Виконав здобувач III курсу, гр. КІ2с-23-1 _____ Ярослав МУШИНСЬКИЙ
Підпис Ініціали, прізвище

Керівник _____ Олексій ЛИГУН
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер _____ Сергій ЛИСЕНКО
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

Мушинському Ярославу Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Спеціалізований багатоядерний процесор підвищеної продуктивності
Керівник проекту (роботи) Лигун Олексій Олегович, асистент
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)

Кіберфізична система спеціалізованого багатоядерного процесора підвищеної продуктивності та постановка задачі щодо його удосконалення

Проєктування системи обробки інформації у кіберфізичній системі спеціалізованого багатоядерного процесора підвищеної продуктивності

Програмно-апаратна реалізація кіберфізичної системи спеціалізованого багатоядерного процесора підвищеної продуктивності

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

Архітектура ПЗ проєкту

Архітектура ПЗ спеціалізованого процесору

Апаратне забезпечення проєкту

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 проекту (роботи)	Термін виконання етапів проекту (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 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір компонентів для проєктування спеціалізованого багатоядерного процесору підвищеної продуктивності	01.04.2026	виконано
5	Робота над розділом 3 – проєктування спеціалізованого багатоядерного процесору підвищеної продуктивності	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач_____ ___Ярослав МУШИНСЬКИЙ__
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи_____ ____Олексій ЛИГУН ___
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 2301119.23.25.46 ПЗ	Пояснювальна записка	64		
			Графічні матеріали			
2		КвРКІ 2301119.23.25.46 Е8	Архітектура ПЗ проєкту	1		
3		КвРКІ 2301119.23.25.46 Е8	Архітектура ПЗ	1		
			спеціалізованого			
			процесору			
4		КвРКІ 2301119.23.25.46 Е8	Апаратне забезпечення	1		
			проєкту			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 2301119.23.25.46 ВП		
Розробив		Мушинський			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Лигун				У	
Н. контр.		Лисенко				ХНУ, КІ2с-23-1	
Затв.		Павлова		01.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Спеціалізований багатоядерний процесор підвищеної продуктивності».

Автор роботи: Ярослав МУШИНСЬКИЙ.

Керівник роботи: Олексій ЛИГУН.

Пояснювальна записка: 64 с., 11 рис., 4 табл., 3 дод., 55 джерел.

Графічна частина: 3 креслення.

АРХІТЕКТУРА, БАГАТОЯДЕРНИЙ ПРОЦЕСОР, КЕШ-ПАМ'ЯТЬ, МОДЕЛЮВАННЯ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, ПРОДУКТИВНІСТЬ, СИНХРОНІЗАЦІЯ.

Кваліфікаційна робота бакалавра присвячена розробленню та аналізу програмної моделі спеціалізованого багатоядерного процесора підвищеної продуктивності. Актуальність теми зумовлена зростанням потреби у швидкій обробці великих обсягів даних, виконанні паралельних обчислень та підвищенні ефективності використання обчислювальних ресурсів. У сучасних процесорних системах подальше збільшення тактової частоти має фізичні та енергетичні обмеження.

Метою роботи є моделювання спеціалізованого багатоядерного процесора та оцінювання його продуктивності під час виконання обчислювальних задач, що можуть бути поділені на окремі фрагменти. Для досягнення поставленої мети проаналізовано особливості сучасних багатоядерних архітектур, розглянуто принципи паралельного виконання, визначено роль між'ядерної взаємодії, кеш-пам'яті, синхронізації та накладних витрат.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Сучасний розвиток інформаційних технологій супроводжується постійним зростанням обсягів даних та ускладненням обчислювальних задач. Обробка потокової інформації, моделювання фізичних процесів, машинне навчання, криптографічні алгоритми та системи реального часу вимагають високої обчислювальної продуктивності. Традиційне підвищення тактової частоти процесорів вже не забезпечує очікуваного приросту швидкодії через фізичні обмеження, зокрема тепловиділення та енергоспоживання. Унаслідок цього основним напрямом розвитку обчислювальної техніки стало впровадження багатоядерних архітектур.

Багатоядерні процесори дозволили перейти від лінійного масштабування продуктивності до паралельної обробки задач. Розподіл навантаження між декількома обчислювальними ядрами забезпечив зменшення часу виконання програм та підвищення ефективності використання апаратних ресурсів. Разом із тим універсальні багатоядерні архітектури не завжди забезпечують максимальну ефективність у спеціалізованих прикладних задачах, оскільки орієнтовані на широкий спектр застосувань. Це створює передумови для розроблення спеціалізованих багатоядерних процесорів, архітектура яких оптимізована під конкретні типи обчислень.

У межах цієї кваліфікаційної роботи розглянуто питання створення моделі спеціалізованого багатоядерного процесора підвищеної продуктивності. Особливу увагу приділено архітектурним рішенням, що забезпечують ефективну взаємодію ядер, оптимізацію доступу до пам'яті та зменшення затримок при міжпроцесорному обміні даними. Проведено аналіз сучасних підходів до організації паралельних обчислень, принципів масштабованості та методів оцінювання продуктивності багатоядерних систем.

Актуальність теми визначається потребою у високопродуктивних обчислювальних засобах, здатних забезпечити ефективну роботу у середовищах

з інтенсивною паралельною обробкою даних. Розвиток технологій систем на кристалі, мереж-на-кристалі та чиплетних архітектур відкрив нові можливості для створення спеціалізованих процесорних структур. У підсумку дослідження архітектурних рішень у сфері багатоядерних систем дозволяє підвищити рівень розуміння сучасних тенденцій комп'ютерної інженерії та сформувати практичні навички проєктування складних обчислювальних структур.

Метою кваліфікаційної роботи є моделювання спеціалізованого багатоядерного процесора та проведення аналізу його продуктивності з урахуванням особливостей архітектурної організації та паралельного виконання задач.

Для досягнення поставленої мети визначено такі завдання: проаналізувати сучасні архітектури багатоядерних процесорів; обґрунтувати вибір структурної організації спеціалізованої системи; розробити модель процесора; провести експериментальне моделювання; оцінити приріст продуктивності та ефективність масштабування.

Об'єктом кваліфікаційної роботи є процес функціонування багатоядерних обчислювальних систем.

Предметом кваліфікаційної роботи є архітектурні принципи побудови спеціалізованого багатоядерного процесора та методи аналізу його продуктивності.

У результаті виконання роботи сформовано модель багатоядерної архітектури, проведено експериментальні дослідження її роботи та визначено фактори, що впливають на ефективність паралельного виконання. Отримані результати створюють основу для подальшого вдосконалення спеціалізованих процесорних систем і впровадження їх у задачах підвищеної обчислювальної складності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Еволюція процесорних архітектур та передумови переходу до багатоядерності

Розвиток процесорних архітектур протягом останніх десятиліть відображає постійне прагнення до підвищення обчислювальної продуктивності [3, 12]. На ранніх етапах становлення обчислювальної техніки основна увага зосереджувалась на збільшенні тактової частоти, удосконаленні мікроархітектури та розширенні набору команд [7]. Одноядерні процесори тривалий час залишалися домінуючими завдяки поступовому зменшенню технологічних норм виробництва напівпровідникових кристалів та збільшенню кількості транзисторів на чипі [9, 14]. Це дозволяло реалізовувати складніші конвеєрні структури, збільшувати розрядність регістрів, впроваджувати механізми позачергового виконання інструкцій та прогнозування переходів [11].

У межах традиційної парадигми підвищення продуктивності досягалося шляхом глибшого конвеєрування та збільшення частоти роботи ядра [6]. Однак з часом такий підхід почав стикатися з фізичними обмеженнями [15]. Зменшення розміру транзисторів супроводжувалося зростанням щільності тепловиділення, що ускладнювало відведення тепла та вимагало збільшення енергоспоживання [18, 22]. Унаслідок цього подальше нарощування тактової частоти перестало забезпечувати пропорційний приріст продуктивності. Виникла так звана “теплова стіна”, яка обмежила можливість екстенсивного розвитку одноядерних систем [19].

Паралельно із цим спостерігалося зростання складності програмного забезпечення [25]. Сучасні застосування почали активно використовувати багатопотокові алгоритми, обробку великих обсягів даних та мережеві обчислення [27, 31]. Одноядерні процесори, навіть із високою тактовою частотою, не могли ефективно обслуговувати такі навантаження [30]. Це призвело до зміни концепції підвищення продуктивності – від збільшення

швидкості виконання однієї інструкції до одночасного виконання декількох потоків обчислень [28].

Перехід до багатоядерних архітектур став логічним етапом розвитку [33, 36]. Розміщення кількох обчислювальних ядер на одному кристалі дозволило розподіляти навантаження між ними, зменшувати час виконання паралельних задач та підвищувати загальну пропускну здатність системи [34]. Такий підхід забезпечив більш раціональне використання транзисторного бюджету: замість ускладнення одного ядра зростала кількість відносно простіших обчислювальних блоків [37].

На цьому етапі архітектурні рішення почали враховувати не лише швидкодію окремого ядра, а й ефективність взаємодії між ядрами [40]. Виникла необхідність у впровадженні механізмів узгодженості кеш-пам'яті, оптимізації доступу до спільних ресурсів та мінімізації затримок міжпроцесорного обміну [42, 45]. З'явилися нові концепції організації внутрішніх шин, між'ядерних мереж та ієрархій пам'яті [44]. Розвиток технологій «мережа-на-кристалі» дозволив масштабувати кількість ядер без істотного зростання латентності [47, 52]. Загальна архітектура багатоядерного процесора зображена на рисункку 1.1.

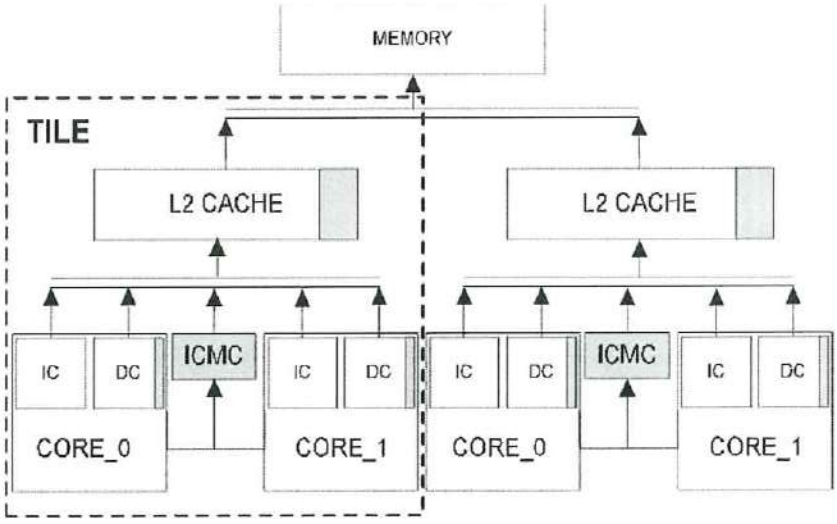


Рисунок 1.1 - Загальна архітектура багатоядерного процесора [58]

Важливим фактором переходу до багатоядерності стало також прагнення підвищити енергоефективність. Робота кількох ядер на помірній частоті виявилася енергетично вигіднішою, ніж функціонування одного високочастотного ядра. Такий підхід дозволив зменшити пікове тепловиділення та підвищити стабільність роботи системи. У результаті багатоядерність стала не лише засобом підвищення продуктивності, а й інструментом оптимізації енергоспоживання.

Подальший розвиток архітектур привів до формування гетерогенних систем, у яких поєднуються ядра різного типу та призначення. Це дозволило адаптувати обчислювальні ресурси під конкретні задачі – від інтенсивних арифметичних операцій до обробки сигналів або графіки. Зростання кількості ядер, у свою чергу, поставило питання масштабованості та ефективності паралельних алгоритмів, що безпосередньо пов'язано з теоретичними обмеженнями прискорення.

У підсумку еволюція процесорних архітектур продемонструвала закономірний перехід від частотного масштабування до структурного паралелізму. Фізичні, теплові та енергетичні обмеження одноядерних систем сформували передумови для розвитку багатоядерних рішень. Саме ці фактори визначили актуальність створення спеціалізованих багатоядерних процесорів, архітектура яких оптимізована під конкретні обчислювальні сценарії та забезпечує підвищену продуктивність у межах заданого класу задач.

1.2 Класифікація багатоядерних процесорів та їх архітектурні особливості

Зростання кількості ядер у межах одного кристала зумовило появу різних підходів до їх організації та взаємодії [33, 36]. Багатоядерні процесори відрізняються не лише кількістю обчислювальних блоків, а й способом розподілу ресурсів, типом пам'яті, топологією внутрішнього з'єднання та рівнем однорідності архітектури [34]. У результаті сформувалася класифікація, що

дозволяє систематизувати сучасні процесорні рішення з позиції їх структурної організації [37].

Одним із базових критеріїв класифікації є однорідність ядер [40]. У симетричних багатоядерних процесорах усі ядра мають однакову мікроархітектуру, тактову частоту та функціональні можливості [41]. Прикладом такого підходу є більшість серверних процесорів, де кожне ядро здатне виконувати повний набір інструкцій та обслуговувати операційну систему [43]. У подібних системах спрощується балансування навантаження, оскільки будь-яке ядро може виконувати довільний потік [44]. Однак при цьому не враховуються особливості різних типів задач, що може знижувати енергоефективність [45].

Гетерогенні багатоядерні архітектури передбачають поєднання ядер різного типу [46, 48]. У таких системах використовуються, наприклад, високопродуктивні ядра для інтенсивних обчислень та енергоощадні ядра для фонових процесів [49]. Подібна організація дозволяє оптимізувати співвідношення продуктивності та енергоспоживання [50]. Крім того, до гетерогенних систем відносяться процесори, що поєднують центральні ядра з графічними або спеціалізованими обчислювальними модулями, призначеними для векторних операцій або машинного навчання [51, 53].

Наступним критерієм є організація пам'яті [42]. Багатоядерні системи зі спільною пам'яттю передбачають, що всі ядра мають доступ до єдиного адресного простору [41]. Такий підхід спрощує програмування, оскільки дані можуть вільно передаватися між потоками [44]. Водночас виникає потреба у механізмах узгодженості кеш-пам'яті, які забезпечують коректність доступу до змінних [45, 47]. Схема багатоядерних систем зі спільною пам'яттю зображена на рисунку 1.2

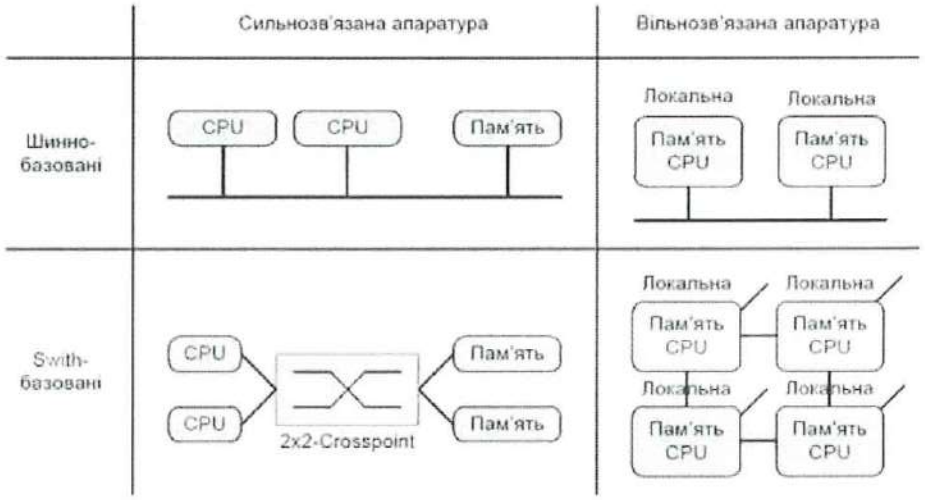


Рисунок 1.2 - Багатоядерні системи зі спільною пам'яттю [59]

У системах із розподіленою пам'яттю кожне ядро або група ядер має власний локальний сегмент пам'яті, а взаємодія відбувається через механізми обміну повідомленнями [42, 46]. Подібна модель краще масштабується при великій кількості ядер, але ускладнює розробку програмного забезпечення [48].

Важливим архітектурним аспектом є топологія між'ядерного з'єднання [44]. На початкових етапах розвитку використовувалася спільна шина, через яку здійснювався обмін даними між ядрами та пам'яттю [41]. Зі збільшенням кількості ядер пропускна здатність шини ставала вузьким місцем [43]. Для подолання цього обмеження почали впроваджувати кільцеві та сіткові структури, а згодом – мережі-на-кристалі (Network-on-Chip) [47, 52]. Сіткова топологія дозволяє розподілити навантаження та зменшити затримки при передачі даних між віддаленими ядрами, що особливо актуально для систем із десятками або сотнями обчислювальних блоків [53].

Окремий клас становлять багатоядерні процесори з кластерною організацією [49]. У таких системах ядра об'єднуються у групи, що мають спільний кеш другого або третього рівня [45]. Це дозволяє зменшити латентність доступу до даних усередині кластера та знизити навантаження на глобальну між'ядерну мережу [50]. Подібний підхід застосовується у високопродуктивних

серверних рішеннях, де важливим є баланс між швидкодією та масштабованістю [36, 40].

З точки зору мікроархітектури ядра можуть бути як скалярними, так і супервекторними [11, 27]. Сучасні багатоядерні процесори часто поєднують багатопоточність на рівні ядра з внутрішнім паралелізмом виконання інструкцій [28]. Використання технологій одночасної багатопоточності дозволяє одному ядру обслуговувати декілька потоків, що підвищує завантаження функціональних блоків [31]. У поєднанні з багатоядерністю це створює багаторівневу систему паралельної обробки [33].

Особливу увагу приділяють організації кеш-ієрархії [45, 47]. Типовою є трирівнева структура, у якій кеш першого рівня є приватним для кожного ядра, кеш другого рівня може бути як приватним, так і спільним для групи ядер, а кеш третього рівня часто є загальним для всього процесора [42]. Від обраної конфігурації залежить швидкість доступу до даних та рівень між'ядерних конфліктів [44]. Неправильна організація кешу може нівелювати переваги збільшення кількості ядер [45].

У сучасних спеціалізованих рішеннях дедалі частіше застосовується модульний підхід із використанням чиплетної архітектури [50, 54]. Обчислювальні ядра розміщуються на окремих кристалах, які з'єднуються високошвидкісними інтерфейсами [52]. Це дозволяє масштабувати кількість ядер без суттєвого ускладнення виробничого процесу та забезпечує гнучкість конфігурації [53].

У підсумку багатоядерні процесори класифікуються за однорідністю ядер, типом організації пам'яті, топологією внутрішнього з'єднання, способом побудови кеш-ієрархії та рівнем спеціалізації функціональних блоків [34, 37]. Кожен із зазначених факторів безпосередньо впливає на продуктивність, масштабованість та енергоефективність системи [36, 40]. Саме врахування цих архітектурних особливостей створює основу для розроблення спеціалізованого багатоядерного процесора підвищеної продуктивності [52].

1.3 Принципи паралельних обчислень і масштабованість

Перехід до багатоядерних архітектур поставив у центр уваги не лише апаратну організацію процесора, а й принципи паралельного виконання програм [28, 33]. Саме здатність алгоритмів ефективно розподіляти обчислення між кількома ядрами визначає реальний приріст продуктивності [30]. У межах багатоядерної системи швидкодія залежить не стільки від кількості ядер, скільки від рівня паралелізації задачі та мінімізації накладних витрат на синхронізацію [31, 36].

Паралельні обчислення передбачають одночасне виконання декількох незалежних або частково незалежних фрагментів програми [27]. Виділяють кілька рівнів паралелізму [25]. На рівні інструкцій використовується можливість одночасного виконання кількох операцій усередині одного ядра завдяки конвеєризації, позачерговому виконанню та векторним розширенням [11, 12]. На рівні потоків або задач реалізується розподіл роботи між різними ядрами [28]. Окремо розглядається паралелізм на рівні даних, коли одна й та сама операція застосовується до великого масиву елементів [27, 31]. У багатоядерних процесорах ці підходи поєднуються, формуючи багаторівневу структуру паралельного виконання [33].

Ефективність паралельних обчислень кількісно оцінюється показником прискорення, яке визначається як відношення часу виконання задачі на одноядерній системі до часу виконання на багатоядерній конфігурації [30]. Ідеальним вважається лінійне масштабування, коли подвоєння кількості ядер призводить до подвоєння продуктивності [36]. Проте на практиці досягти такого результату складно через наявність послідовних ділянок коду та витрат на координацію потоків [25].

Теоретичну межу прискорення описує закон Амдала [24]. Він встановлює, що загальний приріст швидкодії обмежений часткою послідовної частини програми [24, 25]. Якщо певна частина алгоритму не може бути виконана

паралельно, вона стає визначальним фактором продуктивності незалежно від кількості ядер [24]. Збільшення числа обчислювальних блоків у такому випадку дає дедалі менший ефект [30]. У противагу цьому закон Густавсона враховує можливість збільшення розміру задачі при зростанні кількості ядер [26]. У масштабних обчисленнях, де обсяг даних пропорційно зростає разом із ресурсами, ефективність паралельної обробки може залишатися високою [26, 36].

Практична реалізація паралельних обчислень супроводжується низкою проблем [31]. Однією з ключових є синхронізація потоків [28]. Для забезпечення коректності роботи використовуються механізми блокувань, бар'єрів та атомарних операцій [27, 31]. Надмірне використання таких засобів призводить до втрат продуктивності через простої ядер [30]. Ще одним фактором є конкуренція за спільні ресурси, зокрема за кеш-пам'ять та канали доступу до оперативної пам'яті [45, 47]. Конфлікти доступу здатні звести нанівець виграш від паралелізації [44].

Особливу роль відіграє балансування навантаження [28]. Якщо розподіл задач між ядрами є нерівномірним, частина ядер залишається незавантаженою, що знижує ефективність використання апаратних ресурсів [33]. Для мінімізації цього явища застосовуються динамічні алгоритми планування, які перерозподіляють роботу в процесі виконання [31].

Масштабованість багатоядерної системи визначається її здатністю зберігати або підвищувати ефективність при збільшенні кількості ядер [36]. Вона залежить від архітектури внутрішнього з'єднання, пропускної здатності пам'яті та характеристик кеш-ієрархії [44, 47]. У системах із невеликою кількістю ядер прості топології забезпечують прийнятну продуктивність, однак при десятках або сотнях ядер виникає потреба у складніших мережевих структурах [52]. Наявність вузьких місць у підсистемі пам'яті часто стає основним обмеженням масштабування [45].

Додатковим фактором є енергоефективність [50]. Зі збільшенням кількості ядер зростає сумарне енергоспоживання, тому масштабованість має оцінюватися не лише з позиції швидкодії, а й з урахуванням співвідношення продуктивності до споживаної потужності [49, 50]. Спеціалізовані багатоядерні архітектури дозволяють оптимізувати це співвідношення шляхом адаптації структури під конкретний клас задач [53].

У підсумку принципи паралельних обчислень визначають межі досяжної продуктивності багатоядерних процесорів [30, 36]. Реальний приріст швидкодії формується як результат взаємодії апаратних можливостей та властивостей алгоритмів [28]. Розуміння закономірностей масштабування створює основу для розроблення спеціалізованої багатоядерної архітектури, у якій обмеження традиційних підходів мінімізовано завдяки оптимізації структури та організації обміну даними [33, 52].

1.4 Організація між'ядерної взаємодії та узгодженість кеш-пам'яті

Зі збільшенням кількості ядер у межах одного процесора ключовим питанням стає не лише швидкодія окремого обчислювального блоку, а й ефективність їхньої взаємодії [33, 40]. Навіть за наявності великої кількості ядер загальна продуктивність системи може обмежуватися затримками обміну даними, конфліктами доступу до пам'яті та неузгодженістю кешів [44, 45]. Саме тому організація між'ядерної взаємодії є одним із визначальних факторів архітектурної ефективності багатоядерного процесора [36].

У найпростіших багатоядерних системах взаємодія ядер реалізується через спільну системну шину [41]. Кожне ядро підключається до єдиного каналу передачі даних, через який здійснюється доступ до оперативної пам'яті та обмін інформацією [42]. Такий підхід є простим у реалізації, однак із зростанням кількості ядер виникає перевантаження шини, що призводить до зростання

латентності та зменшення пропускної здатності [43]. У результаті шина стає вузьким місцем системи [44].

Для подолання цього обмеження застосовуються кільцеві та сіткові топології [47]. У кільцевій структурі кожне ядро підключене до замкненого каналу, по якому передаються запити та дані [52]. Це дозволяє розподілити навантаження більш рівномірно, однак при великій кількості вузлів зростає середня довжина маршруту передачі [53]. Сіткова або двовимірна решіткова топологія використовується у багатоядерних процесорах із десятками ядер [52, 54]. У такій архітектурі кожне ядро з'єднане з кількома сусідніми вузлами, що забезпечує більш масштабовану структуру передачі даних [47]. Подібні рішення відносяться до класу мереж-на-кристалі, де обмін інформацією здійснюється пакетним способом [52].

Окрім фізичної топології, важливою є логіка узгодження доступу до спільної пам'яті [45]. У багатоядерних процесорах кожне ядро зазвичай має власний кеш першого рівня, а іноді й другого рівня [42]. За відсутності механізму узгодженості виникає ризик, що різні ядра працюватимуть із різними копіями одних і тих самих даних [45]. Це може призвести до логічних помилок та некоректного виконання програм [44].

Для забезпечення коректності використовується протокол узгодженості кеш-пам'яті [45, 47]. Найпоширенішим є протокол типу MESI, у межах якого кожен рядок кешу перебуває в одному з кількох станів: модифікований, ексклюзивний, спільний або недійсний [47]. При зміні даних одним ядром відповідні копії в кешах інших ядер або оновлюються, або переводяться в недійсний стан [45]. Такий механізм гарантує, що всі ядра працюють із актуальними значеннями змінних [44].

Зі зростанням кількості ядер реалізація узгодженості ускладнюється [52]. У системах із невеликою кількістю ядер застосовується централізований контроль, коли спеціальний модуль відстежує стан кеш-рядків [47]. Однак при масштабуванні до десятків ядер централізована схема створює надмірне

навантаження [52]. Тому впроваджуються розподілені протоколи, у яких контроль здійснюється безпосередньо між ядрами або через розподілені каталоги станів [53, 54].

Окремим аспектом є ієрархія кеш-пам'яті [42, 45]. Зазвичай кеш першого рівня є приватним для кожного ядра, що мінімізує затримки доступу [42]. Кеш другого рівня може бути як приватним, так і спільним для групи ядер [45]. Кеш третього рівня часто організовується як загальний ресурс, доступний для всіх обчислювальних блоків [44]. Така багаторівнева структура дозволяє зменшити частоту звернень до оперативної пам'яті, але водночас потребує ефективного механізму синхронізації [47].

Важливим чинником є латентність між'ядерного обміну [52]. У випадку частих операцій із спільними даними затримки, пов'язані з передачею кеш-рядків, можуть суттєво впливати на продуктивність [45]. Особливо це проявляється при реалізації механізмів блокування або при роботі з глобальними змінними [31, 44]. Зменшення таких витрат досягається шляхом оптимізації розміщення даних, використання локальних буферів та мінімізації спільного доступу [30].

У підсумку організація між'ядерної взаємодії визначає масштабованість і реальну ефективність багатоядерної системи [36, 52]. Навіть за наявності значної кількості обчислювальних ядер відсутність ефективного механізму узгодженості кешу та оптимальної топології обміну даними може суттєво обмежити продуктивність [45]. Саме тому при розробленні спеціалізованого багатоядерного процесора підвищеної продуктивності необхідно приділяти особливу увагу архітектурі внутрішнього з'єднання та механізмам забезпечення цілісності пам'яті [47, 53].

1.5 Методи моделювання багатоядерних систем

Проєктування багатоядерних процесорів неможливе без попереднього моделювання їх архітектури та оцінювання поведінки під навантаженням [55, 56]. Реалізація повноцінного апаратного прототипу на ранніх етапах розробки є складною та ресурсомісткою, тому дослідження властивостей системи здійснюється за допомогою моделей різного рівня деталізації [55]. Вибір методу моделювання визначається метою аналізу: оцінювання загальної продуктивності, дослідження латентності пам'яті, перевірка протоколів узгодженості або аналіз енергоефективності [36, 45].

Найбільш узагальненим є функціональне моделювання [55]. На цьому рівні відтворюється логіка виконання інструкцій без детального врахування часових характеристик [12]. Функціональна модель дозволяє перевірити коректність архітектурних рішень, структуру команд та взаємодію модулів [55]. Перевагою такого підходу є висока швидкість симуляції, що дає змогу аналізувати великі обсяги тестових даних [56]. Однак функціональна модель не забезпечує точного оцінювання затримок і не дозволяє дослідити вплив мікроархітектурних факторів на продуктивність [30].

Більш детальним є часово-орієнтоване або цикл-точне моделювання [56]. У цьому випадку враховується тривалість виконання кожної інструкції, затримки доступу до кеш-пам'яті, особливості конвеєра та механізми синхронізації [45, 47]. Цикл-точні моделі забезпечують високу достовірність результатів і дозволяють аналізувати вузькі місця системи [36]. Водночас швидкість симуляції зменшується, оскільки відтворюється кожен такт роботи процесора [55].

Окремим напрямом є апаратне моделювання з використанням мов опису апаратури [56, 57]. На цьому рівні формується структурна модель багатоядерної системи, що відображає з'єднання між ядрами, кешами та контролерами пам'яті [42]. Такий підхід дозволяє досліджувати поведінку системи на рівні логічних

елементів і перевіряти правильність реалізації протоколів узгодженості [45, 47]. Апаратні моделі можуть бути синтезовані для програмованих логічних інтегральних схем, що дає змогу наблизити експеримент до реальних умов функціонування [57].

Для оцінювання продуктивності багатоядерних систем широко застосовується програмна симуляція [55]. У межах цього підходу створюється програмна модель архітектури, яка відтворює структуру ядер, ієрархію пам'яті та внутрішню мережу з'єднань [52]. Такий метод дозволяє варіювати кількість ядер, параметри кешу та частоту обміну даними без фізичної зміни апаратури [36]. Програмна симуляція є гнучким інструментом для аналізу масштабованості та визначення оптимальної конфігурації [33].

Важливою складовою моделювання є вибір метрик оцінювання [30]. До основних показників належать час виконання задачі, коефіцієнт прискорення, ефективність використання ядер, рівень завантаження кеш-пам'яті та пропускна здатність між'ядерного каналу [36, 45]. Для комплексного аналізу доцільно враховувати також енергоспоживання та теплові характеристики, оскільки вони безпосередньо впливають на можливість масштабування системи [49, 50].

Під час моделювання багатоядерних архітектур необхідно враховувати характер навантаження [31]. Результати, отримані для обчислювально-інтенсивних задач, можуть суттєво відрізнятися від результатів для задач із частими операціями введення-виведення або активним використанням спільної пам'яті [28, 44]. Тому доцільним є формування набору тестових сценаріїв, які відображають різні типи паралельних алгоритмів [27].

У підсумку методи моделювання багатоядерних систем охоплюють функціональні, цикл-точні та структурні підходи, кожен із яких забезпечує різний рівень деталізації [55, 56]. Комплексне застосування цих методів дозволяє оцінити як логічну коректність архітектури, так і її реальну продуктивність [36]. Саме моделювання створює основу для обґрунтованого вибору параметрів

спеціалізованого багатоядерного процесора та визначення його потенціалу підвищення швидкодії в умовах паралельного виконання задач [33, 52].

1.6 Постановка задачі

Проведений аналіз еволюції процесорних архітектур, класифікації багатоядерних систем, принципів паралельних обчислень і методів моделювання показав, що підвищення продуктивності сучасних процесорів уже не може розглядатися лише як питання збільшення тактової частоти або формального додавання нових ядер. У багатоядерних системах кінцевий результат залежить від значно ширшого набору чинників, серед яких важливими є організація пам'яті, спосіб передавання даних між ядрами, ефективність розподілу задачі, наявність послідовних ділянок виконання, витрати на синхронізацію та здатність архітектури зберігати прийнятну швидкодію при масштабуванні. Через це кількість ядер сама по собі не гарантує пропорційного приросту продуктивності, якщо інші елементи архітектури не узгоджені між собою.

Основними обмеженнями, які необхідно врахувати під час постановки задачі, є вузькі місця підсистеми пам'яті, затримки між'ядерної взаємодії, нерівномірність завантаження ядер і витрати на синхронізацію результатів. Якщо кілька ядер одночасно звертаються до спільної пам'яті, виникають черги запитів, що збільшує час очікування. Якщо задача поділена між ядрами нерівномірно, частина обчислювальних блоків завершує роботу раніше та просто очікує завершення інших. Якщо ж синхронізація виконується занадто часто, виграш від паралельного виконання частково втрачається. Через це при розробленні моделі спеціалізованого багатоядерного процесора потрібно врахувати не тільки обчислювальну частину, а й усі допоміжні витрати, які впливають на реальний час виконання.

У межах кваліфікаційної роботи сформульовано задачу розроблення програмної моделі спеціалізованого багатоядерного процесора підвищеної

продуктивності з подальшим аналізом його роботи в умовах паралельного виконання обчислювальних задач. Запропонована модель має відображати основні елементи багатоядерної архітектури, зокрема обчислювальні ядра, локальну кеш-пам'ять, спільну пам'ять, блок керування задачами, механізм розподілу фрагментів і модуль оцінювання продуктивності. При цьому модель не повинна надмірно ускладнюватися деталями промислової мікроархітектури, оскільки головним завданням є оцінювання впливу кількості ядер, розміру задачі та накладних витрат на загальну швидкодію.

У межах поставленої задачі також передбачено врахування затримок пам'яті. Для цього в моделі має бути відокремлено локальний доступ до кеш-пам'яті та звернення до спільної пам'яті. Локальний кеш дозволяє зменшити кількість звернень до загального ресурсу, а спільна пам'ять забезпечує зберігання вхідних даних і підсумкових результатів.

Об'єктом кваліфікаційної роботи є процес функціонування багатоядерної обчислювальної системи під час виконання паралельних задач. Предметом кваліфікаційної роботи є архітектурні рішення, параметри моделі та способи організації паралельного виконання, що визначають продуктивність спеціалізованого багатоядерного процесора.

Очікуваним результатом є програмна модель спеціалізованого багатоядерного процесора, яка дає змогу імітувати розподіл задачі між ядрами, враховувати витрати на пам'ять і синхронізацію, а також отримувати кількісні показники продуктивності. Така модель дозволяє оцінити, як змінюється час виконання при збільшенні кількості ядер, які обмеження виникають при масштабуванні та за яких умов багатоядерна архітектура забезпечує найбільший приріст швидкодії. У підсумку поставлена задача спрямована на поєднання архітектурного опису процесора з практичним моделюванням його роботи, що створює основу для подальшого аналізу продуктивності спеціалізованих обчислювальних систем.

2 ПРОЄКТУВАННЯ МОДЕЛІ СПЕЦІАЛІЗОВАНОГО БАГАТОЯДЕРНОГО ПРОЦЕСОРА

2.1 Формування загальної структури спеціалізованого багатоядерного процесора

сформовано загальну структуру спеціалізованого багатоядерного процесора підвищеної продуктивності, який орієнтований на паралельне виконання обчислювальних задач. Основна ідея побудови такої архітектури полягає не лише у збільшенні кількості обчислювальних ядер, а й у правильній організації їх взаємодії між собою, підсистемою пам'яті та блоком керування. Саме тому процесор розглянуто як цілісну обчислювальну систему, у якій кожен функціональний блок виконує власне завдання та водночас працює у зв'язку з іншими компонентами.

Спеціалізований багатоядерний процесор у цій кваліфікаційній роботі подано як архітектуру, що складається з кількох однотипних обчислювальних ядер, локальної кеш-пам'яті, спільної пам'яті, контролера пам'яті, внутрішньої системи обміну даними та блока керування виконанням задач. Такий склад компонентів дозволяє змоделювати роботу процесора на рівні, достатньому для подальшого аналізу продуктивності. При цьому модель не перевантажується зайвими апаратними деталями, які не мають прямого впливу на оцінювання швидкодії, але зберігає основні принципи функціонування реальної багатоядерної системи.

У запропонованій структурі кожне обчислювальне ядро виконує окремий фрагмент задачі. Ядро містить набір базових функціональних елементів, серед яких виділено блок обробки інструкцій, арифметико-логічний блок, регістровий файл і локальний буфер або кеш першого рівня. Наявність локальної пам'яті біля кожного ядра дозволяє зменшити кількість звернень до спільної пам'яті та знизити затримки під час виконання повторюваних операцій. Це особливо

важливо для задач, у яких обробляються масиви даних або виконуються однотипні операції над різними частинами вхідної інформації.

Центральним елементом загальної структури є підсистема між'ядерної взаємодії. Вона забезпечує передавання даних між ядрами, доступ до спільної пам'яті та обмін службовою інформацією про стан виконання задач. Для спрощеної моделі доцільно використати внутрішню шину або умовну сіткову структуру, що імітує принцип роботи мережі-на-кристалі. У першому випадку архітектура залишається простішою для моделювання, однак має обмеження за масштабованістю. У другому випадку краще відображається поведінка сучасних багатоядерних систем, де кожне ядро має власний маршрут обміну даними з іншими вузлами. Для цієї кваліфікаційної роботи доцільним є проміжний варіант, у якому взаємодія ядер подана через спільну комунікаційну підсистему з урахуванням затримок передавання даних.

Особливу роль у структурі процесора відіграє підсистема пам'яті. У моделі передбачено наявність локальної кеш-пам'яті біля кожного ядра та спільної пам'яті, доступної для всіх обчислювальних блоків. Така організація дозволяє показати різницю між швидким локальним доступом і повільнішим зверненням до загального ресурсу. За рахунок цього під час моделювання можна оцінити, як частота звернень до спільної пам'яті впливає на загальний час виконання задачі. Якщо ядра надто часто використовують спільні дані, зростає кількість конфліктів доступу, а отже, зменшується очікуваний приріст продуктивності.

Блок керування у запропонованій структурі виконує функцію розподілу задач між ядрами. Він отримує вхідну задачу, поділяє її на окремі фрагменти, передає ці фрагменти обчислювальним ядрам і контролює завершення їх виконання. Після завершення паралельної обробки результати повертаються до спільної пам'яті або об'єднуються в єдиний вихідний результат. Такий підхід дозволяє змоделювати типовий цикл роботи багатоядерного процесора: отримання задачі, розподіл навантаження, паралельне виконання, синхронізація та формування результату.

Загальна структура спеціалізованого багатоядерного процесора сформована з урахуванням того, що основною метою є підвищення продуктивності за рахунок паралельної обробки. Через це в архітектурі важливо забезпечити рівномірне завантаження ядер. Якщо одне ядро отримує значно більший обсяг роботи, ніж інші, загальний час виконання визначається саме найповільнішим фрагментом. Тому в моделі передбачено принцип поділу задачі на приблизно рівні частини. Це дозволяє зменшити простої обчислювальних ядер і підвищити ефективність використання апаратних ресурсів.

Для спеціалізованої архітектури також важливо врахувати тип задач, під які вона орієнтується. У межах цієї кваліфікаційної роботи процесор доцільно розглядати як засіб для виконання обчислень, які добре піддаються розпаралеленню. До таких задач можна віднести обробку масивів даних, виконання однотипних арифметичних операцій, моделювання простих обчислювальних процесів, фільтрацію даних або обробку сигналів. Саме для таких сценаріїв багатоядерна структура має найбільший практичний сенс, оскільки значна частина роботи може виконуватися одночасно кількома ядрами.

У сформованій структурі не передбачено надмірного ускладнення окремого ядра. Замість цього продуктивність підвищується за рахунок збільшення кількості обчислювальних блоків і раціональної організації обміну даними. Такий підхід відповідає логіці спеціалізованих процесорів, у яких важливим є не універсальність, а ефективне виконання конкретного класу задач. Завдяки цьому модель залишається зрозумілою для аналізу, придатною для програмної симуляції та зручною для подальшого порівняння з одноядерною конфігурацією.

Усі ядра мають бути підключені до внутрішньої комунікаційної підсистеми, яка взаємодіє з контролером пам'яті та спільною пам'яттю. Окремо доцільно розмістити блок керування задачами, що відповідає за розподіл навантаження між ядрами.

Рисунок 2.1 – Загальна структура спеціалізованого багатоядерного процесора підвищеної продуктивності

Основним архітектурним рішенням стало використання багатоядерної структури з кількома однотипними обчислювальними ядрами. Такий підхід обрано через його зрозумілість, прогнозованість і зручність для подальшого моделювання. Однотипні ядра мають однакову функціональну будову, однаковий набір базових операцій і працюють за єдиним принципом. Це спрощує розподіл задач між ними, оскільки кожне ядро може виконувати будь-який фрагмент паралельної задачі без потреби в окремій адаптації алгоритму під конкретний тип обчислювального блоку. У спеціалізованому процесорі така симетрична організація дає змогу зосередитися саме на оцінюванні впливу кількості ядер і способу взаємодії між ними на кінцеву продуктивність. Призначення основних блоків спеціалізованого багатоядерного процесора подано в таблиці 2.1.

Таблиця 2.1 – Призначення основних блоків спеціалізованого багатоядерного процесора

Функціональний блок	Призначення
Обчислювальне ядро	Виконує фрагмент паралельної задачі та здійснює базові арифметико-логічні операції
Локальна кеш-пам'ять	Зберігає часто використовувані дані конкретного ядра та зменшує кількість звернень до спільної пам'яті
Внутрішня комунікаційна підсистема	Забезпечує обмін даними між ядрами, кешами та контролером пам'яті
Контролер пам'яті	Організовує доступ ядер до спільної пам'яті та регулює черговість запитів
Спільна пам'ять	Зберігає вхідні дані, проміжні результати та підсумкові результати обчислень
Блок керування задачами	Розподіляє фрагменти задач між ядрами та контролює завершення паралельного виконання

У результаті сформована загальна структура спеціалізованого багатоядерного процесора поєднує кілька обчислювальних ядер, локальні кеші, спільну пам'ять і підсистему керування обміном даними. Така архітектура створює основу для подальшого розгляду архітектурних рішень, структури обчислювального ядра, механізмів між'ядерної взаємодії та моделювання продуктивності. Запропонована модель дозволяє не лише описати будову процесора, а й підготувати базу для експериментального аналізу впливу кількості ядер, затримок пам'яті та способу розподілу задач на загальну швидкодію системи.

2.2 Обґрунтування архітектурних рішень для підвищення продуктивності процесора

Після формування загальної структури спеціалізованого багатоядерного процесора важливим етапом стало обґрунтування архітектурних рішень, які безпосередньо впливають на його продуктивність. У межах кваліфікаційної роботи процесор розглянуто не як універсальний обчислювальний пристрій для будь-яких типів програм, а як спеціалізовану архітектуру, орієнтовану на ефективне виконання задач, що добре піддаються паралельній обробці. Саме така орієнтація дозволяє не ускладнювати окреме ядро надмірною кількістю функціональних блоків, а спрямувати основний архітектурний акцент на кількість ядер, швидкість обміну даними, організацію пам'яті та рівномірний розподіл навантаження.

Вибір однотипних ядер також пов'язаний із потребою забезпечити рівномірне завантаження обчислювальних ресурсів. Якщо процесор складається з ядер різної продуктивності, то розподіл задач стає складнішим, оскільки необхідно враховувати різну швидкодію окремих блоків. Для моделі, яка розробляється у межах цієї кваліфікаційної роботи, доцільнішою є структура, у якій усі ядра мають однакові можливості. Це дозволяє простіше порівнювати час виконання задачі на одному, двох, чотирьох або більшій кількості ядер і робити висновки щодо масштабованості архітектури.

Наступним важливим рішенням стало використання спрощеного обчислювального ядра. У запропонованій архітектурі ядро не перевантажене складними механізмами позачергового виконання, глибоким прогнозуванням переходів або великою кількістю спеціалізованих блоків. Основний приріст продуктивності має забезпечуватися не ускладненням одного ядра, а одночасною роботою декількох ядер. Такий підхід відповідає логіці спеціалізованих багатоядерних систем, де важливо досягти доброго співвідношення між швидкодією, складністю реалізації та можливістю масштабування.

Окрему увагу приділено архітектурі пам'яті. У процесорі передбачено поєднання локальної кеш-пам'яті біля кожного ядра та спільної пам'яті для зберігання вхідних, проміжних і підсумкових даних. Локальна кеш-пам'ять зменшує кількість звернень до спільного ресурсу, оскільки часто використовувані дані зберігаються ближче до обчислювального ядра.

Це знижує затримки під час виконання повторюваних операцій і підвищує ефективність роботи ядра. Водночас спільна пам'ять забезпечує можливість обміну результатами між ядрами та підтримує цілісність загального обчислювального процесу.

Важливим архітектурним рішенням стало введення внутрішньої комунікаційної підсистеми, через яку ядра отримують доступ до пам'яті та обмінюються даними. У межах моделі така підсистема може бути подана як спільна шина з урахуванням затримок або як спрощена мережа-на-кристалі.

Для підвищення продуктивності також обґрунтовано використання блока керування задачами. Його призначення полягає у поділі вхідної задачі на окремі фрагменти та передаванні цих фрагментів обчислювальним ядрам. Без такого блока процесор не зможе ефективно використовувати наявні ядра, оскільки паралельне виконання потребує організованого розподілу роботи.У моделі блок керування виконує роль координатора, який визначає, які частини задачі мають виконуватися одночасно, контролює завершення обчислень і забезпечує об'єднання результатів.

Ще одним важливим рішенням стало врахування накладних витрат на синхронізацію. У реальних багатоядерних системах прискорення не зростає пропорційно кількості ядер, оскільки частина часу витрачається на передавання даних, очікування доступу до пам'яті та узгодження результатів. У запропонованій моделі ці витрати не ігноруються, а враховуються як окремий фактор продуктивності.

На рисунку 2.2 зображено центральний блок «Підвищення продуктивності процесора», до якого ведуть кілька основних напрямів: збільшення кількості

ядер, використання локальної кеш-пам'яті, оптимізація доступу до спільної пам'яті, організація між'ядерної взаємодії, розподіл задач між ядрами та врахування витрат синхронізації.

Рисунок 2.2 – Вплив архітектурних рішень на підвищення продуктивності спеціалізованого багатоядерного процесора

Водночас враховано, що не всі задачі однаково ефективно виконуються на багатоядерній архітектурі. Якщо програма містить значну послідовну частину або часто звертається до спільних даних, приріст продуктивності зменшується. Саме тому в моделі передбачено можливість аналізу різних сценаріїв навантаження. Це дозволяє не лише підтвердити переваги запропонованої структури, а й виявити умови, за яких збільшення кількості ядер не дає очікуваного результату.

Архітектуру спеціалізованого багатоядерного процесора сформовано з урахуванням кількох ключових принципів: простота окремого ядра, паралельність виконання, локалізація часто використовуваних даних, контрольований доступ до спільної пам'яті та централізований розподіл задач. Поєднання цих принципів створює основу для підвищення продуктивності без

надмірного ускладнення моделі. У результаті процесор залишається достатньо простим для опису та моделювання, але водночас відображає основні проблеми, характерні для реальних багатоядерних систем. Порівняння архітектурних рішень для моделі багатоядерного процесора подано в таблиці 2.2.

Таблиця 2.2 – Порівняння архітектурних рішень для моделі багатоядерного процесора

Архітектурне рішення	Перевага для продуктивності	Обмеження
Однотипні обчислювальні ядра	Спрощують розподіл задач і забезпечують рівномірне навантаження	Менша гнучкість порівняно з гетерогенними структурами
Спрощена структура ядра	Зменшує складність моделі та дозволяє зосередитися на паралельному виконанні	Не враховує всі можливості сучасних високопродуктивних ядер
Локальна кеш-пам'ять біля кожного ядра	Зменшує кількість звернень до спільної пам'яті	Потребує врахування узгодженості даних
Спільна пам'ять	Забезпечує простий обмін даними між ядрами	Може стати вузьким місцем при великій кількості звернень
Блок керування задачами	Організовує розподіл навантаження між ядрами	Додає накладні витрати на координацію
Врахування синхронізації	Робить модель продуктивності реалістичнішою	Ускладнює розрахунок загального часу виконання

У підсумку обґрунтовані архітектурні рішення дозволяють сформувати модель процесора, яка поєднує простоту реалізації та можливість аналізу продуктивності. Обрана структура не зводиться лише до збільшення кількості ядер, а враховує пам'ять, обмін даними, керування задачами та синхронізацію. Це створює основу для подальшого розроблення функціональної структури обчислювального ядра та моделювання виконання паралельних задач у наступних підрозділах.

2.3 Розроблення функціональної структури обчислювального ядра

У межах проєктування спеціалізованого багатоядерного процесора окрему увагу приділено функціональній структурі одного обчислювального ядра, оскільки саме ядро є базовим елементом усієї багатоядерної архітектури. Загальна продуктивність процесора залежить не лише від кількості ядер, а й від того, наскільки раціонально організовано внутрішню будову кожного з них. Якщо ядро має надмірно складну структуру, модель стає важчою для аналізу, а оцінювання продуктивності ускладнюється великою кількістю другорядних факторів. Якщо ж ядро надто спрощене, воно не відображає реальних принципів роботи процесорної архітектури. Через це у кваліфікаційній роботі сформовано проміжний варіант структури, який зберігає основні функціональні блоки процесорного ядра та водночас залишається зручним для подальшого моделювання.

Обчислювальне ядро розглянуто як відносно автономний блок, здатний отримувати фрагмент задачі, виконувати послідовність інструкцій над локальними або спільними даними та передавати результат до загальної підсистеми пам'яті. У складі ядра передбачено блок вибірки інструкцій, блок декодування, пристрій керування, регістровий файл, арифметико-логічний пристрій, локальну кеш-пам'ять і інтерфейс взаємодії з внутрішньою комунікаційною підсистемою. Такий набір компонентів дозволяє описати

повний цикл роботи ядра: отримання інструкції, її розпізнавання, підготовку операндів, виконання операції, запис результату та обмін даними з іншими блоками процесора.

Арифметико-логічний пристрій є основним виконавчим блоком ядра. Він реалізує базові операції додавання, віднімання, порівняння, логічного множення, логічного додавання, зсувів та інших простих операцій, необхідних для обробки даних. У межах кваліфікаційної роботи АЛП не ускладнюється підтримкою великої кількості спеціалізованих команд, оскільки головним завданням є не створення повної промислової мікроархітектури, а побудова моделі, придатної для аналізу впливу багатоядерності на продуктивність. Разом із тим саме АЛП визначає здатність ядра виконувати обчислювальні фрагменти задачі, тому його наявність є обов'язковою частиною функціональної структури.

Локальна кеш-пам'ять у складі ядра використовується для тимчасового зберігання даних, які найчастіше застосовуються під час виконання фрагмента задачі. Її введення до структури ядра є важливим, оскільки без локального кешу кожне ядро змушене часто звертатися до спільної пам'яті, що збільшує затримки й створює конфлікти доступу. У запропонованій архітектурі локальний кеш працює як проміжний рівень між регістровим файлом і загальною підсистемою пам'яті. Він дозволяє пришвидшити повторне використання даних і зменшити навантаження на комунікаційну підсистему.

Інтерфейс взаємодії з внутрішньою комунікаційною підсистемою забезпечує зв'язок ядра з іншими компонентами багатоядерного процесора. Через цей інтерфейс ядро отримує вхідні дані, надсилає результати, звертається до спільної пам'яті та бере участь у процесі синхронізації. У моделі цей блок має особливе значення, оскільки саме через нього проявляється вплив між'ядерного обміну на продуктивність. Якщо інтерфейс працює з великими затримками або часто очікує доступу до спільного ресурсу, ефективність навіть швидкого обчислювального ядра знижується. Через це у подальшому моделюванні

доцільно враховувати час звернення ядра до комунікаційної підсистеми як окремий параметр.

Функціональна структура ядра побудована так, щоб забезпечити зрозумілий цикл виконання інструкції. Спочатку інструкція вибирається з відповідного джерела, після чого декодується та перетворюється у керувальні сигнали. Далі з регістрового файла або кеш-пам'яті отримуються необхідні операнди. Після цього арифметико-логічний пристрій виконує потрібну операцію, а результат записується назад до регістра, локального кешу або передається до спільної пам'яті. Такий цикл є базовим для аналізу часу виконання, оскільки дозволяє приблизно оцінити, які етапи створюють найбільші затримки.

Особливістю запропонованої структури є те, що ядро розглянуто як елемент, оптимізований для повторюваних обчислювальних операцій. Це відповідає загальній ідеї спеціалізованого багатоядерного процесора, у якому продуктивність досягається не за рахунок максимально складного універсального ядра, а за рахунок одночасної роботи кількох простіших виконавчих блоків. Такий підхід добре підходить для задач обробки масивів даних, простого числового моделювання, фільтрації, обчислення проміжних значень і виконання однотипних операцій над незалежними фрагментами інформації.

Важливою перевагою такої функціональної структури є її масштабованість. Оскільки кожне ядро має однакову будову, архітектура процесора може розширюватися шляхом додавання нових ядер без повної перебудови логіки одного обчислювального блоку. При цьому зростання кількості ядер потребує лише коригування комунікаційної підсистеми, механізмів доступу до пам'яті та логіки розподілу задач. Це робить запропоновану структуру зручною для моделювання конфігурацій із різною кількістю ядер.

На рисунку 2.3 показано послідовний зв'язок між блоком вибірки інструкцій, блоком декодування, пристроєм керування, регістровим файлом, арифметико-логічним пристроєм, локальною кеш-пам'яттю та інтерфейсом доступу до внутрішньої комунікаційної підсистеми.

Рисунок 2.3 – Функціональна структура обчислювального ядра спеціалізованого багатоядерного процесора

У процесі розроблення функціональної структури ядра також враховано обмеження, пов'язані з надмірною деталізацією. Для кваліфікаційної роботи недоцільно моделювати всі внутрішні особливості сучасних процесорів, зокрема складні механізми суперскалярного виконання, багаторівневе прогнозування переходів або глибоку оптимізацію конвеєра.

Така схема дозволить візуально пояснити, як усередині ядра проходить інструкція та яким чином ядро взаємодіє з іншими компонентами процесора. Для деталізації призначення основних блоків ядра доцільно подати таблицю 2.3.

Таблиця 2.3 – Призначення функціональних блоків обчислювального ядра

Функціональний блок	Призначення
Блок вибірки інструкцій	Отримує чергову інструкцію для подальшого виконання
Блок декодування	Визначає тип інструкції, джерела операндів і необхідні керувальні дії
Пристрій керування	Координує роботу внутрішніх блоків ядра та формує керувальні сигнали
Регістровий файл	Зберігає операнди, проміжні значення та результати операцій
Арифметико-логічний пристрій	Виконує основні арифметичні й логічні операції над даними
Локальна кеш-пам'ять	Забезпечує швидкий доступ до часто використовуваних даних
Інтерфейс комунікаційної підсистеми	Забезпечує обмін даними між ядром, спільною пам'яттю та іншими блоками процесора

У результаті розроблена функціональна структура обчислювального ядра поєднує необхідні блоки для виконання інструкцій, обробки даних і взаємодії з пам'яттю. Запропоноване ядро не перевантажене надмірною мікроархітектурною складністю, але містить усі елементи, потрібні для моделювання роботи спеціалізованого багатоядерного процесора. Це створює основу для подальшого розгляду організації обміну даними між ядрами та підсистемою пам'яті, оскільки саме взаємодія окремих ядер у межах єдиної архітектури визначає реальну ефективність багатоядерної системи.

2.4 Організація обміну даними між ядрами та підсистемою пам'яті

Організація обміну даними між обчислювальними ядрами та підсистемою пам'яті є одним із ключових елементів проєктування спеціалізованого багатоядерного процесора. Навіть за наявності кількох продуктивних ядер загальна швидкодія системи може залишатися низькою, якщо між ядрами, кеш-пам'яттю та спільною пам'яттю виникають значні затримки. Через це у межах кваліфікаційної роботи обмін даними розглянуто не як допоміжний процес, а як окрему архітектурну складову, що безпосередньо впливає на ефективність паралельного виконання задач.

У запропонованій моделі спеціалізованого багатоядерного процесора кожне ядро має доступ до локальної кеш-пам'яті, внутрішньої комунікаційної підсистеми та спільної пам'яті. Така структура дозволяє поєднати переваги локального зберігання даних і централізованого доступу до загального адресного простору. Локальна кеш-пам'ять використовується для швидкого доступу до тих даних, які найчастіше застосовуються конкретним ядром під час виконання виділеного фрагмента задачі. Спільна пам'ять призначена для зберігання вхідних масивів, проміжних результатів, службових даних і підсумкових значень, які формуються після завершення паралельної обробки.

Основна логіка обміну даними полягає в тому, що перед початком виконання задачі блок керування розподіляє вхідні дані між ядрами. Кожне ядро отримує власний фрагмент даних, завантажує його до локального кешу або безпосередньо до регістрів і виконує обчислення незалежно від інших ядер. Такий підхід дозволяє зменшити кількість звернень до спільної пам'яті під час активної фази обробки. Якщо ядро працює переважно з локальним фрагментом інформації, воно менше навантажує загальну комунікаційну підсистему, а отже, зменшується ймовірність конфліктів доступу.

У структурі процесора передбачено, що обмін між ядрами не відбувається постійно на рівні кожної операції. Це важливо для збереження продуктивності,

оскільки часті між'ядерні звернення призводять до появи додаткових затримок. Обмін даними доцільно виконувати на визначених етапах: під час отримання фрагментів задачі, у разі потреби синхронізації проміжних результатів і після завершення виконання обчислень. Така організація дозволяє зробити паралельне виконання більш передбачуваним і зменшити накладні витрати на комунікацію.

Окремим питанням є узгодженість даних між локальними кешами різних ядер. Якщо кілька ядер працюють із незалежними фрагментами вхідного масиву, проблема узгодженості майже не проявляється. Однак у випадку роботи зі спільними змінними або проміжними результатами виникає потреба контролювати актуальність даних. У запропонованій моделі доцільно використати спрощений механізм узгодження, за якого після запису результату до спільної пам'яті інші ядра отримують доступ уже до оновленого значення. Для повноцінної апаратної реалізації можна було б застосувати складніші протоколи кеш-когерентності, але для моделювання продуктивності достатньо врахувати сам факт появи додаткової затримки під час синхронізації спільних даних.

Для підвищення ефективності обміну даними важливо мінімізувати кількість ситуацій, коли ядра одночасно очікують доступу до спільної пам'яті. У моделі це досягається за рахунок поділу задачі на незалежні частини. Кожне ядро отримує власний блок даних і виконує над ним операції автономно. Лише після завершення обчислень результати передаються до спільної пам'яті для подальшого об'єднання. Такий підхід зменшує навантаження на комунікаційну підсистему та дозволяє краще використати переваги паралельної архітектури.

Важливою складовою є синхронізація результатів. Після завершення роботи окремих ядер система повинна переконатися, що всі фрагменти задачі опрацьовано. Лише після цього виконується об'єднання результатів або перехід до наступного етапу обчислень. У моделі така синхронізація може бути подана як бар'єрна точка, у якій швидші ядра очікують завершення повільніших. Це дозволяє врахувати одну з типових проблем паралельних обчислень: загальний

час виконання визначається не середнім часом роботи ядер, а часом завершення найповільнішого фрагмента задачі.

Підсистема пам'яті у спеціалізованому багатоядерному процесорі має підтримувати не лише зберігання даних, а й раціональний режим доступу до них. Якщо всі ядра постійно звертаються до одного й того самого сегмента пам'яті, виникає перевантаження контролера та збільшується час очікування. Тому під час проєктування моделі важливо врахувати просторову локальність даних, тобто розміщення пов'язаних елементів поруч, і часову локальність, тобто повторне використання вже завантажених значень. Саме ці принципи пояснюють доцільність використання кеш-пам'яті у кожному ядрі.

У запропонованій архітектурі можна виділити три основні режими обміну даними. Перший режим пов'язаний із початковим завантаженням даних, коли блок керування або контролер пам'яті передає кожному ядру відповідний фрагмент задачі. Другий режим відповідає локальній обробці, під час якої ядро переважно працює з власним кешем і регістрами. Третій режим стосується збереження результатів, коли оброблені фрагменти передаються назад до спільної пам'яті. Така послідовність є зручною для моделювання, оскільки дозволяє окремо оцінити витрати на завантаження, виконання та запис результатів.

З практичної точки зору організація обміну даними має бути узгоджена з моделлю виконання паралельних задач. Якщо задача ділиться на незалежні блоки, ефективність процесора буде високою, оскільки ядра майже не заважають одне одному. Якщо ж задача потребує частого обміну проміжними результатами, перевага багатоядерності зменшується через накладні витрати. Саме тому в подальшому аналізі продуктивності доцільно порівнювати різні сценарії: задачі з низькою, середньою та високою інтенсивністю обміну даними. Це дозволить показати, за яких умов запропонована архітектура працює найефективніше.

У результаті організація обміну даними між ядрами та підсистемою пам'яті сформована як поєднання локального зберігання, спільного доступу та

контрольованої синхронізації. Такий підхід дозволяє зменшити кількість звернень до загальної пам'яті, знизити затримки під час виконання паралельних фрагментів задачі та забезпечити коректне об'єднання результатів. Запропонована схема є достатньо простою для програмного моделювання, але водночас відображає основні обмеження, характерні для реальних багатоядерних процесорів. Це створює основу для подальшого розгляду моделі розподілу задач між обчислювальними ядрами та оцінювання впливу комунікаційних витрат на загальну продуктивність процесора.

2.5 Побудова моделі паралельного виконання обчислювальних задач

Після визначення загальної структури спеціалізованого багатоядерного процесора, функціональної будови обчислювального ядра та організації обміну даними між ядрами і підсистемою пам'яті наступним етапом стало формування моделі паралельного виконання обчислювальних задач. Така модель потрібна для того, щоб показати не лише апаратну будову процесора, а й сам принцип використання кількох ядер під час виконання однієї задачі. У багатоядерній архітектурі продуктивність залежить не тільки від кількості обчислювальних блоків, а й від того, наскільки правильно задача поділяється на окремі частини, як ці частини передаються ядрам, як синхронізуються проміжні результати та яким чином формується підсумковий результат.

У межах кваліфікаційної роботи модель паралельного виконання побудовано для задач, які можуть бути поділені на незалежні або частково незалежні фрагменти. Такий підхід відповідає логіці спеціалізованого багатоядерного процесора, оскільки саме задачі з високим рівнем паралелізму найкраще демонструють переваги багатоядерної обробки. До таких задач можна віднести обробку масивів числових даних, фільтрацію сигналів, виконання однотипних арифметичних операцій над різними наборами вхідних значень,

прості елементи моделювання та обчислення проміжних показників у виробничих або технічних системах.

Основою моделі є поділ вхідної задачі на фрагменти однакового або близького за обсягом розміру. Такий принцип дозволяє рівномірно завантажити обчислювальні ядра та зменшити час простою окремих блоків. Якщо одне ядро отримує значно більший фрагмент роботи, ніж інші, загальний час виконання визначається саме цим ядром, а решта ядер після завершення своїх операцій очікують синхронізації. Через це в моделі передбачено попередній етап розбиття задачі, під час якого блок керування оцінює обсяг вхідних даних і розподіляє їх між ядрами відповідно до кількості доступних обчислювальних блоків.

У моделі передбачено, що кожне ядро працює з власним фрагментом даних незалежно від інших ядер протягом основної фази обчислень. Це дозволяє зменшити кількість звернень до спільної пам'яті та мінімізувати конфлікти доступу. Такий підхід є найбільш ефективним для задач, у яких результат одного фрагмента не залежить від проміжних результатів іншого фрагмента. Наприклад, якщо потрібно виконати однакову операцію над елементами великого масиву, кожне ядро може обробляти власну частину масиву без постійного обміну з іншими ядрами.

Разом із тим у моделі враховано ситуації, коли після завершення локальної обробки необхідне об'єднання результатів. Це може бути підсумовування часткових значень, формування спільного вихідного масиву, порівняння результатів або передавання даних на наступний етап обчислень. У таких випадках виникає потреба в синхронізації. Синхронізація в моделі подана як бар'єрна точка, у якій усі ядра мають завершити свою частину роботи перед переходом до наступного етапу. Якщо одне з ядер працює довше, інші ядра очікують. Це дозволяє більш реалістично оцінити ефективність паралельного виконання, оскільки враховується не тільки швидкість окремих обчислень, а й узгодженість роботи всієї системи.

Для оцінювання ефективності моделі важливим є показник прискорення. Він показує, у скільки разів багатоядерне виконання є швидшим за одноядерне. У найпростішому випадку очікується, що збільшення кількості ядер зменшує час виконання задачі. Проте фактичне прискорення зазвичай є меншим за кількість ядер, оскільки присутні послідовні етапи та накладні витрати. Через це в моделі доцільно порівнювати не лише абсолютний час виконання, а й ефективність використання ядер. Якщо чотири ядра забезпечують прискорення лише у два рази, це означає, що архітектура або сама задача мають обмеження, які не дозволяють повністю використати паралельний потенціал.

Важливою частиною моделі є планування роботи ядер. У спрощеному варіанті використовується статичний розподіл, коли блок керування наперед ділить задачу на фрагменти й передає їх ядрам. Такий підхід добре підходить для задач із рівномірним обсягом обчислень. Якщо ж фрагменти мають різну складність, доцільним є динамічний розподіл, за якого ядра після завершення одного фрагмента можуть отримати наступний. У межах цієї кваліфікаційної роботи основною є статична модель, оскільки вона простіша для аналізу та дозволяє чітко простежити вплив кількості ядер на час виконання. Проте сама структура моделі залишає можливість подальшого розширення до динамічного планування.

Побудована модель паралельного виконання дозволяє відобразити повний цикл роботи спеціалізованого багатоядерного процесора під час обробки задачі. Спочатку задача надходить до системи, після чого розподіляється між ядрами. Потім ядра виконують обчислення паралельно, використовуючи локальні кеші й звертаючись до спільної пам'яті лише за потреби. Після завершення локальних операцій відбувається синхронізація, об'єднання результатів і формування вихідного значення.

Для наочного подання моделі паралельного виконання у підрозділі доцільно використати блок-схему.

На рисунку 2.4 варто показано послідовність етапів: надходження задачі, розбиття на фрагменти, передавання фрагментів ядрам, паралельне виконання, локальне збереження проміжних результатів, синхронізація, об'єднання результатів і формування підсумкового результату.

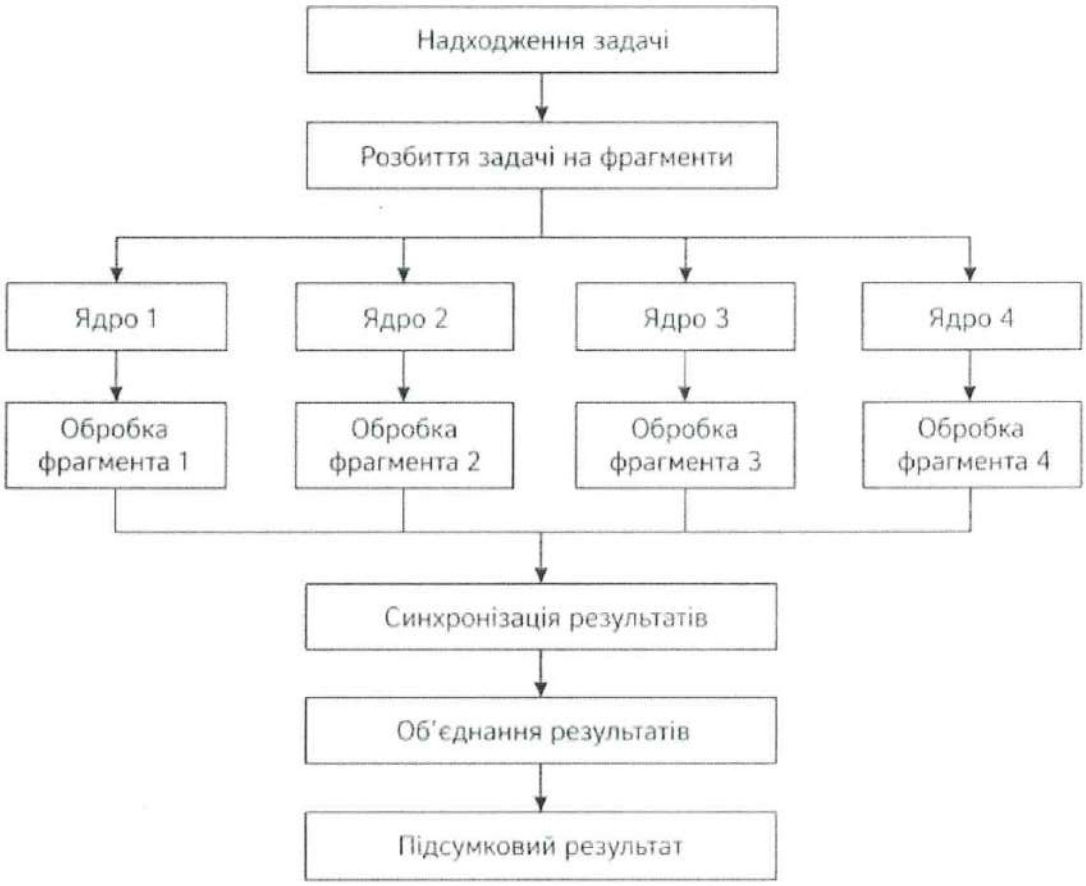


Рисунок 2.4 – Модель паралельного виконання обчислювальної задачі у спеціалізованому багатоядерному процесорі

У результаті побудована модель паралельного виконання обчислювальних задач дозволяє описати не лише сам факт одночасної роботи кількох ядер, а й повний порядок організації такої роботи. Вона враховує розподіл даних, локальне виконання, доступ до пам'яті, синхронізацію та об'єднання результатів. Це створює основу для подальшого вибору програмних засобів моделювання та оцінювання продуктивності, оскільки саме ця модель визначає,

які параметри необхідно вимірювати: час виконання, прискорення, ефективність використання ядер, накладні витрати та вплив обміну даними на загальну швидкодію процесора.

2.6 Вибір програмних засобів для моделювання та оцінювання продуктивності

Для моделювання спеціалізованого багатоядерного процесора підвищеної продуктивності важливо обрати такі програмні засоби, які дозволяють не лише описати загальну архітектуру системи, а й перевірити її поведінку під час виконання паралельних задач. Оскільки у межах кваліфікаційної роботи головну увагу зосереджено на моделюванні роботи багатоядерної структури та аналізі продуктивності, програмне середовище має забезпечувати можливість зміни кількості ядер, задання параметрів пам'яті, врахування затримок доступу до даних, розрахунку часу виконання та порівняння результатів для різних конфігурацій.

Під час вибору засобів моделювання враховано, що повна апаратна реалізація багатоядерного процесора на рівні логічних елементів є складною для бакалаврської кваліфікаційної роботи. Це дозволило б наблизити модель до апаратного рівня, однак суттєво ускладнило б основну задачу, яка полягає саме в аналізі продуктивності багатоядерної архітектури. Через це доцільним є використання програмного моделювання, яке дає можливість зосередитися на логіці роботи системи, розподілі задач між ядрами та оцінюванні впливу архітектурних параметрів на швидкодію.

Основним програмним засобом для побудови моделі доцільно обрати мову Python. Її використання обґрунтовано тим, що вона має простий синтаксис, підтримує роботу зі структурами даних, дозволяє швидко створювати імітаційні моделі та зручно виконувати обчислювальні експерименти. У межах цієї кваліфікаційної роботи Python може бути використано для опису логіки роботи

ядер, моделювання черги задач, розподілу фрагментів даних між обчислювальними блоками, врахування часу доступу до пам'яті та побудови підсумкових таблиць із результатами продуктивності.

Додатково для окремих розрахунків може бути використана бібліотека NumPy. Вона зручна для роботи з масивами даних і виконання математичних операцій. У межах моделі NumPy може застосовуватися для формування тестових масивів, поділу даних на фрагменти, виконання однотипних обчислень і перевірки правильності результатів. Це відповідає тематиці спеціалізованого багатоядерного процесора, оскільки одним із типових сценаріїв його використання є паралельна обробка масивів числової інформації.

Для написання та запуску програмної моделі можна використати середовище Visual Studio Code або Jupyter Notebook. Visual Studio Code є зручним варіантом для створення повноцінного програмного файлу, у якому структура моделі подається у вигляді класів, функцій і окремих модулів. Jupyter Notebook доцільний для поетапного виконання моделі, виведення проміжних результатів, побудови графіків і пояснення логіки експериментів.

Окремо розглянуто можливість використання спеціалізованих симуляторів комп'ютерних архітектур, таких як gem5. Такі інструменти дають змогу моделювати поведінку процесорів на більш глибокому рівні, враховувати кеш-ієрархію, пам'ять, системну шину та виконання інструкцій. Проте для цієї кваліфікаційної роботи використання такого інструменту може бути надмірним, оскільки його налаштування потребує більше часу, а сама модель стає складнішою для пояснення. У роботі, де головним завданням є побудова зрозумілої моделі багатоядерного процесора та оцінювання продуктивності, доцільніше застосувати власну програмну імітаційну модель на Python.

Для оцінювання продуктивності в моделі передбачено використання кількох основних показників. Першим показником є час виконання задачі, який дозволяє порівняти одноядерний і багатоядерний режими. Другим показником є коефіцієнт прискорення, що показує, у скільки разів виконання на кількох ядрах

є швидшим порівняно з одним ядром. Третім показником є ефективність використання ядер, яка показує, наскільки повно використовується паралельний потенціал процесора. Також доцільно враховувати накладні витрати на синхронізацію, кількість звернень до спільної пам'яті та затримки, пов'язані з обміном даними.

У програмній моделі можна передбачити кілька сценаріїв виконання. Перший сценарій відповідає ідеальній паралельній задачі, де кожне ядро працює з незалежним фрагментом даних і майже не звертається до спільних ресурсів. Другий сценарій передбачає середній рівень обміну даними, коли ядра періодично синхронізують проміжні результати. Третій сценарій відображає ситуацію з високим навантаженням на пам'ять, коли значна частина часу витрачається на очікування доступу до спільних даних.

Для наочності у підрозділі подано схему програмного середовища моделювання. Такий рисунок показує взаємозв'язок між вхідними параметрами, програмною моделлю процесора, модулем виконання паралельних задач, модулем розрахунку метрик і блоком візуалізації результатів.

На рисунку 2.5 показано послідовність роботи програмної моделі: вхідні параметри моделювання надходять до модуля опису архітектури процесора, далі передаються до модуля імітації паралельного виконання, після чого результати опрацьовуються модулем розрахунку показників продуктивності та виводяться у вигляді таблиць і графіків.

У результаті для моделювання спеціалізованого багатоядерного процесора обрано програмний підхід на основі Python і допоміжних бібліотек для математичної обробки, табличного подання даних і побудови графіків. Такий набір засобів дозволяє реалізувати гнучку модель, у якій можна змінювати кількість ядер, параметри пам'яті, розмір задачі та накладні витрати на обмін даними.

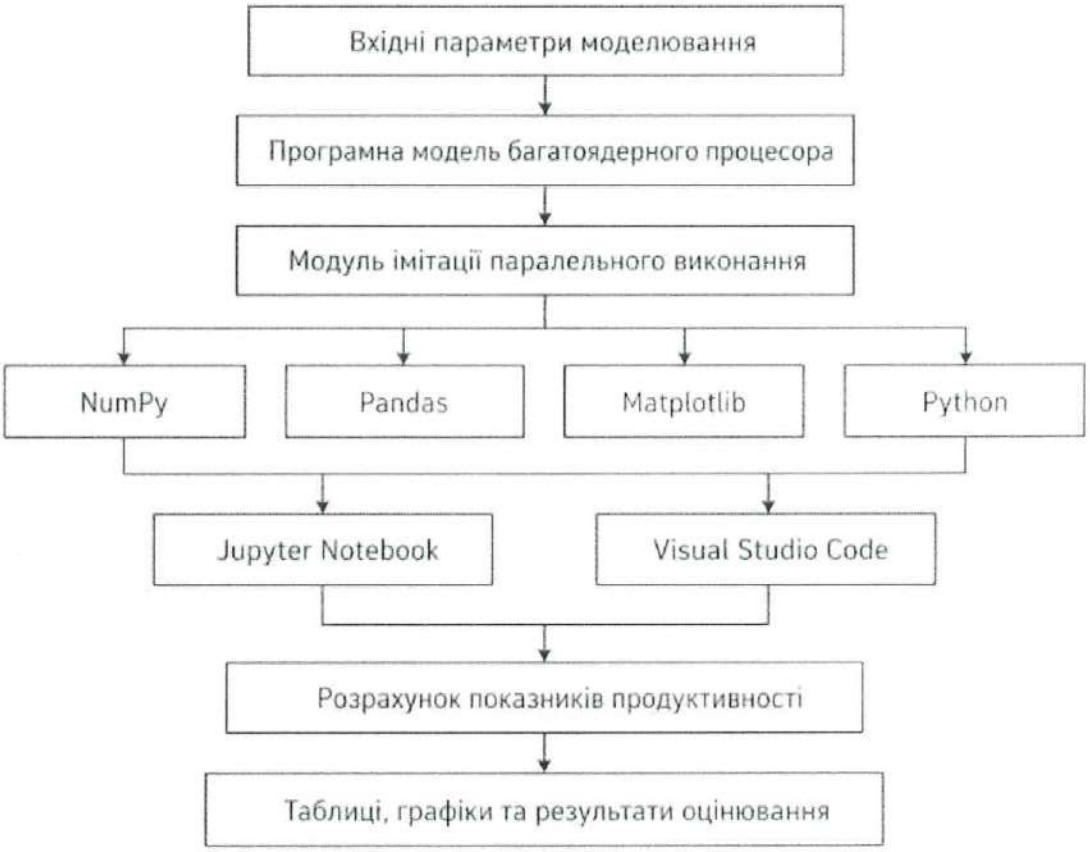


Рисунок 2.5 – Структура програмних засобів моделювання та оцінювання продуктивності багатоядерного процесора

Обрані програмні засоби є достатніми для оцінювання продуктивності, порівняння різних конфігурацій і підготовки експериментальної частини кваліфікаційної роботи. Це створює основу для подальшого розділу, у якому буде виконано програмну реалізацію моделі, проведено тестові запуски та проаналізовано отримані результати.

2.7 Висновок до розділу 2

У другому розділі кваліфікаційної роботи сформовано проєктну основу спеціалізованого багатоядерного процесора підвищеної продуктивності. Розглянуто загальну структуру процесора, до складу якої включено блок керування задачами, обчислювальні ядра, локальну кеш-пам'ять, спільну

пам'ять, контролер пам'яті та внутрішню комунікаційну підсистему. Така побудова дозволила подати процесор як цілісну обчислювальну систему, у якій підвищення продуктивності досягається не лише за рахунок збільшення кількості ядер, а й завдяки узгодженій роботі всіх функціональних блоків.

У межах розділу обґрунтовано архітектурні рішення, спрямовані на підвищення швидкодії процесора. Основну увагу приділено використанню однотипних обчислювальних ядер, локалізації часто використовуваних даних у кеш-пам'яті, організації доступу до спільної пам'яті та зменшенню затримок під час між'ядерної взаємодії. Такий підхід дав змогу сформувати модель, яка залишається достатньо простою для програмного моделювання, але водночас відображає основні особливості реальних багатоядерних архітектур.

Розроблено функціональну структуру обчислювального ядра, у якій передбачено блок вибірки інструкцій, блок декодування, пристрій керування, регістровий файл, арифметико-логічний пристрій, локальну кеш-пам'ять та інтерфейс комунікаційної підсистеми. Така структура дозволяє відтворити основний цикл роботи ядра: отримання інструкції, її обробку, виконання операції, запис результату та взаємодію з іншими компонентами процесора.

Також описано організацію обміну даними між ядрами та підсистемою пам'яті. У моделі враховано, що ефективність багатоядерного процесора значною мірою залежить від кількості звернень до спільної пам'яті, пропускної здатності комунікаційної підсистеми та витрат на синхронізацію результатів. Через це запропонована структура поєднує локальне зберігання даних у кеші кожного ядра та контрольований доступ до спільної пам'яті.

Для практичної реалізації моделі обрано програмний підхід на основі Python, що дає змогу змінювати параметри архітектури, виконувати імітаційне моделювання та формувати результати у вигляді таблиць і графіків. У підсумку другий розділ сформував повну проєктну основу для переходу до практичної частини роботи, у якій реалізовано програмну модель спеціалізованого багатоядерного процесора та виконано аналіз його продуктивності.

3 РЕАЛІЗАЦІЯ МОДЕЛІ СПЕЦІАЛІЗОВАНОГО БАГАТОЯДЕРНОГО ПРОЦЕСОРА ТА АНАЛІЗ ЙОГО ПРОДУКТИВНОСТІ

3.1 Формування програмної структури моделі багатоядерного процесора

Програмну модель побудовано за модульним принципом. Такий підхід дає змогу подати багатоядерний процесор не як один суцільний програмний блок, а як набір взаємопов'язаних частин, кожна з яких виконує окрему функцію. У структурі моделі виділено модуль вхідних параметрів, блок керування задачами, модуль обчислювальних ядер, модуль локального кешу, модуль спільної пам'яті, модуль збирання результатів та модуль оцінювання продуктивності. Завдяки такому поділу модель залишається зрозумілою, зручною для розширення та придатною для перевірки різних конфігурацій багатоядерного процесора.

Основою програмної структури є модуль обчислювального ядра. Кожне ядро подано як окремий логічний елемент, який отримує фрагмент задачі, виконує його обробку та повертає результат до загальної системи.Для кожного ядра задаються номер ядра, обсяг отриманого фрагмента, умовний час виконання операцій, доступ до локального кешу та можливість звернення до спільної пам'яті. Така організація відповідає симетричній багатоядерній архітектурі, у якій усі ядра мають однакову функціональну роль і можуть виконувати однотипні обчислювальні операції.

У програмній моделі передбачено, що ядро не працює з усім обсягом вхідних даних. Спочатку задача надходить до блока керування, де визначається кількість доступних ядер і виконується поділ вхідного набору даних на окремі фрагменти. Після цього кожен фрагмент передається відповідному ядру.

Такий спосіб організації відтворює основний принцип паралельного виконання: загальна задача розкладається на менші частини, які обробляються одночасно кількома обчислювальними блоками.

Блок керування задачами у програмній структурі виконує роль координатора. Він приймає вхідні параметри моделювання, визначає режим роботи процесора, розподіляє дані між ядрами та контролює завершення обчислень. У межах цієї моделі використано рівномірний розподіл фрагментів, за якого кожне ядро отримує приблизно однакову частину задачі. Це дозволяє зменшити простої обчислювальних блоків і забезпечити коректніше порівняння результатів для конфігурацій з різною кількістю ядер.

Загальну програмну структуру моделі спеціалізованого багатоядерного процесора подано на рисунку 3.1.

На ньому показано, що вхідні параметри моделювання передаються до блока керування задачами. Далі задача поділяється на фрагменти, які надходять до окремих обчислювальних ядер. Кожне ядро взаємодіє з локальним кешем і за потреби звертається до спільної пам'яті через модуль пам'яті. Після завершення обчислень результати передаються до модуля збирання результатів, а потім опрацьовуються модулем оцінювання продуктивності.

Програмну структуру сформовано так, щоб параметри моделювання можна змінювати без повного переписування коду. Кількість ядер може змінюватися від одного до кількох, обсяг задачі може збільшуватися для перевірки масштабованості, а затримки доступу до пам'яті та витрати на синхронізацію можуть задаватися окремими параметрами. Завдяки цьому модель придатна для проведення кількох серій експериментів і порівняння різних умов роботи спеціалізованого багатоядерного процесора.

Модуль пам'яті в програмній моделі відповідає за імітацію доступу до локального кешу та спільної пам'яті. У реальному багатоядерному процесорі різні рівні пам'яті мають різну швидкість доступу, тому в моделі враховано умовну різницю між швидким зверненням до локального кешу та повільнішим доступом до спільної пам'яті.

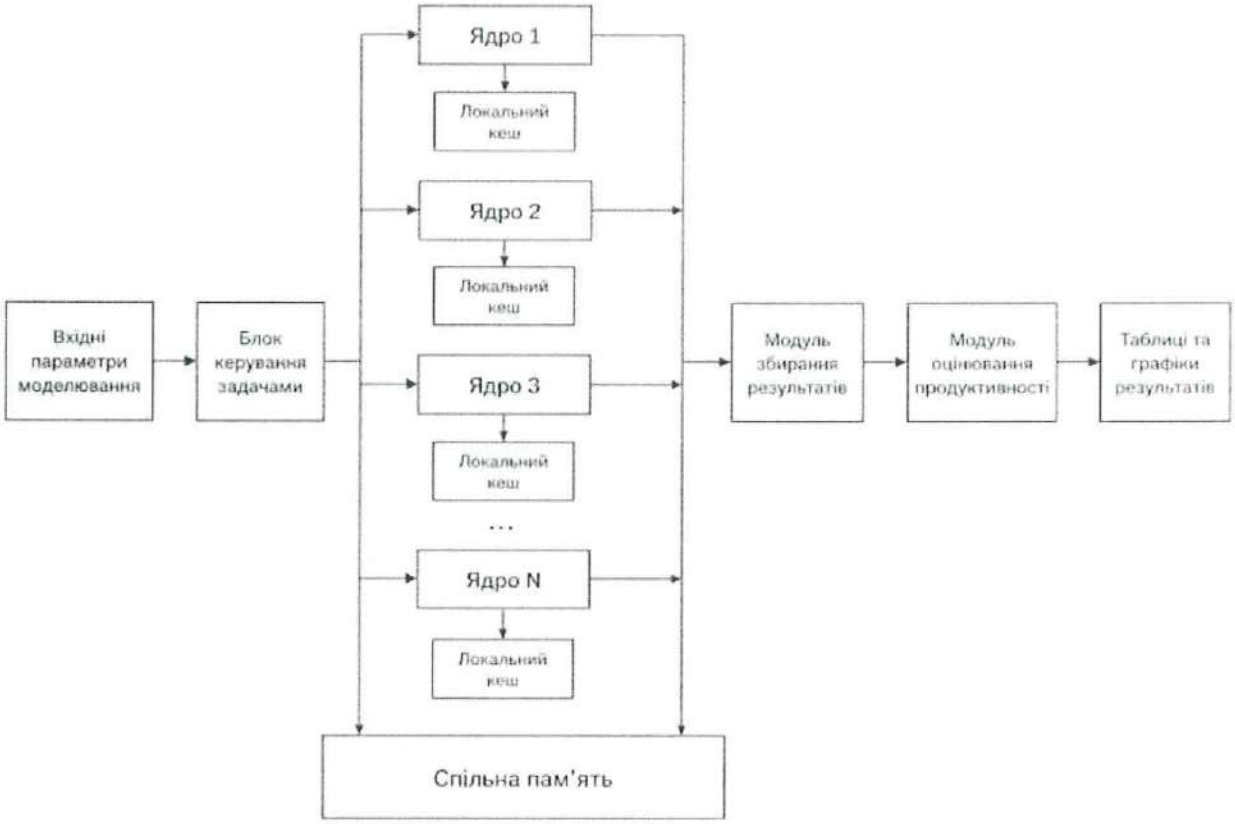


Рисунок 3.1 – Структура програмної моделі спеціалізованого багатоядерного процесора

Основні модулі програмної моделі та їх призначення наведено в таблиці 3.1. У таблиці систематизовано склад програмної структури, що спрощує розуміння ролі кожного елемента у процесі моделювання.

У структурі моделі відокремлено обчислювальну частину від частини оцінювання. Обчислювальні модулі імітують роботу процесора, а окремий модуль продуктивності аналізує отримані значення. Це робить програму більш зрозумілою та зручною для перевірки.

Крім того, така організація дозволяє надалі розширювати модель, наприклад додавати інші алгоритми розподілу задач, змінювати спосіб урахування кеш-пам'яті або вводити додаткові сценарії навантаження.

Таблиця 3.1 – Основні модулі програмної моделі багатоядерного процесора

№	Модуль програмної моделі	Призначення
1.	Модуль вхідних параметрів	Задає кількість ядер, обсяг задачі, затримки пам'яті та параметри синхронізації
2.	Блок керування задачами	Поділяє задачу на фрагменти та розподіляє їх між обчислювальними ядрами
3.	Модуль паралельного виконання	Визначає порядок одночасної роботи ядер і час завершення паралельної фази
4.	Модуль збирання результатів	Об'єднує результати, отримані від окремих обчислювальних ядер
5.	Модуль оцінювання продуктивності	Розраховує час виконання, прискорення та ефективність використання ядер
6.	Модуль виведення результатів	Формує таблиці та графіки для подальшого аналізу

Сформована програмна структура стала основою для подальшої реалізації логіки розподілу задач між обчислювальними ядрами. Вона забезпечує зрозумілий поділ програми на функціональні модулі, підтримує зміну параметрів моделювання та дозволяє оцінювати продуктивність процесора для різних конфігурацій. Це створює технічну базу для наступного етапу, де детальніше розглядається механізм передавання фрагментів задачі окремим ядрам і забезпечення рівномірного навантаження.

3.2 Реалізація логіки розподілу задач між обчислювальними ядрами

Після формування програмної структури моделі багатоядерного процесора реалізовано логіку розподілу обчислювальної задачі між окремими ядрами. Саме

цей етап визначає, наскільки ефективно використовується багатоядерна архітектура, оскільки збільшення кількості ядер саме по собі ще не гарантує пропорційного зменшення часу виконання. Якщо задача поділяється нерівномірно, одні ядра завершують роботу швидше, а інші продовжують обробку своїх фрагментів. У такій ситуації частина обчислювальних ресурсів просто очікує завершення найповільнішого ядра, через що реальна продуктивність знижується.

Для оцінювання роботи моделі прийнято, що обробка одного елемента потребує 1 умовного такту. За такої умови одноядерна конфігурація виконує 100000 умовних операцій. Двоядерна конфігурація зменшує обсяг обчислень для одного ядра до 50000 умовних операцій, чотириядерна - до 25000, а восьмиядерна - до 12500.

Логіку розподілу обчислювальної задачі між ядрами процесора подано на рисунку 3.2. На ньому відображено послідовність дій: надходження вхідної задачі, визначення кількості активних ядер, поділ задачі на фрагменти, передавання фрагментів окремим ядрам, виконання обробки та повернення часткових результатів до загальної системи. Така схема показує, що розподіл задачі є не окремою допоміжною дією, а початковим етапом паралельного виконання, від якого залежить подальша ефективність роботи всієї моделі.

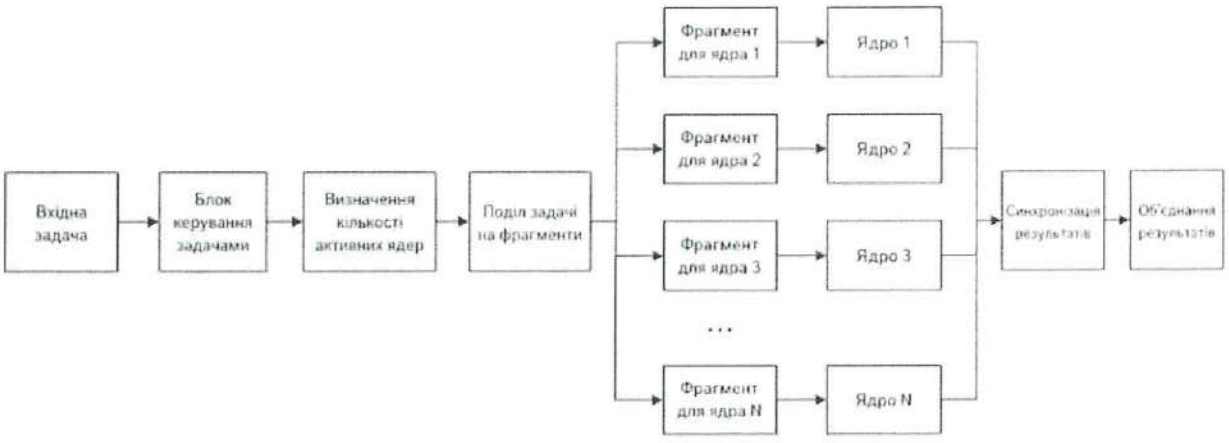


Рисунок 3.2 – Послідовність розподілу обчислювальної задачі між ядрами процесора

У межах реалізованої логіки кожне ядро працює лише зі своїм фрагментом даних. Це зменшує потребу в постійному обміні між ядрами під час основної фази обчислення. Наприклад, якщо задача полягає в обробці числового масиву, перше ядро може працювати з елементами від 1 до 25000, друге - з елементами від 25001 до 50000, третє - з елементами від 50001 до 75000, а четверте - з елементами від 75001 до 100000. У такому випадку кожне ядро виконує однаковий тип операцій, але над різними частинами даних. Це відповідає принципу паралелізму за даними, який добре підходить для спеціалізованого багатоядерного процесора.

Водночас у моделі враховано, що реальний час виконання не зменшується абсолютно пропорційно. Окрім самої обробки даних, виникають витрати на передавання фрагментів до ядер, доступ до пам'яті та синхронізацію результатів. Для прикладу, якщо передавання одного елемента умовно оцінюється як 0,05 такту, то для задачі з 100000 елементів витрати на передавання становлять 5000 умовних тактів незалежно від кількості ядер. Крім цього, синхронізація після паралельної фази також додає затримку. У моделі її можна прийняти як 100 умовних тактів базової затримки та ще 10 умовних тактів на кожне активне ядро. Через це при використанні 1 ядра витрати синхронізації становлять 110 умовних тактів, при 2 ядрах - 120, при 4 ядрах - 140, а при 8 ядрах - 180 умовних тактів.

З урахуванням цих умовних параметрів можна отримати орієнтовне уявлення про поведінку моделі. Для одного ядра загальний час виконання становить приблизно 105110 умовних тактів, оскільки до 100000 тактів обчислення додаються витрати на передавання даних і синхронізацію. Для двох ядер час зменшується приблизно до 55120 умовних тактів. Для чотирьох ядер він становить близько 30140 умовних тактів, а для восьми ядер - близько 17680 умовних тактів. Ці значення показують, що збільшення кількості ядер справді скорочує час виконання, але не в ідеальній пропорції. Причиною є накладні витрати, які залишаються навіть тоді, коли обчислювальна частина задачі добре розподілена.

Реалізована логіка розподілу також дозволяє оцінити коефіцієнт прискорення. Якщо порівняти час виконання на одному ядрі з часом виконання на кількох ядрах, то для двоядерної конфігурації прискорення становить приблизно 1,91 раза. Для чотирьох ядер воно наближається до 3,49 раза, а для восьми ядер - до 5,95 раза. Це підтверджує, що модель демонструє зростання продуктивності при збільшенні кількості ядер, однак реальний приріст обмежується витратами на організацію паралельної роботи.

Важливим елементом реалізації є збереження службової інформації про кожен фрагмент. Для кожного ядра фіксується його номер, початкова і кінцева позиція фрагмента, кількість елементів, орієнтовний час обробки та час завершення роботи. Завдяки цьому після завершення виконання можна визначити, яке ядро працювало найдовше, чи був розподіл рівномірним і наскільки ефективно використано обчислювальні ресурси. Наприклад, у конфігурації з 4 ядрами при обсязі 100000 елементів усі ядра отримують по 25000 елементів, тому час їх роботи майже однаковий. Якщо ж обсяг задачі або складність фрагментів відрізняється, модель дозволяє побачити появу дисбалансу.

У програмній реалізації блок керування задачами не виконує самі обчислення, а лише організовує їх передавання ядрам. Це дає змогу чітко відокремити логіку розподілу від логіки обробки. Такий поділ робить модель зрозумілішою та зручнішою для подальшого розширення. Наприклад, базовий рівномірний розподіл можна замінити динамічним, коли ядро після завершення одного фрагмента отримує наступний. Проте для цієї кваліфікаційної роботи статичний розподіл є достатнім, оскільки він дозволяє просто й наочно показати вплив кількості ядер на продуктивність.

Окремо враховано зв'язок між розподілом задачі та підсистемою пам'яті. Якщо кожне ядро працює з власним фрагментом даних, кількість звернень до спільної пам'яті зменшується. У такому режимі основне навантаження припадає на початкове завантаження даних і завершальне збереження результатів. Якщо

ж задача потребує частого доступу до спільних змінних, ефективність паралельного виконання зменшується через зростання затримок пам'яті. Саме тому в моделі обрано такий тип задачі, де більша частина обробки виконується локально, а звернення до спільної пам'яті зведені до мінімуму.

3.3 Реалізація механізму імітації паралельного виконання

У програмній реалізації кожне обчислювальне ядро подано як окремий логічний виконавець, який отримує власний фрагмент даних і виконує над ним однакову послідовність операцій. Такий підхід відповідає моделі паралелізму за даними, коли загальна задача залишається однією, але її вхідний набір поділяється між кількома ядрами. Наприклад, для задачі обробки масиву з 100000 елементів чотириядерна конфігурація передбачає, що кожне ядро отримує по 25000 елементів. У восьмиядерній конфігурації обсяг фрагмента для одного ядра зменшується до 12500 елементів. Це дозволяє зменшити час безпосередньої обробки, однак водночас у модель внесено додаткові витрати, які виникають під час організації паралельного виконання.

Механізм імітації побудовано так, щоб загальний час паралельної фази визначався не сумою часу роботи всіх ядер, а часом завершення найповільнішого ядра. Це важливо, оскільки в реальній багатоядерній системі результати не можуть бути об'єднані доти, доки всі ядра не завершать обробку своїх фрагментів. Якщо одне ядро працює довше за інші, решта ядер переходить у стан очікування. У моделі це враховано через визначення максимального часу виконання серед усіх активних ядер. Саме цей показник приймається як тривалість основної паралельної частини.

Для базового сценарію прийнято, що обробка одного елемента потребує 1 умовного такту. Якщо задача виконується на одному ядрі, то обчислювальна частина займає 100000 умовних тактів. При використанні двох ядер обчислювальний час одного ядра зменшується до 50000 умовних тактів, при

використанні чотирьох ядер - до 25000 умовних тактів, а при використанні восьми ядер - до 12500 умовних тактів. Таке зменшення показує базову перевагу паралельного виконання, однак у моделі додатково враховано, що реальний час роботи залежить не лише від кількості операцій, а й від витрат на передавання даних і синхронізацію.

У межах імітаційної моделі передбачено два основні типи доступу до даних. Перший тип пов'язаний із локальним кешем ядра. Якщо дані вже розміщені в локальному кеші, ядро виконує обробку без значної затримки. Другий тип пов'язаний зі зверненням до спільної пам'яті. Такий доступ є повільнішим, оскільки запит проходить через внутрішню комунікаційну підсистему та контролер пам'яті. Для моделювання прийнято, що локальний доступ додає незначну затримку, а звернення до спільної пам'яті оцінюється як дорожча операція. Наприклад, для базового сценарію передавання даних до ядер оцінено як 0,05 умовного такту на один елемент. Через це для масиву з 100000 елементів сумарні витрати на передавання становлять приблизно 5000 умовних тактів.

Паралельне виконання в моделі проходить у кілька послідовних етапів. Спочатку блок керування формує фрагменти задачі та передає їх ядрам. Далі ядра одночасно виконують обробку своїх фрагментів. Під час цієї фази кожне ядро працює переважно з власними даними, що зменшує кількість звернень до спільної пам'яті. Після завершення локальної обробки ядра повертають часткові результати. Потім виконується синхронізація, під час якої система перевіряє завершення роботи всіх активних ядер. Лише після цього результати можуть бути об'єднані в загальний підсумковий результат.

Схему імітації паралельної роботи обчислювальних ядер подано на рисунку 3.3. На ньому показано, що після розподілу задачі кожне ядро отримує власний фрагмент даних і виконує його обробку незалежно від інших ядер. Після завершення обчислень часткові результати надходять до блока синхронізації, а

потім передаються до модуля об'єднання результатів. Така схема відображає повний цикл паралельного виконання, який реалізовано в програмній моделі.

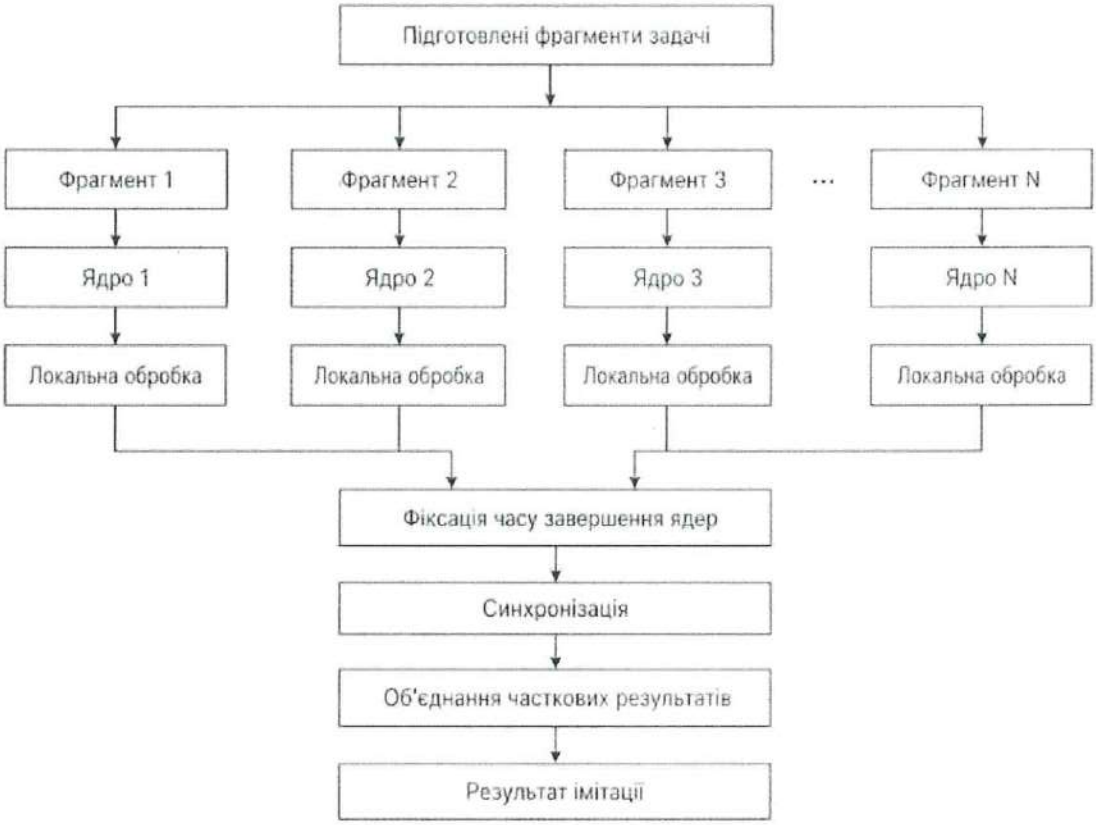


Рисунок 3.3 – Схема імітації паралельної роботи обчислювальних ядер

Окрему увагу в механізмі імітації приділено синхронізації. У моделі прийнято, що після завершення основної обчислювальної фази додається фіксована затримка синхронізації. Для базового сценарію вона складається зі 100 умовних тактів і додаткових 10 умовних тактів на кожне активне ядро. У результаті для одноядерного режиму витрати синхронізації становлять 110 умовних тактів, для двох ядер - 120 умовних тактів, для чотирьох ядер - 140 умовних тактів, а для восьми ядер - 180 умовних тактів. Це дозволяє показати, що зі збільшенням кількості ядер зростає не лише потенційна швидкодія, а й витрати на узгодження результатів.

У програмній моделі така логіка дозволяє отримати більш реалістичну оцінку продуктивності. Якщо враховувати лише обчислювальну частину, восьмиядерна конфігурація мала б зменшити час виконання приблизно у 8 разів. Однак після додавання витрат на передавання даних і синхронізацію фактичне прискорення є меншим. Для базового прикладу з 100000 елементів одноядерний режим дає орієнтовно 105110 умовних тактів загального часу. Двоядерна конфігурація зменшує цей показник приблизно до 55120 умовних тактів, чотириядерна - до 30140 умовних тактів, а восьмиядерна - до 17680 умовних тактів. Такі значення показують, що паралельне виконання суттєво скорочує час обробки, але не усуває накладні витрати повністю.

У механізмі імітації також враховано можливий дисбаланс навантаження. Якщо всі ядра отримують однакові фрагменти, час їх завершення є близьким. Наприклад, при обробці 100000 елементів на 4 ядрах кожне ядро обробляє по 25000 елементів, тому паралельна фаза завершується майже одночасно для всіх ядер. Якщо ж обсяг задачі становить 100003 елементи, перші три ядра отримують по 25001 елементу, а четверте - 25000 елементів. У такому випадку різниця становить лише 1 елемент, тому вплив на загальний час є мінімальним. Проте модель дозволяє показати, що при більш нерівномірному розподілі затримка найповільнішого ядра може впливати на весь процес.

Для кожного ядра в процесі імітації фіксується кілька службових показників: номер ядра, розмір отриманого фрагмента, час локальної обробки, кількість умовних звернень до пам'яті та час завершення. Завдяки цьому після виконання задачі можна не лише отримати загальний час, а й оцінити поведінку кожного окремого ядра. Наприклад, якщо в конфігурації з 8 ядрами одне ядро завершує роботу значно пізніше за інші, це свідчить про нерівномірний розподіл або більшу кількість звернень до пам'яті в його фрагменті. У базовому рівномірному сценарії таких значних відхилень не виникає, що підтверджує коректність обраного способу розподілу.

Імітація паралельного виконання також пов'язана з об'єднанням результатів. Після завершення роботи ядер часткові результати надходять до окремого модуля, який формує загальний підсумок. У моделі цей етап не вважається миттєвим, оскільки в реальних системах об'єднання результатів також потребує часу. Для простих задач цей час може бути невеликим, але при збільшенні обсягу даних або кількості ядер його вплив поступово зростає. Через це в моделі він розглядається як частина накладних витрат, що обмежують ідеальне масштабування.

Реалізований механізм дозволяє перевіряти різні режими роботи процесора. Перший режим відповідає майже ідеальній паралельній задачі, де ядра працюють незалежно і майже не звертаються до спільної пам'яті. Другий режим передбачає помірну кількість звернень до пам'яті, що додає затримки до часу виконання. Третій режим може використовуватися для перевірки гіршого випадку, коли ядра часто очікують доступу до спільного ресурсу. Така побудова дає змогу не лише показати переваги багатоядерної архітектури, а й виявити обмеження, які виникають під час активного обміну даними.

У межах цієї кваліфікаційної роботи основний акцент зроблено на базовому сценарії, де задача добре поділяється на незалежні фрагменти. Це дозволяє продемонструвати потенціал спеціалізованого багатоядерного процесора для задач обробки масивів даних. Разом із тим у модель включено затримки передавання та синхронізації, тому результати не є ідеалізованими. Вони показують, що зі збільшенням кількості ядер час виконання зменшується, але після певного рівня вплив накладних витрат стає помітнішим.

3.4 Аналіз часу виконання та коефіцієнта прискорення

Після реалізації механізму імітації паралельного виконання виконано аналіз часу роботи програмної моделі спеціалізованого багатоядерного процесора для різної кількості обчислювальних ядер. Основна увага приділена

тому, як змінюється загальний час виконання задачі при переході від одноядерної конфігурації до багатоядерної, а також наскільки отриманий приріст продуктивності наближається до очікуваного результату. У межах моделі продуктивність оцінюється не лише за кількістю ядер, а й з урахуванням додаткових витрат, пов'язаних із передаванням даних, синхронізацією результатів і завершальним об'єднанням часткових обчислень.

Для аналізу використано базову задачу обробки масиву зі 100000 елементів. Такий обсяг даних дозволяє показати різницю між конфігураціями з 1, 2, 4 та 8 ядрами. За основу прийнято, що обробка одного елемента потребує 1 умовного такту. У такому разі одноядерна конфігурація виконує основну обчислювальну частину за 100000 умовних тактів. Однак до цього часу додано витрати на передавання даних і синхронізацію. Через це повний час виконання задачі на одному ядрі становить приблизно 105110 умовних тактів.

При переході до двоядерної конфігурації задача поділяється на два однакові фрагменти по 50000 елементів. Кожне ядро обробляє власну частину масиву, тому основна обчислювальна фаза скорочується майже вдвічі. Разом із витратами на передавання даних і синхронізацію повний час виконання становить близько 55120 умовних тактів. Це означає, що використання двох ядер дозволяє скоротити час виконання приблизно на 49 % порівняно з одноядерним режимом. Отриманий результат уже демонструє практичну доцільність розподілу задачі між кількома ядрами.

У чотириядерній конфігурації кожне ядро отримує фрагмент із 25000 елементів. За рахунок цього основна обчислювальна частина стає ще коротшою, а загальний час виконання знижується до 30140 умовних тактів. Порівняно з одноядерним режимом час роботи зменшується більш ніж утричі. При цьому приріст продуктивності залишається достатньо високим, оскільки витрати на синхронізацію ще не переважають над виграшем від паралельного виконання. Саме така конфігурація добре показує баланс між кількістю ядер і накладними витратами.

Для восьмиядерної конфігурації обсяг фрагмента для одного ядра становить 12500 елементів. Це дає змогу ще більше скоротити обчислювальну частину, і загальний час виконання зменшується до 17680 умовних тактів. Порівняно з одним ядром модель показує значне прискорення, однак приріст уже не є пропорційним кількості ядер. Якщо ідеальна восьмиядерна система мала б дати прискорення у 8 разів, то в моделі отримано приблизно 5,95 раза. Це пояснюється наявністю витрат, які не зникають після збільшення кількості ядер.

Коефіцієнт прискорення розраховано шляхом порівняння часу виконання задачі на одному ядрі з часом виконання на багатоядерних конфігураціях. Для одного ядра коефіцієнт прискорення приймається як 1,00, оскільки це базовий режим. Для двох ядер він становить приблизно 1,91, для чотирьох ядер - близько 3,49, а для восьми ядер - близько 5,95. Такі значення показують, що збільшення кількості ядер позитивно впливає на швидкодію, однак кожне наступне подвоєння кількості ядер дає дещо менший відносний приріст.

Важливим показником є ефективність використання ядер. Для двоядерної конфігурації вона становить приблизно 95 %, оскільки отримане прискорення 1,91 є близьким до ідеального значення 2. Для чотирьох ядер ефективність знижується приблизно до 87 %, оскільки прискорення 3,49 уже дещо відстає від ідеального значення 4. Для восьми ядер ефективність становить близько 74 %, оскільки отримане прискорення 5,95 помітно нижче за теоретично можливе значення 8. Це свідчить про те, що зі збільшенням кількості ядер зростає вплив витрат, пов'язаних із керуванням паралельним виконанням.

Окремо варто зазначити, що отримане зниження ефективності не є ознакою помилки моделі. Навпаки, така поведінка є типовою для багатоядерних систем. У реальних процесорах продуктивність також не масштабується ідеально, оскільки існують послідовні ділянки виконання, обмеження підсистеми пам'яті, затримки комунікаційної підсистеми та витрати на синхронізацію потоків. Саме тому в моделі спеціально враховано додаткові затримки, щоб результати не виглядали штучно завищеними.

У межах виконаного аналізу також видно, що багатоядерна архітектура є найбільш ефективною для задач із великим обсягом даних. Якщо масив містить 100000 елементів, обчислювальна частина є достатньо великою, щоб виграш від паралельного виконання переважав витрати на організацію роботи ядер. Для значно меншого масиву, наприклад 1000 або 2000 елементів, накладні витрати могли б становити помітнішу частину загального часу. У такому випадку використання великої кількості ядер не завжди забезпечувало б очікуване прискорення.

Для розробленої моделі важливо, що результати змінюються логічно й послідовно. При збільшенні кількості ядер час виконання зменшується, коефіцієнт прискорення зростає, але ефективність використання кожного ядра поступово знижується. Це означає, що модель відображає не лише позитивний ефект багатоядерності, а й її природні обмеження. Такий результат є важливим для кваліфікаційної роботи, оскільки він показує реалістичний характер побудованої програмної імітації.

У практичному сенсі отримані результати підтверджують доцільність використання спеціалізованого багатоядерного процесора для задач, які добре поділяються на незалежні фрагменти. Якщо обчислювальне навантаження можна рівномірно розподілити між ядрами, продуктивність системи суттєво зростає. Для базового сценарію перехід від одного ядра до восьми дозволив скоротити час виконання з 105110 до 17680 умовних тактів. Це означає зменшення часу більш ніж у п'ять разів, навіть з урахуванням накладних витрат.

Разом із тим результати показують, що подальше збільшення кількості ядер потребує уважнішого врахування підсистеми пам'яті та механізмів синхронізації. Якщо кількість ядер збільшити до 16, можна очікувати подальшого скорочення обчислювальної частини, але витрати на координацію також зростатимуть. Через це після певного рівня додавання нових ядер може давати все менший приріст продуктивності. Такий висновок добре узгоджується з логікою масштабування багатоядерних архітектур.

Аналіз часу виконання та коефіцієнта прискорення показав, що реалізована програмна модель спеціалізованого багатоядерного процесора демонструє очікуване підвищення продуктивності при збільшенні кількості ядер. Для задачі обсягом 100000 елементів отримано скорочення часу виконання з 105110 умовних тактів в одноядерному режимі до 17680 умовних тактів у восьмиядерному режимі. Коефіцієнт прискорення зріс від 1,00 до 5,95, що підтверджує ефективність паралельної обробки. Водночас зниження ефективності використання ядер при масштабуванні показує вплив накладних витрат і створює основу для подальшого аналізу затримок пам'яті та синхронізації.

3.5 Оцінювання впливу затримок пам'яті та синхронізації на продуктивність

Після аналізу часу виконання та коефіцієнта прискорення додатково оцінено вплив затримок пам'яті та синхронізації на продуктивність спеціалізованого багатоядерного процесора. Такий етап є важливим, оскільки збільшення кількості ядер не завжди забезпечує пропорційне зростання швидкодії. Навіть якщо задача добре поділена на фрагменти, частина часу витрачається не на самі обчислення, а на передавання даних, звернення до спільної пам'яті, очікування завершення роботи інших ядер та об'єднання результатів. Саме ці витрати пояснюють, чому восьмиядерна конфігурація в попередньому підрозділі дала прискорення не у 8 разів, а приблизно у 5,95 раза.

У програмній моделі враховано, що кожне ядро працює з локальним фрагментом задачі, але не може повністю ізолюватися від підсистеми пам'яті. Перед початком обробки фрагменти даних мають бути передані ядрам, а після завершення виконання часткові результати мають бути збережені та синхронізовані. Для базового сценарію використано масив із 100000 елементів, де обробка одного елемента займає 1 умовний такт. Додатково враховано

витрати на передавання даних у розмірі 0,05 умовного такту на один елемент. Через це незалежно від кількості ядер загальні витрати на передавання всього масиву становлять приблизно 5000 умовних тактів.

Це значення добре показує одну з особливостей багатоядерної архітектури: зі збільшенням кількості ядер обчислювальна частина задачі зменшується, але витрати на передавання даних не зникають. Наприклад, в одноядерному режимі 5000 умовних тактів на передавання є відносно невеликою частиною порівняно зі 100000 умовних тактів основної обробки. У двоядерному режимі обчислювальна частина для одного ядра становить 50000 умовних тактів, тому частка витрат на передавання вже стає помітнішою. У восьмиядерному режимі обчислювальна частина одного ядра зменшується до 12500 умовних тактів, а витрати на передавання залишаються на рівні 5000 умовних тактів, тобто займають значно більшу частину загального часу.

У базовому сценарії для одного ядра загальний час виконання становить 105110 умовних тактів. Із цього значення 100000 умовних тактів припадає на обчислення, 5000 умовних тактів - на передавання даних, а 110 умовних тактів - на синхронізацію. Частка накладних витрат у такому режимі становить трохи менше 5 %. Для двох ядер загальний час зменшується до 55120 умовних тактів, але накладні витрати вже становлять 5120 умовних тактів, тобто приблизно 9 % від загального часу. Для чотирьох ядер загальний час становить 30140 умовних тактів, а накладні витрати - 5140 умовних тактів, що дорівнює приблизно 17 %. Для восьми ядер загальний час становить 17680 умовних тактів, а накладні витрати - 5180 умовних тактів, тобто майже 29 % від загального часу.

Такі числові результати показують, що зі збільшенням кількості ядер частка корисного обчислювального часу зменшується, а відносна частка накладних витрат зростає. Це не означає, що використання більшої кількості ядер є недоцільним. Навпаки, восьмиядерна конфігурація все одно виконує задачу значно швидше за одноядерну. Проте отримані значення демонструють,

що після певного рівня масштабування саме пам'ять і синхронізація починають помітніше обмежувати подальше зростання продуктивності.

На рисунку 3.4 подано узагальнену схему впливу накладних витрат на ефективність паралельного виконання. У цій схемі доцільно показати, що загальний час виконання формується з трьох основних частин: часу обчислень, витрат на доступ до пам'яті та витрат на синхронізацію. Також варто відобразити, що при збільшенні кількості ядер обчислювальна частина скорочується, але накладні витрати залишаються або поступово зростають. Саме це пояснює відхилення реального прискорення від ідеального.

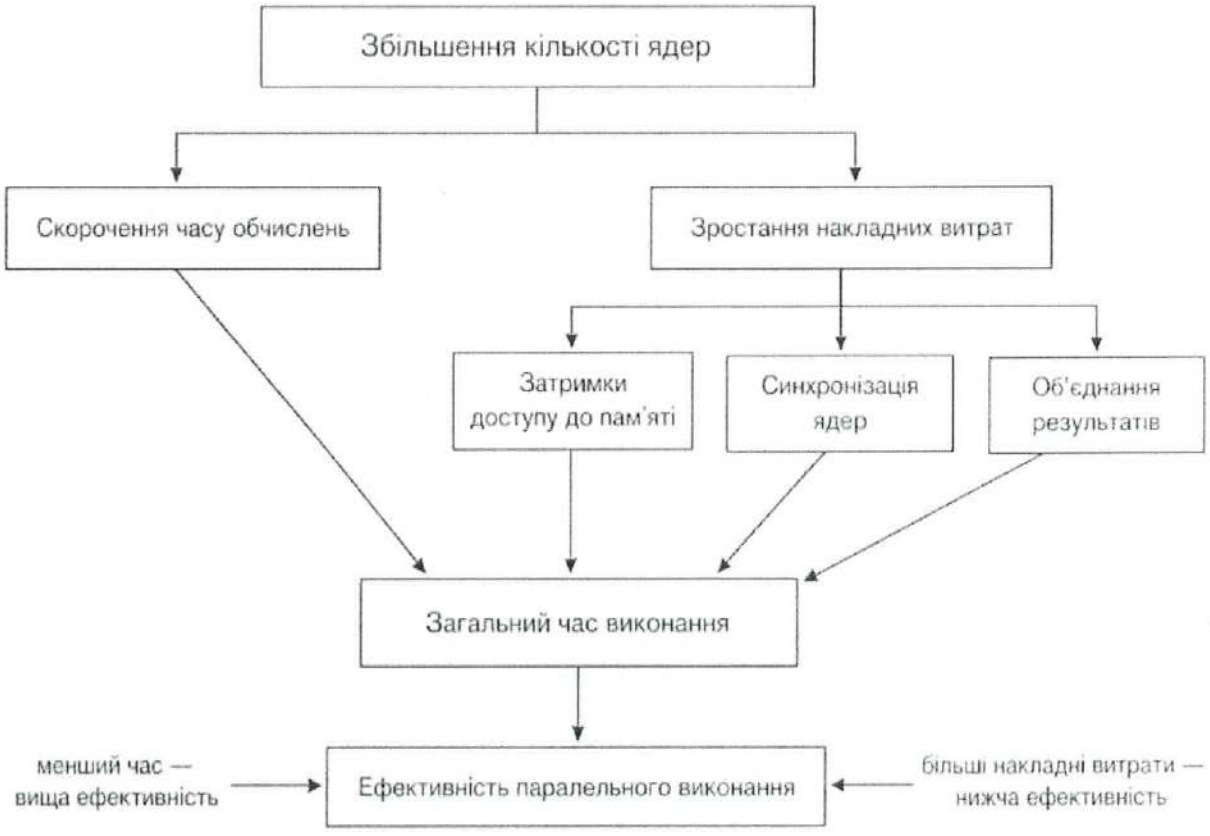


Рисунок 3.4 – Вплив накладних витрат на ефективність паралельного виконання

Вплив затримок пам'яті особливо помітний у конфігураціях із більшою кількістю ядер. Якщо одне ядро працює з усім масивом, то воно послідовно

виконує обробку і не створює конкуренції з іншими ядрами за доступ до спільної пам'яті. У багатоядерному режимі кілька ядер одночасно потребують отримання фрагментів даних або запису результатів. Навіть якщо сама модель використовує умовну оцінку затримок, вона відображає загальну логіку реальних багатоядерних систем: спільна пам'ять має обмежену пропускну здатність, а одночасні запити можуть створювати чергу.

Для зменшення впливу пам'яті у моделі використано локальний кеш кожного ядра. Його роль полягає в тому, щоб частина даних після завантаження оброблялася без постійного звернення до спільної пам'яті. Якщо ядро отримало свій фрагмент і більшу частину часу працює локально, ефективність паралельного виконання зростає. У базовому сценарії саме такий підхід дозволив отримати прискорення 5,95 раза для восьми ядер.

Для наочності можна порівняти два умовні сценарії. У першому сценарії задача поділена рівномірно, і всі 4 ядра обробляють по 25000 елементів. У цьому випадку паралельна фаза завершується приблизно за однаковий час для всіх ядер. У другому сценарії одне ядро отримує 40000 елементів, а решта три ядра - по 20000 елементів. Попри те, що загальний обсяг задачі залишається тим самим, час паралельної фази визначається ядром, яке отримало найбільший фрагмент. Через це інші ядра завершують роботу раніше й очікують синхронізації. Такий приклад показує, що рівномірний розподіл задачі є не менш важливим, ніж сама кількість ядер.

У межах розробленої моделі найбільш збалансованою виглядає конфігурація з 4 ядрами. Вона забезпечує прискорення близько 3,49 раза, а частка накладних витрат становить приблизно 17 %. Восьмиядерна конфігурація є швидшою за абсолютним часом, оскільки виконує задачу за 17680 умовних тактів, однак її ефективність нижча, а частка накладних витрат зростає майже до 29 %.

ВИСНОВКИ

У кваліфікаційній роботі розглянуто питання моделювання спеціалізованого багатоядерного процесора підвищеної продуктивності та виконано аналіз його роботи за основними показниками швидкодії. Актуальність теми зумовлена тим, що сучасні обчислювальні системи дедалі частіше потребують не простого збільшення тактової частоти, а раціональної організації паралельної обробки даних. Саме багатоядерні архітектури дозволяють підвищити продуктивність за рахунок одночасного виконання кількох фрагментів задачі, однак їх ефективність залежить від організації пам'яті, розподілу навантаження, синхронізації та витрат на обмін даними.

У першому розділі проаналізовано еволюцію процесорних архітектур і передумови переходу до багатоядерності. Показано, що традиційне підвищення продуктивності шляхом збільшення тактової частоти поступово зіткнулося з фізичними, тепловими та енергетичними обмеженнями. Через це подальший розвиток процесорів почав орієнтуватися на структурний паралелізм, тобто на використання кількох обчислювальних ядер у межах одного процесора. Також розглянуто класифікацію багатоядерних процесорів, принципи паралельних обчислень, масштабованість, організацію між'ядерної взаємодії та методи моделювання багатоядерних систем.

У другому розділі сформовано модель спеціалізованого багатоядерного процесора підвищеної продуктивності. Запропонована структура включає блок керування задачами, декілька обчислювальних ядер, локальні кеші, внутрішню комунікаційну підсистему, контролер пам'яті та спільну пам'ять. Така архітектура дозволяє подати процесор як цілісну обчислювальну систему, у якій кожне ядро виконує окремий фрагмент задачі, а обмін даними між компонентами здійснюється через спільну підсистему пам'яті.

У третьому розділі реалізовано програмну модель спеціалізованого багатоядерного процесора та проведено оцінювання її продуктивності. Модель

побудовано за модульним принципом, що дозволило окремо подати блок вхідних параметрів, блок керування задачами, обчислювальні ядра, локальний кеш, спільну пам'ять, модуль збирання результатів і модуль оцінювання продуктивності. Така організація забезпечила можливість змінювати кількість ядер, обсяг задачі та параметри накладних витрат без повної перебудови програмної структури.

Для перевірки роботи моделі використано задачу обробки масиву зі 100000 елементів. Розглянуто конфігурації з 1, 2, 4 та 8 обчислювальними ядрами. При одному ядрі весь масив обробляється одним обчислювальним блоком, при двох ядрах задача поділяється на фрагменти по 50000 елементів, при чотирьох - по 25000 елементів, а при восьми - по 12500 елементів. Такий розподіл забезпечив рівномірне навантаження та дозволив оцінити вплив кількості ядер на час виконання задачі.

Практичне значення кваліфікаційної роботи полягає в тому, що реалізована програмна модель може застосовуватися для попереднього аналізу багатоядерних архітектур без створення складного апаратного прототипу. Вона дозволяє швидко оцінювати вплив кількості ядер, затримок пам'яті та витрат синхронізації на загальну продуктивність. Такий підхід є корисним на початкових етапах проєктування спеціалізованих обчислювальних модулів для задач обробки масивів даних, технічного моделювання та керування промисловими системами.

У підсумку мету кваліфікаційної роботи досягнуто. Проаналізовано теоретичні засади побудови багатоядерних процесорів, сформовано архітектуру спеціалізованого багатоядерного процесора, реалізовано програмну модель його роботи та виконано оцінювання продуктивності для різної кількості обчислювальних ядер. Отримані результати підтвердили, що спеціалізована багатоядерна архітектура здатна забезпечити помітне підвищення швидкодії за умови раціонального розподілу задачі та контрольованих накладних витрат.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Hennessy J. L., Patterson D. A., Kozyrakis C. Computer Architecture: A Quantitative Approach. 7th ed. Cambridge : Morgan Kaufmann, 2025. URL: https://shop.elsevier.com/books/computer-architecture/hennessy/978-0-443-15406-5.

2. Patterson D. A., Hennessy J. L. Computer Organization and Design RISC-V Edition: The Hardware Software Interface. 2nd ed. Cambridge : Morgan Kaufmann, 2020. 736 p. URL: https://www.amazon.com/Computer-Organization-Design-RISC-V-Architecture/dp/0128203315.

3. Harris S. L., Harris D. Digital Design and Computer Architecture: RISC-V Edition. 1st ed. Cambridge : Morgan Kaufmann, 2021. 592 p. DOI: 10.1016/C2019-0-00213-0

4. RISC-V International. The RISC-V Instruction Set Manual. Volume I: Unprivileged ISA *Specification.* RISC-V Ratified Specifications Library. 2024. URL: https://docs.riscv.org/reference/isa/unpriv/unpriv-index.html.

5. RISC-V International. The RISC-V Instruction Set Manual. Volume II: Privileged Architecture. *RISC-V* Ratified Specifications Library. 2021. URL: https://docs.riscv.org/reference/isa/index.html.

6. RISC-V International. RISC-V Profiles. RISC-V Ratified Specifications Library. 2023. URL: https://docs.riscv.org/reference/profiles-overview/index.html.

7. RISC-V International. Ratified Specifications. *RISC-V* International. 2026. URL: https://riscv.org/specifications/ratified/ .

8. Arm Ltd. AMBA Specifications: AMBA 5 CHI, AXI. Arm Developer. 2026. URL: https://www.arm.com/architecture/system-architectures/amba/amba-specifications.

9. Arm Ltd. AMBA CHI Architecture Specification. Arm Developer. 2024. URL: https://developer.arm.com/documentation/ihi0050/latest/.

10. UCIe Consortium. Universal Chiplet Interconnect Express (UCIe) Specification. UCIe Consortium. 2022. URL: https://www.uciexpress.org/specifications.

11. Sharma D. D., Pasdast G., Qian Z., Aygun K. Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations with Chiplets at Package Level. *IEEE Transactions on Components, Packaging and Manufacturing Technology*. 2022. Vol. 12, No. 10. P. 1423–1431. DOI: 10.1109/TCPMT.2022.3207195

12. UCIe Consortium. UCIe 2.0 Specification: Advancing an Open Ecosystem for Die-to-Die Interconnect. UCIe Consortium. 2024. URL: https://files.futurememorystorage.com/proceedings/2024/20240808_SPOS-301-1_UCIe_Consortium_Das_Sharma.pdf.

13. Lowe-Power J., Ahmad A. M., Akram A. et al. The gem5 Simulator: Version 20.0+. *arXiv*. 2020. DOI: 10.48550/arXiv.2007.03152

14. Semakin A. N. Simulation of a Multi-Core Computer System in the gem5 Simulator. AIP Conference Proceedings. 2021. DOI: 10.1063/5.0035841

15. Appold C. et al. Enabling gem5 for Side-Channel Power Attack Simulation of Cryptographic Algorithms. Proceedings of the ACM. 2025. DOI: 10.1145/3768725.3768730

16. Maiza C., Rihani H., Rivas J. M., Goossens J., Altmeyer S., Davis R. I. A Survey of Timing Verification Techniques for Multi-Core Real-Time Systems. *ACM Computing Surveys*. 2019. Vol. 52, No. 3. Article 56. DOI: 10.1145/3323212

17. Kaur S. P. et al. A Survey on Mapping and Scheduling Techniques for 3D Network-on-Chip. *Journal of Parallel and Distributed Computing*. 2024. DOI: 10.1016/j.jpdc.2023.104778

18. Asadi Y. et al. Optical Network-on-Chip (ONoC) Architectures: A Survey. *Journal of Semiconductors*. 2024. DOI: 10.1088/1674-4926/24060006

19. Alimi I. A., Patel A. Network-on-Chip Topologies: Potentials, Technical Challenges, Recent Advances and Research Direction. IntechOpen. 2021. DOI: 10.5772/intechopen.97262

20. Liu H., Du Y. et al. Survey of Chiplet Technology: SoC Architecture, Interconnect, EDA, and Advanced Packaging. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*. 2025. DOI: 10.1109/JETCAS.2025.3636408

21. Chen S., Zhang H., Ling Z., Zhai J., Yu B. The Survey of 2.5D Integrated Architecture: An EDA Perspective. Proceedings of the 30th Asia and South Pacific Design Automation Conference. 2025. DOI: 10.1145/3658617.3703134

22. Li F. et al. Chiplet Design Automation: Methodologies, Advances, and Challenges. ACM Transactions on Design Automation of Electronic Systems. 2026. DOI: 10.1145/3796529

23. Arya N. Towards Cache-Coherent Chiplet-Based Architectures. Universitat Politècnica de Catalunya. 2023. URL: https://upcommons.upc.edu/bitstreams/99b4954a-05ab-4833-8823-2c2c8e9b50d5/download.

24. Hill M. D., Sorin D. J. *A* Primer on Memory Consistency and Cache Coherence. 2nd ed. 2020. URL: https://pages.cs.wisc.edu/~markhill/papers/primer2020_2nd_edition.pdf.

25. Sensfelder N. et al. On How to Identify Cache Coherence: Case of the NXP QorIQ Platform. Leibniz International Proceedings in Informatics. 2020. DOI: 10.4230/LIPIcs.ECRTS.2020.13

26. Zhao Y. et al. Research on Cache Coherence Protocol Verification Method Based on Model Checking. *Electronics*. 2023. Vol. 12, No. 16. 3420. DOI: 10.3390/electronics12163420

27. Lezhnev E. V. et al. Reduction Method for a Network-on-Chip Low-Level Modeling Tool. *Micromachines*. 2025. Vol. 16, No. 10. 1096. DOI: 10.3390/mi16101096

28. Tsiramua S. et al. Structural Analysis of Multi-Core Processor and Reliability Indicators. *Mathematics*. 2025. Vol. 13, No. 3. 515. DOI: 10.3390/math13030515

29. Guo B. et al. Network-on-Chip Applications for IoT-Enabled Systems: A Review. *International Journal of Electronics*. 2025. DOI: 10.1142/S0129156425400270

30. Mone G. The Chiplet Revolution. *Communications of the ACM*. 2024. Vol. 67, No. 11. P. 14–16. DOI: 10.1145/3686310

31. Braun R. et al. Expanding Coherence Protocol Stack with a Persistence Layer. ETH Zurich. 2023. URL: https://cores.inf.ethz.ch/files/PLayer.pdf.

32. Khan M. et al. Analyzing Flush+Fault Attack on RISC-V Using gem5. *SciTePress*. 2025. URL: https://www.scitepress.org/Papers/2025/135188/135188.pdf.

33. Intel Corporation. Intel® 64 and IA-32 Architectures Optimization Reference Manual. Intel Developer. 2023. URL: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html.

34. Advanced Micro Devices. AMD Software Optimization Guide for AMD EPYC™ Processors. *AMD* Developer. 2023. URL: https://www.amd.com/en/developer.

35. NVIDIA Corporation. CUDA C++ Programming Guide. NVIDIA Documentation. 2024. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/.

36. OpenMP Architecture Review Board. OpenMP Application Programming Interface Specification. OpenMP. 2023. URL: https://www.openmp.org/specifications/.

37. MPI Forum. MPI: A Message-Passing Interface Standard. MPI Forum. 2021. URL: https://www.mpi-forum.org/docs/.

38. LLVM Project. LLVM Documentation: Code Generation and Optimization. LLVM Documentation. 2025. URL: https://llvm.org/docs/.

39. GCC Team. GCC Documentation: Optimization Options. *GCC* Documentation. 2025. URL: https://gcc.gnu.org/onlinedocs/.

40. Malashonok H. I., Sidko A. A. Паралельні обчислення на розподіленій пам'яті: OpenMPI, Java, Math Partner. Київ : НаУКМА, 2020. 266 с. URL: https://www.ukma.edu.ua/index.php/science/naukovi-vidannya/monohrafii-pidruchnyky.

41. Kravchenko Yu. V., Leshchenko O. O., Herasymenko O. Yu., Trush O. V., Dakhno N. B. Архітектура комп'ютера. Київ : КНУ імені Тараса Шевченка, 2023. 220 с. URL: https://caravela.com.ua/index.php?product_id=258&route=product/product.

42. Matviienko M. P., Rozen V. P., Zakladnyi O. M. Архітектура комп'ютерів. Київ : Ліра-К, 2024. 264 с. URL: https://mybook.biz.ua/ua/informatika-1839/arhitektura-kompyutera/.

43. Huranich P. P. Комп'ютерна схемотехніка та архітектура комп'ютерів. Ужгород : УжНУ, 2024. URL: https://dspace.uzhnu.edu.ua/items/463efd09-2fa8-4b19-a658-3218d4d30c9d.

44. Lymarenko V. V. Комп'ютерні системи та архітектура комп'ютерів. Харків : ХНЕУ, 2022. URL: https://repository.hneu.edu.ua/bitstream/123456789/29016/1/поз_394_Комп'ютерні%20системи%20та%20архітектура%20комп'ютерів.pdf.

45. Zhi H. et al. A Methodology for Simulating Multi-Chiplet Systems Using Open-Source Simulators. Proceedings of the ACM International Conference on Computer-Aided Design. 2021. DOI: 10.1145/3477206.3477459

46. Ababei C., Moghaddam M. G. A Survey of Prediction and Classification Techniques in Multicore Processor Systems. *IEEE Transactions on Parallel and Distributed Systems*. 2019. Vol. 30, No. 5. P. 1184–1200. DOI: 10.1109/TPDS.2018.2878699

47. Alcorta E. S., Gerstlauer A. Learning-Based Phase-Aware Multi-Core CPU Workload Forecasting. ACM Transactions on Design Automation of Electronic Systems. 2022. DOI: 10.1145/3564929

48. Kondoth L., Shankaran R., Sheng Q. Z., Han R. Wireless Network-on-Chip Security Review: Attack Taxonomy, Implications, and Countermeasures. *IEEE Access*. 2023. Vol. 11. P. 122876–122892. DOI: 10.1109/ACCESS.2023.3329572

49. Dauda A., Flauzac O., Nolot F. A Survey on IoT Application Architectures. *Sensors*. 2024. Vol. 24, No. 16. 5320. DOI: 10.3390/s24165320

50. Weng X. et al. A Machine Learning Mapping Algorithm for NoC Optimization. *Symmetry*. 2023. Vol. 15, No. 3. 593. DOI: 10.3390/sym15030593

51. Jagadheesh S. et al. NoC Application Mapping Optimization Using Reinforcement Learning. ACM Transactions on Embedded Computing Systems. 2022. DOI: 10.1145/3510381

52. Magyari A. et al. Review of State-of-the-Art FPGA Applications in IoT Networks. *Electronics*. 2022. Vol. 11, No. 19. 3023. DOI: 10.3390/electronics11193023

53. Agrawal A. An Architectural Power Model for Networks on Chip. University of California, Berkeley. 2023. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-168.pdf.

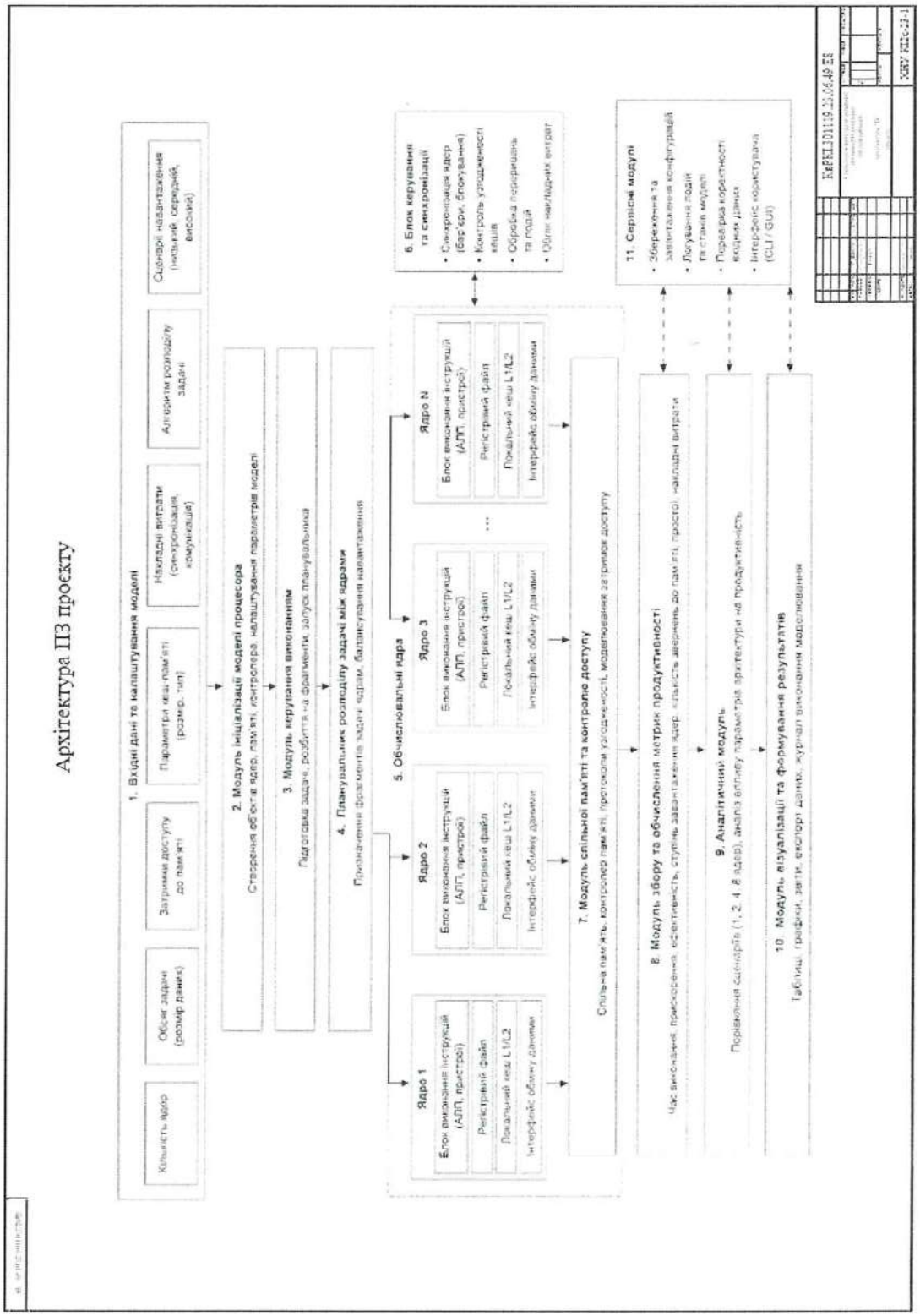
54. Paris P. C. D. et al. A High-Level Simulator for Network-on-Chip. Integrated Computer-Aided Engineering. 2025. DOI: 10.3233/ICA-240743

55. Asadi Y. A Comprehensive Study and Holistic Review of Empowering Network-on-Chip Application Mapping through Machine Learning Techniques. *Journal of Reliable Intelligent Environments*. 2024. DOI: 10.1007/s44291-024-00027-w

ДОДАТОК А

(обов'язковий)

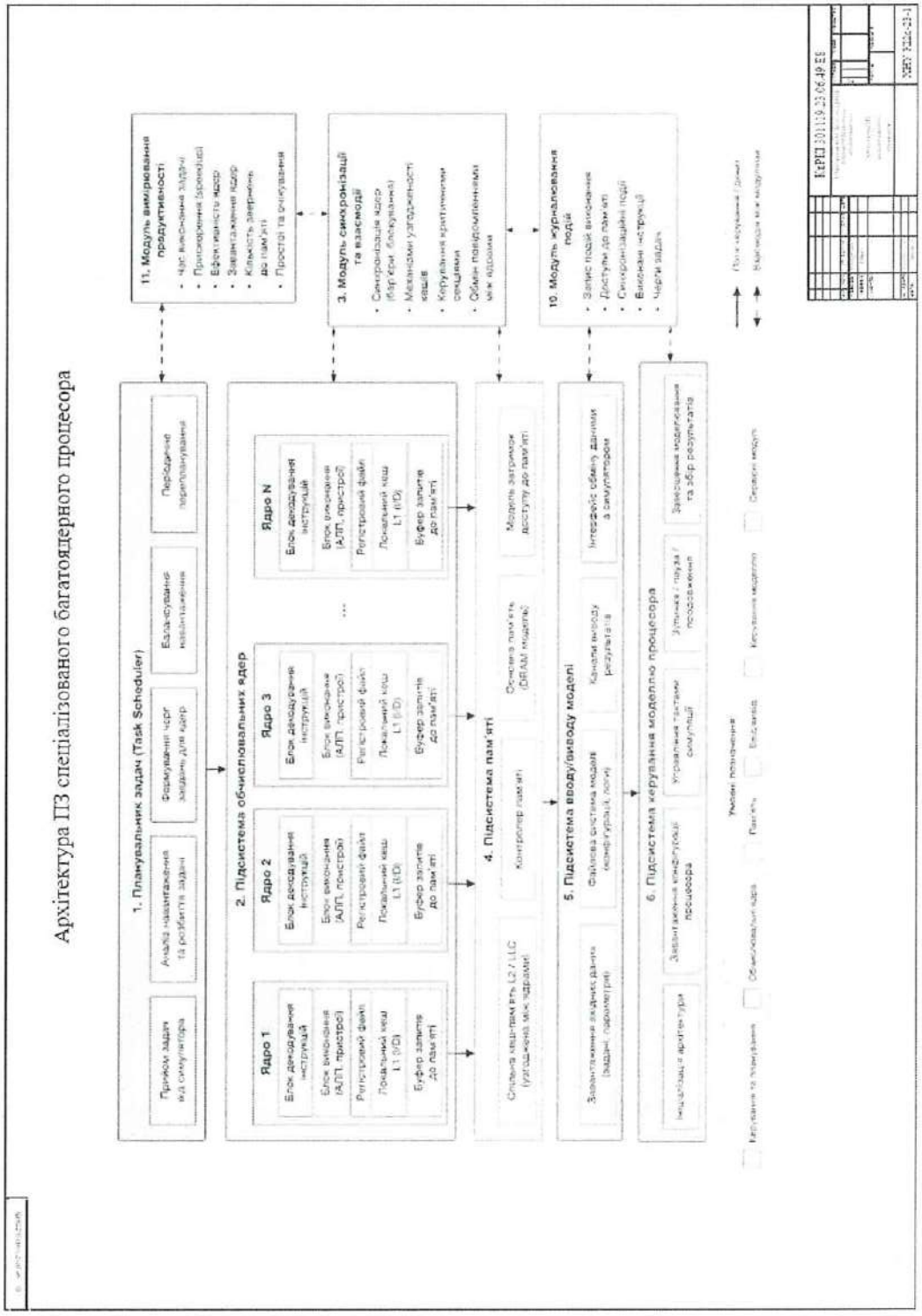
Копія креслення «Архітектура ПЗ проєкту»



ДОДАТОК Б

(обов'язковий)

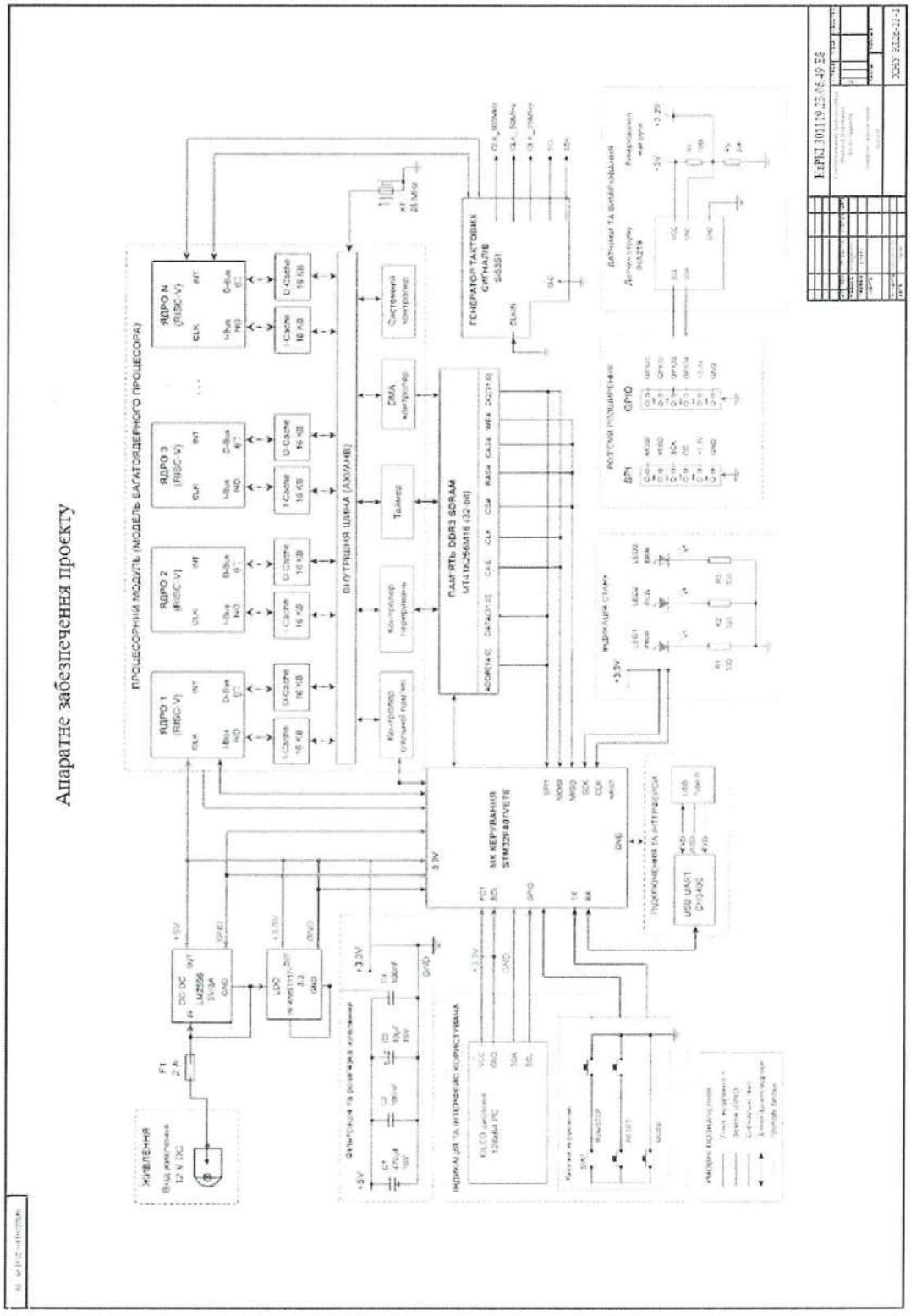
Копія креслення «Архітектура ПЗ спеціалізованого багатоядерного процесора»



ДОДАТОК В

(обов'язковий)

Копія креслення «Апаратне забезпечення проєкту»



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 МУШИНСЬКИЙ

Співавтор:

Назва: Спеціалізований багатоядерний процесор підвищеної продуктивності

Експерт: Олексій ЛИГУН

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:2.26%

Коефіцієнт подібності 2:0.58%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-27 20:49:06.0

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

Thu May 28 08:41:43 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

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: 9%**

ID: 272536 Назва: БКР Спеціалізований багатоядерний процесор підвищеної продуктивності Додано в БД: 2026-05-28 Автора: Ярослав МУШИНСЬКИЙ Керівники: Олексій ЛИГУН Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	114178	899	1429 (1%)	22 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Мушинський Ярослав Олександрович

Тема: Спеціалізований багатоядерний процесор підвищеної продуктивності

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 64

1. Короткий зміст роботи та прийнятих рішень: метою кваліфікаційної роботи є моделювання спеціалізованого багатоядерного процесора та проведення комплексного аналізу його продуктивності з урахуванням архітектурної організації та паралельного виконання обчислювальних задач.

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: в першому розділі кваліфікаційної роботи проведено детальний аналіз предметної області, досліджено еволюцію процесорних архітектур, передумови переходу до багатоядерності, класифікацію сучасних процесорів, принципи паралельних обчислень і масштабованості, а також методи моделювання багатоядерних систем. На основі цього виконано чітку постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено проєктування моделі спеціалізованого багатоядерного процесора. Сформовано загальну структуру обчислювальної системи, обґрунтовано архітектурні рішення для підвищення продуктивності, розроблено функціональну структуру обчислювального ядра, організовано обмін даними між ядрами та підсистемою пам'яті (локальними кешами та спільною пам'яттю), побудовано модель паралельного виконання обчислювальних задач та здійснено вибір програмних засобів для моделювання (мова Python). В третьому розділі кваліфікаційної роботи виконано реалізацію програмної структури моделі багатоядерного процесора та проведено ґрунтовний аналіз його продуктивності.

Реалізовано механізми імітації паралельного виконання обчислень, проаналізовано час виконання та коефіцієнти прискорення, а також виконано оцінювання впливу затримок пам'яті та бар'єрної синхронізації на загальну продуктивність процесора.

4. Позитивні сторони роботи: висока практична цінність роботи, яка полягає у створенні гнучкої програмної моделі для оцінювання параметрів масштабованості та накладних витрат комунікації у багатоядерних системах без необхідності складної апаратної реалізації на ранніх етапах проєктування. Глибокий аналіз впливу затримок пам'яті та синхронізації на коефіцієнт прискорення.

5. Негативні сторони роботи: недостатня увага аналізу предметної області; недостатньо чітко описано процес складання програмно-технічного засобу

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 достатньому технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: задовільно (D / 70)

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

Янцова О. М., доцент кафедри ТЗ

_____

"___"_____2026 р. _____(підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Мушинський Ярослав Олександрович

ПІБ здобувача вищої освіти

ФІТ, ІІІ курсу, групи КІ2с-23-1

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

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

Назва кваліфікаційної роботи Спеціалізований багатоядерний процесор підвищеної продуктивності
Автор Ярослав МУШИНСЬКИЙ
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: Лигун Олексій Олегович

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 2.26%; 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 Олексій ЛИГУН Ім'я, ПРІЗВИЩЕ