МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Кафедра «Електронні обчислювальні машини»

«ДО ЗАХИСТУ»

Завідувач кафедри
Жуковицький І.В.
(підпис) (ПІБ)
«____» _____ 20____р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань 12 Інформаційні технології
(шифр) (назва)

Спеціальність 123 Комп'ютерна інженерія
(код) (повна назва)

Тема Розробка і дослідження реконфігуруємого процесора для хмарних обчислень з використанням ПЛІС

Theme Development and research of reconfigurable processor for cloud computing using FPGA

Керівник дипломного проекту	_____ (посада)	_____ (підпис)	Шаповалов В. О. (ПІБ)
Консультант розділу з БЖД	_____ (посада)	_____ (підпис)	Музикін М. І. (ПІБ)
Нормоконтролер	_____ (посада)	_____ (підпис)	Шаповалов В. О. (ПІБ)
Студент групи	КС1921 (група)	_____ (підпис)	Оганесов Б. А. (ПІБ)
Student			Ohanesov Borys (family name)

Дніпро
2020

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Факультет _Комп'ютерних технологій і систем__ кафедра _____ЕОМ_____

Спеціальність __Комп'ютерна інженерія_____

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

_____

(підпис)

«___»_____20_ р.

ЗАВДАННЯ

до дипломної роботи на здобуття освітнього ступеня ____магістр_____

(освітнього ступеня)

студента групи _КС1921 Оганесова Бориса Артуровича_____

(номер групи) (ПІБ)

1 Тема дипломної роботи Розробка і дослідження реконфігуруємого процесора для хмарних обчислень з використанням ПЛІС_____

_____

_____

_____

затверджена наказом по університету від «17» січня 2020 р. № 57 .

2 Термін подання студентом закінченої роботи 11.12.2020 р.

3 Вихідні дані до дипломної роботи_____

Необхідно проаналізувати можливості динамічної реконфігурації ПЛІС і спроектувати процесор з найбільш ефективним типом реконфігурації та ядром під хмарні обчислення

4 Зміст пояснювальної записки (перелік питань до розробки) _____

Вступ

1. Принципи роботи хмарних обчислень

2. Огляд та вибір ПЛІС з можливістю динамічного реконфігурування

3. Розробка структури процесору

4. Реалізація та дослідження процесору

5. Охорона праці та безпека в надзвичайних ситуаціях

Висновки

5 Перелік креслень (демонстраційного матеріалу) _____

Моделі обслуговування хмарних обчислень

Функціональна схема процесору

Часові діаграми роботи процесору

_____

_____

6 Розділи та консультанти

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці в надзвичайних ситуаціях	Музикін М.І.		

КАЛЕНДАРНИЙ ПЛАН

Назва розділу	Термін виконання	Обсяг розділу, %
Вступ	25.01.2020 р.	2%
Принципи роботи хмарних обчислень	14.02.2020 р.	8%
Огляд та вибір ПЛІС з можливістю динамічного реконфігурування	23.03.2020 р.	15%
Розробка структури процесору	29.04.2020 р.	15%
Реалізація та дослідження процесору	20.10.2020 р.	40%
Охорона праці та безпека в надзвичайних ситуаціях	14.11.2020 р.	10%
Висновки	27.11.2020 р.	5%
Оформлення диплому, підготовка демонстраційних матеріалів та доповіді	11.12.2020 р.	5%

Дата видачі завдання: «___»_____ 20___ р.

Керівник дипломної роботи _____ Шаповалов В. О.
(підпис) (ПІБ)

Завдання прийняв до виконання _____ Оганесов Б. А.
(підпис) (ПІБ)

РЕФЕРАТ

Робота виконана на 74 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 34 найменування. У роботі наведено 18 рисунків та 9 таблиць.

Актуальним напрямом підвищення ефективності обробки інформації є побудова динамічних реконфігурованих процесорів на базі ПЛІС. Загальною концепцією підвищення ефективності запроектованих таких процесорах є сумісний програмно-апаратний підхід, який ґрунтується на певних функціях хмарних обчислень.

В дипломній роботі використовується часткова динамічна реконфігурація для декількох створених компонентів процесору.

Метою дипломної роботи є розробка і дослідження реконфігуруємого процесора для хмарних обчислень на базі ПЛІС.

Поставлена мета досягається розв’язанням таких основних задач:

- аналіз існуючих підходів хмарних обчислень;
- розробка структури реконфігуруємого процесору;
- реалізація процесору на практиці і його дослідження.

Ключові слова: РЕКОНФІГУРОВАНИЙ ПРОЦЕСОР, ХМАРНІ ОБЧИСЛЕННЯ, ПЛІС, XILINX, VHDL, FPGA, ARTIX-7, NEXYS A7, САПР, VIVADO DESIGN SUITE.

ABSTRACT

The work is performed on 74 sheets, contains 3 appendices and a link to the list of used literature sources from 34 titles. The paper contains 18 figures and 9 tables.

An important direction to increase the efficiency of information processing is the construction of dynamic reconfigurable processors based on FPGA. The general concept of improving the efficiency of designed such processors is a compatible software and hardware approach, which is based on certain functions of cloud computing.

The thesis uses a partial dynamic reconfiguration for several created processor components.

The purpose of the thesis is to develop and study a reconfigurable processor for cloud computing based on FPGA.

This goal is achieved by solving the following main tasks:

- it is necessary to analyze the existing approaches to cloud computing;
- design the structure of the reconfigurable processor;
- Implement the processor in practice.

Keywords: RECONFIGURED PROCESSOR, CLOUD COMPUTING, XILINX, VHDL, FPGA, ARTIX -7, NEXYS A7, VIVADO DESIGN SUITE, CAD.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IaaS - Infrastructure as a Service

SaaS - Software as a Service

PaaS - Platform as a Service

ПЗ - Програмне забезпечення

ЦОД - Центр обробки даних

RC - Reconfigurable Computer

РОС - Реконфігуровані обчислювальні системи

ЧР - Часткова реконфігурація

ЧДР - Часткова динамічна реконфігурація

РЧЗ - Реконфігурація часу зупинки

ПЛІС - Програмована логічна інтегральна схема

FPGA - Field programmable gate array

SoC - System on a Chip

САПР - Система автоматизованого проектування

LUT - LookUp Table

DSP - Digital Signal Processor

ВСТУП

Переважна більшість реконфігурованих обчислювальних систем працюють на статичній реконфігурації кристалів ПЛІС, що обумовлює широке використання методів статичного планування обчислювального процесу та розподілу задач на обчислювальне середовище. Використання статичної структури є досить ефективним в межах цілей, що перед ними ставляться, але у той же час необхідно витрачати значний час на перебудову системи, що інколи є критичним критерієм при виборі типу конфігурування. Технології постійно розвиваються і сьогодні все частіше ПЛІС використовують в хмарних обчисленнях, наприклад компанія Amazon пропонує брати в оренду хмарні сервери, які побудовані за технологією FaaS (FPGA as a Service).

В останні роки було розроблено багато ПЛІС, які підтримують можливість динамічного реконфігурування, це створило передумови та нові горизонти для розвитку та підвищення ефективності паралелізму у ПЛІС та зменшило час на їх перепрограмування, а також були відкриті нові можливості реконфігурації ПЛІС (наприклад Artix-7) в динамічному режимі (Run Time). В дипломній роботі використовується часткова динамічна реконфігурація для декількох створених компонентів процесору.

Наразі загальною концепцією для підвищення ефективності ПЛІС є сумісний програмно-апаратний підхід, який будується на апаратному пришвидшенні деяких функцій.

Проте потрібно мати на увазі, що використання динамічної реконфігурації небажано, якщо система планується використовуватись в оборонному секторі, проектуванні компонентів для космічних ракет та інших об’єктів критичної інфраструктури, де будь-яка помилка або втручання із зовні може призводити до небажаних (катастрофічних) наслідків.

Метою дипломної роботи є розробка і дослідження реконфігуруємого процесора для хмарних обчислень на базі ПЛІС.

Поставлена мета досягається розв’язанням таких основних задач:

- аналіз існуючих підходів хмарних обчислень та можливості реконфігурацій ПЛІС на даний час;
- розробка структури реконфігуруємого процесору;
- реалізація процесору на практиці на мові VHDL і його дослідження.

1 ПРИНЦИПИ РОБОТИ ХМАРНИХ ОБЧИСЛЕНЬ

1.1 Поняття хмара та хмарне обчислення

Хмарні обчислення (англ. cloud computing) – це модель забезпечення зручного доступу на вимогу до деякого загального фонду обчислювальних ресурсів, що можна налаштувати за власними потребам (наприклад, мережам передачі даних, серверам, пристроям зберігання даних, додаткам та сервісам – як разом, так і поодинці), що можуть бути своєчасно надані та звільнені з мінімальними експлуатаційними витратами та зверненнями до провайдерів даних послуг [1].

Під хмарними технологіями варто розуміти модель системи процедур, які забезпечують цілеспрямовану зміну матеріальних і віртуальних об’єктів.

Певним аналогом хмари є мейнфрейми(mainframe), між ними дійсно є багато чого спільного, але ключова відмінність хмари від мейнфреймів в тому, що її обчислювальна потужність теоретично не обмежена, іншою не менш принциповою відмінністю є те, що термінали для мейнфреймів служили з метою взаємодії користувача з запущеним на обробку завданням. В хмарі ж термінал сам є потужним обчислювальним пристроєм. Він здатен не тільки накопичувати проміжну інформацію, а й здійснювати безпосереднє управління глобальною системою обчислювальних ресурсів.

На сьогодні великі обчислювальні хмари складаються з тисяч серверів, розміщених в центрах обробки даних (ЦОД). Вони дають ресурси для десятків тисяч застосунків, якими одночасно користуються мільйони користувачів. Хмарні технології є зручним інструментом для підприємств, яким занадто дорого утримувати власні CRM(Customer Relationship Management), ERP(Enterprise Resource Planning) або інші сервери, що вимагають придбання і налаштування додаткового обладнання. І ринок не стоїть на місці, а пропонує багато рішень для користувачів і бізнесу. Одним з цих рішень є використання grid computing це нова технологія, яка надає можливість використовувати паралельно цілий ряд обчислювальних ресурсів. Вона передбачає можливості налагодження спільної і зачасту паралельної роботи ряду обчислювальних

апаратів відповідно до схеми, розробленої користувачем цієї технології. Тут треба додати, що масштабування буває двох видів: горизонтальним та вертикальним. Горизонтальне масштабування передбачає додавання паралельних обчислювальних пристроїв, а вертикальне - збільшення кількості оперативної та постійної пам’яті.

Серед приватних користувачів значне поширення завдяки своїй зручності поступово отримують такі хмарні послуги, як, наприклад, такі що надаються сервісами Google («Диск», «Документи», «Календар» та ін.).

Зрозуміло, що використовувати хмарні обчислення набагато зручніше, вони мають більшу надійність, бо відмова одного із серверів, який надає послуги, не веде до падіння системи, прикладом є система перевірки унікальності тексту Unicheck, яка працює на AWS (Amazon Web Services) і надає гарантію роботи без збоїв - 99,9%. Основним недоліком є повна залежність від постачальника хмарних послуг. Фактично підприємство (користувач) стає заручником провайдера сервісів і провайдера доступу в мережу Internet. І хоча надійність постачальників послуг дедалі збільшується, самій компанії та/або користувачу потрібно теж думати про забезпечення надійності своїх даних, шляхом створення бекапів (дублікати).

1.2 Різновиди хмар за моделлю розгортання

За моделлю розгортання хмари поділяють на загальнодоступні, приватні та гібридні.

Приватні хмари – це внутрішня хмарна інфраструктура і служби підприємства. Такі хмари розгортаються в межах корпоративної мережі, наприклад фінансової установи. Інколи навіть приватні хмари передають на керування і підтримку зовнішнім підрядчикам, але загалом організація сама керує приватною хмарою, наймаючи до себе відповідний персонал. Інфраструктуру можна розмістити або в приміщеннях замовника, або у зовнішнього оператора, або частково у замовника і частково у оператора. Найкращим, з точки зору безпеки і надійності, варіант приватної хмари –

хмара, яка розгорнута на території організації, що обслуговується і контролюється її співробітниками. Приватні хмари володіють здебільше схожі із загальнодоступними, але з однією важливою особливістю: підприємство власноруч займається установкою і підтримкою хмари. Вартість створення, складність розгортання і подальшого обслуговування та експлуатації може бути дуже дорогим, все буде залежати від її масштабів і загалом ці витрати перевищують вартість користування загальнодоступними хмарами.

Загальнодоступні (публічні) хмари представляють собою хмарні послуги, які надаються постачальником. Вони знаходяться за межами корпоративних мереж.

Користувачі цих хмар не мають можливості управляти конфігурацією хмари або обслуговувати її, вся відповідальність і повнота керування перекладена на власника такої хмари. Постачальник хмарних послуг приймає зобов'язання з установки, управління, надання та обслуговування ПЗ, інфраструктури застосунків чи фізичної інфраструктури. Клієнти сплачують тільки ресурси, якими користуються. Клієнтом таких сервісів може бути будь-яка компанія або індивідуальний користувач. На сьогодні компаній, що пропонують такі послуги, дуже багато, тому і ціни на них постійно знижуються і навіть невеликій компанії можна взяти недорого базовий пакет для користування по привабливій ціні. Прикладами таких онлайн-сервісів є: Amazon EC2, Salesforce.com, Microsoft Office Web, Google Apps / Docs.

Гібридні хмари це поєднання загальнодоступних і приватних хмар. Гібридна хмара надає послуги, частину яких можна віднести до загальнодоступних, а частину – до приватних. Зазвичай такий тип хмар використовують, коли компанія має сезонні періоди активності. Інакше кажучи, як тільки внутрішня IT-інфраструктура не може виконати поточні завдання, частина потужностей перекидається на публічні хмари (наприклад, великі обсяги статистичної інформації, які в необробленому вигляді не уявляють цінності для організації), а також для надання доступу користувачам (до приватної хмари) через публічну хмару.

Добре спроектована гібридна хмара може обслуговувати критичні з точки зору безпеки процеси, такі як отримання платежів від клієнтів і більш другорядні. Головним недоліком цього типу хмари є складність ефективного створення подібних рішень і управління ними. Необхідно отримувати послуги з різних джерел і організувати їх так, якби це було єдине джерело. Тісна взаємодія між приватним і загальнодоступним компонентами може лише ускладнити рішення. Оскільки це порівняно нова архітектурна концепція в сфері хмарних обчислень, для такої моделі з'являються нові підходи, практичні рекомендації з налаштування та обслуговування, відтак її широке поширення може затягнутися до того моменту, поки вона не буде більше вивчена.

На таблиці (див. таблицю 1.1) показано, як в залежності від виду хмарного сервісу, ним можуть володіти та розпоряджатися як провайдер, так і користувач, або і той та інший і відмінності прав доступу до відповідних ресурсів.

Таблиця 1.1 – Різновиди хмарних ресурсів та види їх обслуговування та управління [1]

Вид хмари	**Ким обслуговується інфраструктура**	**Хто с власником інфраструктури**	**Де знаходиться інфраструктура**	**У кого е доступ**
Публічна	Зовнішнім провайдером	Зовнішній провайдер	У зовнішнього провайдера	У будь-якого користувача
Приватна	Користувачем або зовнішнім провайдером	Користувач або зовнішній провайдер	У користувача, інколи у зовнішнього провайдера	У авторизованого користувача
Гібридна	Користувачем і зовнішнім провайдером	Користувач і зовнішній провайдер	У зовнішнього провайдера і у користувача	У авторизова-них і у будь-яких зовнішніх користувачів

Хмара – це не суто віртуалізація. І хоча віртуалізація серверів та інфраструктури має важливий фундамент для приватних хмарних обчислень, сама по собі віртуалізація і управління віртуалізованим середовищем це ще не є приватною хмарою. Так, віртуалізація дозволяє краще структурувати, об'єднувати в пул і динамічно надавати ресурси інфраструктури, але таке

середовище ще не є хмарним, потрібно щоб були і інші складові: віртуальні машини, операційні системи(ОС) або контейнери для емуляції їх, як наприклад Docker. відповідне програмне забезпечення.

1.3 Характеристики хмар

Сьогодні найбільш швидко зростаючим сегментом хмарних обчислень є IaaS. Вона надає найбільш низькорівневі ресурси ЦОД у простий для користувача формі, проте фундаментально не змінює принципи роботи. Аби створити нові додатки, що спочатку призначені для хмари і надають нові послуги, які можуть багато в чому відрізнятися від того, що давали колишні застосунки, розробникам більш зручно використовувати PaaS. Приватна хмара може перестати бути приватною. З одного боку, приватна хмара надає усі переваги хмари: швидкість перебудови, масштабованість і ефективність, позбавляє від багатьох загроз безпеки, потенційних і реальних, які є характерними для загальнодоступних хмар. З іншого боку, з плином часу рівень обслуговування, безпека й контроль дотримання вимог у загальнодоступних хмарних сервісах буде дедалі підвищуватися. Тому деякі приватні хмари, вірогідно, цілком перейдуть в категорію загальнодоступних. А більшість сервісів приватних хмар, швидше за все, будуть еволюціонувати у гібридні хмарні сервіси, дедалі розширюючи доступні можливості за рахунок користування загальнодоступними хмарними послугами та іншими сторонніми ресурсами.

Національний Інститут стандартів і технологій NIST (National Institute of Standards & Technology, USA) надає наступні визначення характеристикам хмар:

«Самообслуговування на вимогу» є ключовою і, навіть, основною характеристикою хмарних обчислень. Якщо розглядати IT як складний ланцюжок поставок з застосунком і кінцевим користувачем, то здатність до самозабезпечення ресурсами в типових IT-середовищах фундаментально порушує робочий процес і процеси, які розвивалися як функція корпоративних

ІТ за останні кілька десятиліть. Це включає в себе робочі процеси, пов’язані із закупівлею пристроїв зберігання, серверів, мережевих вузлів, ліцензій на ПЗ і т. д. Хмарні архітектури проектуються і створюються з урахуванням необхідності самозабезпечення. Ця умова має на увазі використання складних програмних рамок чи порталів для управління функціями ініціалізації і допоміжного офісу. Історично склалося так, що брак комерційного готового ПЗ, розробленого для автоматизації хмарних обчислень, змусив багато компаній створювати власні портали і структури для підтримки таких функцій.

«Широкосмуговий доступ до мережі». Доступ до хмарних ресурсів можна отримати використовуючи декілька типів пристроїв. До їх числа входять не тільки найпоширеніші пристрої (ноутбуки, робочі станції і т. д.), але і мобільні телефони, тонкі клієнти тощо. Розрахункові ресурси ще сорок років тому були мізерними і дорогими. Їх використання було обмеженим через пріоритетності та важливості робочого навантаження з метою збереження цих ресурсів. Аналогічно, і ресурси мережі також були обмежені. Мережі на основі Internet-протоколу (IP) були широко поширені в той час, тому не існувало мереж з високою пропускною здатністю і низькою затримкою. Згодом витрати, пов’язані з мережею (наприклад, витрати, пов’язані з обчисленнями і зберіганням даних), скоротилися в зв’язку з масштабністю виробництва і комерціалізацією відповідних технологій. У міру збільшення пропускної здатності мережі відповідно збільшився доступ до неї та її масштабованість. «Широкосмуговий доступ до мережі» можна розглядати як характеристику хмарних обчислень, так і як фактор, що сприяє їх розвитку.

«Об’єднання ресурсів у пули» є фундаментальною передумовою масштабованості в хмарі. Без об’єднаних обчислень мереж і сховищ постачальник послуг має надавати послуги через кілька ізольованих чи дискретних, незалежних ресурсів з невеликою кількістю з’єднань чи без них. При збільшенні клієнтської бази у постачальника спостерігається неминуче

збільшення експлуатаційних витрат, що потрібно буде зменшувати за рахунок конфігурацій обладнання та різних програмних рішень.

«Вимірюваний сервіс» означає, що користування цими «об'єднаними ресурсами» контролюється і повідомляється споживачу, забезпечуючи тим самим прозорість норм споживання і витрат.

Ще однією характеристикою є «швидка еластичність». Еластичні обчислення мають велике значення для скорочення витрат і часу виходу на ринок. Поняття гнучких ресурсів в ланцюжку поставок IT настільки бажано, що, наприклад, Amazon назвала власну хмарну платформу «Elastic Compute Cloud» (EC2).

1.4 Огляд моделей обслуговування хмарних технологій

Наразі прийнято виділяти три базові моделі обслуговування хмарних технології, це – послуги інфраструктури, послуги платформи і послуги застосунків. Вони відображають побудову не лише хмарних технологій, а й інформаційних технологій в цілому. Далі розглянемо кожен з них.

До послуг інфраструктури (Infrastructure as a Service – IaaS) можна віднести поєднання фізичних ресурсів, таких як сервери, мережеве обладнання, пропоновані замовникам як послуги, що надаються[2]. Послуги інфраструктури вирішують завдання належного оснащення центру обробки даних (ЦОД), надаючи обчислювальні потужності по мірі необхідності. Як правило, ці послуги підтримують інфраструктуру і набагато більше число споживачів в порівнянні з послугами застосунків. Схематичний рисунок взаємодії між хмарою за моделлю IaaS та девайсами відображено на рисунку 1.1.

Рис. 1.1 - Модель обслуговування Iaas

Послуги платформи (Platform as a Service – PaaS) – це модель обслуговування, у якій споживачі використовують вже готові застосунки, які були створені або придбані на цій платформі[2]. Як приклад можна привести інтеграцію сайтів з чат ботами. Існують декілька підрозділів цієї моделі, а саме:

Дані як послуга (Data as a Service – DaaS), ця модель надає користувачеві дисковий простір, який він може використовувати для зберігання інформації, наприклад Google Drive.

Безпека як послуга (Security as a Service – SaaS) надає можливість користувачам швидко розгортати продукти, які дозволяють забезпечити безпечне користування веб-технологіями, безпеку електронного листування, а також безпеку локальної системи, наприклад Датагруп пропонує пакети додаткових послуг по шифруванню передачі даних «VPN to Azure».

Модель PaaS доповняє IaaS, а саме PaaS це IaaS разом з операційною системою з доступним API (Application Programming Interface). Споживач в такому випадку не керує базовою інфраструктурою хмари, проте має контроль над розгорнутими застосунками і, інколи, деякими параметрами конфігурації

хостингового середовища. Схематичний рисунок взаємодії між хмарою за моделлю PaaS та девайсами відображено на рисунку 1.2.

Рисунок 1.2 - Модель обслуговування PaaS

Застосунки як послуга (Software as a Service – SaaS) надають доступ до застосунків як до сервісу, тобто застосунки провайдера запускаються в хмарі і надаються користувачам на вимогу як послуги [2]. Інакше кажучи, користувач отримує доступ до програмного забезпечення, розгорнутого на віддалених серверах, за допомогою мережі Інтернет, причому усі питання оновлення та ліцензій на дане програмне забезпечення регулюються постачальником даної послуги. Оплата в даному випадку здійснюється за фактичне використання програмного забезпечення. Із Saas звичайні користувачі взаємодіють найчастіше, це, наприклад, поштові сервіси, такі як GMail, Ukr.net, веб-версії мобільних додатків, таких як Telegram, Instagram та інші. Також багато серед цих служб безліч застосунків, спрямованих на корпоративних користувачів, це корпоративні окремі програми для чатів, звітів і т.д. Серед переваг цієї послуги можна додати невисоку ціну розгортання ПЗ і обслуговування, а недоліком є безпека цього програмного продукту.

Схематичний рисунок взаємодії між хмарою за моделлю SaaS та девайсами відображено на рисунку 1.3.

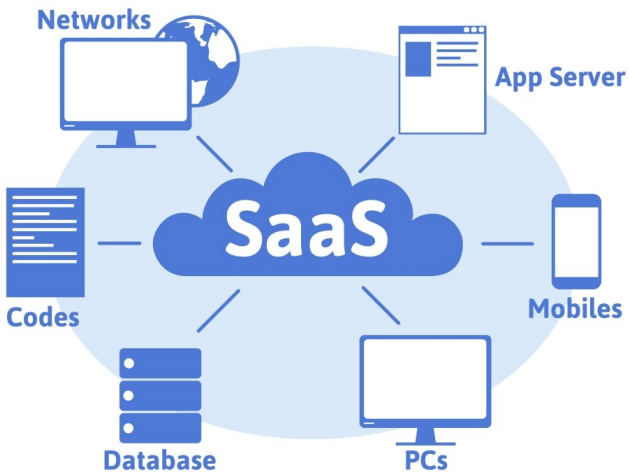

Рисунок 1.3 - Модель обслуговування SaaS

Моделі обслуговування за засобами доступу зазначені в таблиці 1.2.

Таблиця 1.2 - Моделі обслуговування за засобами доступу і управління

Моделі обслуговування	Засоби доступу і управління	Вміст
ПЗ як сервіс (SaaS)	Веб-браузер	Хмарні програми: офісні додатки, соціальні мережі, інтелектуальна обробка даних, системи управління вмістом.
Інфраструктура як сервіс (IaaS)	Система управління віртуальною інфраструктурою	Хмарна інфраструктура: сховища даних, обчислювальні сервера, організація мережевих з'єднань.
Платформа як сервіс (PaaS)	Хмарна середовище розробки	Хмарна платформа: бібліотеки, мови програмування, утиліти конфігурації сервісів, структуровані дані.

На рисунку 1.4 наведено загальний розподіл галузей для усіх вище наведених моделей обслуговування.

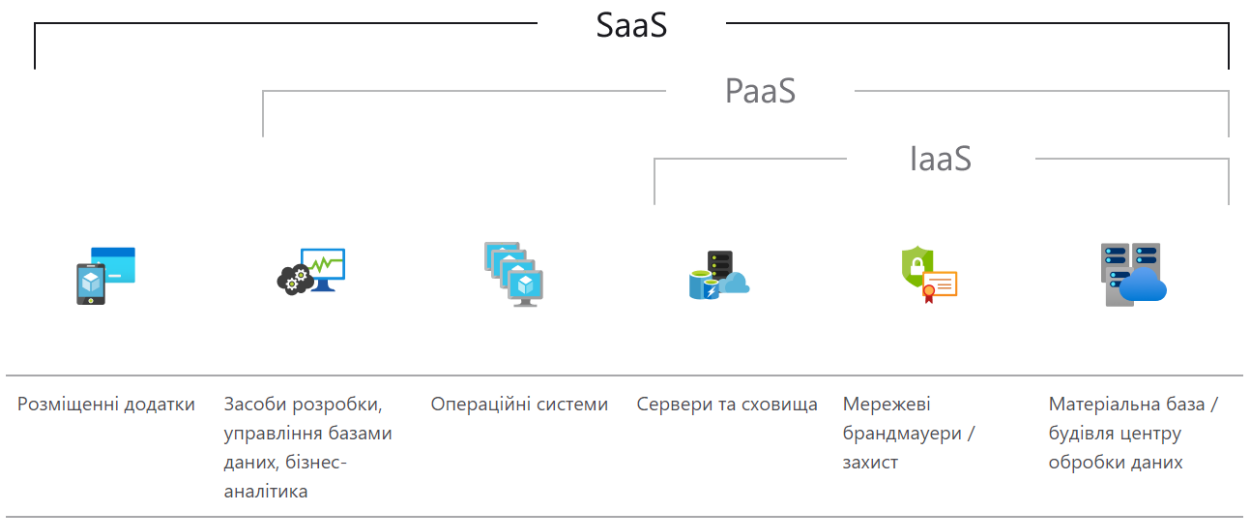

Рисунок 1.4 – Розподіл галузей між моделями обслуговування

1.5 Особливості використання ПЛІС в хмарних обчисленнях

Окремо, від інших послуг існує нова модель хмарних обчислень - ПЛІС як послуга, FaaS (FPGA as a Service). Застосування такої концепції не лише надасть ресурс програмованої логіки тим, хто його не має, але і дасть змогу амортизувати витрати від надання послуг тим, хто ними володіє. Тому реконфігуровані обчислення подешевшають внаслідок більш оптимального їх використання.

Спочатку роботу з виділення апаратного ресурсу можна буде здійснити на одному сервері. Після цього один HDL-запит можна буде виконати на ряді розподілених серверних локалізацій у всьому світі.

Базовою одиницею обладнання запропонованої системи буде звичайна мережа плат з інтерфейсом Ethernet, на яких знаходитимуться кристали ПЛІС. Також буде потрібна як мінімум одна серверна станція для кожної локалізації базового обладнання для виконання системних функцій. Відтак, кожна така структурна одиниця буде мати порти для зв’язку з локальною мережею обчислювального центру й порти для під’єднання до мережі Internet.

В роботі[3] досліджували динамічно реконфігуруємий процесор з використанням двох ПЛІС та хмарного модуля за допомогою технології Ethernet.

Клієнтській стороні буде достатньо встановити необхідне програмне забезпечення для користування таким хмарним сервісом, виконане у вигляді web-додатка і завантажуване з відповідного сервера.

В останні роки швидкість передачі даних через мережу Інтернет виросла в рази, і навіть мережі мобільного доступу до всесвітньої мережі вже сягають швидкодії до 20,7 МБ/с, і прогрес не стоїть на місці, швидкість буде рости і надалі. Але окремим стовпом стоїть затримка, зменшення якої є набагато складнішою задачею, оскільки навіть якщо в найближчому майбутньому і будуть розроблені маршрутизатори і супутнє обладнання з дуже малою затримкою, то вже саме поширення сигналу, швидкість якого хоча і значна, але скінченна, у масштабах планети все одно буде мати велику затримку. Відтак швидше за все в найближчому майбутньому цю проблему не вирішать, і на верхньому рівні обмін даними серед великих сегментів пристрою відбуватиметься з великою затримкою, оскільки реально використовуватимуться географічно віддалені сервери. Тож розробник HDL-опису має добре сегментувати своє рішення – створити його з якомога більшої кількості паралельних частин, які не потребуватимуть швидкого обміну інформацією між собою. І якщо навіть організувати сегментування буде досить складно для якихось задач, то як мінімум така система все одно буде мати щонайменше два сегменти, сегмент на сервері та сегмент (хай і незначний) на клієнтській стороні, який буде виконувати первинну обробку даних.

Імплементація HDL-коду відбувається в п’ять етапів (див. рисунок 1.5). Під час кожного з етапів спочатку відбуватиметься аналіз доступності ресурсів і після цього буде виконуватись декомпозиція HDL-коду, а на останньому етапі – його відображення у конфігураційному файлі відповідного кристала.

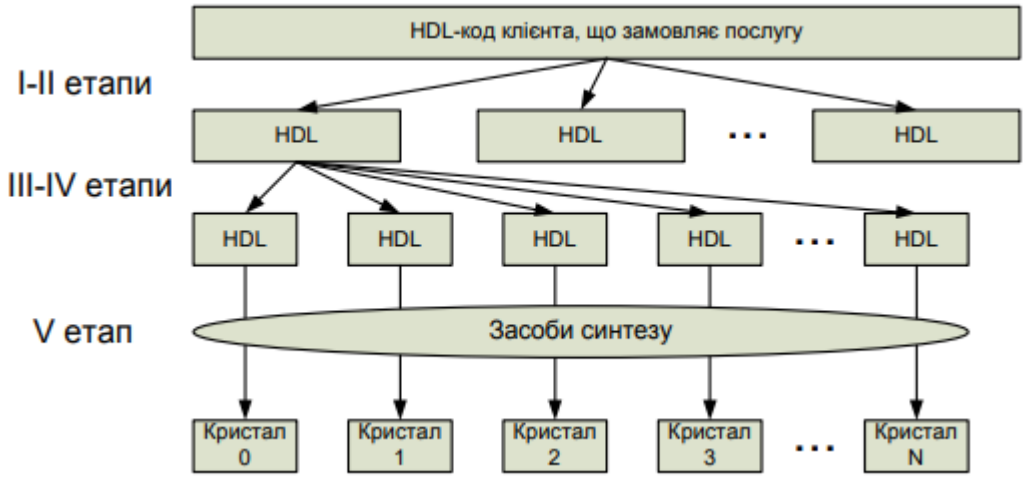

Рисунок 1.5 - Імплементація HDL-коду клієнта

Отже, на першому етапі має бути виконано аналіз доступності вільного обладнання з Інтернету із створенням відповідного графу, в якому кожен вузол – це обчислювальний центр, який має в наявності вільні ресурси програмованої логіки. Вага для кожного вузла буде вказувати на вільні еквівалентні логічні вентилі для конфігурації. Вітки графу, які зв'язують вершини, показуватимуть на затримку і швидкість, з якою інформація може поширюватися серед відповідних обчислювальних центрів.

На другому етапі буде виконуватися декомпозиція HDL-коду у відповідності до отриманого графу.

На третьому етапі передбачається виконання аналізу вільного сегмента програмованої логіки для обраних обчислювальних центрів. На базі такого аналізу для кожного обчислювального центру можна сформувати еквівалентний граф вільних ресурсів кристалів.

Четвертий етап – декомпозиція HDL-коду у відповідності з отриманим графом вільних кристалів.

На п'ятому етапі отримаємо кінцевий набір HDL-файлів для створення прошивок, які вкупі з додатковими файлами є кінцевим продуктом імплементації.

Що ж до процесу генерування самих прошивок, то для цільових кристалів він відбуватиметься у відповідності з технологією обраної фірми

виробника кристалів програмованої логіки. Для прикладу, для кристалів XILINX це може здійснюватися за допомогою відповідних консольних утиліт [4].

Можливість динамічної реконфігурації, яка підтримується у нових ПЛІС робить їх вигідним варіантом для використання у хмарних обчисленнях, бо коли потрібно виконувати багато різних розподілених операцій і час простою є дуже критичним - динамічна реконфігурація допомагає мінімізувати витрати у часі, а мова HDL (Hardware Description Language) в силу своєї абстрактності є універсальним інструментом для створення HDL-коду для різних синтезаторів.

2 ОГЛЯД ТА ВИБІР ПЛІС З МОЖЛИВІСТЮ ДИНАМІЧНОГО РЕКОНФІГУРУВАННЯ

2.1 Огляд ПЛІС з можливістю реконфігурування

Основна особливість ПЛІС – можливість багатократного перепрограмування. Ця ідея покладена в основу створення цифрових пристроїв і навіть потужних обчислювальних систем, в архітектурі яких є властивість реконфігурації в залежності від визначених умов з метою збільшення продуктивності обчислень. Такі системи відносяться до класу реконфігурованих обчислювальних систем (РОС), загальна концепція їх створення – забезпечення адаптивності архітектури відповідно до класів вирішуваних завдань.

До проблем РОС у першу чергу належать значні витрати часу на фізичне здійснення реконфігурації – витрати часу, непродуктивні витрати пам’яті й каналів зв’язку, енерговитрати та ін. Для подолання багатьох цих проблем використовують часткову реконфігурацію (ЧР), яка полягає у зміні лише частини пристрою в той час, коли інша частина продовжує працювати. Слід зазначити, що при цьому час реконфігурації пропорційний розмірам реконфігурованої області.

Найбільшими компаніями з розробки та випуску ПЛІС є Xilinx та Altera. ПЛІС компанії Altera є більш технічно прогресивні, але вони досить не підтримують часткову динамічну реконфігурацію на потрібному рівні.[5]. Натомість багато ПЛІС компанії Xilinx мають можливість часткової реконфігурації[6]. Але інструментальні засоби розробки, а саме повна підтримка часткової динамічної реконфігурації з’явилась лише в програмному пакеті Xilinx Design Suite, у версії 12 та більш пізніх[7]. В таблиці 2.1 наведено початок підтримки динамічної реконфігурації для різних ПЛІС.

Таблиця 2.1 - Підтримка часткової динамічної реконфігурації в ПЛІС компанії Xilinx

ТИП РЕКОНФІГУРАЦІЇ	РІК	СІМЕЙСТВО ПЛІС	ПІДТРИМКА САПР
Перше покоління ПЛІС підтримує часткову реконфігурацію (знято з виробництва)	1996	XC6200	–
Друге покоління ПЛІС – реконфігурація стовпцями	1999	Virtex, VirtexII, Virtex II Pro,	Xilinx ISE Design Suite 10. X (2008 р.) 11. X (2009 р.)
Сімейства невисокої ціни без офіційної підтримки часткової реконфігурації. Але, групами ентузіастів було успішно перевірено її можливість на окремих пристроях	2003	Spartan 3, Spartan 3E, Spartan 3A	Xilinx ISE Design Suite 10.X – інтегрований Plan Ahead
Третє покоління ПЛІС, підтримує часткову динамічну реконфігурацію фреймами, розміром 16-20 СББ, розпочинаючи з 2010 – розділами	2006	Virtex 4	Xilinx ISE Design Suite 10.1 – інтегрований Plan Ahead; 11. X (2009 р.) – інтегрований Plan Ahead; 12. X (2010 р.) – автономний Plan Ahead
	2006	Virtex 5	
Сімейство четвертого покоління, підтримує часткову динамічну реконфігурацію фреймами, розміром 40 СББ, розпочинаючи з 2010 – розділами	2009–2010	Virtex 6, Spartan 6	Xilinx ISE Design Suite 11.3 (2009 р.) – інтегрований Plan Ahead, підтримка порту ICAP; 12. X (2010 р.) – вбудована підтримка часткової реконфігурації (розділами); підтримка автононої версії Plan Ahead (модульна); 13. X (2011 р.)
Сучасні сімейства підтримують часткову динамічну реконфігурацію на базі розділів	2011–2013	Virtex 7, Artix 7, Kintex 7, Zynq 7000	Xilinx ISE Design Suite 13. X (2011 р.) 14. X (2013 р.)

Сьогодні ПЛІС стає одним з ключових елементів концепції побудови SoC. Нові покоління мікросхем (наприклад, Artix-7) здатні конкурувати з надвеликими інтегральними схемами (НВІС) як за продуктивністю, кількістю вентилів, надійності, так і по функціональності. Зараз вже існують ПЛІС, які можуть працювати без зовнішніх пристроїв для збереження конфігурацій і працюють з моменту подачі живлення. Таким чином, впровадження концепції SoC є одним з пріоритетних напрямів розвитку в електроніці (крім ПЛІС можна навести приклади використання SoC у телефонах та невеликих гаджетах), що і визначає архітектуру електронної апаратури в майбутніх поколінь (рис. 2.1.).

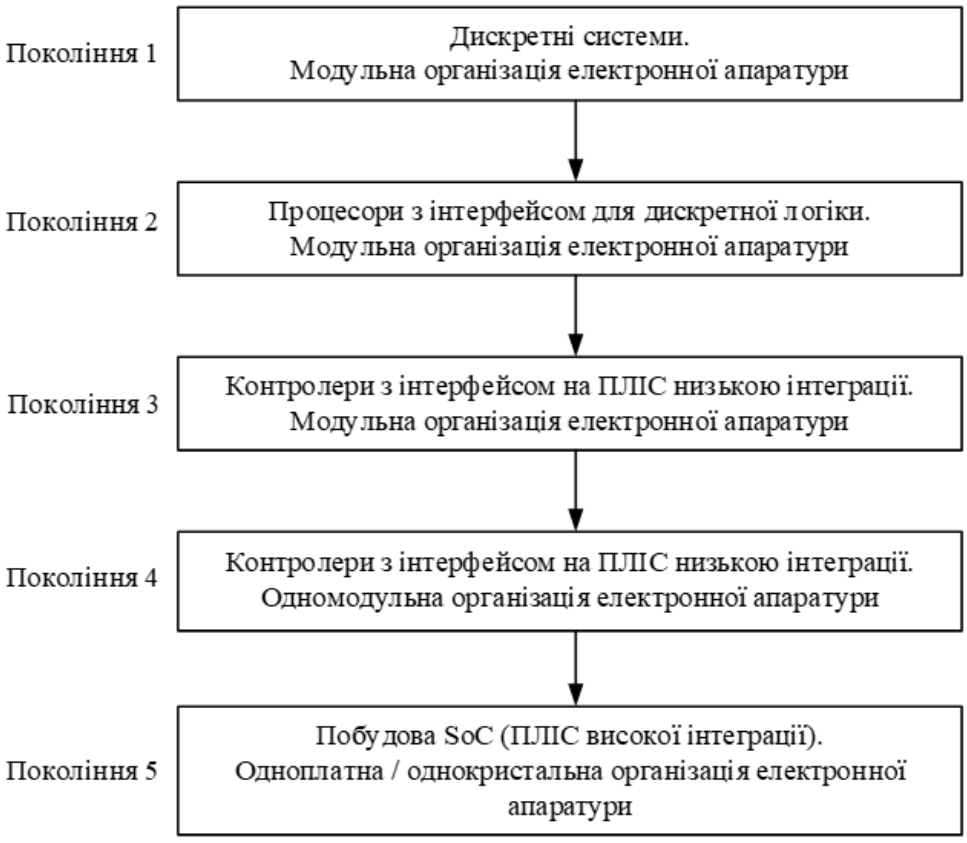

Рисунок 2.1 - Покоління електронної апаратури

2.2 Класифікація реконфігурованих обчислювальних систем

Далі розглянемо головні особливості методології часткової динамічної реконфігурації компанії Xilinx.

Головна проблема реалізації ЧДР – обмеження мікросхем і інструментальних засобів розробки. Методологія реалізації ЧДР, в першу чергу, потребує поділу ПЛІС на статичні й динамічні області. Найчастіше поділ роблять на основую область, де і будуть знаходитись базові і ключові компоненти, та динамічну область, де за необхідності будуть підгружатись і розташовуватись динамічні компоненти [8].

ЧДР за ступенем деталізації бувають локальними і модульними [9]. Локальна реконфігурація – це заміна одного конфігураційного логічного блоку. Натомість, модульна реконфігурація – заміна частини ПЛІС використовуючи, заздалегідь, окремо створені апаратні модулі, що активуються, або деактивуються під час реконфігурації. Частину ПЛІС, яка змінюється, називають реконфігурованим модулем.

За часом виконання - фактором класифікації є стан системи під час реконфігурації. Також окремо відрізняють реконфігурацію часу зупинки **(**РЧЗ**)** та реконфігурацію часу виконання (РЧВ) (рис. 2.2). У відповідності до назви, реконфігурація відбувається або за зупинки системи або ж під час її роботи. Для реалізації реконфігурації способом РЧЗ, потрібно зупинити систему і зовнішніми засобами реконфігурацію завантажити, цей спосіб авжеж, є досить легким і простішим з точки зору реалізації, а другий спосіб реалізується під час працюючої системи, без її вимикання.

Класифікація за типом ініціатора. Існує пасивна реконфігурація, коли реконфігурація системи відбувається по зовнішньому запиту з використанням зовнішнього алгоритму, і активної реконфігурації, за якої є певна підсистема управління, яка може ініціювати реконфігурацію незалежно від впливу зовнішніх факторів [8].

За способом виконання реконфігурації їх класифікують на повну і часткову. Повна реконфігурація (ПР) – це реконфігурація одного або кількох фізичних модулів системи у повному обсязі. А часткова реконфігурація (ЧР) є процесом реконфігурації частини фізичного модуля системи. Якщо повна реконфігурація відбувається при повній зупинці фізичного модуля на час

виконання реконфігурації, то при неповній модуль може виконувати певні функції: це функції підтримки, фонові функції, та зберігання даних у пам'яті, а часткова реконфігурація відбувається в режимі часу виконання[8, 10].

Класифікація РОС наведена на рис. 2.2.

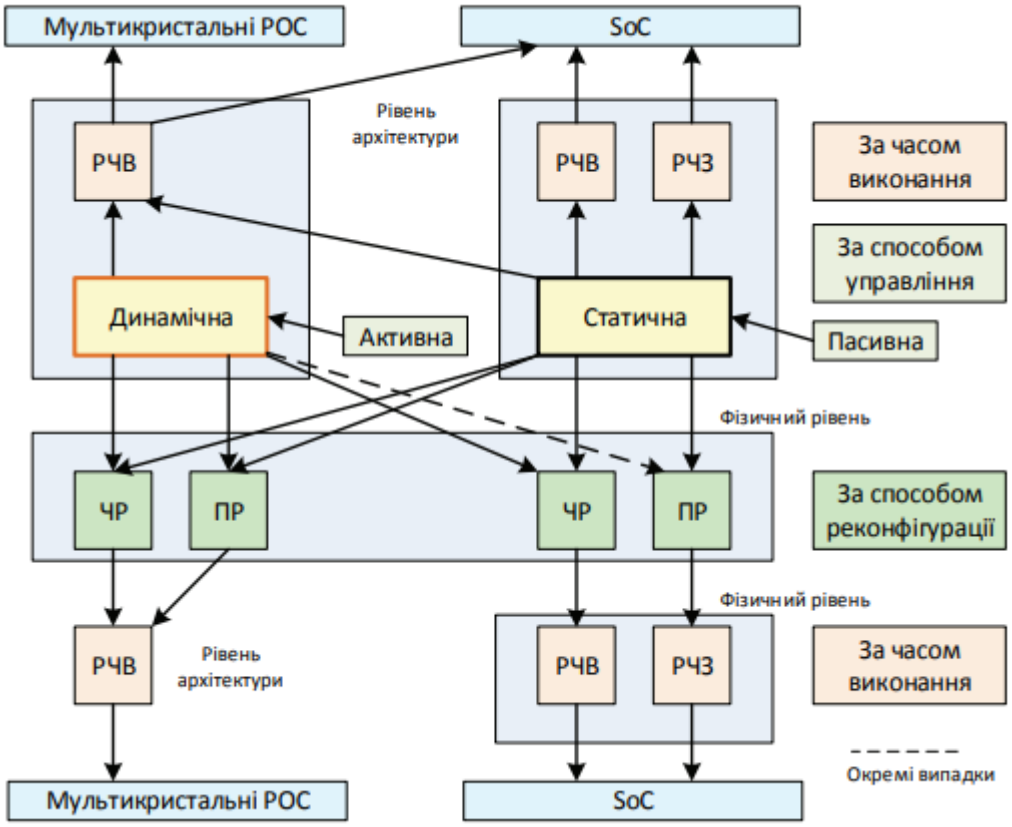

Рис. 2.2 - Класифікація реконфігурованих обчислювальних систем (ПЛІС)

2.3 Характеристика основних сімейств ПЛІС FPGA фірми Altera

ПЛІС від компанії Altera, яку нещодавно поглинула корпорація Intel, пропонує широкий перелік вбудованих блоків пам'яті під назвою Static Random Access Memory (SRAM), високошвидкісних блоків входу/виходу, трансиверів, логічних блоків і блоків маршрутизації.

Фірма Altera пропонує наступні сімейства ПЛІС типу FPGA:

1. Серія FPGA Cyclone. Вона створена в бік низького енергоспоживання і ефективної розробки на цих ПЛІС. У кожному поколінні FPGA Cyclone вирішує технічні проблеми, пов'язані з більшою інтеграцією, збільшеною продуктивністю, меншою потужністю споживання і швидким виходом на

ринок, дотримуючись при цьому основних вимог до витрат на виробництво цього типу ПЛІС.

Основними типами Cyclone з сімейства FPGA фірми Altera є: Intel Cyclone X, Cyclone V, Cyclone IV, Cyclone III [11].

2. FPGA фірми Intel типу Stratix і серія SoC поєднують в собі високу щільність й високу продуктивність ПЛІС завдяки притаманній гнучкості та ефективності, що надає змогу реалізувати більше функцій пристрою і максимально збільшити пропускну здатність системи.

Основними типами Stratix з сімейства FPGA фірми Intel є: Intel Stratix X, Stratix V, Stratix IV, Stratix III [12].

3. Сімейство FPGA фірми Intel Arria представляє оптимальний баланс між енергоефективністю та продуктивністю розроблених цифрових пристроїв. Сімейство цих пристроїв має багатофункціональний набір пам'яті, логічних блоків і блоків обробки цифрових сигналів (DSP) у поєднанні з високою надійністю сигналу до 25,8 Гбіт / с, що дає змогу інтегрувати більше функцій і максимально збільшити пропускну здатність системи. Крім того, використання SoC у сімействах пристроїв Intel Arria та Arria V передбачає жорсткі процесорні системи (HPS) на основі ARM для ще глибшої інтеграції та економії потужності споживання.

Головними типами Arria сімейства FPGA фірми Intel є: Intel Arria X, Arria V, Arria II та Arria GX [13].

2.4 Опис основних ПЛІС сімейства FPGA фірми Xilinx

ПЛІС компанії Xilinx дуже різнопланові і широко використовуються для вирішення задач різної складності. На сьогодні сімейство проектованих логічно інтегральних схем фірми Xilinx складається з чотирьох ПЛІС: Spartan, Virtex, Kintex та Artix.

Зазначені типи ПЛІС відповідають найкращим системним вимогам для спеціалізованих цифрових пристроїв, які розробляються починаючи від дешевих і простих додатків до дорогих та складних у проектуванні та

реалізації. Ці ПЛІС мають високошвидкісні смуги пропускання та велику кількість логічних елементів, а також можливості обробки сигналів для різних цифрових пристроїв, які розробляються. Сімейство FPGA включає такі серії, як:

1. Сімейство FPGA Spartan-7 є добре оптимізованим для своєї невисокої вартості, має низьку потужність споживання та високу продуктивність, пристрою вводу / виводу [14].
2. Сімейство FPGA Artix-7 добре оптимізовані й гнучкі при створенні пристроїв з низьким рівнем енергоспоживання, використовуються для створення пристроїв, що потребують послідовного використання зовнішніх периферійних пристроїв з високою пропускною спроможністю [15].
3. Сімейство FPGA Kintex-7: розроблено й оптимізовано для найкращої продуктивності цифрових пристроїв, які розробляються, з двократним покращенням аспектів роботи у порівнянні з попереднім поколінням, за невисоку вартість [16].
4. Сімейство ПЛІС Virtex-7 володіють найвищою продуктивністю, великим обсягом логічних ресурсів та в два рази перевершує попереднє за цими показниками покоління ПЛІС [17].

Сімейство ПЛІС компанії Xilinx 7-го покоління побудовано на основі сучасної, високопродуктивної, низькоенергозатратної технології high-k metal gate (техпроцесс 28-нм). Воно забезпечує значне збільшення продуктивності системи з пропускною здатністю елементів вводу / виводу до 2,9 Тб/с, близько 2 млн. логічних елементів, а також 5,3 TMAC/s DSP, споживаючи при цьому на 50% менше енергопотужності, ніж цифрові пристрої, що побудовані на основі попереднього покоління сімейства FPGA [18].

3 РОЗРОБКА СТРУКТУРИ ПРОЦЕСОРУ

3.1 Загальна схема роботи процесору

В даній дипломній роботі потрібно було реалізувати процесор на базі ПЛІС компанії Xilinx з можливістю динамічного реконфігурування. З типів реконфігурування було обрано часткове динамічне реконфігурування, бо воно є найбільш швидким і менш витратним з точки зору витрачених на це ресурсів.

Реалізацію процесору на базі ПЛІС Artix-7 буде зроблено для плати Nexys A7, це середня за потужністю плата, що як раз підійде під необхідність динамічного реконфігурування.

Загальна схема роботи процесору наведена на рисунку 3.1.

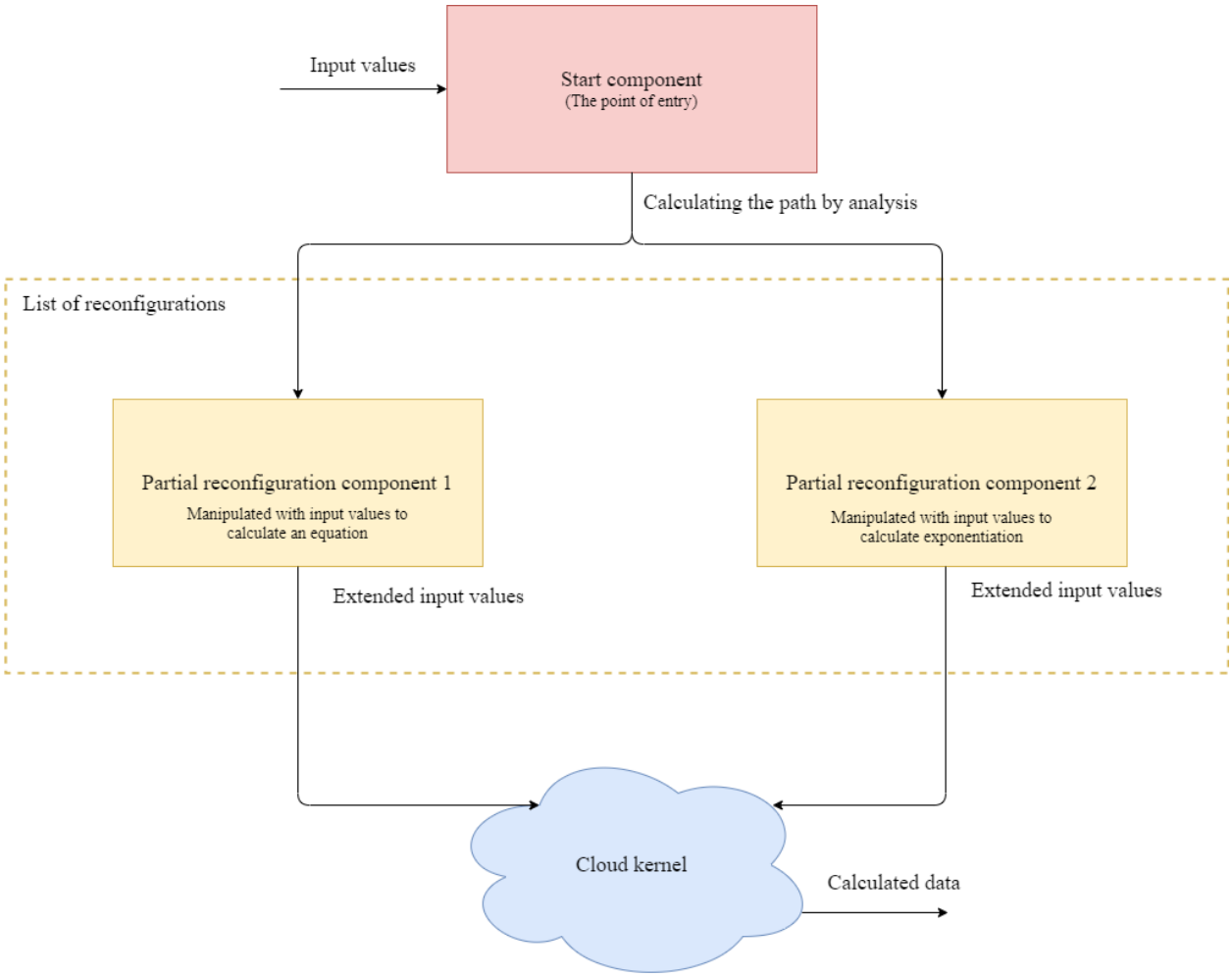

Рисунок 3.1 – Схема роботи розробленого процесору

Функціонує розроблений процесор наступним чином: У користувача для вибору операції є дві кнопки для двох типів операції зашитих у ПЛІС, три управляючі кнопки (сигнал скидання, завантаження даних та вибору результату), і перемикачі (шістнадцять штук) для вводу даних по шинам. Після вводу даних на перемикачах і натисканням на кнопку - усі сигнали поступають у «Start component», він виконує першу обробку даних і, в залежності, від обраної операції відправляє сигнали далі і ось у цьому місті буде відбуватись часткове динамічне реконфігурування, на рис. 3.1 можна побачити, що реконфігуємих компонентів два і їх активізація, тобто, завантаження с флеш пам'яті на фізичне місце на ПЛІС буде відбуватися по сигналу, який вибере користувач. Після завантаження обраний компонент виконає свою запрограмовану операцію і передасть дані в «хмарний» компонент (в даній роботі всі компоненти були реалізовані на одній платі, а в реальності хмарний компонент був би розміщений віддалено), який виконує обрану операцію і запише результат на вихідну шину на світлодіоди.

Опис основних компонентів системи представлено в Таблиці 3.1

Таблиця 3.1 - Опис елементів комплексу

№	Компонент процесору	Опис компоненту
1	StartComponent	Являє собою точкою взаємодії користувача з платою і виконує першу обробку вхідних даних
2	quadraticP	Динамічно завантажується та розраховує дискримінант за формулою приближеного розрахунку кореню квадратного
3	degreeP	Динамічно завантажується та обробляє переданий на вхід шину степені і видає її далі для розрахунків
4	cloudKernel	Виконує основі обчислення і видає результат на вихідну шину у «зовнішній світ»

Розрядність вхідної шини дорівнює 8, а вихідної 32. Розмірність шин були обрані по-перше з оглядом на фізичну кількість перемикачів та світлодіодів, в проекті вже буде використовуватись їх гранична кількість, а по-друге для операцій, які будуть виконуватись на ПЛІС, а це розв’язання квадратного рівняння і знаходження ступеня числа, тобто циклічне множення, цього діапазону буде достатньо для вхідних даних(числа до 127) і для вихідних - до 2 147 483 648, старший розряд вказує на знак.

Функціональну схему розробленого процесору зображено на рисунку 3.2

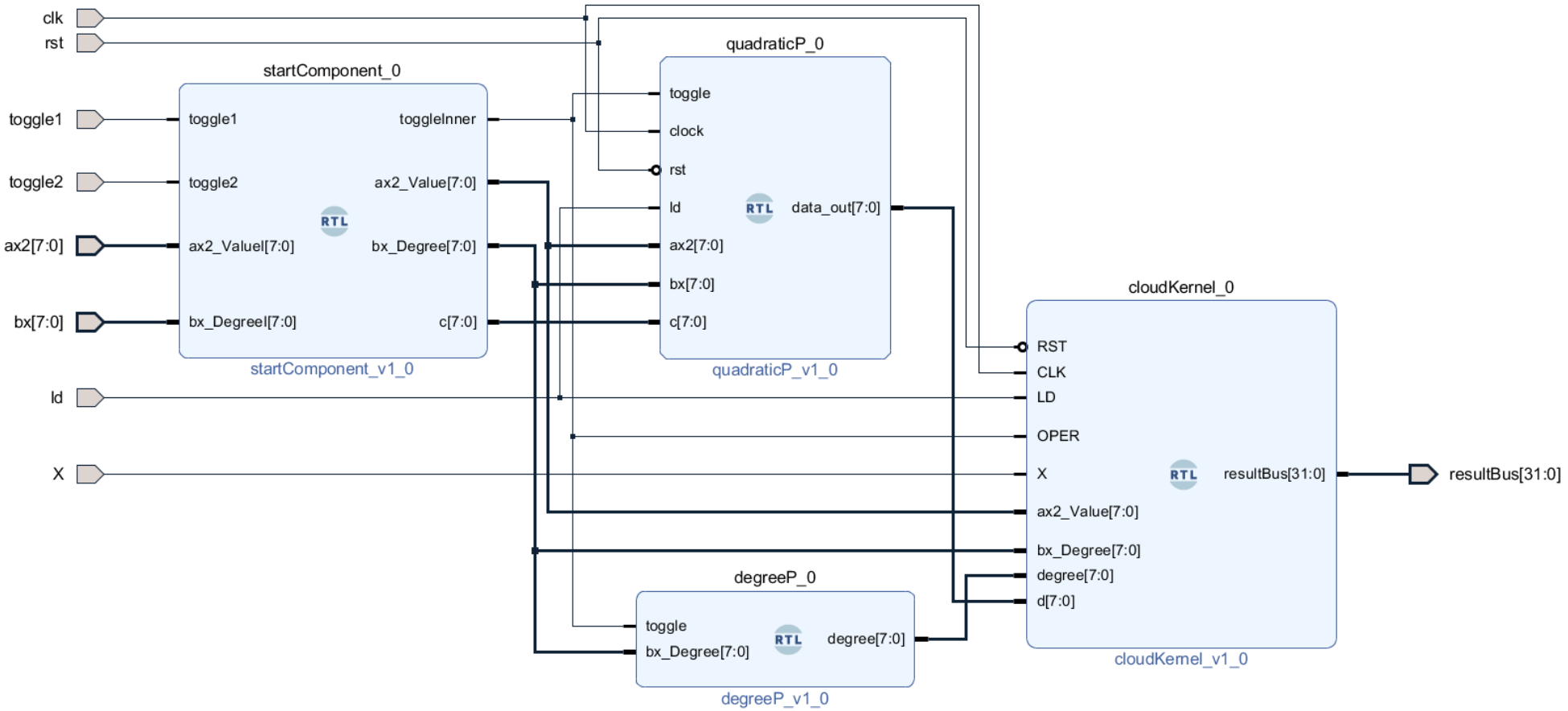

Рисунок 3.2 – Функціональна схема розробленого процесора

На функціональній схемі можна побачити вхідні сигнали, шини, та зв’язки між компонентами. Компоненти “quadraticP_0” та “degreeP_0” і є компонентами динамічної реконфігурації.

3.2 Алгоритми операцій реалізованих в ПЛІС

В розділі 3.1 вже вказувались операції, які зашиті у цю ПЛІС. Зі сторони може здатись, що знаходження степені числа або розв’язання квадратного рівняння є досить легкою операцією, але це лише так здається. На рисунку 3.3 наведено алгоритм знаходження ступеня заданого числа, а на рисунку 3.4 наведено алгоритм знаходження коренів квадратного рівняння, детальніше їх розберемо нижче.

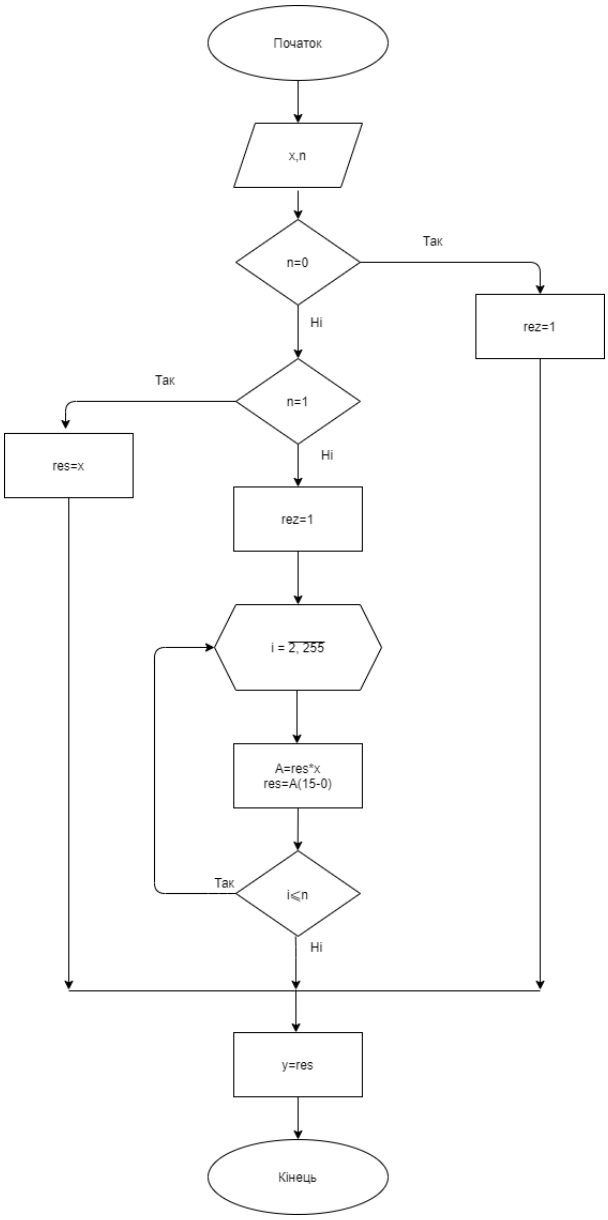

Рисунок 3.3 - Алгоритм знаходження ступеня числа

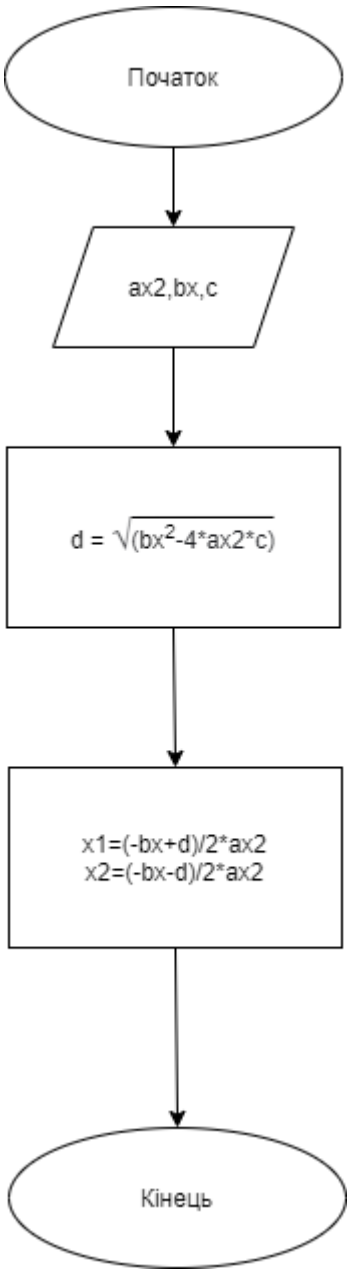

Рисунок 3.4 – Алгоритм знаходження коренів квадратного рівняння

На відміну від багатьох мов програмування, де є функції знаходження ступеня числа, наприклад pow() у java, в VHDL довелось його реалізовувати. В реалізації (див. рис. 3.3) закладено начальну перевірку на передану степінь і якщо воно дорівнює нулю то результат буде одиниця, і перевірка числа, що зводиться до ступеню, якщо воно дорівнює одиниці то і результат буде такий самий. В інших випадках відбувається циклічне (до 255 ітерацій) множення на передане число, умовою виходу є проходження усіх ітерацій згідно переданого ступеню.

Розв’язання квадратного рівняння на блок-схемі виглядає значно простішою операцією, але також, по аналогії з розрахунком ступені, проблема полягає у відсутності функції знаходження кореню квадратного в VHDL (функція насправді є, але детальну причину використання обраних методів описано в розділі 4.4).

Є багато методів знаходження кореня квадратного[19], в даній роботі використовувались два з них, один основний та другий для досліджень. Перший - метод CORDIC, для знаходження кореня квадратного для двійкових чисел[20] та другий - метод половинного ділення [21]. Метод CORDIC обчислює корінь квадратний в циклі за допомогою побітового зсуву на одиницю або нуль, а метод половинного ділення в циклі спочатку обирає середнє значення, виконує обернену операцію (зводить в квадрат) і перевіряє наскільки число наближено до номінального, далі обирає середнє значення з того інтервалу що ближче до істині.

Всюди де не написано, що використовувався метод половинного ділення, за замовчанням використовувався метод CORDIC.

4 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ПРОЦЕСОРУ

4.1 Огляд основних засобів проектування на ПЛІС

Xilinx створює сучасний продукт – ПЗ для конфігурування кристалів і розробки проектів. Перехід на покоління САПР стався 18 років тому. Раніше ISE використовувалось як альтернатива Foundation Series (попередня версія САПР). Перевагами ISE є скорочення часу на розробку, збільшення ефективності результатів завдяки удосконаленим методам проектування, алгоритмам синтезу, розміщенню та трасуванню проекту на кристалі.

САПР Xilinx наразі існують у таких конфігураціях:

1) програмний доданок Foundation ISE. Ця САПР є найбільш повноцінною ISE серед усіх САПР фірми Xilinx, адже вона підтримує функціонування із усіма сімействами й типами ПЛІС;

2) для спряження з САПР других виробників є конфігурація Alliance ISE що підтримує усі кристали Xilinx;

3) програмний доданок ISE Design Suite. Дана САПР знаходиться у вільному доступі на офіційному сайті Xilinx. Обмеженням САПР WebPack ISE є те, що за допомогою цієї САПР можна розробляти проекти на основі кристалів сімейства CPLD і FPGA, але логічна ємність не може сягати понад 300 тис. системних вентилів;

4) нова САПР Xilinx – Vivado High Level Synthesis (надалі VHLS). VHLS створена для будування цифрових пристроїв на основі мов програмування високого рівня. Головна її мета – спрощення процесу проектування для тих розробників, які володіють програмуванням. Проте на практиці ПЛІС може мати складності для чисто профільних програмістів, коли вони зустрічають велику кількість незвичних завдань, якщо ті не є еквівалентами оператору мови програмування (для прикладу, оператори апаратури).

Усі згадані вище САПР відрізняються такими показниками як сімейства та типи ПЛІС, що здатна підтримувати та або інша САПР; додаткові або унікальні інструменти при проектуванні. Однак усі описані вище САПР мають

однотипну структуру й однотиповий інтерфейс використання (за виключенням САПР Alliance ISE).

В даній роботі буде використовуватись саме САПР Vivado Design Suite.

САПР Vivado Design Suite призначено для підвищення продуктивності розроблювального пристрою. Вона містить набір інструментів, призначених для збільшення загальної продуктивності для проектування, інтеграції та впровадження систем з використанням пристроїв фірми Xilinx.

4.2 Етапи проектування цифрових пристроїв на ПЛІС

Реалізація проекту на ПЛІС від фірми Xilinx відбувається по наступним етапам:

1 - Обрати сімейство та тип ПЛІС

2 - Створити проекту в Vivado Design Suite

3 - Обговорити опису пристрою, це може бути схема або опис алгоритму.

4 - Зробити синтез пристрою

5 - Виконати функціональне моделювання

6 - Визначити місце розміщення і трасування розробленого проекту на ПЛІС

7 - Провести часове моделювання

8 - Виконати програмування ПЛІС

9 - Завантажити проект на кристал через створений bitstream file

При виборі конкретної ПЛІС необхідно враховувати декілька факторів, а саме:

- граничну вартість розробки
- необхідна швидкодія
- складність проекту

Іноді деякі обрані параметри потрібно нагально змінити під час проектування пристрою, в САПР Vivado Design Suite є такий інструментарій, розробник може змінити тип ПЛІС або навіть сімейство.

Розроблюваний пристрій може бути представлений у декількох видах: опис на мові VHDL, принципова схема, діаграма станів або пакети та бібліотеки розробника.

В процесі синтезу, враховуючи вхідні дані проекту, формується спеціальний список з'єднань де містяться набори примітивів і компонентів, які потім реалізуються на базі ресурсів обраного сімейства та конкретного кристалу. Результат синтезу далі використовується як вхідні дані.

Процес функціонального моделювання відбувається, невраховуючі реальні значення затримок при проходження сигналів, що використовується для перевірки відповідності роботи ПЛІС від отримання вхідних даних до вихідних сигналів під час роботи алгоритму.

Фінальний етап розробки пристрою - це завантаження конфігурації на кристал. Зазвичай це робиться через фізичний застарілий порт JTAG або сучасний USB чи Ethernet.

Під час кожного з етапів опису потрібно ретельно дотримуватися всіх рекомендацій, в першу чергу для зберігання працездатності ПЛІС і авжеж для скороченню часу розробки пристрою, ці рекомендації можна знайти на веб-сайтах компаній розробників ПЛІС, наприклад Xilinx.

4.3 Етапи створення проекту в Vivado Design Suite 2018.2

Проект було створено у САПР Vivado Design Suite 2018.2 фірми Xilinx.

Для того, щоб створити новий проект у САПР Vivado Design Suite 2018.2 розробнику необхідно відкрити відповідну програму, створити новий проект обравши пункт меню «Project» - «New» у «Files». Після цього розробник вказує назву проекту та його короткий опис проекту.

Далі потрібно вказати такі основні налаштування, як (див. рисунок 4.1):

1 - необхідне сімейство ПЛІС

2 - тип ПЛІС

3 - засоби для синтезу пристрою

4 - обрати тип опису пристрою

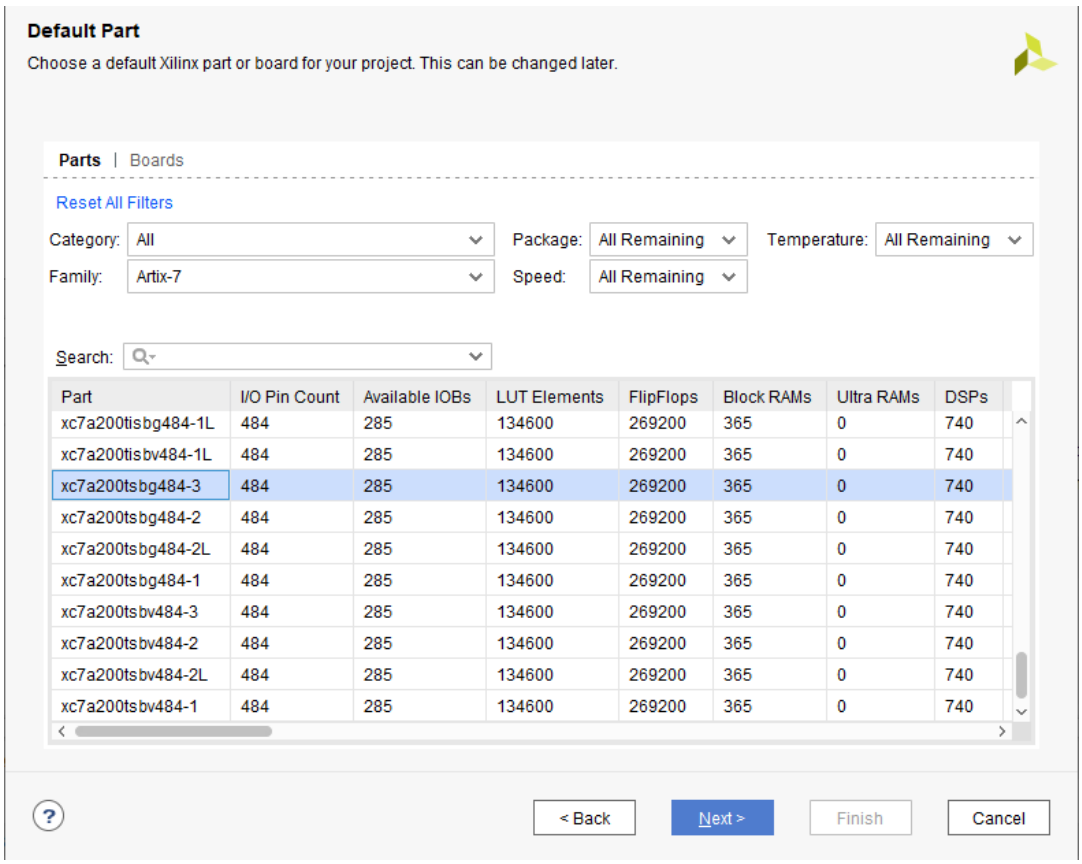

Рисунок 4.1 – Параметри ПЛІС нового проекту на САПР Vivado Design Suite

Після налаштування відкриється стандартне вікно з повним інструментарієм (див. рисунок 4.2).

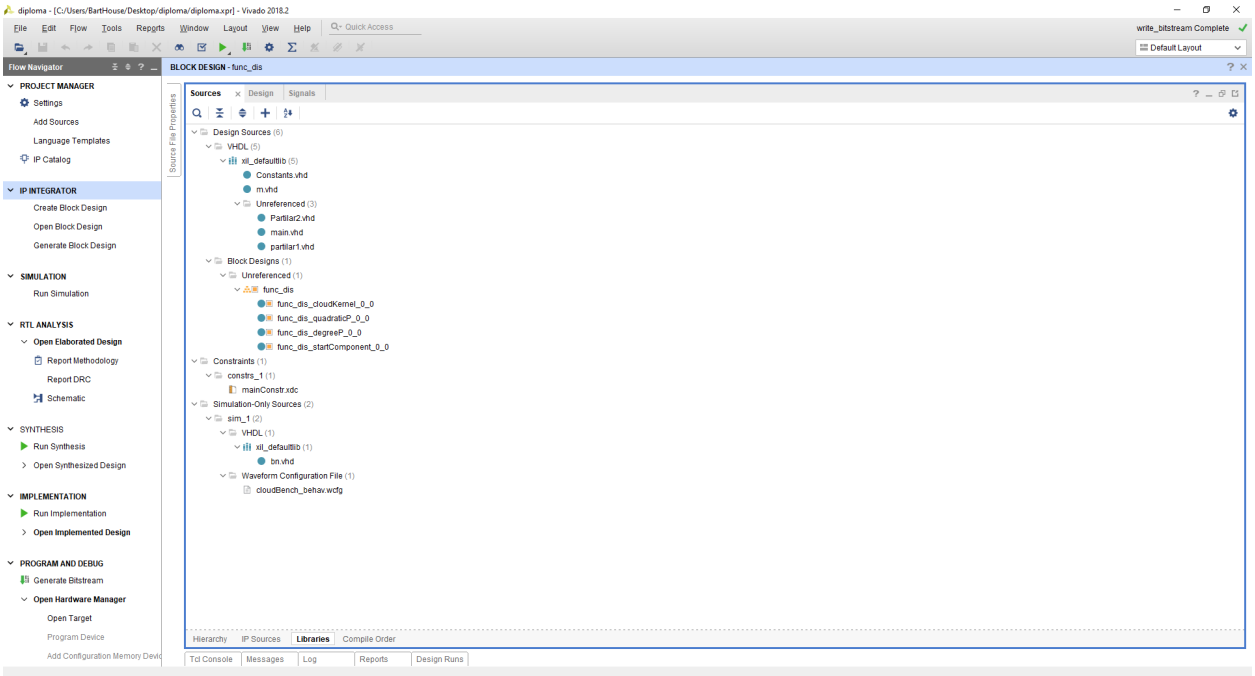

Рисунок 4.2 – Стандартне вікно САПР Vivado Design Suite 2018.2

Далі проводиться розробка пристрою з використанням мови VHDL. Весь код наведено в додатку А.

Фрагменти коду (компонентів) приведені нижче (Лістинг 4.1).

Лістинг 4.1 – Фрагмент коду компонента розрахунку кореня квадратного за допомогою метода CORDIC

```
signal part_done  : std_logic := '0';
signal count      : integer := 3;
signal dInt       : integer := 0;
signal ax2T,bxT,cT: integer range 0 to LENGTH_INTEGER_PARAM;
signal data_in    :  std_logic_vector(LENGTH_INPUT_PARAM - 1 downto 0) := "00000000";
signal result     : std_logic_vector(4 downto 0) := "00000";
signal partialq   : std_logic_vector(5 downto 0) := "000000";
……
dInt <= ((bxT * bxT) - 4 * ax2T * cT);
data_in <= conv_std_logic_vector(dInt, LENGTH_INPUT_PARAM);
For i in 2 to 10
    Loop
     if (part_done='0') then
        if (count>=0) then
           partialq(1 downto 0)  <= data_in((count*2)+ 1 downto count*2);
           part_done <= '1';
       else
        data_out <= conv_std_logic_vector(conv_integer(result(3 downto 0)),
LENGTH_INPUT_PARAM);
      end if;
      count <= count - 1;
 elsif(part_done='1')then
     if((result(3 downto 0) & "01") <= partialq)then
        result   <= result(3 downto 0) & '1';
        partialq(5 downto 2) <= partialq(3 downto 0) - (result(1 downto 0)&"01");
      else
        result   <= result(3 downto 0) & '0';
        partialq(5 downto 2) <= partialq(3 downto 0);
     end if;
        part_done  <= '0';
     end if;
  end Loop;
```

Якщо проект робиться під конкретну плату то необхідно зв’язати вхідні та вихідні порти плати та ПЛІС. Для цього потрібно спочатку ознайомитися з документацією вибраної плати (наприклад Nexys A7) на офіційному сайті компанії Xilinx. Для того, щоб вручну налаштувати входи та виходи, потрібно створити Xilinx Constrain File (.xcf), фрагмент конфігурації приведено нижче(Лістинг 4.2).

Лістинг 4.2 – Фрагмент конфігурації .xcf файлу

```
## Clock signal
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];

## Buttons
set_property -dict { PACKAGE_PIN C12 IOSTANDARD LVCMOS33 } [get_ports { rst }];
set_property -dict { PACKAGE_PIN E16 IOSTANDARD LVCMOS33 } [get_ports { toggle1 }];
set_property -dict { PACKAGE_PIN F15 IOSTANDARD LVCMOS33 } [get_ports { toggle2 }];
set_property -dict { PACKAGE_PIN R10 IOSTANDARD LVCMOS33 } [get_ports { ld }];
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { x }];

## Switches
set_property -dict { PACKAGE_PIN U9 IOSTANDARD LVCMOS33 } [get_ports { ax2[0] }];
set_property -dict { PACKAGE_PIN U8 IOSTANDARD LVCMOS33 } [get_ports { ax2[1] }];
set_property -dict { PACKAGE_PIN R7 IOSTANDARD LVCMOS33 } [get_ports { ax2[2] }];
set_property -dict { PACKAGE_PIN R6 IOSTANDARD LVCMOS33 } [get_ports { ax2[3] }];
set_property -dict { PACKAGE_PIN R5 IOSTANDARD LVCMOS33 } [get_ports { ax2[4] }];
set_property -dict { PACKAGE_PIN V7 IOSTANDARD LVCMOS33 } [get_ports { ax2[5] }];
set_property -dict { PACKAGE_PIN V6 IOSTANDARD LVCMOS33 } [get_ports { ax2[6] }];
set_property -dict { PACKAGE_PIN V5 IOSTANDARD LVCMOS33 } [get_ports { ax2[7] }];
```

Далі створюється bitstream файл і виконується завантаження проекту на ПЛІС.

4.4 Дослідження створеного процесору

В розділі 3.2 вже вказувалось, що для знаходження кореню квадратного використовувались методи приблизного знаходження, бо не існує стандартної функції в VHDL його знаходження. Функція насправді є і вона йде в бібліотеці IEEE.MATH_REAL і називається sqrt(r : real). Проблема її використання полягає в наступному: для тестування, дослідження через створення simulation/test-bench ця функція добре підходить, але вона не дасть зробити синтез проекту, бо дробові числа (типи даних real та float) не можливо представити для ПЛІС у двійковому форматі. Саме через цю проблему довелось використовувати наближені методи, щоб зробити синтез, получити дані з кількості використовуваних компонентів для створеного процесору (див. табл. 4.1, 4.2) та сгенерувати bitstream файл. Далі наведено квадратні рівняння, які були протестовані в процесорі.

1. $x^2 + 5 * x + 12 = 0;\ \ d = \sqrt{b^2 - 4 * a * c} = 4;\ x_1 = -2, x_2 = -6$

На рисунку 4.3 відображений результат розв’язання першого рівняння, по сигналу X обирається який корінь виводити у вихідну шину, а в dRound корінь дискримінанту.

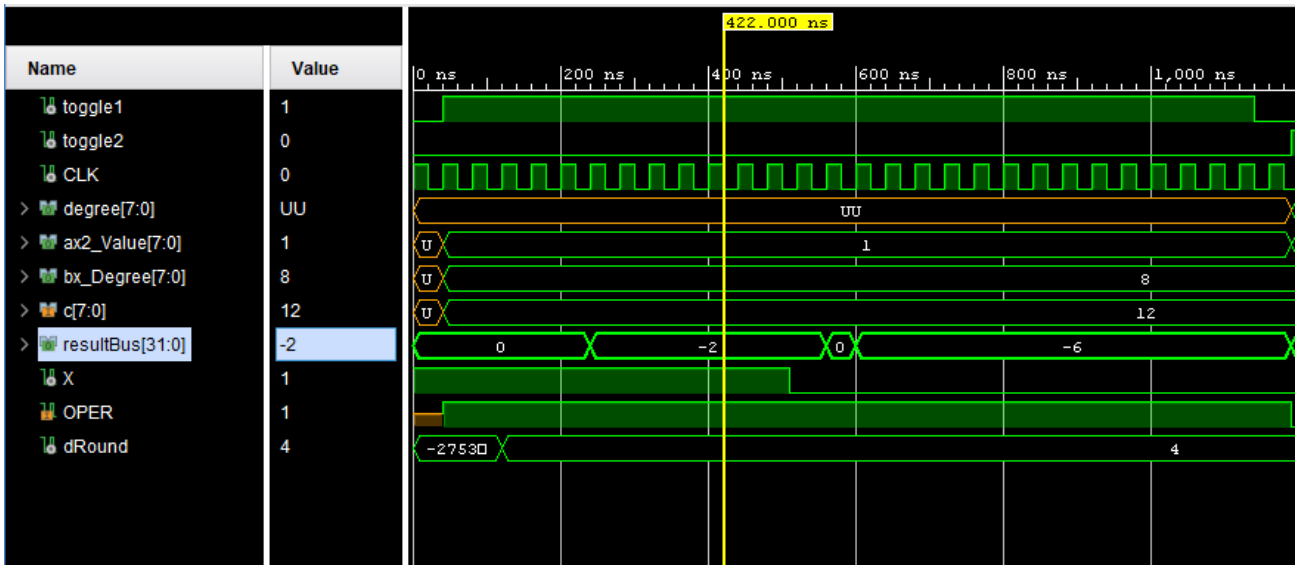

Рисунок 4.3 – Результат розв’язання першого квадратного рівняння

2. $x^2 + 9 * x + 5 = 0;\ d = \sqrt{b^2 - 4 * a * c} = 7;\ x_1 = -1, x_2 = -8$

На рисунку 4.4 відображений результат розв'язання другого рівняння, дискримінант з $\sqrt{61}$ дорівнює 7; тобто впроваджений алгоритм приводить до найменшого цілого числа.

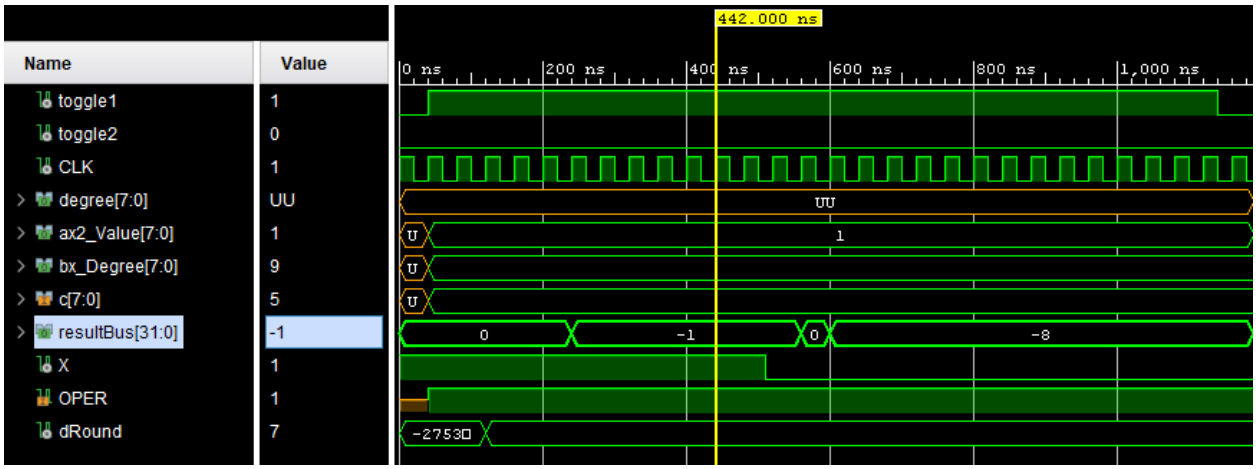

Рисунок 4.4 – Результат розв'язання другого квадратного рівняння

3. $x^2 + 9 * x + 7 = 0;\ d = \sqrt{b^2 - 4 * a * c} = 7;\ x_1 = -1, x_2 = -8$

На рисунку 4.5 відображений результат розв'язання третього рівняння.

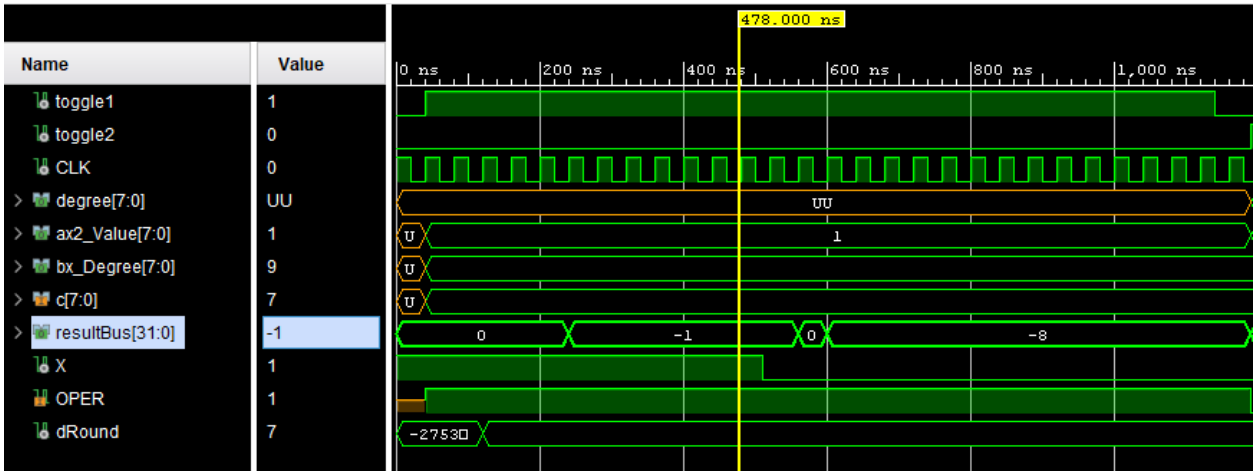

Рисунок 4.5 – Результат розв'язання третього квадратного рівняння

При використанні методу половинного ділення результати були однакові.

Далі наведені числа та ступені в які потрібно було їх звести.

1. $3^5 = 243$

На рисунку 4.6 відображений результат зведення в степінь першої пари число - ступень. З нього видно, що при перемиканні з кнопки toggle1 на кнопку toggle2 проходить загрузка ступеню числа та самого числа, скидається шина результату та розраховується зведення в степінь.

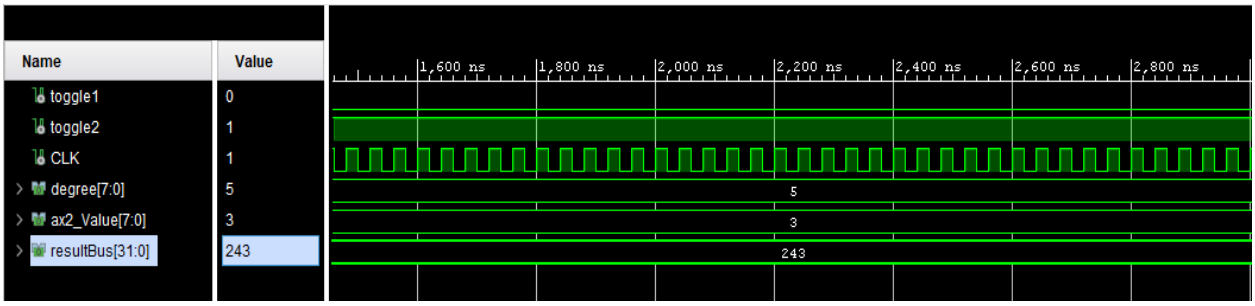

Рисунок 4.6 – Результат розрахунку першої пари число – степінь

2. $6^6 = 46656$

На рисунку 4.7 відображений результат зведення в степінь другої пари число - ступень. Як і очікувалось, таке велике число розрахувалось успішно.

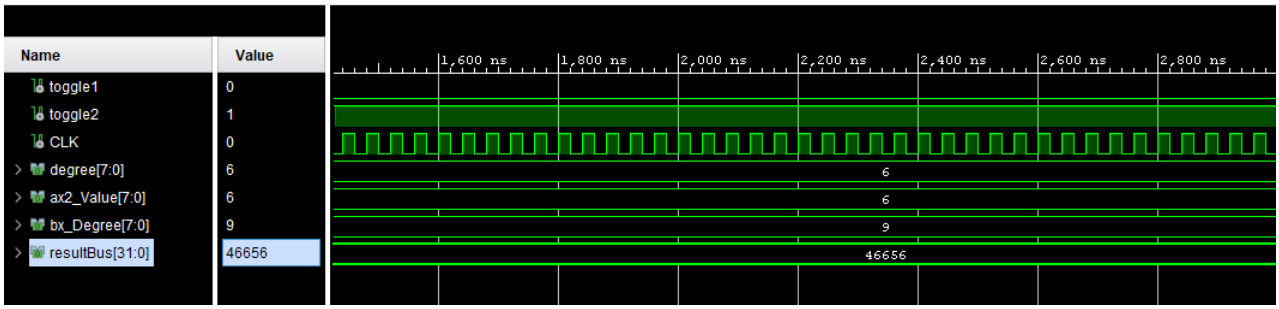

Рисунок 4.7 – Результат розрахунку другої пари число – степінь

3. $13^8 = 815\ 730\ 721$

На рисунку 4.8 відображений результат зведення в степінь третьої пари число - ступень. Також, як і очікувалось, число до 2^{31}розрахувалось успішно.

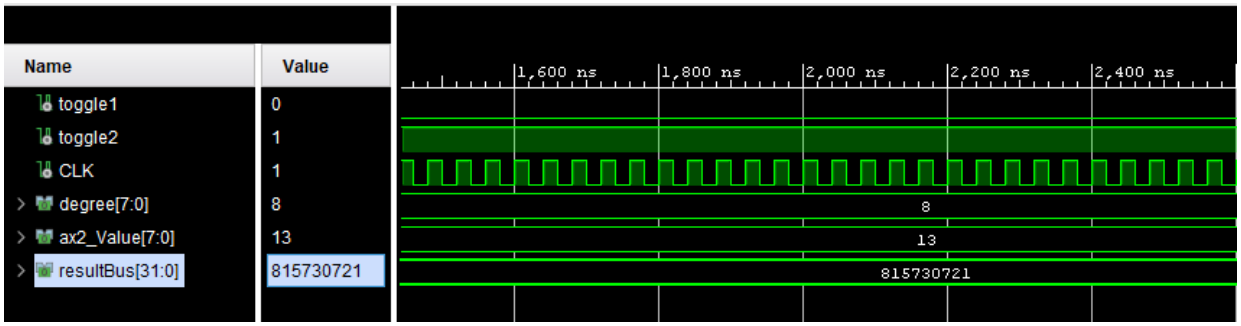

Рисунок 4.6 – Результат розрахунку третьої пари число – степінь

4. $8^{13} = 549\,755\,813\,888$

На рисунку 4.8 відображений результат зведення в степінь четвертої пари число - ступень. Результуюче число вийшло більше ніж розмірність вихідної шини 2^{31}, тому в результаті нуль.

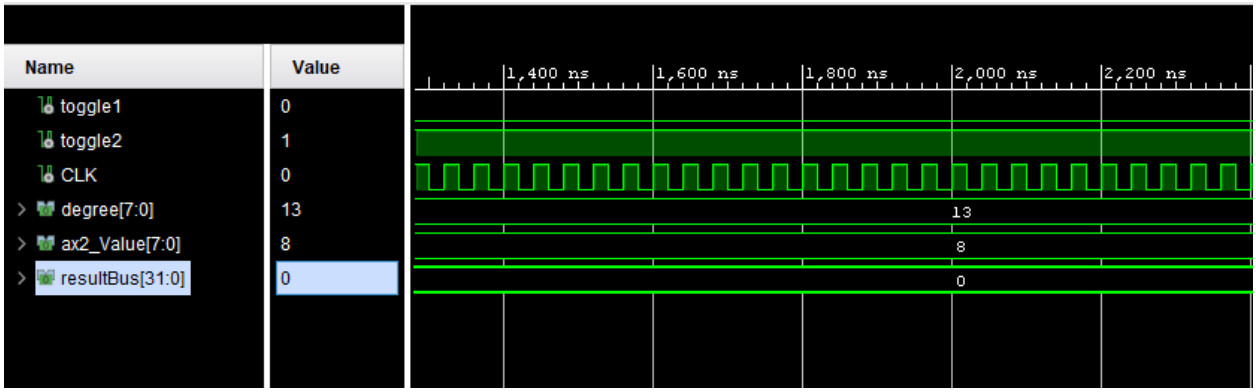

Рисунок 4.6 – Результат розрахунку четвертої пари число – степінь

4.5 Порівняння затрат елементної бази для двох різних алгоритмів знаходження квадратного кореню

В таблиці 4.1 наведено кількість використовуваних елементів після синтезу.

Таблиця 4.1 - Кількість використовуваних елементів процесору

Site Type	Used	Available	Utils, %
LUT as Logic	5974	134600	4.44
Register as Flip Flop	61	269200	0.02
Register as Latch	32	269200	0.01
DSP48E1	254	740	34.32
Bonded IOB	69	285	24.21
BUFGCTRL	1	32	3.13

В таблиці 4.2 наведено кількість використовуваних примітивів

Таблиця 4.2 - Кількість використовуваних примітивів

Ref Name	Used	Functional Category
LUT5	3326	LUT
LUT6	1760	LUT
LUT2	907	LUT
DSP48E1	254	Block Arithmetic
LUT4	160	LUT
LUT3	160	LUT
LUT1	62	LUT
CARRY4	45	CarryLogic
IBUF	37	IO
OBUF	32	IO
LDCE	32	Flop & Latch
FDRE	26	Flop & Latch
FDCE	19	Flop & Latch
FDPE	16	Flop & Latch
BUFG	1	Clock

В таблиці 4.3 наведено кількість використовуваних елементів після синтезу при використанні методу половинного ділення.

Таблиця 4.3 - Кількість використовуваних елементів спроектованого процесору методом половинного ділення

Site Type	Used	Available	Utils, %
LUT as Logic	8608	134600	6.40
Register as Flip Flop	77	269200	0.03
Register as Latch	48	269200	0.02
DSP48E1	504	740	68.11
Bonded IOB	69	285	24.21
BUFGCTRL	1	32	3.13

В таблиці 4.4 наведено кількість використовуваних примітивів

Таблиця 4.4 - Кількість використовуваних примітивів спроектованого процесору методом половинного ділення

Ref Name	Used	Functional Category
LUT3	4130	LUT
LUT5	4006	LUT
DSP48E1	504	Block Arithmetic
LUT6	331	LUT
LUT4	76	LUT
LUT2	70	LUT
LUT1	62	LUT
LDCE	48	Flop & Latch
CARRY4	45	CarryLogic
IBUF	37	IO
OBUF	32	IO
FDPE	32	Flop & Latch
FDRE	26	Flop & Latch
FDCE	19	Flop & Latch
BUFG	1	Clock

На рисунку 4.7 наведено порівняння використаних ресурсів для обох методів, згідно ним, використання методу CORDIC для знаходження квадратного кореню є більш економним, ніж методом половинного ділення.

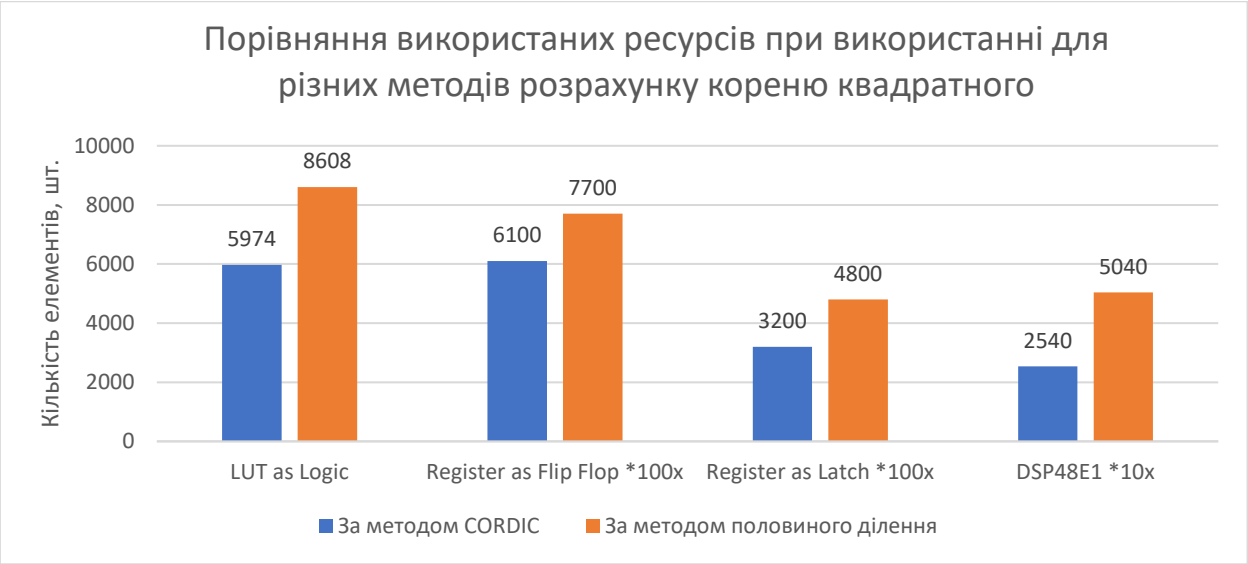

Рисунок 4.7 – Порівняння кількості використаних ресурсів для різних методів розрахунку кореню квадратного

Таким чином, імплементація методу CORDIC в ПЛІС витрачає у 1.4 рази менше ресурсів по елементам LUT as Logic, Register as Flip Flop, у 1.5 рази для Register as Latch, та аж у 2 рази для DSP. Така різниця пов’язана з методом розрахунку самого методу, метод половинного ділення використовує циклічне зведення у квадрат, а CORDIC - циклічний побітовий зсув.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вимоги безпеки при виконанні робіт на робочому місці

Згідно з документом: ”Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями від 14.02.2018 №207”[22], який був розроблен на основі Директиви 90/270/ЄЕС від 29 травня 1990 року про мінімальні вимоги безпеки та здоров’я при роботі з екранними пристроями, екранними пристроями вважаються: електронні засоби для відтворення будь-якої графічної або алфавітно-цифрової інформації (рідкокристалічні, проекційні, плазмові, органічні світлодіодні монітори та інші новітні розробки у сфері інформаційних технологій), а робочим місцем вважається: сукупність устаткування, що включає екранний пристрій, який може доповнюватися клавіатурою або пристроєм введення та/або програмним забезпеченням та іншими приладами (периферійні пристрої, що включають пристрої для електронних носіїв, друкувальний пристрій, тримач документів, телефон, модем, робоче крісло, робочий стіл або робоча поверхня „розумного” столу, а також інше необхідне виробниче середовище).

Роботодавець повинен забезпечити умови праці працівнику, які вказані в даному наказі. Нижче я приведу декілька з них.

- Роботодавець повинен проінформувати працівників під розписку про умови праці.
- Роботодавець під час облаштування робочого місця повинен обрати таке устаткування, яка не буде створювати зайвого шуму, рівень шуму повинен відповідати постанови: “Санітарні норми виробничого шуму, ультразвуку та інфразвуку від 1 грудня 1999р. ДСН 3.3.6.037-99” [23]
- Роботодавець повинен забезпечити за свій рахунок проведення медичних оглядів своїх працівників згідно з постановою: “Про затвердження Порядку проведення медичних оглядів працівників

певних категорій від 23 липня 2007р. зі змінами від 14 лютого 2012р. № 846/14113” [24]

Далі наведені вимоги безпеки до робочих місць працівників

- Для забезпечення безпеки та захисту уже випромінювання від екранних пристроїв повинне бути в рамках допустимих значень.
- Освітлення робочого місця має створювати контраст між екраном та навколишнім середовищем.
- Робочі місця повинні бути спроектовані таким чином, щоб працівники мали простір для зміни робочого положення.
- Мікроклімат на робочому місці повинен відповідати вимогам постанови: “Санітарні норми мікроклімату виробничих приміщень від 1 грудня 1999 року ДСН 3.3.6.042-99”[25]
- Робоче крісло має бути стійким і дозволяти легко рухатися.

До самих екранних пристроїв діють наступні вимоги:

- Екранні пристрої повинні бути безпечні для працівника
- Усе випромінювання повинне бути на безпековому рівні
- Зображення на екранному пристрої має бути стабільним
- Яскравість та/або контрастність має легко регулюватися працівником.
- Екран не має відбивати світло або відблискувати, щоб не викликати дискомфорту.

5.2 Шкідливі виробничі фактори на робочому місці

При використанні ЕОМ виникають нові психологічні і психофізіологічні проблеми, суть яких потрібно враховувати при проектуванні трудового процесу. Іншою особливістю є значне інформаційне навантаження. Також досить значне навантаження на зорову та центральну нервову системи. Окрім цього навантаження є також вплив від самої машини (ЕОМ) це шум, статичне навантаження, електромагнітні хвилі які можуть бути у різних частотних

діапазонах мікрокліматичні фактори, електростатичне поле, що утворюється на екрані монітора та багато інших.

В галузях, де працівник переважну більшість робочого часу проводить з використанням комп'ютерів зазвичай виникає цілий ряд ергономічних проблем, при вирішенні яких може значно знизити навантаження. До ергономічних проблем можна віднести: висоту робочого стола, висоту робочого крісла, дистанція до монітору, рівень шуму у приміщенні, відстань між столами та інші. [26] Ергономічні проблеми не охоплюють питання яскравості дісплею, формування раціонально побудованих символів на екрані і багато інших проблем, зміна яких можлива лише при конструюванні нової техніки.

Працівник ЕОМ як і інші часто взаємодіє з іншими людьми, тому також доволі часто виникають питання міжособистісних взаємин, що включають психологічні і соціально-психологічні аспекти.

Узагальнюючи вищесказане, на працівника ЕОМ впливають чотири групи факторів трудового середовища: ергономічні, фізичні, соціально-психологічні та інформаційні [27].

Відповідно до ГОСТ 12.0.003-74 ССБТ «Небезпечні та шкідливі виробничі фактори. Класифікація» (який діє до 1 січня 2022 року) всі виробничі чинники поділяються на шкідливі та небезпечні. Небезпечні і шкідливі виробничі чинники в свою чергу поділяються на фізичні, біологічні, хімічні та психофізіологічні чинники.

Небезпечний виробничий чинник – виробничий чинник, вплив якого на працівника в певних умовах призводить до травм, отруєння, іншого раптового різкого погіршення здоров’я або до смерті. Шкідливий виробничий чинник – виробничий чинник, вплив якого за певних умов може призвести до захворювання, зниження працездатності і (або) негативно позначитися на здоров’ї нащадків [28].

Працівники ЕОМ найчастіше піддаються впливу фізичних і психофізіологічних виробничих чинників. Є також імовірність впливу

хімічних та біологічних чинників, але вона досить незначна, але відразу варто зазначити, що вона значно зростає в переповнених і неправильно вентильованих приміщеннях.

Робота на ЕОМ передбачає візуальне сприйняття відображеної на екрані монітора інформації, тому значне навантаження піддається саме зоровий апарат. Симптоми порушення зору можна умовно розділити на дві групи:

- очні симптоми (біль, печіння, роздратування, почервоніння, свербіж);
- зорові симптоми (пелена перед очима, двоїння або миготіння).

За даними ВООЗ зорові порушення спостерігаються у 45-94% користувачів ПЕОМ час від часу, а у 15-45% - щодня.[29]

Можна виділити наступні основні порушення здоров'я користувачів ЕОМ:

- зоровий дискомфорт і хвороби органів зору;
- розлади ЦНС і хвороби серцево-судинної системи;
- перенапруження опорно-рухової системи;
- захворювання шкіри;
- порушення репродуктивної функції.

Крім того, виявлено також негативний вплив на інші системи організму - зниження імунітету, аритмія, атеросклероз, гіпертонія, інфаркт міокарда, хвороби органів травлення (багато з яких виникають через малорухливу працю) і інші.

Аналіз умов праці користувачів ПЕОМ дозволяє визначити напрямки профілактики порушень здоров'я. Основними напрямками профілактики є наступні:

- раціональна організація режиму праці та відпочинку;
- раціональна організація робочого простору;
- технічні засоби профілактики;

- медичні способи забезпечення здоров'я та оптимальної працездатності.

Види трудової діяльності поділяються на три групи:

- розробники програм (інженер-програміст);
- оператор ЕОМ;
- оператор комп'ютерного набору.

Тривалість регламентованих перерв за робочу зміну в залежності від виду трудової діяльності на ЕОМ наведена в табл. 5.1.

Таблиця 5.1

Тривалість регламентованих перерв користувачів ПЕОМ в залежності від категорії робіт

Категорія роботи	Загальний час регламентованих перерв	
	При 8-годинній зміні	При 12-годинній зміні
Розробники програм	15 хв через кожну годину роботи	Перші 8 годин роботи перерви аналогічні 8-годинній зміні, а протягом наступних 4 годин - 15 хв через кожну годину роботи
Оператори ЕОМ	15 хв через кожні 2 години роботи	
Оператори комп'ютерного набору	10 хв через кожну годину роботи	

Під час регламентованих перерв з метою зниження у користувачів нервово-емоційної напруги, втоми зорового аналізатора, запобігання розвитку втоми бажано виконувати комплекси спеціальних профілактичних вправ таких як: кругові рухи голови, переклад погляду з найближчих точок на далекі одним оком, знизування плечима, загальне потягування, потягування м'язів спини, напруга нижній частині спини, вироблення правильного миготіння, швидке миготіння, вправу на змикання повік і інші в залежності від симптомів.

З метою зменшення негативного впливу монотонності роботи рекомендовано періодично змінювати темпу роботи. При високому рівні

напруженості роботи на ПЕОМ необхідно проводити психологічне розвантаження в спеціально обладнаних приміщеннях під час перерв та\або в кінці робочого дня.

Трудовий процес здійснюється в певних умовах виробничого середовища, які характеризуються сукупністю елементів і факторів матеріально-виробничого середовища. Далі я розгляну мої умови праці, як розробника програмного забезпечення. Для роботи використовується наступне обладнання: ноутбук Lenovo з монітором на 13.6 дюймів, Монітор Samsung на 23.5 дюйма, лазерний принтер HP DeskJet 2070. Робоче місце знаходиться в приміщенні, довжина якого 6 м, ширина - 4 м, висота - 3 м, висота робочого стола 1м, відстань між моніторами за очами 70см. Рівень шуму в приміщенні 60 дБ, освітленість робочого місця становить 450 лк. Повітря робочої зони має наступні параметри: температура - 21 ° С, вологість - 60%, швидкість руху повітря не перевищує 0,1 м / с. Тривалість зосередженого спостереження становить 60%.

5.3 Дії працівників в надзвичайних ситуаціях

В роботі я наводжу порядок надання домедичної допомоги постраждалим для декількох хвороб які можуть статися на робочому місці.

1. Порядок надання домедичної допомоги постраждалим при підозрі на інсульт

Згідно з документом: «Порядок надання домедичної допомоги постраждалим при підозрі на інсульт»[30] інсультом вважається гостре порушення мозкового кровообігу, що спричинює ушкодження тканин мозку і розлади його функцій. Існують наступні ознаки інсульту: втрата рівноваги, асиметрія обличчя, раптова слабкість, втрата свідомості, порушення мовлення.

Існує наступна послідовність дій для надання постраждалому домедичної допомоги, а саме: негайно викликати швидку допомогу, перевести постраждалого в горизонтальне положення, не давати їсти та пити, постійно моніторити стан постраждалого до приїзду бригади допомоги, при погіршенні

стану до приїзду бригади потрібно повторно зателефонувати в диспетчерську швидкої допомоги.

2. Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою

Згідно з документом: «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою»[31] електротравмою вважається – місцеве і загальне пошкодження, що виникають у результаті впливу електричного струму великої сили або розряду атмосферної електрики (блискавки).

Існує наступна послідовність дій для надання постраждалому домедичної допомоги, а саме: переконатися у відсутності небезпеки для себе, якщо постраждалий під струмом то потрібно вимкнути джерело струму, або витягнути постраждалого від дії струму безпечно для себе, провести огляд постраждалого, викликати швидку, надати постраждалому стабільне положення, постійно перевіряти дихання та пульс, при затриманні швидкої допомоги повторно зателефонувати до швидкої допомоги.

3. Порядок надання домедичної допомоги постраждалим при травмах та пошкодженнях очей

Згідно з документом: «Порядок надання домедичної допомоги постраждалим при травмах та пошкодженнях очей» [32] пошкодженням очей вважається вплив на орган зору різних пошкоджуючих факторів, що може викликати порушення його функції або втрату зору.

Існує наступна послідовність дій для надання постраждалому домедичної допомоги, а саме: переконатися у відсутності небезпеки, провести огляд самостійно, викликати швидку допомогу, допомогти постраждалому зайняти зручне положення, якщо попало в око пісок чи бруд то обережно промити око, при попаданні гострого предмета чи іншого відносно великого предмету не видаляти самостійно його і зафіксувати око, постійно перевіряти стан постраждалого, при затриманні швидкої допомоги повторно зателефонувати до швидкої допомоги.

4. Загальні рекомендації для надання постраждалому допомоги.

Згідно з документом: «Про затвердження порядків надання домедичної допомоги особам при невідкладних станах»[33]

Існує наступна послідовність дій для надання постраждалому домедичної допомоги, а саме: переконатися у відсутності небезпеки для себе, провести огляд постраждалого, викликати екстрену медичну (швидку) допомогу, перевірити дихання та пульс людини, перевести в горизонтальне положення та постійно моніторити стан постраждалого.

Згідно з документом: «Про затвердження Правил пожежної безпеки в Україні»[34]

До первинних засобів пожежогасіння відносяться:

- Вогнегасники
- Покривала з негорючого теплоізоляційного полотна, грубововняної тканини або повсті
- Ящики з піском
- Бочки з водою
- Пожежні відра
- Совкові лопати
- Пожежний інструмент (гаки, ломи, сокири тощо)

Пожежні щити (стенди) встановлюються на території об’єкта з розрахунку один щит на площу 5000 м2.

До комплекту засобів пожежогасіння, які розміщаються на ньому, входять: вогнегасники – 3 шт., ящик з піском – 1 шт., покривало з негорючого теплоізоляційного матеріалу або повсті розміром 2 х 2 м – 1 шт., лопати – 2 шт., гаки – 3 шт., ломи – 2 шт., сокири – 2 шт.

Ящики для піску повинні мати місткість 0,5, 1,0 або 3,0 $м^3$ та бути укомплектованими совковою лопатою.

Будівлі та споруди, які зводяться або реконструюються, мають бути забезпечені первинними засобами пожежогасіння з розрахунку:

- на 200 м2 площі підлоги – один вогнегасник, бочка з водою, ящик з піском;
- на кожні 20 м довжини риштування (на поверхах) – один вогнегасник (але не менше двох на поверсі), а на кожні 100 м довжини риштування – бочка з водою;
- на 200 м2 площі покриття з утеплювачем та покрівлями з горючих матеріалів груп Г3, Г4 – один вогнегасник, бочка з водою, ящик з піском;
- на кожну люльку агрегату для будівництва градирень – по два вогнегасники;
- у місці встановлення теплогенераторів, калориферів – два вогнегасники та ящик з піском на кожний агрегат.

У вищезазначених місцях потрібно застосовувати вогнегасники пінні чи водяні місткістю 10 л та\або порошкові місткістю не менше 5 л.

На території будівництва в місцях розташування тимчасових будівель, складів, майстерень встановлюються пожежні щити (стенди) та бочки з водою.

Приміщення, в яких розміщено оргтехніку, слід оснащувати переносними газовими вогнегасниками з розрахунку один вогнегасник ВВК-1,4 чи ВВК-2, але не менше ніж один вогнегасник зазначених типів на приміщення.

ВИСНОВКИ

В магістерській дипломній роботі було досліджено та проаналізовано принципи роботи хмарних обчислень. Розглянуті сучасні напрямки розвитку ПЛІС технологій. Окреслено актуальність та проблеми створення обчислювальних пристроїв на їх основі.

Були розглянуті види реконфігурацій ПЛІС (динамічна та статична), види динамічної реконфігурації (повна та неповна), та шляхи її впровадження при проектуванні процесорів. Також були оглянуті типи хмарних служб, проаналізовані їх недоліки та переваги, і окремо була розглянута хмарна служба FaaS з огляду на використання ПЛІС в роботі.

Було спроектовано процесор з можливістю часткової динамічної реконфігурації та з перспективою його використання в хмарних обчисленнях.

Процесор був спроектований на мові VHDL, був зроблений синтез та проаналізовані часові діаграми. Проектувався він під конкретну плату, а саме під Nexyx A7. Це середня за потужністю, що як раз підійшло під необхідність динамічного реконфігурування. Сутність роботи полягає у виконані двох математичних операцій, а саме зведення числа в степінь та розв’язання квадратного рівняння. Після вибору операції відбувається часткова динамічна реконфігурація обраного компонента, він виконує свою операцію, передає дані в хмарне ядро, яке проводить основні математичні операції та відправляє результат роботи відображатись на світлодіоди.

Для знаходження кореню квадратного використовувались два методи: CORDIC та половинного ділення, після синтезу та порівняння використовуваних елементів, було виявлено, що використання методу CORDIC є більш ефективним.

Результати дипломної роботи були опробовані на двох конференціях [35, 36].

ПЕРЕЛІК ПОСИЛАНЬ

1. Вакалюк Т.А. Можливості використання хмарних технологій в освіті /Т.А. Вакалюк // Актуальні питання сучасної педагогіки. Матеріали міжнародної науково-практичної конференції (м. Острог, 1-2 листопада 2013 року). – Херсон: Видавничий дім «Гельветика», 2013. – С. 97–99.
2. Stephen Watts, Muhammad Raza - SaaS vs PaaS vs IaaS: What's The Difference & How To Choose [Електронний ресурс] – Режим доступу: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/
3. Виноградов Ю.Н. Выбор архитектуры конфигурируемого процессора для облачных вычислений/Ю.Н. Виноградов, А.М. Сергиенко, В.П. Симоненко// Міжнародна конференція "Високопродуктивні обчислення" – Київ, 2012. С.122-127.
4. Malik N.A. Threat modeling in pervasive computing paradigm [Text] / N.A. Malik, M.Y. Javed, U. Mahmud // Proceedings of the Mobility and Security, New Technologies, Tangier, November 5-7, 2008 y. - pp. 1-5
5. Kumar S. Fundamental limits to Moore's law [Електронний ресурс] / S. Kumar // arXiv preprint arXiv:1511.05956. – 2015. – Режим доступу : https://www.researchgate.net/profile/Suhas_Kumar5/publication/284219009_Fundamental_Limits_to_Moore's_Law/links/5663fd9408ae192bbf901e85.pdf
6. George A. Novo-G: At the forefront of scalable reconfigurable supercomputing / A. George, H. Lam, and G. Stitt // 2011 Computing in Science & Engineering. – IEEE, 2011. – Vol. 13 (1). – P. 82 – 86.
7. Каляев И. А. Высокопроизводительные реконфигурируемые вычислительные системы на основе плис Virtex-6 и Virtex-7 / И. А. Каляев, А. И. Дордопуло, И. И. Левин, Е. А. Семерников // Параллельные вычислительные технологии (ПаВТ'2012) (Новосибирск, 26—30 марта 2012 г.). — Челябинск : Издательский центр ЮУрГУ, 2012. — С. 449—458.

8. Сергієнко А. М. Архітектура комп'ютерів: Конспект лекцій / А. М. Сергієнко. – К. : ІСЗЗІ НТУУ «КПІ», 2013. – 198 с.
9. Estrin G. Organization of computer system: the fixed plus variable structure computer / G. Estrin. // Western Joint IRE-AIEE-ACM Computer Conference New York, 2003. — P. 33—40.
10. Дунець Р. Б. Проблеми побудови частково реконфігурованих систем на ПЛІС / Р. Б. Дунець, Д. Я. Тиханський // Радіоелектронні і комп'ютерні системи, 2010. — № 7 (48). — С. 200—204.
11. Intel Cyclone [Електронний ресурс] - Режим доступу: https://www.intel.com/content/www/us/en/products/programmable/cyclone-series.html
12. Intel Stratix [Електронний ресурс] - Режим доступу: https://www.intel.ru/content/www/ru/ru/products/programmable/stratix-series.html
13. Intel Arria. [Електронний ресурс] - Режим доступу: https://www.intel.com/content/www/us/en/products/programmable/arria-series.html.
14. Xilinx Spartan-7 [Електронний ресурс] - Режим доступу: https://www.xilinx.com/products/silicon-devices/fpga/spartan-7.html
15. Xilinx Artix-7 [Електронний ресурс] - Режим доступу: https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html
16. Xilinx Kintex-7 [Електронний ресурс] - Режим доступу: https://www.xilinx.com/products/silicon-devices/fpga/kintex-7.html
17. Xilinx Virtex-7 [Електронний ресурс] - Режим доступу: https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html
18. 7 Series FPGAs Data Sheet: Overview. [Електронний ресурс] - Режим доступу: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

19. Methods of computing square roots [Електронний ресурс] - Режим доступу: https://en.wikipedia.org/wiki/Methods_of_computing_square_roots
20. Бікташева С. Р. CORDIC-метод обчислення квадратного кореня/С. Р. Бікташева, Л. В. Мороз, М. Ю. Стахів//Комп'ютерні системи проектування. Теорія і практика, 2010. №685. – С. 152-155.
21. Алгоритм методу половинного ділення [Електронний ресурс] - Режим доступу: https://math.semestr.ru/optim/dichotomy-algorithm.php
22. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207, затверджені наказом Міністерства соціальної політики України
23. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» від 01.12.1999 №37, затверджено постановою Міністерства охорони здоров'я України
24. «Про затвердження Порядку проведення медичних оглядів працівників певних категорій» від 21.05.2007 №246, затверджено наказом Міністерства охорони здоров'я України
25. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» від 01.12.1999 №42, затверджено наказом Міністерства охорони здоров'я України
26. Ергономічні обґрунтування й оцінки у безпеці життєдіяльності [Електронний ресурс] – Режим доступу до ресурсу: http://opcb.kpi.ua/wp-content/uploads/2014/09/%D0%A2%D0%B5%D0%BC%D0%B0-5.-%D0%95%D0%A0%D0%93%D0%9E%D0%9D%D0%9E%D0%9C_%D0%A7%D0%9D_%D0%9E%D0%91_%D0%A0%D0%A3%D0%9D%D0%A2%D0%A3%D0%92%D0%90%D0%9D%D0%9D%D0%AF.pdf
27. Охрана труда в автоматизированном производстве [Електронний ресурс] – Режим доступу до ресурсу: http://www.dgma.donetsk.ua/docs/kafedry/hiop/metod/20_otofam.pdf
28. Охорона праці. Навчальний посібник [Електронний ресурс] – Режим доступу до ресурсу: http://vlp.com.ua/node/6699?page=0,1

29. Світові епідеміологічні характеристики поширеності порушень зору [Електронний ресурс] – Режим доступу до ресурсу: https://oculist.in.ua/number3/105-svitovi-epidemiologichni-kharakteristiki-poshirenosti-porushen-zoru.html
30. «Порядок надання домедичної допомоги постраждалим при підозрі на інсульт», від 14.06.2014 №398, затверджено наказом Міністерства охорони здоров'я Україниі
31. «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» , від 14.06.2014 №398, затверджено наказом Міністерства охорони здоров'я України
32. «Порядок надання домедичної допомоги постраждалим при травмах та пошкодженнях очей» , від 14.06.2014 №398, затверджено наказом Міністерства охорони здоров'я України
33. «Про затвердження порядків надання домедичної допомоги особам при невідкладних станах» від 16.06.2014 №398, затверджено наказом Міністерства охорони здоров'я України
34. «Про затвердження Правил пожежної безпеки в Україні» від 30.12.2014 №1417, затверджено наказом Міністерства внутрішніх справ України
35. Оганесов Б.А. Можливості розробки реконфігуруємого процесора для хмарних обчислень з використанням ПЛІС / Б.А. Оганесов // Всеукраїнська конференція студентів та молодих вчених. – 2020. – С. 17.
36. Оганесов Б.А. Можливості розробки реконфігуруємого процесора для хмарних обчислень з використанням ПЛІС / Б.А. Оганесов // XIV Міжнародна науково-практична конференція. – 2020.