ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**ВИСОКОПРОДУКТИВНИЙ ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЛЯ ОБРОБКИ МАТРИЧНИХ ОБ'ЄКТІВ**»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 – Комп'ютерна інженерія
освітньо-професійної програми Комп'ютерна інженерія

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Бондарчук Анастасія

Виконав: здобувач вищої освіти гр. КСЗ–51
Бондарчук Анастасія

Керівник Світлана Куфтеріна___

Рецензент _____

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти - «Бакалавр»
Спеціальність -123 – Комп'ютерна інженерія
Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
Наталія Лащевська
"____" _____2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бондарчук Анастасія

1.Тема кваліфікаційної роботи: **ВИСОКОПРОДУКТИВНИЙ ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЛЯ ОБРОБКИ МАТРИЧНИХ ОБ'ЄКТІВ**

керівник кваліфікаційної роботи Світлана Куфтеріна , затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи « 11 » травня 2024 р.

3. Вхідні дані до роботи:
- Технічна документація;
- Теоретичні та статичні дані;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
1. Аналіз існуючих процесорів.
2. Розробка архітектури та загальної структури систем.
3. Розробка структур окремих підсистем.
4. Тестування та оцінка.
5. Перелік ілюстративного матеріалу: *презентація.*
Дата видачі завдання « 10 » лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	08.03-18.03.24	Викон.
2	Аналіз завдання	18.03-25.03.24	Викон.
3	Обробка матеріалу	25.03-28.03.24	Викон.
4	Висновки за результатами роботи	28.03-29.03.24	Викон.
5	Розробка архітектури та загальної структури системи	30.03-31.03.24	Викон.
6	Розробка структур окремих підсистем	05.04-20.04.24	Викон.
7	Передзахист	17.05.24	Викон.
8	Здача в деканат	06.06.24	Викон.

Здобувач вищої освіти _____ Бондарчук Анастасія

Керівник
кваліфікаційної роботи _____ Світлана Куфтеріна

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра 60 стор., 3 табл., 11 рис., 30 джерел.

Дипломна робота присвячена розробці програмно-апаратного комплексу, що включає детальне дослідження апаратної частини, програмного забезпечення, а також інтеграцію та оптимізацію системи. У роботі проаналізовано сучасні методи та технології розробки, особливості вибору компонентів та платформ. Основну увагу приділено плануванню архітектури системи, вибору ефективних рішень для оптимізації продуктивності та енергоспоживання. Також розглянуто питання тестування компонентів і системи в цілому, забезпечення надійності та високої продуктивності виробу. Результати дослідження можуть бути використані для побудови надійних і ефективних програмно-апаратних комплексів у різних галузях.

У дипломній роботі використовувалися різні методи дослідження:

1. Аналіз сучасних методів та технологій розробки для визначення найбільш ефективних підходів.
2. Проектування архітектури системи з урахуванням вимог до продуктивності та надійності.
3. Експериментальне тестування компонентів та системи для виявлення можливих проблем.
4. Різні види тестування, такі як юніт-тестування, інтеграційне тестування та системне тестування.
5. Використання інструментів автоматизації для полегшення процесу тестування та забезпечення неперервної доставки програмно-апаратного комплексу.

ABSTRACT

Text part of the master's qualification work:
60 pages, 11 pictures, 3 table, 30 sources.

The diploma thesis is dedicated to the development of a hardware-software complex, which includes detailed research of the hardware components, software, as well as system integration and optimization. The work analyzes modern methods and technologies of development, features of component and platform selection. The main focus is on system architecture planning, selection of efficient solutions for performance and power optimization. Additionally, issues of component and system testing, ensuring reliability and high performance of the product are considered. The research results can be utilized for constructing reliable and efficient hardware-software complexes in various fields.

Various research methods were used in the diploma thesis:

1. Analysis of modern development methods and technologies to determine the most effective approaches.
2. System architecture design considering performance and reliability requirements.
3. Experimental testing of components and the system to identify potential issues.
4. Different types of testing, such as unit testing, integration testing, and system testing.
5. Utilization of automation tools to facilitate the testing process and ensure continuous delivery of the hardware-software complex.

ЗМІСТ

ВСТУП…….

РОЗДІЛ 1. АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПАК……..1

1.1 Багатоядерні процесори Intel……..……1

1.2 Багатоядерні процесори AMD……5

1.3 Порівняння процесорів Intel та AMD ……..……...…9

Висновки до розділу 1.....14

РОЗДІЛ 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПАК……...17

2.1 Паралельні обчислення.....17

2.2 Мови паралельного програмування.....22

2.3 Бібліотеки паралельного програмування.....29

Висновки до розділу 2.....32

РОЗДІЛ 3. РОЗРОБКА ПАК…….....35

3.1 Створення апаратної частини.....35

3.2 Розробка програмного забезпечення.....41

3.3 Тестування програмного забезпечення.....45

Висновки до розділу 3.....49

ВИСНОВКИ.....

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....

ВСТУП

У сучасному світі, де технологічні інновації швидко трансформують промислові та споживчі ринки, програмно-апаратні комплекси відіграють критичну роль у вирішенні складних інженерних та обчислювальних завдань. Ці системи, що включають як апаратне, так і програмне забезпечення, є основою для найрізноманітніших застосувань: від розумних пристроїв інтернету речей до складних систем управління даними великого обсягу. Через їх здатність виконувати великомасштабні обчислення, керувати даними в реальному часі та підтримувати роботу критично важливих систем, роль програмно-апаратних комплексів стає все більш значущою.

Розвиток та впровадження цих комплексів вимагає глибокого розуміння технічних можливостей та потенційних викликів. Важливість цього процесу полягає не лише у забезпеченні високої продуктивності та надійності, але й у оптимізації енергоспоживання та вартості обладнання. Правильно спроектовані та впроваджені програмно-апаратні комплекси можуть значно знизити загальні операційні витрати та підвищити ефективність технологічних процесів.

Аналіз та інтеграція систем

Ключовим аспектом в розробці програмно-апаратних комплексів є інтеграція апаратних та програмних компонентів. Цей процес включає детальний аналіз системних вимог, вибір відповідних технологій та архітектур, а також проектування здатності системи до масштабування та адаптації до змінних умов експлуатації. Інтеграція повинна бути гнучкою, щоб підтримувати оновлення та модифікації, які можуть виникати у відповідь на нові технологічні тенденції або зміни в бізнес-процесах.

Інноваційні стратегії впровадження

Впровадження програмно-апаратних комплексів включає стратегічне планування та використання інноваційних методик, таких як гнучка розробка та ітераційні підходи, які дозволяють вчасно реагувати на виклики та вдосконалювати систему на кожному етапі її життєвого циклу. Використання автоматизованих інструментів тестування та моніторингу стану системи є важливим для забезпечення її стабільності та надійності.

Рекомендації для майбутнього розвитку

На основі проведеного дослідження можна рекомендувати подальше вивчення впливу новітніх технологій, таких як штучний інтелект та машинне навчання, на розробку програмно-апаратних комплексів. Також важливою є розробка нових методів захисту даних та підвищення безпеки цих систем, щоб вони могли ефективно протистояти зростаючим кіберзагрозам.

Завершуючи, можна стверджувати, що успішне впровадження програмно-апаратних комплексів вимагає не тільки технічної компетентності, але й стратегічного планування та готовності до адаптації перед обличчям швидких змін у технологічному світі.

Мета дипломної роботи полягає в ретельному розгляді процесу розробки програмно-апаратного комплексу, з фокусом на ефективній інтеграції апаратного та програмного забезпечення.

Завдання дослідження включають:

1. Аналіз апаратного забезпечення та вибір апаратних компонентів для забезпечення необхідної продуктивності та ефективності;
2. Вивчення мов програмування та інструментальних засобів для розробки програмного забезпечення, що забезпечує повне використання можливостей апаратного забезпечення;
3. Розробка тестових стратегій та проведення ретельного тестування на кожному етапі розробки для ідентифікації та виправлення помилок;

Предметом дослідження є процес розробки програмно-апаратного комплексу, включаючи вибір апаратного забезпечення, розробку програмного забезпечення та тестування. Об'єктом дослідження є ефективність інтеграції апаратного та програмного забезпечення в програмно-апаратному комплексі.

АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ВИСОКОПРОДУКТИВНИХ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

1.1 Багатоядерні процесори Intel

Багатоядерні процесори Intel — це ключовий компонент в архітектурі сучасних обчислювальних систем, які вимагають високої оброблювальної потужності та ефективності. З появою перших багатоядерних процесорів Intel Core Duo у 2006 році, компанія Intel постійно розширює можливості своїх чіпів, впроваджуючи інноваційні технології, такі як Hyper-Threading, Turbo Boost, і, згодом, інтеграцію графічних процесорів у ті ж кремнієві пластини.[1]

Архітектура та розвиток

Розвиток багатоядерних процесорів Intel можна простежити через кілька поколінь, від Sandy Bridge (2-ге покоління Core) до найновіших Ice Lake (10-те покоління Core) та майбутніх Alder Lake (12-те покоління Core) чіпів. Кожне нове покоління вносило значні поліпшення в IPC (інструкції за цикл), енергоефективність та підтримку передових мультимедійних та обчислювальних інструкцій.[2]

Рисунок 1.1 - **Sandy Bridge** Рисунок 1.2 - **Ice Lake**

Рисунок 1.3 - **Alder Lake**

Технологічні нововведення

Однією з ключових особливостей процесорів Intel є технологія Hyper-Threading, яка дозволяє кожному фізичному ядру процесора виконувати два потоки інструкцій одночасно, ефективно подвоюючи кількість потоків обробки. Технологія Turbo Boost дозволяє ядрам процесора динамічно збільшувати частоту в залежності від завантаження та температурного режиму, забезпечуючи оптимальне співвідношення продуктивності та енергоспоживання. [1]

Intel Turbo Boost

Історія та розвиток

Intel Turbo Boost була вперше представлена у 2008 році як частина мікроархітектури Nehalem, яка замінила попередню архітектуру Core. Ця технологія з'явилася на ринку разом з процесорами Intel Core i7, і була спрямована на підвищення обчислювальної продуктивності за допомогою збільшення тактової частоти процесора понад базовий рівень, коли умови теплового дизайну дозволяли. З того часу технологія пройшла кілька ітерацій удосконалення і була включена майже в усі процесори Intel вище середнього та високого рівня, включаючи лінійки Core i5 та Core i3 в подальших поколіннях. [3]

Таблиця 1.1 – порівняння характеристик процесорів **Intel**

	Core i9-14900K	**Core i7-14700K**	**Core i7-14600K**
Cores/threads	24 (8+16)/32	16(8+12)/28	14(6+8)/20
Cache (L2+L3)	68MB	61MB	44MB
Base clock	3.2GHz (P-core), 2.4GHz (E-core)	3.4GHz (P-core), 2.5GHz (E-core)	3.5GHz (P-core), 2.6GHz (E-core)
Boost clock	Up to 6GHz	Up to 5.6GHz	Up to 5.3GHz
TDP	125W/253W	125W/253W	125W/181W
Price	$590	$410	$320

Майбутнє розвитку

Intel продовжує розвивати та вдосконалювати Turbo Boost, включаючи нові версії, такі як Turbo Boost Max Technology 3.0, яка забезпечує ще більш цілеспрямоване збільшення продуктивності за допомогою ідентифікації "найкращих" ядер у процесорі та їх пріоритизації для найважчих задач. Такі інновації допомагають підтримувати конкурентоспроможність Intel на ринку процесорів у боротьбі за надання споживачам ще більшої потужності та ефективності у їхніх обчислювальних рішеннях. [2]

Вплив на продуктивність системи

Багатоядерні процесори Intel значно підвищують продуктивність системи завдяки здатності паралельно обробляти великі обсяги даних. Це особливо важливо в таких застосуваннях, як віртуалізація, великі бази даних та обробка зображень. Висока пропускна спроможність і швидкість передачі даних між ядрами та кеш-пам'яттю також відіграють важливу роль в оптимізації загальної продуктивності системи. [4]

Майбутнє багатоядерних процесорів

Intel продовжує інвестувати в розвиток нових технологій, таких як інтеграція штучного інтелекту та машинного навчання безпосередньо в процесори, що відкриває нові можливості для створення ще більш потужних і енергоефективних систем. Очікується, що наступні покоління процесорів Intel продовжать встановлювати стандарти в області високопродуктивних обчислень. [6]

Висновки

Багатоядерні процесори Intel залишаються на передньому краї інновацій у комп'ютерній техніці. З кожним новим поколінням компанія продовжує вдосконалювати технологію, забезпечуючи більшу продуктивність, кращу

енергоефективність та нові функціональні можливості, що відповідають зростаючим вимогам до сучасних обчислювальних систем.

1.2 Багатоядерні процесори AMD

Багатоядерні процесори AMD відіграють важливу роль у сучасній індустрії високопродуктивних обчислень, пропонуючи значні переваги в термінах ціни, продуктивності та енергоефективності. З часу запуску своєї архітектури Zen, AMD зробила значний вплив на ринок процесорів, запропонувавши потужні рішення для геймерів, ентузіастів, а також професіоналів у сфері обробки даних та обчислень.

Архітектура та розвиток

AMD здійснила значний прорив зі своєю архітектурою Zen, представленою в 2017 році, і продовжує її розвивати з кожним наступним поколінням. Основними характеристиками архітектури Zen є підвищена щільність операцій на такт (IPC), поліпшена підтримка множинності потоків та енергоефективність. AMD Zen використовує масштабовану архітектуру з модульним дизайном, де кожен модуль містить декілька ядер з окремим кешем L3, що оптимізує внутрішню пропускну спроможність і мінімізує затримки. [5]

Рисунок 1.4 - **Ryzen™ Threadripper**

Рисунок 1.5 - **ryzen 9 7950x3d**

Рисунок 1.6 - **AMD Athlon™**

Рисунок 1.7 - **AMD EPYC™**

Рисунок 1.8 - **AMD A-Series**

Розробка та випуск

Розробкою архітектури Zen керував головний архітектор Майк Кларк та команда інженерів AMD. Процес розробки почався в 2012 році, а перший процесор з новою архітектурою був офіційно представлений у 2017 році під брендом Ryzen. Zen відрізняється від попередників, таких як архітектури

Bulldozer, які були критиковані за низьку продуктивність і високе енергоспоживання.[5]

Модульний дизайн:

- Zen використовує модульну структуру, де кожен модуль містить декілька ядер CPU з власними кешами та контролером пам'яті, що дозволяє забезпечити кращу масштабованість і гнучкість при виготовленні процесорів з різною кількістю ядер.

Покращення IPC:

- Завдяки переробленій мікроархітектурі та більш ефективній шині даних, архітектура Zen забезпечує значне збільшення IPC (інструкцій за цикл), що призводить до зростання загальної продуктивності на ядро. [3]

Технології енергозбереження:

- Застосування передових технологій управління живленням та енергоефективних матеріалів допомагає знизити споживання енергії без втрати продуктивності.

Технологічні нововведення

Зі зростанням попиту на більшу потужність обробки даних, AMD впровадила в архітектуру Zen технології, такі як "Infinity Fabric", яка дозволяє швидку взаємодію між ядрами, пам'яттю та іншими компонентами процесора. Ця технологія забезпечує високу пропускну спроможність і масштабованість для систем з великою кількістю процесорів, що особливо важливо для серверних та обчислювальних систем. [6]

Таблиця 1.2 – порівняння характеристик процесорів **AMD**

	Ryzen 9 7950X	**Ryzen 9 7900X**	**Ryzen 7 7700X**	**Ryzen 5 7600X**
Cores/threads	16/32	12/24	8/16	6/12
Cache (L2+L3)	80MB	76MB	40MB	38MB
Base clock	4.5GHz	4.7GHz	4.5GHz	4.7GHz
Boost clock	5.7GHz	5.6GHz	5.4GHz	5.3GHz
TDP	170W	170W	105W	105W
Price	$699	$549	$399	$299

Вплив на продуктивність системи

Процесори AMD забезпечують значну продуктивність за рахунок використання додаткових ядер та оптимізації потоків даних. Архітектура Zen спроможна забезпечувати конкурентоспроможну продуктивність у застосуваннях, які потребують інтенсивних обчислень, таких як рендеринг відео, наукові обчислення та ігрові додатки.

Майбутнє багатоядерних процесорів AMD

AMD продовжує розширювати можливості своєї архітектури Zen з планами на подальше підвищення IPC, зменшення енергоспоживання та поліпшення масштабованості. Компанія активно працює над впровадженням нових технологій, таких як підтримка PCIe 4.0 та DDR5, що відкривають нові горизонти для розробки високопродуктивних програмно-апаратних комплексів.

Висновки

AMD значно зміцнила свої позиції на ринку багатоядерних процесорів завдяки інноваційній архітектурі Zen та стратегії постійного вдосконалення продукції. Завдяки цьому, процесори AMD є відмінним вибором для розробників систем, які шукають надійне та ефективне рішення для своїх потреб в обчисленнях. Це приклад структури для глибокого аналізу багатоядерних процесорів AMD в контексті вашої дипломної роботи. Текст можна далі розширювати, включаючи більше технічних даних, порівнянь, діаграм та прикладів застосувань, які допоможуть детально описати кожен аспект.

1.3 Порівняння процесорів Intel та AMD

З розвитком інформаційних технологій виникає потреба в оцінці потужності та ефективності процесорів, які є серцем будь-якої комп'ютерної системи. Intel і AMD вже багато років є головними конкурентами в області виробництва процесорів. Цей розділ детально розглядає ключові аспекти їхніх продуктів, порівнюючи їх за різними параметрами.

Технологічні інновації

Intel традиційно фокусується на високій продуктивності пер ядро і впровадженні технологій, які забезпечують кращу підтримку корпоративних та наукових обчислень. Технології, такі як Hyper-Threading і Turbo Boost, стали стандартом в процесорах Intel, що забезпечують відмінну багатозадачність та адаптивність до навантажень. AMD, з іншого боку, концентрує свої зусилля на виробництві процесорів з більшою кількістю ядер за доступною ціною, що робить їх популярними серед геймерів і тих, хто займається відео обробкою та іншими видами мультимедійного контенту. Їхні процесори серії Ryzen

демонструють вражаючі результати в многопоточних завданнях завдяки архітектурі Zen, яка ефективно масштабується зі збільшенням кількості ядер.[1]

Таблиця 1.3 – порівняння процесорів **Intel та AMD**

Intel	**AMD**
Turbo Boost Technology: Як уже було зазначено, технологія Intel Turbo Boost дозволяє процесорам автоматично збільшувати частоту роботи ядер в залежності від поточного завантаження та теплових умов. Це забезпечує додаткову продуктивність коли потрібно, не перевищуючи енергетичні обмеження.	**Zen Architecture**: Архітектура Zen значно підвищила IPC (інструкцій на цикл) завдяки своїй ефективній мікроархітектурі, що включає покращену систему передбачення розгалужень та великі кеші. Це дозволило AMD значно підвищити продуктивність своїх процесорів у порівнянні з попередніми поколіннями.
Hyper-Threading: Технологія Hyper-Threading від Intel дозволяє кожному фізичному ядру процесора обробляти два потоки інструкцій одночасно, що підвищує загальну продуктивність системи при виконанні багатопоточних завдань.	**Infinity Fabric**: Інноваційна технологія з'єднання, Infinity Fabric, дозволяє з'єднувати між собою ядра, контролери пам'яті, та інші компоненти в процесорах AMD, забезпечуючи швидкий та ефективний обмін даними. Це критично важливо для масштабування продуктивності у багатоядерних системах.
Optane Memory: Intel Optane Memory використовує технологію пам'яті 3D XPoint, що	**Precision Boost**: Технологія Precision Boost від AMD дозволяє процесорам

надає високу швидкість читання/запису та зниження затримок порівняно з традиційними SSD. Це поліпшує час завантаження системи та додатків, а також швидкість відгуку системи.	динамічно регулювати тактову частоту в залежності від завдань, забезпечуючи оптимальну продуктивність без перевищення енергетичних чи теплових обмежень. Це схоже на Turbo Boost від Intel, але з більш динамічним управлінням частотою.

Продуктивність та енергоефективність

У порівнянні продуктивності обох виробників, важливо зазначити, що процесори Intel зазвичай краще справляються з однопоточними завданнями через вищу частоту і кращу оптимізацію на ядро. Це робить їх ідеальним вибором для застосунків, що вимагають великої потужності на ядро, таких як деякі види наукових моделювань і професійних програм обробки даних. AMD виходить на передні позиції, коли йдеться про многопоточність та енергоефективність відносно кількості ядер, що надає їм переваги в ігрових додатках та програмах, які добре масштабуються з кількістю ядер. [11]

Ціна та доступність

Ціна і доступність процесорів від Intel і AMD відіграють важливу роль у виборі споживачів та ринковій динаміці. Обидві компанії пропонують широкий спектр процесорів, призначених для різних сегментів ринку, від бюджетних варіантів до високопродуктивних рішень для ентузіастів і професіоналів. [11]

Intel

Історично Intel мала домінуюче становище в сегменті високопродуктивних і корпоративних рішень, що часто відбивалося на більш високій ціні їх продуктів порівняно з конкурентами. Intel використовує свою інноваційну технологію, таку як вищезгадана Hyper-Threading і Turbo Boost, для виправдання преміум цінової політики. Вони також інвестують значні кошти в маркетинг та дослідження, що також позначається на вартості їхніх чіпів. Однак, останнім часом Intel стикається з конкуренцією від AMD, яка вплинула на цінову стратегію. В результаті Intel почала пропонувати більш конкурентоспроможні ціни, особливо в середньому сегменті ринку, щоб утримати свою частку ринку.

AMD

AMD традиційно вважається більш доступним варіантом, особливо в контексті співвідношення ціна-продуктивність. Запровадження архітектури Zen дозволило AMD значно покращити продуктивність своїх процесорів, при цьому зберігаючи привабливі ціни. Це допомогло компанії здобути популярність серед геймерів, ентузіастів, а також в корпоративному секторі, де вартість обладнання відіграє важливу роль. Запровадження лінійки Ryzen забезпечило AMD сильні позиції на ринку завдяки високій продуктивності при порівняно низькій ціні. AMD також активно конкурує в сегменті серверних процесорів зі своєю серією EPYC, яка пропонує вражаючі характеристики за значно нижчою ціною порівняно з аналогічними пропозиціями від Intel.

Майбутнє розвиток та інновації

Intel і AMD продовжують інвестувати в нові технології. Intel активно розвиває свої зусилля в області штучного інтелекту та інтегрованих графічних

рішень. AMD продовжує вдосконалювати свою архітектуру Zen, впроваджуючи підтримку новіших технологій пам'яті та швидших інтерфейсів. [1]

Висновки

Після ретельного аналізу процесорів від Intel та AMD, можна зробити висновок, що обидва виробники пропонують високопродуктивні рішення, які відрізняються своїми технологічними інноваціями, ціновою політикою та цільовими сегментами ринку. Intel, із своєю Turbo Boost та Hyper-Threading технологіями, продовжує домінувати у високопродуктивних і корпоративних сегментах, пропонуючи продукти, які забезпечують високу продуктивність за відносно вищою ціною. Ці технології дозволяють процесорам Intel ефективно справлятися з однопоточними завданнями, що є критично важливим для деяких спеціалізованих застосунків.

З іншого боку, AMD зі своєю архітектурою Zen та технологією Infinity Fabric значно підвищила продуктивність своїх процесорів, що дозволило їй відновити позиції на ринку і здобути популярність завдяки вигідному співвідношенню ціни до продуктивності. AMD вдалося досягти значних успіхів у багатопоточній обробці, що робить її процесори особливо привабливими для геймерів та професіоналів, які займаються відеообробкою та іншими ресурсомісткими задачами.

Цінова політика AMD, яка традиційно пропонує більш доступні рішення, допомагає споживачам отримати більшу вартість за свої інвестиції, особливо в умовах сучасної економічної невизначеності. Це робить продукцію AMD особливо привабливою для широкого кола користувачів, які шукають якісне та доступне обладнання.

Обидві компанії продовжують інвестувати у розвиток нових технологій, адаптуючись до змінних потреб ринку та технологічних умов. Їхні продукти

демонструють вражаючі показники продуктивності та інновацій, що стимулює здорову конкуренцію та сприяє подальшому технологічному розвитку. Вибір між Intel та AMD буде залежати від специфічних технічних вимог, бюджету та особистих уподобань споживачів, при цьому обидві марки пропонують достойні варіанти для різноманітних застосувань.

Висновки до розділу 1: Апаратне забезпечення програмно-апаратного комплексу

Загальний огляд розділу

Розділ 1 дипломної роботи детально розглядає апаратне забезпечення, зосереджуючись на багатоядерних процесорах від двох провідних виробників у галузі: Intel та AMD. Аналіз кожної технології, їх розвитку, інновацій та впливу на продуктивність системи дозволяє нам зробити обґрунтовані висновки щодо їх застосування в різних обчислювальних середовищах.

Технологічні інновації та розвиток

Процесори Intel та AMD продемонстрували значний технологічний прогрес за останні декілька десятиліть. Intel відома своїми інноваціями у сфері мікропроцесорних технологій, такими як Hyper-Threading та Turbo Boost, які були спрямовані на підвищення продуктивності ядра та краще управління енергоспоживанням. З іншого боку, AMD зробила акцент на архітектурі Zen, яка виявилася революційною для багатоядерних процесів зі своєю здатністю масштабувати продуктивність і ефективно управляти потоками даних на більшому числі ядер.

Продуктивність і вартість

Обговорення продуктивності та вартості показало, що процесори Intel, як правило, пропонують кращу продуктивність на ядро, що робить їх ідеальними для задач, що потребують великої обчислювальної потужності на один потік. AMD, натомість, надає краще співвідношення ціна-продуктивність з більшою кількістю ядер за нижчою ціною, що робить їх вигідним варіантом для багатозадачних додатків та ігрових систем.

Енергоефективність і багатозадачність

Вивчення енергоефективності та багатозадачності підтвердило, що AMD має переваги у багатопоточних застосуваннях завдяки своїй архітектурі Zen, що оптимально використовує енергію і ресурси процесора. Тим часом, Intel продовжує вдосконалювати свою технологію, зменшуючи енергоспоживання при високій продуктивності, особливо у своїх новітніх моделях.

Майбутнє апаратного забезпечення

Огляд майбутнього розвитку в апаратному забезпеченні показує, що обидва виробники прагнуть до інновацій, які б могли задовольнити зростаючі вимоги до обчислювальних систем. Intel та AMD інвестують у дослідження і розробку нових технологій, таких як штучний інтелект, машинне навчання та квантові обчислення, що будуть формувати майбутнє програмно-апаратних комплексів.

Заключні висновки

Підсумовуючи, обидва виробники пропонують унікальні переваги, які можуть бути використані в залежності від конкретних потреб та вимог

користувачів. Вибір між Intel та AMD повинен базуватися на детальному аналізі поточних та майбутніх обчислювальних завдань, які потрібно вирішити. Оцінка їх продуктів в контексті останніх технологічних досягнень та майбутнього потенціалу дозволить зробити обґрунтований вибір, оптимально використовуючи ресурси і бюджет.

2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ВИСОКОПРОДУКТИВНИХ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

2.1.Паралельні обчислення

Паралельні обчислення є важливою складовою сучасних обчислювальних систем, здатних оптимізувати виконання великих і складних обчислювальних задач. Цей підхід дозволяє обробляти дані за допомогою кількох обчислювальних ядер, які працюють одночасно, тим самим значно підвищуючи ефективність і зменшуючи час обробки.

Принципи та механізми

Паралельні обчислення базуються на концепції розподілу задач між різними процесорними ядрами або комп'ютерами в мережі. Задачі можуть бути розбиті на менші підзадачі, які виконуються одночасно, що є особливо ефективним при великих обсягах даних або складних розрахунках. Системи, що використовують паралельні обчислення, зазвичай включают в себе апаратні та програмні компоненти, здатні забезпечити синхронізацію і ефективний обмін даними між ядрами або вузлами. [7]

Технології та архітектури

Серед основних технологій паралельних обчислень можна виділити:

1.Мультипроцесорні системи (SMP - Symmetric Multiprocessing):

Мультипроцесорні системи, або SMP (Symmetric Multiprocessing), це архітектура комп'ютерів, у яких два або більше ідентичних процесори (ЦПУ) з'єднані в єдину інтегровану систему, спільно використовуючи одну оперативну пам'ять (ОЗП) та інші ресурси. У SMP всі процесори працюють незалежно, але обмінюються однією операційною системою та пам'яттю, що дозволяє забезпечити високий рівень обчислювальної потужності та ефективності управління задачами. [9]

Принцип роботи

У SMP системі кожен процесор може одночасно виконувати різні завдання, або може працювати над одним завданням, розділяючи його на підзадачі. Це розподілення завдань сприяє збільшенню продуктивності та ефективності обчислень. Операційна система, яка керує SMP системою, забезпечує розподіл навантаження між процесорами, а також управління пам'яттю та вводом-виводом.

Технічні проблеми SMP

Однією з ключових проблем у SMP є проблема когерентності кешу, коли кілька копій одних і тих же даних зберігаються в кешах різних процесорів. Вирішення цієї проблеми потребує спеціальних алгоритмів когерентності кешу, таких як MESI (Modified, Exclusive, Shared, Invalid), які допомагають уникнути неузгодженості даних у кешах.

Переваги SMP

1. **Підвищення продуктивності:**

- Завдяки паралельній обробці даних, SMP може значно збільшити загальну продуктивність системи, зокрема в багатозадачному середовищі;

2. **Збільшення надійності:**

- Якщо один із процесорів вийде з ладу, інші можуть продовжувати роботу, забезпечуючи високий рівень надійності та доступності системи;

3. **Економія коштів:**

- Замість використання одного дорогого багатоядерного процесора, можна використовувати декілька більш дешевих процесорів, досягаючи аналогічної загальної продуктивності;

2. Багатоядерні процесори:

Багатоядерні процесори — це центральні обробні одиниці (ЦПУ), які містять два або більше незалежних ядра, кожне з яких може обробляти інструкції. Це ядра знаходяться на одному інтегральному чипі або в одному корпусі процесора. Ідея полягає в тому, щоб збільшити обчислювальну потужність, не підвищуючи тактову частоту, що дозволяє виконувати більше операцій за одиницю часу без значного збільшення споживання енергії або тепловиділення.

Технічні аспекти

1. **Паралельна обробка даних:**

- Багатоядерні процесори забезпечують можливість паралельної обробки даних, що дозволяє виконувати декілька завдань одночасно. Це покращує багатозадачність і здатність системи швидко реагувати на запити користувача;

2. **Ефективність використання енергії:**

- Замість збільшення частоти процесора, що вимагає більше енергії та створює більше тепла, багатоядерні процесори розподіляють навантаження між кількома ядрами, що може зменшити загальне енергоспоживання при виконанні певних задач;

3. **Зменшення затримок:**

- У багатопоточних або високонавантажених середовищах якісне управління потоками даних між ядрами може зменшити затримки, відкриваючи шлях для більш швидкого відгуку системи;

Проблеми

1. **Потреба в оптимізації програмного забезпечення:**

Програмне забезпечення повинно бути спеціально розроблене або оптимізоване для використання переваг багатоядерної обробки. Додатки, що не підтримують багатопоточність, можуть не отримати повної вигоди від багатоядерних процесорів;

2. **Спільний доступ до ресурсів:**

Ядра часто ділять деякі системні ресурси, такі як кеш-пам'ять або шини даних. Конкуренція за ці спільні ресурси може призвести до конфліктів і зниження продуктивності;

3. **Масивно паралельні обчислювальні системи (MPP - Massively Parallel Processing):**

Масивно паралельні обчислювальні системи (MPP) є типом комп'ютерних систем, які використовують велику кількість процесорів (або окремих комп'ютерів), працюючих паралельно для виконання обчислювальних завдань. Ці системи здатні виконувати розподілену обробку даних на великому масштабі, що робить їх ідеальними для складних наукових розрахунків, великих баз даних та інших завдань, які потребують інтенсивних обчислень.

Технічні аспекти

1. **Розподілення ресурсів:**

- У MPP системах кожен процесор або вузол має свою власну локальну пам'ять та операційну систему. Процесори спілкуються один з одним через

високошвидкісні мережі, що дозволяє обмінюватися даними для синхронізації процесів обробки;

2. **Масштабованість:**

- MPP системи вирізняються високою масштабованістю, дозволяючи додавати більше процесорів або вузлів залежно від потреб у обчислювальній потужності. Ця гнучкість робить їх ідеальними для додатків, які вимагають розширення обчислювальних ресурсів;

3. **Паралельна обробка даних:**

- MPP системи ефективно виконують паралельну обробку даних, розбиваючи великі обчислювальні завдання на менші підзадачі, які можуть виконуватися одночасно на різних процесорах. Це знижує час виконання завдань та підвищує загальну продуктивність системи;

Проблеми

1. **Складність програмування:**

- Розробка програмного забезпечення для MPP систем може бути складною через потребу в ефективному розподілі завдань і управлінні паралельністю. Програмісти повинні враховувати синхронізацію, мертві блокування та конкуренцію за ресурси;

2. **Вартість та обслуговування:**

- MPP системи є дорогими у встановленні та обслуговуванні через складність апаратних та програмних компонентів, а також потребу в спеціалізованих знаннях для ефективної експлуатації;

Програмування та інструменти

Для реалізації паралельних обчислень використовуються спеціалізовані мови програмування та бібліотеки, такі як:

MPI (Message Passing Interface): Стандарт для обміну повідомленнями між процесами, який широко використовується в обчислювальних кластерах і MPP системах.

OpenMP (Open Multi-Processing): API, яке дозволяє розробникам легко писати програми для мультипроцесорних систем, використовуючи директиви препроцесора для створення паралельних блоків коду. [12]

Застосування

Паралельні обчислення знаходять застосування в різних областях, включаючи наукові дослідження, інженерію, фінанси, медицину та розваги. Вони використовуються для моделювання складних систем, обробки великих масивів даних, розрахунку фінансових ризиків, аналізу медичних зображень і розробки відеоігор.

Висновки

Паралельні обчислення забезпечують значний приріст продуктивності в багатьох сферах завдяки своїй здатності оптимізувати використання обчислювальних ресурсів. Інновації в технологіях і методах програмування продовжують відкривати нові можливості для розвитку і вдосконалення паралельних систем, що робить їх незамінними для сучасних обчислювальних вимог.

2.2 Мови паралельного програмування

Мови паралельного програмування дозволяють розробникам ефективно використовувати можливості сучасних багатоядерних і многопроцесорних систем, оптимізуючи виконання коду через одночасне виконання завдань. Ці мови і інструменти допомагають управляти складними задачами обробки даних, які вимагають значних обчислювальних ресурсів.

Огляд мов паралельного програмування

Мови паралельного програмування можна розділити на кілька основних категорій:

1. **Низькорівневі мови (C, C++):**

Мова C

Розроблена у 1972 році Деннісом Рітчі в AT&T Bell Labs, мова C була створена для реімплементації операційної системи UNIX. Вона стала однією з найважливіших мов програмування завдяки своїй гнучкості та ефективності. C дозволяє програмістам контролювати майже кожен аспект комп'ютерної системи, що робить її ідеальною для системного програмування, вбудованого програмування та розробки додатків, де потрібна висока продуктивність; [1]

Мова C++

C++ була розроблена Б'ярном Страуструпом у ранніх 1980-х як розширення мови C, що включає об'єктно-орієнтовані можливості. Це дозволяє C++ забезпечити більш високий рівень абстракції, покращуючи управління великими програмними проектами. C++ зберігає здатність до низькорівневого програмування, що дозволяє використовувати її в таких галузях, як розробка системного програмного забезпечення, ігри, реалізація операційних систем, і багато інших, де потрібна велика продуктивність та ефективне управління ресурсами; [11]

Переваги та недоліки

Переваги:

- Прямий контроль над апаратними ресурсами: Забезпечує максимальну продуктивність та ефективність;
- Гнучкість: Можливість оптимізації коду для специфічних архітектур;
- Портабельність: Програми на C та C++ можуть бути скомпільовані на багатьох платформах;

Недоліки:

- Складність: Вищий поріг входження у порівнянні з вищерівневими мовами;
- Безпека: Невірне управління пам'яттю може призвести до витоку пам'яті або інших помилок;
- Відсутність вбудованої підтримки потоків на рівні мови: (до C++11), що ускладнює розробку многопоточних додатків;
- **C/C++ з бібліотеками MPI та OpenMP:** C і C++ традиційно використовуються для паралельного програмування завдяки своїй високій продуктивності та гнучкості. MPI (Message Passing Interface) є стандартом для програмування паралельних додатків в кластерних обчислювальних середовищах, а OpenMP використовується для створення многопоточних додатків на основі спільної пам'яті;

2. **Високорівневі мови (Python, Julia):**

Python

Python — це універсальна, інтерпретована мова програмування, створена Гвідо ван Россумом у 1991 році. Її філософія наголошує на читабельності коду та ефективності розробки, з використанням значної кількості вбудованих високорівневих структур даних. Особливості Python включають:

- **Динамічна типізація:** Python використовує динамічну типізацію, що дозволяє розробникам писати менше специфічного коду і забезпечує більшу гнучкість у розробці;
- **Велика стандартна бібліотека:** Python має обширну стандартну бібліотеку, яка включає інструменти для різноманітних завдань від веб-розробки до наукових розрахунків;
- **Спільнота та підтримка:** Велика глобальна спільнота і наявність численних бібліотек та фреймворків сприяють швидкій розробці проектів у різних областях; [11]

Julia

Julia — це високорівнева мова програмування, розроблена для високопродуктивних наукових і технічних розрахунків. Її розробка почалася у 2009 році і була офіційно випущена у 2012 році. Julia має наступні характеристики:

- **Швидкість виконання:** Julia спроектована так, щоб бути швидкою, нерідко наближаючись до швидкості виконання коду, написаного на C. Використання LLVM (Low Level Virtual Machine) компілятора забезпечує оптимізації на льоту для високоефективних обчислень;
- **Легкість використання:** Julia об'єднує легкість використання і читабельність Python із потужністю мов, призначених для системного програмування;
- **Обробка великих даних:** Julia ефективно обробляє великі обсяги даних, інтеграцію з іншими мовами та засобами, такими як Python, R, SQL, що робить її ідеальною для наукових досліджень, що вимагають обробки великих наборів даних;

- **Python з бібліотеками, такими як NumPy, SciPy, та multiprocessing:** Python став популярним вибором для наукових досліджень та аналізу даних завдяки простоті написання коду і наявності потужних бібліотек для паралельної обробки;
- **Julia:** Julia розроблена спеціально для наукових обчислень з вбудованою підтримкою паралельної обробки, забезпечуючи високу швидкість виконання, яка може конкурувати з такими мовами, як C та Fortran;

3. **Функціональні мови (Haskell, Erlang):**

Haskell

Haskell — це строго типізована, чисто функціональна мова програмування, яка була створена в 1990 році. Вона використовує ліниві обчислення, де вирази обчислюються лише тоді, коли їх значення дійсно потрібне. Це дозволяє ефективно працювати з дуже великими або навіть нескінченними структурами даних;

Особливості Haskell:

- **Чистота:** Всі функції в Haskell є чистими, тобто їх результати залежать лише від їхніх вхідних даних і не мають побічних ефектів, які могли б змінити стан програми;
- **Типізація:** Haskell має потужну систему типів, яка допомагає уникнути багатьох поширених помилок ще під час компіляції;
- **Модульність:** Код легко модульовано, що сприяє підтримці та рефакторингу; [12]

Erlang

Erlang був створений у 1980-х роках у компанії Ericsson як мова для розробки масштабованих, високонадійних телекомунікаційних систем. Він

став широко відомим за свою здатність ефективно управляти паралельними процесами і виняткову підтримку розподілених систем.

Особливості Erlang:

- **Паралельність:** Erlang забезпечує вбудовану підтримку легковагових процесів, що дозволяє створювати велику кількість паралельних процесів;
- **Відмовостійкість:** Системи, написані на Erlang, можуть автоматично відновлюватися після помилок, що робить мову ідеальною для місійно-критичних застосунків;
- **Гаряче оновлення коду**: Erlang дозволяє оновлювати код вже працюючих систем без зупинки чи перезавантаження;

Переваги функціонального програмування

1. **Надійність:** Функціональні мови сприяють написанню коду, який легше розуміти, тестувати і верифікувати, зменшуючи кількість помилок;
2. **Легкість тестування:** Чисті функції спрощують написання тестів, оскільки кожна функція може бути перевірена окремо від інших частин програми;
3. **Конкурентність:** Функціональні мови відмінно підходять для паралельної обробки, що стає все більш актуальним у світі мультиядерних процесорів і розподілених систем;

- **Haskell та Erlang:** Функціональні мови надають унікальні можливості для створення надійних і масштабованих паралельних програм. Erlang використовується для розробки систем з високою доступністю і може легко управляти тисячами паралельних процесів. [7]

Технологічні виклики і стратегії

Паралельне програмування має свої технологічні виклики, зокрема, управління пам'яттю, синхронізація потоків та визначення ефективних алгоритмів розподілу завдань. Розробники повинні вибирати мову і інструменти, які дозволяють максимально оптимізувати код для конкретних обчислювальних завдань, враховуючи специфіку використовуваної апаратної платформи.

Застосування та приклади

Мови паралельного програмування знаходять застосування в різних галузях:

- **Наукові дослідження:** Використання паралельних обчислень для моделювання клімату, квантових обчислень та біоінформатики;
- **Фінанси:** Розрахунки ризиків та великі дані для аналізу ринку;
- **Розробка програмного забезпечення:** Ігри, реалізація серверів великої потужності для обслуговування веб-запитів та інше;

Висновки

Розуміння та правильне використання мов паралельного програмування відіграють ключову роль в ефективному використанні сучасних обчислювальних ресурсів. Вибір відповідної мови та технології дозволяє оптимізувати продуктивність програм і систем, забезпечуючи високу ефективність та продуктивність вирішення складних обчислювальних задач.

2.3 Бібліотеки паралельного програмування

Бібліотеки паралельного програмування надають розробникам інструменти для ефективного використання апаратних можливостей, зокрема багатопроцесорності та багатоядерності сучасних комп'ютерів. Ці бібліотеки забезпечують абстракцію низькорівневих деталей управління потоками, синхронізацією і комунікацією, дозволяючи розробникам концентруватися на алгоритмічній частині задачі.

Ключові бібліотеки

1. **MPI (Message Passing Interface)** [10]

- MPI є де-факто стандартом для написання програм, які виконуються на кластерах і грідах. Вона дозволяє процесам обмінюватися повідомленнями в різноманітних топологіях обчислювальних мереж, підтримуючи шкалимість на рівні тисячі вузлів;
- **Застосування:**

MPI використовується в академічних дослідженнях, інженерії, та моделюванні великих систем, де потрібна висока продуктивність обробки даних;

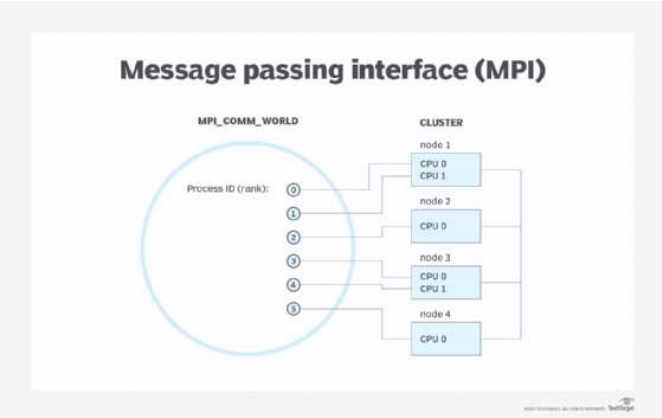

Рисунок 2.1 - **Message Passing Interface**

2. **OpenMP (Open Multi-Processing)** [8]

- OpenMP — це API, яке підтримує багатопоточне програмування на мовах C, C++ та Fortran, дозволяючи розробникам створювати додатки, які ефективно використовують багатоядерні процесори;

- **Застосування:**

OpenMP використовується в програмах, що вимагають значної обчислювальної потужності, таких як обробка зображень, наукові обчислення та великі обчислювальні додатки;

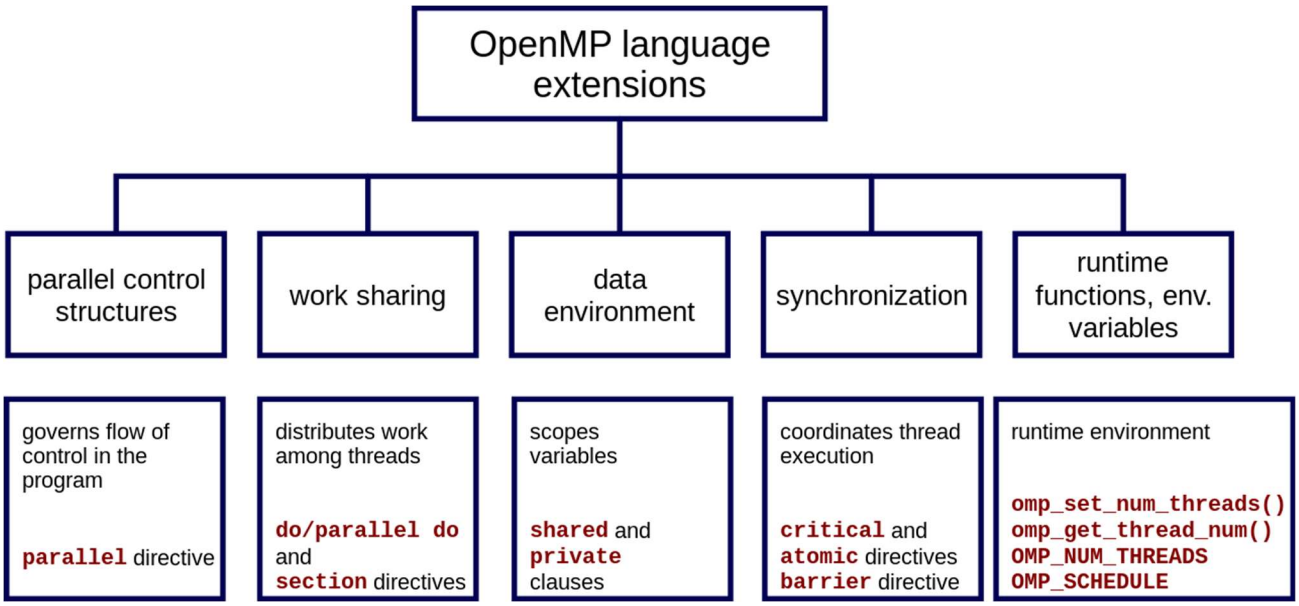

Рисунок 2.2 - **OpenMP**

3. **Intel Threading Building Blocks (TBB)**

- Intel TBB — це потужна C++ бібліотека для створення розширюваних багатопоточних додатків. Вона має вдосконалені алгоритми для динамічного розподілу завдань та обробки великих даних;

- **Застосування:**

TBB використовується в промислових додатках, що потребують оптимальної продуктивності та ефективного використання ресурсів, включаючи графічні редактори, відео обробку та системи реального часу;

4. CUDA (Compute Unified Device Architecture)

- NVIDIA CUDA дозволяє розробникам використовувати могутність GPU для паралельних обчислень. Це надає драматичне прискорення за рахунок виконання тисячі потоків одночасно.
- Застосування: CUDA широко застосовується в областях, що потребують величезної кількості обчислень, таких як глибоке навчання, наукові обчислення, біоінформатика та фінансовий аналіз;

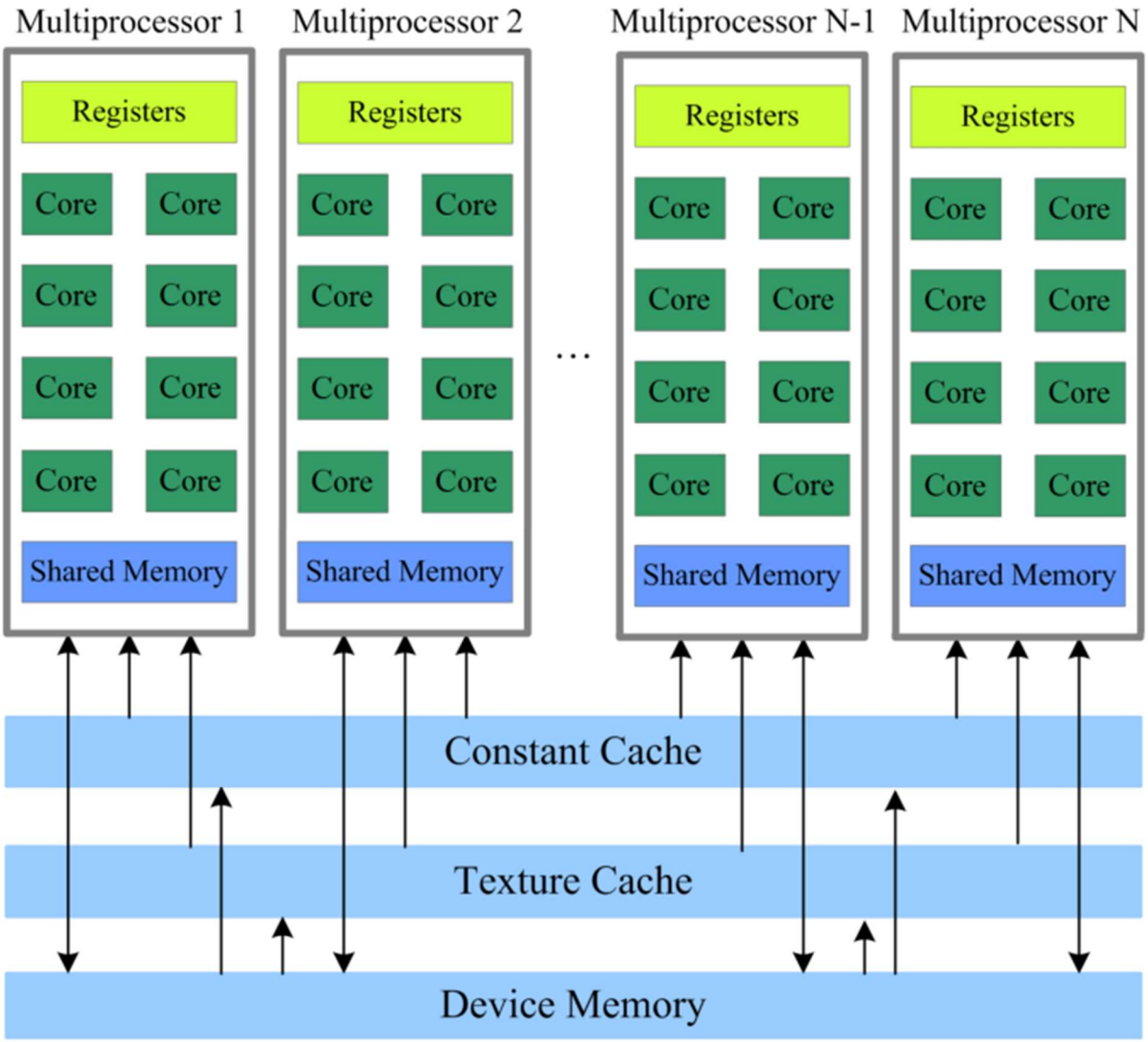

Рисунок 2.3 - **CUDA**

Проблеми і виклики

Паралельне програмування може бути складним через проблеми, пов'язані з мертвими блокуваннями, гонками за даними, та управлінням пам'яттю.

Бібліотеки, такі як MPI та OpenMP, надають інструменти для вирішення цих проблем, але вимагають глибоких знань та розуміння особливостей паралельного виконання.

Майбутнє розвитку

Інновації в бібліотеках паралельного програмування продовжують відкривати нові можливості для оптимізації програмного забезпечення. Розвиток технологій, таких як машинне навчання та штучний інтелект, збільшує потребу в більш потужних і ефективних бібліотеках для паралельної обробки.

Висновки

Бібліотеки паралельного програмування відіграють ключову роль у створенні ефективних, швидких і надійних програмних рішень, які можуть використовувати повний потенціал сучасної апаратної архітектури. Вони дозволяють розробникам зосередитися на задачах вищого рівня, мінімізуючи складність управління ресурсами та паралельним виконанням.

Висновки до розділу 2: Програмне забезпечення програмно-апаратного комплексу

Загальний огляд розділу

Розділ 2 дипломної роботи зосереджений на програмному забезпеченні, яке є вирішальним для ефективного використання апаратних ресурсів у

програмно-апаратних комплексах. Ми детально розглянули ключові аспекти паралельних обчислень, мови паралельного програмування, а також бібліотеки, які підтримують ці технології. Аналіз цих елементів дозволив нам оцінити потенціал та виклики, що стоять перед розробниками сучасних програмних систем.

Паралельні обчислення та їх значення

Паралельні обчислення відіграють критичну роль у підвищенні продуктивності обчислювальних систем. Вони дозволяють значно скоротити час обробки великих обсягів даних та забезпечують можливість вирішувати складні задачі, які були б недосяжні для послідовних систем. Технології, такі як MPI та OpenMP, дозволяють ефективно масштабувати додатки на кластерах та багатопроцесорних системах, відкриваючи нові можливості для наукових досліджень та промислових застосувань.

Мови програмування для паралельних обчислень

Різноманітність мов програмування, які підтримують паралельні обчислення, свідчить про важливість гнучкості в підходах до розробки програмного забезпечення. Мови високого рівня, такі як Python та Julia, забезпечують простоту розробки та знижують поріг входження для науковців і інженерів, які не є професіоналами в програмуванні. З іншого боку, потужні низькорівневі мови, такі як C та C++, продовжують бути важливими в сферах, де необхідна максимальна продуктивність і контроль над ресурсами системи.

Бібліотеки паралельного програмування

Бібліотеки паралельного програмування, такі як Intel TBB та CUDA, відіграють ключову роль у спрощенні розробки високопродуктивних додатків,

що використовують багатопроцесорні та багатоядерні системи. Вони не тільки надають розробникам потужні інструменти для оптимізації програмного коду, але й забезпечують сумісність з сучасними архітектурами обчислювальних систем.

Майбутнє програмного забезпечення для паралельних систем

Розвиток технологій паралельних обчислень та програмування продовжуватиме впливати на дизайн майбутніх обчислювальних систем. Подальші дослідження та інновації в архітектурі обчислювальних систем та методах програмування будуть спрямовані на підвищення ефективності, зниження споживання енергії та підтримку більш складних додатків в реальному часі.

Заключні висновки

Вивчення програмного забезпечення для паралельних обчислень показує, що сучасні технології забезпечують значні переваги для різноманітних застосувань, від баз даних і веб-серверів до наукових симуляцій та штучного інтелекту. Постійний розвиток інструментів та мов, що підтримують паралельні обчислення, відіграють ключову роль у цьому процесі, дозволяючи більш широкому колу розробників використовувати переваги сучасної обчислювальної техніки.

3 РОЗРОБКА ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

3.1 Створення апаратної частини

Створення апаратної частини програмно-апаратного комплексу вимагає вдумливого підходу до вибору компонентів, дизайну системи та її інтеграції. Цей процес охоплює не тільки вибір і збірку апаратних компонентів, але й оптимізацію архітектури системи для досягнення необхідної продуктивності та надійності.

Планування апаратної частини

1. **Аналіз вимог:**

Аналіз вимог є фундаментальним кроком у процесі планування апаратної частини, оскільки він визначає основні параметри і обмеження, з якими буде працювати команда розробників; [13]

Важливо чітко визначити наступні аспекти:

- **Функціональні вимоги:** Що повинен робити апаратний комплекс? Включає деталі про обробку даних, інтерфейси користувача, необхідні взаємодії з іншими системами, і специфікації вводу/виводу;
- **Нефункціональні вимоги:** Ці вимоги включають параметри продуктивності, надійності, енергоефективності, фізичні розміри, вартість компонентів та системи загалом, а також вимоги до довговічності і можливості модернізації;
- **Вимоги до середовища експлуатації:** Врахування умов, у яких апаратний комплекс буде використовуватися, наприклад, температурний діапазон, вологість, потенційні механічні впливи;

2. **Вибір архітектури:** [13]

Вибір архітектури визначає загальну структуру апаратного комплексу. На цьому етапі розглядаються такі аспекти:

- **Тип процесора:** Вибір між мікроконтролерами, загальнопризначеними процесорами, DSP (цифрові сигнальні процесори), або FPGA (польові програмовані вентильні матриці) залежно від потреб у обробці даних та інших функціональних вимог;
- **Топологія системи:** Рішення про використання одноплатних систем, модульних систем або високоінтегрованих платформ;
- **Стратегія масштабування:** Планування можливостей для майбутнього розширення системи, включно з додаванням процесорів, пам'яті або спеціалізованих модулів;

3. **Селекція компонентів:**

Селекція компонентів передбачає вибір конкретних електронних компонентів, які використовуються в апаратній частині:

- **Процесори, пам'ять та сховище:** Залежно від обраної архітектури і вимог до процесорної потужності та обсягу пам'яті;
- **Периферійні пристрої:** Включення компонентів для управління мережею, графікою, звуком та іншими зовнішніми пристроями;
- **Інтерфейси:** Визначення необхідних інтерфейсів для комунікації з іншими системами, такими як USB, Ethernet, HDMI, або бездротові технології;

4. **Розробка прототипу:** [14]

Розробка прототипу є наступним кроком, де створюється робочий зразок апаратної частини для перевірки концепції, виявлення проблем та подальшої ітерації:

- **Створення прототипу:** Використання плат розробки, спеціалізованих інструментів для проектування друкованих плат, та виробництво початкових зразків;
- **Тестування функціональності:** Перевірка працездатності прототипу, включаючи тестування процесора, пам'яті, периферійних пристроїв та інтерфейсів;
- **Оптимізація та налагодження:** Виправлення виявлених проблем, зміни в дизайні для покращення продуктивності або зниження вартості виробництва;

Оптимізація та інтеграція системи [15]

Оптимізація системи є ключовим етапом у розробці апаратної частини, де здійснюються покращення щодо продуктивності, ефективності та надійності. Важливо зосередитись на наступних аспектах:

1. **Зниження споживання енергії:**

- Використання енергоефективних компонентів та алгоритмів управління живленням може значно знизити енергоспоживання, що є критично важливим для портативних і вбудованих систем;

2. **Підвищення продуктивності:**

- Оптимізація процесора, пам'яті, та системи кешування для забезпечення максимальної швидкості обробки даних;

3. **Мінімізація затримок:**

- Поліпшення внутрішньої архітектури системи для зменшення затримок при обробці даних, що є особливо важливим для реального часу та критичних застосунків;

Інтеграція системи забезпечує злиття всіх апаратних та програмних компонентів в єдину функціональну систему. Це включає:

Документація

1. Технічна документація:

- Схеми - схематичні діаграми (показують з'єднання між різними електронними компонентами, такими як мікросхеми, резистори, конденсатори), монтажні схеми (використовуються для планування фізичного розташування компонентів на платах друкованих монтажів (ПДМ)), блокові схеми (використовуються для представлення вищого рівня архітектури системи. Вони показують основні компоненти системи (наприклад, процесори, пам'ять, вхід/вихід) і зв'язки між ними.) Схеми сигнальних шляхів(Ці схеми фокусуються на показі шляхів передачі сигналів між компонентами у системі, включаючи мікропроцесори, контролери, пам'ять і периферійні пристрої.) Функціональні схеми (Вони описують, як різні частини системи співпрацюють для виконання конкретних завдань.) ;
- Специфікація компонентів - специфікація компонентів не тільки впливає на продуктивність і функціональність кінцевої системи, але також на її вартість, розмір, енергоспоживання та здатність до масштабування. Акуратне визначення і специфікація кожного компонента дозволяють інженерам оптимізувати загальну конструкцію, мінімізувати ризики і забезпечити високу надійність та ефективність системи на всіх етапах її життєвого циклу;
- Інструкції зі збірки та налаштування - вони надають докладні кроки та процедури, які необхідні для правильного зібрання, налаштування та першої експлуатації апаратних систем. Ось основні аспекти, які покривають ці інструкції: перелік компонентів, вимоги до інструментів, покрокова процедура, перевірка правильності збірки, початкове налаштування,

інсталяція операційної системи, конфігурація мережі, тестування та діагностика;

- Процедури тестування - є критичним компонентом процесу розробки в комп'ютерній інженерії, який забезпечує відповідність продукції технічним специфікаціям та вимогам безпеки. Це включає ряд процедур, кожна з яких має свою мету та методологію. Ось декілька основних типів тестування, що застосовуються у комп'ютерній інженерії: функціональне тестування, тестування продуктивності, тестування надійності, тестування зручності використання, регресивне тестування. Ефективні процедури тестування дозволяють не тільки гарантувати якість продукту перед його випуском на ринок, але й ідентифікувати потенційні проблеми на ранніх стадіях розробки, що може значно знизити витрати на подальше виправлення помилок та підтримку. Таким чином, добре плановане тестування є ключем до успішного проектування, розробки та експлуатації комп'ютерних систем і технологій;

2. Керівництво для користувачів:

- Надає інструкції та поради кінцевим користувачам про те, як використовувати та обслуговувати апаратну частину;

3. Документація API:

- Якщо апаратна частина взаємодіє з програмним забезпеченням через API, необхідно надати документацію, що описує всі доступні функції та їх використання;

Підтримка

1. Технічна підтримка:

- Забезпечення підтримки користувачам на етапах впровадження, експлуатації та обслуговування апаратного комплексу;

2. Оновлення:

- Регулярне оновлення програмного забезпечення апаратної частини для виправлення помилок, поліпшення функціональності та адаптації до нових вимог та технологій;

Висновки

У процесі розробки апаратної частини програмно-апаратного комплексу для обробки матричних об'єктів було здійснено ряд ключових дій, кожна з яких мала на меті забезпечити високу продуктивність, надійність і ефективність системи. Основні етапи включали аналіз вимог, вибір архітектури, селекцію компонентів, розробку прототипу, оптимізацію та інтеграцію системи.

1. Аналіз вимог дозволив чітко визначити функціональні і нефункціональні параметри системи, включаючи специфікації вводу/виводу, продуктивність, енергоефективність і вимоги до довговічності. Цей етап поклав основу для всіх подальших рішень у процесі розробки;
2. Вибір архітектури визначив структурну організацію апаратної частини, включаючи тип процесора, топологію системи та стратегію масштабування, забезпечуючи гнучкість та можливість майбутньої модернізації;
3. Селекція компонентів забезпечила вибір відповідних процесорів, пам'яті, сховища і периферійних пристроїв, що відповідають встановленим вимогам. Кожен компонент був вибраний на основі його продуктивності, вартості та сумісності з іншими елементами системи;
4. Розробка прототипу імплементувала теоретичні розробки у фізичний зразок, який був використаний для тестування та валідації концепції. Цей етап дозволив ідентифікувати та усунути потенційні проблеми на ранніх стадіях;

5. Оптимізація та інтеграція системи завершили процес створення апаратної частини, фокусуючись на підвищенні ефективності, зниженні енергоспоживання і мінімізації затримок. Кінцевий продукт демонструє інтеграцію всіх компонентів у єдину функціональну систему, готову до реального застосування;

Таким чином, апаратна частина програмно-апаратного комплексу була розроблена з урахуванням сучасних технологічних стандартів та вимог до продуктивності і надійності, що робить її здатною ефективно вирішувати поставлені задачі обробки матричних об'єктів.

3.2 Розробка програмного забезпечення

Розробка програмного забезпечення для програмно-апаратних комплексів є критичним аспектом, що впливає на функціональність, продуктивність та здатність системи задовольняти користувацькі вимоги. Цей процес охоплює вибір відповідних технологій, дизайн архітектури програми, кодування, тестування та підтримку. [16]

Планування та вибір технологій

1. Аналіз вимог:

- Збір та аналіз вимог користувачів та стейкхолдерів для забезпечення відповідності програмного забезпечення очікуванням та потребам кінцевих користувачів;

- Визначення функціональних, надійності, зручності та інших вимог до програмного забезпечення;

2. Вибір мови програмування та інструментальних засобів:

- Вибір мов програмування, таких як C/C++, Java, Python, залежно від потреб проекту, зокрема враховуючи їх здатність до інтеграції з апаратними ресурсами;
- Використання інструментів розробки, систем управління версіями та інших інструментальних засобів для підтримки колаборативної роботи і контролю якості;

Архітектура та дизайн

1. Модульна архітектура:

- Розробка модульної архітектури для полегшення тестування, розширення та супроводу програмного забезпечення;
- Використання шаблонів проектування для забезпечення повторного використання коду та ефективної інтеграції компонентів;

2. Інтерфейси та інтеграція:

- Забезпечення здатності програмного забезпечення до інтеграції з іншими системами та компонентами через чітко визначені API;
- Розробка графічного користувацького інтерфейсу (GUI), якщо це потрібно, для забезпечення зручності користувачів;

Розробка та тестування

1. Кодування:

- Виконання програмування відповідно до затверджених специфікацій і стандартів розробки.
- Застосування технік рефакторингу для підтримки чистоти та зрозумілості коду;

2. Тестування:

- Використання автоматизованого та ручного тестування для забезпечення високої якості програмного продукту;
- Проведення юніт-тестування, інтеграційного тестування, системного тестування та приймального тестування для забезпечення відповідності всім технічним та функціональним вимогам;

Документація та підтримка

1. Документація:

- Розробка технічної та користувацької документації для забезпечення легкості супроводу та використання програмного забезпечення;
- Оновлення документації відповідно до змін у програмному продукті;

2. Підтримка та обслуговування:

- Забезпечення технічної підтримки користувачів та вирішення проблем, що можуть виникнути під час експлуатації програмного забезпечення;
- Планування та виконання оновлень програмного забезпечення для поліпшення функціональності та безпеки;

Висновки

Розробка програмного забезпечення для програмно-апаратного комплексу є фундаментальним аспектом, що забезпечує виконання вимог щодо функціональності, продуктивності та зручності користування системи. Цей процес включає кілька критичних етапів: аналіз вимог, вибір технологій, проектування архітектури, кодування, тестування, а також підтримку та документування.

1. Аналіз вимог став основою для всього процесу розробки, встановлюючи чіткі критерії для функціональних та нефункціональних характеристик програмного продукту. Цей етап

забезпечив глибоке розуміння потреб користувачів та технічних вимог до системи;

2. Вибір технологій включав визначення мов програмування та інструментальних засобів, що відповідають сучасним стандартам розробки та забезпечують ефективну реалізацію поставлених задач. Використання потужних і гнучких мов програмування та систем управління версіями дозволило підвищити продуктивність розробки та спростити подальшу підтримку продукту;
3. Архітектура та дизайн програми були розроблені з акцентом на модульність та інтегрованість, що сприяє легкій адаптації до мінливих вимог та легкості в тестуванні та супроводженні системи. Використання шаблонів проектування та чітко визначених API сприяло створенню надійної та масштабованої системи;
4. Розробка та тестування програмного забезпечення відбувалися в кілька етапів, включаючи ретельне кодування та різнорівневе тестування від юніт-тестів до системних тестів. Ці дії забезпечили високу якість програмного продукту та його відповідність усім поставленим специфікаціям;
5. Документація та підтримка включали створення вичерпних керівництв для користувачів та розробників, що забезпечують легке впровадження та ефективне використання системи. Надання постійної технічної підтримки та оновлень програмного забезпечення гарантує довготривалу стабільність і безпеку продукту;

Завершуючи, розробка програмного забезпечення була проведена з високим ступенем професіоналізму та уваги до деталей, що забезпечило створення ефективної та надійної системи, спроможної задовольнити вимоги сучасного ринку і впоратися з складними обчислювальними завданнями.

3.3 Тестування програмного забезпечення

Тестування програмного забезпечення є вирішальним етапом у розробці програмно-апаратних комплексів, воно гарантує, що продукт відповідає заданим вимогам та є вільним від помилок, що можуть негативно вплинути на його функціональність або безпеку. Цей процес включає кілька видів тестування, кожен з яких спрямований на ідентифікацію та усунення потенційних проблем у програмі. [17]

Планування тестування

1. **Аналіз вимог:**

- Ключове завдання полягає в тому, щоб зібрати та аналізувати всі технічні та функціональні вимоги до програмного забезпечення. Це включає вивчення специфікацій продукту, взаємодію зі стейкхолдерами для збору їхніх вимог та використання анкет, опитувань чи інтерв'ю;
- Важливо визначити, які критичні функції продукту потребують особливої уваги під час тестування, і розробити чіткі тестові випадки для їх перевірки;

2. **Розробка тестової стратегії:**

- Встановлення цілей тестування, вибір методологій та інструментів. Важливо визначити, які ресурси будуть потрібні для тестування, включаючи час;
- Планування процедур для ручного та автоматизованого тестування, визначення основних етапів тестування та критеріїв для переходу між ними;

Види тестування

1. **Юніт-тестування:**

- Перевірка окремих модулів програмного забезпечення на відповідність їхнім внутрішнім специфікаціям та забезпечення їх правильної роботи незалежно від інших частин системи;
- Юніт-тестування часто автоматизується і виконується розробниками з використанням інструментів, як-от JUnit або pytest, що дозволяє швидко ідентифікувати та вирішувати проблеми на ранніх стадіях розробки

2. **Інтеграційне тестування:**

- Перевірка взаємодій між різними модулями програми або між програмним забезпеченням та апаратною платформою;
- Цілісність та правильність обміну даними між компонентами або системами;

3. **Системне тестування:**

- Перевірка комплексної системи на відповідність всім інтегрованим функціональним і нефункціональним вимогам. Це включає тестування продуктивності, безпеки та зручності користувача;
- Виконання стрес-тестів, тестів навантаження та інших видів тестування, щоб забезпечити, що система може працювати в умовах реального використання;

4. **Приймальне тестування:**

- Залучення кінцевих користувачів для оцінки системи в реальних умовах, забезпечуючи, що вона відповідає їхнім потребам і очікуванням;
- Це останній етап тестування перед реальним впровадженням системи, важливий для забезпечення задоволеності користувачів;

Автоматизація тестування

1. **Інструменти та інфраструктура:**

- Використання інструментів для автоматизації тестування, таких як Selenium, Jenkins, або Travis CI, для забезпечення постійної інтеграції та неперервної доставки;
- Застосування скриптів для автоматизації рутинних тестових задач, зменшення часу тестування та підвищення точності результатів;

Документація та аналіз результатів

1. **Звітність:**

- Ретельне документування процесу тестування і результатів для забезпечення можливості аналізу ефективності тестів, виявлення помилок, та їх впливу на загальну функціональність системи;
- Аналіз виявлених помилок, їх приоритетів і впливу на загальну функціональність програмного продукту;
- Забезпечення повної прозорості процесу для стейкхолдерів і команди розробників;

2. **Управління якістю:**

- Застосування методів контролю якості для постійного покращення процесів тестування та розробки;
- Внесення змін і вдосконалень на основі зворотного зв'язку від тестувальників, розробників і кінцевих користувачів;

Заключні висновки

Тестування програмного забезпечення виявилося вирішальним етапом у розробці програмно-апаратних комплексів, гарантуючи, що кінцевий продукт відповідає усім заданим технічним і функціональним вимогам, а також є вільним від помилок, які могли б негативно вплинути на його функціональність чи безпеку. Процес тестування був ретельно спланований і включав аналіз вимог, розробку тестових стратегій, а також застосування

різних видів тестувань, які охоплювали юніт-тестування, інтеграційне, системне і приймальне тестування. [18]

Кожен етап тестування зіграв ключову роль у забезпеченні цілісності та надійності системи:

- Юніт-тестування забезпечило перевірку окремих модулів, виявлення та усунення помилок на ранніх стадіях.
- Інтеграційне тестування допомогло підтвердити коректність взаємодії між модулями.
- Системне тестування оцінило загальну функціональність та продуктивність системи, включаючи безпеку та зручність користувача.
- Приймальне тестування включило кінцевих користувачів для забезпечення відповідності системи реальним умовам експлуатації.

Автоматизація тестування, застосування сучасних інструментів та технологій значно підвищили ефективність тестувальних процедур, зменшивши час виконання тестів і підвищивши точність результатів. Документація та аналіз результатів тестування забезпечили важливу зворотну інформацію для подальшого вдосконалення проекту, а також забезпечили засоби для постійного управління якістю і контролю за відповідністю продукту встановленим вимогам.

У підсумку, систематичний і всебічний підхід до тестування програмного забезпечення значно зміцнив надійність, безпеку та ефективність розробленої програмно-апаратної системи, забезпечуючи її готовність до впровадження та успішного використання в практичних умовах.

Висновки до розділу 3: Розробка програмно-апаратного комплексу

Загальний огляд розділу

Розділ 3 дипломної роботи присвячений всебічному аналізу процесу розробки програмно-апаратного комплексу, включаючи створення апаратної частини, розробку програмного забезпечення та їх тестування. Цей розділ наголошує на важливості інтеграції між апаратними та програмними компонентами для досягнення оптимальної продуктивності та функціональності в кінцевому продукті.

Апаратна частина Розробка апаратної частини вимагає ретельного вибору компонентів та ефективної архітектури системи для задоволення специфічних потреб проекту. Визначення правильного балансу між потужністю, вартістю та енергоефективністю є ключовим для створення ефективної та конкурентоспроможної системи. Також, інтеграція і взаємодія між апаратними компонентами вимагають детального планування та тестування для забезпечення стабільності та високої продуктивності.

Розробка програмного забезпечення

Розробка програмного забезпечення є важливим етапом, який визначає функціональність та інтерфейс користувача системи. Використання сучасних мов програмування та розробка з врахуванням модульності та масштабованості дозволяє створити гнучке та адаптивне рішення. Розробники повинні також зосередитись на безпеці, стабільності та інтеграції програмного забезпечення з апаратною частиною.

Тестування програмно-апаратного комплексу

Тестування є критичним етапом, який забезпечує відповідність кінцевого продукту первісно заданим вимогам та очікуванням. Воно включає юніт-тестування, інтеграційне тестування, системне тестування та приймальне тестування, кожне з яких спрямоване на ідентифікацію та виправлення специфічних проблем. Автоматизація тестування забезпечує повторюваність та ефективність процесу, знижуючи ризики помилок, які можуть виявитись вже після впровадження системи в експлуатацію.

Заключні висновки

Розробка програмно-апаратного комплексу є складним, але вкрай важливим процесом, що вимагає взаємодії між різними дисциплінами інженерії. Ключ до успіху полягає в точному плануванні, використанні перевірених технологій та ретельному тестуванні кожного аспекту системи перед її впровадженням. Створення ефективної програмно-апаратної інтеграції може значно підвищити продуктивність та надійність комплексу, забезпечуючи високу якість та задоволення потреб користувачів у вимогливих обчислювальних завданнях.

ВИСНОВКИ

У своїй дипломній роботі я зосередилась на розробці програмно-апаратного комплексу для обробки матричних об'єктів, орієнтованого на високу продуктивність та надійність. Робота охоплювала різні етапи від проектування апаратної частини до розробки програмного забезпечення та його тестування.

Основні результати та узагальнення моїх науково-дослідних зусиль можна поділити на кілька ключових аспектів:

1. **Апаратна частина:**
   - Я провела детальний аналіз технічних вимог, вибір оптимальних компонентів та проектування системи з урахуванням не тільки поточних потреб, але й можливостей для майбутнього розширення. Важливість цього етапу не може бути недооцінена, адже від правильного вибору апаратної бази залежить загальна продуктивність та ефективність системи.
   - Розробка прототипу дозволила мені перевірити всі теоретичні розрахунки на практиці, виявити можливі проблемні зони та оптимізувати процес перед фінальним виробництвом;
2. **Програмне забезпечення:**
   - Розробка програмного забезпечення була зосереджена на створенні стабільної та надійної системи, здатної ефективно взаємодіяти з апаратною частиною. Я вибрав мови програмування та інструменти, які найкраще відповідали потребам проекту, і впровадив сучасні підходи до проектування архітектури;
   - Інтеграція та тестування програмного забезпечення дозволили забезпечити його високу функціональність та відповідність поставленим вимогам. Зокрема, я зосередився на автоматизації тестування, що значно підвищило його ефективність та знизило можливість помилок;
3. **Практичне застосування та рекомендації:**
   - Результати дослідження показали, що розроблена система може бути широко застосована у різних галузях, де висока продуктивність і точність обробки даних є критично важливими; Я рекомендую подальше впровадження розроблених рішень в

промисловість та науку, де вони можуть значно підвищити ефективність обчислювальних процесів;

- Також я вважаю за необхідне продовжити дослідження у напрямку підвищення енергоефективності та зменшення вартості виробництва апаратних компонентів, що дозволить зробити технологію доступнішою та конкурентоспроможнішою;

Завершуючи, я підкреслюю, що успішне виконання цього проекту відкриває нові можливості для розробки та оптимізації програмно-апаратних систем, що можуть відіграти важливу роль у технологічному прогресі в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Horowitz, P., & Hill, W. (2015). The Art of Electronics. Cambridge University Press.
2. Singh, A. (2016). Embedded Systems: Architecture, Programming and Design. McGraw-Hill Education.
3. Liu, D. (2018). Computer Architecture: A Quantitative Approach. Elsevier.
4. Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design MIPS Edition: The Hardware/Software Interface. Morgan Kaufmann.
5. Zhou, P. (2019). Microprocessor Theory and Applications with 8051. Wiley.
6. Tanenbaum, A. S., & Austin, T. (2020). Structured Computer Organization. Pearson.
7. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
8. McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.
9. Fowler, M. (2018). Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional.
10. Sommerville, I. (2015). Software Engineering. Pearson Education.
11. Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing. Wiley.
12. Osherove, R. (2013). The Art of Unit Testing: with Examples in C#. Manning Publications.
13. Kaner, C., Falk, J., & Nguyen, H. Q. (1999). Testing Computer Software. Wiley.

14. Fewster, M., & Graham, D. (1999). Software Test Automation. Addison-Wesley Professional.
15. Dustin, E., Rashka, J., & Paul, J. (1999). Automated Software Testing: Introduction, Management, and Performance. Addison-Wesley Professional.
16. Spillner, A., Linz, T., & Schaefer, H. (2014). Software Testing Foundations. Rocky Nook.
17. Crispin, L., & Gregory, J. (2009). Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley Professional.
18. Kolawa, A., & Huizinga, D. (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley.
19. Ulrich, K. T., & Eppinger, S. D. (2015). Product Design and Development. McGraw-Hill Education.
20. Brooks, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley Longman Publishing Co.
21. Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
22. Hunt, A., & Thomas, D. (1999). The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional.
23. Beck, K. (2000). Extreme Programming Explained: Embrace Change. Addison-Wesley Professional.
24. North, D. (2012). Implementing Domain-Driven Design. Addison-Wesley Professional.
25. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Longman.
26. Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change (2nd Edition). Addison-Wesley.
27. Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.

28. Coplien, J. O., & Bjørnvig, G. (2010). Lean Architecture: for Agile Software Development. Wiley.
29. Ambler, S. W., & Lines, M. (2012). Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press.
30. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd Edition). Addison-Wesley.