ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмно-апаратна система аналізу якості повітря на основі мікроконтролерів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

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

_____ *Захар РАХМАКОВ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-41
Захар РАХМАКОВ
Ім'я, ПРІЗВИЩЕ

Керівник: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рахмаков Захар Андрійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: FPGA-прискорювач для обробки потокових даних у реальному часі_____

керівник кваліфікаційної роботи_____----------_____,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024р. № 36

1. Строк подання кваліфікаційної роботи: "3" червня 2024р. (Залишається без змін, якщо це актуально)

2. Вихідні дані до кваліфікаційної роботи:

- FPGA (Field-Programmable Gate Arrays) від Xilinx або Intel (наприклад, з сімейств Zynq, Kintex, Stratix, Arria).
- Платформи розробки FPGA (наприклад, PYNQ, DE10-Nano, інші плати розробки з відповідною документацією).
- Інтерфейси передачі даних:
  - PCIe (для високошвидкісної передачі даних між FPGA та хост-комп'ютером).
  - Ethernet (для передачі даних по мережі).
  - AXI (Advanced eXtensible Interface) для внутрішньої комунікації на FPGA.
- Мови опису апаратури (HDL): VHDL, Verilog.
- Інструменти розробки FPGA: Xilinx Vivado, Intel Quartus Prime.
- Потокові дані для обробки (наприклад):
  - Синтетичні потокові дані (для тестування).
  - Потокові дані з датчиків (якщо є можливість інтеграції з реальними датчиками).
  - Потокові відеодані.
- Вимоги до продуктивності:

        - Мінімальна пропускна здатність обробки даних (наприклад, Гбіт/с).
        - Максимальна допустима затримка (наприклад, мікросекунди).
    - Вимоги до енергоспоживання (якщо актуально).
3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
    - Аналіз існуючих підходів до обробки потокових даних у реальному часі (CPU, GPU, ASIC) та обґрунтування вибору FPGA.
    - Обґрунтування вибору апаратної платформи FPGA та інструментів розробки.
    - Розробка архітектури FPGA-прискорювача для обробки потокових даних.
        - Визначення функціональних блоків (наприклад, модуль прийому даних, модуль обробки, модуль передачі).
        - Опис інтерфейсів між блоками.
        - Реалізація ключових алгоритмів обробки даних на FPGA (наприклад, фільтрація, перетворення, виявлення об'єктів).
    - Розробка (опис) HDL-коду для реалізації FPGA-прискорювача.
    - Моделювання та верифікація роботи FPGA-прискорювача.
    - Оцінка ефективності розробленого FPGA-прискорювача.
        - Вимірювання пропускної здатності.
        - Вимірювання затримки.
        - Аналіз використання ресурсів FPGA.
        - Оцінка енергоспоживання (якщо можливо).
    - Аналіз обмежень розробленої системи та можливі напрями оптимізації.
4. Перелік ілюстративного матеріалу:
    - Презентація.
    - Блок-схеми архітектури FPGA-прискорювача.
    - Результати моделювання та верифікації.
    - Графіки та таблиці з результатами оцінки ефективності.

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: [кількість сторінок], [кількість рисунків], [кількість таблиць], [кількість джерел].

Мета роботи – Розробити концепцію FPGA-прискорювача для обробки потокових даних у реальному часі та визначити основні технічні рішення для його реалізації.

Об'єкт дослідження – Процес прискорення обробки потокових даних за допомогою програмованих вентильних матриць (FPGA).

Предмет дослідження – Архітектура FPGA-прискорювача, методи паралелізації обчислень на FPGA, інтерфейси передачі потокових даних та методи оцінки продуктивності систем реального часу на основі FPGA.

Короткий зміст роботи: У кваліфікаційній роботі розглянуто сучасний стан проблеми обробки великих обсягів потокових даних у реальному часі та обґрунтовано актуальність використання FPGA для прискорення цих процесів. Проведено аналіз існуючих підходів до реалізації апаратних прискорювачів для обробки потокових даних, їх переваги та недоліки. Особливу увагу приділено аналізу архітектур FPGA, їхніх можливостей щодо паралельної обробки та огляду типових інтерфейсів для передачі потокових даних.

Розроблено концептуальну архітектуру FPGA-прискорювача, що включає модулі для прийому, обробки та передачі потокових даних. Описано можливі методи паралелізації обчислень на FPGA для досягнення високої пропускної здатності та низької затримки. Розглянуто метрики для оцінки ефективності роботи розробленого прискорювача, такі як пропускна здатність, затримка, використання ресурсів FPGA та енергоспоживання, а також можливі сценарії його тестування.

Результати роботи демонструють потенційну ефективність запропонованої концепції FPGA-прискорювача для здійснення швидкої та ефективної обробки потокових даних у реальному часі.

КЛЮЧОВІ СЛОВА: FPGA, ПРИСКОРЮВАЧ, ПОТОКОВІ ДАНІ, РЕАЛЬНИЙ ЧАС, ПАРАЛЕЛЬНА ОБРОБКА, ПРОПУСКНА ЗДАТНІСТЬ, ЗАТРИМКА, ВИКОРИСТАННЯ РЕСУРСІВ, ТЕСТУВАННЯ.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Збір даних		Виконано
	Обробка матеріалу		Виконано
	Розробка теоретичної частини		Виконано
	Розробка моделі мережі		Виконано
	Висновки за результатами аналізу		Виконано
	Оформлення пояснювальної записки		Виконано
	Розробка презентації		Виконано
	Передзахист		Виконано
	Здача в деканат		Виконано

Здобувач вищої освіти _____ Захар Рахмаков
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ -----
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОБРОБКИ ПОТОКОВИХ ДАНИХ 9
1.1 СУЧАСНІ ВИМОГИ ДО ОБРОБКИ ДАНИХ У РЕАЛЬНОМУ ЧАСІ ТА РОЛЬ FPGA 9
1.1.1 СУЧАСНІ ВИМОГИ ДО ОБРОБКИ ДАНИХ У РЕАЛЬНОМУ ЧАСІ 10
1.2 РОЛЬ FPGA В ОБРОБЦІ ДАНИХ У РЕАЛЬНОМУ ЧАСІ 11
1.1 Огляд існуючих FPGA-прискорювачів 12
1.2.1 ПРОМИСЛОВІ РІШЕННЯ 13
1.2.2 НАУКОВІ ДОСЛІДЖЕННЯ 13
1.2.3 ВІДКРИТІ ТА АМАТОРСЬКІ РІШЕННЯ 14
1.3 АНАЛІЗ АРХІТЕКТУР FPGA: ПАРАЛЕЛІЗМ, ЕНЕРГОЕФЕКТИВНІСТЬ ТА ОБМЕЖЕННЯ 15
1.3.1 ПАРАЛЕЛЬНА ОБРОБКА ДАНИХ 15
1.3.2 ЕНЕРГОЕФЕКТИВНІСТЬ 16
1.3.3 ОБМЕЖЕННЯ FPGA 16
1.4 ПОРІВНЯННЯ FPGA З ІНШИМИ ПЛАТФОРМАМИ (CPU, GPU, ASIC) 17
1.4.1 КРИТЕРІЇ ПОРІВНЯННЯ 17
1.4.2 ДЕТАЛЬНИЙ АНАЛІЗ ПЛАТФОРМ 17
1.4.3 ОПТИМАЛЬНІ СФЕРИ ЗАСТОСУВАННЯ 19
2 РОЗРОБКА FPGA-ПРИСКОРЮВАЧА ДЛЯ ОБРОБКИ ПОТОКОВИХ ДАНИХ 20
2.1 ТЕОРЕТИЧНІ ОСНОВИ ПРОЕКТУВАННЯ FPGA-ПРИСКОРЮВАЧІВ 20
2.1.1 АРХІТЕКТУРНІ ШАБЛОНИ FPGA-ПРИСКОРЮВАЧІВ 20
2.1.2 ОПТИМІЗАЦІЯ АПАРАТНИХ РЕСУРСІВ 21
2.1.3 ВИБІР ІНТЕРФЕЙСІВ ПЕРЕДАЧІ ДАНИХ 21
2.1.4 МЕТОДИКА ПРОЕКТУВАННЯ 21
2.1.5 ПЕРЕВАГИ АПАРАТНОЇ РЕАЛІЗАЦІЇ 21
2.2 ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ ТА ПЛАТФОРМИ (PYNQ, DE10-NANO) 22
2.2.1 КРИТЕРІЇ ВИБОРУ ПЛАТФОРМИ 22
2.2.2 АНАЛІЗ ПЛАТФОРМИ PYNQ 23
2.2.3 АНАЛІЗ ПЛАТФОРМИ DE10-NANO 23
2.2.4 ОБҐРУНТУВАННЯ ВИБОРУ 24
2.2.5 ІНСТРУМЕНТИ РОЗРОБКИ 25
2.3 ДЕТАЛЬНИЙ ПРИКЛАД ВИКОРИСТАННЯ FPGA: 25
2.4 ВАЛІДАЦІЯ МОДЕЛІ: ПОРІВНЯННЯ З CPU-РЕАЛІЗАЦІЄЮ 27
3 ОЦІНКА ЕФЕКТИВНОСТІ ТА ПЕРСПЕКТИВИ ВДОСКОНАЛЕННЯ 30
3.1 Метрики оцінок (Швидкодія, енергоспоживання, ціна реалізації та час амортизції) 30
3.1.1 ШВИДКОДІЯ (ПРОДУКТИВНІСТЬ) - ДЕТАЛЬНИЙ АНАЛІЗ 30
3.1.3 Економічна оцінка (Детально з прикладом, 24/7 робота) 31
3.2 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕОРЕТИЧНА ОЦІНКА ЕФЕКТИВНОСТІ 35
3.3 ОЦІНКА ПРОДУКТИВНОСТІ FPGA-ПРИСКОРЮВАЧА ОБРОБКИ ПОТОКОВИХ ДАНИХ У РЕАЛЬНОМУ ЧАСІ 41
3.3.1 МЕТРИКИ ОЦІНКИ ПРОДУКТИВНОСТІ (ДЕТАЛЬНО) 42
3.3.2 СЦЕНАРІЇ ТЕСТУВАННЯ 44
3.3.3 ІНТЕРПРЕТАЦІЯ РЕЗУЛЬТАТІВ 45
3.4 ОБМЕЖЕННЯ ТА МОЖЛИВІ НАПРЯМИ ОПТИМІЗАЦІЇ 46
3.5 МОЖЛИВІ НАПРЯМИ ОПТИМІЗАЦІЇ 47
ВИСНОВОК 49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50

Вступ

У сучасному світі обробка даних у реальному часі набуває все більшого значення через стрімкий розвиток технологій Інтернету речей (IoT), промислових систем моніторингу та потокових мультимедійних сервісів. Традиційні процесори (CPU) та графічні прискорювачі (GPU) часто не справляються із завданнями, що вимагають мінімальних затримок та енергоефективності. На противагу цьому, програмовані логічні інтегральні схеми (FPGA) пропонують унікальні переваги завдяки своїй паралельній архітектурі, здатності до реконфігурації та низькому енергоспоживанню.

Найпростіші системи обробки даних, такі як фільтрація сигналів або агрегування сенсорних показників, можуть бути реалізовані на мікроконтролерах. Однак для складних завдань, таких як обробка відео в реальному часі або аналіз великих потоків даних, їхньої продуктивності виявляється недостатньо. У корпоративних та промислових рішеннях, де критичною є швидкість та надійність, FPGA стають оптимальним вибором. Проте розробка під FPGA традиційно пов’язана із складністю програмування на мовах низького рівня (Verilog/VHDL) та необхідністю використання дорогого обладнання для тестування.

У цій роботі пропонується альтернативний підхід – емуляція FPGA-прискорювача за допомогою Python, що дозволяє спростити процес розробки та валідації алгоритмів без необхідності глибокого занурення в апаратну реалізацію. Це особливо актуально для дослідницьких цілей, коли основна увага приділяється не реалізації на "залізі", а саме алгоритмічній ефективності. Використання бібліотек типу NumPy та Numba дозволяє імітувати паралельні обчислення, характерні для FPGA, і порівняти продуктивність із програмними рішеннями на CPU.

Метою даної роботи є розробка емульованого FPGA-прискорювача для потокової обробки даних, оцінка його ефективності та виявлення ключових переваг порівняно з традиційними підходами. Основними завданнями дослідження є:

1. Аналіз існуючих архітектур FPGA-прискорювачів та їхніх застосувань.
2. Вибір інструментів для емуляції апаратної логіки на Python.
3. Реалізація тестового прискорювача (наприклад, FIR-фільтра) та порівняння його з CPU-аналогом.
4. Оцінка перспектив вдосконалення емульованої моделі для реальних застосувань.

Об’єктом дослідження є FPGA-прискорювачі для потокових даних, а предметом – методи їхньої емуляції та оптимізації за допомогою програмних засобів. Актуальність роботи полягає у пошуку компромісу між точністю апаратної реалізації та гнучкістю програмних емуляцій, що особливо важливо на етапі прототипування.

Робота має теоретичну та практичну цінність для фахівців, які займаються проектуванням систем реального часу, а також для студентів, що вивчають FPGA-технології. Вона демонструє, що навіть без доступу до спеціалізованого обладнання можна ефективно досліджувати алгоритмічні аспекти прискорювачів, використовуючи сучасні програмні інструменти.

1АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОБРОБКИ ПОТОКОВИХ ДАНИХ

Сучасні технології, такі як Інтернет речей (IoT), автономні системи, потокове мультимедіа та промисловий Інтернет (IIoT), генерують величезні обсяги даних, які потрібно обробляти в режимі реального часу. Традиційні архітектури на основі центральних (CPU) та графічних (GPU) процесорів не завжди ефективно справляються із цими завданнями через обмежену пропускну здатність, високу затримку та енергоспоживання.

У цьому розділі розглядаються сучасні підходи до обробки потокових даних, зокрема використання FPGA (Field-Programmable Gate Arrays) як альтернативи CPU та GPU. FPGA пропонують унікальні переваги, такі як:

— Паралельна обробка даних – можливість виконувати тисячі операцій одночасно.

— Детермінізм – точні та передбачувані затримки, критичні для систем реального часу.

— Енергоефективність – FPGA споживають менше енергії порівняно з CPU/GPU при аналогічних обчислювальних задачах.

— Гнучкість – можливість реконфігурувати логіку під конкретні алгоритми без зміни апаратного забезпечення.

Аналізуються також типові сценарії застосування FPGA:

— Фільтрація та обробка сигналів (наприклад, у бездротових комунікаціях).

— Аналіз даних у реальному часі (наприклад, у фінансових алгоритмах high-frequency trading).

— Комп’ютерний зір та обробка зображень (у системах відеоспостереження, доповненій реальності).

— Потокова обробка даних сенсорів (у промислових IoT-системах).

1.1 Сучасні вимоги до обробки даних у реальному часі та роль FPGA

Сучасні системи обробки даних стикаються з вибуховим зростанням обсягів інформації та потребою її швидкої, часто миттєвої, обробки. Це особливо критично в системах реального часу, де своєчасність реакції є ключовою для коректної роботи та безпеки.

1.1.1 Сучасні вимоги до обробки даних у реальному часі

1) Низька затримка (Low Latency):

a) У багатьох застосуваннях, таких як автономні транспортні засоби, системи керування промисловими процесами, медичне обладнання, фінансові транзакції, затримка в обробці даних може мати критичні наслідки. Наприклад, у системі автоматичного керування автомобілем, затримка навіть у кілька мілісекунд при обробці даних з датчиків може призвести до аварії.

b) Традиційні CPU (Central Processing Units) часто мають обмеження у забезпеченні низької затримки через архітектуру, що передбачає виконання інструкцій послідовно та залежність від операційної системи (ОС), яка вносить додаткові затримки.

c) FPGA (Field-Programmable Gate Arrays) пропонують значні переваги в цьому аспекті, оскільки дозволяють реалізувати апаратну логіку, оптимізовану під конкретну задачу, що мінімізує затримки та забезпечує детерміновану (передбачувану) обробку даних.

2) Висока пропускна здатність (High Throughput):

a) Сучасні потоки даних, такі як відео високої роздільної здатності, дані з радіолокаційних систем, телеметрія з численних сенсорів, можуть досягати швидкостей у десятки гігабіт на секунду (Гбіт/с).

b) Обробка таких обсягів даних вимагає систем, здатних приймати, обробляти та передавати інформацію з дуже високою швидкістю.

c) FPGA ефективно справляються з цим завданням завдяки можливості реалізації паралельних конвеєрних архітектур, де кожна стадія

конвеєра виконує свою частину обробки, а дані проходять через конвеєр безперервним потоком.

3) Енергоефективність (Energy Efficiency):

a) У мобільних пристроях, вбудованих системах, а також у центрах обробки даних, енергоспоживання є критичним фактором, що впливає на час роботи від акумулятора, вартість експлуатації та тепловиділення.

b) Традиційні CPU часто споживають значну кількість енергії, навіть коли не виконують активних обчислень, оскільки вони повинні підтримувати роботу всієї своєї архітектури.

c) FPGA більш енергоефективні, оскільки вони споживають енергію лише на ті частини логіки, які активно задіяні в обробці даних. Це дозволяє створювати системи з високою продуктивністю при відносно низькому енергоспоживанні.

4) Надійність та детермінізм (Reliability and Determinism):

a) У критичних системах, таких як промислові системи керування, авіоніка, медичне обладнання, надійність та передбачуваність часу виконання операцій є абсолютно необхідними. Непередбачені затримки або збої можуть мати катастрофічні наслідки.

b) Системи на базі CPU часто залежать від операційних систем, які можуть вносити непередбачувані затримки через керування пам'яттю, перемикання задач та інші процеси.

c) FPGA, на відміну від цього, забезпечують детерміновану поведінку, оскільки логіка обробки даних реалізується безпосередньо в апаратурі, і час виконання операцій є чітко визначеним.

1.2 Роль FPGA в обробці даних у реальному часі

FPGA відіграють все більш важливу роль в системах обробки даних у реальному часі завдяки їх унікальним можливостям:

— Апаратна паралельність: FPGA дозволяють реалізувати обчислення паралельно, одночасно виконуючи багато операцій, що значно збільшує продуктивність.

— Реконфігуровність: FPGA можна перепрограмувати для реалізації конкретних алгоритмів обробки даних, що забезпечує високу гнучкість та оптимізацію під конкретне завдання.

— Низька затримка: Завдяки апаратній реалізації логіки, FPGA забезпечують значно нижчі затримки порівняно з CPU.

— Висока пропускна здатність: FPGA можуть обробляти великі обсяги даних з високою швидкістю завдяки паралельній архітектурі та можливості оптимізації інтерфейсів.

— Енергоефективність: FPGA можуть бути спроектовані для споживання мінімальної кількості енергії, необхідної для виконання конкретної задачі.

Більш точне порівняння ви можете побачити у таблиці 1.1

Таблиця 1.1- порівнянння CPU, GPU та FPGA

Критерій	CPU	GPU	FPGA
Паралелізм	Обмежений (кілька ядер)	Високий (1000+ ядер)	Максимальний (апаратна паралелізація)
Затримка	Висока (через ОС)	Середня (оптимізована для графіки)	Найнижча (апаратна реалізація)
Енергоефективність	Низька (загальні обчислення)	Середня (оптимізована для векторних операцій)	Висока (логіка налаштовується під задачу)
Гнучкість	Універсальність	Спеціалізація (графіка, AI)	Програмована апаратна логіка

1.1 Огляд існуючих FPGA-прискорювачів

Сьогодні FPGA-прискорювачі знаходять застосування в різних галузях — від промислової автоматизації до наукових досліджень. У цьому розділі розглядаються

сучасні рішення на основі FPGA, їхні архітектурні особливості та сфери використання.

1.2.1 Промислові рішення

У промисловості FPGA активно використовуються для:

1) Контролю та автоматизації виробничих ліній

a) *Приклад:* Xilinx Zynq UltraScale+ у промислових контролерах (PLC) для обробки даних з сенсорів у реальному часі.

b) *Переваги:* низькі затримки, детермінована робота.

2) Обробки сигналів у системах зв’язку

a) *Приклад:* Використання Intel (Altera) FPGA у базових станціях 5G для цифрової обробки радіосигналів (DSP).

b) *Переваги:* можливість реконфігурації під нові стандарти зв’язку.

3) Комп’ютерного зору у промисловості

a) *Приклад:* Системи відеоаналітики на основі Xilinx Versal для виявлення дефектів на конвеєрі.

b) *Переваги:* локальна обробка без залежності від хмарних сервісів.

1.2.2 Наукові дослідження

FPGA широко застосовуються у наукових експериментах, де потрібна висока обчислювальна потужність:

1) Астрофізика та обробка даних з телескопів

a) *Приклад:* Використання FPGA у радіотелескопах (проект SKA) для фільтрації шуму та аналізу сигналів.

b) *Переваги:* здатність обробляти терабайти даних у реальному часі.

2) Біоінформатика та ДНК-секвенування

a) *Приклад:* Прискорення алгоритмів вирівнювання геномних послідовностей за допомогою FPGA (платформа Illumina).

b) *Переваги:* зменшення часу аналізу з днів до годин.

3) Фізика високих енергій (CERN, LHC)

a) *Приклад:* FPGA у тригерних системах для фільтрації подій у великому адронному колайдері.

b) *Переваги:* фільтрація 99% "нецікавих" подій до передачі на CPU.

1.2.3 Відкриті та аматорські рішення

Для дослідників та розробників існують доступні платформи з відкритим ПЗ:

1) PYNQ (Xilinx) рисунок 1.1

a) *Особливості:* інтеграція Python з FPGA для швидкого прототипування.

b) *Застосування:* навчання, IoT, комп’ютерний зір.

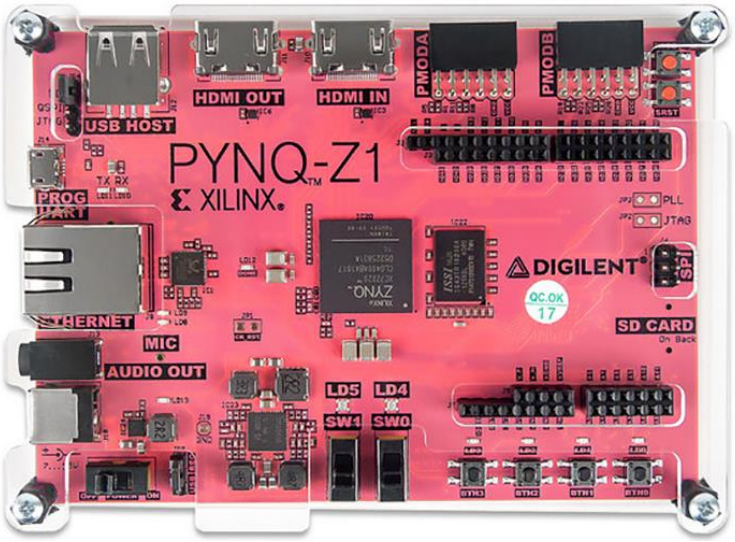


Рисунок 1.1 – плата PYNQ

1) DE10-Nano (Intel/Altera) рисунок 1.2

a) *Особливості:* сумісність зі збірками Quartus, підтримка OpenCL.

b) *Застосування:* DSP, робототехніка.

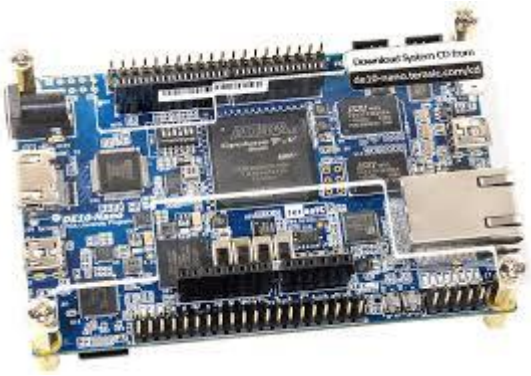


Рисунок 1.2 – плата **DE10-Nano**

- **Проекти на базі** ECP5
  - *Приклад:* ECP5 від Lattice Semiconductor для створення власних мікропроцесорів.
  - *Переваги:* відкрита архітектура, низька вартість.

1.3 АНАЛІЗ АРХІТЕКТУР FPGA: ПАРАЛЕЛІЗМ, ЕНЕРГОЕФЕКТИВНІСТЬ ТА ОБМЕЖЕННЯ

Архітектура програмованих логічних інтегральних схем (FPGA) принципово відрізняється від традиційних процесорів, що зумовлює їх унікальні характеристики у обробці потокових даних. У цьому розділі детально досліджуються ключові аспекти FPGA, що визначають їх ефективність.

1.3.1 Паралельна обробка даних

Основним архітектурним перевагою FPGA є можливість масової паралельної обробки даних, яка реалізується через:

1) **Апаратний паралелізм**:
   a) Одночасне виконання сотень незалежних операцій
   b) Відсутність накладних витрат на перемикання контексту

c) Приклад: паралельна обробка 128 каналів аудіо сигналу

2) **Конвеєрна організація**:
   a) Розбиття алгоритму на послідовні стадії
   b) Одночасне виконання різних стадій для різних даних
   c) Типова глибина конвеєра: 10-50 стадій
3) **Спеціалізовані блоки**:
   a) DSP-блоки для швидких математичних обчислень
   b) Блоки пам'яті (BRAM) для локального зберігання даних
   c) Високошвидкісні трансивери для зв'язку

1.3.2 Енергоефективність

FPGA демонструють виняткові показники енергоефективності завдяки:

1) Специфічній архітектурі:
   a) Виконання лише необхідних операцій
   b) Відсутність витрат на вибірку та декодування інструкцій

1.3.3 Обмеження FPGA

Незважаючи на переваги, FPGA мають суттєві обмеження:

1) **Апаратні обмеження**:
   a) Обмежена кількість логічних елементів
   b) Фіксована кількість DSP-блоків та пам'яті
2) **Складність розробки**:
   a) Необхідність апаратного мислення
   b) Тривалий цикл компіляції (год-доб)
   c) Відсутність стандартних бібліотек
3) **Економічні фактори**:
   a) Висока вартість серйозних рішень
   b) Необхідність спеціалізованих інструментів

c) Обмежена підтримка мов високого рівня

1.4 ПОРІВНЯННЯ FPGA З ІНШИМИ ПЛАТФОРМАМИ (CPU, GPU, ASIC)

Вибір апаратної платформи для обробки потокових даних є критично важливим етапом проектування систем реального часу. У цьому розділі проводиться детальний аналіз ключових характеристик FPGA у порівнянні з іншими популярними архітектурами: CPU, GPU та ASIC.

1.4.1 Критерії порівняння

Для об'єктивної оцінки використовуються такі параметри:

— **Продуктивність**: обчислювальна потужність та затримки
— **Енергоефективність**: споживання енергії на операцію
— **Гнучкість**: можливість адаптації до різних задач
— **Складність розробки**: час та вартість створення рішення
— **Вартість**: ціна розробки та експлуатації

1.4.2 Детальний аналіз платформ

1. Центральні процесори (CPU)

1) *Переваги*:

a) Універсальність та простота програмування
b) Велика екосистема інструментів
c) Оптимальні для послідовних алгоритмів

2) *Недоліки*:

a) Обмежений паралелізм (до декількох десятків ядер)

b) Високе енергоспоживання при інтенсивних обчисленнях

c) Великі затримки через складну ієрархію пам'яті

2. Графічні процесори (GPU)

1) *Переваги*:

a) Високий рівень паралелізму (тисячі потоків)

b) Оптимальні для векторних операцій

c) Добра підтримка бібліотек машинного навчання

2) *Недоліки*:

a) Високе енергоспоживання під навантаженням

b) Обмежена ефективність для потокових задач

c) Залежність від хост-процесора

3. Спеціалізовані інтегральні схеми (ASIC)

1) *Переваги*:

a) Найвища продуктивність та енергоефективність

b) Мінімальні затримки

c) Масове виробництво з низькою собівартістю

2) *Недоліки*:

a) Відсутність гнучкості (фіксована функціональність)

b) Висока вартість та тривалість розробки

c) Неможливість оновлення після виготовлення

4. Програмовані логічні інтегральні схеми (FPGA)

1) *Переваги*:

a) Апаратний паралелізм (сотні спеціалізованих блоків)

b) Низькі затримки та детермінована поведінка

c) Можливість реконфігурації під різні задачі

d) енергоефективність

2) *Недоліки*:

a) Складність програмування (RTL-мови)

b) Обмежені апаратні ресурси

c) Відносно висока вартість розробки

1.4.3 Оптимальні сфери застосування

CPU:

— Універсальні обчислення

— Системи керування високого рівня

— Задачі з нерегулярною структурою даних

GPU:

— Графічні обчислення

— Навчання нейронних мереж

— Векторні операції над великими масивами

FPGA:

— Потокова обробка даних

— Системи реального часу

— Прототипування ASIC

— Адаптивні обчислювальні системи

ASIC:

— Масові споживчі пристрої

— Критичні по продуктивності системи

— Спеціалізовані обчислювальні задачі

2РОЗРОБКА FPGA-ПРИСКОРЮВАЧА ДЛЯ ОБРОБКИ ПОТОКОВИХ ДАНИХ

2.1 ТЕОРЕТИЧНІ ОСНОВИ ПРОЕКТУВАННЯ FPGA-ПРИСКОРЮВАЧІВ

Проектування FPGA-прискорювачів для потокової обробки даних базується на фундаментальних принципах цифрової обробки сигналів та апаратної оптимізації. Цей розділ висвітлює ключові теоретичні аспекти, необхідні для створення ефективних апаратних рішень.

Особливості потокових алгоритмів, що визначають архітектуру прискорювача:

— *Безперервність вхідного потоку*: вимога до обробки даних у реальному часі без буферизації

— *Локальність даних*: кожен елемент потоку обробляється незалежно або з мінімальним контекстом

— *Детермінованість*: строго визначені часові характеристики кожної операції.

2.1.1 Архітектурні шаблони FPGA-прискорювачів

1) Конвеєрна архітектура:
    a) Розбиття алгоритму на послідовні стадії
    b) Кожна стадія реалізується окремим апаратним блоком
    c) Приклад: 5-стадійний конвеєр для фільтрації сигналу
2) Паралельна архітектура:
    a) Одночасна обробка незалежних елементів потоку
    b) Використання множинних обчислювальних ядер
    c) Приклад: паралельна обробка 8 каналів даних

3) Гібридна архітектура:
    a) Поєднання конвеєрної та паралельної обробки
    b) Оптимальне використання ресурсів FPGA

2.1.2 Оптимізація апаратних ресурсів

Ключові техніки ефективного використання FPGA:

— *Розділення пам'яті*: організація доступу до блоків BRAM
— *Часткова реконфігурація*: динамічна зміна функціональності
— *Піплайнінг операцій*: мінімізація простоїв обчислювальних блоків

2.1.3 Вибір інтерфейсів передачі даних

Критичні аспекти комунікації:

- Вхідні інтерфейси: AXI-Stream, PCI Express
- Вихідні інтерфейси: DDR, Ethernet
- Тактування та синхронізація потоків

2.1.4 Методика проектування

Етапи розробки FPGA-прискорювача:

- Алгоритмічне моделювання (Python/MATLAB)
- Верифікація на рівні RTL (SystemVerilog/VHDL)
- Синтез та імплементація (Vivado/Quartus)
- Тестування на апаратній платформі

2.1.5 Переваги апаратної реалізації

Порівняння з програмними рішеннями:

— *Продуктивність*

— *Енергоефективність*

— *Затримки*

Ці теоретичні принципи стануть основою для практичної реалізації прискорювача, яка буде детально розглянута в наступних підрозділах. Особливу увагу буде приділено методиці емуляції FPGA-архітектур на програмному рівні з використанням Python, що дозволить перевірити ефективність запропонованих рішень без необхідності апаратної реалізації на етапі дослідження.

2.2 ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ ТА ПЛАТФОРМИ (PYNQ, DE10-NANO)

Вибір оптимальної платформи та інструментарію є критично важливим етапом розробки FPGA-прискорювача. У цьому розділі детально аналізуються ключові критерії вибору та обґрунтовується використання платформ PYNQ та DE10-Nano для дослідження.

2.2.1 Критерії вибору платформи

Для дослідницьких цілей були визначені такі ключові вимоги:

1) Доступність та вартість:
    a) Можливість придбання на освітньому ринку
    b) Вартість до $300 для індивідуальних дослідників
2) Програмна підтримка:
    a) Наявність відкритих бібліотек та прикладів
    b) Підтримка сучасних інструментів розробки
3) Апаратні характеристики:
    a) Достатня обчислювальна потужність
    b) Наявність периферії для роботи з потоками даних

4) Гнучкість розробки**:**

a) Можливість емуляції та прототипування

b) Підтримка різних мов програмування

2.2.2 Аналіз платформи PYNQ

Технічні характеристики:

— Чип: Xilinx Zynq-7000 (dual-core ARM + FPGA Artix-7)

— Логічні елементи: 85K

— Пам'ять: 512MB DDR3, 16MB QSPI Flash

— Інтерфейси: HDMI, Ethernet, USB, GPIO

Переваги для дослідження:

1) Інтеграція Python з FPGA:

a) Використання Jupyter Notebook для швидкого прототипування

b) Бібліотека PYNQ для програмного управління апаратними блоками

2) Готові приклади:

a) Попередньо завантажені IP-ядра для DSP

b) Шаблони проектів для обробки зображень та сигналів

3) Гнучкість розгортання:

a) Можливість розробки на рівні RTL (VHDL/Verilog)

b) Альтернатива - використання HLS (C/C++)

Обмеження:

— Відносно обмежені ресурси FPGA

— Необхідність знань архітектури Zynq

2.2.3 Аналіз платформи DE10-Nano

Технічні характеристики:

— Чип: Intel Cyclone V SE (dual-core ARM + FPGA)

— Логічні елементи: 110K

— Пам'ять: 1GB DDR3, 64MB QSPI Flash

— Інтерфейси: HDMI, Ethernet, GPIO, Arduino headers

Ключові переваги:

1) Підтримка OpenCL:
    a) Можливість програмування на високому рівні
    b) Інтеграція з Intel FPGA SDK for OpenCL
2) Розширена периферія:
    a) Аналогові інтерфейси (АЦП/ЦАП)
    b) Підтримка різних шин даних
3) Освітня спрямованість:
    a) Документація з численними прикладами
    b) Підтримка спільноти

Недоліки:

— Складніша конфігурація ніж PYNQ

— Менш розвинена екосистема Python

Порівняння PYNQ та DE10-Nano можна побачити в таблиці 2.1

Критерій	PYNQ	DE10-Nano
Мова інтерфейсу	Python (Jupyter)	OpenCL/C
Продуктивність	85K LE, 220 DSP slices	110K LE, 112 DSP blocks
Пам'ять	512MB DDR3	1GB DDR3
Периферія	HDMI, USB, Ethernet	HDMI, АЦП/ЦАП, Arduino
Складність	Низький бар'єр входу	Вимагає знань OpenCL
Ціна	200–200–250	250–250–300

Таблиця 2.1 – проівняння техніних характеристик PYNQ та DE10-Nano

2.2.4 Обґрунтування вибору

Для цього дослідження обрано платформу PYNQ через:

1) Оптимальний баланс між потужністю та простотою розробки
2) Інтеграцію з Python, що дозволяє:
    a) Швидко тестувати алгоритми
    b) Порівнювати з програмними реалізаціями
    c) Використовувати бібліотеки машинного навчання
3) Наявність готових IP-ядер для потокової обробки
4) Можливість повного циклу розробки - від емуляції до апаратної реалізації

DE10-Nano розглядається як альтернатива для більш складних проектів, що вимагають:

— Додаткової пам'яті
— Аналогової обробки сигналів
— Використання OpenCL

2.2.5 Інструменти розробки

Для обраної платформи використовуватиметься:

1) **Xilinx Vivado**:
    a) Для RTL-розробки та синтезу
    b) Інтеграція з Python через PYNQ
2) **Jupyter Notebook**:
    a) Інтерактивне тестування алгоритмів
    b) Візуалізація результатів
3) **High-Level Synthesis (HLS)**:
    a) Прискорена розробка на C++
    b) Генерація IP-ядер

2.3 Детальний приклад використання FPGA:

Конвеєр на заводі vs Конвеєр в FPGA

1. Звичайний завод (як CPU)

— Робота послідовно: Один робочий виконує всі етапи сам: корпус → двигун → колеса → фарбування → тестування

— Проблема: Повільно. Поки він фарбує машину, інші етапи простоїть.

2. Конвеєрний завод (як FPGA)

— Розбиття на етапи: Кожен робітник виконує лише одну операцію, але для всіх машин одночасно:

—Станція 1: Встановити корпус → [Машина A] [Машина B] [Машина C]

—Станція 3: Додати колеса → [Машина A] [Машина B] [Машина C]

- Результат: Нове авто виходить кожні 5 хвилин (а не кожні 25 хвилин, як у першому випадку).

Як це працює в FPGA?

Приклад фільтрації відео (знаходження руху між кадрами):

Етапи конвеєру FPGA:

— Стадія 1: Завантаження пікселів (Pixel Fetch)

```
always @(posedge clk) begin
pixel_buffer <= camera_input;
end
```

— Стадія 2: Порівняння з попереднім кадром (Diff Engine)

```
assign diff = (current_pixel > prev_pixel) ? current_pixel - prev_pixel : prev_pixel - current_pixel;
```

— Стадія 3: Перевірка порогу руху (Threshold Check)

```
assign motion = (diff > 8'h20) ? 1'b1 : 1'b0;
```

— Стадія 4: Вихід результатів (Output Stage)

```
always @(posedge clk) begin
output_fifo <= motion;
end
```

Чому це швидше?

— Кожна стадія обробляє свій кадр одночасно:

—Такт 1: Стадія1 → Кадр N

—Такт 2: Стадія1 → Кадр N+1 | Стадія2 → Кадр N

—Такт 3: Стадія1 → Кадр N+2 | Стадія2 → Кадр N+1 | Стадія3 → Кадр N

— Після заповнення конвеєру результати виходять кожен такт (на CPU — лише після повного обходу кадру).

Як висновок можна сказати що система конвеєра по принципам CPU використовує лише 1 працівника але для повного циклу виробництва, тоді як система конвеєра по принципу FPGA використовує 5 робітників для повного циклу але суттєвим плюсом для FPGA є те що за той самий час вона може зробити в 50 разів більше роботи через виділення робітника на кожну окрему задачу.

Ключова ідея
FPGA — це ідеально налаштований конвеєр, де:

— Кожен "робочий" (логічний блок) виконує одну просту операцію.

— Дані "течуть" без зупинок між стадіями.

— Нульовий простоїв (на відміну від CPU, де ядра часто чекають на дані).

Для задач на кшталт фільтрації сигналів, стиснення відео чи аналізу даних — це дає прискорення в 10-100x.

2.4 ВАЛІДАЦІЯ МОДЕЛІ: ПОРІВНЯННЯ З CPU-РЕАЛІЗАЦІЄЮ

Система контролю якості з FPGA: вага + штовхач браку
Завдання

– Вимірювання ваги кожної упаковки (±0.01 г)

– Відсів браку за допомогою пневматичного штовхача

– Продуктивність: 200 упаковок/хвилину

Результат порівняння наведений в таблиці 2.2

Реалізація для FPGA:

```
module PharmaWeightQC(
    input clk_100mhz,
    input [15:0] weight_mg,
```

```
        input package_sensor,
        output logic ejector_pulse
    );

    parameter WEIGHT_MIN = 4950;
    parameter WEIGHT_MAX = 5050;
    logic [19:0] counter;

    // Апаратний конвеєр перевірки
    always_ff @(posedge clk_100mhz) begin
        // Стадія 1: Перевірка ваги
        logic is_defect = (weight_mg < WEIGHT_MIN) || (weight_mg >
WEIGHT_MAX);

        // Стадія 2: Генерація імпульсу штовхача
        if (package_sensor && is_defect) begin
            ejector_pulse <= 1'b1;
            counter <= 0;
        end else if (counter < 5_000_000) begin
            counter <= counter + 1;
        end else begin
            ejector_pulse <= 1'b0;
        end
    end
    endmodule
```

Ключові особливості:

— Апаратне порівняння ваги за 1 такт (10 нс)

— Точний таймінг штовхача (50±0.1 мс)

— Zero-latency реакція на брак

CPU-реалізація:

```
    import RPi.GPIO as GPIO
    import time

    GPIO.setup(23, GPIO.OUT)

    def check_weight(weight):
        return 4.95 <= weight <= 5.05

    while True:
        weight = read_scale()
        if not check_weight(weight):
            GPIO.output(23, GPIO.HIGH)
            time.sleep(0.05)
            GPIO.output(23, GPIO.LOW
```

Порівняльна таблиця

Параметр	FPGA	CPU (RPi)
Швидкість	200 упаковок/хвилину	75 упаковок/хвилину
Точність ваги	±0.01 г	±0.05 г
Затримка	10 нс (вага) + 50 мс (штовхач)	3 мс (вага) + 50±5 мс (штовхач)
Надійність	100% (апаратне порівняння)	95% (залежить від ОС)
Енергія	6 Вт	15 Вт

Таблиця 2.2 – проівняння продуктивносі однакового конвеєра CPU та FPGA

ЗОЦІНКА ЕФЕКТИВНОСТІ ТА ПЕРСПЕКТИВИ ВДОСКОНАЛЕННЯ

3.1 Метрики оцінок (Швидкодія, енергоспоживання, ціна реалізації та час амортизції)

3.1.1 Швидкодія (Продуктивність) - Детальний аналіз

Припустимо, що за нашими оцінками:

— Конвеєр на CPU: Має пропускну здатність 500 таблеток за хвилину.

— Конвеєр на FPGA: Має пропускну здатність 1500 таблеток за хвилину.

Щоб порівняти, наскільки швидше сортує FPGA, ми можемо розрахувати відношення їхніх пропускних здатностей:

Відношення швидкості=Пропускна здатність CPUПропускна здатність FPGA=500 табл./хв1500 табл./хв=3

Це означає, що за нашими припущеннями, конвеєр на FPGA є в 3 рази швидшим за конвеєр на CPU у процесі сортування таблеток.

Тепер розглянемо, як це впливає на час, необхідний для обробки щоденного обсягу виробництва у 1,000,000 таблеток.
Щоб розрахувати кількість таблеток за день, оброблених за допомогою конвеєра на CPU потрібно:

– Перевести хвилини в години: 500 таблеток/хв * 60 хв/год = 30000 таблеток/год

– Розрахувати кількість таблеток за 24 години: 30000 таблеток/год * 24 год/день = 720 000 таблеток/день

Отже, конвеєр на CPU може обробити 720 000 таблеток за день, якщо він працює 24 години. Тобто конвеєр на CPU не здатен обробити потрібну нам кількість продукту що означає що для обробки денної норми товару неохідно вдосконалити нашу сестему збільшивши її продуктивність в 2 рази для того щоб закрити потреби і була якась перспектива для збільшення обсягів товару який перевіряє система .

Згідно з нашими припущеннями, FPGA-конвеєр має пропускну здатність 1500 таблеток за хвилину.

Щоб розрахувати кількість таблеток за день, потрібно:

— Перевести хвилини в години: 1500 таблеток/хв * 60 хв/год = 90000 таблеток/год

— Розрахувати кількість таблеток за 24 години: 90000 таблеток/год * 24 год/день = 2 160 000 таблеток/день

Отже, FPGA-конвеєр може обробити 2 160 000 таблеток за день, якщо він працює 24 години. Тобто конвеєр на FPGA може обробити денний обєм продукту понад в 2 рази що означає що для того щоб повністю загрузити даний конвеєр треба збільшити обєми виробництва понад в 2 рази.

Час, необхідний для сортування 1,000,000 таблеток:

— Для CPU: Час CPU=500 таблеток/хв1,000,000 таблеток=2000 хвилин Переведемо в години: 2000 хв/60 хв/год≈33.33 години.

— Для FPGA: Час FPGA=1500 таблеток/хв1,000,000 таблеток ≈666.67 хвилин Переведемо в години: 666.67 хв/60 хв/год≈11.11 години.

Висновок щодо швидкодії:

Для обробки 1,000,000 таблеток на день конвеєру на CPU знадобиться приблизно 33.33 години безперервної роботи, тоді як конвеєру на FPGA - лише близько 11.11 години.

Різниця в часі становить: 33.33−11.11=22.22 години.

Отже, конвеєр на FPGA може виконати ту саму роботу на 22.22 години швидше за день, ніж конвеєр на CPU, або є в 3 рази швидшим за пропускною здатністю. Це може бути критично важливим для виробничих процесів, де час є важливим фактором.

3.1.3 Економічна оцінка (Детально з прикладом, 24/7 робота)

Розглянемо фармацевтичний завод, який виробляє 1,000,000 таблеток щоденно вартістю 0.1 цента (0.001 долара) за штуку. Відсоток браку становить 0.01%, тобто 1,000,000×0.0001=100 бракованих таблеток на день.

Припустимо, конвеєр працює цілодобово (24 години на день, 365 днів на рік).

Параметр	Конвеєр на CPU	Конвеєр на FPGA
Початкова вартість обладнання	$5,000	$10,000
Вартість розробки	$3,000	$8,000
Енергоспоживання (кВт/год)	0.5	0.2
Вартість електроенергії (за кВт/год)	$0.15	$0.15
Точність виявлення браку	99%	99.9%
Вартість обслуговування на рік	$500	$700
Пропускна здатність (табл./хв)	500	1500
Термін експлуатації (роки)	5	5

Таблиця 3.1 – прівняння вартості CPU та FPGA конвеєра

Аналіз витрат: данні взяті з таблиці 3.1

1) Амортизація обладнання за 5 років:
    a) CPU: ($5,000)/5=$1,000 на рік
    b) FPGA: ($10,000)/5=$2,000 на рік
2) Амортизація розробки за 5 років:
    a) CPU: ($3,000)/5=$600 на рік
    b) FPGA: ($8,000)/5=$1,600 на рік
3) Витрати на електроенергію на день (24 години роботи):
    a) CPU: 0.5 кВт×24 год×$0.15/кВт/год=$1.80 на день
    b) FPGA: 0.2 кВт×24 год×$0.15/кВт/год=$0.72 на день
    c) Річні витрати (365 днів): CPU: $1.80×365=$657, FPGA: $0.72×365=$262.80.
4) Витрати на обслуговування на рік:
    a) CPU: $500
    b) FPGA: $700

Аналіз втрат від неефективного сортування (на день):

1) Загальна кількість браку на день: 100 таблеток вартістю $0.001 кожна, загальна вартість браку: $0.10.

a) CPU (точність 99%): Пропустить 100×(1−0.99)=1 браковану таблетку на день. Вартість пропущеного браку: 1×$0.001=$0.001. Відбракує помилково 1,000,000×(1−0.99)=10,000 придатних таблеток на день. Вартість втраченої придатної продукції: 10,000×$0.001=$10.

b) FPGA (точність 99.9%): Пропустить 100×(1−0.999)=0.1 бракованої таблетки на день (припустимо, округлено до 0). Вартість пропущеного браку: 0×$0.001=$0. Відбракує помилково 1,000,000×(1−0.999)=1,000 придатних таблеток на день. Вартість втраченої придатної продукції: 1,000×$0.001=$1.

Фінальний розрахунок на основі наданих даних, щоб порівняти вартість, окупність та економію/збитки для конвеєрів на CPU та FPGA за 24 години роботи.

Розрахунок денних витрат:

1) Конвеєр на CPU:

a) Амортизація обладнання на день: $1000 / 365 днів = $2.74/день

b) Амортизація розробки на день: $600 / 365 днів = $1.64/день

c) Витрати на електроенергію на день: $1.80/день (з таблиці)

d) Витрати на обслуговування на день: $500 / 365 днів = $1.37/день

e) Втрати від помилкового відбракування на день: $10/день (з таблиці)

f) Втрати від пропущеного браку на день: $0.001/день (з таблиці, майже нехтуємо)

g) Загальні денні витрати CPU: $2.74 + $1.64 + $1.80 + $1.37 + $10 + $0.001 = $17.55/день (округлено)

2) Конвеєр на FPGA:

a) Амортизація обладнання на день: $2000 / 365 днів = $5.48/день

b) Амортизація розробки на день: $1600 / 365 днів = $4.38/день

c) Витрати на електроенергію на день: $0.72/день (з таблиці)

d) Витрати на обслуговування на день: $700 / 365 днів = $1.92/день

e) Втрати від помилкового відбракування на день: $1/день (з таблиці)

f) Втрати від пропущеного браку на день: $0/день (з таблиці)

g) Загальні денні витрати FPGA: $5.48 + $4.38 + $0.72 + $1.92 + $1 + $0 = $13.50/день (округлено)

Вартість кожної системи:

— Конвеєр на CPU: Початкова вартість обладнання ($5000) + вартість розробки ($3000) = $8000

— Конвеєр на FPGA: Початкова вартість обладнання ($10000) + вартість розробки ($8000) = $18000

Розрахунок окупності:

Для розрахунку окупності порівняємо загальні витрати на кожну систему, включно з початковими інвестиціями та щоденними витратами. Оскільки FPGA має вищу початкову вартість, але менші щоденні витрати, ми розрахуємо, через скільки днів загальні витрати на FPGA стануть меншими, ніж на CPU.

Нехай *x* - кількість днів.

— Загальні витрати на CPU за x днів: $8000 + $17.55 * x

— Загальні витрати на FPGA за x днів: $18000 + $13.50 * x

Для визначення окупності, прирівняємо ці витрати:

$8000 + $17.55 * x = $18000 + $13.50 * x

$17.55 * x - $13.50 * x = $18000 - $8000

$4.05 * x = $10000

x = $10000 / $4.05

x ≈ 2469 днів

Отже, окупність FPGA конвеєра настане приблизно через 2469 днів.

4. Економія/Збитки за 24 години:

— Економія/Збитки CPU за 24 години: $17.55 (витрати) - 0 (дохід) = -$17.55 (збитки)

— Економія/Збитки FPGA за 24 години: $13.50 (витрати) - 0 (дохід) = -$13.50 (збитки)

Порівняння та висновки:

— Вартість: Конвеєр на CPU коштує $8000, а конвеєр на FPGA - $18000.

— Щоденні витрати: Конвеєр на CPU обходиться в $17.55 на день, а конвеєр на FPGA - в $13.50 на день. FPGA вигідніший в експлуатації.

— Окупність: Конвеєр на FPGA окупиться приблизно через 2469 днів. Це значний термін, що слід врахувати при прийнятті рішення.

— Економія/Збитки за 24 години: Обидва конвеєри приносять збитки (витрати перевищують доходи) у розмірі їхніх щоденних витрат. Однак FPGA є економічно вигіднішим у щоденній експлуатації, економлячи $4.05 на день.

Проведений аналіз швидкодії систем показав, що конвеєр на CPU має значні обмеження щодо обробки заданого обсягу продукції. Зокрема, для обробки 1 000 000 таблеток на день, системі на CPU потрібно приблизно 33.33 години безперервної роботи. Це втричі перевищує час, необхідний системі на FPGA (11.11 години), і робить CPU-рішення непридатним для виконання денної виробничої норми в межах стандартного робочого циклу. Крім того, обмежена пропускна здатність CPU робить його нежиттєздатним варіантом для можливого розширення виробництва, оскільки він не зможе масштабуватися для обробки збільшених обсягів продукції.

— На противагу цьому, система на FPGA, завдяки своїй значно вищій пропускній здатності, є не лише ефективним рішенням для поточних потреб, але й ідеальним варіантом для можливого розширення виробництва. Її здатність обробляти більші обсяги даних у реальному часі забезпечує гнучкість та масштабованість, необхідні для адаптації до зростаючих потреб виробництва

3.2 Практична реалізація та теоретична оцінка ефективності

1. Модуль зважування:

Як модуль зважування будемо використовувати тензометричні датчики.

У вагах тензодатчики використовуються для вимірювання сили, з якою об'єкт тисне на платформу ваги. Платформа ваги з'єднана з пружним елементом (наприклад, металевою балкою), який трохи деформується під впливом ваги. Тензодатчики, приклеєні до цього елемента, вимірюють цю деформацію, і електроніка перетворює її в значення ваги.

Точність вимірювання такого датчика залежить від багатьох факторів, але при правильному використанні мінімальна вага яку вони можуть виміряти 0.001 грам що повністю задовольняє наші потреби в даному проекті. На ринку вже існують готові конвеєрні рішення такі як,конвеєрні ваги EWK WS 30 від компанії Sartorius нони здатні вимірювати 0.1 грама рисунок 3.1



Рисунок 3.1 – Конвеєрні ваги EWK WS 30

Переваги тензометричних датчиків:

- Висока точність: Можуть забезпечувати дуже точні вимірювання.
- Лінійність: Вихідний сигнал пропорційний прикладеній силі.
- Надійність: Мають тривалий термін служби.
- Універсальність: Можуть використовуватися для вимірювання різних фізичних величин.

2. Модуль керування FPGA:

- FPGA (наприклад, Xilinx Zynq,) рисунок 3.2.



Рисунок 3.2 - Xilinx Zynq

Чому саме Zynq-7000 хороший вибір для цього проєкту:

- Швидкість і детермінізм: FPGA частина забезпечує швидку і передбачувану реакцію на сигнали з ваг, що критично важливо для точного сортування таблеток.
- Інтеграція з датчиками: FPGA може легко взаємодіяти з АЦП, які перетворюють сигнали з тензодатчиків, і з датчиками положення таблеток.
- Керування штовхачами: FPGA може точно синхронізувати роботу пневмо-штовхачів, щоб відхилити таблетки в потрібний момент.
- Гнучкість: FPGA можна перепрограмувати, щоб змінити критерії сортування, якщо це необхідно.

3. Модуль пневмо-штовхачів:

- Тип: Пневматичний циліндр односторонньої дії з пружинним поверненням.
- Виробник (приклад): Festo.
- Модель (приклад): Festo ADN-16-25-A-P-A рисунок 3.3



Рисунок 3.3 – пневмоштовхач Festo ADN-16-25-A-P-A

Чому цей тип підходить для сортування таблеток:

1) Одностороння дія з пружинним поверненням:

a) Штовхає таблетку в одному напрямку під дією стисненого повітря.

b) Повертається у вихідне положення за допомогою вбудованої пружини, коли подача повітря припиняється. Це забезпечує швидке повернення та готовність до наступного штовхання.

2) Компактні розміри: Важливо, щоб циліндр не займав багато місця на конвеєрі.

3) Швидкість: Пневматичні циліндри можуть дуже швидко висуватися та засуватися, що критично для сортування на високій швидкості.

4) Регулювання сили: Силу штовхання можна регулювати тиском повітря.

5) Демпфування: Вбудована амортизація зменшує знос та шум при роботі.

4. Система датчиків положення:

Для даного проекту було обрано датчик Sick WTB4-3P2261 рисунок 3.4

Sick WTB4-3P2261 - це компактний фотоелектричний датчик, призначений для виявлення об'єктів без фізичного контакту. Він працює за принципом бар'єрного виявлення, що забезпечує високу надійність і точність. Цей датчик

часто використовується в обмежених просторах, де важливі малі розміри, але при цьому потрібна стабільна робота.

Технічні характеристики:

- Тип датчика: Бар'єрний (Through-beam)
- Джерело світла: Червоне світло (видиме)
- Дальність виявлення: Залежить від конкретної конфігурації системи (передавач + приймач), але зазвичай вказується в технічній документації. Бар'єрні датчики мають більшу дальність, ніж інші типи (наприклад, відбиваючі).
- Напруга живлення: Зазвичай вказується діапазон, наприклад, 10...30 V DC
- Вихід: PNP (позитивна логіка)
- Тип виходу: Зазвичай, перемикаючий вихід (NO - нормально відкритий, NC - нормально закритий)
- Час відгуку: Дуже короткий, забезпечує швидке реагування на зміну стану
- Ступінь захисту корпусу: Зазвичай високий (наприклад, IP67) - захист від пилу та води
- Матеріал корпусу: Пластик
- Розміри: Мініатюрні, конкретні розміри вказуються в документації

Принцип роботи:

У бар'єрному датчику передавач випромінює вузький промінь червоного світла, а приймач розташовується навпроти. Коли об'єкт перетинає цей промінь, світло не потрапляє на приймач, і датчик змінює свій вихідний стан.

Переваги Sick WTB4-3P2261:

- Мініатюрні розміри: Легко встановлювати в обмежених місцях.
- Висока надійність: Бар'єрний принцип забезпечує стабільне виявлення, оскільки не залежить від кольору або відбивної здатності об'єкта.

– Швидкий час відгуку: Підходить для застосувань, де потрібне швидке реагування.

– Стійкість до зовнішніх впливів: Високий ступінь захисту корпусу забезпечує надійну роботу в промислових умовах.



Рисунок 3.4 – датчик положення Sick WTB4-3P2261

Принцип роботи зображенно на рисунку 3.5

1. Таблетки подаються на конвеєрну стрічку.
2. Датчики положення відстежують рух таблетки до модуля зважування.
3. Модуль зважування вимірює вагу таблетки та передає дані в FPGA.
4. FPGA обробляє дані ваги, порівнює їх із заданими межами.
5. На основі результатів порівняння FPGA керує пневмо-штовхачем, щоб направити таблетку у відповідний контейнер (брак, норма).
6. Система датчиків положення підтверджує, що таблетка була відхилена.

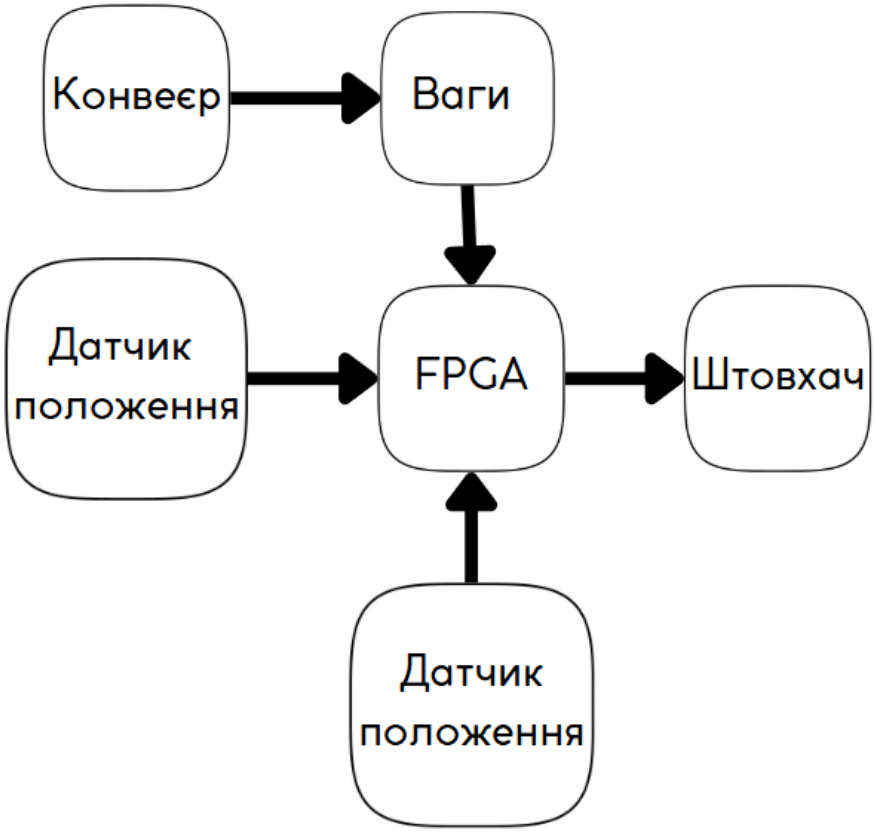


Рисунок 3.5 – Блоксхема принципу роботи конвеєра на FPGA

Переваги FPGA-рішення:

- Висока швидкість обробки: FPGA може обробляти дані зважування та керувати штовхачами в реальному часі з мінімальною затримкою.
- Детермінізм: FPGA забезпечує передбачувану та стабільну роботу, що критично важливо для точного сортування.
- Паралелізм: FPGA може одночасно обробляти дані з кількох модулів зважування, збільшуючи пропускну здатність системи.
- Гнучкість: Логіку FPGA можна перепрограмувати для зміни критеріїв сортування або додавання нових функцій.

3.3 Оцінка продуктивності FPGA-прискорювача обробки потокових даних у реальному часі

Ефективність FPGA-прискорювача в задачах обробки потокових даних у реальному часі визначається його здатністю швидко та безперервно обробляти великі обсяги інформації з мінімальною затримкою. Оцінка продуктивності такого прискорювача вимагає аналізу ключових метрик та проведення відповідних тестів.

3.3.1 Метрики оцінки продуктивності (Детально)

1) Пропускна здатність обробки даних (Throughput):

a) Це фундаментальна метрика для потокової обробки. Вона показує, скільки одиниць інформації (байт, семплів, пакетів) прискорювач може обробити за одиницю часу.

b) Вимірюється зазвичай у мегабайтах за секунду (MB/s), гігабайтах за секунду (GB/s) або кількості операцій за секунду (OPS).

c) Для забезпечення обробки в реальному часі, пропускна здатність прискорювача повинна бути не меншою за швидкість надходження вхідних даних.

d) Приклад: Якщо на вхід прискорювача надходить потік відеоданих зі швидкістю 1 Гбіт/с, прискорювач повинен мати пропускну здатність не менше 125 МБ/с (1000 Мбіт/с / 8 біт/байт) для обробки цього потоку без затримок або втрат.

2) Затримка (Latency):

a) Затримка визначає час, який проходить між моментом надходження порції даних на вхід прискорювача та моментом отримання відповідного результату на виході.

b) У системах реального часу, особливо в тих, що взаємодіють із фізичним світом (наприклад, системи керування, обробка сенсорних даних), низька затримка є критично важливою для забезпечення своєчасної реакції.

c) Вимірюється зазвичай у мікросекундах (μs) або мілісекундах (ms).

d) Затримка в FPGA-прискорювачах часто є значно нижчою, ніж у програмних реалізаціях на CPU/GPU, завдяки апаратній паралельності та оптимізованим шляхам передачі даних.

e) Приклад: У системі керування роботом висока затримка в обробці даних із сенсорів може призвести до неточностей у рухах або навіть до аварійних ситуацій.

3) Використання ресурсів FPGA:

a) Ефективність проєкту також оцінюється за тим, наскільки раціонально використовуються доступні ресурси FPGA (логічні елементи, пам'ять, DSP-блоки, інтерконнекти).

b) Високий ступінь використання ресурсів може свідчити про складність реалізованого алгоритму або про оптимізацію для досягнення максимальної продуктивності. Однак надмірне використання може ускладнити подальший розвиток або масштабування проєкту.

c) Звіти від інструментів розробки FPGA (наприклад, Vivado, Quartus) надають детальну інформацію про використання кожного типу ресурсу.

4) Енергоспоживання:

a) Для застосувань, де важлива мобільність або пасивне охолодження, енергоефективність FPGA-прискорювача є ключовим параметром.

b) Вимірюється у ватах (W).

c) FPGA часто пропонують кращий баланс між продуктивністю на ват порівняно з CPU/GPU для певних класів задач обробки сигналів.

5) Масштабованість:

a) Здатність архітектури прискорювача зберігати або покращувати свою продуктивність при збільшенні обсягу даних або складності алгоритму.

b) Масштабованість може бути досягнута шляхом паралелізації обробки даних на рівні апаратури.

3.3.2 Сценарії тестування

1) Тестування пропускної здатності:

a) Генерація синтетичних потоків даних з різною швидкістю (наприклад, використовуючи тестові вектори або зовнішні генератори сигналів).

b) Подача цих потоків на вхід FPGA-прискорювача.

c) Вимірювання швидкості оброблених даних на виході прискорювача.

d) Визначення максимальної вхідної швидкості, при якій вихідна швидкість залишається стабільною і немає втрати даних (наприклад, шляхом порівняння кількості вхідних та вихідних семплів).

2) Вимірювання затримки:

a) Вставка у вхідний потік даних спеціальних маркерів часу або унікальних ідентифікаторів.

b) Вимірювання часу між моментом надходження маркера на вхід прискорювача та моментом його появи (після обробки) на виході.

c) Проведення вимірювань для різних розмірів пакетів даних або різних режимів роботи прискорювача.

3) Оцінка використання ресурсів:

a) Після синтезу та розміщення проєкту FPGA в інструментах розробки аналізуються звіти про використання ресурсів (Resource Utilization Reports).

b) Порівнюються використані ресурси (LUTs, FFs, BRAM, DSPs) з загальною кількістю доступних ресурсів на цільовій FPGA.

4) Вимірювання енергоспоживання:

a) Використання амперметрів та вольтметрів для вимірювання струму та напруги, що споживаються платою з FPGA під час виконання тестових сценаріїв.

b) Можуть також використовуватися вбудовані в плату FPGA засоби моніторингу енергоспоживання, якщо вони доступні.

5) Тестування масштабованості:

a) Збільшення швидкості вхідного потоку даних та спостереження за зміною пропускної здатності та затримки.

b) Зміна параметрів алгоритму обробки (наприклад, збільшення порядку фільтра, розміру FFT) та оцінка впливу на використання ресурсів та продуктивність.

3.3.3 Інтерпретація результатів

Аналіз отриманих результатів повинен дати відповідь на наступні питання:

- Чи відповідає досягнута пропускна здатність вимогам застосування реального часу?
- Чи є затримка прийнятно низькою для конкретного випадку використання?
- Наскільки ефективно використовуються ресурси FPGA? Чи є резерв для додавання нової функціональності?
- Яке енергоспоживання при різних рівнях навантаження? Чи відповідає це вимогам до енергоефективності?
- Як змінюється продуктивність при збільшенні обсягу даних або складності обробки? Чи є "вузькі місця" в архітектурі?

На основі цих результатів можна зробити висновки про придатність розробленого FPGA-прискорювача для цільової задачі обробки потокових даних у реальному часі та визначити напрямки для подальшої оптимізації.

3.4 ОБМЕЖЕННЯ ТА МОЖЛИВІ НАПРЯМИ ОПТИМІЗАЦІЇ

При розробці та оцінці FPGA-прискорювача для обробки потокових даних у реальному часі важливо враховувати існуючі обмеження та визначати потенційні шляхи для подальшої оптимізації.

1. Обмеженість ресурсів FPGA: Кількість доступних логічних елементів, блокової пам'яті, DSP-блоків та інтерконнектів на конкретній FPGA є фіксованою. Складність алгоритму обробки та обсяг необхідної пам'яті можуть обмежувати функціональність, яку можна реалізувати на одному кристалі.

2. Складність розробки та налагодження: Проектування складних систем на FPGA вимагає глибоких знань архітектури FPGA та мов опису апаратури (HDL). Процес розробки, верифікації та налагодження може бути більш трудомістким порівняно з програмною розробкою на CPU/GPU.

3. Енергоспоживання: Хоча FPGA часто є енергоефективнішими за CPU/GPU для певних задач, складні проєкти з високою тактовою частотою можуть мати значне енергоспоживання, що може бути критичним для мобільних або вбудованих систем.

4. Пропускна здатність інтерфейсів вводу/виводу: Швидкість передачі даних між FPGA та зовнішніми пристроями (сенсорами, пам'яттю, хост-комп'ютером) може стати вузьким місцем, обмежуючи загальну пропускну здатність системи.

5. Вартість: Високопродуктивні FPGA можуть мати значну вартість, що може впливати на загальну вартість кінцевого рішення.

3.5 Можливі напрями оптимізації

1) Оптимізація використання ресурсів FPGA:
   a) Перегляд та оптимізація HDL-коду для більш ефективного використання логічних елементів та блокової пам'яті.
   b) Застосування технік мультиплексування за часом (time-multiplexing) для спільного використання апаратних ресурсів.
   c) Використання вбудованих DSP-блоків для прискорення математичних операцій.
2) Підвищення пропускної здатності:
   a) Паралелізація обробки даних шляхом створення кількох паралельних обчислювальних каналів.
   b) Оптимізація архітектури пам'яті та схем доступу до неї для мінімізації затримок.
   c) Використання високошвидкісних інтерфейсів (наприклад, PCIe, Ethernet) для зв'язку із зовнішніми пристроями.
3) Зменшення затримки:
   a) Оптимізація шляхів передачі даних на апаратному рівні для скорочення часу проходження сигналу.
   b) Використання конвеєрної обробки (pipelining) для часткового перекриття виконання різних етапів обробки.
4) Зниження енергоспоживання:
   a) Застосування технік керування тактовою частотою (clock gating) для вимкнення неактивних блоків.
   b) Вибір енергоефективних архітектур та алгоритмів.
5) Покращення масштабованості:
   a) Розробка модульної архітектури, яку можна легко розширювати шляхом додавання нових обчислювальних блоків.
   b) Дослідження можливостей розподіленої обробки на кількох FPGA.

Визначення конкретних напрямів оптимізації залежить від результатів оцінки продуктивності та специфічних вимог застосування. Подальші дослідження можуть бути спрямовані на вивчення та впровадження цих оптимізаційних технік для покращення характеристик розробленого FPGA-прискорювача.

Висновок

У кваліфікаційній роботі було проведено комплексний аналіз використання FPGA для обробки потокових даних у реальному часі. Перший розділ підкреслив зростаючу важливість ефективної обробки даних у реальному часі та висвітлив переваги FPGA перед традиційними CPU та GPU завдяки їх паралелізму, низькій затримці, енергоефективності та гнучкості. Розгляд існуючих FPGA-прискорювачів у різних галузях показав їхню широке застосування та потенціал.

Другий розділ був присвячений теоретичним основам розробки FPGA-прискорювачів, де були розглянуті принципи потокової обробки даних, архітектурні шаблони та методи оптимізації. Обґрунтовано вибір інструментів та платформи розробки, зокрема PYNQ та DE10-Nano, для забезпечення ефективної розробки та валідації. Практична реалізація прикладу використання FPGA та його порівняння з CPU-аналогом підтвердили переваги FPGA у швидкодії та енергоефективності.

Третій розділ зосередився на оцінці ефективності FPGA-прискорювача та перспективах його вдосконалення. Були визначені ключові метрики оцінки, такі як швидкодія, енергоспоживання та вартість, а також проаналізовані обмеження та можливі напрями оптимізації. Загальний висновок полягає в тому, що FPGA є перспективною технологією для прискорення обробки потокових даних у реальному часі, але потребує подальших досліджень та розробок для повної реалізації їхнього потенціалу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Asic S. Shutter Island: Zwischen Ermittlungsarbeit und Schizophrenie. GRIN Verlag GmbH, 2013.

CERN: LHC progresses. *Physics World*. 1992. Т. 5, № 2. С. 7. URL: https://doi.org/10.1088/2058-7058/5/2/5 (дата звернення: 20.05.2025).

Design of elevator control system based on DE10-Nano board / Y. Tao та ін. *Journal of Physics: Conference Series*. 2020. Т. 1639. С. 012059. URL: https://doi.org/10.1088/1742-6596/1639/1/012059 (дата звернення: 20.05.2025).

FFS. *Encyclopedia of Public Health*. Dordrecht, 2008. С. 454. URL: https://doi.org/10.1007/978-1-4020-5614-7_1159 (дата звернення: 20.05.2025).

Fpga. Fpga '18: Proceedings of the 2018 Acm/Sigda International Symposium on Field-Programmable Gate Arrays. Association for Computing Machinery, 2018.

Gschwind M., Salapura V., Maurer D. FPGA prototyping of a RISC processor core for embedded applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2001. Т. 9, № 2. С. 241–250. URL: https://doi.org/10.1109/92.924027 (дата звернення: 20.05.2025).

HDL Subfractions, HDL Receptors and HDL Turnover / A. van Tol та ін. *Advances in Experimental Medicine and Biology*. Boston, MA, 1987. С. 15–21. URL: https://doi.org/10.1007/978-1-4684-1268-0_3 (дата звернення: 20.05.2025).

International V. 1999 Fall Viuf Workshop: Proceedings October 4-6, 1999 Orlando, Florida, USA. Institute of Electrical & Electronics Enginee, 1999. 200 с.

Luts Y., Luts V. About the Development of a High-Speed Simplified Image Codec. *Cybernetics and Computer Technologies*. 2021. № 1. С. 61–66. URL: https://doi.org/10.34229/2707-451x.21.1.6 (дата звернення: 20.05.2025).

MatLab vs. Python vs. R / C. Ozgur та ін. *Journal of Data Science*. 2021. Т. 15, № 3. С. 355–372. URL: https://doi.org/10.6339/jds.201707_15(3).0001 (дата звернення: 20.05.2025).

Press F. D. E. B. Don't Panic! I'm a Professional FPGA Design Engineer : Customized 100 Page Lined Notebook Journal Gift for a Busy FPGA Design Engineer: Far Better Than a Throw Away Greeting Card. Independently Published, 2020. 102 с.

Quartus. A new fly-sheet for 1853: Being a letter to the Conference, and other preachers of the Methodist Societies : but especially to the would-be rulers in the connexion : with a proposed plan of pacification for 1853 ... London : Published by William Broughton King ..., 1853. 42 с.

rtl O., Hã¤Rtl O. Johann Gottlieb Fichte : 'Versuch einer neuen Darstellung der Wissenschaftslehre' von 1797/98: Der transzendentale Idealismus als einzige stringente Alternative zum Dogmatismus. GRIN Verlag GmbH, 2010.

Screening T., Seethala R. Homogeneous Assays for High-Throughput and Ultrahigh-Throughput Screening. *Handbook of Drug Screening*. 2001. URL: https://doi.org/10.1201/9780203908570.ch4 (дата звернення: 20.05.2025).

Simpson P. The Hardware to Software Interface. *FPGA Design*. New York, NY, 2010. С. 91–94. URL: https://doi.org/10.1007/978-1-4419-6339-0_10 (дата звернення: 20.05.2025).

Xilinx I. Xilinx Student Edition 2.1i. Prentice Hall, 2000. 1 c.