Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка методів і програмних засобів підтримки вбудованих процесорних систем

Виконав: студент 2-го курсу
групи 1ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Овод Дмитро Васильович

Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
«12» грудня 2023 р.

Опонент: к.т.н., доц. каф. ЗІ Лукічов В.В.
«12» грудня 2023 р.

Допущено до захисту

Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«12» грудня 2023 р.

Вінниця ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« _19_ » вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Оводу Дмитру Васильовичу

1. Тема роботи – розробка методів і програмних засобів підтримки вбудованих процесорних систем.

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2023 р.

3. Вихідні дані до роботи: Для 8-бітних мікроконтролерів: серія PIC та AVR, тактова частота від 1 до 20 МГц, розрядність 8 біт, пам'ять RAM від кількох десятків байт до кількох кілобайт, Flash-пам'ять від кількох кілобайт до 256 КБ. Для 16-бітових мікроконтролерів: серія MSP430, тактова частота до 25 МГц, розрядність 16 біт, пам'ять RAM від кількох сотень байт до кількох кілобайт Flash-пам'ять до 512 КБ. Для 32-бітових мікроконтролери: серія ARM Cortex-M, тактова частота від 20 МГц до кількох сотень МГц, розрядність 32 біти, пам'ять RAM від кількох кілобайт до кількох мегабайт, Flash-пам'ять від 256 КБ до кількох мегабайт.

4. Зміст розрахунково-пояснювальної записки: вступ, методи аналізу роботи вбудованих процесорних систем за допомогою прикладних програмних засобів, програмні та апаратні методи дослідження роботи вбудованих процесорних систем, задачі дослідження, програмні та апаратні особливості використання методів вбудованих процесорних систем, порівняльна характеристика використання програмних та апаратних методів вбудованих

процесорних систем, запропонований метод підвищення ефективності роботи вбудованих процесорних систем, розробка вимог до основних параметрів роботи вбудованих процесорних систем, налаштування основних та додаткових параметрів роботи вбудованих процесорних систем, економічна частина, висновки.

5. Перелік графічного матеріалу: вступ, актуальність теми, вибір методів аналізу, методи аналізу роботи вбудованих процесорних систем за допомогою прикладних програмних засобів, задачі дослідження, запропонований метод підвищення ефективності роботи вбудованих процесорних систем, розробка вимог до основних параметрів роботи вбудованих процесорних систем, налаштування основних та додаткових параметрів роботи вбудованих процесорних систем, економічний розділ, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О. М., к.т.н., доцент кафедри ПЗ	19.09.23	01.12.23
5	Причепа І.В., к.е.н., доц. кафедри ЕПВМ	22.10.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Прим.
1	Аналіз методів роботи вбудованих систем, визначення програмних та апаратних методів дослідження.	19.09.2023-02.10.2023	Вик.
2	Дослідження програмних і апаратних особливостей використання вбудованих систем.	03.10.2023-16.10.2023	Вик.
3	Розробка вимог щодо основних параметрів роботи вбудованих систем, налаштування параметрів.	17.10.2023-28.10.2023	Вик.
4	Створення програмного засобу для аналізу роботи вбудованих процесорних систем.	29.10.2023-12.11.2023	Вик.
5	Економічна частина	13.11.2023-01.12.2023	Вик.

Студент _____ **Овод Д. В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Хошаба О. М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 37.091.33

Овод Д. В. Розробка методів і програмних засобів підтримки вбудованих процесорних систем: магістерська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 1?? с.

На укр. мові. Бібліогр. : 23 назв ; рис. : 15; табл. 2.

В магістерській роботі було розроблено методи та програмні засоби для підвищення ефективності вбудованих процесорних систем. Задачі дослідження включали аналіз методів роботи вбудованих систем, визначення програмних та апаратних методів дослідження, а також дослідження програмних і апаратних особливостей їх використання. Робота також передбачала розробку вимог щодо основних параметрів роботи вбудованих систем, налаштування параметрів, та створення програмного засобу для аналізу роботи вбудованих процесорних систем. В результаті виконаної роботи був запропонований метод та розроблені програмні засоби, що реалізують даний метод.

Запропонований метод вирішення задачі полягав у перетворені загальної структури програмно-апаратного засобу до ярусно-паралельної форми.

Ключові слова: вбудовані процесорні системи, програмні засоби, апаратні засоби, графові моделі перетворення.

ABSTRACT

Ovod D.V. Development of methods and software tools for supporting embedded processor systems : master's qualification thesis on the specialty 121 Software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 1?? with.

In Ukrainian language. Bibliographer : 23 titles; Fig. : 15; table 2.

In the master's thesis, methods and software tools were developed to improve the efficiency of embedded processor systems. The tasks of the research included the analysis of the methods of operation of embedded systems, the definition of software and hardware research methods, as well as the study of software and hardware features of their use. The work also included the development of requirements for the main operating parameters of embedded systems, parameter settings, and creating a software tool for analyzing the operation of embedded processor systems. As a result of the work performed, a method was proposed, and software tools were developed to implement this method.

The proposed method of solving the problem consisted of transforming the general structure of the hardware and software into a tiered-parallel form.

Keywords: embedded processor systems, software, hardware, graph models of transformation.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасне велике зростання складності задач вимагає від споживачів та промислової індустрії постійно змінювати та висувати нові вимоги до вбудованих систем. Серед таких змін зазвичай є підтримка складніших алгоритмів обробки сигналів, штучного інтелекту, машинного навчання. Використання таких галузей знань вимагають більшої продуктивності та оптимізованого використання ресурсів від вбудованих систем.

При цьому, зменшення розмірів та енергоспоживання у вбудованих системах породжують багато проблем. Тому, саме розробка ефективних методів, що забезпечують високу продуктивність при обмежених ресурсах, є вкрай важливою задачею.

Також, нові задачі постають під час постійного подальшого розвитку апаратного забезпечення, де на ринок праці приходять нові архітектури процесорів та апаратних прискорювачів, що вимагає розробки програмних методів, які ефективно використовують ці нові можливості.

Подальша інтеграція з хмарними технологіями дозволяє більш ефективно використовувати різні методи для інтеграції вбудованих систем з хмарними сервісами та іншими великими системами.

Залишається також велике коло питань, яке існувало з моменту зародження вбудованих систем, а саме проблеми до роботи таких програмно-апаратних засобів у режимі реального часу. При цьому, у багатьох вбудованих системах ще постають питання щодо критично важливо виконання операцій у визначених таймінгах забезпечення технологічних процесів. Тому, розробка програмних методів для гарантії реального часу та управління завданнями є дуже актуальною задачою.

Таким чином, розвиток вбудованих процесорних систем виконується у напрямку високої продуктивності, енергоефективності, безпеки та інтеграції з виробництвом. Тому, актуальним завданням є розробка новаторських методів та програмно-апаратних рішень, що має велику наукову та практичну важливість.

Мета та завдання дослідження:

Метою роботи є підвищення ефективності використання вбудованих процесорних систем.

У відповідності до поставленої мети потрібно виконати такі **завдання**:

- виконати аналіз методів роботи вбудованих процесорних систем;

- визначити програмні та апаратні методи дослідження роботи вбудованих процесорних систем;

- виконати дослідження програмних особливостей у використанні методів вбудованих процесорних систем;

- виконати дослідження апаратних особливостей у використанні методів вбудованих процесорних систем;

- розробити вимоги щодо основних параметрів роботи вбудованих процесорних систем;

- дослідити налаштування основних параметрів роботи вбудованих процесорних систем;

- дослідити налаштування додаткових параметрів роботи вбудованих процесорних систем;

- розробити програмний засіб з аналізу роботи вбудованих процесорних систем на основі програмних та апаратних методів дослідження.

Об’єктом дослідження є процеси роботи вбудованих процесорних систем.

Предметом дослідження є методи та засоби підвищення ефективності використання вбудованих процесорних систем.

Методи дослідження. У процесі дослідження використовувались: теорія дерева рішень, системний аналіз процесу обробки даних, теорія оптимізації систем, методи прийняття рішень, графові моделі перетворень.

Наукова новизна одержаних результатів:

- вперше запропоновано метод підвищення ефективності роботи вбудованих процесорних систем, особливість якого полягає у виконанні перетворення із загальної структури програмно-апаратного засобу на основі

графової моделі до ярусно-паралельній форми, що виконується на основі закону Амдала і, як наслідок, дає можливість підвищити ефективність обробки даних до 34%;

- подальшого розвитку отримав метод перетворення оптимізованого графа в бінарне дерево, у якому, на відміну від існуючих, вдається більш ефективно до 10% виконувати перетворення за рахунок повторних рекурсивних дій кожної з двох сусідніх груп до піддерева графової форми.

Практична цінність отриманих результатів. Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритм підвищення ефективності роботи вбудованих процесорних систем та розроблено програмні засоби.а

Особистий внесок здобувача. У магістерській кваліфікаційній роботі усі результати дослідження здобуті автором даної роботи самостійно. У роботі [1], опублікованій самостійно, автору належить формування постановки проблеми, мети роботи та основної частини.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів "Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023", (Одеса, 2023).

Публікації. Основні результати досліджень опубліковано в науковій праці у матеріалах конференції.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ РОБОТИ ВБУДОВАНИХ ПРОЦЕСОРНИХ СИСТЕМ

1.1. Методи аналізу роботи вбудованих процесорних систем за допомогою прикладних програмних засобів

Вбудовані процесорні системи (англ. Embedded Processor Systems) - це комп'ютерні системи, що використовуються для керування і контролю різних електронних пристроїв і систем [1-3]. Вони вбудовані безпосередньо в пристрої або обладнання, з яким вони взаємодіють, і зазвичай працюють в режимі реального часу.

Вбудовані процесорні системи можуть бути знайдені в широкому спектрі пристроїв, включаючи мобільні телефони, автомобілі, побутову техніку, медичні прилади, промислове обладнання, роботів, літаки та багато іншого (рис. 1.1). Вони зазвичай виконують специфічні функції або задачі, які вимагають високої швидкості обробки даних, ефективності енергоспоживання і можливості роботи в режимі реального часу.

Особливістю вбудованих процесорних систем є їх спеціалізований характер їх використання. Вони зазвичай мають обмежені обсяги пам'яті та ресурсів, враховуючи вимоги конкретного застосування. Часто вони працюють на мікроконтролерах або спеціально розроблених процесорах, які забезпечують оптимальну продуктивність для конкретних завдань.

Вбудовані процесорні системи розробляються з врахуванням вимог надійності, стійкості, ефективного використання ресурсів і довговічності [2,4]. Їх розробка вимагає спеціальних навичок і знань, оскільки вони пов'язані з розробкою апаратного забезпечення, вбудованим програмуванням та оптимізацією продуктивності.

Вбудовані процесорні системи можуть бути використані в різних галузях і пристроях. Наведемо деякі поширені з них (рис. 1.1).

Смартфони, планшети, носимі пристрої, такі як годинники або фітнес-трекери, використовують вбудовані процесорні системи для керування операційною системою, виконання програм та обробки даних [3-6].

Рисунок 1.1 - Поширені галузі використання вбудованих процесорних систем

Вбудовані процесорні системи використовуються в автомобілях для керування двигунами, системами безпеки, системами розваг та навігації, системами керування кліматом і багато іншого. Вони допомагають забезпечити безпеку, ефективність та комфорт водіїв і пасажирів.

Вбудовані процесорні системи використовуються в пристроях побутової техніки, таких як холодильники, пральні машини, пилососи, мікрохвильові печі тощо. Вони забезпечують керування, контроль та взаємодію з користувачем.

Вбудовані процесорні системи використовуються в медичних пристроях, таких як ЕКГ-апарати, рентгенівські апарати, імплантовані медичні пристрої,

інфузійні системи тощо. Вони забезпечують контроль, діагностику та лікування пацієнтів [6,7].

Вбудовані процесорні системи використовуються в промисловому обладнанні для керування, моніторингу та автоматизації процесів. Вони використовуються в робототехніці, автоматичних системах контролю, системах безпеки, пристроях управління енергозбереженням та багатьох інших промислових застосуваннях.

Вбудовані процесорні системи використовуються в мережевому обладнанні, маршрутизаторах, комутаторах, телефонних системах та інших пристроях зв'язку для забезпечення передачі даних, маршрутизації сигналів і забезпечення зв'язку.

Це наведені лише кілька прикладів галузей, де вбудовані процесорні системи широко використовуються. У залежності від потреб і вимог конкретного застосування, їх можна знайти в різних індустріях та пристроях.

Для аналізу роботи вбудованих процесорних систем існує декілька методів. Наведемо найпоширеніши з них (рис. 1.2).

Логування як метод включає збір і реєстрацію даних про роботу системи під час її виконання. Лог-файли можуть містити інформацію про події, стани системи, помилки, споживання ресурсів та інше [5-8]. За допомогою спеціальних програмних засобів, таких як логгери або системи керування журналами, можна аналізувати логи та вивчати роботу системи для виявлення проблем або оптимізації.

Дебагери є потужними інструментами для аналізу вбудованих систем. Вони дозволяють розробникам вставляти точки зупинки (breakpoints) у програмний код і стежити за його виконанням крок за кроком. За допомогою дебагера можна перевіряти значення змінних, виконувати пошук помилок, спостерігати послідовність виконання коду і визначати причини некоректної роботи системи.

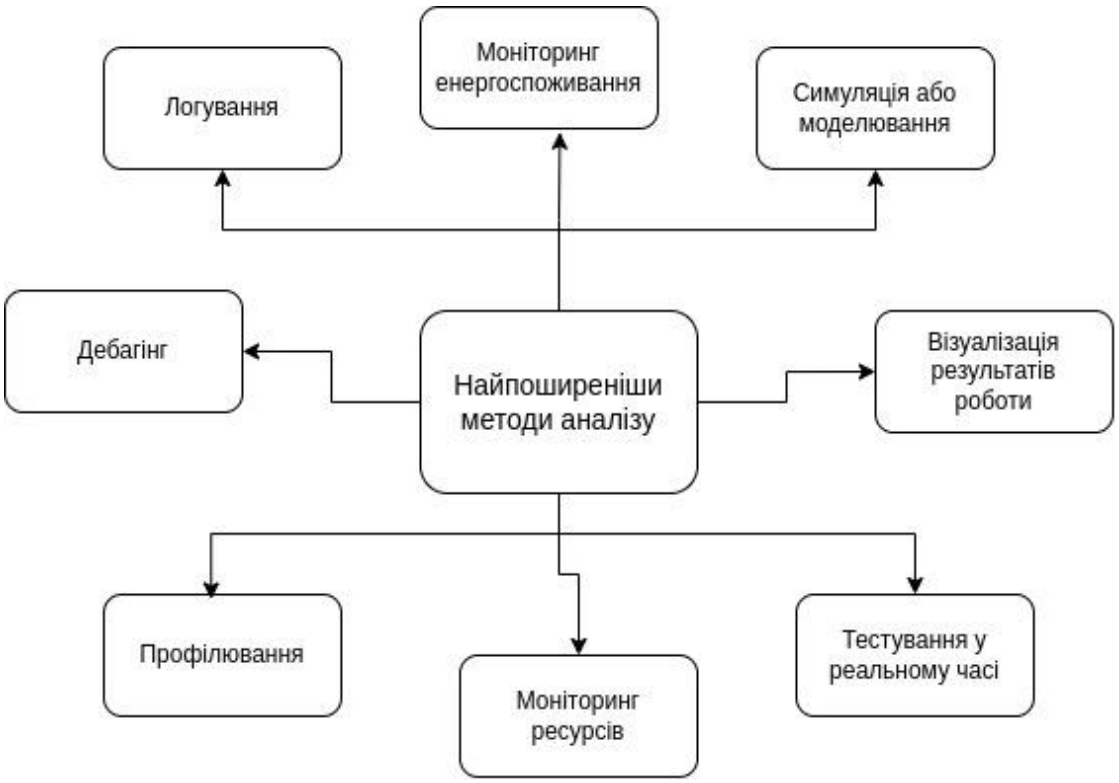


Рисунок 1.2 - Найпоширеніши методи аналізу вбудованих процесорних систем

Профілювання дозволяє збирати детальну статистику про роботу вбудованої системи, включаючи час виконання різних функцій, використання пам'яті, споживання енергії та інше. Це допомагає ідентифікувати гарячі точки, тобто частини коду, які займають багато ресурсів або сповільнюють роботу системи [9-11]. За допомогою профілювання можна виявити проблеми ефективності і оптимізувати роботу системи.

Деякі прикладні програмні засоби дозволяють моніторити використання ресурсів вбудованими процесорними системами. Це включає спостереження за використанням пам'яті, процесорного часу, енергії, мережевих ресурсів тощо. За допомогою такого моніторингу можна аналізувати споживання ресурсів, виявляти проблеми надмірного використання і оптимізувати роботу системи.

Тестування у реальному часі як метод використовує спеціальні програмні засоби для створення тестових сценаріїв, що моделюють різні ситуації реального часу. Це дозволяє перевірити реакцію вбудованої системи на реальних або симульованих подіях, ідентифікувати можливі проблеми з відповідністю тимчасовим вимогам і забезпечити коректну роботу в режимі реального часу.

Візуалізація результатів роботи як метод використовує програмні засоби для візуального представлення роботи вбудованої системи. Візуалізація може включати графіки, діаграми, анімацію та інші засоби для відображення стану системи, потоків даних, взаємодії компонентів тощо [9-11]. Це допомагає аналізувати роботу системи, виявляти залежності та взаємодії між компонентами та визначати проблеми.

Симуляція або моделювання як метод використовує програмні засоби для створення віртуальних моделей вбудованої системи. Симуляція дозволяє відтворити роботу системи у контрольованому середовищі і проводити різні експерименти. Вона може бути корисною для аналізу роботи системи в умовах, які важко або небезпечно відтворити в реальному світі, або для вивчення поведінки системи при зміні параметрів.

Деякі прикладні програмні засоби спеціалізуються на аналізі споживання енергії вбудованими системами. Вони дозволяють вимірювати та моніторити споживання енергії різних компонентів системи, виявляти енергоємні операції або неправильні використання ресурсів, ідентифікувати шляхи оптимізації та енергоефективності.

Таким чином, в даному підрозділі було розглянуті лише декілька поширених методів аналізу роботи вбудованих процесорних систем. В залежності від конкретного завдання і характеристик системи можуть використовуватися інші програмні засоби або комбінації різних методів.

1.2. Програмні методи дослідження роботи вбудованих процесорних систем

Методи дослідження за допомогою програмних засобів включають в себе використання прикладних програмних інструментів, які працюють на вбудованій системі або на зовнішньому комп'ютері, що взаємодіє з нею. Ці засоби дозволяють збирати, аналізувати та відображати дані про роботу системи, проводити тестування, дебагінг, профілювання та інші операції аналізу (табл. 2.1).

Таблиця 1.1

Програмні методи дослідження роботи вбудованих процесорних систем

Метод дослідження	Опис
Моніторинг	Відстеження та запис різних параметрів роботи системи, таких як використання CPU, використання пам'яті, мережевий трафік тощо.
Дебагінг	Виявлення та усунення помилок у програмному коді, включаючи аналіз стек-трейсів, логів, точок зупинки тощо.
Профілювання	Збір та аналіз інформації щодо часу виконання різних функцій та інструкцій у програмі для оптимізації продуктивності.
Тестування	Виконання різних тестів, які допомагають перевірити коректність та надійність роботи системи, включаючи функціональне тестування та модульне тестування.
Аналіз ресурсів	Моніторинг та аналіз використання ресурсів системи, таких як CPU, пам'ять, диски, для оптимізації їх використання.
Імітація	Створення імітованих середовищ для тестування та аналізу системи без впливу на реальну систему.
Аналіз даних	Обробка та аналіз великих обсягів даних, що збираються від системи для виявлення закономірностей та тенденцій.

В загальному випадку методи дослідження, що засновані на роботі на вбудованій системі або на зовнішньому комп'ютері, що взаємодіє з нею, мають свої особливості. Вбудовані системи зазвичай мають обмежені обчислювальні, пам'ятеві та енергетичні ресурси. При роботі з такими системами необхідно враховувати обмеженість ресурсів і мінімізувати накладні витрати програмних засобів аналізу, щоб не перевантажувати систему та не спотворювати результати.

При використанні програмних засобів на вбудованій системі або взаємодії з нею зовнішнім комп'ютером, необхідно мати засіб комунікації із системою. Це може включати засоби зв'язку, такі як послідовний порт, USB, Ethernet,

бездротові протоколи тощо. Крім того, необхідно мати налагоджену зв'язок між програмними засобами та вбудованою системою для передачі команд, даних і отримання результатів аналізу.

Багато вбудованих систем працюють у режимі реального часу, де важлива точність таймінгів і реакція на події в обмежені строки. Під час дослідження роботи таких систем необхідно враховувати вплив програмних засобів аналізу на реальний час. Важливо, щоб аналітичні процеси не порушували режим реального часу системи і не призводили до нестабільності або пропуску подій.

Певні аналітичні задачі можуть вимагати доступу до внутрішніх ресурсів вбудованої системи, таких як регістри, пам'ять, реєстри периферійних пристроїв тощо [10-13]. У таких випадках можуть використовуватися спеціальні інтерфейси або програмні засоби, які дозволяють здійснювати читання і запис до цих ресурсів з метою аналізу.

Ці особливості вимагають уважного планування та налаштування прикладних програмних засобів для аналізу роботи вбудованих процесорних систем. Враховуючи ці фактори, можна ефективно досліджувати та аналізувати роботу вбудованих систем і отримувати змістовні результати.

Вбудовані системи, які працюють у режимі реального часу, мають кілька особливостей, які варто враховувати при їх розробці та аналізі. У режимі реального часу важлива точність виконання операцій та відповідь системи на події в обмежені строки [12,13]. Вбудована система повинна здатися заздалегідь визначеним таймінгам, щоб запобігти виникненню помилок або проблем зі збіжністю.

Вбудовані системи реалізують різні функціональності з різними пріоритетами. Деякі операції можуть бути критичними і потребувати негайної відповіді, тоді як інші операції можуть бути менш важливими. Розробка вбудованих систем реального часу вимагає налагодження пріоритетів для ефективного розподілу ресурсів та виконання важливих операцій вчасно.

Вбудовані системи реального часу мають реагувати на події в режимі реального часу. Це можуть бути зовнішні події, такі як сигнали від датчиків або

команди від користувача, або внутрішні події, які відбуваються в системі. Система повинна бути організована таким чином, щоб забезпечити надійну та швидку обробку подій без затримок.

Деякі вбудовані системи реального часу потребують гарантії щодо часу виконання операцій. Наприклад, у системах автоматичного керування або системах безпеки можуть бути обмеження на час виконання певних дій. Такі системи повинні забезпечувати виконання операцій у встановлених таймінгах, щоб забезпечити надійну та безпечну роботу.

Вбудовані системи реального часу повинні бути здатні обробляти переривання і винятки швидко та надійно. Це можуть бути непередбачувані події або помилки, які виникають у системі. Система повинна бути відповідно налаштована, щоб виявляти, обробляти і відновлюватися від таких помилок, забезпечуючи продовження роботи в режимі реального часу.

Ці особливості вимагають уважного проектування, розробки та аналізу вбудованих систем реального часу. Враховуючи їх, можна забезпечити стабільну та надійну роботу системи у вимогливих режимах реального часу.

Метод дослідження роботи вбудованих процесорних систем за допомогою логування є досить поширеним і ефективним підходом. Логування включає запис інформації про події, стан системи, діагностичні повідомлення та інші корисні дані під час роботи системи. При цьому, існує декілька способів, як можна використовувати логування для дослідження роботи вбудованих процесорних систем та полягають у наступному.

Лог-файли можуть містити інформацію про важливі події, які відбуваються у системі. Це можуть бути команди, вхідні та вихідні дані, запуски або завершення операцій, помилки тощо. Запис цих подій дозволяє аналізувати послідовність подій та виявляти потенційні проблеми або аномальну поведінку системи [14,16].

Лог-файли можуть містити діагностичні повідомлення, які допомагають розуміти стан системи та виявляти помилки або невідповідності. Ці повідомлення можуть включати інформацію про стан регістрів, значення

змінних, виконані операції тощо. Аналізуючи ці повідомлення, можна виявити проблеми та вдосконалити роботу системи.

Лог-файли можуть також містити дані про час виконання операцій, навантаження на систему, використання ресурсів тощо. Записуючи ці дані під час роботи системи, можна оцінити продуктивність системи, виявити проблемні місця та здійснити оптимізацію.

В певних ситуаціях лог-файли можуть служити інструментом для відлагодження проблем та пошуку помилок у системі. Записуючи дані під час виникнення проблеми, можна проаналізувати послідовність подій, стан системи та знайти причини помилки.

Використовуючи логування, можна отримати значну кількість даних про роботу вбудованих процесорних систем і аналізувати їх для виявлення проблем, оптимізації та вдосконалення роботи системи.

Метод дебагінгу є потужним інструментом для дослідження роботи вбудованих процесорних систем. Він дозволяє аналізувати та відлагоджувати програми, виявляти й усувати помилки, а також розуміти внутрішню логіку та стан системи. При цьому, існує декілька способів, як можна використовувати дебагінг для дослідження роботи вбудованих процесорних систем наступним чином [17,19].

Дебагери дозволяють встановлювати точки зупинки у програмі, що призупиняють її в заданих місцях виконання. Це дозволяє аналізувати стан системи в цей момент, перевіряти значення змінних, реєстрів та інших ресурсів [18-20]. Встановлення точок зупинки допомагає виявляти та усувати проблеми у програмі.

Дебагери надають можливість стежити за послідовним виконанням програми та аналізувати її крок за кроком. Це дозволяє виявити помилки в логіці програми, некоректне виконання операторів, невідповідності вхідних і вихідних даних тощо.

Дебагери дозволяють відстежувати стек викликів програми, тобто послідовність функцій, які були викликані в момент виконання. Це допомагає

розуміти послідовність виконання функцій, виявляти помилки у передачі параметрів, аналізувати використання пам'яті та ресурсів.

Дебагери можуть надавати можливість виводити різноманітну інформацію на консоль або у вікна спостереження. Це можуть бути значення змінних, реєстрів, стану ресурсів або будь-яка інша корисна інформація. Виведення на консоль допомагає аналізувати стан системи та виявляти помилки.

Дебагери зазвичай надають набір команд, які можна виконувати під час виконання програми. Ці команди дозволяють контролювати виконання програми, модифікувати значення змінних, перейменовувати функції, виводити додаткову інформацію тощо. Використання дебагерних команд дозволяє більш гнучко контролювати і досліджувати роботу системи.

Використовуючи дебагінг, можна виявляти й усувати помилки, аналізувати стан системи та внутрішню логіку програми. Це допомагає покращити роботу вбудованих процесорних систем та забезпечити їх стабільну та надійну роботу.

Метод профілювання є ефективним інструментом для дослідження роботи вбудованих процесорних систем. Він дозволяє отримувати детальну інформацію про виконання програми, вимірювати час виконання окремих функцій, визначати використання ресурсів та ідентифікувати узагальнені зони оптимізації. Використовувати метод профілювання для дослідження роботи вбудованих процесорних систем можливо наступним чином.

Профілювання може вимірювати час виконання окремих функцій програми. Це дозволяє виявити, які функції займають більше часу та ресурсів, ідентифікувати "гарячі" місця в програмі і потенційні проблемні ділянки.

Профілювання може визначати використання пам'яті під час виконання програми. Це допомагає виявити витоки пам'яті, надмірне використання пам'яті та інші проблеми, пов'язані з управлінням пам'яттю.

Профілювання може вимірювати, як часто викликаються окремі функції в програмі. Це дозволяє виявити, які функції викликаються найбільш часто, і встановити пріоритети для оптимізації.

Профілювання може вимірювати використання процесора під час виконання програми. Це допомагає виявити, які частини програми вимагають більше обчислювальних ресурсів [16,18], і допомагає здійснити оптимізацію використання процесора.

Профілювання може стежити за стеком викликів програми, тобто послідовністю функцій, які були викликані під час виконання. Це допомагає зрозуміти послідовність виконання функцій і ідентифікувати зайві або зациклені виклики.

Профілювання може генерувати докладні звіти, які містять інформацію про час виконання, використання ресурсів, статистику функцій та інші показники. Ці звіти допомагають аналізувати роботу системи та виявляти проблеми для подальшої оптимізації.

Використовуючи методи профілювання, можна отримати значну кількість даних про роботу вбудованих процесорних систем і провести детальний аналіз їх продуктивності, ефективності та оптимізаційних можливостей.

Моніторинг ресурсів є важливим методом дослідження роботи вбудованих процесорних систем. Він дозволяє в реальному часі спостерігати використання ресурсів системи, таких як процесор, пам'ять, ввод/вивід, мережа та інші. Використовувати моніторинг ресурсів для дослідження роботи вбудованих процесорних систем можливо наступним чином.

За допомогою спеціальних програмних засобів можна відслідковувати завантаженість процесора у реальному часі. Це дозволяє виявити, які процеси або задачі споживають багато процесорного часу та визначити пріоритети для оптимізації.

Моніторинг ресурсів дозволяє відстежувати використання пам'яті системою. Це допомагає виявити витоки пам'яті, надмірне використання пам'яті та інші проблеми, пов'язані з управлінням пам'яттю.

За допомогою моніторингу можна аналізувати швидкість вводу/виводу даних, кількість операцій вводу/виводу, розмір передачі даних та інші параметри [19,20]. Це допомагає виявляти проблеми з швидкістю передачі даних,

розраховувати оптимальний розмір буферу та виконувати оптимізацію вводу/виводу.

Для вбудованих систем, які мають мережеве підключення, моніторинг мережі дозволяє відстежувати пропускну здатність, пакети передачі даних, затримки та інші параметри мережевого зв'язку. Це допомагає виявити проблеми зі з'єднанням, швидкістю передачі даних та виконувати оптимізацію мережевої роботи.

Моніторинг ресурсів може генерувати докладні звіти, які містять інформацію про використання ресурсів, статистику, тренди та інші показники. Ці звіти допомагають аналізувати роботу системи, виявляти проблеми та встановлювати пріоритети для оптимізації.

Моніторинг ресурсів дозволяє отримати важливу інформацію про використання ресурсів вбудованих процесорних систем і виявляти потенційні проблеми для подальшої оптимізації та покращення продуктивності.

Методи дослідження роботи вбудованих процесорних систем за допомогою тестування у реальному часі полягають у виконанні спеціальних тестів або сценаріїв, які спрямовані на перевірку роботи системи в реальному часі. Основна ідея використання цих методів полягає в тому, що з'являється можливість перевіріти реакцію системи на реальні вхідні дані або події. Для цього аналізуються результати досліджень, щоб оцінити продуктивність, надійність та інші характеристики системи. Наведемо декілька способів, як можна проводити методи дослідження роботи вбудованих процесорних систем за допомогою тестування у реальному часі.

Розробка та виконання сценаріїв навантаження, які моделюють реальне навантаження на систему. Це може включати введення великої кількості даних, паралельне виконання багатьох завдань або симуляцію реальних умов роботи.

Виконання тестів, які моделюють реакцію системи на реальні події або вхідні дані. Наприклад, вхідні дані можуть бути сигнали з датчиків або команди зовнішніх пристроїв. Тестування дозволяє оцінити швидкість реакції, точність обробки та здатність системи працювати у реальному часі.

Виконання тестів, спрямованих на виявлення можливих проблем, таких як витоки пам'яті, переповнення буфера або відмови системи. Тестування надійності допомагає забезпечити, що система працює стабільно та надійно у реальному середовищі.

Вимірювання часу, який потрібно системі для обробки певних операцій або відповіді на певні запити. Це допомагає оцінити продуктивність системи та виявити можливі проблеми зі затримками.

Використання автоматизованих тестів і фреймворків для виконання різних тестів у реальному часі. Це дозволяє швидко та ефективно виконувати багаторазові тести та отримувати повторювані результати.

Використання методів тестування у реальному часі дозволяє детально вивчити роботу вбудованих процесорних систем, оцінити їх продуктивність та надійність, а також виявити та вирішити можливі проблеми.

1.3. Апаратні методи дослідження роботи вбудованих процесорних систем

Методи дослідження за допомогою апаратних засобів включають в себе використання спеціальних апаратних засобів для спостереження, моніторингу або вимірювання показників роботи вбудованої системи (табл. 1.2). Ці засоби можуть бути інтегрованими в саму систему або підключатися ззовні. Вони дозволяють отримувати більш точні, деталізовані та неперервні дані про роботу системи, споживання ресурсів, енергії, сигналів тощо.

Таблиця 1.2

Апаратні методи дослідження роботи вбудованих процесорних систем

Метод дослідження	Опис
Профілювання процесора	Вимірювання часу виконання окремих функцій та інструкцій процесора за допомогою лічильників продуктивності. Дозволяє аналізувати продуктивність та оптимізовувати код.
Запис та аналіз трасування виконання	Запис послідовності виконаних інструкцій процесором для аналізу програмного коду та

завдань	виявлення помилок.
Аналіз роботи кеш-пам'яті	Вимірювання активності та промахів кеш-пам'яті процесора для виявлення проблем з кешуванням та оптимізації пам'яті.
Моніторинг апаратних подій	Перехоплення та аналіз апаратних подій, таких як переривання та помилки пам'яті, для відлагодження та виявлення апаратних проблем.
Зондування подій	Встановлення зондів на внутрішні шини процесора для перехоплення та аналізу даних та комунікації між компонентами системи.
Моніторинг температури	Вимірювання температури процесора та інших компонентів для виявлення перегріву та керування температурою.
Моніторинг енергоспоживання	Вимірювання енергоспоживання процесора та системи для оцінки енергоефективності та зменшення споживання енергії.
Моніторинг пам'яті	Вимірювання використання оперативної та кеш-пам'яті для виявлення проблем з витоком пам'яті та оптимізації пам'яті.
Моніторинг використання ресурсів	Вимірювання використання ресурсів, таких як CPU та пам'ять, для оцінки продуктивності та оптимізації використання ресурсів.
Вимірювання швидкості обробки даних	Вимірювання кількості операцій, які система виконує за одиницю часу для оцінки продуктивності.
Вимірювання часу відгуку	Вимірювання часу, необхідного для виконання операцій або завдань системою для виявлення затримок та відповіді.
Вимірювання латентності та пропускної здатності	Вимірювання затримок та швидкості передачі даних для оцінки продуктивності та латентності системи.

Апаратні методи дослідження вбудованих процесорних систем з використанням спеціальних апаратних засобів для спостереження дозволяють отримувати детальну інформацію про поведінку та функціонування процесорних

систем [16,19]. Особливості таких методів включають наступні напрямки досліджень.

Метод профілювання процесора дозволяє отримати інформацію про час виконання окремих функцій та інструкцій процесора. Для цього використовуються спеціальні апаратні засоби, такі як лічильники продуктивності (performance counters), які можуть рахувати кількість виконаних інструкцій, кількість циклів, кількість кеш-промахів тощо. Зібрані дані можуть бути використані для аналізу продуктивності процесорної системи, виявлення гарячих точок та оптимізації коду.

Запис та аналіз трасування виконання завдань (execution trace). Цей метод полягає в записі послідовності виконаних інструкцій процесором. Використовуються спеціальні апаратні засоби, які дозволяють перехопити інструкції та параметри їх виконання. Записана траса виконання може бути використана для аналізу програмного коду, виявлення помилок, профілювання та оптимізації алгоритмів.

Аналіз роботи кеш-пам'яті. Деякі спеціальні апаратні засоби дозволяють вимірювати активність та промахи кеш-пам'яті процесора. Це дозволяє виявити проблеми з кешуванням, такі як часті промахи, недостатній розмір кеш-пам'яті або неправильна організація даних, що може призводити до падіння продуктивності.

Моніторинг апаратних подій. Деякі процесорні системи мають спеціальні апаратні засоби для перехоплення та аналізу апаратних подій, таких як переривання, помилки пам'яті, зміни стану реєстрів тощо. Це може бути корисним для налагодження системи, виявлення апаратних проблем або відлагодження підпрограм, що викликають проблеми.

Зондування подій. Цей метод полягає в встановленні зондів (probes) на внутрішні шини процесора для перехоплення даних або контролю над ними. Зонди можуть бути використані для перехоплення та аналізу даних, що передаються між різними компонентами системи, виявлення проблем з комунікацією, дослідження впливу деяких операцій на систему тощо.

Ці апаратні методи дослідження дозволяють здійснити детальний аналіз процесорних систем і отримати значну кількість інформації про їх функціонування та продуктивність. Вони є корисними інструментами для розробників, які дозволяють виявити проблеми, оптимізувати код та покращити продуктивність вбудованих процесорних систем.

Апаратні методи дослідження вбудованих процесорних систем з використанням моніторингу надають можливість отримати інформацію про різні аспекти функціонування процесора та системи загалом. Особливості таких методів включають наступні напрямки досліджень.

Спеціальні апаратні засоби можуть вимірювати різні показники продуктивності процесора, такі як використання ресурсів (CPU utilization), завантаження системи (system load), час відгуку (response time) та інші [18,20]. Це дозволяє оцінити ефективність роботи процесора та виявити проблеми з продуктивністю.

Засоби моніторингу температури дозволяють вимірювати температуру процесора та інших компонентів системи. Це особливо важливо в вбудованих системах, де ефективне керування температурою є критичним. Моніторинг температури дозволяє виявити перевищення допустимих значень температури та вжити заходи для запобігання перегріву.

Деякі апаратні засоби дозволяють вимірювати енергоспоживання процесора та системи в цілому. Це допомагає виявити енергоефективність системи, знайти енергозатратні ділянки коду або компоненти, а також розробити стратегії енергозбереження.

Деякі апаратні методи дозволяють перехоплювати та аналізувати дані, що передаються через шини або канали зв'язку в системі. Це дозволяє виявити проблеми з комунікацією, аналізувати швидкість передачі даних та виявляти конфлікти чи помилки передачі.

Засоби моніторингу пам'яті дозволяють вимірювати використання оперативної пам'яті та кеш-пам'яті процесора. Це корисно для виявлення

проблем з витоком пам'яті, оптимізації використання пам'яті та виявлення пам'яткових помилок.

Ці апаратні методи дослідження з моніторингу надають розширені можливості для аналізу та вдосконалення вбудованих процесорних систем. Вони допомагають виявляти проблеми, покращувати продуктивність, забезпечувати надійність та знижувати споживання енергії системою.

Апаратні методи дослідження вбудованих процесорних систем з використанням вимірювання показників роботи надають можливість отримати кількісні дані про різні аспекти функціонування системи. Особливості таких методів включають наступні напрямки досліджень.

Вимірювання швидкості обробки даних. Спеціальні апаратні засоби дозволяють вимірювати кількість операцій, які система може виконати за одиницю часу. Це дозволяє оцінити продуктивність системи та виявити проблеми з швидкістю обробки даних.

Вимірювання часу відгуку. Засоби вимірювання часу дозволяють вимірювати час, необхідний для виконання певних операцій або задач системою. Це корисно для оцінки ефективності системи та виявлення проблем зі затримками або підвищеною відповідь на запити.

Вимірювання обсягу використання ресурсів. Деякі апаратні методи дозволяють вимірювати використання ресурсів системою, таких як CPU, пам'ять, мережеві ресурси тощо [21]. Це допомагає виявити перевантаження або недостатнє використання ресурсів і прийняти відповідні заходи для оптимізації використання ресурсів.

Вимірювання енергоспоживання. Апаратні засоби дозволяють вимірювати енергоспоживання системи або окремих компонентів. Це дозволяє оцінити енергоефективність системи та виявити можливості для енергозбереження.

Вимірювання латентності та пропускної здатності. Деякі апаратні засоби дозволяють вимірювати латентність (затримку) та пропускну здатність системи під час обробки даних або передачі інформації. Це корисно для оцінки продуктивності системи та виявлення проблем з продуктивністю.

Ці апаратні методи дослідження з вимірювання показників роботи надають кількісну інформацію про функціонування вбудованих процесорних систем. Вони допомагають виявляти проблеми, вдосконалювати продуктивність та забезпечувати ефективну роботу системи.

Таким чином варто зазначити, що деякі методи можуть поєднувати програмні та апаратні засоби для досягнення більш повного аналізу роботи вбудованих процесорних систем. Залежно від завдань та потреб аналізу можуть використовуватися різні комбінації методів для отримання найбільш повної та точної інформації про систему.

1.4. Задачі дослідження

1. Виконати аналіз методів роботи вбудованих процесорних систем.
2. Визначити програмні та апаратні методи дослідження роботи вбудованих процесорних систем.
3. Виконати дослідження програмних особливостей у використанні методів вбудованих процесорних систем.
4. Виконати дослідження апаратних особливостей у використанні методів вбудованих процесорних систем.
5. Розробити вимоги щодо основних параметрів роботи вбудованих процесорних систем.
6. Дослідити налаштування основних параметрів роботи вбудованих процесорних систем.
7. Дослідити налаштування додаткових параметрів роботи вбудованих процесорних систем.
8. Розробити програмний засіб з аналізу роботи вбудованих процесорних систем на основі програмних та апаратних методів дослідження.

1.5. Висновки

В першому розділі магістерській роботі було проведено аналіз методів роботи вбудованих процесорних систем, який включав в себе вивчення

прикладних програмних засобів, програмних методів та апаратних методів дослідження цих систем. За допомогою прикладних програмних засобів було відстежено та проаналізована робота вбудованих процесорних систем на рівні програмного забезпечення. Це дозволило виявляти проблеми та здійснювати оптимізацію на рівні алгоритмів та програмного коду.

Деякі програмні методи дослідження роботи вбудованих систем включали в себе моделювання та симуляцію їхньої роботи. Ці методи дозволяли аналізувати програмний засіб в різних умовах.

Апаратні методи дослідження роботи вбудованих систем включали в себе використання спеціалізованого обладнання для моніторингу та аналізу роботи різних систем. Ці методи дозволяють отримувати точні дані про фізичну роботу процесорів та інших апаратних компонентів системи.

Необхідно відзначити, що комбінування різних методів аналізу є найбільш ефективним підходом до вивчення та оптимізації роботи вбудованих процесорних систем. Використання прикладних програмних та апаратних засобів разом з різними методами дослідження дозволяє підвищити ефективність роботи систем в цілому.

РОЗДІЛ 2
ОСОБЛИВОСТІ ВИКОРИСТАННЯ МЕТОДІВ ВБУДОВАНИХ ПРОЦЕСОРНИХ СИСТЕМ

2.1. Програмні особливості використання методів вбудованих процесорних систем

Використання методів вбудованих процесорних систем (embedded systems) вимагає уваги щодо ряду програмних особливостей, які забезпечують ефективну роботу таких систем. Розглянемо деякі з них (рис. 2.1).

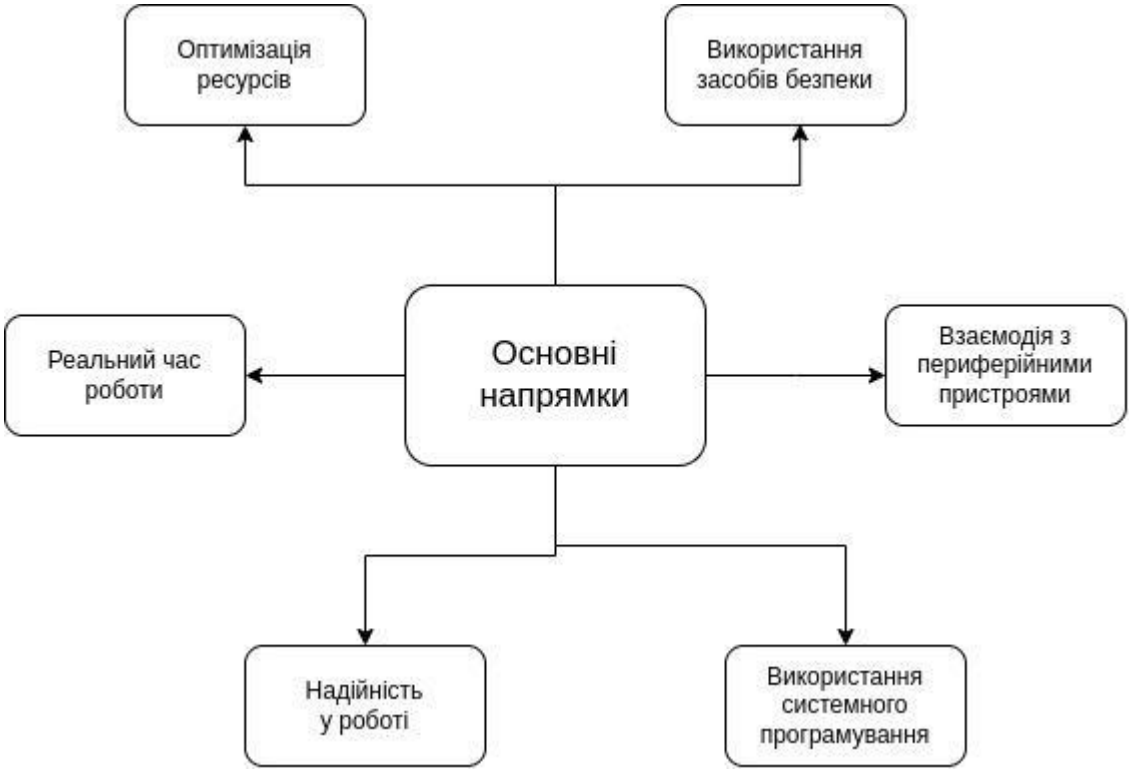


Рисунок 2.1 - Основні напрямки програмних особливостей використання методів вбудованих процесорних систем

Вбудовані системи мають обмежені обчислювальні потужності, пам'ять та енергопотребу. Тому програми для вбудованих систем повинні бути оптимізовані для використання обмежених ресурсів, наприклад, шляхом ефективного використання пам'яті, оптимізованого коду та енергозбереження.

Оптимізація ресурсів в контексті використання методів вбудованих процесорних систем означає забезпечення ефективного використання

обмежених обчислювальних потужностей, пам'яті, енергії та інших ресурсів, що доступні вбудованій системі. Це важливий аспект розробки програмного забезпечення для вбудованих систем, оскільки такі системи зазвичай мають обмежені ресурси порівняно з загальними комп'ютерами або серверами.

Оптимізація ресурсів може включати різні підходи та методики, зокрема такі. Розробники вбудованих систем повинні ефективно використовувати доступну пам'ять, уникати непотрібного витрачання пам'яті та уникати переповнення буферів. Вони можуть використовувати компактні структури даних, уникати непотрібних алокацій пам'яті, використовувати локальні змінні та оптимізовані алгоритми для економії пам'яті.

Розробники систем повинні стежити за ефективністю обчислювальних операцій, зокрема циклів, операцій з плаваючою комою та інших обчислювальних завдань. Вони можуть використовувати оптимізований код, вбудовані функції апаратного прискорення, використовувати апаратну підтримку операцій для покращення швидкодії та зменшення споживання енергії. Вбудовані системи, особливо ті, що працюють на батареї, повинні ефективно використовувати енергію. Розробники можуть використовувати різні методи енергозбереження, такі як режими сну, вимкнення непотрібних пристроїв або модулів, оптимізація роботи алгоритмів та планування ресурсів.

Використання оптимізованих компіляторів та оптимізація коду можуть покращити швидкодію та ефективність вбудованих систем. Розробники можуть використовувати специфічні для платформи оптимізації, такі як оптимізація пам'яті, згортання циклів, унікальні властивості апаратного забезпечення тощо.

Врахування та виконання цих оптимізацій допомагає забезпечити ефективне використання ресурсів вбудованої системи, що є особливо важливим для досягнення необхідного функціоналу, продуктивності та надійності.

Багато вбудованих систем мають вимоги до обробки в реальному часі. Це означає, що програми повинні виконувати завдання відповідно до визначених строків, щоб уникнути відставання або втрати даних. Для досягнення цього

можуть використовуватися методи планування та пріоритетів задач, а також оптимізація алгоритмів.

Реальний час (real-time) в контексті використання методів вбудованих процесорних систем означає виконання задач та обробку даних з мінімальними затримками або гарантованою обробкою визначеними строками. Основна вимога реального часу полягає в тому, щоб система була здатна взаємодіяти зі своєму навколишнім середовищем у визначених строках, незалежно від навантаження та інших факторів.

При цьому, існує два типи систем реального часу. Системи реального часу "жорсткі" (hard real-time) мають певні особливо. У цих системах дотримання обмежень часу є критично важливим. Порушення строків виконання може призвести до неприпустимих наслідків, таких як втрата даних, невдача в системі або навіть загроза життю і безпеці. Прикладами таких систем можуть бути системи управління ракетоносіями, медичні пристрої або системи автопілоту в літаках [21,23].

Системи реального часу "м'які" (soft real-time) мають також певні особливо. У цих системах дотримання строків є важливим, але може бути деяка гнучкість в тому, що трохи відхиляється від визначених строків. Порушення строків може призвести до погіршення продуктивності системи або якості обслуговування, але не викликає неприпустимих наслідків. Прикладами можуть бути системи відеострімінгу, системи управління трафіком або системи контролю виробництва.

Для досягнення реального часу роботи вбудованої системи необхідно використовувати методи планування та пріоритетів задач, а також оптимізацію алгоритмів. Задачі в системі повинні бути виконані у визначені строки, щоб уникнути відставання та втрати даних, забезпечуючи відповідну реакцію на зовнішні події та вимоги системи. Крім того, можуть використовуватися реального часу операційні системи (RTOS), які спеціально призначені для вбудованих систем і мають механізми для керування задачами в реальному часі.

Забезпечення реального часу є важливою програмною особливістю вбудованих процесорних систем, особливо в тих випадках, коли промахи строків можуть мати критичні наслідки. Розглянемо деякі з них.

Вбудовані системи часто використовуються в критичних застосунках, де недостаток надійності може мати серйозні наслідки. Тому програми для вбудованих систем повинні бути добре протестовані і перевірені на відповідність вимогам надійності.

Надійність у роботі в контексті використання методів вбудованих процесорних систем означає здатність системи виконувати свої функції без відмов або непередбачуваних помилок протягом тривалого періоду часу. Надійність є критично важливою властивістю вбудованих систем, особливо тих, що використовуються в критичних застосунках, де відмова системи може призвести до серйозних наслідків, таких як втрата даних, матеріальні збитки або навіть загроза життю.

Для досягнення надійності вбудованої системи можуть використовуватися різні методи та стратегії. Використання відмовостійких компонентів та алгоритмів допомагає забезпечити продуктивність системи навіть у разі відмови окремих компонентів. Це може включати використання дублювання компонентів, виконання резервних копій даних, механізми виявлення та відновлення відмов тощо.

Вбудовані системи можуть включати механізми для перевірки цілісності даних та самодіагностики. Це допомагає виявляти помилки та відновлюватися після них. Наприклад, можуть бути використані контрольні суми, коди перевірки четності (CRC), механізми контролю та відновлення помилок (ECC) та інші методи для виявлення та корекції помилок.

Ретельне тестування та верифікація системи допомагають виявити та усунути потенційні помилки та дефекти ще до введення системи в експлуатацію [22,23]. Це включає тестування на рівні модулів, інтеграційне тестування, тестування на реальних сценаріях роботи та перевірку відповідності специфікаціям.

Вбудовані системи можуть бути піддаються різним зовнішнім загрозам, таким як злам, вторгнення або віруси. Для забезпечення надійності необхідно використовувати механізми захисту, такі як шифрування, автентифікація, контроль доступу та інші методи для забезпечення конфіденційності, цілісності та доступності даних та системи.

Загалом, надійність у роботі вбудованих процесорних систем забезпечується шляхом застосування відповідних стратегій, які мінімізують можливі відмови, забезпечують швидке виявлення та відновлення від них, та забезпечують захист від зовнішніх загроз.

Вбудовані системи часто вимагають низькорівневого програмування, так як вони безпосередньо взаємодіють з апаратними компонентами. Це означає, що розробники повинні мати глибокі знання архітектури процесорів, мов асемблера та системного програмування.

Системне програмування в особливостях використання методів вбудованих процесорних систем відноситься до розробки програмного забезпечення, яке працює на рівні операційної системи або нижче, безпосередньо взаємодіючи з апаратурою вбудованої системи. Це включає програмування драйверів пристроїв, ядра операційної системи, системних сервісів та інших компонентів, які забезпечують функціонування системи.

Особливості системного програмування вбудованих процесорних систем можуть включати наступні напрямки.

Системні програми мають взаємодіяти з різними апаратними пристроями, такими як процесори, пам'ять, периферійні пристрої (наприклад, сенсори, актуатори, комунікаційні інтерфейси тощо). Розробники системного програмного забезпечення повинні мати розуміння апаратури та вміти ефективно взаємодіяти з нею.

У системному програмуванні важливо досягти високої продуктивності системи, особливо в умовах обмежених ресурсів вбудованих систем. Це включає оптимізацію коду, використання оптимізованих алгоритмів та структур даних,

ефективне управління пам'яттю та оптимізацію використання процесорного часу [21].

Вбудовані системи можуть мати спеціалізовані операційні системи або використовувати покрокові операційні системи (RTOS). Системні програмісти повинні мати знання про роботу з цими операційними системами, включаючи розробку драйверів пристроїв, планування задач, синхронізацію та керування пам'яттю.

Системні програми повинні бути надійними та безпечними, оскільки вони працюють у низькорівневих шарах системи та мають прямий доступ до апаратних ресурсів. Розробники системного програмного забезпечення повинні уникати помилок програмування, розробляти механізми перевірки та валідації даних, а також використовувати методи захисту від зловмисних атак та вторгнень.

Всі ці особливості системного програмування вбудованих процесорних систем допомагають забезпечити ефективну та надійну роботу вбудованих систем у різних застосуваннях.

Вбудовані системи зазвичай взаємодіють з різними периферійними пристроями, такими як сенсори, актуатори, комунікаційні модулі тощо. Розробники повинні мати знання про протоколи зв'язку та специфікації пристроїв, щоб правильно інтегрувати їх у програму.

Взаємодія з периферійними пристроями в особливостях використання методів вбудованих процесорних систем означає комунікацію та керування зовнішніми пристроями, які підключені до вбудованої системи. Периферійні пристрої можуть включати сенсори, актуатори, комунікаційні інтерфейси (наприклад, USB, Ethernet, UART), аналогово-цифрові та цифро-аналогові перетворювачі, пам'ять і багато іншого.

Взаємодія з периферійними пристроями включає наступні аспекти. Перед використанням периферійного пристрою його потрібно налагодити та налаштувати. Це може включати встановлення параметрів комунікації, режимів роботи, вибір розміру даних та інших налаштувань.

Вбудована система може зчитувати дані з периферійних пристроїв, таких як сенсори, або записувати дані на актуатори. Це включає взаємодію з реєстрами, буферами або іншими інтерфейсами, щоб передавати дані в обидва напрямки.

Деякі периферійні пристрої можуть генерувати переривання для сповіщення вбудованої системи про певні події або стан пристрою. Розробники системного програмного забезпечення повинні обробляти ці переривання та вживати необхідні дії відповідно до них.

Взаємодія з периферійними пристроями може включати управління їх життєвим циклом, таким як увімкнення, вимкнення, переведення в режим сну або збудження з режиму сну.

При взаємодії з периферійними пристроями можуть виникати помилки, такі як втрата зв'язку, неправильні дані або перевантаження буфера. Розробники системного програмного забезпечення повинні передбачити та обробляти ці помилки, вживаючи відповідних заходів, таких як повторна спроба, перезавантаження пристрою або інші відновлювальні дії.

Таким чином, взаємодія з периферійними пристроями є важливою складовою вбудованих процесорних систем [22-24], оскільки дозволяє їм взаємодіяти з реальними світом та виконувати потрібні функції залежно від зовнішніх умов і вхідних даних.

Вбудовані системи, особливо ті, що використовуються в критичних інфраструктурних застосунках, повинні бути захищеними від вторгнень та зловмисних дій. Розробники повинні враховувати аспекти безпеки, такі як шифрування даних, захист від вразливостей та автентифікацію.

Засоби безпеки в особливостях використання методів вбудованих процесорних систем означають набір заходів і технологій, що використовуються для захисту вбудованих систем від різних загроз і зловмисних дій. Ці заходи спрямовані на забезпечення конфіденційності, цілісності та доступності даних та функціональності системи.

Деякі засоби безпеки, що використовуються в вбудованих процесорних системах, включають наступні напрями.

Засоби аутентифікації використовуються для перевірки ідентичності користувачів або компонентів системи. Це може включати використання паролів, ключів доступу, біометричних методів (наприклад, сканер відбитків пальців) або інших методів для підтвердження правильності доступу до системи. Шифрування використовується для захисту конфіденційності даних шляхом перетворення їх у незрозумілу форму для несанкціонованих осіб. Вбудовані системи можуть використовувати різні алгоритми шифрування для захисту даних, передаваних через мережі або зберігаються на носіях інформації.

Засоби контролю доступу визначають, хто має право отримати доступ до різних ресурсів і функцій системи. Це може включати налаштування прав доступу для користувачів, ролей або груп, а також реалізацію механізмів обмеження доступу до певних функцій чи областей пам'яті.

Засоби безпеки можуть включати механізми для виявлення атак і зловмисних дій, таких як інтрузія, віруси або зловживання протоколами. Це може включати використання антивірусного програмного забезпечення, мережевих механізмів виявлення вторгнень, механізмів контролю цілісності програмного забезпечення та інших технік для запобігання та виявлення атак.

Засоби безпеки також можуть включати фізичні заходи для захисту вбудованих систем від фізичного доступу несанкціонованих осіб. Це може включати використання фізичних бар'єрів, замків, контролю доступу до приміщень або захисту від зламу корпусу системи.

Таким чином, використання засобів безпеки в особливостях вбудованих процесорних систем допомагає забезпечити захищеність і надійність вбудованих систем в умовах, де вони можуть бути піддаються ризику атак, зломів або несанкціонованого доступу.

2.2. Апаратні особливості використання методів вбудованих процесорних систем

Апаратні особливості використання методів вбудованих процесорних систем можуть значно впливають на ефективність і функціональність у їх роботі. До найбільш важливих з них є наступні.

Вбудовані процесорні системи зазвичай призначені для використання у вбудованих пристроях, де обмежений простір. Тому, вони зазвичай мають компактний розмір, споживають мало енергії та інтегруються на одному чіпі.

Також, у багатьох вбудованих пристроях обмежена ємність батареї або відсутність доступу до постійного джерела живлення. Тому вбудовані процесорні системи мають бути енергоефективними, щоб забезпечити тривалу автономну роботу.

Часто такі вбудовані пристрої мають обмежену пам'ять. Тому вбудовані процесорні системи часто мають вбудовану пам'ять (ROM або Flash) для зберігання програмного забезпечення і даних.

В деяких вбудованих застосуваннях, таких як вимогливі системи реального часу, вимагається швидка обробка даних і висока продуктивність. Вбудовані процесорні системи можуть бути оптимізовані для виконання конкретних завдань, щоб забезпечити високу швидкодію.

Вбудовані процесорні системи можуть мати вбудовані інтерфейси для підключення до різних зовнішніх пристроїв, таких як датчики, актуатори, дисплеї, комунікаційні модулі тощо.

Залежно від використання, вбудовані процесорні системи можуть працювати в екстремальних умовах, що вимагає здатності працювати в широкому температурному діапазоні.

У деяких вбудованих системах, таких як медичні пристрої або системи автономних транспортних засобів, безпека може бути критично важливою. Вбудовані процесорні системи можуть мати апаратну підтримку для криптографії та захисту від злому.

Залежно від застосування, вбудовані процесорні системи можуть мати апаратну підтримку для спеціалізованих операцій, таких як обробка сигналів, графіки, шифрування, обчислення векторів та інші.

Ці апаратні особливості дозволяють вбудованим процесорним системам ефективно виконувати свої завдання в специфічних вбудованих пристроях і системах. Завдяки оптимізації під конкретні задачі вбудовані системи можуть бути меншими, ефективнішими та надійнішими, що робить їх ідеальними для використання в багатьох промислових, медичних, автомобільних та інших застосуваннях.

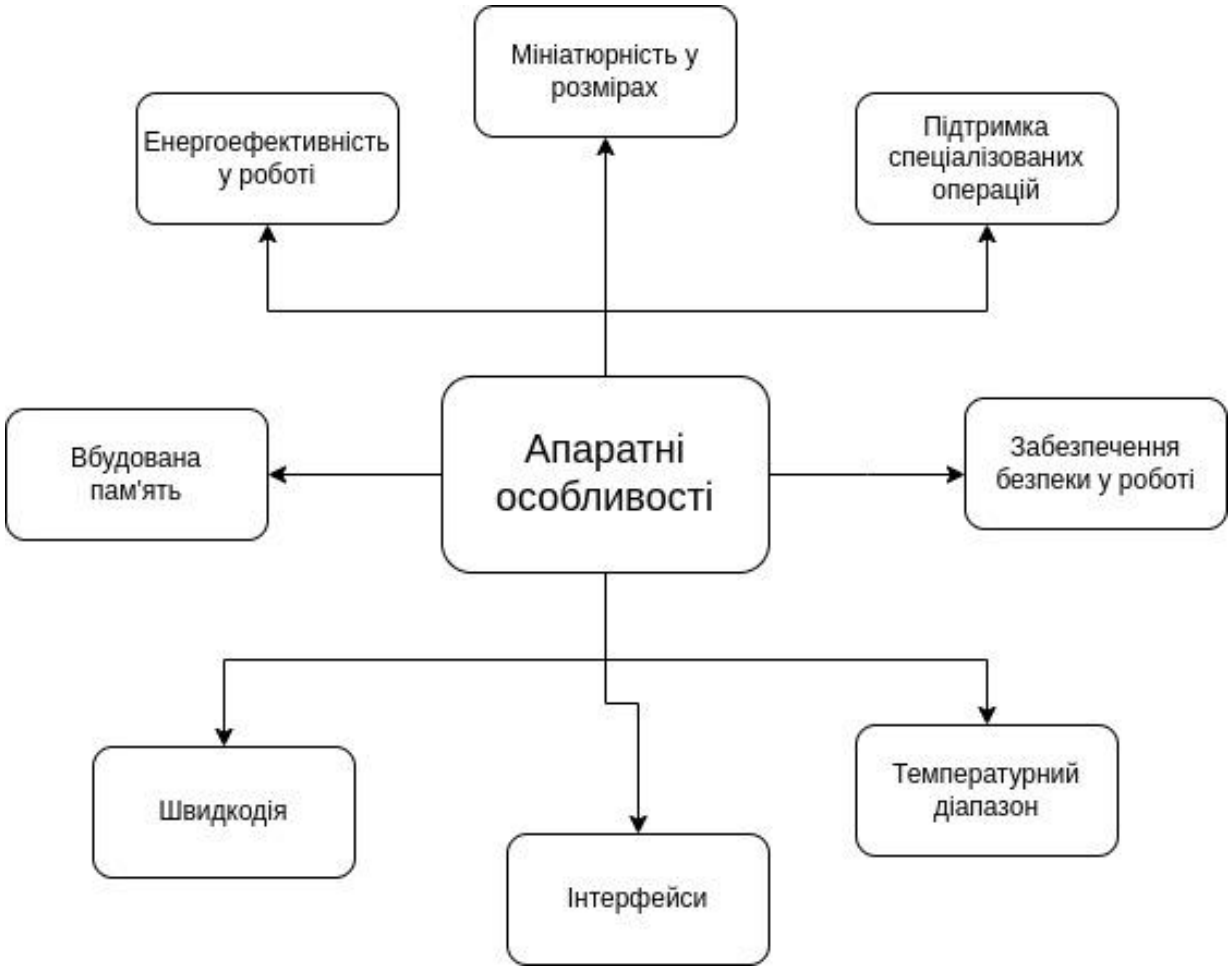


Рисунок 2.2 - Апаратні особливості використання методів вбудованих процесорних систем

Розглянемо такі апаратні особливості використання методів вбудованих процесорних систем більш ретельно (рис. 2.2).

Мініатюрність у розмірах у використанні вбудованих процесорних систем означає, що ці системи виготовлені у відносно невеликому розмірі та компактній формі, щоб вони могли бути успішно вбудовані в пристрої з обмеженим

простором. Основна ідея полягає в тому, щоб забезпечити ефективну роботу пристрою або системи з використанням найменшого можливого обсягу простору. Це особливо важливо в засобах, які мають бути портативними або вбудованими в обмежений простір, наприклад наступні.

Вбудовані процесорні системи у мобільних пристроях мають бути дуже компактними, щоб забезпечити стрункy конструкцію та легку переносимість.

У вбудованих системах IoT часто є обмежені ресурси, тому мініатюрність у розмірах дозволяє легше інтегрувати їх у різні предмети або пристрої.

Вбудовані процесорні системи у медичних пристроях повинні бути досить маленькими, щоб забезпечити зручність для пацієнтів і мобільність медичного персоналу. Вбудовані процесорні системи у сучасних автомобілях використовуються для керування всіма аспектами автомобіля, тому їх розміри мають бути адаптовані для інтеграції у обмеженому просторі.

Мініатюрність у розмірах вбудованих процесорних систем може бути досягнута через технологічний прогрес в мікроелектроніці, такий як зменшення розміру транзисторів, що дозволяє створювати потужні процесори з меншим енергоспоживанням та мініатюрними розмірами [21,23]. Це дозволяє зберігати простір, енергію і сприяє створенню більш розумних і ефективних вбудованих пристроїв.

Енергоефективність у вбудованих процесорних системах означає здатність таких систем ефективно використовувати електричну енергію для виконання своїх завдань з мінімальним споживанням електроенергії. Це дуже важлива характеристика для багатьох вбудованих пристроїв, особливо тих, які працюють в режимі автономного живлення або мають обмежені джерела живлення, такі як батареї або акумулятори.

Основні аспекти енергоефективності вбудованих процесорних систем включають наступні. Вбудовані процесорні системи повинні бути здатні регулювати свою енергоспоживання залежно від потреби. Це може включати зниження тактової частоти, призупинення роботи неактивних блоків або повний сплячий режим при тривалій неактивності.

Багато вбудованих пристроїв перебувають у режимі очікування більшу частину свого часу. Таким чином, енергоефективні процесорні системи мають споживати дуже мало енергії в цьому режимі.

Багато вбудованих процесорних систем підтримують різні енергозберігаючі режими, такі як режими "глибокого сну" або "сплячого режиму". Це дозволяє пристрою вимикати або знижувати споживання енергії у стані неактивності, що забезпечує подовження тривалості живлення.

Ефективні вбудовані процесорні системи використовують оптимальні алгоритми і оптимізований апаратний дизайн для мінімізації витрат електроенергії при виконанні обчислювальних завдань.

У деяких випадках вбудовані процесорні системи можуть використовувати спеціалізовані апаратні блоки для виконання певних завдань [23], що дозволяє знизити енергоспоживання порівняно з використанням загальнопризначених блоків.

Підвищення температури пристрою може спричинити зниження продуктивності та збільшення споживання енергії. Оптимізовані вбудовані процесорні системи допомагають знизити тепловиділення через ефективне керування енергією та розсіювання тепла.

Енергоефективність є критично важливою характеристикою для вбудованих пристроїв, таких як мобільні телефони, носимі пристрої, IoT-пристрої, автономні системи і багато інших. Зниження енергоспоживання не тільки дозволяє подовжити тривалість роботи пристрою, але також може знизити вартість експлуатації та вплинути на екологічну стійкість, зменшивши споживання електроенергії та викиди вуглецю.

Вбудована пам'ять у використанні процесорних систем означає, що на процесорному чипі або на тій же платі, де знаходиться процесор, є пам'ять, яка призначена для зберігання програмного забезпечення, даних та інших інформаційних ресурсів, необхідних для функціонування пристрою або системи [24]. У вбудованих процесорних системах вбудована пам'ять використовується зазвичай для наступних цілей.

Вбудовані процесорні системи часто мають програмне забезпечення (firmware), який представляє собою постійне програмне забезпечення, яке вбудоване безпосередньо в мікросхему пристрою. Це може бути операційна система, контролери пристрою, драйвери та інші низькорівневі компоненти, необхідні для працювання пристрою.

Вбудовані системи можуть мати вбудовану пам'ять для зберігання конфігураційних даних, налаштувань і параметрів, які визначають поведінку пристрою або системи.

Вбудовані процесорні системи можуть мати вбудовані буфери, які використовуються для зберігання проміжних даних під час операцій, що прискорює роботу пристрою.

Деякі вбудовані процесорні системи мають вбудовані пам'ять для зберігання даних, зібраних пристроєм або отриманих зовнішніми датчиками. Це може бути важливо для системи безпеки, збору даних або зберігання логів.

Вбудовані процесорні системи в бездротових пристроях, таких як маршрутизатори або Wi-Fi-модулі, можуть мати вбудовану пам'ять для зберігання налаштувань мережі та сертифікатів безпеки.

Такі вбудовані пам'яті, розташовані на чіпі з процесором, дозволяють спростити дизайн пристрою та підвищити його надійність, оскільки зменшується кількість зовнішніх компонентів і контактів, що можуть вплинути на знос та помилки. Оптимізація інтегрованих рішень також може підвищити продуктивність та знизити споживання енергії, що особливо важливо для вбудованих пристроїв з обмеженими ресурсами.

Швидкодія у використанні вбудованих процесорних систем означає здатність таких систем ефективно та швидко виконувати обчислення та завдання з мінімальним часом відгуку. Це важлива характеристика для багатьох вбудованих пристроїв, особливо тих, які вимагають реального часу або обробки великих обсягів даних. Основні аспекти швидкодії вбудованих процесорних систем включають наступні.

Частота процесора, що визначає, скільки операцій процесор може виконати за одну секунду. Чим вища частота, тим більше операцій може виконати процесор за одиницю часу, що призводить до збільшення швидкодії.

Вбудовані процесори можуть мати одне або кілька ядер, що дозволяє виконувати паралельні операції [22,24]. Чим більше ядер, тим більше завдань може одночасно обробляти процесор, що підвищує швидкодію у багатозадачних середовищах.

Ефективний дизайн апаратного забезпечення, такий як використання швидких кеш-пам'ятей, оптимізованих арифметичних операцій, прискорювачів апаратного рівня тощо, може підвищити швидкодію виконання деяких операцій.

Ефективне програмне забезпечення, оптимізоване для конкретних завдань, може підвищити швидкодію виконання програм та завдань на процесорі.

Швидкодія доступу до пам'яті також може впливати на загальну швидкодію системи. Швидка пам'ять дозволяє процесору швидше звертатись до даних та інструкцій, що прискорює обробку.

Деякі процесори мають можливість виконувати багато операцій одночасно, використовуючи техніку подвійного виконавчого потоку (pipelining) або використання SIMD-інструкцій (Single Instruction, Multiple Data).

Забезпечення високої швидкодії є критично важливим для багатьох вбудованих пристроїв, таких як мобільні пристрої, автономні транспортні засоби, системи відеонагляду, медичні прилади, промислові контролери тощо. Швидкодія дозволяє забезпечити більш відзивчиву та продуктивну роботу пристроїв, а також підвищує їх здатність виконувати вимогливі обчислювальні завдання. У використанні вбудованих процесорних систем інтерфейси означають способи зв'язку та комунікації між процесором та іншими компонентами пристрою або системи. Інтерфейси дозволяють процесорній системі взаємодіяти з різними зовнішніми пристроями, датчиками, актуаторами, пам'яттю та іншими пристроями чи компонентами, що можуть бути підключені до пристрою.

Розглянемо деякі типові інтерфейси, які можуть використовуватися у вбудованих процесорних системах, до яких відносяться наступні.

Серійний порт (Serial Port), що дозволяє передавати дані по одному бітові у вигляді послідовних сигналів. Наприклад, UART (Universal Asynchronous Receiver/Transmitter) є типовим серійним портом для зв'язку з пристроями, такими як сенсори, мікроконтролери тощо.

Шини зовнішніх пристроїв (External Device Interfaces), що призначені для підключення до зовнішніх пристроїв, таких як USB (Universal Serial Bus), Ethernet, HDMI (High-Definition Multimedia Interface) та інші.

I2C (Inter-Integrated Circuit), який являє собою протокол I2C дозволяє підключати пристрої за допомогою двох провідників для передачі даних. Це часто використовується для зв'язку з датчиками та іншими периферійними пристроями.

SPI (Serial Peripheral Interface), що є послідовним протоколом зв'язку, який дозволяє підключати багато пристроїв до одного процесора, використовуючи кілька провідників.

GPIO (General Purpose Input/Output), що являють собою звичайні цифрові вхідні/вихідні піни, які можуть використовуватися для зчитування стану датчиків або керування зовнішніми пристроями.

CAN (Controller Area Network), що являє собою протокол CAN використовується в автомобільній промисловості для зв'язку між різними компонентами автомобільної системи.

SPI (Serial Peripheral Interface), що являє собою ще один послідовний протокол зв'язку, який широко використовується для з'єднання мікроконтролерів, датчиків, дисплеїв та інших периферійних пристроїв.

Ці інтерфейси дозволяють процесорній системі здійснювати обмін даними та комунікацію з різними зовнішніми пристроями, що дозволяє створювати багатофункціональні та здатні до спілкування системи. Залежно від типу пристрою або застосування, вбудовані процесорні системи можуть підтримувати

різні комбінації інтерфейсів для оптимального зв'язку та інтеграції з зовнішніми компонентами.

Температурний діапазон у використанні вбудованих процесорних системах визначає межі температур, при яких пристрій або система може працювати надійно і стабільно. Це важлива характеристика, оскільки температура може сильно впливати на ефективність, надійність і тривалість роботи електронних компонентів, зокрема процесорів. Температурний діапазон вказує на три основних параметри, що означає наступне.

Мінімальна робоча температура, що є найнижчою при якій пристрій або система може нормально функціонувати. Нижче цієї температури електронні компоненти можуть не працювати або працювати зі зниженою ефективністю.

Максимальна робоча температура, що є найвищою при якій пристрій або система може нормально функціонувати. Вище цієї температури можуть виникати проблеми з ефективністю, можуть виникати перебої в роботі або навіть пошкодження компонентів.

Температурний діапазон (Operating Temperature Range) означає інтервал температур між мінімальною та максимальною робочими температурами, в межах якого пристрій або система повинна працювати безперебійно.

Наприклад, температурний діапазон для вбудованих процесорних систем може бути вказаний як -40°C до +85°C, що означає, що пристрій здатен працювати стабільно при температурах від -40°C до +85°C.

Важливо враховувати температурний діапазон при проектуванні і використанні вбудованих процесорних систем, особливо якщо вони будуть встановлені у зовнішніх середовищах або у пристроях, які працюють у важких умовах. Перевищення максимальної робочої температури може призвести до перегріву і збоїв у роботі, тоді як нижча температура може вплинути на надійність і швидкість роботи. Тому важливо підібрати пристрої з температурним діапазоном, що відповідає вимогам середовища, в якому вони будуть використовуватися.

Забезпечення безпеки у роботі у використанні вбудованих процесорних систем означає прийняття заходів і застосування технологій для захисту пристроїв, систем або даних від потенційних загроз і атак, а також для забезпечення надійності, конфіденційності та доступності вбудованих пристроїв. Забезпечення безпеки є особливо важливим у вбудованих процесорних системах, оскільки ці системи часто використовуються в критичних застосуваннях, де можуть бути серйозні наслідки при збої або компрометації. Основні аспекти забезпечення безпеки вбудованих процесорних систем включають наступні заходи.

Криптографічні заходи при яких в пристрої існує використання криптографічних методів, таких як шифрування, хешування, цифровий підпис та інші, для захисту даних від несанкціонованого доступу та зміни.

Аутентифікація і авторизація надає користувачам пристроїв впевненість у тому, що лише дозволені користувачі мають доступ до системи і її ресурсів. Це може включати використання паролів, біометричних ідентифікаторів, токенів тощо. Захист від зовнішніх атак дозволяє застосування заходів для запобігання або ускладнення атак зовнішніх зловмисників, таких як віруси, хакерські атаки, внесення змін у програмне забезпечення або підміна даних.

Фізичні заходи для захисту пристроїв від фізичного доступу до компонентів, що може включати використання захисних корпусів, захисту від ударів і вібрацій.

Аналіз безпеки і поновлення дозволяє проводити аналіз безпеки пристроїв та програмного забезпечення з метою виявлення потенційних загроз і уразливостей, а також регулярне оновлення програмного забезпечення для усунення виявлених проблем.

Захист від внутрішніх загроз забезпечує заходи для захисту від несанкціонованого доступу та дій внутрішніх користувачів або процесів.

Захист інтелектуальної власності забезпечує захист цінної інформації і технологій, що містяться в пристрої або системі, від несанкціонованого використання або крадіжки.

Відновлюваність та стійкість надають можливість у забезпеченні здатності системи відновлюватися після збоїв, а також забезпечення стійкості до електромагнітних перешкод, електричних шоків, радіаційних впливів тощо.

Тому, забезпечення безпеки у вбудованих процесорних системах є складним і багатогранним завданням, оскільки вони можуть використовуватися у різних сферах, від мобільних пристроїв і до автономних систем управління та промислових контролерів. Відповідна інженерна робота та застосування сучасних методів забезпечення безпеки є критичними для успішного функціонування вбудованих процесорних систем і захисту їх користувачів і даних. Підтримка спеціалізованих операцій у використанні вбудованих процесорних системах означає, що процесор має апаратну підтримку для виконання певних операцій або інструкцій, які є спеціалізованими і призначеними для виконання конкретних завдань або обчислень.

Ці спеціалізовані операції можуть стосуватися різних сфер в залежності від типу пристрою або системи, для якої призначений процесор. Деякі приклади спеціалізованих операцій у вбудованих процесорах включають наступні операції [24-26].

Криптографічні операції де існує апаратна підтримка алгоритмів шифрування (наприклад, AES - Advanced Encryption Standard, DES - Data Encryption Standard), хеш-функцій (наприклад, SHA - Secure Hash Algorithm) та інших криптографічних операцій дозволяє прискорити обробку даних для забезпечення безпеки та конфіденційності.

Сигнальний процесінг (Signal Processing) дозволяє проводити підтримку векторних операцій та SIMD-інструкцій (Single Instruction, Multiple Data) може допомогти виконувати обробку аудіо-, відео- та інших сигналів більш ефективно.

Обробка сенсорних даних існують для операцій швидкого оброблення даних з різних типів сенсорів, таких як акселерометри, гіроскопи, датчики дотику тощо.

Кодування і декодування медіа дозволяє виконувати підтримку апаратного кодування та декодування аудіо- та відеоформатів для оптимізації роботи мультимедійних пристроїв.

Математичні операції необхідні для підтримки операцій з плаваючою комою, векторних операцій, операцій з матрицями тощо для обробки числових даних. Керування периферією виконують операції для простішого керування периферійними пристроями, такими як зчитування або запис даних у регістри контролерів.

Таким чином, підтримка спеціалізованих операцій у вбудованих процесорних системах дозволяє підвищити продуктивність, зменшити завантаження процесора та знизити споживання енергії, що особливо важливо для вбудованих пристроїв з обмеженими ресурсами. Вона дозволяє ефективно виконувати спеціалізовані завдання, що допомагає розширити функціональність і забезпечити оптимальну роботу пристроїв у вбудованих системах.

2.3. Порівняльна характеристика використання програмних та апаратних методів вбудованих процесорних систем

Порівняльна характеристика використання програмних та апаратних методів у вбудованих процесорних системах допоможе краще зрозуміти переваги та обмеження кожного підходу. Розглянемо декілька ключових аспектів порівняльної характеристики використання програмних та апаратних методів у вбудованих процесорних системах (рис. ...).

Параметр швидкодія. Програмні методи вимагають більшого часу на виконання, оскільки програмне забезпечення повинно інтерпретувати та виконувати інструкції.

Апаратні методи виконуються безпосередньо апаратурою, що дозволяє їм працювати швидше та ефективніше.

Параметр енергоефективності. Програмні методи зазвичай споживають менше енергії, оскільки використовують менше апаратних ресурсів.

Апаратні методи можуть бути менш енергоефективними, оскільки потребують більше ресурсів для обробки.

Параметр гнучкості та зручності в розробці. Програмні методи забезпечують більшу гнучкість та простоту в розробці, оскільки можуть легко змінюватися за допомогою програмних оновлень.

Апаратні методи можуть бути менш гнучкими, оскільки вимагають фізичних змін апаратури для зміни функціональності.

Параметр розміру та вартості. Програмні методи зазвичай займають менше апаратної площі та коштують менше, оскільки вони не потребують додаткових апаратних блоків.

Апаратні методи можуть потребувати більше фізичного простору та бути дорожчими через використання спеціалізованих компонентів.

Параметр надійності. Програмні методи можуть бути менш надійними через можливі програмні помилки.

Апаратні методи зазвичай надійніші, оскільки вони реалізовані на апаратному рівні і менше піддаються програмним помилкам.

Параметр вартості розробки та час впровадження. Програмні методи зазвичай дешевше розробляти та швидше впроваджувати, оскільки не потрібно виготовляти нові апаратні компоненти.

Апаратні методи можуть бути більш складними та дорогими у розробці, але можуть забезпечувати оптимальну швидкодію та ефективність у довгостроковій перспективі.

У багатьох випадках комбінація програмних і апаратних методів використовується для досягнення найкращого балансу між швидкодією, ефективністю та гнучкістю. Для успішного розробки вбудованих процесорних систем важливо обрати оптимальний підхід, який відповідав би конкретним потребам і вимогам проекту.

Розглянемо основні аспекти порівняльної характеристики використання програмних та апаратних методів у вбудованих процесорних системах більш детально.

Швидкодія (Performance) - це характеристика, яка відображає швидкість та продуктивність виконання обчислень і завдань системою, включаючи вбудовані процесорні системи. Відмінність швидкодії між програмними та апаратними методами полягає у тому, як швидкодія досягається в кожному з підходів наступним чином. Програмні методи для досягнення швидкодії мають такі відмінності.

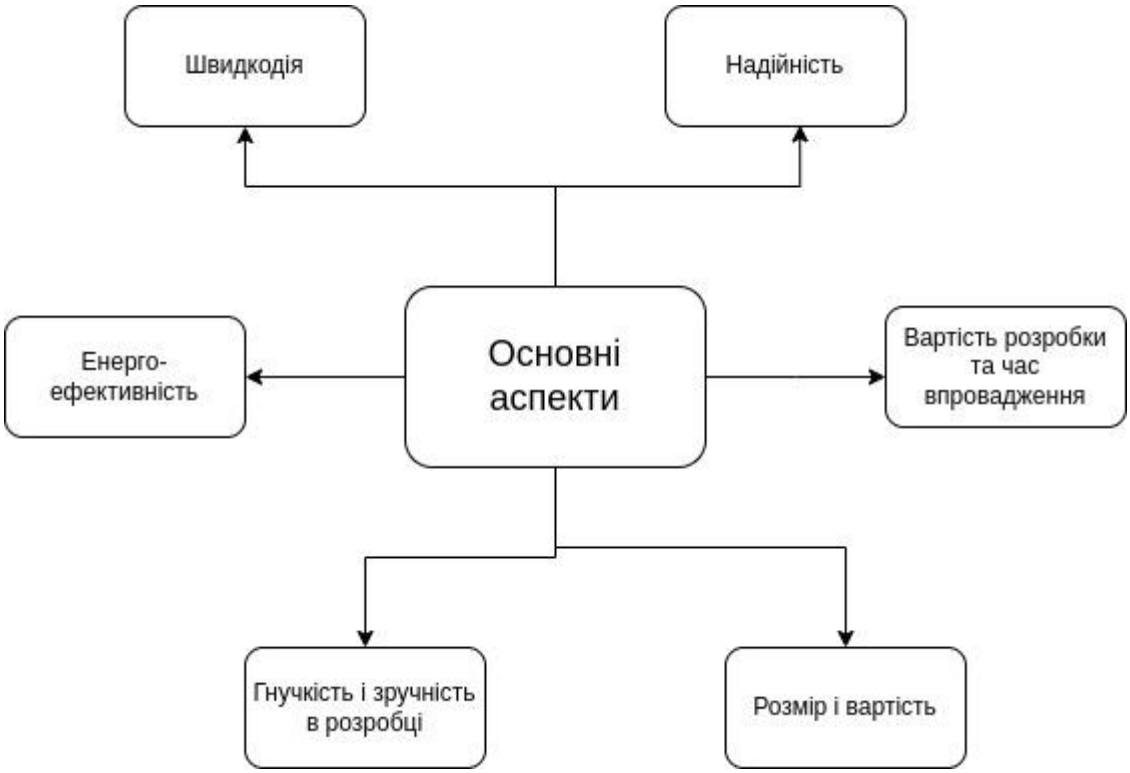


Рисунок 2.3 - Основні аспекти порівняльної характеристики використання програмних та апаратних методів у вбудованих процесорних системах

Ретельна оптимізація програмного коду може знизити час виконання завдань шляхом використання швидших алгоритмів, уникнення непотрібних операцій та покращення алгоритмічної ефективності.

Розбиття завдань на окремі потоки або процеси дозволяє виконувати їх одночасно на багатоядерних або багатопроцесорних системах, що забезпечує прискорення обчислень. Апаратні методи для досягнення швидкодії можуть бути виконані наступним чином.

Використання спеціалізованих апаратних прискорювачів, таких як графічні процесори (GPU), апаратний цифровий сигнальний процесор (DSP) або апаратний прискорювач криптографії, дозволяє виконувати певні операції значно швидше, ніж це було б можливо на звичайному процесорі загального призначення.

Вбудовані процесори можуть мати багатоядерні або паралельні структури, які дозволяють виконувати декілька операцій одночасно, що призводить до покращення швидкодії.

Отже, хоча і програмні, і апаратні методи можуть призвести до покращення швидкодії, апаратні методи зазвичай є більш ефективними для обробки великих обсягів даних або виконання спеціалізованих операцій. Програмні методи, зі свого боку, забезпечують більшу гнучкість та зручність у розробці, особливо коли необхідно швидко внести зміни або адаптувати систему під нові вимоги. При розробці вбудованих процесорних систем важливо ретельно обирати правильний баланс між програмними та апаратними методами для досягнення оптимальної швидкодії та ефективності системи.

Енергоефективність (Energy Efficiency) - це характеристика, яка відображає співвідношення між енергією, яку використовує система для виконання обчислень або завдань, та досягнутою продуктивністю. Основна відмінність енергоефективності між програмними та апаратними методами полягає у тому, як ефективність досягається в кожному з підходів. Програмні методи для досягнення енергоефективності виконуються наступним чином.

Використання більш ефективних алгоритмів, які вимагають менше обчислювальних ресурсів, дозволяє знизити споживання енергії.

Застосування режиму засинання для вимкнення або зниження енергоспоживання пристрою у періоди, коли він не використовується.

Використання алгоритмів управління енергією для зниження частоти роботи процесора, вимикання неактивних компонентів або зменшення напруги під час необхідності.

Апаратні методи для досягнення енергоефективності виконуються наступним чином. Використання ефективних апаратних архітектур, здатних забезпечити необхідну продуктивність при меншому споживанні енергії.

Використання спеціалізованих апаратних прискорювачів може забезпечити виконання певних завдань швидше і енергоефективніше, ніж це було б можливо на звичайному процесорі загального призначення.

Апаратне управління напругою та частотою процесора дозволяє змінювати їх в реальному часі в залежності від обсягу завдань, що забезпечує ефективне використання енергії.

Гнучкість і зручність в розробці виконуються наступним чином. Програмні методи зазвичай дозволяють більшу гнучкість у контролі енергоспоживання, оскільки програмне забезпечення може бути легко змінене і адаптоване під потреби.

Апаратні методи можуть бути менш гнучкими, оскільки вимагають фізичних змін апаратури для зміни споживання енергії.

Отже, програмні методи можуть бути більш гнучкими та зручними в контролі енергоспоживання, але апаратні методи зазвичай є більш ефективними у зниженні споживання енергії завдяки використанню спеціалізованих апаратних компонентів і оптимізованих апаратних структур. При розробці вбудованих процесорних систем важливо обирати підходи, які найкраще відповідають потребам проекту з точки зору енергоефективності, продуктивності та зручності в розробці.

Гнучкість і зручність в розробці - це характеристики, які описують, наскільки легко і ефективно можна розробляти та змінювати систему, включаючи вбудовані процесорні системи, для відповіді на змінні вимоги і потреби. Основна відмінність між гнучкістю і зручністю в розробці полягає у тому, як ці характеристики досягаються в кожному з підходів.

Гнучкість в розробці з програмними методами виконуються наступним чином. Використання модульної структури програмного коду дозволяє легко

розширювати функціональність і змінювати окремі частини програми без впливу на інші [25,26].

Надання добре задокументованого інтерфейсу програмування (API) дозволяє розробникам легко взаємодіяти з функціями і компонентами системи.

Гнучкість в розробці з апаратними методами виконуються наступним чином. Апаратурні компоненти можуть мати різні налаштування та конфігурації, що дозволяє розробникам вибирати оптимальні параметри для конкретних потреб.

Деякі апаратні компоненти можуть бути програмовані або перепрограмовані, що дозволяє змінювати їх поведінку безпосередньо з програмного забезпечення.

Зручність в розробці з програмними методами виконуються наступним чином. Використання високорівневих мов програмування дозволяє розробникам писати код на більш зрозумілому та компактному рівні.

Для програмних методів існує багато доступних розробникам інтегрованих середовищ розробки (IDE), бібліотек та інших інструментів, які полегшують розробку та дебаггінг.

Зручність в розробці з апаратними методами виконуються наступним чином. Апаратні компоненти можуть мати певний набір вбудованих функцій, що спрощує їх інтеграцію та використання.

Існують готові апаратні модулі та плати, які можна використовувати у своїх проектах, що зменшує час і зусилля, витрачені на розробку власних компонентів.

Загальною тенденцією є те, що програмні методи часто забезпечують більшу гнучкість у розробці, оскільки програмне забезпечення може бути змінене і адаптоване за допомогою програмних оновлень. Апаратні методи можуть бути менш гнучкими, але зазвичай надають більшу швидкодію та ефективність у виконанні спеціалізованих операцій. При розробці вбудованих процесорних систем важливо обирати підходи, які найкраще відповідають

потребам проекту з точки зору гнучкості, зручності розробки та продуктивності системи.

Розмір і вартість - це дві взаємопов'язані характеристики, які відображають фізичний розмір і вартість вбудованої процесорної системи. Основна відмінність між розміром і вартістю полягає у тому, як ці характеристики залежать одна від одної в кожному з підходів.

Розмір вбудованої процесорної системи. Програмні методи зазвичай займають менше фізичного простору, оскільки програмне забезпечення може бути реалізоване на існуючому апаратному обладнанні без необхідності виготовлення нових компонентів.

Апаратні методи можуть займати більше фізичного простору, особливо якщо використовуються спеціалізовані апаратні компоненти або прискорювачі.

Вартість вбудованої процесорної системи визначаються наступним чином. Програмні методи зазвичай коштують менше, оскільки вони не потребують додаткових витрат на виготовлення апаратних компонентів.

Апаратні методи можуть бути дорожчими, особливо якщо потрібно виготовлення спеціалізованих апаратних компонентів або модулів.

Програмні методи можуть бути менш продуктивними, оскільки вони виконуються на загальнопризначеному процесорі, який може мати обмежену швидкодію для певних операцій.

Апаратні методи зазвичай надають кращу швидкодію та продуктивність, оскільки вони виконуються на спеціалізованих апаратних пристроях. Програмні методи зазвичай споживають менше енергії, оскільки вони можуть працювати на існуючому апаратному обладнанні, яке може бути енергоефективним.

Апаратні методи можуть бути менш енергоефективними, особливо якщо використовуються спеціалізовані апаратні компоненти з високим енергоспоживанням. Тому, програмні методи зазвичай менш складні у розробці, оскільки розробникам не потрібно виготовляти апаратні компоненти або працювати з електронікою.

Апаратні методи можуть бути більш складними у розробці через потребу проектування і виробництва апаратурних компонентів.

Загальна тенденція полягає в тому, що програмні методи зазвичай забезпечують менший розмір і нижчу вартість, але можуть бути менш продуктивними і менш енергоефективними. Апаратні методи можуть бути більш продуктивними, але зазвичай мають більший розмір і вартість, особливо якщо потрібні спеціалізовані апаратні компоненти. При розробці вбудованих процесорних систем важливо збалансувати розмір, вартість, продуктивність та енергоефективність для досягнення найкращого рішення відповідно до конкретних потреб проекту.

Надійність (Reliability) - це характеристика, яка відображає здатність системи, включаючи вбудовані процесорні системи, виконувати свої функції без збоїв та відмов протягом певного періоду часу і за певних умов. Відмінність надійності між програмними та апаратними методами полягає у тому, як надійність досягається в кожному з наступних підходів.

Надійність з програмними методами. Ретельна перевірка програмного коду на наявність помилок допомагає забезпечити більшу надійність програмного забезпечення. Використання захисних механізмів, таких як перевірки коректності даних і обмеження доступу, може запобігти несправедливим діям інтерфейсів програми. Використання подвійних або потрійних апаратних компонентів дозволяє продовжити роботу системи у разі відмови одного з компонентів.

Використання механізмів автоматичного відновлення дозволяє системі повернутися до нормальної роботи після виникнення відмови.

Зазвичай менш вартісні та менш складні у реалізації, оскільки не потребують додаткових апаратних компонентів і можуть бути забезпечені за допомогою програмних алгоритмів.

Надійність з апаратними методами можуть бути більш вартісні та складні у реалізації через потребу в додаткових апаратних компонентах та механізмах

дублювання. Часто застосовується в менш критичних системах або там, де забезпечення надійності не є критичним завданням.

Застосовується в критичних системах, де забезпечення безперебійної роботи є важливим завданням. Тому, загальною метою надійності є забезпечення безперебійної роботи системи у всіх умовах, зменшення ризику відмов та забезпечення стабільної та надійної роботи протягом тривалого часу. Кожен підхід має свої переваги та обмеження, де при розробці вбудованих процесорних систем важливо збалансувати функціональність, надійність, вартість та складність для досягнення оптимального рішення для конкретного проекту. При цьому, вартість розробки і час впровадження - це дві ключові характеристики процесу створення і впровадження вбудованих процесорних систем. Відмінність між ними полягає у тому, як вони впливають на бюджет і графік розробки проекту.

Вартість розробки охоплює усі затрати, пов'язані з проектуванням, розробкою, тестуванням та документацією вбудованої системи. Цей параметр може бути впливати на вибір методів розробки та обладнання. Наприклад, програмні методи можуть бути менш вартісними, оскільки не потребують додаткових апаратних компонентів.

Час впровадження визначає тривалість всього процесу від початку розробки до запуску системи в експлуатацію. Цей параметр може бути критичним для деяких проектів, особливо якщо необхідно швидко впровадити систему на ринок або виконати проект в обмежений термін.

Загалом, час впровадження може впливати на вартість розробки, і навпаки. Наприклад, якщо потрібно прискорити процес розробки і впровадження, може бути необхідно збільшити кількість залучених ресурсів, що призведе до збільшення вартості.

Вибір підходу до розробки (програмні або апаратні методи) може впливати на час впровадження і вартість. Наприклад, програмні методи можуть прискорити процес розробки, але використання апаратних методів може забезпечити більшу продуктивність і надійність.

При плануванні розробки вбудованих процесорних систем важливо збалансувати вартість і час впровадження, з урахуванням проектних обмежень, бюджету і вимог замовника. Компроміс між вартістю і часом може бути необхідним, щоб досягти успішного завершення проекту і запуску вбудованої системи на ринок.

Програмні та апаратні методи визначення основних характеристик роботи та продуктивності вбудованих процесорних систем відрізняються у своїх підходах та засобах впливу на ці характеристики. Розглянемо відмінності більш ретельно. Програмні методи визначення характеристик базуються на оптимізації програмного забезпечення, що працює на процесорі загального призначення. Використання більш ефективних алгоритмів може знизити час виконання завдань і споживання ресурсів.

Оптимізація алгоритмів - це процес зміни або поліпшення алгоритмів, що використовуються для вирішення певних завдань, з метою зниження часу обчислень, використання менше пам'яті або зниження інших ресурсів, таких як енергія або обчислювальна потужність.

Оптимізація алгоритмів може бути досягнута різними способами. Вибір більш ефективних алгоритмів, які забезпечують той самий результат, але з меншими ресурсами. Наприклад, використання швидкого сортування (наприклад, швидке сортування QuickSort) замість медленого (наприклад, бульбашкове сортування Bubble Sort).

Зменшення обсягу пам'яті, що використовується алгоритмом, або оптимізація роботи з пам'яттю для ефективного використання кеш-пам'яті та мінімізації часу доступу до пам'яті.

Розбиття алгоритму на частини, які можуть бути виконані одночасно на різних обчислювальних ядрах або процесорах, з метою прискорення обчислень.

Апаратна оптимізація полягає в наступному. Використання спеціалізованих апаратних прискорювачів або інструкцій процесора для ефективнішої реалізації певних операцій або алгоритмів.

Використання оптимізованих структур даних або алгоритмів збереження даних для зниження часу доступу або операцій з даними.

Метою оптимізації алгоритмів є забезпечення більшої продуктивності, ефективності та економії ресурсів. Вбудовані процесорні системи, особливо ті, що мають обмежені ресурси, часто потребують оптимізованих алгоритмів для ефективного виконання різноманітних завдань. Оптимізація алгоритмів може бути важливою частиною розробки вбудованих систем, оскільки вона допомагає забезпечити кращу продуктивність і енергоефективність цих систем.

Програмне управління енергією дозволяє знижувати частоту роботи процесора або вимикати неактивні компоненти для зниження споживання енергії. Компроміси між продуктивністю та енергоефективністю полягає в наступному. Під час розробки програмного коду можна вибирати оптимальний баланс між продуктивністю і енергоспоживанням залежно від потреб системи.

Управління енергією - це процес контролю та оптимізації споживання енергії в електронних пристроях і системах з метою забезпечення ефективності, економії енергії і продовження тривалості роботи на одній зарядці акумулятора або зменшення вартості споживаної електроенергії.

Управління енергією може бути реалізовано на різних рівнях, включаючи апаратний рівень, програмний рівень та рівень системи. Основні методи управління енергією включають наступні компоненти.

Сплячий режим (Sleep Mode). Пристрій або система переходить у сплячий режим, коли не використовується активно. Це дозволяє знизити споживання енергії, зберігаючи при цьому стан, що дозволяє відновити роботу в короткий термін.

Зниження напруги та частоти роботи (DVFS - Dynamic Voltage and Frequency Scaling). Деякі процесори можуть змінювати напругу і частоту своєї роботи в залежності від потреби. Зниження частоти і напруги забезпечує зменшення споживання енергії в тих періодах, коли потужність обчислень не є критичною.

Системи можуть адаптивно змінювати своє споживання енергії в залежності від зовнішніх умов, стану зарядження акумуляторів або інших факторів, що впливають на споживання енергії.

Вбудовані сенсори можуть виявляти присутність людини, руху або інших подій, що можуть активувати або дезактивувати пристрої або системи для ефективного управління енергією.

При розробці програмного забезпечення можна використовувати оптимізовані алгоритми для виконання завдань з меншим споживанням енергії.

Управління енергією особливо важливе для вбудованих процесорних систем, особливо для мобільних пристроїв, IoT-пристроїв і систем з живленням від батарей або акумуляторів. Зниження споживання енергії дозволяє збільшити тривалість роботи на одній зарядці, зменшити нагрівання, покращити надійність та знизити вартість енергоспоживання.

Апаратні методи визначення характеристик базуються на фізичних змінах апаратури та оптимізації апаратних компонентів.

Використання спеціалізованих апаратних компонентів і прискорювачів може забезпечити високу швидкодію для виконання певних операцій.

Деякі апаратні методи, такі як динамічке управління напругою та частотою (DVFS), дозволяють змінювати напругу і частоту роботи процесора для досягнення кращої енергоефективності.

Деякі вбудовані процесорні системи мають прискорювачі для спеціалізованих операцій, таких як обробка сигналів або криптографічні операції.

Хоча програмні та апаратні методи можуть впливати на різні аспекти продуктивності і функціональності вбудованих процесорних систем, найкращі результати зазвичай досягаються за допомогою комбінації обох підходів. Успішна розробка вбудованих процесорних систем вимагає вибору підходів, які найкращим чином відповідають потребам конкретного проекту та враховують обмеження бюджету, часу та продуктивності.

Спеціалізовані операції - це операції або завдання, які вимагають виконання конкретних і спеціалізованих обчислень або операцій, що можуть бути важкими або вимагати значних ресурсів, якщо вони виконуються на загальнопризначених процесорах загального використання.

Вбудовані процесорні системи, які мають спеціалізовані операції, зазвичай мають вбудовані апаратні прискорювачі або спеціалізовані функціональні блоки, які можуть виконувати ці операції більш ефективно і швидко. Такі прискорювачі або блоки можуть бути призначені для конкретних завдань, якими є наступні.

Спеціалізовані процесори або прискорювачі можуть виконувати швидку обробку аудіо- або відеосигналів, що знадобиться в апаратах, які працюють зі звуком, зображенням або відео.

Вбудовані процесори можуть мати апаратну підтримку шифрування, розшифрування і підписування даних для забезпечення безпеки і захисту інформації.

Графічні прискорювачі можуть забезпечувати ефективну обробку графічних даних, що становить основу для роботи зі зображеннями та графікою.

В деяких вбудованих системах використовують спеціалізовані пам'ятьові блоки для швидкого доступу до даних або забезпечення максимальної надійності пам'яті.

Прискорювачі можуть забезпечувати обробку аудіо- або відеокодеків для забезпечення відтворення аудіо та відео у вбудованих системах.

Використання спеціалізованих операцій може збільшити продуктивність і надійність вбудованих процесорних систем, оскільки вони можуть бути оптимізовані для конкретних завдань і дозволяють звільнити загальнопризначений процесор від складних обчислень. Це дозволяє системі більш ефективно виконувати специфічні операції, які можуть знадобиться у конкретному застосуванні.

2.4. Запропонований метод підвищення ефективності роботи вбудованих процесорних систем

В даній роботі запропоновано метод підвищення ефективності роботи вбудованих процесорних систем. Підвищення ефективності обробки даних у вбудованих процесорних систем виконується за рахунок зміни загальної архітектури програмно-апаратних засобів яка полягає в наступному. Задана схема архітектури (рис. 2.4) яка має певну структуру, що забезпечує передачу даних між певними вузлами. Такі вузли показані цифрами від 1 до 11.

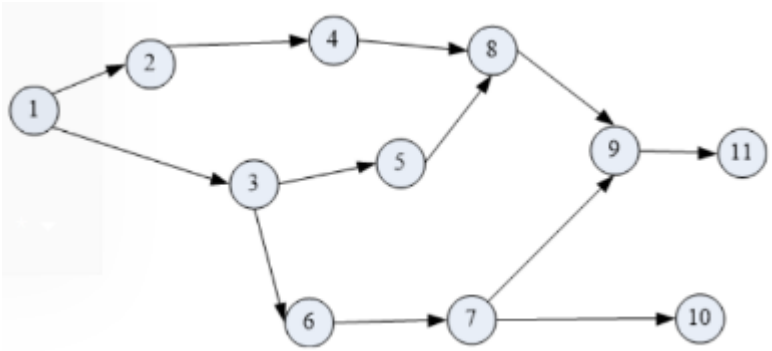


Рисунок 2.4 - Загальна структура програмно-апаратного засобу, що складається з вбудованих процесорів та представлена у вигляді моделі графа

В даній структурі (рис. 2.4) мається один вхід, два виходи та деякі ділянки, в яких можливе паралельне надсилання даних.

Необхідно розробити іншу структуру з обміну даними між вузлами, яка буде більш ефективною, тобто скоротити час надсилання інформацією. Скорочення часу пересилки даних буде відбуватись за оптимізацією шляхів за критерієм зменшення кроків, що показано на рис. 2.5.

Опишемо алгоритм підвищення ефективності роботи вбудованих процесорних систем (рис. 2.5), який включає такі кроки.

Крок 1. Аналіз структури архітектури системи, де визначається графова структура з вузлами, що обмінюються даними.

Крок 2. Створення ефективнішої структури обміну даними, де виконується розробка нової структури обміну даними між вузлами, з упором на скорочення часу передачі інформації. Це може включати оптимізацію шляхів обміну даними.

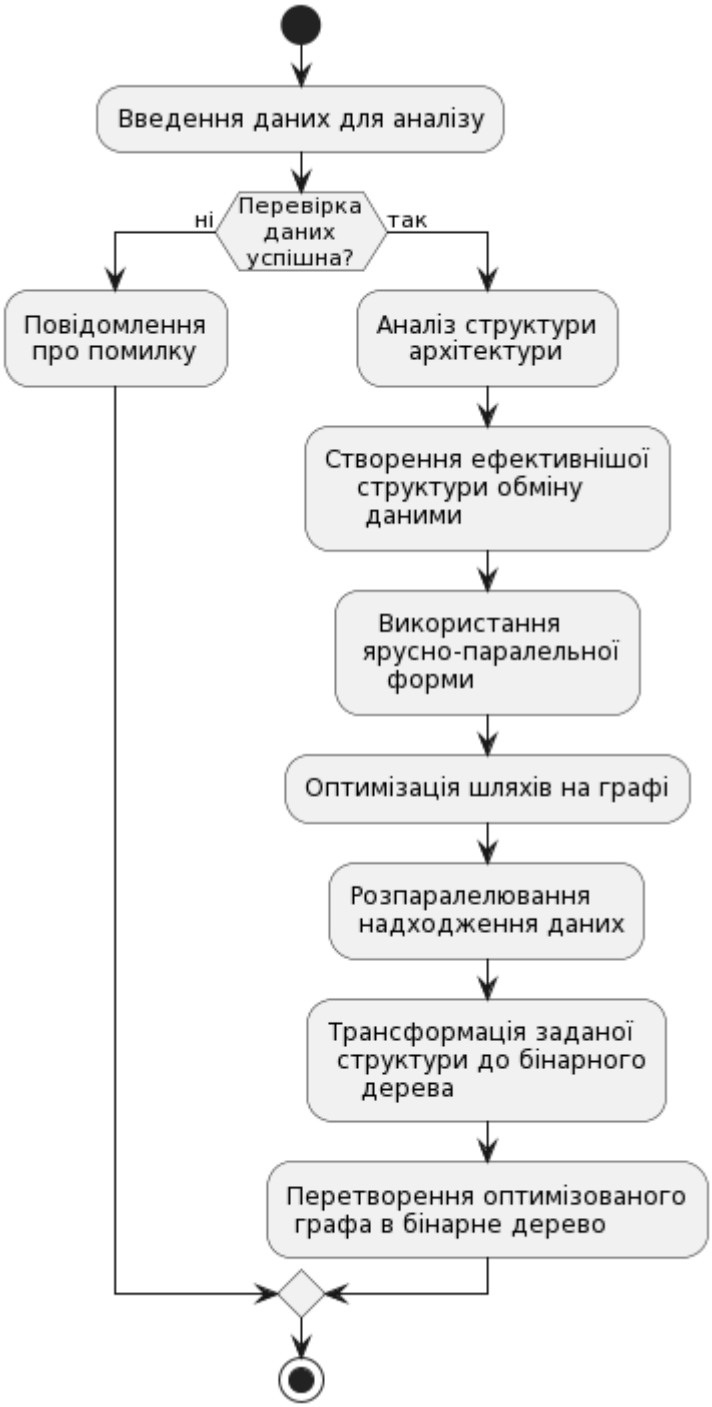


Рисунок 2.5 - UML підвищення ефективності роботи вбудованих процесорних систем

Крок 3. Використання ярусно-паралельної форми, де виконується застосування цієї форми для оптимізації ширини обміну даними, зменшення висоти форми, і використання більше виконавців для оптимізації графа без зміни структури алгоритму.

Крок 4. Оптимізація шляхів на графі ярусно-паралельної форми, де виконується пошук найкоротшого шляху на графі з метою забезпечення максимальної швидкості та максимального використання виконавців.

Крок 5. Розпаралелювання надходження даних, де використовуються методи розпаралелювання для оптимізації обробки даних. Це може призводити до значного покращення ефективності обробки даних.

Крок 6. Трансформація заданої структури до бінарного дерева, де представлення оптимізованої структури програмно-апаратного засобу виконується у вигляді бінарного дерева, що допомагає у візуалізації структури алгоритму та виділенню залежностей.

Крок 7. Перетворення оптимізованого графа в бінарне дерево, де цей крок включає процес вибору корня дерева, а також виконується поділ на поддерева та рекурсивне перетворення до того часу, поки кожна операція не стане вузлом бінарного дерева.

Таким чином, наданий описовий алгоритм спрямований на оптимізацію обробки даних в системах з вбудованими процесорами за допомогою оптимізованих графових структур та розпаралелювання. Розглянемо цей алгоритм більш ретельно.

Перетворення із загальної структура програмно-апаратного засобу, що складається з вбудованих процесорів до ярусно-паралельній форми виконується на основі закону Амдала (2.1). При цьому будь-який набір процесів у вузлах загальної структури використовує кінцеву кількість системних або прикладних ресурсів. Тому, для кожного елемента на множині системних ресурсів на вузлах структури є ряд факторів, що визначають напрямок у виконанні того чи іншого процесу в обробці даних.

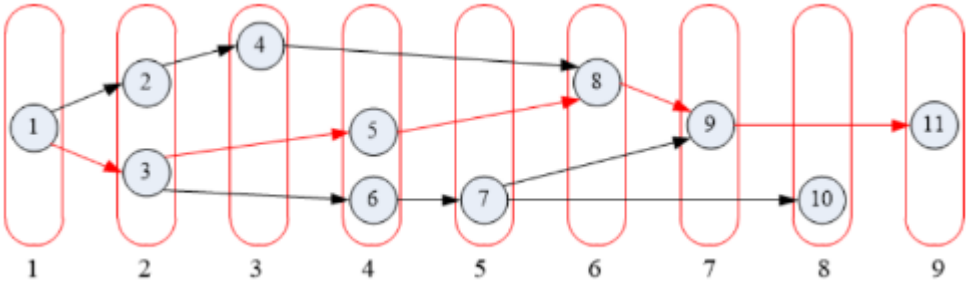


Рисунок 2.6 - Загальна структура програмно-апаратного засобу, що складається з вбудованих процесорів та представлена у ярусно-паралельній формі

Закон Амдала (2.1) стверджує, що існує вигода від використання паралельних обчислень яка багато в чому зумовлена властивостями методів і алгоритмів. Такі властивості використовуються в програмно-апаратних засобах. Тому, максимальне прискорення, яке можна отримати при виконанні завдань на паралельній обчислювальній системі за законом Амдала, розраховується як:

$$\square = \frac{1}{\square + \frac{1 - \square}{\square}}, \tag{2.1}$$

де *S* – коефіцієнт максимального збільшення швидкості роботи вбудованої структури;

– p – частка паралельних вузлів вбудованої структури;

– c – кількість вузлів.

Якщо існує велика кількість вузлів в структурі, то можливе наступне співвідношення (2.2):

$$\square \approx \frac{1}{\square}, \tag{2.2}$$

Формула з моделі закону Амдала (2.1) використовується для оцінки максимально можливого прискорення роботи структури де впроваджуються паралельні процеси з обміну даними. Якщо *p* (частка паралельних операцій) наближається до 1, тоді прискорення *S* також наближається до нескінченності. Це означає, що запровадження паралельних обчислень може призвести до значного прискорення роботи системи.

Однак важливо відзначити, що реальне прискорення ще залежить від багатьох факторів, включаючи структуру алгоритмів, характер завдання,

ефективність паралельної реалізації та інші. Тому, наданих закон Амдала є теоретичною моделлю і може досить точно прогнозувати реальне прискорення структури на вузлах вбудованих процесорів у конкретних сценаріях.

Коротко розглянемо обидва аспекти оптимізації процесу обробки даних з використанням ярусно-паралельної форми (ЯПФ) та розпаралелювання надходження даних.

Оптимізація з використанням ЯПФ є алгоритмом збільшення ширини на ярусно-паралельній формі. Це досягається шляхом використання певного алгоритму, який зменшує висоту форми після розрахунку канонічної висоти ЯПФ. Важливо, що це дозволяє використовувати більше виконавців (вузлів у структурі) у процесі оптимізації форми графа без зміни програмної структури алгоритму.

При цьому використовується метод пошуку найкоротшого шляху, який відбувається наступним чином. Оптимізація включає пошук найкоротшого шляху на графі ЯПФ. В даному випадку це шлях 1-3-5-8-9-11. Цей шлях визначає оптимальний порядок виконання завдань для досягнення максимальної швидкості та максимальної "жадібності" (використання максимальної кількості виконавців).

Наступним аспектом оптимізації процесу обробки даних є розпаралелювання надходження даних, який полягає в наступному.

Перетворення відносин між елементами масиву на графі (рис. 2.6) описує використання перетворень відносини між елементами масиву для розпаралелювання надходження даних. Це призводить до покращення ефективності обробки даних наступним чином. Якщо взяти приклад схематичного зображення перетворення (рис. 2.6), можна побачити як послідовна обробка даних за 9 тактів оптимізується до 6 (рис. 2.7) з допомогою методів розпаралелювання.

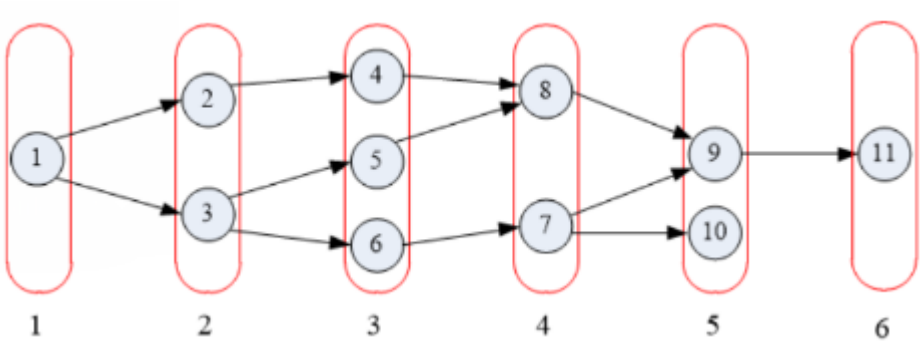


Рисунок 2.7 - Загальна структура програмно-апаратного засобу, яка була оптимізована з 9 (рис. 2.6) до 6

При цьому підвищення ефективності обробки даних відбувається в результаті використання такого розпаралелювання, коли виходить збільшення з 9 до 6, тобто на 33,3%.

Таким чином, обидва ці підходи, ярусно-паралельна форма та розпаралелювання надходження даних, є важливими стратегіями оптимізації для збільшення продуктивності алгоритмів обробки даних у паралельних обчислювальних середовищах.

Зазначимо, що існує перетворення оптимізованої загальної структури програмно-апаратного засобу (рис. 2.7) до бінарного дерева (рис. 2.8).

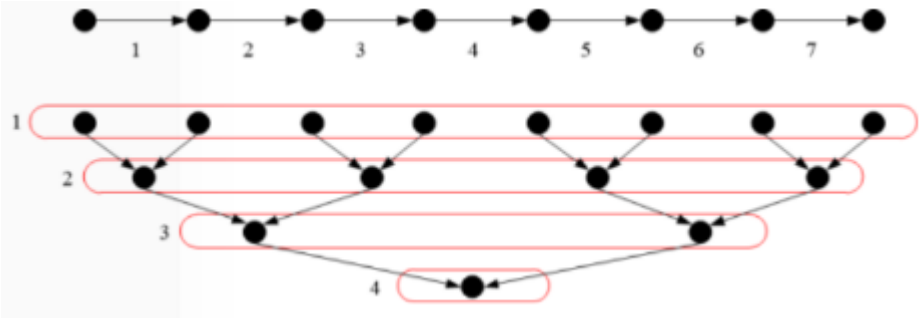


Рисунок 2.8 - Представлення загальної оптимізованої структури програмно-апаратного засобу у вигляді бінарного дерева

Перетворення оптимізованого графа як бінарного дерева зазвичай означає структурне уявлення оптимізованого алгоритму як бінарної деревоподібної структури даних. У контексті оптимізованих алгоритмів таке перетворення може допомогти побачити структуру виконання операцій та взаємодії між ними.

У бінарному дереві кожен вузол представляє операцію чи етап алгоритму, а гілки представляють залежності та порядок виконання цих операцій. Кожен вузол має трохи більше двох нащадків: лівого і правого.

Метод перетворення оптимізованого графа в бінарне дерево може включати наступні кроки.

Вибір кореня дерева полягає у визначенні початкової операції або етапу, який буде коренем бінарного дерева. Це може бути, наприклад, операція з найвищим пріоритетом або операція, з якої починається перетворення до бінарного дерева.

Поділ на поддерева являє собою множину поділу операцій, де в наслідок перетворень залишаються дві групи відповідно до їх порядку виконання. При цьому одна група може представляти операції, які виконуються перед коренем, а інша - після кореня.

Рекурсивне перетворення коли процес рекурсивно повторюється кожної з двох груп, перетворюючи у піддерева. Цей процес виконується до того часу, поки кожна операція стане вузлом бінарного дерева.

Описовий алгоритм методу перетворення оптимізованого графа в бінарне дерево включає наступні кроки.

Крок 1. Вибір кореня дерева, де визначається початкова операція, яка стане коренем бінарного дерева. Це може бути операція з найвищим пріоритетом або та, яка має найбільший вплив на наступні кроки алгоритму.

Крок 2. Поділ на поддерева, де всі операції поділяються на дві групи залежно від їх послідовності виконання. Одна група може містити операції, які відбуваються перед кореневою операцією, а інша - після неї.

Крок 3. Рекурсивне перетворення, де кожен процес повторюється для кожної з двох груп операцій. Кожна з цих груп перетворюється у піддерево. При цьому, операції відбуваються рекурсивно, де кожна операція цієї підструктури стає коренем нового піддерева, яке буде знаходитися на певній глибині в основної гільки дерева.

Крок 4. Завершення алгоритму, де кожен процес перетворення триває до того моменту, поки кожна операція не стане вузлом бінарного дерева, яке описує оптимізовану структуру алгоритму. Цей алгоритм допомогає побудувати структуру виконання операцій та залежності між ними у вигляді бінарного дерева, що сприяє аналізу та подальшій оптимізації.

Таким чином, перетворення оптимізованого графа в бінарне дерево може значно підвищите ефективність (на 33,3%) в галузі обробки даних, допомогти візуалізувати структуру алгоритму, виділити залежності та вузькі місця, що може бути необхідним при аналізі та оптимізації.

2.4. Висновки

В другому розділі показані програмні особливості використання методів вбудованих процесорних систем. Проведено аналіз вказав на важливість прикладних програмних засобів для вивчення та оптимізації вбудованих систем. Використання програмних методів дослідження дозволило ефективно виявляти та вирішувати проблеми на рівні програмного коду та алгоритмів. Проте, важливо враховувати, що програмні методи можуть мати обмеження в тих випадках, коли доступ до внутрішньої роботи системи обмежений або коли потрібно вивчати фізичні аспекти роботи апаратних компонентів.

Розглянуті апаратні особливості використання методів вбудованих процесорних систем показало, що апаратні методи дослідження вбудованих систем є корисними для отримання точних даних про фізичну роботу процесорів та інших апаратних компонентів. Вони надають можливість спостерігати систему в реальному часі та вимірювати параметри, які можуть бути важко отримати програмними методами. Проте, використання апаратних методів може бути дорожчим та більш складним завданням, і вони можуть мати обмеження в тому, які аспекти системи можуть бути вивчені.

За допомогою проведення порівняльної характеристика використання програмних та апаратних методів вбудованих процесорних систем було визначено, що використання програмних та апаратних методів вбудованих

процесорних систем має свої переваги, недоліки та певні та обмеження. Так, програмні методи найбільш підходять для аналізу програмного коду та алгоритмів, тоді як апаратні методи надають можливість вивчати фізичні аспекти роботи системи. Кращий підхід може залежати від конкретної задачі та обмежень дослідження.

Результати аналізу виконаних досліджень показали, що комбінування програмних та апаратних методів може бути найбільш ефективним підходом до вивчення та оптимізації вбудованих процесорних систем. Використання прикладних програмних засобів разом з апаратними засобами дозволяє отримати комплексний аналіз роботи системи, що сприяє підвищенню їхньої ефективності та надійності.

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ТА ВПРОВАДЖЕННЯ ВБУДОВАНИХ ПРОЦЕСОРНИХ СИСТЕМ

3.1. Розробка вимог до основних параметрів роботи вбудованих процесорних систем

Вимоги щодо програмних засобів підтримки вбудованих процесорних систем можуть різнитися в залежності від конкретного типу і призначення вбудованих систем, а також від конкретних вимог проекту (рис. 3.1). Однак існують деякі загальні вимоги та рекомендації, які зазвичай враховуються при розробці програмного забезпечення для вбудованих систем.

Рисунок 3.1 - Вимоги щодо програмних засобів підтримки вбудованих процесорних систем

Вбудовані системи, зазвичай, мають обмежені ресурси, такі як обсяг пам'яті, обчислювальна потужність і енергоспоживання. Тому програмне забезпечення повинно бути оптимізованим щодо використання цих ресурсів.

Ефективність ресурсів в контексті програмних засобів підтримки вбудованих процесорних систем означає здатність програми працювати з обмеженими ресурсами, такими як пам'ять, обчислювальна потужність і енергоспоживання, з максимальною продуктивністю і ефективністю. Основні аспекти ефективності ресурсів включають наступні вимоги.

Таблиця 3.1

Вимоги щодо програмних засобів підтримки вбудованих процесорних систем

Вимога	Опис
Ефективність ресурсів	Програмне забезпечення повинно бути оптимізованим для ефективного використання обмежених ресурсів (пам'яті, обчислювальної потужності).
Відсутність ОС або легковага ОС	Вбудовані системи можуть не використовувати загальні операційні системи через обмежені ресурси; можливе використання RTOS або спеціалізованих ОС.
Відповідність вимогам реального часу	Деякі вбудовані системи мають суворі вимоги до відповідності часовим обмеженням; програмне забезпечення повинно гарантувати їх виконання.
Низьке енергоспоживання	Програмне забезпечення повинно бути розроблене з урахуванням ефективного управління енергоспоживанням та сплячим режимом.
Надійність та стійкість	Вбудоване програмне забезпечення повинно бути надійним і стійким до виникнення помилок, особливо в критичних застосунках.
Можливість оновлення	Деякі вбудовані системи можуть вимагати можливості оновлення програмного забезпечення після релізу.
Підтримка специфічного обладнання	Програмне забезпечення повинно бути здатним взаємодіяти з конкретним обладнанням вбудованої системи, включаючи сенсори і периферійні пристрої.

За оптимізованим використанням пам'яті програмне забезпечення повинно ефективно використовувати доступну пам'ять, не розміщуючи зайву інформацію або структури даних у великих обсягах.

За мінімальним споживанням обчислювальних ресурсів програмне забезпечення повинне виконувати завдання, не витрачаючи зайву обчислювальну потужність, уникати надмірного виконання операцій, які можуть бути зайвими.

За ефективним управлінням енергоспоживанням вбудовані системи, особливо портативні або акумуляторні пристрої, повинні оптимізувати енергоспоживання, увійшовши в режими економії енергії під час простою або неактивного режиму роботи.

За швидкістю та відповідністю часовим обмеженням програмне забезпечення повинно працювати швидко та відповідати встановленим часовим обмеженням, особливо в системах реального часу.

По навантаженню програмне забезпечення, що мінімізує вводи/виводу та мережеве навантаження повинно допомогти зменшити затрати на обчислювальні ресурси.

За ефективним використанням батарей у вбудованих системах з живленням від батарей, програмне забезпечення повинно дбати про збереження заряду та максимальну тривалість роботи.

За мінімальним обсягом коду і даних зменшення обсягу програмного коду і даних, які потрібно зберігати в пам'яті та зменшити вимоги щодо пам'яті, тим самим покращити продуктивність самої системи.

Таким чином, забезпечення ефективності ресурсів є важливим аспектом розробки програмного забезпечення для вбудованих систем, оскільки вони часто працюють в умовах обмежених ресурсів і повинні забезпечувати оптимальну продуктивність і функціональність при цьому.

Багато вбудованих систем не використовують операційні системи загального призначення, такі як Windows чи Linux, через зайвий обсяг ресурсів, які вони витрачають. Замість цього, може використовуватися легковагова

система реального часу (RTOS) або інша спеціалізована операційна система. Тому, "Відсутність операційної системи (ОС) або легковага операційної системи" в контексті вбудованих процесорних систем означає, що програмне забезпечення для таких систем може функціонувати без використання загальної операційної системи, які зазвичай використовуються на персональних комп'ютерах чи серверах. Замість цього, вбудовані системи можуть використовувати спеціалізовану ОС або ж взагалі обходитися без ОС.

Основні аспекти щодо вимоги до "відсутності операційної системи або легковаго ОС" включають наступне.

Директний доступ до апаратного обладнання, що означає програмне забезпечення, яке працює на вбудованих системах без ОС, може мати безпосередній доступ до апаратного обладнання. Це дозволяє більш точно контролювати обладнання та підгоняти роботу системи під конкретні потреби.

Низький обсяг системних ресурсів означає те, що вбудовані системи, як правило, обмежені в пам'яті і обчислювальній потужності. Легковага або спеціалізована ОС може бути менш вимогливою до ресурсів порівняно з загальними операційними системами.

Спеціалізовані рішення у таких систем виконуються таким чином, що у вбудованих системах може вимагатися спеціалізована логіка та функціональність, яку складні ОС можуть бути не здатні забезпечити. Тому, програмне забезпечення може бути розроблене, виходячи з конкретних потреб системи. Низький рівень складності означає те, що вбудовані системи можуть бути більш простими з точки зору програмного забезпечення, оскільки вони не потребують ряду функцій і служб, які зазвичай присутні в загальних операційних системах.

Більший контроль означає те, що розробники програмного забезпечення для вбудованих систем без ОС мають більший контроль над кожним аспектом системи і можуть максимально адаптувати її під конкретні потреби.

Тому, у вбудованих системах відсутність ОС або використання легковаго ОС є стратегічним вибором, який допомагає забезпечити оптимальну

продуктивність, ефективність ресурсів і надійність в рамках конкретних завдань і обмежень даної системи. Деякі вбудовані системи мають суворі вимоги до відповідності часових обмежень. Програмне забезпечення повинно забезпечувати виконання завдань в строк.

Відповідність вимогам реального часу (Real-Time) в контексті програмних засобів підтримки вбудованих процесорних систем означає здатність системи або програмного забезпечення виконувати завдання і реагувати на події в межах встановлених часових обмежень. Основним вимогам реального часу відводиться наступних два типи.

По-перше, це жорсткі (Hard Real-Time) вимоги. В цьому випадку, система або програмне забезпечення повинно завжди виконувати завдання вчасно і точно. Навіть невелике відхилення від часових обмежень може призвести до серйозних наслідків. Прикладами систем, які вимагають жорсткої відповідності вимогам реального часу, є системи керування авіаційними апаратами або системи реакції на медичні показники пацієнтів під час операцій.

По-друге, це м'які (Soft Real-Time) вимоги. У цьому випадку, система або програмне забезпечення також повинно виконувати завдання вчасно, але може приймати незначні відхилення від часових обмежень. Відхилення можуть бути допустимими, але надмірне порушення часових меж може призвести до деградації продуктивності або якості обслуговування. Прикладами систем з м'якими вимогами реального часу є відеоконференції або системи електронної комерції. Для досягнення відповідності вимогам реального часу в програмних засобах підтримки вбудованих систем потрібно виконати наступні дії.

Оптимізація продуктивності, де розробка ефективного коду та алгоритмів для забезпечення мінімального часу виконання завдань і мінімальних затрат обчислювальних ресурсів.

Планування та реалізація пріоритетів, де встановлення пріоритетів завдань і виконання їх у встановленому порядку. Завдання з вищим пріоритетом повинні завжди виконуватися першими.

Відстеження і управління часовими обмеженнями, де ведення обліку часу виконання завдань і забезпечення їх виконання в межах встановлених часових обмежень.

Резервне планування та відновлення, де врахування можливості виникнення відмов або інших непередбачених обставин і встановлення механізмів резервного планування та відновлення роботи.

Використання спеціалізованих апаратних компонентів, які прискорюють обчислення або забезпечують точну обробку часових подій. Тому, відповідність вимогам реального часу є важливим аспектом для вбудованих систем, де збій вчасного виконання завдань може призвести до серйозних наслідків для безпеки та ефективності системи.

У багатьох вбудованих системах ефективне управління енергоспоживанням є критичним. Програмне забезпечення повинно бути розроблене з урахуванням ефективного використання енергії та можливості переходу в сплячий режим.

"Низьке енергоспоживання" в контексті вбудованих процесорних систем означає, що програмне забезпечення та апаратне обладнання розроблені таким чином, щоб споживати якнайменше електроенергії або іншого джерела енергії при роботі. Це є критичним фактором для багатьох вбудованих систем, особливо для портативних пристроїв, систем живлення від батарей, датчиків інтернету речей (IoT) та інших областей, де обмеження енергоспоживання є важливим.

Основні аспекти низького енергоспоживання включають наступні чинники. Програмне забезпечення та апаратне обладнання повинні мати можливість входити в сплячі режими або режими низького споживання, коли вони не використовуються. Це допомагає зберегти енергію в моменти, коли система не активна.

Деякі апаратні компоненти, такі як процесори і сенсори, можуть бути налаштовані на роботу з меншим споживанням енергії при збереженні або відсутності активності. У вбудованих системах зазвичай є спеціалізовані

мікросхеми для управління живленням, які дозволяють ефективно регулювати живлення апаратних компонентів залежно від потреби.

Програмне забезпечення повинно уникати надмірного використання фонових задач, які можуть споживати енергію безперервно. Вони мають бути активовані лише тоді, коли це дійсно необхідно. У вбудованих системах, які використовують комунікацію, передача даних повинна бути оптимізована для мінімізації споживання енергії, наприклад, шляхом оптимізації протоколів передачі даних.

Розробка програмних алгоритмів, які споживають менше енергії при виконанні певних завдань, також важлива для забезпечення низького енергоспоживання. Забезпечення низького енергоспоживання важливо для забезпечення довготривалої роботи вбудованих систем і зменшення залежності від живлення.

Вбудовані системи, зазвичай, використовуються в критичних застосунках, тому програмне забезпечення повинно бути надійним і стійким до виникнення помилок. Тому, "Надійність та стійкість" в контексті програмних засобів підтримки вбудованих процесорних систем означає здатність системи або програмного забезпечення функціонувати безперервно та беззавадно протягом тривалого періоду часу в умовах можливих помилок, відмов апаратного обладнання або інших небажаних ситуацій. Надійність і стійкість є критичними для вбудованих систем, які використовуються в критичних застосунках, таких як медицинська апаратура, авіоніка, автомобільні системи безпеки та інші.

Розробка програмного забезпечення з урахуванням можливих помилок і виправлення їх попередньо або реагування на них в реальному часі. Це може включати в себе валідацію даних, перевірку введених даних, а також заходи безпеки.

Механізми для виявлення помилок, їх відновлення та відновлення нормальної роботи системи після виникнення помилок. Це може включати в себе обробку винятків, контроль цілісності даних та резервне копіювання.

Системи моніторингу та діагностики, які дозволяють виявляти аномалії та відмови та надавати інформацію про стан системи оператору або обслуговуючому персоналу.

Здатність системи продовжувати роботу, навіть коли виникають відмови апаратного обладнання або інших складних ситуацій. Це може включати в себе використання резервних компонентів або запуск резервних копій програмного забезпечення.

Грунтовне тестування та верифікація програмного забезпечення для виявлення та усунення помилок перед впровадженням у виробництво або в операційне середовище. Заходи для захисту системи від зловмисних атак та недобросовісних користувачів, такі як заборона несанкціонованого доступу та шифрування даних.

Можливість регулярного резервного копіювання даних та налаштувань, щоб забезпечити можливість швидкого відновлення в разі втрати даних або відмови системи.

Забезпечення надійності та стійкості є завданням важливим для вбудованих систем, які працюють у вимогливих і критичних застосунках, де відмови можуть мати серйозні наслідки.

В деяких вбудованих системах може бути необхідно оновлювати програмне забезпечення після релізу. Тому слід розглядати можливості вдалого оновлення. Тому, "Можливість оновлення" в контексті програмних засобів підтримки вбудованих процесорних систем означає здатність системи чи програмного забезпечення до приймання та встановлення нових версій або патчів для покращення функціональності, виправлення помилок або підвищення безпеки. Цей аспект надзвичайно важливий для підтримки вбудованих систем в актуальному і надійному стані протягом їх життєвого циклу. Завантаження та встановлення оновлень передбачає те, що система повинна бути здатною завантажувати нові версії програмного забезпечення або патчі з відповідних джерел і автоматично встановлювати їх. Повернення до попередньої версії

означає можливість відмінити або відкатити оновлення в разі виникнення проблем або несумісності з новою версією.

Забезпечення безпеки під час оновлення, включаючи перевірку цілісності та автентифікацію патчів, щоб запобігти введенню шкідливого коду є вкрай необхідною вимогою. Тому, оновлення ядра системи та драйверів надає можливість оновлювати ядро операційної системи та драйвери для підтримки нового апаратного обладнання або покращення продуктивності.

Оновлення конфігурації та налаштувань передбачає змогу змінювати конфігурацію і налаштування системи без необхідності переустановки або переведення системи в режим обслуговування.

Планування та автоматичні оновлення надає можливість запланувати час і дату оновлень, а також встановити автоматичне оновлення, щоб зменшити втрати часу та забезпечити актуальність системи.

Журнал оновлень та історія змін необхідний для ведення журнала оновлень і документування історії змін, щоб мати змогу відстежувати, що було оновлено і коли. Така можливість оновлення дозволяє підтримувати вбудовану систему в актуальному і безпечному стані, забезпечуючи її надійну і довготривалу роботу.

Програмне забезпечення повинно бути здатним взаємодіяти з конкретним обладнанням вбудованої системи, включаючи сенсори, комунікаційні інтерфейси і периферійні пристрої. Тому, "Підтримка специфічного обладнання" в контексті програмних засобів підтримки вбудованих процесорних систем означає можливість програмного забезпечення взаємодіяти з конкретними апаратними пристроями або компонентами, які використовуються в даній вбудованій системі. Це дозволяє забезпечити функціональність, сумісність та оптимальну роботу з цим обладнанням.

Основні аспекти підтримки специфічного обладнання включають наступні дії. Розробку драйверів або програмних компонентів, які дозволяють програмному забезпеченню взаємодіяти з конкретними апаратними пристроями.

Драйвери перетворюють відомості з обладнання на дані та функції, які програма може використовувати.

Підтримку інтерфейсів та API для керування специфічними функціями обладнання. Це може включати в себе регулювання параметрів, контроль стану та взаємодію зі спеціальними функціями обладнання.

Розробку програмного забезпечення з можливостями адаптації до різних версій обладнання, розширень або інших модифікацій.

Оптимізацію продуктивності, де здатність оптимізувати роботу з специфічним обладнанням для досягнення максимальної продуктивності і використання його можливостей є вкрай важливою.

Журнал подій та діагностика повинна надавати можливість відстежувати та аналізувати події та стан специфічного обладнання для виявлення проблем, відмов або аномалій.

Підтримка резервних компонентів полягає у взаємодії з резервними або альтернативними компонентами обладнання у випадку відмови або несправності основного обладнання.

Інтеграція зі стороннім обладнанням забезпечує можливість взаємодії зі сторонніми апаратними пристроями, які можуть бути підключені до вбудованої системи.

Тому, підтримка специфічного обладнання є важливим аспектом розробки вбудованих систем, оскільки вони можуть використовувати різні апаратні компоненти в залежності від конкретних завдань і вимог. Вона допомагає забезпечити сумісність і оптимальну роботу програмного забезпечення з апаратним обладнанням і підвищити функціональність вбудованої системи.

Таким чином, загальні вимоги можуть суттєво різнитися в залежності від конкретного завдання та області вбудованих систем. Розробники програмного забезпечення повинні завжди бути готовими враховувати конкретні вимоги свого проекту та вибирати підходящі технології та інструменти для досягнення поставлених цілей.

3.2. Налаштування основних параметрів роботи вбудованих процесорних систем

Для створення програмної системи обліку основних параметрів роботи вбудованих процесорних систем, можна врахувати наступні властивості параметрів (рис. 3.2):

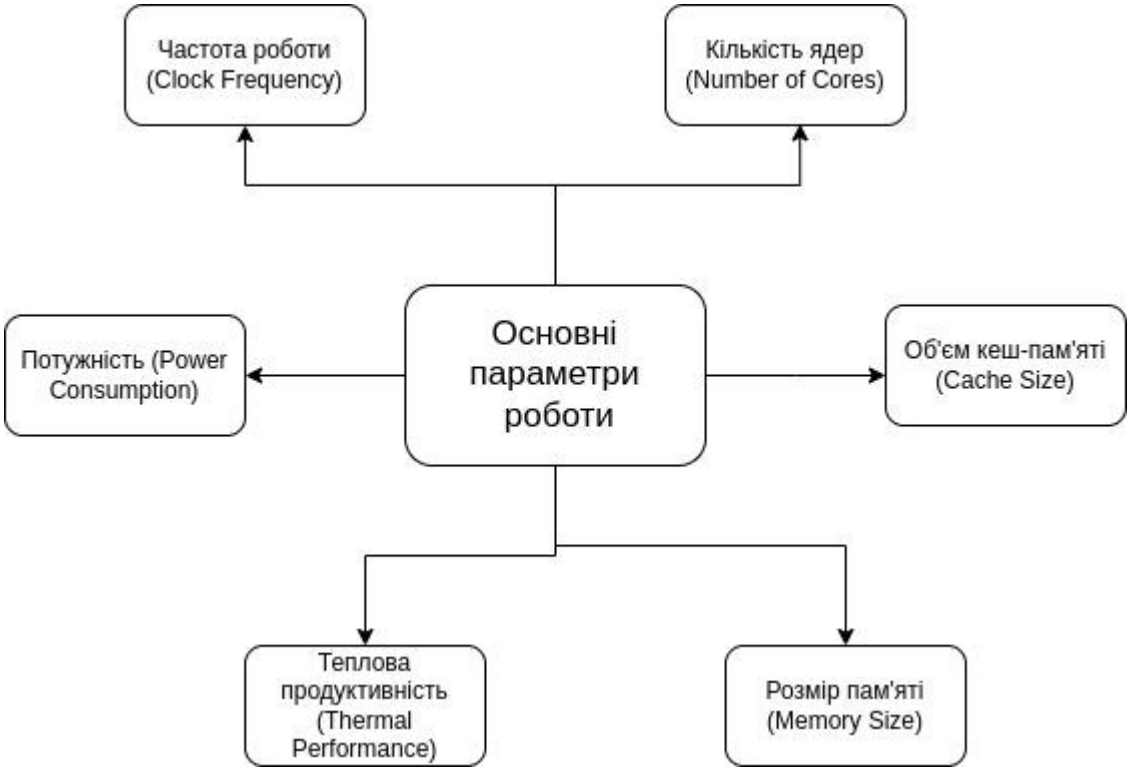


Рисунок 3.2 - Основні параметри роботи вбудованих процесорних систем

Частота роботи (Clock Frequency) процесора визначає, скільки інструкцій може виконатися за одну секунду. Вимірюється у гігагерцах (GHz) або мегагерцах (MHz).

Потужність (Power Consumption) визначає споживання енергії процесором під час роботи, вимірюється у ваттах (W) або міліваттах (mW). Важливо враховувати енергоефективність для продуктивності та тривалості роботи на одній зарядці.

Теплова продуктивність (Thermal Performance) визначає кількість тепла, що виробляється процесором під час роботи, вимірюється у ватах (W). Контроль

теплової продуктивності важливий для забезпечення стабільної температури вбудованої системи.

Розмір вбудованої пам'яті (Memory Size) це об'єм пам'яті, яка використовується для збереження інструкцій та даних. Вимірюється у кілобайтах (KB), мегабайтах (MB) або гігабайтах (GB).

Об'єм кеш-пам'яті (Cache Size) зберігає часто використовувані дані для зменшення часу доступу до пам'яті. Вимірюється у кілобайтах (KB) або мегабайтах (MB).

Кількість ядер (Number of Cores) визначає кількість обчислювальних ядер в процесорі. Багатоядерні процесори можуть виконувати більше завдань одночасно.

Частота роботи (Clock Frequency) вбудованих процесорних систем визначає швидкість, з якою внутрішній годинник (так званий "тактовий генератор") процесора генерує імпульси (такти). Ці такти використовуються для синхронізації роботи всіх операцій і команд в процесорі, а також виконання операцій і інструкцій у пристроях вбудованої системи.

Частота роботи виражається у герцах (Hz) і під час роботи процесора представляє кількість тактових імпульсів, що генеруються протягом одної секунди. Вбудовані процесорні системи, зазвичай, мають частоту роботи в мегагерцах (MHz) або гігагерцах (GHz).

Важливо зазначити, що частота роботи процесора не є єдиною характеристикою, яка визначає продуктивність процесорної системи. Інші фактори, такі як кількість ядер, оптимізація алгоритмів, розмір кеш-пам'яті та інші, також впливають на загальну продуктивність. Однак частота роботи є важливою характеристикою, оскільки вона визначає, наскільки швидко процесор може виконувати операції та обчислення.

Збільшення частоти роботи може підвищити продуктивність процесора, але також може призвести до збільшення енергоспоживання та тепловиділення. Таким чином, при розробці вбудованих систем важливо знайти оптимальний

баланс між частотою роботи, продуктивністю та енергоефективністю в залежності від конкретних потреб і обмежень системи.

Налаштування частоти роботи (Clock Frequency) вбудованих процесорних систем можуть залежати від типу процесора, виробника та можливостей вбудованої системи. Розглянемо деякі з налаштувань частоти роботи, які можуть бути доступні для вбудованих процесорних систем.

Базова частота (Base Clock) полягає в тому, що деякі процесори мають фіксовану базову частоту, яку не можна змінити. Вона визначається виробником і є стандартною частотою роботи для процесора.

Деякі процесори мають функцію Turbo Boost, яка дозволяє автоматично збільшувати частоту роботи при необхідності. При підвищенні вимог до продуктивності процесор може збільшити частоту понад базовий рівень.

Множник частоти (Frequency Multiplier) полягає в тому, що деякі процесори дозволяють змінювати множник частоти, який множиться на базову частоту для отримання кінцевої частоти роботи. Зміна множника дозволяє підвищити частоту роботи над базовим рівнем.

В деяких випадках користувачі можуть виконувати розгон процесора, збільшуючи частоту роботи понад рекомендований виробником максимум. Однак такий режим вимагає додаткової уваги до теплового режиму і може призвести до втрати гарантії. Деякі процесори мають функції управління енергією, які дозволяють автоматично знижувати частоту роботи в періоди малоактивності, зберігаючи енергію.

Динамічна частота роботи (Dynamic Frequency Scaling) полягає в тому, що процесори можуть автоматично змінювати частоту роботи в залежності від завантаження і температури. Це дозволяє зберігати енергію та підтримувати оптимальний тепловий режим.

Конкретні налаштування частоти роботи можуть бути доступні через BIOS або UEFI на рівні материнської плати для деяких систем. У інших випадках, особливо для вбудованих процесорних систем на замовлення, доступ до цих налаштувань може бути обмежений або відсутнім. Важливо пам'ятати, що

неправильне налаштування частоти роботи може призвести до проблем з стабільністю і надійністю системи. Тому, якщо необхідні зміни в налаштуваннях, рекомендується робити їх з обережністю та зрозуміти наслідки цих змін.

Потужність (Power Consumption) вбудованих процесорних систем визначає кількість енергії, яку вони споживають під час роботи або під навантаженням. Це важлива характеристика, яка вказує на електричну потужність, яка використовується для живлення процесора і його внутрішніх компонентів.

Потужність вимірюється в ватах (W) або міліваттах (mW). У вбудованих системах, де часто використовуються енергоефективні процесори, значення потужності може бути досить невеликим, що дозволяє продовжити тривалість роботи пристрою на одній зарядці або батареї.

Потужність є важливим параметром для вбудованих систем з мобільною або обмеженою енергією, таких як смартфони, планшети, IoT-пристрої, безпілотні літальні апарати (дрони) та інші пристрої, які живляться від акумуляторів або батарей. Низьке споживання енергії дозволяє продовжити тривалість роботи на одній зарядці, зменшити нагрівання і покращити продуктивність пристрою.

Також потужність є важливим аспектом для систем, які працюють на віддалених місцях або в обмежених умовах, де забезпечення стійкого живлення може бути важливим фактором.

Виробники процесорів зазвичай надають технічні характеристики, включаючи потужність, для своїх продуктів, що допомагає розробникам вбудованих систем вибирати відповідний процесор для своїх потреб з урахуванням енергоефективності та продуктивності.

Налаштування потужності (Power Consumption) вбудованих процесорних систем можуть варіюватися в залежності від типу процесора та можливостей вбудованої платформи. Ось деякі з налаштувань, які можуть бути доступні для управління потужністю вбудованих процесорних систем:

Тепловий режим (Power Management) полягає в тому, що вбудовані процесори часто мають вбудовані функції управління тепловим режимом, які дозволяють адаптувати потужність в залежності від температури. При перевищенні певної температури може виконуватися автоматичне зниження частоти роботи для зменшення нагрівання.

Dynamic Frequency Scaling (DFS) полягає в тому, що DFS дозволяє динамічно змінювати частоту роботи процесора в залежності від завантаження. При малоактивному стані процесор може працювати на низькій частоті для зниження споживання енергії.

Управління потужністю ядер (Core Power Management) полягає в тому, що деякі процесори дозволяють вимикати або знижувати частоту роботи окремих ядер в залежності від навантаження. Це дозволяє економити енергію, коли не всі ядра потрібні для виконання завдань.

Управління напругою (Voltage Regulation) полягає в тому, що зниження напруги живлення процесора може допомогти знизити його споживання енергії. Деякі процесори можуть автоматично адаптувати напругу в залежності від частоти роботи та навантаження.

У вбудованих системах можуть бути використані функції бюджетування потужності, які дозволяють встановити максимальну допустиму потужність для певних ділянок системи або підсистем. Це дозволяє керувати енергоспоживанням та розподіляти його між різними компонентами.

Управління енергозбереженням (Power Management) полягає в тому, що операційні системи та системний програмний забезпечення можуть мати свої функції управління енергозбереженням, які можуть контролювати частоту роботи та енергоспоживання процесора залежно від активності пристрою.

Ці налаштування можуть бути доступні через системні налаштування, BIOS або UEFI на рівні материнської плати, або на рівні програмного забезпечення системи. Важливо знати, що неправильні налаштування можуть призвести до проблем зі стійкістю, недостатньою продуктивністю або

збільшеним споживанням енергії, тому слід діяти обережно і з урахуванням потреб та обмежень вашої вбудованої системи.

Теплова продуктивність (Thermal Performance) вбудованих процесорних систем визначає, наскільки ефективно процесор може керувати тепловим режимом і видаляти тепло, яке виробляється під час роботи.

Під час функціонування процесор виробляє тепло, оскільки він виконує мільйони обчислень за секунду. Це тепло є результатом електричного опору і призводить до підвищення температури процесора і навколишніх компонентів.

Теплова продуктивність вбудованої системи стає важливим аспектом, особливо в умовах обмеженого простору, де пристрій може бути підвергнутий високим температурам або в зонах, де немає достатньої вентиляції.

Важливо забезпечувати ефективне охолодження процесора, щоб уникнути перегріву, який може призвести до аварійної зупинки пристрою або пошкодження компонентів.

Для підтримання оптимального теплового режиму вбудованих процесорних систем можуть використовуватися різні технології. Розглянемо їх більш ретельно. Збільшення площі поверхні для розсіювання тепла за допомогою радіаторів та вентиляторів. Використання теплових трубок для ефективнішого розподілу тепла із гарячих ділянок до більш прохолодних зон. Використання термічного гелю або пасти для поліпшення теплового контакту між процесором і радіатором.

Вбудовані процесори можуть мати функції управління тепловим режимом, які дозволяють адаптувати частоту роботи та напругу для зниження теплового навантаження. Вимкнення окремих функцій або обмеження продуктивності для зниження енергоспоживання та тепловиділення.

Теплова продуктивність є критично важливою для забезпечення стійкої та надійної роботи вбудованої системи. При проектуванні та впровадженні вбудованих процесорних систем необхідно ретельно враховувати аспекти теплової продуктивності, щоб уникнути проблем з перегрівом та забезпечити довгий термін служби пристрою.

Налаштування теплової продуктивності (Thermal Performance) вбудованих процесорних систем можуть варіюватися в залежності від типу процесора, виробника та підтримки платформи. Ось деякі з налаштувань, які можуть бути доступні для управління тепловим режимом наступним чином.

Теплова радіація (Heat Radiators) та Вентиляція (Ventilation) полягає в тому, що вбудовані системи можуть мати теплові радіатори та вентилятори для ефективного розсіювання тепла. Користувачі можуть налаштувати швидкість вентиляторів або встановити режими роботи, щоб забезпечити оптимальне охолодження.

Термічне управління (Thermal Management) полягає в тому, що деякі процесори мають вбудовані функції управління тепловим режимом, які дозволяють автоматично адаптувати частоту роботи та напругу для зниження теплового навантаження.

Термічна захист (Thermal Throttling) полягає в тому, що вбудовані процесори можуть мати функцію термічного зниження продуктивності (throttling), яка автоматично знижує частоту роботи або обмежує продуктивність, якщо температура перевищує допустимі значення. Це допомагає уникнути перегріву та пошкодження процесора.

Вбудовані системи можуть мати функції управління енергозбереженням, які дозволяють встановити максимальну допустиму потужність для певних ділянок системи або підсистем. Це дозволяє контролювати теплове навантаження і розподіляти його між різними компонентами.

Системи можуть мати сенсори для моніторингу температури процесора та інших компонентів. Користувачі можуть перевіряти ці дані і вживати заходів для забезпечення належного охолодження.

Також, деякі системи можуть підтримувати розширені методи охолодження, такі як водяне охолодження або теплові трубки, що забезпечують високу ефективність охолодження.

В налаштуваннях BIOS або UEFI можуть бути доступні деякі опції для контролю теплової продуктивності, такі як налаштування вентиляторів, вимкнення функцій та інші.

Наявність та доступність цих налаштувань може різнитися для різних процесорних систем і залежати від рівня контролю, який надає виробник процесора або платформи користувачам. Важливо ретельно вивчати документацію та дотримуватися рекомендацій виробника при налаштуванні теплової продуктивності, щоб забезпечити оптимальний тепловий режим і надійну роботу вбудованої системи.

Розмір пам'яті (Memory Size) вбудованих процесорних систем визначає обсяг доступної оперативної пам'яті, яку процесор може використовувати для збереження даних та виконання програм. Оперативна пам'ять (RAM - Random Access Memory) є одним з ключових елементів комп'ютерної системи, який використовується для тимчасового зберігання даних, виконання програм та завдань.

Розмір пам'яті вимірюється в байтах (B), кілобайтах (KB), мегабайтах (MB), гігабайтах (GB) або терабайтах (TB). Зазвичай розмір пам'яті вбудованих процесорних систем є обмеженим порівняно зі стандартними настільними комп'ютерами або серверами, оскільки вбудовані системи, такі як мікроконтролери, системи на кристалі (SoC) та вбудовані системи на замовлення, мають обмежені ресурси і зазвичай оптимізовані для певних завдань або застосувань.

Розмір пам'яті є важливим аспектом при розробці вбудованих систем, оскільки він визначає, скільки даних та програм може бути збережено одночасно в оперативній пам'яті. Занадто обмежений розмір пам'яті може призвести до проблем зі збереженням та виконанням програм, особливо для складних та об'ємних додатків. У той же час, перевищення пам'яті може призвести до збільшеного споживання енергії та зростання вартості системи.

При розробці вбудованих процесорних систем слід ретельно планувати та оптимізувати розмір пам'яті, щоб забезпечити належну продуктивність та

ефективне використання обмежених ресурсів. Це може включати вибір підходящих типів пам'яті, використання стиснення даних або оптимізацію алгоритмів для зменшення обсягу зберіганої інформації.

Налаштування розміру пам'яті (Memory Size) вбудованих процесорних систем можуть бути різними залежно від типу процесора, архітектури, виробника та підтримки платформи. Розглянемо деякі з налаштувань, які можуть бути доступні для управління розміром пам'яті.

Вибір типу пам'яті. Вбудовані системи можуть підтримувати різні типи пам'яті, такі як SRAM (Static RAM), DRAM (Dynamic RAM), NOR Flash, NAND Flash тощо. Вибір пам'яті впливає на швидкодію, доступність та розмір пам'яті.

Кількість пам'яті. Можливість встановлення певної кількості пам'яті вбудованої системи. Виробники процесорів можуть випускати різні версії процесорів з різною кількістю доступної оперативної пам'яті.

Конфігурація пам'яті. Деякі вбудовані процесорні системи дозволяють налаштовувати конфігурацію пам'яті, таку як розподіл на банки, розмір секторів, спосіб адресації тощо.

Налаштування кешування може впливати на розмір кеш-пам'яті, використання рівнів кешування та стратегії кешування.

Деякі вбудовані системи підтримують функції захисту пам'яті, які дозволяють обмежувати доступ до певних ділянок пам'яті для захисту від несанкціонованого доступу. Налаштування відображення адреси пам'яті може впливати на доступ до пам'яті та збереження даних. В деяких системах можуть бути доступні налаштування часових параметрів пам'яті, які впливають на швидкодію доступу до пам'яті.

Доступність цих налаштувань може залежати від конкретної архітектури процесора та його підтримки платформи. Налаштування розміру пам'яті є важливим аспектом при розробці вбудованих систем, оскільки воно може вплинути на продуктивність, стійкість та ефективність використання ресурсів. Розробники повинні уважно планувати розмір пам'яті, щоб задовольнити

потреби своїх додатків та завдань, забезпечити стійку роботу системи та оптимальне використання ресурсів.

Об'єм кеш-пам'яті (Cache Size) вбудованих процесорних систем визначає обсяг доступної кеш-пам'яті, яка використовується для збереження тимчасових копій даних та інструкцій, що недавно були використані процесором.

Кеш-пам'ять є дрібним швидкодіючим буфером між процесором та оперативною пам'яттю (RAM), який забезпечує швидкий доступ до найбільш використовуваних даних та інструкцій. Використання кеш-пам'яті дозволяє знизити час доступу до даних, що може відбуватися значно швидше, ніж при звертанні до великого і повільного обсягу оперативної пам'яті.

Об'єм кеш-пам'яті вимірюється в байтах (B), кілобайтах (KB), мегабайтах (MB) або гігабайтах (GB). Зазвичай вбудовані процесорні системи мають обмежені об'єми кеш-пам'яті порівняно з настільними комп'ютерами або серверами, оскільки вбудовані системи зазвичай працюють з обмеженими ресурсами і оптимізовані для певних завдань або застосувань. Існує кілька рівнів кеш-пам'яті у більшості сучасних процесорів, зазвичай позначених як L1, L2, L3 та ін. Рівень L1 є найшвидший, але найменший, а рівень L3 може бути найбільшим, але менш швидкодіючим порівняно з L1 та L2.

Об'єм кеш-пам'яті є важливим аспектом при розробці вбудованих систем, оскільки він впливає на продуктивність і швидкодію процесора. Для оптимальної продуктивності вбудованих процесорних систем розробники повинні уважно планувати об'єм та організацію кеш-пам'яті, враховуючи особливості своїх додатків та завдань. Зменшення обсягу кеш-пам'яті може призвести до більшого числа промахів кешу та погіршення продуктивності, а перевищення обсягу може призвести до збільшеного споживання енергії та зростання вартості системи.

Налаштування об'єму кеш-пам'яті (Cache Size) вбудованих процесорних систем можуть варіюватися залежно від типу процесора, архітектури, виробника та підтримки платформи. Зазвичай розмір кеш-пам'яті вбудованих процесорних систем визначається виробником процесора і залежить від специфікацій та

обмежень апаратної архітектури. Проте, деякі аспекти, які можуть впливати на налаштування об'єму кеш-пам'яті, включають наступне.

Різні типи вбудованих процесорів можуть мати різний розмір кеш-пам'яті. Наприклад, мікроконтролери можуть мати менший кеш порівняно з вбудованими системами на основі SoC (систем на кристалі).

Деякі архітектури процесорів мають специфічні вимоги до кеш-пам'яті, і виробники можуть змінювати розмір кеш-пам'яті для оптимізації продуктивності та вартості. Різні моделі процесорів можуть мати різний розмір кеш-пам'яті, навіть у межах одного виробника та типу процесора.

Кеш-пам'ять може бути багатоасоціативною (N-way associative), де N вказує на кількість наборів, в яких можуть зберігатися дані. Вибір значення N впливає на розмір та продуктивність кеш-пам'яті.

Деякі процесори можуть підтримувати налаштування розмірів кеш-пам'яті для різних рівнів кешу (наприклад, L1, L2, L3). Кількість та розміри рівнів кешу можуть змінюватися залежно від конкретної реалізації процесора.

Вбудовані процесори можуть підтримувати різні технології кешування, такі як інтелектуальне кешування даних (Data Caching), кешування інструкцій (Instruction Caching), асоціативне кешування (Associative Caching) та інші. Кожна з цих технологій може впливати на розмір та ефективність кеш-пам'яті.

Таким чином, багато з цих налаштувань можуть бути фіксованими і визначеними виробником процесора, тобто не можуть бути змінені користувачем. Розробники вбудованих систем повинні уважно вивчати документацію виробника процесора для зрозуміння обмежень та можливостей стосовно кеш-пам'яті та вибрати відповідний процесор залежно від вимог своїх додатків та завдань.

Кількість ядер (Number of Cores) вбудованих процесорних систем визначає, скільки окремих обчислювальних ядер (процесорних ядер) міститься на одному чипі або в одному процесорі. Кожен окремий ядро може виконувати обчислення та операції незалежно від інших ядер, що дозволяє розпаралелювання задач та підвищення продуктивності відповідно до кількості

ядер. Збільшення кількості ядер у вбудованих процесорних системах дозволяє виконувати наступне. При наявності багатьох ядер система може виконувати кілька задач одночасно, що прискорює обробку даних та виконання операцій. Завдяки розпаралелюванню обчислень можна підвищити загальну продуктивність системи, зокрема для багатопотокових додатків та завдань.

Використання декількох менших ядер може бути енергоефективніше, ніж використання одного великого ядра для виконання всіх операцій. Деякі додатки та завдання можуть бути розділені на паралельні процеси, які можна виконувати одночасно на різних ядрах.

Проте, слід зазначити, що не всі завдання та додатки можуть однаково ефективно використовувати багатоядерні процесори. Деякі додатки можуть бути залежні від одного потоку виконання та не отримувати значного приросту продуктивності від багатоядерної архітектури. Тому розробники вбудованих систем повинні уважно розглянути характеристики своїх додатків та завдань для вибору оптимальної кількості ядер в процесорі. Також, кількість ядер може впливати на енергоспоживання та теплову продуктивність системи, тому також слід ураховувати фактори енергоефективності та охолодження при виборі процесора для вбудованої системи.

Нажаль, налаштування кількості ядер (Number of Cores) вбудованих процесорних систем зазвичай не є доступними для зміни користувачем або розробником. Кількість ядер у вбудованих процесорних системах визначається фізичною архітектурою процесора і зазвичай не може бути змінена після виробництва.

Виробники процесорів випускають різні моделі з різною кількістю ядер, які підходять для різних застосувань і потреб користувачів. Деякі вбудовані процесорні системи можуть мати одне ядро (одноядерні процесори), тоді як інші можуть мати два, чотири або більше ядер (багатоядерні процесори).

Розробники вбудованих систем повинні уважно вибирати процесор з потрібною кількістю ядер, враховуючи характеристики своїх додатків та завдань. Для додатків, які можуть бути розділені на паралельні процеси або вимагають

високої продуктивності, можуть бути корисні багатоядерні процесори. Для менш потужних додатків, які зазвичай працюють з одним потоком виконання, одноядерний процесор може бути достатнім.

Також, існують спеціалізовані процесори, які мають багато ядер і призначені для конкретних завдань, таких як обробка сигналів, машинне навчання, криптографічні операції тощо. Однак, такі спеціалізовані процесори часто мають фіксовану кількість ядер та специфічні характеристики, які підходять лише для певних застосувань.

Таким чином, вибір процесора з певною кількістю ядер є важливим етапом при розробці вбудованих систем, і розробники повинні уважно зрозуміти вимоги своїх додатків та завдань для забезпечення належного рівня продуктивності та ефективного використання ресурсів.

3.3. Налаштування додаткових параметрів роботи вбудованих процесорних систем

Інструкційний набір (Instruction Set) - це множина операцій, які процесор може виконувати. Розмірність інструкційного набору може впливати на продуктивність і зручність програмування (рис. 3.3).

Температурний діапазон (Operating Temperature Range) визначає діапазон температур, при якому процесор може надійно працювати, вимірюється у градусах Цельсія (°C).

Оперативна пам'ять (RAM) - це обсяг пам'яті, доступної для виконання програм та операцій. Вимірюється у кілобайтах (KB), мегабайтах (MB) або гігабайтах (GB).

Консумпція пам'яті (Memory Footprint) - це об'єм пам'яті, яку використовують програми та дані під час роботи.

Джерела живлення (Power Supply) - це тип і параметри джерела живлення, які можуть використовуватися для вбудованих процесорних систем, такі як батареї, акумулятори або зовнішні блоки живлення.

Ці параметри можуть варіюватися в залежності від конкретної вбудованої процесорної системи і її призначення. Під час розробки програмної системи обліку необхідно враховувати ці властивості для ефективного контролю та моніторингу роботи вбудованої системи і оптимізації її характеристик.

Рисунок 3.3 - Додаткові параметри роботи вбудованих процесорних систем

Інструкційний набір (Instruction Set) вбудованих процесорних систем визначає множину команд, які процесор може виконувати. Кожна команда представляється в двійковому коді і виконує певну операцію, таку як арифметичні операції, операції збереження та завантаження даних, управління пам'яттю, логічні операції та інші.

Інструкційний набір є основою архітектури процесора і визначає, які операції можуть бути виконані процесором та яким чином. Від інструкційного набору залежить функціональність та можливості процесора.

Інструкційний набір може бути різним для різних типів процесорів та архітектур. Наприклад, інструкційний набір процесорів x86, ARM та MIPS відрізняється один від одного. Це означає, що програми, написані для одного

типу процесора, можуть не працювати на іншому, якщо вони використовують інструкції, які не підтримуються цим процесором.

Для розробки програмного забезпечення для вбудованих процесорних систем важливо знати інструкційний набір конкретного процесора, оскільки від цього залежить сумісність програм та їх продуктивність. Крім того, інструкційний набір визначає можливості оптимізації програм для отримання кращої продуктивності та ефективності вбудованих процесорних систем.

Налаштування Інструкційного набору (Instruction Set) вбудованих процесорних систем зазвичай не є доступними для зміни користувачем або розробником, оскільки це визначається фізичною архітектурою процесора та мікроархітектурою. Інструкційний набір визначається виробником процесора під час розробки апаратної архітектури, і зазвичай він фіксується і не змінюється після виробництва.

Тип та набір інструкцій залежить від конкретної архітектури процесора, і він є одним із основних чинників, який визначає продуктивність та функціональність процесора. Основні типи інструкційного набору для вбудованих процесорних систем мають наступні структурні властивості.

RISC (Reduced Instruction Set Computer). Архітектура зі скороченим набором інструкцій. Такі процесори мають прості інструкції, але часто здатні виконувати операції відносно швидко. Приклади: ARM, MIPS.

CISC (Complex Instruction Set Computer). Архітектура з багатофункціональним набором інструкцій. Такі процесори мають складні інструкції, що дозволяє зменшити кількість інструкцій, які потрібно виконати для виконання певних завдань. Наприклад, це x86 (Intel, AMD).

VLIW (Very Long Instruction Word). Архітектура, у якій кожна інструкція має декілька операцій, які виконуються паралельно. Це дозволяє розпаралелювати виконання інструкцій та підвищує продуктивність. Наприклад, це Intel Itanium.

DSP (Digital Signal Processor). Спеціалізований тип процесора, який призначений для обробки сигналів, зазвичай з малим інструкційним набором,

оптимізованим для цієї конкретної задачі. Наприклад, це Texas Instruments DSP, Analog Devices Blackfin.

GPU (Graphics Processing Unit). Спеціалізований тип процесора, який призначений для обробки графіки та відео. Інструкційний набір включає операції, які дозволяють швидку обробку графіки. Наприклад, це NVIDIA GeForce, AMD Radeon.

Для розробки програмного забезпечення для вбудованих процесорних систем важливо знати інструкційний набір конкретного процесора, оскільки це визначає сумісність програм та продуктивність системи. Розробники повинні вибирати процесори з інструкційним набором, який найкраще підходить для їхніх додатків та завдань.

Температурний діапазон (Operating Temperature Range) вбудованих процесорних систем визначає межі температур, при яких процесор може нормально функціонувати без перевантаження або пошкодження. Цей діапазон вказує на температури, які допустимі для безпечної та надійної роботи процесора.

Зазвичай температурний діапазон складається з двох значень: мінімальної і максимальної робочої температури. Мінімальна робоча температура - це найнижча температура, при якій процесор може нормально функціонувати. Вище цієї температури процесор може не запускатися або працювати ненадійно.

Максимальна робоча температура - це найвища температура, при якій процесор може нормально працювати без ризику пошкодження. Вище цієї температури можуть виникати проблеми з перегрівом, що може призвести до аварійного відключення процесора або збільшити ризик його пошкодження.

Температурний діапазон зазвичай вказується в градусах Цельсія (°C) і залежить від конкретного процесора та його виробника. Різні процесори можуть мати різні температурні діапазони в залежності від їхньої архітектури, використаної технології, додаткових функцій та призначення.

Вбудовані процесорні системи, які працюють у важких умовах або в екстремальних середовищах, можуть потребувати спеціальної температурної

стійкості або охолодження. Наприклад, процесори, які використовуються в автомобілях, літаках або промислових системах, можуть бути призначені для роботи в широких температурних діапазонах, щоб забезпечити надійну роботу в різних умовах.

Дотримання температурного діапазону є важливим для забезпечення стабільності та надійності роботи вбудованих процесорних систем і попередження можливих несправностей, що можуть виникнути внаслідок перегріву або переохолодження процесора.

Налаштування температурного діапазону (Operating Temperature Range) вбудованих процесорних систем не є параметром, який може бути змінений користувачем або розробником. Температурний діапазон визначається виробником процесора під час його проектування та виробництва і зазвичай є фіксованим значенням для конкретної моделі процесора.

Кожен процесор має свій власний температурний діапазон, який залежить від фізичних характеристик і технології виготовлення чипу, а також від призначення процесора. Такі параметри, як теплова стійкість матеріалів, охолодження, технологія виготовлення (наприклад, 14 нм, 10 нм тощо), визначають максимальні та мінімальні температури, при яких процесор може надійно працювати.

Типовий температурний діапазон для вбудованих процесорних систем може бути, наприклад, від -40°C до +85°C. Однак, існують спеціалізовані процесори для окремих застосувань, які можуть мати розширений температурний діапазон для роботи в більш екстремальних умовах.

Виробники процесорів зазвичай вказують температурний діапазон у технічних характеристиках продукту. При використанні процесора вбудованих систем, розробники повинні дотримуватися вказаних меж температур, щоб забезпечити стабільну та надійну роботу системи, а також для забезпечення довгого терміну служби процесора.

Оперативна пам'ять (RAM) вбудованих процесорних систем - це тип пам'яті комп'ютера, який використовується для тимчасового зберігання даних та

програм під час їх обробки. RAM є одним із ключових компонентів комп'ютерної системи, в якому зберігаються дані, з якими працює процесор, для швидкого доступу та обробки.

Оперативна пам'ять є відмінною від постійної пам'яті, такої як жорсткий диск або флеш-пам'ять, оскільки вона є летючою пам'яттю, що втрачає дані при вимкненні живлення. Тобто, коли комп'ютер вимикається, дані в RAM стираються. Це дозволяє швидко записувати та зчитувати дані, але вимагає постійного живлення для підтримки інформації. Оперативна пам'ять дозволяє процесору швидко звертатися до потрібних даних та виконувати програми. Коли ви відкриваєте програму або документ на комп'ютері, дані з диска завантажуються в RAM для подальшої обробки процесором. Чим більший обсяг оперативної пам'яті, тим більше даних та програм можна зберігати в RAM, що забезпечує більш високу продуктивність системи.

Вбудовані процесорні системи, зазвичай, мають обмежений об'єм оперативної пам'яті у порівнянні зі стандартними комп'ютерами. Оскільки вбудовані системи призначені для специфічних завдань і зазвичай працюють в обмеженому середовищі, вони часто використовують оптимізовані рішення з низьким споживанням енергії та компактними розмірами.

Використання оперативної пам'яті у вбудованих процесорних системах вимагає уважного управління ресурсами та оптимізації програмного забезпечення для ефективного використання обмежених ресурсів. Відповідно до цього, розробники повинні ретельно розраховувати обсяг RAM, необхідний для оптимальної роботи їхніх вбудованих систем.

Налаштування оперативної пам'яті (RAM) вбудованих процесорних систем можуть залежати від конкретного процесора, архітектури системи та її призначення. Зазвичай, налаштунки оперативної пам'яті включають такі параметри. Об'єм пам'яті визначає кількість фізичної оперативної пам'яті, яка доступна вбудованій системі. Вбудовані системи можуть мати обмежений обсяг RAM, зазвичай від кількох мегабайт до декількох гігабайт.

Тип пам'яті визначає технологію пам'яті, яка використовується в системі. Наприклад, DDR3, DDR4, LPDDR, SDRAM та інші.

Тактова частота (Clock Speed) - це швидкість роботи оперативної пам'яті і вимірюється в мегагерцах (МГц) або гігагерцах (ГГц). Вища тактова частота дозволяє пам'яті працювати швидше, але може вимагати більше енергії.

Таймінги (Timings) - це параметри, які визначають затримки при доступі до пам'яті. Найважливіші з них - CAS (Column Access Strobe) latency, RAS (Row Access Strobe) latency та інші. Оптимальні значення таймінгів допомагають покращити продуктивність пам'яті.

В деяких системах можуть бути підтримані різні режими роботи пам'яті, наприклад, двоканальний режим, трьохканальний режим тощо. Ці режими дозволяють використовувати кілька каналів пам'яті для підвищення пропускної здатності. Вбудовані системи можуть підтримувати різні рівні напруги для економії енергії або для підвищення стійкості системи.

Вбудовані системи зазвичай мають менші можливості по налаштуванню оперативної пам'яті, ніж загальнопризначені комп'ютери, оскільки вони часто використовують оптимізовані інтегральні рішення, а не стандартні десктопні компоненти. Виробники вбудованих процесорних систем зазвичай надають документацію з рекомендаціями та обмеженнями на Налаштування пам'яті, щоб забезпечити оптимальну працездатність та стабільність системи.

Консумпція пам'яті (Memory Footprint) вбудованих процесорних систем означає обсяг оперативної пам'яті, який займає програмне забезпечення, включаючи операційну систему, додатки та інші компоненти, під час їх виконання. Це вимірюється в байтах або кілобайтах, і вказує на обсяг пам'яті, який необхідний для коректної роботи вбудованої системи під час виконання програм та завдань.

Консумпція пам'яті є важливим фактором у вбудованих системах, особливо тих, які мають обмежені ресурси, такі як малий обсяг оперативної пам'яті. Такі системи можуть використовувати спеціальні оптимізації та

компроміси, щоб зменшити споживання пам'яті та забезпечити ефективне використання обмежених ресурсів.

Деякі підходи до зменшення консумпції пам'яті вбудованими системами включають наступні параметри.

Мінімізація розміру коду. Використання компактного коду та оптимізація програмного забезпечення для зменшення кількості інструкцій та даних, які необхідно зберігати в пам'яті.

Використання стиснутих форматів даних. Застосування стиснення для збереження даних у більш компактному вигляді, що дозволяє економити пам'ять.

Мінімізація функціональності. Обмеження функціональності програми або операційної системи для зменшення використаної пам'яті.

Використання спеціалізованих бібліотек. Замість стандартних бібліотек можна використовувати спеціалізовані, які мають менший розмір і більш адаптовані під вимоги вбудованої системи.

Динамічне управління пам'яттю. Звільнення пам'яті, яка вже не потрібна, а також динамічне розподілення пам'яті, може допомогти зменшити консумпцію пам'яті вбудованою системою.

Зниження консумпції пам'яті дозволяє забезпечити оптимальну роботу вбудованої системи з обмеженими ресурсами та забезпечити ефективне використання доступної оперативної пам'яті.

Налаштування консумпції пам'яті (Memory Footprint) вбудованих процесорних систем зазвичай є результатом оптимізації програмного забезпечення та вибору відповідних компонентів для системи. Враховуються такі параметри.

Розмір операційної системи. Вибір операційної системи може значно впливати на консумпцію пам'яті. Існують спеціалізовані легковагові операційні системи, які мають менший обсяг пам'яті, ніж загальнопризначені ОС. Вибір оптимальної ОС залежить від вимог та завдань системи.

Вибір компонентів. Вбудовані процесорні системи можуть використовувати спеціалізовані мікроконтролери, системи на кристалах (SoC) або інші компоненти, які оптимізовані для мінімізації консумпції пам'яті.

Мінімізація функціональності. Відмова від непотрібних функцій та модулів програмного забезпечення дозволяє знизити консумпцію пам'яті. Вбудовані системи можуть працювати з обмеженим функціоналом, але забезпечувати виконання важливих завдань.

Оптимізація програмного коду. Використання оптимальних алгоритмів, стиснутих форматів даних, ефективного управління пам'яттю та мінімізація використання змінних та констант допомагають знизити консумпцію пам'яті.

Динамічне управління пам'яттю. Використання динамічного розподілу пам'яті, такого як підтримка вбудованими системами механізмів керування стеком та кучею, дозволяє ефективно розподіляти та звільняти пам'ять.

Ефективне використання кеш-пам'яті може знизити навантаження на оперативну пам'ять і підвищити продуктивність системи. Використання алгоритмів стиснення даних дозволяє зменшити обсяг пам'яті, що потрібний для зберігання даних.

Для досягнення оптимальної консумпції пам'яті, розробники вбудованих систем повинні ретельно вивчити вимоги та специфіку своєї системи, вибрати підходящі компоненти та програмне забезпечення, а також використовувати оптимізаційні техніки для досягнення ефективного використання оперативної пам'яті.

Джерела живлення (Power Supply) вбудованих процесорних систем відносяться до компонентів, які забезпечують електричне живлення всіх елементів системи, зокрема процесорів, пам'яті, периферійних пристроїв і т.д. Це можуть бути зовнішні блоки живлення або внутрішні джерела живлення, в залежності від конструкції і вимог вбудованої системи.

Джерела живлення забезпечують стабільний та надійний постачання електроенергії всім елементам системи, для того, щоб вони могли працювати належним чином. Оскільки вбудовані системи можуть застосовуватися в різних

умовах, джерела живлення можуть бути розроблені з урахуванням різноманітних особливостей, зокрема такі.

Стабільність напруги. Джерела живлення повинні забезпечувати стабільний рівень напруги та струму, особливо для таких елементів, як процесори, які вимагають точного живлення.

Ефективність. Оптимізація джерел живлення дозволяє знизити втрати електроенергії та підвищити ефективність системи в цілому.

Захист від впливу різних зовнішніх факторів. Джерела живлення можуть мати захист від перенапруги, пониження напруги, перенапруги тощо, щоб захистити вбудовану систему від можливих пошкоджень.

Джерела живлення можуть бути спроектовані для роботи в широкому діапазоні вхідних напруг, щоб забезпечити сумісність з різними джерелами живлення (наприклад, батареї, сонячні панелі, джерела змінного струму тощо). Джерела живлення можуть бути розроблені з урахуванням обмеженого простору вбудованої системи і з використанням енергоефективних технологій.

Кожна вбудована система має свої вимоги до джерел живлення, тому виробники повинні враховувати характеристики системи і умови експлуатації для вибору оптимального джерела живлення. Успішний вибір джерела живлення є критичним аспектом для надійної та стабільної роботи вбудованих процесорних систем.

Налаштування джерела живлення (Power Supply) вбудованих процесорних систем можуть залежати від типу системи, її призначення та вимог. Основні налаштування, які можуть бути доступні, включають такі.

Напруга живлення (Supply Voltage). Джерела живлення можуть підтримувати різні рівні напруги в залежності від вимог процесорів і інших компонентів системи. Зменшення напруги може допомогти знизити споживану потужність та підвищити енергоефективність системи.

Максимальна вихідна потужність (Maximum Power Output) - це максимальна потужність, яку може постачати джерело живлення. Вибір джерела

живлення з підходящою вихідною потужністю залежить від загальної потужності системи та її резервування.

Вбудовані системи, особливо ті, що працюють в обмежених умовах, можуть вимагати високої ефективності джерела живлення для економії енергії. Вища ефективність дозволяє знизити втрати енергії і підвищити тривалість роботи від батарей або акумуляторів.

Джерела живлення можуть мати вбудовані захисти від короткого замикання, перенапруги, заміщення, перевищення струму тощо, щоб захистити систему від можливих пошкоджень. Деякі джерела живлення можуть мати інтерфейси комунікації, такі як I2C або SMBus, які дозволяють зчитувати статус та налаштовувати параметри живлення з процесора або інших компонентів системи. Деякі джерела живлення можуть включати термічний контроль для автоматичного регулювання потужності або вимикання в разі перегріву.

Джерела живлення можуть бути спроектовані для роботи в широкому діапазоні вхідних напруг, що дозволяє їх використання з різними джерелами живлення. Джерела живлення можуть мати різний розмір та форм-фактор, що важливо для інтеграції вбудованих систем.

Налаштування джерела живлення можуть бути встановлені за допомогою фізичних перемикачів або за допомогою програмного забезпечення через вбудований інтерфейс, який дозволяє керувати параметрами живлення з програмного рівня. Вибір оптимальних налаштувань залежить від вимог і специфіки конкретної вбудованої системи.

3.4. Висновки

В третьому розділу проводилась розробка вимог щодо основних параметрів роботи вбудованих процесорних систем, де були розглянуті ключові параметри. Визначення таких параметрів вбудованих процесорних систем, як продуктивність, витрати енергії, надійність та інші, є важливим кроком у створенні ефективних вбудованих систем. При цьому, визначені вимоги відповідають потребам розробки програмного засобу.

Процес налаштування основних параметрів роботи вбудованих процесорних систем включала в себе вибір конфігурації процесорів, пам'яті, та інших апаратних компонентів, а також оптимізацію параметрів для досягнення найкращих результатів у визначених вимогах.

Також, в цьому розділі досліджувався процес налаштування додаткових параметрів роботи вбудованих процесорних систем, таких як взаємодія з іншими системами, використання спеціалізованих датчиків, зберігання даних та інші функції. При цьому було відзначено важливість врахування цих параметрів для створення вбудованих систем, які відповідають потребам конкретних застосувань під час розробки програмного засобу.

Таким чином, результати розділу проектування та впровадження вбудованих процесорних систем підкреслювали важливість ретельного розробки вимог та налаштування параметрів для створення високоефективних та надійних систем. Цей процес вимагав глибокого розуміння специфіки конкретних застосувань, вдосконалення підходів до проектування вбудованих процесорних систем та використання методів підвищення ефективності роботи вбудованих процесорних систем, що були впроваджені у програмному засобі.

РОЗДІЛ 4.
ЕКОНОМІЧНИЙ РОЗДІЛ

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1. Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем» є оцінювання науково-технічного рівня та

рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 **[Козловський, Лесько, Кавецький]**.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування	Потрібні незначні фінансові ресурси. Джерела фінансування	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовують
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	4	3
2. Ринкові переваги (наявність аналогів)	4	3	4
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	4	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	39	39	39
Середньоарифметична сума балів $СБ_c$	39		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 **[Козловський, Лесько, Кавецький]**.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмних засобів аналізу відеоадаптерів на робочих станціях» становить 39 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

4.2 Розрахунок витрат на розробку методів та засобів обробки масивів даних з використанням мікросервісів

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[Козловський, Лесько, Кавецький]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.1)$$

4.3 Розрахунок економічного ефекту від можливого впровадження розроблених методів та засобів обробки масивів даних з використанням мікросервісів

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = 22000{,}00 \cdot 63 / 21 = 66000 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	22000	1047,62	63	66000,00
Інженер-розробник програмного забезпечення	18000	857,14	58	49714,29
Консультант	20000	952,38	40,00	38095,24
Всього				153809,52

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{4.2}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \tag{4.3}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) **[Козловський, Лесько, Кавецький]**;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 21 день;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 6700,00 \cdot 1,5 \cdot 1,65 / (21 \cdot 8) = 98,71$ грн.

$З_{р1} = 98,71 \cdot 8,00 = 789,64$грн.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Первинне налаштування робочих станцій	8	4	1,5	98,71	789,64
Налаштування локальної мережі для обладнання	10	4	1,5	98,71	987,05
Всього					1776,70

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод} = (153809,52+ 1776,70) \cdot 12 / 100\% = 18670,35$ грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн = (153809,52+ 1776,70+ 18670,35) \cdot 22 / 100\% = 38336,44$ грн.

4.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці,

які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (4.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 3 \cdot 180{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 660{,}0$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір А4 MAESTRO 80г/м2 500арк	180	3	0	0	660
Папір для записів Economix	110	1	0	0	121
Настільний набір Buromax 16 предметів	215	2	0	0	473
Картридж для принтера Canon LBP6500	1300	1	0	0	1430
Flesh-пам'ять Kingston 16 GB	130	1	0	0	143
Всього					2761

4.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем» відсутні.

4.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \tag{4.7}$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ *= 25000· 3 · 1,1 = 82500 грн.*

Отримані результати зведемо до таблиці.

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Робоча станція (ПК)	3	25000	82500
Маршрутизатор	1	2300	2530
Відеокамера Logitech C920	1	4000	4400
Термокамера Fluke	1	4100	4510
Мікроскоп AmScope B120C	1	3800	4180
Всього			85030

4.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *12000,00 · 3 · 1,12 = 40320 грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Операційна система Microsoft Windows 10	3	12000	40320
Система розробки Project Rider	1	5400	6048
Прикладний пакет Microsoft Office 2019	1	5230	5857,6
Prime95	1	7200	8064
Aida64	1	3700	4144
Всього			58385,6

4.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (82500{,}00 \cdot 3) / (5 \cdot 12) =$ 4125,00 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Робоча станція (ПК)	82500	5	3	4125,00
Маршрутизатор	2530	3	3	210,83
Оргтехніка	6000	5	2	200,00
Приміщення лабораторії	212000	20	3	2650,00
ОС Windows 11	40320	3	3	3360,00
Система розробки Project Rider	4144	2	2	345,33
Prime95	8064	3	2	448,00
Aida64	4144	3	2	230,22
Всього				11569,39

4.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \qquad (4.10)$$

де $W_{уі}$ – встановлена потужність обладнання на визначеному етапі розробки, кВт;

$t_і$ – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 7,50 грн;

$K_{впі}$ – коефіцієнт, що враховує використання потужності, $K_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i<1$.

$B_e = 0,5 \cdot 504,0 \cdot 7,50 \cdot 0,95 / 0,97 = 1851,03$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Робоча станція (ПК)	0,5	504	1851,03
Роутер	0,02	504	74,04
Сервер	0,7	180	925,52
Робоче місце дослідника	0,15	504	555,31
Оргтехніка	0,45	20	66,11
Всього			3472,01

4.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (4.11)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св} = (153809,52 + 1776,70) \cdot 20 / 100\% = 31117,24$ *грн.*

4.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (1.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (153809,52+ 1776,70) · 30 / 100% = 46675,87 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (4.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (153809,52+ 1776,70) · 50 / 100% = 77793,11 грн.

4.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$B_{нзв} = (153809,52 + 1776,70) \cdot 120 / 100\% = 186\,703,46$ грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.15)$$

$B_{заг} = 716100,69$ грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

$ЗВ = 716100,69 / 0,9 = 795667,43$грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 45000 користувачів;

2-й рік – 50000 користувачів;

3-й рік – 35000 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 280000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 1850 грн за ліцензію на один комп'ютер;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 100 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою **[Козловський, Лесько, Кавецький]:**

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (280000{,}00 \cdot 100{,}00 + 1950{,}00 \cdot 45000) \cdot 0{,}83 \cdot 0{,}3 \cdot (1\text{-}0{,}18/100\%) =$ 23719258,5 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (280000{,}00 \cdot 100{,}00 + 1950{,}00 \cdot (45000 + 50000)) \cdot 0{,}83 \cdot 0{,}3 \cdot (1 - 0{,}18/100\%) = 43698763{,}5$грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (280000{,}00 \cdot 100{,}00 + 1950{,}00 \cdot (45000 + 50000 + 35000)) \cdot 0{,}83 \cdot 0{,}3 \cdot (1 - 0{,}18/100\%) =$ 57684417 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 23719258{,}5/(1+0{,}25)^1 + 43698763{,}5/(1+0{,}25)^2 + 57684417/(1+0{,}25)^3 = 76477036{,}94$$

грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 709229,86 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 795667,43= 1591334,866 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 76477036,94грн;

PV – теперішня вартість початкових інвестицій, 1591334,866грн.

$E_{абс} = ПП - PV$ = 76477036,94- 1591334,866= 74885702,08 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 74885702,08 грн;

PV – теперішня вартість початкових інвестицій, 1591334,866грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+74885702{,}08/1591334{,}866)^{1/3} = 2{,}64.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.22}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,11+0,18 = 0,29 < 2,64 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.23}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,64 = 0,38 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.4. Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем» становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

Також термін окупності становить 0,38 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методу та програмних засобів підвищення ефективності роботи вбудованих процесорних систем».

ВИСНОВКИ

В першому розділі магістерській роботі було проведено аналіз методів роботи вбудованих процесорних систем, який включав в себе вивчення прикладних програмних засобів, програмних методів та апаратних методів дослідження цих систем. За допомогою прикладних програмних засобів було відстежено та проаналізована робота вбудованих процесорних систем на рівні програмного забезпечення. Це дозволило виявляти проблеми та здійснювати оптимізацію на рівні алгоритмів та програмного коду.

Деякі програмні методи дослідження роботи вбудованих систем включали в себе моделювання та симуляцію їхньої роботи. Ці методи дозволяли аналізувати програмний засіб в різних умовах.

Апаратні методи дослідження роботи вбудованих систем включали в себе використання спеціалізованого обладнання для моніторингу та аналізу роботи різних систем. Ці методи дозволяють отримувати точні дані про фізичну роботу процесорів та інших апаратних компонентів системи.

Необхідно відзначити, що комбінування різних методів аналізу є найбільш ефективним підходом до вивчення та оптимізації роботи вбудованих процесорних систем. Використання прикладних програмних та апаратних засобів разом з різними методами дослідження дозволяє підвищити ефективність роботи систем в цілому.

В другому розділі показані програмні особливості використання методів вбудованих процесорних систем. Проведено аналіз вказав на важливість прикладних програмних засобів для вивчення та оптимізації вбудованих систем. Використання програмних методів дослідження дозволило ефективно виявляти та вирішувати проблеми на рівні програмного коду та алгоритмів. Проте, важливо враховувати, що програмні методи можуть мати обмеження в тих випадках, коли доступ до внутрішньої роботи системи обмежений або коли потрібно вивчати фізичні аспекти роботи апаратних компонентів.

Розглянуті апаратні особливості використання методів вбудованих процесорних систем показало, що апаратні методи дослідження вбудованих

систем є корисними для отримання точних даних про фізичну роботу процесорів та інших апаратних компонентів. Вони надають можливість спостерігати систему в реальному часі та вимірювати параметри, які можуть бути важко отримати програмними методами. Проте, використання апаратних методів може бути дорожчим та більш складним завданням, і вони можуть мати обмеження в тому, які аспекти системи можуть бути вивчені.

За допомогою проведення порівняльної характеристика використання програмних та апаратних методів вбудованих процесорних систем було визначено, що використання програмних та апаратних методів вбудованих процесорних систем має свої переваги, недоліки та певні та обмеження. Так, програмні методи найбільш підходять для аналізу програмного коду та алгоритмів, тоді як апаратні методи надають можливість вивчати фізичні аспекти роботи системи. Кращий підхід може залежати від конкретної задачі та обмежень дослідження.

Результати аналізу виконаних досліджень показали, що комбінування програмних та апаратних методів може бути найбільш ефективним підходом до вивчення та оптимізації вбудованих процесорних систем. Використання прикладних програмних засобів разом з апаратними засобами дозволяє отримати комплексний аналіз роботи системи, що сприяє підвищенню їхньої ефективності та надійності.

В третьому розділу проводилась розробка вимог щодо основних параметрів роботи вбудованих процесорних систем, де були розглянуті ключові параметри. Визначення таких параметрів вбудованих процесорних систем, як продуктивність, витрати енергії, надійність та інші, є важливим кроком у створенні ефективних вбудованих систем. При цьому, визначені вимоги відповідають потребам розробки програмного засобу.

Процес налаштування основних параметрів роботи вбудованих процесорних систем включала в себе вибір конфігурації процесорів, пам'яті, та інших апаратних компонентів, а також оптимізацію параметрів для досягнення найкращих результатів у визначених вимогах.

Також, в цьому розділі досліджувався процес налаштування додаткових параметрів роботи вбудованих процесорних систем, таких як взаємодія з іншими системами, використання спеціалізованих датчиків, зберігання даних та інші функції. При цьому було відзначено важливість врахування цих параметрів для створення вбудованих систем, які відповідають потребам конкретних застосувань під час розробки програмного засобу.

Таким чином, результати розділу проектування та впровадження вбудованих процесорних систем підкреслювали важливість ретельного розробки вимог та налаштування параметрів для створення високоефективних та надійних систем. Цей процес вимагав глибокого розуміння специфіки конкретних застосувань, вдосконалення підходів до проектування вбудованих процесорних систем та використання методів підвищення ефективності роботи вбудованих процесорних систем, що були впроваджені у програмному засобі.

Список використаної літератури

1. Ovod D., Khoshaba O. The impact of the development of embedded processor systems on gaming software //III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів "Комютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023". Збірник матеріалів - Одеса, 2023. – С. 151-152.

2. Програмування вбудованих систем : метод. вказівки до виконання лабораторних робіт для студентів денної та заочної форми навчання за спеціальністю 123 “Комп’ютерна інженерія ” / уклад. Дрєєва Г.М., Дрєєв О.М., Денисенко О.О., Коноплицька-Слободенюк О.К. — Кропивницький: ЦНТУ, 2022. — 90 с.

3. Програмування вбудованих систем : метод. вказівки до виконання лабораторних робіт для студентів денної та заочної форми навчання за спеціальністю 123 “Комп’ютерна інженерія ” / уклад. Дрєєва Г.М., Дрєєв О.М., Денисенко О.О., Коноплицька-Слободенюк О.К. — Кропивницький: ЦНТУ, 2022. — 90 с.

4. Хошаба О.М., Гречанінов В.Ф., Лопушанський А.В. Особливості аналізу та проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах /На шляху до індустрії 4.0:інформаційні технології, моделювання, штучний інтелект, автоматизація. Монографія.Одеса, 2021. -С.232-242.

5. Хошаба О.М., Гречанінов В.Ф., Лопушанський А.В. Особливості аналізу та проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах /На шляху до індустрії 4.0:інформаційні технології, моделювання, штучний інтелект, автоматизація. Монографія.Одеса, 2021. -С.232-242.

6. Гладкова О.М., Пархоменко А.В. Дослідження та практична реалізація рекомендаційної системи для вибору апаратно-програмних платформ при автоматизованому проектуванні вбудованих систем. Наукові праці ДонНТУ. Серія

«Інформатика, кібернетика та обчислювальна техніка». 2017. № 2(25). С. 22–31..

7. Поджаренко В.О., Кучерук В.Ю., Севастьянов В.М. Основи мікропроцесорної техніки. Навчальний посібник. - Вінниця: ВНТУ, 2006. - 226 с.

8. Конспект лекцій з дисципліни «Архітектура та проектування програмного забезпечення» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Інженерія програмного забезпечення» із спеціальності 121 – «Інженерія програмного забезпечення» / Укл. В.В.Завгородній, К.М.Ялова.– Кам’янське: ДДТУ, 2019.– 144с.

9. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. – Житомир : ЖДТУ, 2021. – 383 с.

10. Спеціалізовані мікроконтролерні системи. Теорія і практика : Підручник / Є. І. Сокол, І. Ф. Домнін, О. М. Рисований та ін. – Харків: НТУ “ХПІ”, 2007. – 252 с.

11. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2022. – 434 с.: іл.

12. Гомонай-Стрижко М.В. Інформаційні системи та технології на підприємстві:. Конспект лекцій. – Львів: НЛТУ, 2021. – 200 с.

13. Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In IEEE Symposium on Security and Privacy (SP), pages 36–52. IEEE, 2018.

14. Chengwel Wang, Soila P Kavulya, Jiaqi Tan, Liting Hu, Mahendra Kutare, Mike Kasick, Karsten Schwan, Priya Narasimhan, and Rajeev Gandhi. Performance troubleshooting in data centers: an annotated bibliography. ACM SIGOPS Operating Systems Review, 47(3):pages 50–62, 2019.

15. Qingyang Wang, Yasuhiko Kanemasa, Jack Li, Deepal Jayasinghe, Toshihiro Shimizu, Masazumi Matsubara, Motoyuki Kawaba, and Calton Pu. Detecting transient bottlenecks in n-tier applications through fine-grained analysis. In IEEE 33rd International Conference on Distributed Computing Systems (ICDCS), pages 31–40. IEEE, 2019.

16. Qingyang Wang, Yasuhiko Kanemasa, Jack Li, Deepal Jayasinghe, Toshihiro Shimizu, Masazumi Matsubara, Motoyuki Kawaba, and Calton Pu. bottlenecks in n-tier applications. An experimental study of rapidly alternating In IEEE Sixth International Conference on Cloud Computing (CLOUD), pages 171–178. IEEE, 2019.

17. Matt Welsh and David E Culler. Adaptive overload control for busy internet servers. In USENIX Symposium on Internet Technologies and Systems (USITS), pages 124–131. Seattle, WA, 2020.

18. Pengcheng Xiong, Calton Pu, Xiaoyun Zhu, and Rean Griffith. vperfguard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments. In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE), pages 271–282. ACM, 2020.

19. Tianyin Xu, Xinxin Jin, Peng Huang, Yuanyuan Zhou, Shan Lu, Long Jin, and Shankar Pasupathy. Early detection of configuration errors to reduce failure damage. In 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pages 619–634, 2021.

20. Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy. Do not blame users for misconfigurations. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP), pages 244–259. ACM, 2021.

21. Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N Bairavasundaram, and Shankar Pasupathy. An empirical study on configuration errors in commercial and open source systems. In Proceedings of the Twenty-Third

ACM Symposium on Operating Systems Principle (SOSP), pages 159–172. ACM, 2021.

22. Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In Proceedings of the twenty-fourth ACM symposium on operating systems principles (SOSP), pages 423–438. ACM, 2021.

23. Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, et al. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In Proceedings of the 2021 International Conference on Management of Data (SIGMOD), pages 415–432. ACM, 2021.

24. Junzhe Zhang, Sai Ho Yeung, Yao Shu, Bingsheng He, and Wei Wang. Efficient memory management for gpu-based deep learning systems. arXiv preprint arXiv:1903.06631, 2021.

25. Xiao Zhang, Eric Tune, Robert Hagmann, Rohit Jnagal, Vrigo Gokhale, and John Wilkes. CPI 2 : CPU performance isolation for shared compute clusters. In Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys), pages 379–391. ACM, 2022.

26. Hao Zhou, Ming Chen, Qian Lin, Yong Wang, Xiaobin She, Sifan Liu, Rui Gu, Beng Chin Ooi, and Junfeng Yang. Overload control for scaling wechat microservices. In Proceedings of the ACM Symposium on Cloud Computing (SOCC), pages 149–161. ACM, 2021.

27. Tao Zou, Ronan Le Bras, Marcos Vaz Salles, Alan Demers, and Johannes Gehrke. deployment advisor for public clouds. The VLDB Journal, 24(5): pages 633–653, 2021.

28. Waheed Iqbal, Matthew N Dailey, David Carrera, and Paul Janecek. Sla-driven automatic bottleneck detection and resolution for read intensive multi-tier

applications hosted on a cloud. In International Conference on Grid and Pervasive Computing (GPC), pages 37–46. Springer, 2020.

29. Gueyoung Jung, Galen Swint, Jason Parekh, Calton Pu, and Akhil Sahai. Detecting bottleneck in n-tier it applications through analysis. In International Workshop on Distributed Systems: Operations and Management (DSOM), pages 149–160. Springer, 2020.

30. A Karve, Tracy Kimbrel, Giovanni Pacifici, Mike Spreitzer, Malgorzata Steinder, Maxim Sviridenko, and A Tantawi. Dynamic placement for clustered web applications. In Proceedings of the 19th international conference on World Wide Web (WWW), pages 595–604. ACM, 2020.

31. Gunjan Khanna, Ignacio Laguna, Fahad A Arshad, and Saurabh Bagchi. Stateful detection in high throughput distributed systems. In 26th IEEE International Symposium on Reliable Distributed Systems (SRDS), pages 275–287. IEEE, 2019.

32. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : **В. О. Козловський, О. Й. Лесько, В. В. Кавецький**. Вінниця : ВНТУ, 2021. 42 с.

33. **Кавецький В. В.** Економічне обґрунтування інноваційних рішень: **практикум** / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, **2016**. 113 с.

ДОДАТКИ

Міністерство освіти і науки України
Вінницький національний технічний університет
факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
" 19 " вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів підтримки вбудованих процесорних систем» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доцент О.М. Хошаба
" 19 " 09 2023 р.

Виконав:
студент гр. 1ПІ-22м Д. В. Овод
" 19 " 09 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів підтримки вбудованих процесорних систем».

Галузь застосування – вбудовані процесорні системи, апаратне та програмне забезпечення обчислювальних систем.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності підвищення ефективності використання вбудовані процесорні системи за рахунок використання сучасних методів оптимізації оптимізації даних.

Призначення роботи – розробка методів і засобів підвищення ефективності використання вбудованих процесорних систем.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Ovod D., Khoshaba O. The impact of the development of embedded processor systems on gaming software //III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів "Комютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023". Збірник матеріалів - Одеса, 2023. – С. 151-152.
2. Програмування вбудованих систем : метод. вказівки до виконання лабораторних робіт для студентів денної та заочної форми навчання за спеціальністю 123 “Комп’ютерна інженерія ” / уклад. Дрєєва Г.М., Дрєєв

О.М., Денисенко О.О., Коноплицька-Слободенюк О.К. — Кропивницький: ЦНТУ, 2022. — 90 с.

3. Конспект лекцій з дисципліни «Програмування систем реального часу» напрям підготовки 6.050202 «Автоматизація та комп’ютерно-інтегровані технології» /Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2021. – 76 с.

4. Хошаба О.М., Гречанінов В.Ф., Лопушанський А.В. Особливості аналізу та проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах /На шляху до індустрії 4.0:інформаційні технології, моделювання, штучний інтелект, автоматизація. Монографія.Одеса, 2021. - С.232-242.

4. Технічні вимоги

Для 8-бітних мікроконтролерів: серія PIC та AVR, тактова частота від 1 до 20 МГц, розрядність 8 біт, пам'ять RAM від кількох десятків байт до кількох кілобайт, Flash-пам'ять від кількох кілобайт до 256 КБ. Для 16-бітових мікроконтролерів: серія MSP430, тактова частота до 25 МГц, розрядність 16 біт, пам'ять RAM від кількох сотень байт до кількох кілобайт Flash-пам'ять до 512 КБ. Для 32-бітових мікроконтролери: серія ARM Cortex-M, тактова частота від 20 МГц до кількох сотень МГц, розрядність 32 біти, пам'ять RAM від кількох кілобайт до кількох мегабайт, Flash-пам'ять від 256 КБ до кількох мегабайт.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів роботи вбудованих систем, визначення програмних та апаратних методів дослідження.	19.09.2023-02.10.2023
2	Дослідження програмних і апаратних особливостей використання вбудованих систем.	03.10.2023-16.10.2023
3	Розробка вимог щодо основних параметрів роботи вбудованих систем, налаштування параметрів.	17.10.2023-28.10.2023
4	Створення програмного засобу для аналізу роботи вбудованих процесорних систем.	29.10.2023-12.11.2023
5	Економічна частина	13.11.2023-01.12.2023

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б. Протокол перевірки навчальної (кваліфікаційної) роботи

Назва роботи: **Розробка методів і програмних засобів підтримки вбудованих процесорних систем**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 22м

Науковий керівник: к.т.н. доц. Хошаба О. М.

Unicheck	
Оригінальність	**96,6%**
Схожість	3,4 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методів і програмних засобів підтримки вбудованих процесорних систем».

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку — Черноволик Г. О.
(підпис) (прізвище, ініціали)

Автор — Овод Д. В.

Керінник роботи — Хошаба О.М.

Додаток В. Лістинг програми

Форма на javascript для виклику основних модулів.

```
<!DOCTYPE html>
<html>
<head>
  <title>Виклик модулів</title>
</head>
<body>
  <h1>Виклик модулів</h1>
  <ul>
    <li><a href="#" onclick="викликати модуль('Частота Роботи')">Частота роботи</a></li>
    <li><a href="#" onclick="викликати модуль('Потужність')">Потужність</a></li>
    <li><a href="#" onclick="викликати модуль('Теплова Продуктивність')">Теплова продуктивність</a></li>
    <li><a href="#" onclick="викликати модуль('Розмір Пам'яті')">Розмір пам'яті</a></li>
    <li><a href="#" onclick="викликати модуль('Обсяг Кеш Пам'яті')">Об'єм кеш-пам'яті</a></li>
    <li><a href="#" onclick="викликати модуль('Кількість Ядер')">Кількість ядер</a></li>
  </ul>

  <script>
    function викликати модуль(назваМодуля) {
      // Виклик модуля
      console.log('Виклик модуля:', назваМодуля);
      // Виконати логіку для виклику певного модуля
```

```
        // Наприклад, відправити запит на сервер або показати певний контент на сторінці
      }
    </script>
  </body>
  </html>

  <!DOCTYPE html>
  <html>
  <head>
    <title>Виклик модулів</title>
  </head>
  <body>
    <h1>Виклик модулів</h1>
    <ul>
      <li><a href="#" onclick="викликатиМодуль('Інструкційний Набір')">Інструкційний набір</a></li>
      <li><a href="#" onclick="викликатиМодуль('Температурний Діапазон')">Температурний діапазон</a></li>
      <li><a href="#" onclick="викликатиМодуль('ОперативнаПамять')">Оперативна пам'ять</a></li>
      <li><a href="#" onclick="викликатиМодуль('КонсумпціяПамяті')">Консумпція пам'яті</a></li>
      <li><a href="#" onclick="викликати модуль('Джерела Живлення')">Джерела живлення</a></li>
    </ul>

    <script>
      function викликати модуль(назва модуля) {
        // Виклик модуля
```

```
        console.log('Виклик модуля:', назва модуля);
        // Виконати логіку для виклику певного модуля
        // Наприклад, відправити запит на сервер або показати певний контент на сторінці
      }
    </script>
  </body>
  </html>
```

Реалізація бекенду на Java з використанням Spring та взаємодії з базою даних MySQL. Для роботи з базою даних використовується JPA (Java Persistence API). В якості збірки Java коду використовується maven.

Додамо залежності для Spring та JPA в файл pom.xml:

```
<dependencies>
    <!-- Залежності для Spring та JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Створимо сутності для роботи з базою даних для класів Module та ModuleRepository:

```
// Module.java
@Entity
```

```
@Table(name = "modules")
public class Module {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    // геттери, сеттери та інші поля
}

// ModuleRepository.java
@Repository
public interface ModuleRepository extends JpaRepository<Module, Long> {
    // Власні методи для роботи з модулями, які використовують JPA
}
// Module.java
@Entity
@Table(name = "modules")
public class Module {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

    // геттери, сеттери та інші поля
}
```

Контролер для обробки запитів, пов'язаних з модулями:

```
// ModuleController.java
@RestController
@RequestMapping("/modules")
```

```
public class ModuleController {
    private final ModuleRepository moduleRepository;

    @Autowired
    public ModuleController(ModuleRepository moduleRepository) {
        this.moduleRepository = moduleRepository;
    }

    @GetMapping("/{moduleId}")
    public Module getModule(@PathVariable Long moduleId) {
        return moduleRepository.findById(moduleId).orElse(null);
    }
    @PostMapping
    public Module createModule(@RequestBody Module module) {
        return moduleRepository.save(module);
    }
    // Інші методи для обробки запитів
}
```

Налаштування щодо підключення до бази даних у application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/base_database
spring.datasource.username=user
spring.datasource.password=user12345
spring.jpa.hibernate.ddl-auto=update
```

```
import javax.persistence.*;

@Entity
@Table(name = "processor_frequency") // назва таблиці в базі даних
public class ProcessorFrequency {
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@Column(name = "frequency")
private double frequency;

// Конструктори (потрібні для JPA)
public ProcessorFrequency() {
}

public ProcessorFrequency(double frequency) {
    this.frequency = frequency;
}

// Геттери та сеттери
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public double getFrequency() {
    return frequency;
}

public void setFrequency(double frequency) {
```

```
        this.frequency = frequency;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "processor_power") // назва таблиці в базі даних
public class ProcessorPower {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "power")
    private double power;

    // Конструктори (потрібні для JPA)
    public ProcessorPower() {
    }

    public ProcessorPower(double power) {
        this.power = power;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }
```

```
    public void setId(Long id) {
        this.id = id;
    }

    public double getPower() {
        return power;
    }

    public void setPower(double power) {
        this.power = power;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "embedded_device_memory") // назва таблиці в базі даних
public class EmbeddedDeviceMemory {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "memory_size")
    private int memorySize;

    // Конструктори (потрібні для JPA)
    public EmbeddedDeviceMemory() {
    }
```

```
    public EmbeddedDeviceMemory(int memorySize) {
        this.memorySize = memorySize;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public int getMemorySize() {
        return memorySize;
    }

    public void setMemorySize(int memorySize) {
        this.memorySize = memorySize;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "embedded_device_cache_memory") // назва таблиці в базі даних
public class EmbeddedDeviceCacheMemory {

    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@Column(name = "cache_memory_size")
private int cacheMemorySize;

// Конструктори (потрібні для JPA)
public EmbeddedDeviceCacheMemory() {
}

public EmbeddedDeviceCacheMemory(int cacheMemorySize) {
    this.cacheMemorySize = cacheMemorySize;
}

// Геттери та сеттери
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public int getCacheMemorySize() {
    return cacheMemorySize;
}

public void setCacheMemorySize(int cacheMemorySize) {
    this.cacheMemorySize = cacheMemorySize;
}
```

```
}

import javax.persistence.*;

@Entity
@Table(name = "embedded_device_core_count") // назва таблиці в базі даних
public class EmbeddedDeviceCoreCount {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "core_count")
    private int coreCount;

    // Конструктори (потрібні для JPA)
    public EmbeddedDeviceCoreCount() {
    }

    public EmbeddedDeviceCoreCount(int coreCount) {
        this.coreCount = coreCount;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
```

```
    }

    public int getCoreCount() {
        return coreCount;
    }

    public void setCoreCount(int coreCount) {
        this.coreCount = coreCount;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "instruction_set") // назва таблиці в базі даних
public class InstructionSet {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "instruction_set_name")
    private String instructionSetName;

    // Конструктори (потрібні для JPA)
    public InstructionSet() {
    }

    public InstructionSet(String instructionSetName) {
        this.instructionSetName = instructionSetName;
```

```
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getInstructionSetName() {
        return instructionSetName;
    }

    public void setInstructionSetName(String instructionSetName) {
        this.instructionSetName = instructionSetName;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "temperature_range") // назва таблиці в базі даних
public class TemperatureRange {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @Column(name = "min_temperature")
    private double minTemperature;

    @Column(name = "max_temperature")
    private double maxTemperature;

    // Конструктори (потрібні для JPA)
    public TemperatureRange() {
    }

    public TemperatureRange(double minTemperature, double maxTemperature)
{
        this.minTemperature = minTemperature;
        this.maxTemperature = maxTemperature;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public double getMinTemperature() {
        return minTemperature;
    }

    public void setMinTemperature(double minTemperature) {
```

```
        this.minTemperature = minTemperature;
    }

    public double getMaxTemperature() {
        return maxTemperature;
    }

    public void setMaxTemperature(double maxTemperature) {
        this.maxTemperature = maxTemperature;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "ram") // назва таблиці в базі даних
public class RAM {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "memory_size")
    private int memorySize;

    // Конструктори (потрібні для JPA)
    public RAM() {
    }

    public RAM(int memorySize) {
```

```
        this.memorySize = memorySize;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public int getMemorySize() {
        return memorySize;
    }

    public void setMemorySize(int memorySize) {
        this.memorySize = memorySize;
    }
}

import javax.persistence.*;

@Entity
@Table(name = "memory_consumption") // назва таблиці в базі даних
public class MemoryConsumption {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @Column(name = "consumption")
    private int consumption;

    // Конструктори (потрібні для JPA)
    public MemoryConsumption() {
    }

    public MemoryConsumption(int consumption) {
        this.consumption = consumption;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public int getConsumption() {
        return consumption;
    }

    public void setConsumption(int consumption) {
        this.consumption = consumption;
    }
}
```

```
import javax.persistence.*;

@Entity
@Table(name = "power_sources") // назва таблиці в базі даних
public class PowerSources {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "source_type")
    private String sourceType;

    // Конструктори (потрібні для JPA)
    public PowerSources() {
    }

    public PowerSources(String sourceType) {
        this.sourceType = sourceType;
    }

    // Геттери та сеттери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

```
    public String getSourceType() {
        return sourceType;
    }

    public void setSourceType(String sourceType) {
        this.sourceType = sourceType;
    }
}
```

Додаток Г. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Слайд 1 – Тема, автор, науковий керівник бакалаврської дипломної роботи:

Розробка методів і програмних засобів підтримки вбудованих процесорних систем

Виконав:
студент групи 1ПІ-22м

Овод Д. В.

Керівник:
к.т.н., доц. каф. ПЗ

Хошаба О.М.

Слайд 2 – Мета, об'єкт та предмет дослідження:

Метою роботи є підвищення ефективності використання вбудованих процесорних систем.

Об’єктом дослідження є процеси роботи вбудованих процесорних систем.

Предметом дослідження є методи та засоби підвищення ефективності використання вбудованих процесорних систем.

Слайд 3 – Задачі дослідження:

Основними задачами дослідження є:

- виконати аналіз методів роботи вбудованих процесорних систем;
- визначити програмні та апаратні методи дослідження роботи вбудованих процесорних систем;
- виконати дослідження програмних та апаратних особливостей у використанні методів вбудованих процесорних систем;
- розробити вимоги щодо основних параметрів роботи вбудованих процесорних систем;
- дослідити налаштування основних та додаткових параметрів роботи вбудованих процесорних систем;
- розробити програмний засіб з аналізу роботи вбудованих процесорних систем на основі програмних та апаратних методів дослідження.

Слайд 4 – Актуальність розробки:

Стрімке зростання складності задач вимагає від споживачів та промислової індустрії постійно змінювати та висувати нові вимоги до вбудованих систем.

Серед таких змін зазвичай є підтримка складніших алгоритмів обробки сигналів, штучного інтелекту, машинного навчання. Використання таких галузей знань вимагають великої продуктивності та оптимізованого використання ресурсів від вбудованих систем.

При цьому, зменшення розмірів та енергоспоживання у вбудованих системах породжують багато проблем.

Відповідно до цього, саме розробка ефективних методів, що забезпечують високу продуктивність при обмежених ресурсах, є вкрай важливою задачею.

Таким чином, розвиток вбудованих процесорних систем виконується у напрямку високої продуктивності, енергоефективності, безпеки та інтеграції з виробництвом. Тому, актуальним завданням є розробка методів та програмно-апаратних рішень, що направлені на підвищення ефективності їх використання.

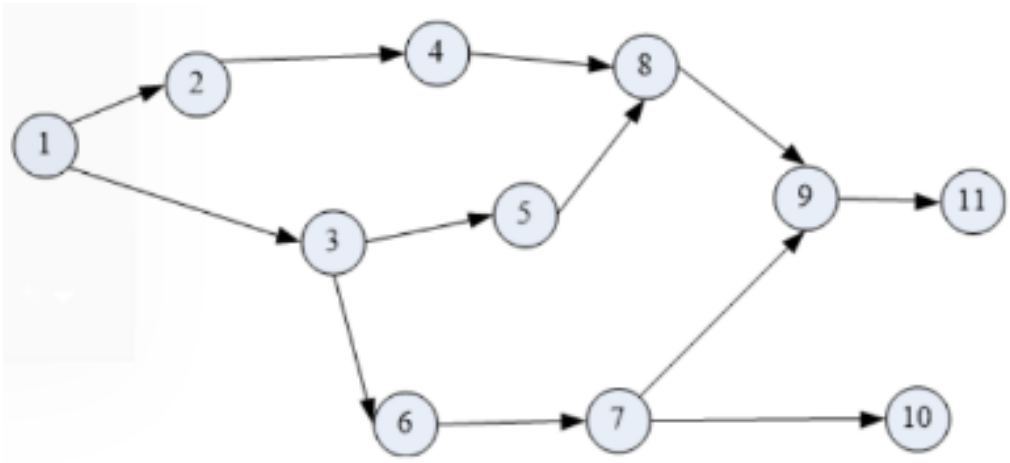
Слайд 5 – Програмні методи дослідження роботи вбудованих процесорних систем

Метод дослідження	Опис
Моніторинг	Відстеження та запис різних параметрів роботи системи, таких як використання CPU, використання пам'яті, мережевий трафік тощо.
Дебагінг	Виявлення та усунення помилок у програмному коді, включаючи аналіз стек-трейсів, логів, точок зупинки тощо.
Профілювання	Збір та аналіз інформації щодо часу виконання різних функцій та інструкцій у програмі для оптимізації продуктивності.
Тестування	Виконання різних тестів, які допомагають перевірити коректність та надійність роботи системи, включаючи функціональне тестування та модульне тестування.
Аналіз ресурсів	Моніторинг та аналіз використання ресурсів системи, таких як CPU, пам'ять, диски, для оптимізації їх використання.
Імітація	Створення імітованих середовищ для тестування та аналізу системи без впливу на реальну систему.
Аналіз даних	Обробка та аналіз великих обсягів даних, що збираються від системи для виявлення закономірностей та тенденцій.

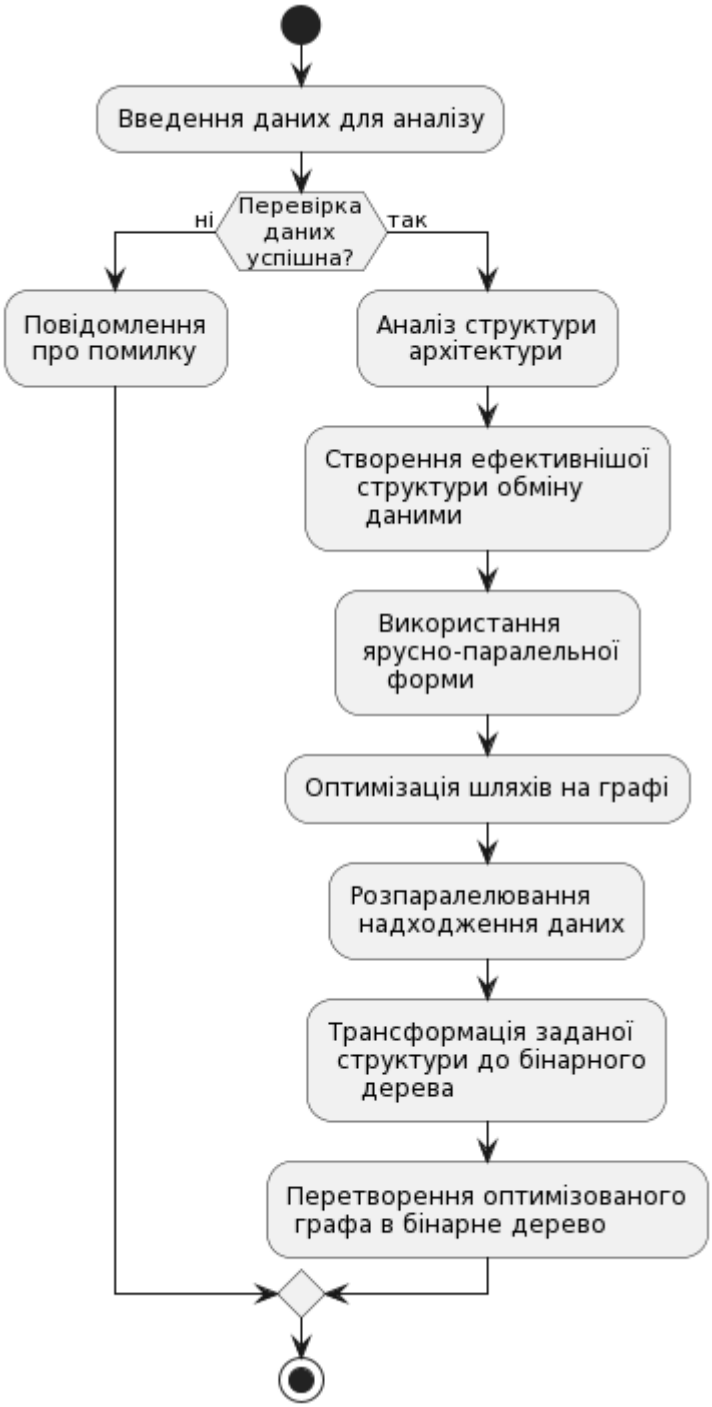
Слайд 6 — Апаратні методи дослідження роботи вбудованих процесорних систем

Метод дослідження	Опис
Профілювання процесора	Вимірювання часу виконання окремих функцій та інструкцій процесора за допомогою лічильників продуктивності. Дозволяє аналізувати продуктивність та оптимізовувати код.
Запис та аналіз трасування виконання завдань	Запис послідовності виконаних інструкцій процесором для аналізу програмного коду та виявлення помилок.
Аналіз роботи кеш-пам'яті	Вимірювання активності та промахів кеш-пам'яті процесора для виявлення проблем з кешуванням та оптимізації пам'яті.
Моніторинг апаратних подій	Перехоплення та аналіз апаратних подій, таких як переривання та помилки пам'яті, для відлагодження та виявлення апаратних проблем.
Зондування подій	Встановлення зондів на внутрішні шини процесора для перехоплення та аналізу даних та комунікації між компонентами системи.
Моніторинг температури	Вимірювання температури процесора та інших компонентів для виявлення перегріву та керування температурою.
Моніторинг енергоспоживання	Вимірювання енергоспоживання процесора та системи для оцінки енергоефективності та зменшення споживання енергії.
Моніторинг пам'яті	Вимірювання використання оперативної та кеш-пам'яті для виявлення проблем з витоком пам'яті та оптимізації пам'яті.
Моніторинг використання ресурсів	Вимірювання використання ресурсів, таких як CPU та пам'ять, для оцінки продуктивності та оптимізації використання ресурсів.
Вимірювання швидкості обробки даних	Вимірювання кількості операцій, які система виконує за одиницю часу для оцінки продуктивності.

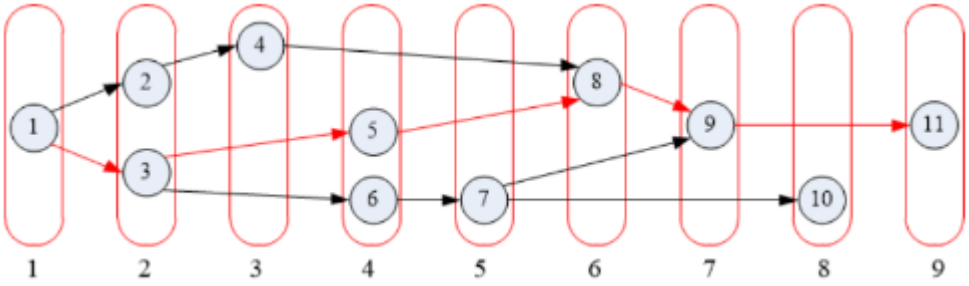
Слайд 7 — Загальна структура програмно-апаратного засобу, що складається з вбудованих процесорів та представлена у вигляді моделі графа



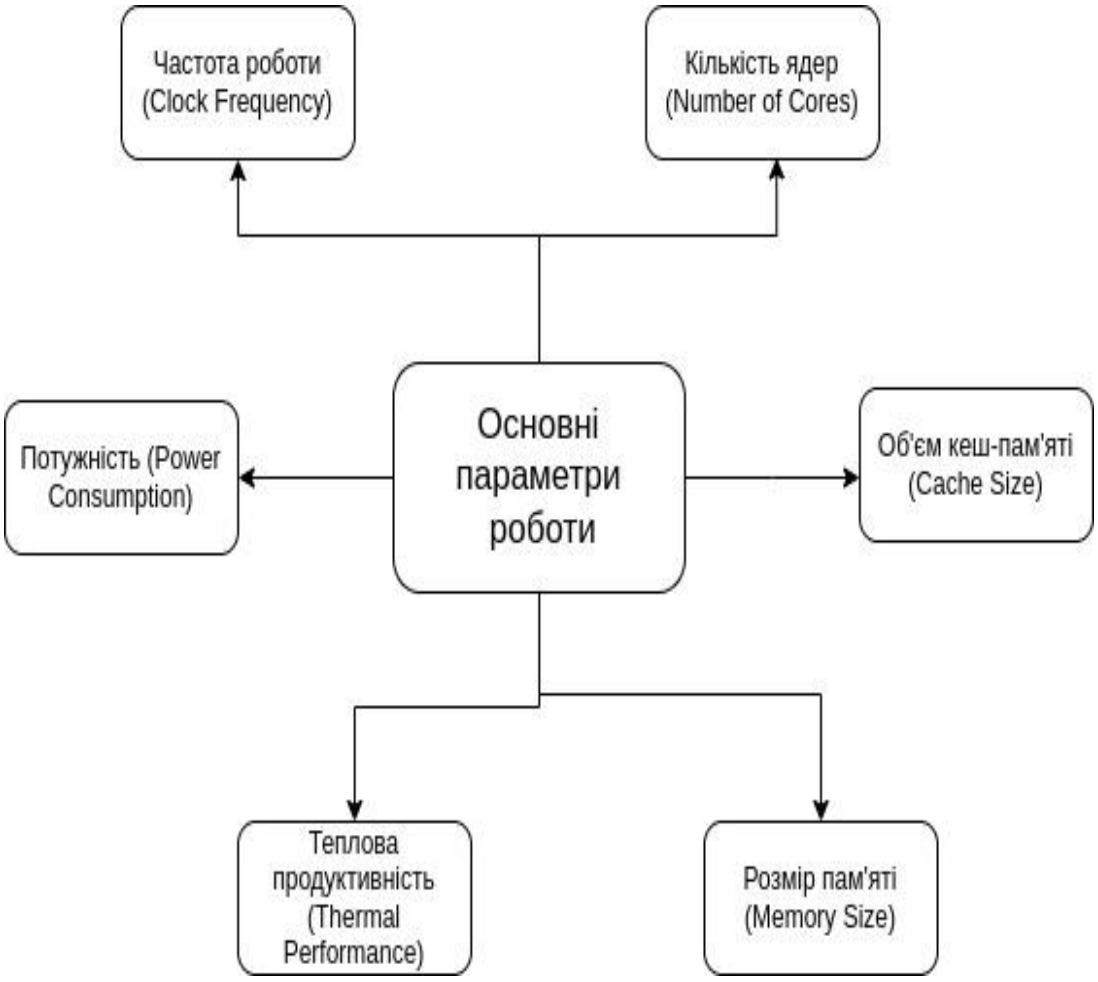
Слайд 8 — UML підвищення ефективності роботи вбудованих процесорних систем



Слайд 9 – Загальна та оптимізована структура програмно-апаратного засобу, яка була оптимізована з 9 (рис. 2.6) до 6



Слайд 10 — Основні параметрів роботи вбудованих процесорних систем



Слайд 11 — Додаткові параметри роботи вбудованих процесорних систем

Слайд 12 – Наукова новизна одержаних результатів:

- вперше запропоновано метод підвищення ефективності роботи вбудованих процесорних систем, особливість якого полягає у виконанні перетворення із загальної структури програмно-апаратного засобу на основі графової моделі до ярусно-паралельній форми, що виконується на основі закону Амдала і, як наслідок, дає можливість підвищити ефективність обробки даних до 34%;

- подальшого розвитку отримав метод перетворення оптимізованого графа в бінарне дерево, у якому, на відміну від існуючих, вдається більш ефективно до 10% виконувати перетворення за рахунок повторних рекурсивних дій кожної з двох сусідніх груп до піддерева графової форми.

Слайд 13 – Висновки:

В роботі:

- виконано аналіз методів роботи вбудованих процесорних систем;
- визначено програмні та апаратні методи дослідження роботи вбудованих процесорних систем;
- виконано дослідження програмних та апаратних особливостей у використанні методів вбудованих процесорних систем;
- розроблені вимоги щодо основних параметрів роботи вбудованих процесорних систем;
- досліджено налаштування основних та додаткових параметрів роботи вбудованих процесорних систем;
- розроблено програмний засіб з аналізу роботи вбудованих процесорних систем на основі програмних та апаратних методів дослідження.

Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів "Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023", (Одеса, 2023).

Основні результати досліджень опубліковано в науковій праці у матеріалах цієї конференції.