НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК __ 004.896____

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Апаратний блок навчання нейронних мереж генетичним алгоритмом»

Виконав:
студент VI курсу, групи ІТ-93мп
Новохацький Ілля Валерійович _____

Керівник:
Професор каф. АУТС, д. ф.-м. н., професор
Дорошенко Анатолій Юхимович _____

Консультант з реалізації апаратного блоку:
Старший викладач каф. АУТС
Шимкович Володимир Миколайович _____

Рецензент:
Доцент каф. ТК, к.т.н., доцент,
Тимошин Юрій Афанасійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Новохацького Іллі Валерійовича

1. Тема дисертації «Апаратний блок навчання нейронних мереж генетичним алгоритмом», науковий керівник дисертації Дорошенко Анатолій Юхимович, професор, доктор фіз.-мат. наук, затверджені наказом по університету від «26» 10 2020 р. №3132-с

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження: апаратний блок для навчання багатошарових нейронних мереж

4. Вихідні дані: сімейство ПЛІС Xilinx Spartan

5. Перелік завдань, які потрібно розробити: Ознайомлення з предметною областю, аналіз існуючих рішень, ознайомлення з процесом та методами апаратно-програмної реалізації штучних нейронних мереж, розробка апаратного блоку для навчання штучного нейрона, розробка апаратного блоку для навчання штучних нейронних мереж за допомогою генетичного алгоритму

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: Блок схема обчислення результуючого значення нейрона, схема навчання нейронної мережі, граф, що демонструє роботу апаратного блоку генетичного алгоритму, структурна схема роботи модуля навчання та результати емуляції навчання нейрона на ПЛІС

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
7	Шимкович В.М., старший викладач		

9. Дата видачі завдання _____04.09.2020_____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Ознайомлення з предметною областю	03.10.2020	
2	Аналіз існуючих рішень	17.10.2020	
3	Ознайомлення з процесом та методами апаратно-програмної реалізації штучних нейронних мереж	25.10.2020	
4	Розробка апаратного блоку для навчання штучного нейрона	11.11.2020	
5	Розробка апаратного блоку для навчання штучних нейронних мереж за допомогою генетичного алгоритму	29.11.2020	
6	Оформлення ПЗ	05.12.2020	

Студент Ілля НОВОХАЦЬКИЙ

Науковий керівник Анатолій ДОРОШЕНКО

РЕФЕРАТ

У магістерській дисертації було розроблено апаратний блок та його навчання генетичним алгоритмом на ПЛІС багатошарової нейронної мережі. Дисертація містить 100 сторінок, 39 рисунків, 24 таблиці, 33 літературних джерела та 8 додатків.

Актуальність: Актуальність роботи пов’язана з наростаючим інтересом використання штучного інтелекту та нейронних мереж в автономних системах, питаннями збільшення швидкодії навчання багатошарових нейронних мереж.

Такі системні комплекси витрачають багато часу для свого навчання. В будь-якому випадку це допомагає розвиватися новому напрямку в сфері розробки засобів еволюційної апаратури.

Мета: Метою роботи є розробка апаратного блоку для навчання багатошарових нейронних мереж генетичним алгоритмом. За апаратну частину відповідатиме ПЛІС, яка буде запрограмована за допомогою мови опису апаратури VHDL. Таким чином ми зможемо підвищити швидкість навчання цих мереж. Так як апаратні засоби мають функцію паралельних обчислень це дозволить не тільки збільшити швидкодію, а й скоротити використання обчислювального ресурсу.

Задачі: для досягнення мети були поставлені та розв’язані такі завдання:

- розроблено модель апаратного блоку;
- описано кейси його застосування;
- встановлено технології на яких буде рішення;
- реалізовано рішення.

Об’єкт: об’єктом дослідження в роботі є апаратний блок для навчання багатошарових нейронних мереж.

Предмет: предметом дослідження є генетичний алгоритм при його апаратній реалізації функцій активації нейронних мереж для їх навчання.

Ключові слова: АПАРАТНИЙ БЛОК, НЕЙРОННІ МЕРЕЖІ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ПРОГРАМОВАНА ЛОГІЧНА ІНТЕГРАЛЬНА СХЕМА.

ABSTRACT

In the master's dissertation the hardware unit and its training by genetic algorithm on FPGA of a multilayer neural network were developed. The dissertation contains 100 pages, 39 figures, 24 tables, 33 literature sources and 8 appendices.

Relevance: The relevance of the work is due to the growing interest in the use of artificial intelligence and neural networks in autonomous systems, increasing the learning speed of time-consuming multilayer neural networks, and the development of a new direction in hardware design, called evolutionary hardware.

Purpose: Developing a hardware unit for training multilayer neural networks by genetic algorithm on FPGA using the language of hardware description of VHDL integrated circuits, which will significantly increase the speed of their work, due to parallel computing in their hardware implementation with minimal computing resources.

The tasks: To achieve this goal, the following tasks have been set and solved:

- hardware unit model was developed;
- describes the cases of its application;
- to establish technologies on which the decision will be made;
- the solution is implemented.

Object: Hardware unit for learning multilayer neural networks.

Subject matter: The subject of research is a genetic algorithm in its hardware implementation of the functions of activation of neural networks for their training.

Keywords: HARDWARE UNIT, NEURAL NETWORKS, GENETIC ALGORITHM, PROGRAMMED LOGICAL INTEGRATED CIRCUIT.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЛІС – програмована логічна інтегральна схема

ГА – генетичний алгоритм

ДНК – дезоксирибонуклеїнова кислота

FPGA – напівпровідниковий пристрій, що може бути налаштований виробником або розробником після виготовлення

ПКВМ – програмована користувачем вентильна матриця

ASIC – інтегральна схема, спеціалізована для вирішення конкретного завдання

VHDL – мова для проектування апаратури сучасних обчислювальних систем

САП – система автоматизованого проєктування

ПЛБ – нескоммутовані програмовані логічні блоки

БВВ – блоки введення-виведення

LUT – таблиця істини

ВСТУП

Із розвитком науки глибинного навчання та сфери апаратних засобів складність задач які потребують вирішення постійно зростає. Це стимулює розвиток і підвищений інтерес до теми не тільки програмних, але й апаратних обчислень за допомогою високопродуктивних нейрочіпів та логічних схем. Більшість створюваних систем є за своєю природою динамічними системами в яких існує безліч проблем управління та розподілу ресурсів. Це все створює нову галузь для досліджень та велику перспективу для створення дійсно потрібних продуктів.

Для створення пристроїв управління в автоматичних системах велику роль грає процес розпаралелювання даних, що в свою чергу гарантує високу швидкодію алгоритму та пришвидшує процес навчання нейронних мереж.

Одними з основних методів реалізації нейромережевих систем управління є програми, з використанням комп'ютеризованих технологій. Такі системи мають ряд недоліків пов’язаних із їхньою вартістю та обмеженість у швидкості. Це все призводить до непрактичності у використанні подібних систем та значно зменшую їхню ефективність порівняно з логічними схемами. Для більшої практичності потрібно зменшити ціну на такі пристрої та зробити їх швидшими та точнішими у своїй роботі.

Важливу роль у процесі навчання ШНМ відіграє повторюваність та послідовність всіх дій. Притому що більшість систем нехтує процесом паралельного обчислення що допомагає вирішити більшу кількість проблем за менший час. Таким чином це стає важливим питанням для динамічних об’єктів які оперують різного роду складними структурами даних.

Альтернативою вище згаданих факторів є функція розпаралелення процесу навчання та роботи елементів нейромережевих структур. Такі можливості з'являються при створенні апаратного блоку побудованого на програмованих логічних інтегрованих структурах ПЛІС.

Метою роботи є розробка апаратного блоку для навчання багатошарових нейронних мереж генетичним алгоритмом на ПКВМ за допомогою мови проектування апаратури VHDL, що дозволить значно підвищити швидкодію їх роботи, за рахунок паралельних обчислень при їх апаратній реалізації з мінімальним використанням обчислювального ресурсу.

Об’єкт досліджень виступає апаратний блок для навчання багатошарових нейронних мереж.

Предмет досліджень є генетичний алгоритм при його апаратній реалізації функцій активації нейронних мереж для їх навчання.

Актуальність роботи пов’язана з наростаючим інтересом використання штучного інтелекту та нейронних мереж в автономних системах, питаннями збільшення швидкодії навчання багатошарових нейронних мереж.

Такі системні комплекси витрачають багато часу для свого навчання. В будь-якому випадку це допомагає розвиватися новому напрямку в сфері розробки засобів еволюційної апаратури.

Таким чином відбувається розширення області використання, тому постає питання розробки методів підвищення ефективності навчання нейронних мереж та використання генетичних алгоритмів з цілого набору критеріїв. Щоб вирішити дане питання потрібно реалізувати апаратний блок за допомогою якого навчити різного роду нейронні мережі та заміряти результати.

1 ОГЛЯД ТА АНАЛІЗ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

1.1 Штучний нейрон

Вузлом ШНМ називається такий елемент що відповідає має всі характеристики притаманні нейрону головного мозку. Проте така модель має спрощену структуру на відміну від природної моделі.

Мережа створюється таким чином коли нейрони поєднуються один з одним за допомогою певних зв’язків. Таким чином інформація переходить з одного шару нейронів до іншого і об’єднуються в мережі.

На рисунку 1.1 показаний штучний нейрон з трьома вхідними векторами, таким чином з’єднання зважені та мають найменування $w_1,\ w_2,\ w_3$.

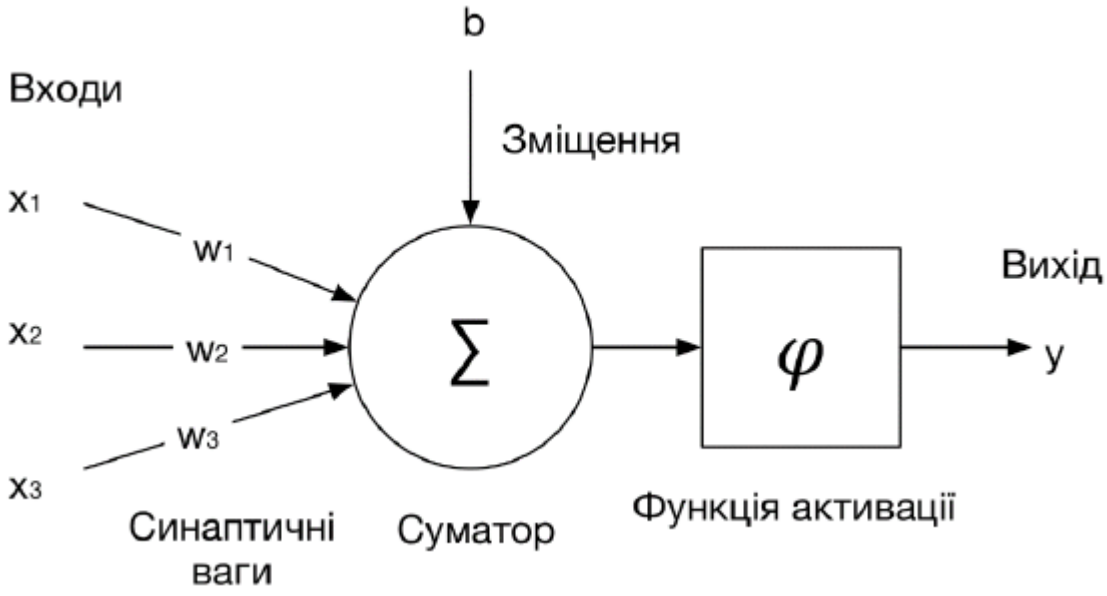

Рисунок 1.1 – Модель штучного нейрона[23]

Нехай до синапсів приходять імпульси таких сил як $x_1,\ x_2,\ x_3$, в результаті до нейрона надходять імпульси силою $w_1x_1,\ w_2x_2,\ w_3x_3$. Результуючий імпульс сил дорівнює сумі імпульсів:

$$w_1x_1 + w_2x_2 + w_3x_3 \tag{1.1}$$

Функція, що позначає вихідний імпульс записується як:

$$y = f(x) = f(w_1x_1 + w_2x_2 + w_3x_3) \quad (1.2)$$

Описати модель нейрону можна цілком за допомогою функції активації та ваговими коефіцієнтами. Після отримання якихось даних на вхід ми генеруємо вихід в залежності від передатної функції.

Як результат, вихідне значення нейрону залежить від зваженої суми всіх його входів. Прийнято використовувати певну функцію для позначення цього виразу. В природі існують різного роду функції які називатимуться функціями активації або передатними функціями.

На рисунку 1.2 представлено активаційну функцію що симулює певний поріг для отримання результату.

Значення функції активації може варіюватися в залежності від вхідного значення. Таким чином, що якщо воно менше ніж поріг, то результуюче значення дорівнює мінімальному, в іншому випадку – максимальному.

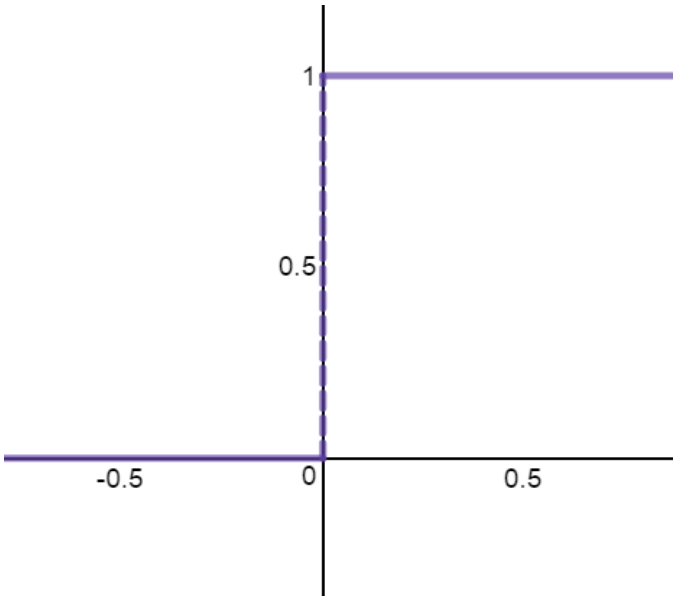

Рисунок 1.2 – Порогова функція активації

Лінійна функція активації представлена на рисунку 1.3, вона має лише дві ділянки в яких функція дорівнює максимальному та мінімальному значенням відповідно. Така функція має лінійну природу та потенційно зростає з однаковою кількістю кроків за раз.

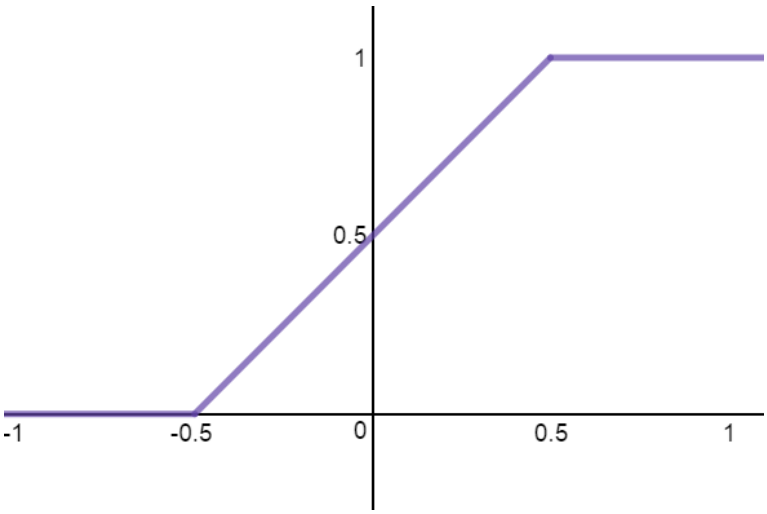

Рисунок 1.3 – Лінійна функція активації

На початку розвитку сфери нейронних мереж застосовувалися в основному такі передатні функції що зображені на рисунку 1.2 та рисунку 1.3. На рисунку 1.4 зображена сигмоїда. Це функція яка дає змогу активуватися плавно та дати точніший результат. За рахунок плавності переходу сигмоїдальна функція не перенасичується від сильніших сигналів та не слабне від дуже слабких. Це робить її найкращим варіантом поміж усіх передатних функцій.

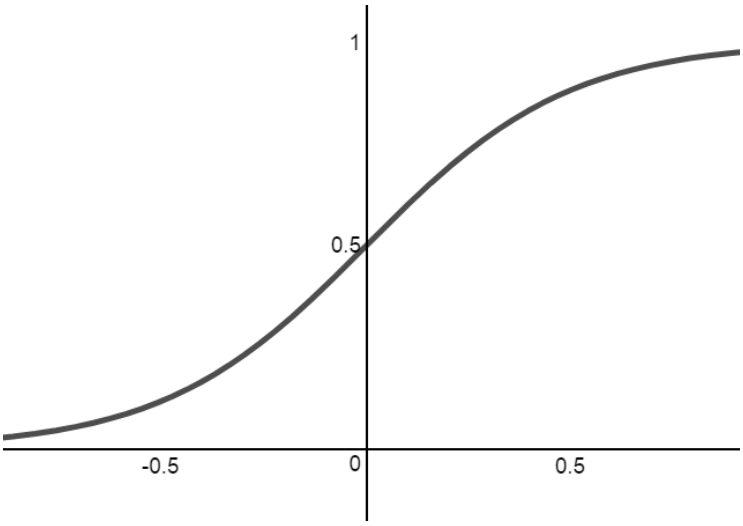

Рисунок 1.4 – Сигмоїдальна функція активації

В даному випадку була обрана сигмоїдальна функція за її більшу точність в показі результатів та більш плавну природу яка допомагає приймати правильні рішення.

1.2 Штучні нейронні мережі. Будова та архітекртура

Штучна нейронна мережа – це математична модель, що копіює структуру природньої моделі та представляє собою систему з'єднаних і взаємодіючих між собою штучних нейронів.

Кожен нейрон отримує сигнали від сусідніх нейронів по спеціальним нервовим волокнам. Такі сигнали можуть мати як позитивний так і негативний характер. Їхня сума складає результуюче значення нейрона. З того моменту коли сума приймає деяке порогове значення використовуючи певну функцію, то нейрон переходить в новий стан і передає цей сигнал далі.

Кожна мережа має свою архітектуру в якій існує різноманітне поєднання нейронних зв'язків між собою. Таким чином ми можемо проектувати довільну кількість мереж із різною архітектурою, правилами навчання і можливостями.

ШНМ в більшості випадків зображаються як граф з вершинами та ребрами. Існує два основних класи це мережі з прямим розповсюдженням помилки та рекурентні, що мають за зворотні зв'язки.

Багатошарові перцептрони це одне з найбільш поширених сімейств мереж першого класу, у них нейрони розташовані шарами і мають однонаправлені зв'язки між шарами.

Мережі прямого розповсюдження є в деякому сенсі статичними, тому що на заданий вхід вони виробляють одну множину вихідних значень, незалежних від попереднього стану мережі.

Рекурентні мережі є динамічними, оскільки через зворотні зв'язки відбувається модифікація вхідних векторів, а це в свою чергу призводить до різного роду непередбачуваних результатів мережі.

Структурний склад ШНМ це декілька шарів, що мають своє призначення. Обов'язковим є вхідний шар він обробляє зовнішні сигнали та передає далі в мережу прихованих шарів. Далі відбувається обробка цих сигналів прихованими шарами та передача результату до вихідного шару. Він в свою чергу накладає

результуючі сигнали від прихованих шарів на функцію активації та формує відповідь мережі. Дані процеси зображено на рисунку 1.5.

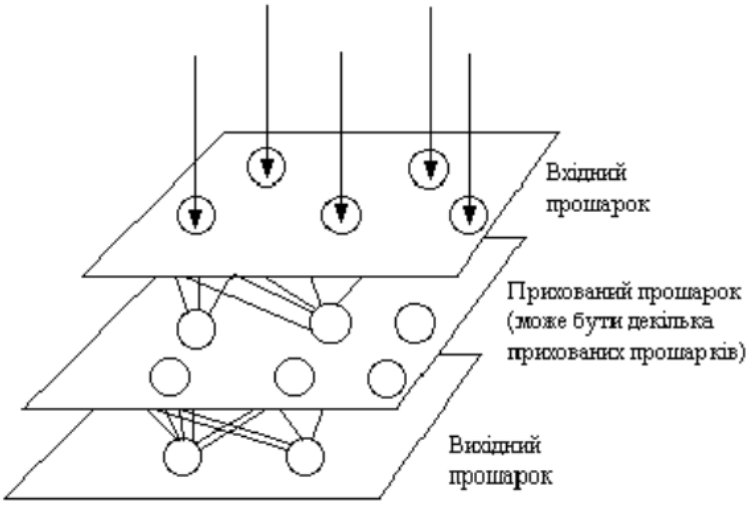

Рисунок 1.5 – Схема нейронної мережі[6]

Топології ШНМ, залежно від того чи містять вони зворотні зв’язки, чи ні:

- ШНМ без зворотних зв’язків;
- ШНМ зі зворотними зв’язками;
- ШНМ із прямими і зворотними зв’язками;
- ШНМ із непрямими та зворотними зв’язками;
- ШНМ із латеральними зв’язками;
- ШНМ із повнозв’язними зв’язками.

ШНМ прямого поширення працює таким чином що потребує зв’язків між різними шарами мережі, прицьому схованих шарів має бути декілька. Якщо існує зв’язок лише між двома сусідніми шарами то такі мережі іменуються мережами першого порядку, прицьому такі зв’язки називаються пошаровими. Проте якщо ж кожен нейрон з’єднаний з кожним нейроном іншого шару, то така мережа є повнозв’язною.

Для систем реального часу та для інших динамічних об’єктів потрібно використовувати ШНМ зворотного поширення. Це допомагає у вивченні та дослідженні систем для вирішення проблем різної складності. Такі системи

спроможні використовувати інформацію під час навчання для всієї своєї структури це допомагає ділитися даними та приймати вірні рішення в таких системах.

На рисунку 1.6 зображено ШНМ до якої надходять вихідні сигнали що поширюються прямими зворотними зв’язками. Це в свою чергу допомагає такій мережі вирішувати більшість стандартних задач машинного навчання, таких як розпізнавання образів або задач розпізнавання тексту.

Використання даної мережі не входить в нашу роботу так як граничний стан активації досягається в більш пізньому етапі та не надає таких результатів які потрібно щоб вирішити задачі точного обчислення від яких може залежати доля людини.

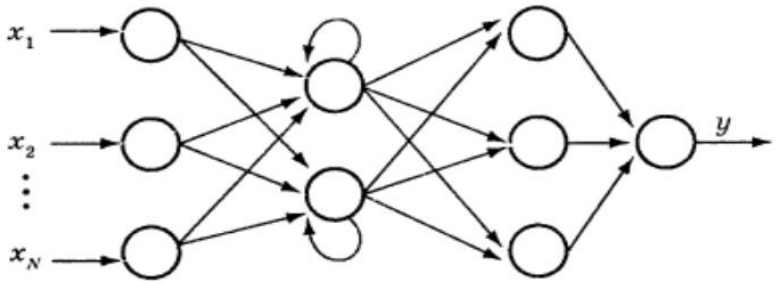

Рисунок 1.6 – ШНМ із прямими і зворотними зв’язками

На рисунку 1.7 показано ШНМ із латеральними зв’язками. Це такі дивні зв’язки, що сформовані тільки між нейронами що належать до одного шару. Даний тип мереж використовується коли потрібно зробити активними лише один шар з цілої групи нейронів. Він гарно пристосований до різного роду задач класифікації та розпізнавання коли ми маємо певні характеристики що мають кардинально більшу вагу ніж інші.

В такому разі відбувається обмін сигналами. В нашому випадку на вхід передається послаблюючий сигнал якщо поріг не подолано у активаційної функції, інакше передається посилюючий сигнал для зворотнього шару нейронної мережі. Це все робить навчання можливим так як у кожного нейрона своя реакція на дані.

Коли найбільш активний нейрон придушує інших своєю активністю то кажуть, що це топологія мережі переможець отримує все. Такі мережі використовуються у вирішенні задач класифікації.

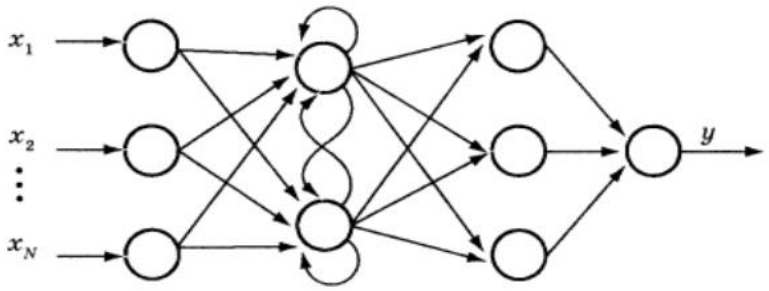

Рисунок 1.7 – ШНМ із латеральними зв’язками

Повнозв’язні ШНМ характеризуються наявністю зв’язків між усіма нейронами мережі, вона зображена на рисуноку 1.8. Цей вид топології відомий також як мережа Хопфілда.

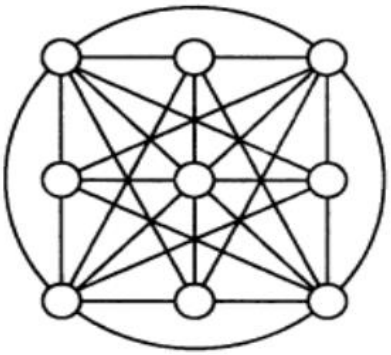

Рисунок 1.8 – ШНМ із повними зв’язками

Дана мережа має чітку особливість та перевагу над іншими. Її повна зв’язність забезпечується методом з’єднання кожного нейрону з кожним. Таким чином створюється мережа що обмінюється своїми даними із кожним елементом. Це робить її кращою для вирішення багатьох задач класифікації та кластеризації.

Сигмоїдальна функція це найбільш розповсюджений тип активаційних функцій. Сигмоїдальні функції є монотонно зростаючими, безперервними і

диференційованими. Диференційованість є важливою властивістю для деяких методів навчання і аналізу. Мають універсальні апроксимаційні властивості.

При насиченні мережі від сильних сигналів виникає проблема придушення більш сильними нейронами слабких. Для вирішення цієї проблеми використовують іншу функції активації що спроможна не насичуватись від сильних сигналів.

Одна з причин через яку сигмоїда використовується в нейронних мережах – це простий вираз її похідної через саму функцію, що дозволило істотно скоротити обчислювальну складність методу зворотного поширення помилки, зробивши його придатним на практиці.

Не менш важливою причиною введення нелінійності є математично доведена можливість отримати як завгодно точне наближення будь-якої неперервної функції багатьох змінних, використовуючи операції додавання та множення на число, суперпозицію функцій, лінійні функції, а також одну довільну неперервну нелінійну функцію однієї змінної.

Таким чином в цьому розділі було розглянуто будову нейрону та нейронних мереж. Було описано модель штучного нейрону та його складові елементи. Ми дійшли висновку навіщо нейрону функція активації та які їх види існують.

Також ми розглянули різні архітектури багатошарових нейронних мереж та розібралися з їхніми особливостями.

Переглянули схему та структуру багатошарової мережі та яким чином ми можемо з’єднувати окремі нейрони між собою щоб сворити мережу того чи іншого виду.

2 АНАЛІЗ МЕТОДІВ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

2.1 Поняття про навчання нейронних мереж

Створення правильної реакції на подані вхідні порти є основною властивістю ШНМ. Саме це допомагає приймати нові, а іноді непередбачувані рішення.

Для створення систем для обробки великих даних потрібно з’єднати нейрони в мережу. Така модель пристосована до різного роду змін в середовищі. Після обробки сигналів активаційною функцією відбувається передача цих сигналів далі по ланцюжку наступному шару нейронів та формування все нових і нових сигналів.

Існують такі різновиди підходів до навчання ШНМ:

- через створення нових та видалення існуючих з’єднань відбувається шляхом зміни конфігурації;
- через утворення нових елементів вагових коефіцієнтів;
- через перетворення параметрів передатньої функції.

Вид перетворення залежить від архітектури нейронної мережі, засобами передачі та обробки окремих процесів, що виконуються в паралельних потоках та обмінюються даними поміж собою.

Сучасний підхід який отримав багато розголосу і підтримки коли структура мережі задається до самого досвіду навчання, а мережа навчається шляхом налаштування зв’язків матриці(вагових коефіцієнтів).

Кінцевий результат мережі залежить від правильності побудови матриці. Потрібно впевнитись в тому що всі елементи пов’язані між собою і мають гарну структуру. Після чого потрібно спостерігати аби зміна елементів матриці відбувалася за певним алгоритмом при послідовному поданні мережі деяких векторів, що навчаються.

Передача інформації в мережі відбувається від прихованого попереднього шару до поточного. Прицьому розрізняється зв’язки передачі інформації. Це дозволяє створювати та розширювати мережі в залежності від поставленої задачі.

Для того щоб навчання мережі відбувалося правильно і отриманий результат нас задовільнив потрібно випадковим чином підібрати ваги, але такі щоб під час навчання мережа сама виробляла необхідні вихідні сигнали. Є два основних напрямки навчання мереж: з учителем і без учителя.

Мережі прямого поширення одні з найчастіш використовуваних. Вони відрізняються своєю простотою та легкістю реалізації. Після виконання функції активації результуючий сигнал передається далі в наступний шар в якому відбувається тей самий процес.

Перший тип навчання з учителем припускає, що задано певний набір даних на основі якого мережа навчається і в цьому наборі наперед відомо якому вхідному значенню відповідає яке вихідне значення таким чином начебто сам вчитель наперед вже знає всі відповіді.

Кожен вхідний компонент вже наперед має результат та значення похибки. Порівняння входу і виходу формує цю помилку. На її основі відповідним чином відбувається зміна вагових коефіцієнтів.

Тестові дані передаються на входи до мережі, далі відбувається ініціалізація вагів. Після цього уточнюється значення помилки та формуються масиви різниці між очікуваним та дійсним значеннями. Після цього ці значення мінімізуються і так відбувається навчання. Звичайно багато хто заперечить що це зовсім не справжній варіант навчання та виглядає якось штучно, проте це один із найбільш розповсюджених видів навчання ШНМ.

Спосіб що краще підходить до вирішення задач та класифікації це навчання з вчителем. В цей час до мережі на вхід отримуються певні дані. А тоді мережа сама використовуючи тільки алгоритм прийняття рішень редагує певним чином ваги та сама генерує значення похибки в залежності від результату який видав нам алгоритм навчання.

Реакція мережі до навчання може бути досить непередбачуваною оскільки ми не можемо наперед знати яке вхідне значення вектора з певного класу може викликати який результат. Таким чином алгоритм має створити в процесі навчання певну зрозумілу форму класу і обмежитися ним. Зрозуміло, що це не сильно

скорочує процес ідентифікації зв’язків між вхідними векторами й відповідним результатом мережі.

Один із доволі цікавих способів навчати мережу це навчання з підкріпленням. В даному випадку існує ментор, але який не підказує на відміну від учителя правильних відповідей, а лише дає знати чи правильно відпрацювала мережа вхідні дані чи ні. На основі цього відбувається зміна параметрів мережі та інші налаштування. Розподіляючи відповідним чином значення хибних та правдивих вагів.

На даний час створено велику кількість підходів до навчання мереж. Тут ми розглянемо найбільш відомі варіанти та зупинимося на тих які були використані в роботі.

2.2 Генетичні алгоритми

Згідно з головним правилом еволюційної теорії розвиток і зміна кожного компоненту природи направлена для того щоб краще адаптуватися до змін в навколишньому середовищі.

Процес еволюції це достатньо складний процес спрямований на розумну розробку нових або видалення непотрібних функцій у живих організмів. Таким чином деякі рослини як троянди отримали захисні голки, баран в свою чергу отримав рога для захисту, а людина сильну м’язову та нервову системи за допомогою яких може створювати та використовувати у своїй діяльності більш складні механізми. Можна стверджувати, що еволюція в деякому сенсі, – це процес оптимізації та покращення всіх живих організмів.

Розглянемо, які засоби використовує природа щоб вирішити цю задачу оптимізації.

Основна ідея цього процесу полягає в тому що сильніші особини завжди мають кращі шанси на виживання, а тому вони краще пристосовані до умов середовища. В свою чергу слабкі особини довго не проживуть в середовищі без

суттєвих на то змін які нададуть їм певних переваг для виживання. Як результат сильніші особини принесуть більше нащадків що дозволить їм розвиватися надалі.

Згідно з генетичним правилом в якому йдеться про те що всі діти успадковують від батьків основні їхні якості. Тому якщо батьки були сильними особинами то і їхні нащадки будуть доволі сильними за своєю природою, а їхня кількість буде тільки зростати.

Щоб краще зрозуміти як працює цей механізм потрібно розібратися в основах роботи генетичних алгоритмів, що дають нам повну уяву про те як влаштовано генетичне спадкування в природі.

Кожна клітина містить у собі всю необхідну генетичну інформацію. Ця інформація представлена у вигляді множини дуже довгих молекул ДНК. Кожна молекула ДНК – це ланцюжок, що складається з молекул нуклеотидів чотирьох типів.

Таким чином, порядок нуклеотидів і несе необхідну інформацію. У клітини живої істоти кожна молекула ДНК оточена оболонкою – таке новоутворення називається хромосомою.

Аллелі позначають різні значення гена. Наприклад, ген що відповідає за колір волосся кодує лише певний колір волосся. В свою чергу гени складаються з визначеної кількості хромосом.

При розмноженні живих істот відбувається злиття двох полових кліток і їхні ДНК поєднуються між собою, створючи таким чином ДНК нащадка. Схрещування це основний спосіб взаємодії між клітинами. За допомогою схрещування ДНК предків діляться на дві частини, а потім міняються своїми половинами.

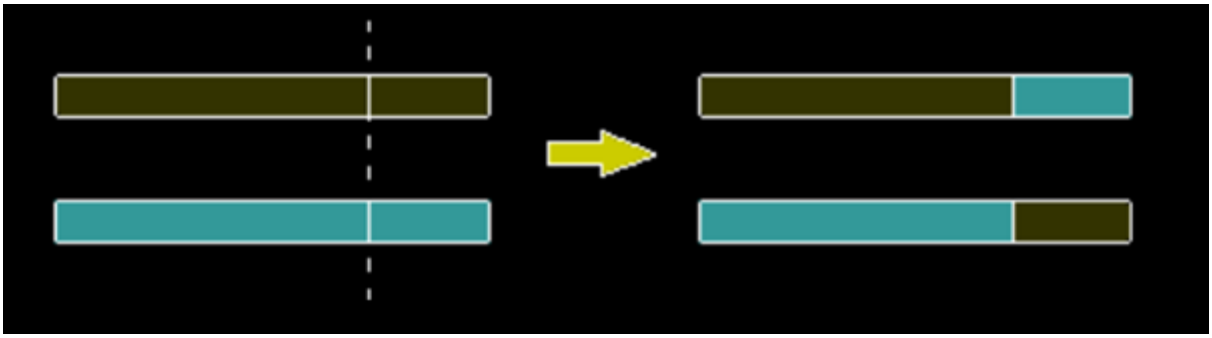

Рисунок 2.1 – Процес обміном генетичної інформації[28]

Мутації виникають через радіоактивність або сторонні впливи. Це все може впливати на спадкування та змінити гени в полових клітинах особин. Таким чином дані гени передадуться до нащадка і додадуть нові функції Якщо ці властивості мають шкідливий вплив то вони скоріш за все не уживуться, а якщо мають позитивний характер то скоріш за все збережуться та будуть передаватися із покоління в покоління.

Генетичний алгоритм – це модель еволюції в природі, яка реалізована у виді комп'ютерної функції.

Генетичний алгоритм симулює еволюцію популяції як деякий циклічний процес схрещування індивідуумів та перерозподілу поколінь.

Операція відбору грає ключову роль у генетичному алгоритмі – це процес формування нової популяції на основі попередньої, після чого стара популяція помирає. Після відбору до нової популяції знову застосовуються операції кросовера і мутації, потім знову відбувається відбір, і так далі.

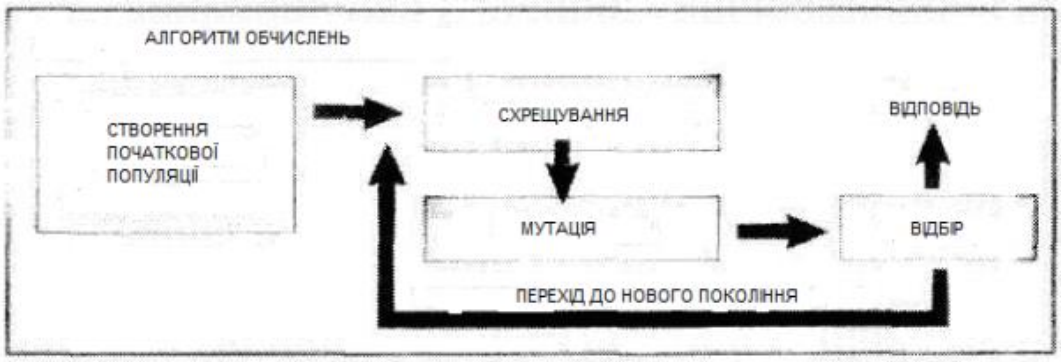

Рисунок 2.2 – Структурна схема роботи генетичного алгоритму[27]

Таким чином, алгоритм обчислення для роботи генетичного алгоритму, зображений на рисунку 2.2 показує декілька важливих стадій в роботі алгоритму. Починаючи зі створення початкової популяції, де відбувається ініціалізація роботи алгоритму.

Далі йдуть дві операції по схрещуванню та мутації, які в свою чергу створюють різні комбінації з початкових індивидів. І закінчується все процесом відбору та формуванням кінцевої відповіді.

В процесі відбору ми маємо відібрати найбільш підходящі і найсильніших індивидів з популяції для точного формування відповіді на поставлену задачу.

2.3 Різновиди та застосування генетичних алгоритмів

Алгоритми побудовані на засадах генетичного виживання мають застосунки для вирішення різноманітних технічних та наукових проблем.

Застосування цих речей не зупиняється на проектуванні та реалізації нових обчислювальних структур. Наприклад, в машинному навчанні використовуються при проектуванні нейронних мереж або керуванням роботами.

Відомі застосування в різних предметних областях, таких як біологія, екологія, імунологія, популярна генетика, різні соціальні, економічні і політичні системи.

Основна частина проблем мають математичну модель формулюються як пошук оптимального значення, де деяке значення – це складна функція, яка залежить від певних вхідних портів. В деяких випадках, потрібно знайти повноцінні значення параметрів, при яких досягається бажане значення функції. В інших випадках, точний результат не потрібен тому рішенням може бути будь-яке значення яке знаходиться в межах границі. В нашому випадку, навчання генетичним алгоритмом – це прийнятний метод для пошуку прийнятних значень.

Можливість маніпуляції багатьма версіями параметрів є основноє властивістю генетичного алгоритму. Це дозволяє використовувати його в багатьох видах прикладних задач. Наприклад у військовій справі чи науково-дослідницьких цілях.

2.4 Методика оптимізації вагів

В цьому підрозділі будуть розглянуті різні способи знаходження екстремуму функції кількох змінних. Дані методи діляться на 3 категорії – нульового, першого і другого порядку.

У методиках для відшукання потрібного нам нульового екстремуму ми опираємося тільки на дані що у нас є про задані точки.

У методиках для відшукання потрібного нам першого порядку екстремуму реалізується функціональний градієнт за певними параметрами:

$$x_{\mathrm{k}+1} = x_{\mathrm{k}} - \alpha_k g_k, \tag{2.1}$$

де x_{k} – вектор параметрів; α_k – параметр швидкості навчання; g_k – функціональний градієнт, що відповідає ітерації з номером k.

Для зменшення значення помилки потрібно використовуючи градієнт рухатися в його сторону та робити невеликі кроки для того щоб не перескочити глобальний мінімум. При русі паралельно змінюються параметри вагових коефіцієнтів таким чином мережа навчається. Проте градієнт при початковому значенні може показувати хибний напрямок.

Одним зі складних рішень є вибір параметра швидкості навчання α_k. При великому значенні параметра α_k швидко відбудеться збіжність, тому існує небезпека пропустити рішення або навіть піти в неправильному напрямку. Стандартним прикладом є ситуація, коли алгоритм дуже швидко просувається, і перестрибує потрібне або прийнятне рішення.

А при малому кроці, ймовірно, буде обрано вірний напрям, але при цьому буде потрібно багато ітерацій та часу щоб дійти до правильного результату.

Для правильного вибору параметру який відповідає за швидкість навчання ми маємо опиратися на наш досвід та алгоритм який використовується. Це дозволить нам підібрати потрібний параметр та вирішити завдання. В більшості випадків підбір цього параметру відбувається експерементальним шляхом.

Коли параметр має характер постійно змінюватися його значення буде ставати все меншим і меншим якщо він правильно рухається в напрямку до свого локального мінімуму. Це робить стратегію навчання швидшою порівнянно з постійним кроком де алгоритм може навіть і не знайти правильного рішення якщо крок буде обрано занадто великим.

Так як пошук мінімуму може відбуватися з двох боків це дасть нам прискорення в пошуках. Якщо алгоритм виконується таким чином він дасть нам швидшу збіжність а значить і швидший спуск.

Спочатку вібувається рух у напрямку протилежному до градієнта, це відбувається через випадкові ваги:

$$p_0 = -g_0 \tag{2.2}$$

Для того щоб визначити наступний крок нам потрібно розглянути що відбувалося на попередньому. Для цього ми маємо скомбінувати попередній крок із напрямком нового руху:

$$p_k = -g_k + \beta_k p_{k-1}, \tag{2.3}$$

де p_k – напрямок руху, g_k – градієнт функціоналу помилки, β_k – ітеративний коефіцієнт з номером *k*.

Визначившись з новим напрямком руху для спуску ми можемо налаштувати нове значення параметрів $x_{\mathrm{k}+1}$ обчислюється за формулою:

$$x_{\mathrm{k}+1} = x_{\mathrm{k}} + \alpha_k p_k \tag{2.4}$$

Визначення кроку використовуючи методи другого порядку задаче не із легких. Для цього потрібно вміти брати похідні другого порядку. В нашому випадку ми використаємо метод Ньютона для знаходження шляху:

$$x_{\mathrm{k}+1} = x_{\mathrm{k}} - H_k^{-1} g_k \tag{2.5}$$

де x_{k} – вектор значень параметрів на *k*-ій ітерації; H – матриця градієнта Гессе на *k*-ій ітерації.

У більшості випадках метод Ньютона сходиться швидше, ніж методи спряженого градієнта, проте такий метод вимагає великих затрат ресурсів через обчислення матриці Гессе.

Для пришвидшення роботи метода Ньютона можна замінити виконання обрахунків матриці схожими виразами, але це в свою чергу породжує так звані квазіньютонові алгоритми.

В деяких випадках побудова нейромоделей шляхом застосування градієнту є неприйнятним. Тому що породжує цілу множину проблем таких як багатоекстремальність цільової функції або недиференційованість функцій активації.

Доцільним є використання алгоритмів що дозволяють на етапі еволюції виконувати пошук і в свою чергу не такі складні що містять похідні високого рівня, що в свою чергу дозволяє ефективніше застосовувати їх для вирішення задач побудови нейромоделей.

Найпоширенішим із варіантів архітектури є багатошарові мережі. Ці нейрони об'єднуються у шари, які отримують вхідні дані лише з одного каналу.

Для вирішення різних завдань використовуються такі методи еволюційної оптимізації як: пошук та підбір характеристик, зміна та налаштунок вагів, архітектурна та проектні рішення щодо мережі, налаштування правил навчання, рандомізація та створення вагових коефіцієнтів, оптимізація значень змінюваних параметрів системи.

Ефектори це вихідні сигнали мережі. Крім вхідного та вихідного шарів, ШНМ має один або кілька прихованих шарів нейронів. Нейрони прихованого шару не мають контактів із зовнішнім середовищем.

Для вирішення задачі управління складними динамічними об'єктами з використанням нейромережевих регуляторів та моделей потрібно щоб нейромережа навчалася швидко з динамікою об'єкта управління, тобто потрібні алгоритми які за найменший відрізок часу навчать мережу та можуть бути легко розпаралелені.

Отже, використання генетичних алгоритмів пов’язано із родом задач для яких їх можна примінити. Вони можуть справлятися як із задачами широкого так і вузького спектрів. Це все залежить від параметрів досліджуваної системи та її основних характеристик.

Передавальна функція f(x) грає ключове місце у визначенні результату мережі від вхідних даних так як напряму залежить від цих параметрів. Для нелінійності роботи нейрона та нейронної мережі в цілому використовують різні передавальні функції.

В цьому розділі було розглянути стратегії та їхню специфіку з навчання мереж. Було розглянуто типові задачі та області застосування таких методик навчання та налаштування параметрів вагів.

Далі було досліджено різні методи по оптимізації вагів та виявлено різну кількість недоліків в кожному з них. Було обрано метод Ньютона як стандартний метод для вирішення задач наукового спектру та оптимізації вагів у вирішенні цих задач.

3 ОГЛЯД ТА АНАЛІЗ ПРОГРАМНИХ І АПАРАТНИХ ПЛАТФОРМ ПО РЕАЛІЗАЦІЇ НЕЙРОННИХ МЕРЕЖ І МЕТОДІВ ЇХ НАВЧАННЯ

3.1 Оптимізаційні компоненти для покращення швидкодії навчання ШНМ

Програмована логічна інтегральна схема – це напівпровідниковий компонент, який налаштовується програмістом після виготовлення.

На відміну від логічного елемента, який має фіксовану функцію, ПЛІС має не визначену функцію під час виготовлення. Перед використанням ПЛІС в схемі вона повинна бути запрограмована.

Існують різні види вентильних матриць, основними серед них є:

- FPGA – схема яка програмується розробником, це найбільш складний за своїм різновидом вид схем;
- CPLD – середня за своєю складністю схема, має у своєму складі енергонезалежну конфігуровану пам’ять;
- PAL – одні з найпростіших схем, представляють собою простий варіант ПЛІС.

Сфери застосування та можливі варіанти проектування електронних компонентів за допомогою ПЛІС представлені нижче:

- системи реального часу що передають та отримують велику кількість даних за один раз;
- різноманітні високошвидкісні передатчики даних;
- обробники сигналів телекомунікаційних веж;
- захист інформації та її маскування за допомогою алгоритмів криптографії;
- квантові комп’ютери та їхні комплектуючі;
- з’єднувальні елементи між системами із непоєднуваною логікою, виконують роль інтеграторів;
- нейроінтерфейси та різноманітні чіпи до них.

Варіанти які можуть застосовуватись як алтернитива ПЛІС:

- БМК – це кристали що дуже схожі на ПЛІС, але потребують програмування на етапі виробництва. Це в свою чергу їх обмежувало, тому вони були замінені на ПЛІС які могли б бути перепрограмованими та не потребували виготовлення на замовлення. Така схема нагадує бібліотеку із функцій для мови програмування;

- ASIC – схема спеціально створена для виконання одного чіткого завдання. Функції в таких пристроях виконуються набагато швидше, а значить дешевше, проте ця схема не має можливості для кастомізації або перепрограмування, що робить її вузьконаправленою.

Сьогодні при створенні нейрообчислювачів, використовується гібридна схема, коли матричний блок обчислень базується на кластерних процесорах, а управління логікою лягає на плечі ПЛІС.

Таким чином матричне ядро буде реалізовано використовуючи нейроінтерфейс, а ПЛІС буде відповідати за правильну роботу логіки та управління.

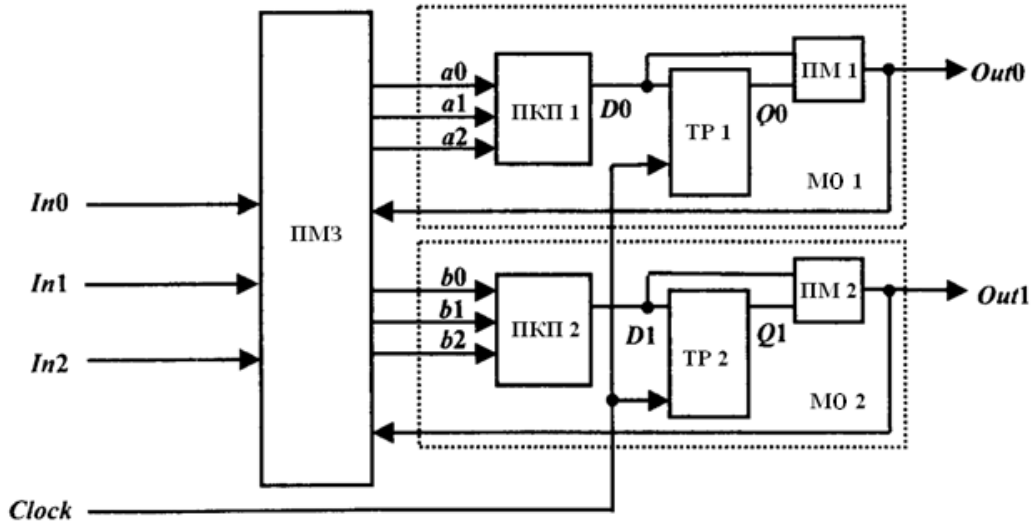

Рисунок 3.1 – Структурна схема ПЛІС[16]

Із рисунку 3.1 видно, що ПЛІС складається з вузлів макроосередків МО1 і МО2, що обведені пунктиром і програмованої матриці з'єднань (ПМЗ). На пристрій подається чотири входи, три з яких (In0, In1 і In2) логічні, а четвертий – Clock,

який подає тактовий сигнал. Виходи пристрою (Out0 і Out1) – це виходи макроосередків.

Сучасні системи містять в собі десятки тисяч макроосередків. Через що вони можуть розпаралелювати велику кількість різних операцій.

Програмований комбінаційний пристрій (ПКП) призначається для реалізації комбінаційних цифрових пристроїв. В даному прикладі вони мають три входи (a2, a1, a0/b2, b1, b0) та один вихід (DO/D1). Здійснюється функціонування таких пристроїв логічними функціями.

ПЛІС надзвичайно гнучка у своєму використанні. Вона має бути запрограмована перед тим як використовуватися. При завантаженні конфігурації функції вводяться в ПКП, після чого вони стають доступними для схем, що виконують запрограмовані логічні функції.

Тригер затримки (ТР1, ТР2) запам'ятовує значення сигналів ПКП, коли на нього подається тактовий вхід сигналу Clock. Тригер в макроосередку дозволяє побудову цифрових автоматів з пам'яттю.

Програмований мультиплексор (ПМ) передає сигнал на вихід макроосередку вихідного сигналу тригера або безпосередньо сигналу з ПКП. У першому випадку макроосередок виконує функції цифрового пристрою з пам’ятю, а у другому – комбінаційного цифрого пристрою.

При проектуванні ПЛІС ми маємо використовувати такі основні програмні компонентів: синтезатор логіки, фіттер і асемблер.

Компілятор повинен проаналізувати користувацький проект (схеми і текстові описи на Verilog HDL або VHDL) і згенерувати нетліст – список всіх елементів схеми і зв’язків між ними.

Нетліст має бути оптимізованим – логічні функції потрібно мінімізувати, можливі дубльовані регістри треба видалити.

Фіттер (fitter) повинен вмістити всю логіку з netlist в наявну архітектуру ПЛІС. Він розміщує логічні елементи і виконує трасування зв'язків між ними (процес place and route). Складність полягає в тому, що різні проекти можуть бути

розміщені в ПЛІС різними способами і цих способів мільйони. Деяке розміщення і трасування виявляється кращим, інші гірші.

Головний критерій якості отриманої системи - максимальна частота, на якій зможе працювати проект при даному розміщенні елементів і при даній трасування зв'язків. Тут впливає довжина зв'язків між логічними блоками і кількість програмованих комутаторів між ними.

На даний момент багато компаній в світі займається розробкою і випуском ПЛІС різного роду, проте, лідирують дві компанії Xilinx та ALTERA. Виділити окремі продукти будь-якої з цих фірм немає можливості, адже схожість виробничого процесу та проектування майже не відрізняються що в свою чергу робить їх заміняємими.

Для прикладу на рисунку 3.2 зображено зовнішній вигляд схеми Kintex-9 приблизно на 400 комірок компанії Xilinx.

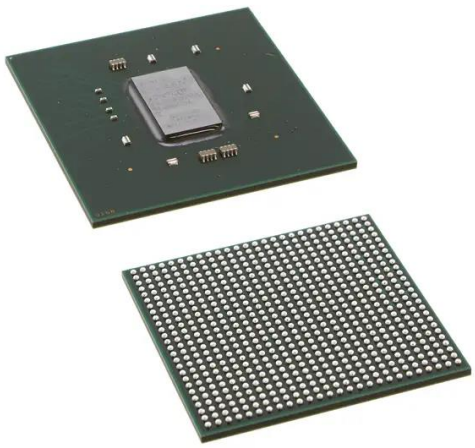

Рисунок 3.2 – Xilinx Inc. Kintex-9 XC7K160T-2FFG568L[6]

Компанія конкурент ALTERA має дуже схожий зовнішній вигляд плат проте містять деякі відмінності які варто зазначити:

- кристал може складаєтися з досить значного обсягу вентилів понад 200 тисяч;
- системні частоти налаштовані чітко тому їх вистачає в подальшій роботі;

- час компіляції рішення може відрізнятися так само як і час проектування здебільшого залежать від досвіду розробника;
- в залежності від типів ціна може відрізнятися дуже сильно;
- є зручне середовище для програмування;
- існує програма для візуального проектування, рішення без коду;
- тип виконання залежить від документації та регламентується в ній;
- споживання енергії знаходиться на досить низькому рівні;
- є програма для створення та замовлення проектів на схеми.

Компанія Xilinx в своїх проектах використовує різного роду конфігурації для своїх схем:

- конфігурація на базі SRAM, яка використовує в своєму арсеналі техніку збереження різних конфігурацій у внутрішньому ОЗУ. Це призводить до того що доводиться робити ініціалізацію через зовнішній масив пам’яті. Така технологія дозволяє пришвидшити роботу ПЛІС проте зменшує кількість ресурсів для використання;
- конфігурація на базі FLASH, яка зберігає роботу всієї програми у внутрішній проте ні від кого незалежній пам’яті. Це дозволяє перезавантажувати програми без втрати даних чи певних результатів. Таку технологію використовують в найбільш відповідальних сферах де потрібна безперебійна чіткість в роботі;
- конфігурація на базі EEPROM, схожа на флеш пам’ять проте використовується на більш бюджетних пристроях. Вона також знаходиться у внутрішній пам’яті та може бути перевантажена у будь який момент часу. Не працює з PC навідміну від флеш пам’яті, потрібен окремий пристрій.

Розглянувши та проаналізувавши предметну область в цьому розділі можна винести певні результати.

Програмована логічна інтегральна схема — це електронний компонент, який застосовується для реалізації інтегральних цифрових схем.

Для того, щоб апаратно реалізувати потрібний нам алгоритм потрібно запрограмувати ПЛІС. Завдяки цьому ми можемо розраховувати на пришвидшення роботи програми без вдосконалення та оптимізації самого алгоритму.

Існують також різні альтернативи ПЛІС які використовуються для побудови різних за складністю і можливостями цифрових пристроїв.

Певну роль відіграє сама схема, адже від її технічних характеристик буде напряму залежати результат та швидкодія виконання програми.

3.2 Реалізація штучних нейронних мереж на багатоядерних процесорах SEAFORTH

Як згадувалось в минулому розділі, нейронні мережі потребують різну кількість необхідних ресурсів, мають певну похибку та навчаються протягом різного часу.

Фактори які впливають на ці чинники: тип ШНМ, архітектура, кількість нейронів та спосіб навчання. Всі ці фактори, атрибути та платформи будуть розглянуті та проаналізовані далі.

В даному прикладі розглянуто різні способи подання нейрона на процесорі даної архітектури. Це в свою чергу дасть нам потрібну швидкодію із розумним використанням ресурсів та невиликою ціною.

Раніше вважалося, що якщо вже існує навчена нейронна мережа з відомими параметрами, то її ваги зв'язків відомі, знаходяться у зовнішній пам'яті і в початковий момент часу завантажуються в ядра разом з виконуваним кодом.

При цьому ваги можуть зберігатися в оперативній пам'яті або на стеках. Загальний алгоритм обчислення вихідного значення нейрона зображено на додатку а.

У даному варіанті є схема, коли нейрон обчислюється на більшості ядер процесора, інші ядра можуть бути конвеєром, по якому рухається потік даних: вагові коефіцієнти, вхідні і вихідні вектори.

Перший можливий варіант це коли кожен вузол являє собою повноцінний нейрон з *i* входами. При розміщенні коефіцієнтів на стеку можливо реалізувати 6-7 зв'язків.

Кількість зв'язків при розміщенні ваг в оперативній пам'яті буде залежати від складності алгоритмів обчислення нейрона та функції активації та від розрядності ваг. У загальному випадку можна розраховувати в середньому на 8-15 зв'язків.

Обробку вхідних даних у вигляді вектору можна зробити двома способами: або записом усього вектору в оперативну пам’ять ядра або передачею послідовно через один з портів.

Розглядаючи варіант де увесь процесор завантажений виконанням розрахунків, він зможе максимум представляти мережу з 30-40 нейронів. Подібна реалізація дозволяє реалізувати багатошарову мережу, тобто в задані проміжки часу процесор являє собою певний шар нейронної мережі.

Потрібно реалізувати унікальні шляхи передачі даних для багатошарової мережі, бо слова, що їх реалізують, відповідним чином модифікуються.

Зв’язки між шарами можуть бути унікальними, так само як і сам потік даних. При цьому можливо конфігурувати вільні ядра разом з ядрами, які вже відпрацювали шар, для вирішення інших задач.

Все це входить в обов’язки розробника. Недоліком даної схеми є великі тимчасові затримки при обчисленні вихідного результату шару і витрати пам'яті на зберігання коефіцієнтів та вхідного вектора.

У другому варіанті було розглянуто ситуацію, коли ядро обчислює певний шар ШНМ. Цей варіант добре підходить, в тих випадках, коли створюється достатньо велика мережа рахуючи у кількості шарів та кількості нейронів в шарі.

У третьому варіанті є схема, коли нейрон обчислюється на більшості ядер процесора, таким чином ці ядра можуть бути конвеєром, по якому рухається потік даних: вагові коефіцієнти, вхідні і вихідні вектори.

В даному випадку частина ядер відповідає за синаптичні зв'язки (в залежності від числа синапсів нейрона таких зв'язків може бути до 15), частина ядер сумують і обраховують результат, і є ядра, які обчислюють передавальну

функцію нейрона. Переваги даної схеми однозначні це постійне завантаження процесорних ядер, що забезпечує розумне використання обчислювального ресурсу.

Це один з варіантів який нам підходить так як майже всі ядра виконують якісь обрахунки і не простоюють. Це в свою чергу дає можливість максимально розпаралелити роботу і покращити продуктивність.

В нашій роботі ми використовуємо підхід коли потік даних рухається по конвеєру в якому відбуваються різні операції схрещування, мутації та селекції популяції генетичним алгоритмом.

Коли кожне ядро являє собою унікальний нейрон і нам дано персептрона з чотирма входами, то час обчислення складає близько 800 нс, що є гарним показником, в порівняння з більшістю нейрочипів.

Наступний крок це синтез нейронної мережі.

Як раніше вже згадувалося, з точки зору ефективного використання обчислювальних ресурсів бажано, щоб всі ядра процесора брали участь в обчислення штучної нейронної мережі.

На рисунку 3.3 зображено загальний процес обчислення нейронної мережі на архітектурі процесора SEAForth S40C18.

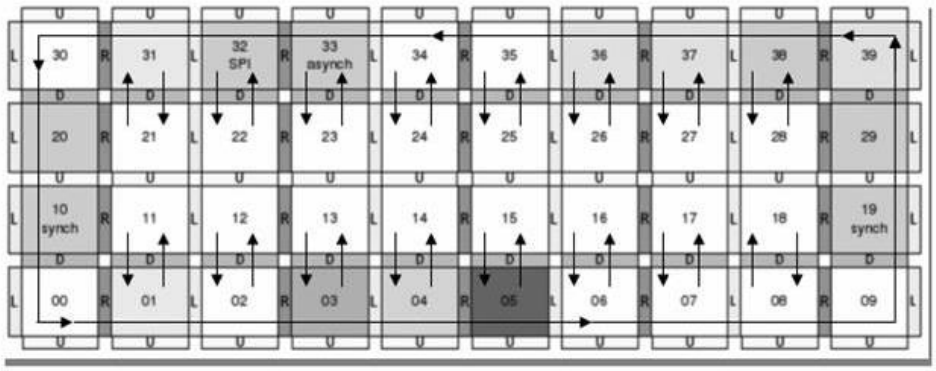

Рисунок 3.3 – Процес обчислення нейронної мережі на процесорі[24]

Нейрони кожного шару нейронної мережі представлені ядрами №11-18 та №21-28. Решта ядер беруть участь в передачі виходів від попереднього шару наступному, представляючи собою конвеєр, стрілками показаний процес руху вихідного потоку попереднього шару нейронів до наступного шару. Перевагою

даного методу є можливість реалізації програмістом будь-якої архітектури нейронної мережі.

Загальний алгоритм обчислення нейронної мережі на даному процесорі:

а) вхідний вектор $\bar{x}$ розмірністю n – надходить в ядро №10 через послідовний синхронний порт, а потім в ядро №00;

б) вектор $\bar{x}$ з ядра №00 побітно надходить на «конвеєр», за яким дані надходять на нейрони і-го шару мережі (ядра №11-18, №21-28);

в) після того як вихідний вектор $\bar{y}$ отриманий, він надходить на «конвеєр», за яким потім формується в ядрі №00;

г) згідно архітектури мережі через інтерфейс SPI можливе завантаження в ядро як коду програми. Так і необхідних вагових коефіцієнтів синапсів наступного шару мережі;

д) вектор, отриманий на попередній ітерації, з ядра №00 надходить на «конвеєр» і далі на нейрони наступного шару;

е) після того як вихідний вектор мережі отримано, він надходить до ядра №10 і далі до інших елементів через послідовний синхронний порт.

Як результат, коли ми змінюємо код того чи іншого стану автомата, програміст отримує можливість реалізовувати найрізноманітніші зв'язки між нейронами штучної нейронної мережі.

Що в свою чергу дає повну свободу кастомізації з програмування схеми і таким чином більше способів з оптимізації та покращення швидкодії роботи.

3.3 Порівняння ПЛІС з іншими платформами

Так як в даній роботі велися досліди швидкодії та робилися заміри швидкості навчання, варто порівняти ПЛІС з іншими платформами.

Альтернативні платформи для ПЛІС є:

- інтегральна схема для специфічного застосування(ASIC);
- графічний процесор(GPU).

Одним з мінусів при використанні ASIC це те що, вони виготовляються тільки під замовлення і для вирішення задач вузького спектру, в той час як для ПЛІС ми просто пишемо необхідне програмне забезпечення.

ПЛІС можна перепрограмовувати на відміну від ASIC, що робить ПЛІС кращим рішенням в плані гнучкості використання. Серед плюсів ASIC швидкість виконання поставленої задачі та дешевша ціна однієї операції.

Методи та алгоритми апаратно-програмної реалізації компонентів нейронних систем управління повинні бути орієнтовані на реалізацію в ПЛІС, вони повинні бути адаптовані до їх елементного базису. ПЛІС різних фірм – виробників мають в своєму складі логічні таблиці, тригери, елементи суматорів, блоки оперативного запам’ятовуючого пристрою та постійного запам’ятовуючого пристрою.

З використанням GPU, на відміну від ПЛІС, не можна апаратно реалізувати алгоритми, вони в свою чергу займаються лише виконанням програми.

Таким чином, графічному процесору потрібно діставати програмні інструкції з пам’яті, виконати їх та пересилати назад в пам’ять. Це все робить графічний процесор набагато повільнішим ніж ПЛІС.

Всі інші пристрої реалізуються на їх основі. Для конкретизації методів, алгоритмів та критеріїв їх оптимізації була вибрана архітектура ПЛІС фірми Xilinx. Параметри критеріїв, такі як апаратні затрати та затрати часу, можуть бути перераховані для інших виробників.

Значний виграш у швидкості ПЛІС демонструє через велику кількість однотипних блоків які можуть виконуватися паралельно та надавати результати в десятки разів швидше аніж це робить графічний процесор. ПЛІС не витрачає часу для пересилання даних через загальну шину, а це ще один великий плюс.

Також є велика перевага ПЛІС чрез те що така конструкція не дорога так як не потребує для своєї роботи процесора на відміну від GPU, що може коштувати в десятки разів дорожче.

Єдиною перевагою графічного процесора є його робота з числами з плаваючою комою. Він набагато краще підходить для обробки таких сигналів та

для задач роботи з графікою, де ПЛІС сильно програє через свою неспроможність ефективно обробляти такі дані.

Переваги ПЛІС:

- не існує витрат часу на пересилання даних;
- менше споживання енергії;
- менша ціна.

Проаналізувавши низку платформ можна зробити певні висновки. В цьому розділі було розглянуто: задачі, для яких використовувалися апаратні платформи та мікросхеми, їхня ефективність та ціна виконання операції, що в свою чергу впливає на швидкодію та вирішення конкретного роду задач.

Проаналізовано низку апаратних та програмних платформ для реалізації ШНМ, розглянуто приклад реалізації ШНМ на багатоядерних процесорах.

Отже, на швидкодію навчання нейронів, швидкість навчання ШНМ, точність вихідних результатів, затрачений ресурс чіпів мають вплив такі речі як:

- пристрій за допомогою якого вирішується задача;
- обсяг нейронної мережі та їхні зв’язки;
- шари що приймають участь у навчанні мережі та зв’язки між ними;
- оптимізація компонентів схеми та їхній взаємозв’язок;
- алгоритм що використовується для навчання та тестування мережі.

4 МЕТОД АПАРАТНОЇ РЕАЛІЗАЦІЇ ГЕНЕТИЧНОГО АЛГОРИТМУ

4.1 Апаратна частина для реалізації генетичного алгоритму

Мова опису апаратури сьогодні відіграє ключову роль в світі ПЛІС. За допомогою неї відбувається автоматична трансляція опису алгоритму в опис інструкцій зрозумілий машині, та інших низькорівневих елементів ПЛІС.

Сьогодні лідери цієї глазу це дві мови VHDL та Verilog, вони використовуються для розробки та проектування різного роду логічних схем.

З їхньою допомогою зручно описувати інструкції щодо програмування ПЛІС, які в свою чергу перетворюються на більш низькорівневі команди за допомогою компіляторів цих мов.

Елементарні функції в ПЛІС реалізовані як окремі пакети чи замовлені віртуальні модулі, в яких вказується внутрішня будова та розрядність даних.

Виконання програм на ПЛІС не проста задача, так як потрібно або підібрати або адаптувати потрібний алгоритм. Кожен елемент може бути описаний та задокументований на мові програмування яка дає можливості інтеграції програмних та апаратних засобів.

ПЛІС різних виробників мають в своєму складі велику кількість різних компонентів таких як: логічні таблиці, тригери, елементи суматорів, блоки оперативного запам’ятовуючого пристрою та постійного запам’ятовуючого пристрою. Ці елементи є основою для реалізації всіх інших пристроїв.

В даному випадку ми обрали декілька різних архітектур різних виробників ПЛІС, задля порівняння їх між собою. Була створена таблиця в якій детально описані результати виконання алгоритму та критерії по їх оптимізації. Апаратні витрати обчислювального ресурсу та витрати часу, були перераховані і для інших виробників.

Сучасна розробока ПЛІС високої інтеграції «нейрочіп-2000» (рисунок 4.1), що виконаний на базі Virtex / Virtex-E.

Перспективна розробка «нейрочіп-8» це інструментальна плата KMFC-787, що базується на сімействі ПЛІС Spartan від компанії Xilinx з використанням нейроінтерфейсної структури.

Виробники які займаються створенням ПЛІС випускають також цілий набір інструментальних плат на базі ПЛІС різних серій та модулів різного призначення, що дозволяють ефективно і швидко створювати обчислювальні системи різного функціонального призначення.

Всі вищезгадані розробки треновані на мереж з передатною функцією сигмоїда, що в свою чергу дозволяє їх використовувати для різного числа систем як загального так і вузького напрямку.

Основні вимоги які ставляться до сучасних нейроконтролерів це розробка можливостей за допомогою яких вони здатні будуть функціонувати і адаптуватися в режимі реального часу.

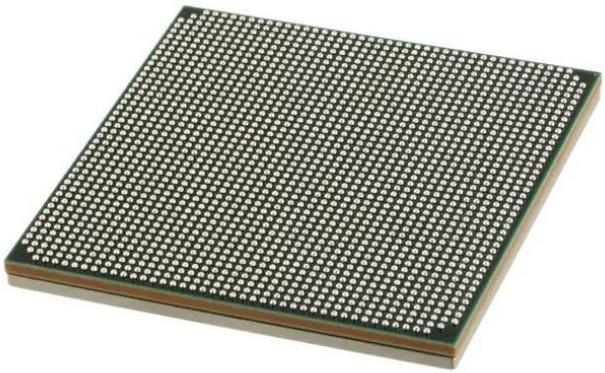

Рисунок 4.1 – Інтегральна схема QW8Z3000K на базі Artix-12[7]

Основна проблема яка виникає при реалізації апаратного блоку є процес інтеграції та адаптації самого алгоритму. Ця проблема виникає через певну кількість інших обмежень, що накладає на себе ПЛІС та схожі апаратні засоби.

Після створення обгорток для оминання проблем з обчистленнями такими як ділення чисел та представлення чисел з плаваючою комою, можна було реалізувати в повній мірі ключові аспекти генетичного алгоритму.

4.2 Навчання нейронних мереж за допомогою генетичного алгоритму

На рисунку 4.2 зображено загальну схему навчання нейронних мереж, у відповідністю з якою слід вести навчання:

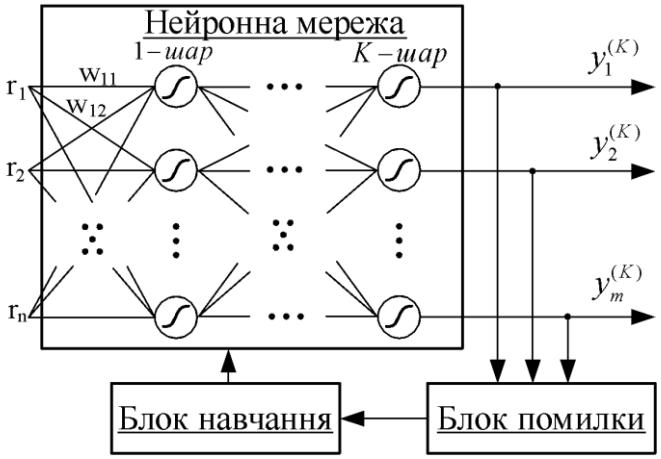

Рисунок 4.2 – Апаратний блок по навчанню ШНМ[11]

«Блок помилки» порівнює значення виходів мережі, отриманих після подачі на вхід заданих сигналів, і виходів, які повинні вийти при подачі на вхід еталонних сигналів. Значення помилки визначається по формулі.

$$Q = Y_i^{target} - Y_i^{output} \quad (4.1)$$

Одним з найвідоміших на даний момент представником еволюційних алгоритмів є генетичний алгоритм і за своєю суттю є набором інструкцій для знаходження глобального екстремуму багатоекстремальної функції.

Він полягає в паралельній обробці безлічі альтернативних рішень. При цьому пошук концентрується на найбільш перспективних з них. Це говорить про можливість використання генетичних алгоритмів при вирішенні будь-яких завдань штучного інтелекту, оптимізації та прийняття рішень.

Адаптована до апаратних компонентів оптимізація вагових коефіцієнтів відбувається наступним чином:

Крок 1. Ініціалізація початкової популяції хромосом.

Крок 2. Оцінка хромосом поточної популяції.

Крок 3. Перевірка критеріїв закінчення пошуку. У випадку, якщо критерії зупинки задоволено, виконати перехід до кроку 7.

Крок 4. Вибір особин для генерації нових рішень.

Крок 5. Застосування операторів схрещування та мутації для хромосом.

Крок 6. Формування нового покоління з елітних хромосом. Перехід до виконання кроку 2.

Крок 7. Зупинка.

На рисунку 4.3 показано яким чином кожна хромосома з популяції може бути представлена. Її вигляд нагадую вектора даних, кожен елемент вектора складається з певного об’єкту з даними.

Розмір хромосоми визначається за формулою:

$$K = N_1(L+1) + \sum_{\mu=1}^{M} N_\mu\,(N_{\mu-1}+1), \qquad (4.2)$$

де N_μ – кількість нейронів на μ-ому шарі; L – кількість характеристик у тестовій множині; M – кількість шарів нейромережі.

Рисунок 4.3 – Схематичне подання хромосоми при параметричному синтезі нейромоделей[12]

Всі потрібні операції для початку потрібно розпаралелити та пристосувати для FPGA це потрібно для успішної апаратної реалізації еволюційних методів навчання нейронних мереж.

При розробці паралельних обчислювальних систем необхідно проаналізувати характеристики алгоритму який буде спроектований.

Граф є основним елементо для обчислення вагів. Адаптація до створення апаратного блоку відбувається саме на етапі проектування графа ШНМ. Це в свою чергу дає повне розуміння роботи апаратної версії:

$$\mathrm{GDF} = (A,\ D), \tag{4.3}$$

де A – масив вершин, який складається з функцій; D – масив з'єднань, який складається з самих даних.

Необхідним критерієм для правильного функціонування апаратного блоку є правильний розподіл рандомних чисел та чіткий розподіл даних при ініціалізації хромосом.

При багатопроцесорній обробці більше ніж один процесор в системі робить обробку вступників команд, тим самим дозволяючи здійснювати повністю паралельну обробку.

Так як основна характеристик генерації випадкових чисел є генерація чисел які не можна було б якимось алгоритмом віднайти, вони мали би бути дійсно дуже схожі на випадкові, а не штучні. То було обрано конгурентний метод створення таких чисел:

$$r_{i+1} = \mathrm{mod}(k * r_i,\ M), \tag{4.4}$$

де M – модуль ($0 < M$); k – коефіцієнт ($0 \leq k < M$).

Щоб стоврити генератор потрібен алгоритм по підбору коефіцієнтів. Необхідно щоб число M, було достатньо велике, бо інакше почнуть виникати конфлікти які не дадуть справно працювати алгоритму.

Одна з найповільніших операцій для виконання алгоритму в нашому випадку є операція ділення так як вона необхідна в ряді інших функцій алгоритму, причому що для ПЛІС ця операція не є стандартною і потребує реалізації з боку програміста, тому логічно обрати $M = 2^N$, в такому випадку ми маємо замінити

операцію ділення на операцію зсуву що дасть нам певний виграш у швидкості та вирішить обмеження що накладає на нас ПЛІС.

Нам потрібно згенерувати певний масив чисел від 0 до 7 для того щоб наші хромосоми коректно відпрацювали.

Такий масив можна отримати якщо використати цю формулу:

$$r_{i+1} = \mathrm{mod}(r_i,\ 8), \tag{4.5}$$

Таким чином, щоб алгоритм запрацював нам потрібно взяти саме три останні розряди згенерованого рандомного числа та використати їх в подальшій роботі.

Існує теорема, про те, що, при застосуванні конгруентного метода для генерації випадкових чисел, молодші розряди випадкового числа поводять себе так само випадково, як і старші.

При реалізації апаратниого блоку на ПЛІС, кожен шар мережі працює в паралельному процесі, тут використовується принцип конвеєра. Нейрони в кожному шарі також працюють паралельно за принципом багатопроцесорної обробки даних. Тобто кожний штучний нейрон мережі є окремим процесором і обробка інформації в кожному нейроні проходить одночасно.

В даному випадку ми використовуємо число 8, так як воно зумовлено тим що, це кількість бітів у тестовій хромосомі. Якщо змінити кількість біт, то виникне потреба для генерації випадкових чисел у іншому діапазоні.

Розгортання внутрішніх циклів є відомим методом, який у ряді випадків може привести до значного зростання продуктивності. Суть методу полягає в поданні циклічних алгоритмів обробки з кінцевим числом ітерацій у вигляді довгої комбінаційної ланцюжка, що обробляється за один такт.

На рисунку 4.4 добре видно основні операції які виконує апаратна реалізація генетичного алгоритму. Основними в даному випадку є операції відбору, схрещування, мутації та оцінки результатів.

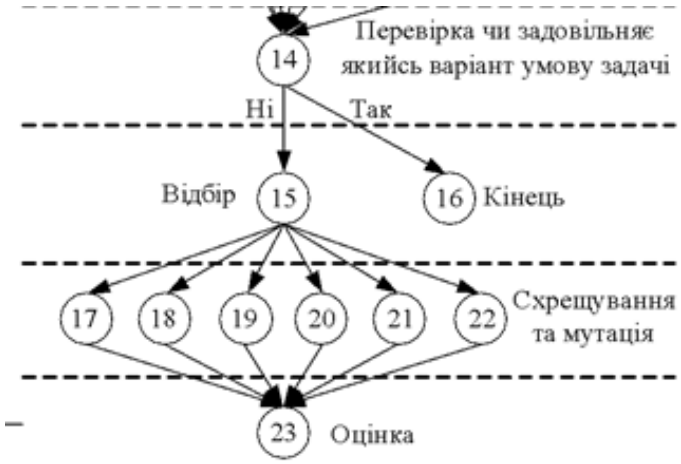

Рисунок 4.4 – Фрагмент графу, що демонструє роботу апаратної версії генетичного алгоритму

Проаналізувавши методи апаратної частини для генерації генетичного алгоритму можна зробити певні результати.

Отже, в цьому розділі було розглянуто структуру апаратної частини ПЛІС. Було досліджено схеми різних виробників. Велика кількість елементів забезпечує роботу плати та робить її настільки швидкою. Окремої уваги заслуговують пристрої таблиць та різного роду трігерів які складають основу схеми.

Програмувати ПЛІС можна за допомогою спеціальних мов опису апаратури. На даний момент є дві основні мови це VHDL та Verilog. У своїй роботі для програмування ПЛІС я використовував мову опису VHDL, вона має високорівневий синтаксис та зручна у використанні.

В науковій роботі було реалізовано еволюційну оптимізацію вагових коефіцієнтів що в свою чергу допомагає використати функцію розпаралелення обчислень на ПЛІС якнайкраще.

Певну роль грає архітектура ПЛІС та нейромережі, адже це може як допомогти пришвидшити навчання так і зменшити її в більш довготривалій перспективі. Тому потрібно звертати увагу на елементи та алгоритм який планується реалізовувати на даній схемі.

5 СПОСІБ АПАРАТНОЇ РЕАЛІЗАЦІЇ ГЕНЕТИЧНОГО АЛГОРИТМУ

5.1 Операції та основні вирази генетичного алгоритму

Генетичний алгоритм складається з базових операцій: операція відбору особин після оцінки, операція схрещування особин одна з одною, причому вибір особин відбувається випадковим чином та операція мутації особини, при якій мутується випадковий ген випадкової особини.

Для початку відбувається відбір з певної структури. Відібрані особини допускаються до наступних етапів таких як схрещування між собою та мутація. Ці операції створюють нове покоління з якого по колу відбираються найсильніші особини.

Особи наступного покоління оцінюються, потім відбираються, застосовуються генетичні оператори і т.д. Таким чином відбувається симуляція процесу що емулює життєвий процес та продовжується доти доки не буде отримано результат або не буде виконана умова виходу з алгоритму.

Можна встановити наступні умови припинення роботи алгоритму:

- якщо ми отримуємо оптимальне рішення та задоволені його точністю і не бажаємо чекати довше;
- якщо ми задаємо певну кількість поколінь яку може виконуватися алгоритм, після виконання яких він просто припиняє роботу та вертає останній результат;
- якщо ми заздалегідь задаємо час за який алгоритм має відшукати відповідь, якщо відведеного часу не достатньо алгоритм також припиняє роботу та повертає останній результат.

На першому кроці виконання алгоритму потрібно випадковим чином створити якусь початкову популяцію; навіть якщо вона виявиться зовсім неконкурентоспроможною, генетичний алгоритм всеодно досить швидко переведе її в життєздатну популяцію.

Таким чином, можна особливо не намагатися зробити пристосованих індивідуумів, достатньо, щоб вони мали однакові характеристики притаманні даній множині рішень і на них відбувається розрахунок передатної функції.

На етапі відбору алгоритм з усієї популяції вибрає певну частку, що залишиться «у живих» на цьому етапі еволюції.

Існують різні способи проводити відбір. Ймовірність виживання особи повинна залежати від значення функції пристосованості. Множину об'єктів що вижили задають раніше і ця частина відправляється до генетичного алгоритму як вхідний параметр.

Останнім кроком виконання алгоритму є операція по відбору найсильніших рішень та формування на їхній основі окремої популяції. Після відбору одна множина з найменш підходящими рішеннями помирає, а найбільш сильніші відправляються на новий етап роботи алгоритму.

Розмноження в генетичному алгоритмі зазвичай полове – щоб зробити нащадка, потрібно декілька батьків. Розмноження в різних алгоритмах визначається по-різному, зазвичай, залежить від представлення даних. Головна вимога до розмноження – щоб нащадок мав можливість успадкувати риси обох батьків, «змішавши» їх яким-небудь досить розумним способом.

Для того щоб провести операцію розмноження, потрібно вибрати $(1-S)p/2$ пар гіпотез з H і провести з ними розмноження, одержавши по два нащадка від кожної пари (якщо розмноження визначене так, щоб давати одного нащадка, потрібно вибрати $(1 - S)p$ пар), і додати цих нащадків у H'. У результаті H' буде складатися з N хромосом.

З усієї популяції H, вибираються особи для розмноження. Це робиться для того, щоб збільшити розмаїтність в особах так як це є головним недоліком останнього підходу. Якщо буде створене таке рішення яке дасть нам лише частковий результат, то існує вірогідність що дані рішення породить багато схожих до себе і таким чином заб'є усю популяцію такими частковими рішеннями.

Один із варіантів боротьби з таким негативним ефектом це – вибір для розмноження не найпристосованіших, але взагалі всіх осіб.

5.2 Апаратна реалізація

Для апаратної реалізації, потрібно виділити наступні кроки роботи алгоритму:

Крок 1. Ініціалізація популяції хромосом, вони повинні містити інформацію про значення вагових коефіцієнтів мережі заданої структури.

В даному випадку, інформація представлена у вигляді двовимірного масиву на рисунку 5.1.

Кожна хромосома відповідає рядоку в масиві і зберігає інформацію про весь набір вагових коефіцієнтів мережі. Таким чином, в залежності від кількості вхідних зв'язків, кожному нейрону може відповідати різна кількість вагових коефіцієнтів.

Кожен коефіцієнт знаходиться в окремому елементі масиву, він являє собою дійсне число.

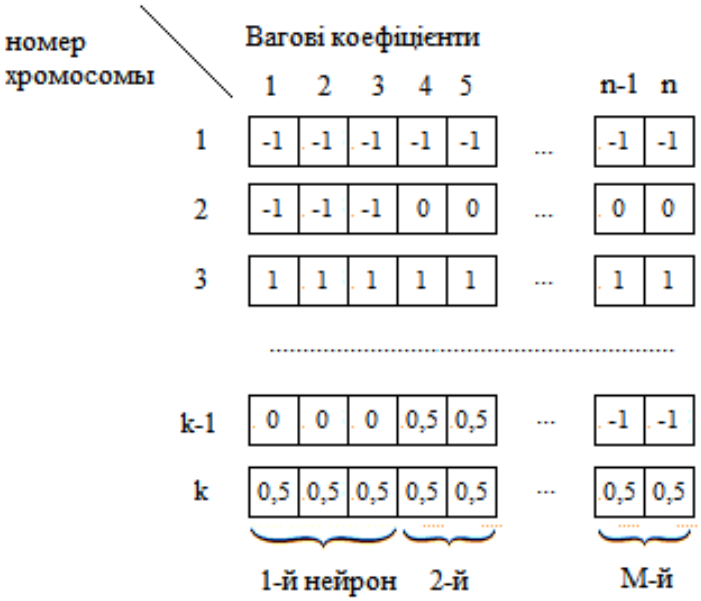

Рисунок 5.1 – Представлення інформації про значення вагових коефіцієнтів мережі[29]

У мові VHDL не можна стандартними засобами реалізувати операції з дійсними числами. Тому, дійсні числа було представлено як дріб:

$$\frac{a}{b}, b = 2^c \tag{5.1}$$

Використовуючи стандартні засоби мови VHDL ми можемо реалізувати операцію ділення на число 2^c, це дозволить реалізувати нам операцію зсуву вліво на c біт і таким чином позбавить від обмежень які накладає на нас мова опису апаратури.

Приймемо змінну b як 16, а кожен ваговий коефіцієнт будемо представляти як восьми бітне слово, таким чином число -1 дорівнює набору бітів «11110000», а число 1 – «00010000» і так далі.

Так як 11110000b = -16d, 00010000b = 16d, 00001000b = 8d.

Потрібно враховувати той факт, що b=16, -16/b=-16/16=-1, 16/b=16/16=1, 8/b=8/16=0.5.

Відповідно, ми можемо представити хромосоми початкової популяції у вигляді тривимірного масиву (рисунок 5.2):

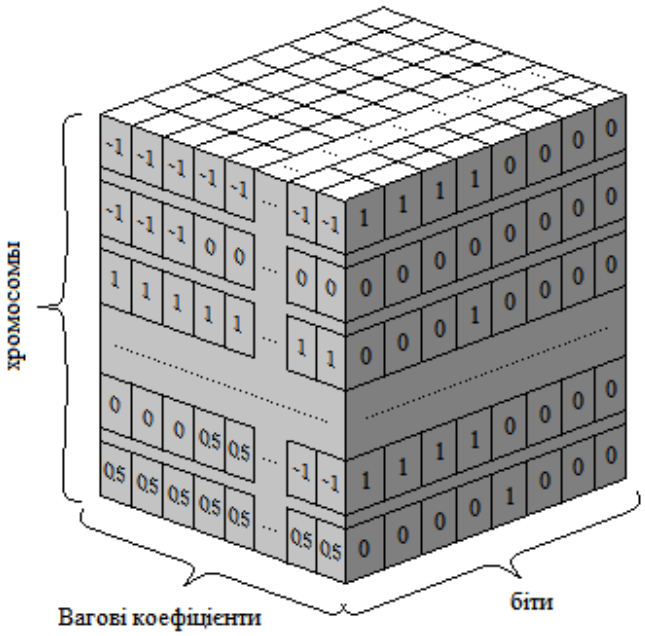

Рисунок 5.2 – Представлення вагових коефіцієнтів у вигляді тривимірного масиву[29]

Крок 2. Оцінка хромосом поточної популяції.

Крок 2.1. Декодування кожної хромосоми в множину параметрів нейронної мережі. Для операції по декодуванню використовуються стандартні методи мови VHDL: To_integer і Signed, зображені на рисунку 5.3, таким чином, що:

$$W := \text{To\_integer}(\text{Signed}(\text{chrs}(2)(1))); \quad (5.2)$$

де chrs(2)(1) – дргуий ваговий коефіцієнт першої хромосоми, множина бітів; W – ціле число.

Для правильного визначення знаку числа використовується метод Signed. З його допомогою ми можемо розуміти додатнє ми отримали число чи від'ємне. Якщо поглянути глибше то буде видно що ця функція не перетворює біти числа, а лише визначає що подано даними бітами. Вона розуміє що для від'ємних чисел знак визначається за допомогою старшого біта, а для додатніх чисел використовується прямий код що дозволяє з легкістю розрізняти ці числа.

Щоб перетворити поданий масив бітів як хромосома у ціле число для порівняння результатів використовується функція To_integer. Вона включає в собі алгоритм описаний і для функції signed.

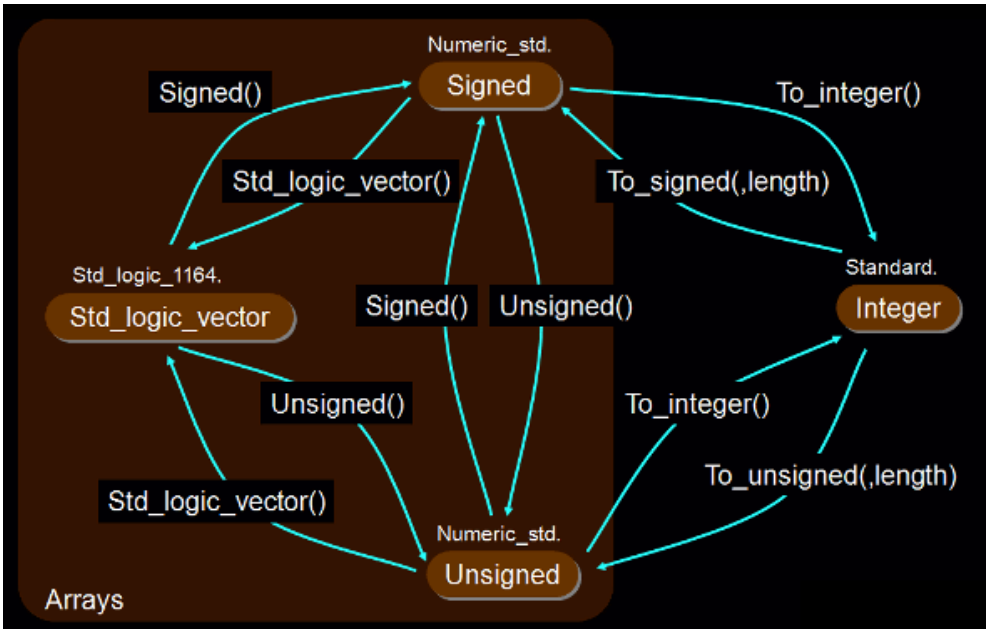

Рисунок 5.3 – Стандартні функції перетворення типів мови VHDL

Крок 2.2. Знайти вихід нейромережі для розрахованих вагових коефіцієнтів.

Нейрон першого шару мережі запускає сигнал start, що сповіщає про завершення перетворення вагових коефіцієнтів хромосоми. Зміна сигналів на виході останнього шару мережі запускає процес обчислення функції активації.

Крок 2.3. Обчислити значення функції активації хромосом що оцінюються, та враховувати помилку й складність мережі.

Функція активації визначає наскільки отриманий вихід мережі відрізняється від необхідного, та вираховує похибку як різницю очікуваного та отриманого.

Крок 3. Відбувається перевірка умови закінчення роботи алгоритму. Тут вирішується питання того чи було отримано бажаний результат. Якщо критерії виходу відповідають дійсності то алгоритм припиняє свою роботу. Якщо ж все ж таки жоден з критеріїв виходу не дав позитивний результат то алгоритм продовжує свою роботу доти доки хоча б один з критеріїв виходу не спрацює.

Крок 4. Обрати особини для генерації нових рішень в залежності від значення функції активації.

Крок 5. Примінити операції по схрещуванню та мутації для індивідуумів, що були отримані на попередньому кроці.

Так як усі вхідні дані розглядаються як масив бітів. То, перша сума визначатиме першу особину для схрещення (її номер *i*), друга сума визначає другу (номер *j*):

$$H' = \sum_{i=1}^{n_1} \sum_{j=i+1}^{n_2} \left(\sum_{k=1}^{b-1} H_{ik} 2^k + H_{jb} 2^b + \sum_{k=b+1}^{B} H_{ik} 2^k\right), \tag{5.3}$$

де H' – новий набір хромосом; H – попередній; b і k – номери бітів; 2^b, 2^k – позначають позицію біта у двійковому числі; H_{ik} – k-й біт i-ї хромосоми, H_{jb} – біт з номером b у j-ї хромосомі.

Як результат по формулі відбувається копіювання генів однієї хромосоми до результуючої. Копіювання відбувається не повним так як виключається один біт

який дозволяє створити нову хромосому яка не буде схожою на будь яку іншу. Це дозволяє нас уникнути проблем з конфліктами рішень в подальшому так як може статися так що після схрещення буде створене одне рішення яке своїми копіями заб’є всю популяцію.

Невеликий відривок коду основних операцій:

```
for first in 1 to 5 loop
  for second in first+1 to 6 loop
    chrNumber:=chrNumber+1;
    for data in 1 to countBits-1 loop
      newPopulation(chrNumber)(data):=prevPopulation(first)(data);
    endloop;
    newPopulation (chrNumber)(countBits):= prevPopulation (second)(countBits);
    for third in countBits +1 to 24 loop
      chrs(chrNumber)(third):=chrs2(first)(third);
    endloop;
  endloop;
endloop;
```

Записаний сніпет коду працює, але має ряд недоліків:

- наслідування лише одного біту робить алгоритм недієздатним для більшої кількості даних, а щоб виправити це потрібно докласти значних зусиль;
- спрощене уявлення структур даних якими оперує алгоритм, для вирішення простих задач його буде достатньо проте для вирішення більш серйозних задач він не буде давати правильних відповідей так як структура яка використовується занадто спрощена.

```
for first in 1 to 5 loop
  for second in first+1 to 6 loop
    chrNumber:= chrNumber +1;
    newPopulation (chrNumber):= prevPopulation (first);
    for third in 1 to 3 loop
      for chrInheritedBit in 1 to chrInheritedBits/2
```

randomInheritedBit = random(chrInheritedBits);

newPopulation (chrNumber)(third)(randomInheritedBit):= prevPopulation (second)(third)(randomInheritedBit);

endloop;

endloop;

endloop;

endloop;

Враховуючи складність структури даних і те що хромосома містить в собі ще й ваги отримуємо наступну формулу:

$$H' = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{m=1}^{m_{length}} \left(\sum_{k=1}^{b-1} H_{ik} 2^k + H_{jb} 2^b + \sum_{k=b+1}^{B} H_{ik} 2^k \right), \quad (5.4)$$

де m – номер вагового коефіцієнту.

Крок 6. Формування нового покоління з відібраних об’єктів та результуючих об’єктів, що були отримані через процеи схрещування й мутації. Перехід до виконання кроку 2.

Крок 7. Зупинка.

В нашому випадку хромосома представляє собою рішення задачі. Вона складається з масиву бітів що називаються генами та в свою чергу містять більш простіші об’єкти це самі дані та значення вагових коефіцієнтів. Для кодування та декодування використовуються окремі функції це допомогає нам порівнювати результат навчання мережі.

Після дослідження програмної частини генетичного алгоритму можна зробити деякі висновки.

Таким чином, щоб рішення можна було використовувати в самому апаратному блоці його потрібно певним чином закодувати. Це рішення має відображати результат задачі яку ми намагаємося вирішити.

Після створення першої популяції ми маємо оцінити всіх її особин для подальших дій. Для цього існує певна функція яка проводить оцінку рішень. Після

оцінки рішень популяція ділиться на тих хто пройшов оцінку і переходить на новий етап та на тих хто її не пройшов тому вимушений померти. Це допомагає нам відсіяти слабкі та недоречні рішення задачі а також зробити більш сильну наступну популяцію яка наблизить нас до правильної відповіді.

Було описано яким чином буде функціонувати апаратний блок та створено першу версію програмного коду який мав певний ряд недоліків:

- наслідування лише одного біту робить алгоритм недієздатним для більшої кількості даних, а щоб виправити це потрібно докласти значних зусиль;
- спрощене уявлення структур даних якими оперує алгоритм, для вирішення простих задач його буде достатньо проте для вирішення більш серйозних задач він не буде давати правильних відповідей так як структура яка використовується занадто спрощена.

Далі було проаналізовано ці недоліки та проведені дослідження по їхній ліквідації. Після чого було приведена друга версія коду яка вирішувала вище згадані проблеми.

Було створено програму на мові VHDL яка реалізує принцип паралельного програмування та дозволяє створити процеси як окремі функцію для обчислень та навчання мережі в короткі терміни.

Створений на основі математичної моделі код продемонстрував здатність апаратного блоку проводити базові операції генетичного алгоритму і таким чином навчати мережу хромосом. Це все дає нам змогу досліджувати дану систему далі.

6 ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ АПАРАТНОЇ РЕАЛІЗАЦІЇ ГЕНЕТИЧНОГО АЛГОРИТМУ ОДНОГО НЕЙРОНУ

6.1 Блок штучного нейрону на ПЛІС

Для того щоб реалізувати апаратного блоку для навчання нейрону потрібно створити декілька його основних частин:

- блок штучного нейрону;
- блок генетичного алгоритму.

Для початку опишемо пристрій штучного нейрону, далі розглянемо реалізацію інтерфейсу, який проводить навчання цього нейрону.

Розроблений штучний нейрон з 4-ма входами та функцією сигмоїдою активації на ПЛІС з використанням 32-розрядних чисел зайняв 10627 LUTs (Look Up Table - вентилі логічної матриці). Блок штучного нейрону розроблений на ПЛІС зображений на рисунку 6.1.

Швидкодія, як сумарна затримка схеми блоку нейрона, склала 67.7 нс. При зміні параметрів таких як: кількість лінійних відрізків, кількість входів нейрона або зміну розрядності чисел потрібно редагувати ресурси що використовуються для ПЛІС апаратного блоку.

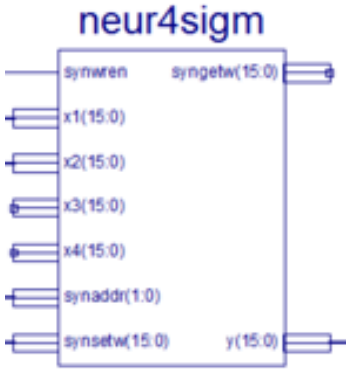

Рисунок 6.1 – Блок штучного нейрону на ПЛІС

Всередині кожного блоку нейрона є блок оперативної пам'яті де зберігаються ваги. Для того щоб присвоювати значення вагам у кожному нейроні використовуються такі сигнали:

synapseWeightAddress – змінна яка відповідає за адресу синапсу вага якого буде прочитана або записана за цією адресою.

synapseSetWeight – функція яка виконує запис ваги по даному синапсу.

enableSynapseWriting – змінна яка при отриманому значенні 1 записує в даний синапс значення з synapseSetWeight, якщо значення змінної дорівнює 0, то нічого не відбувається.

Базовий елемент який займається сумуванням добутку вхідного вектора даних та вагових коефіцієнтів називається суматором передатної функції. За допомогою нього ми можемо формувати необхідні дані для подальших шарів мережі:

$$x = \sum_{i=1}^{N} w_i a_i, \tag{6.1}$$

де a – вхідні значення нейрона; w – вагові коефіцієнти.

Обчислення відбуваються за допомого стандартних бібліотек мови, всі числа якими оперує алгоритм мають обмеження в 16 біт та мають фіксовану точку, що дозволяє отримувати точні розрахунки та не торбуватися про проблеми переповнення чи неправильних арифметичних операцій з плаваючою точкою.

Для всіх типів обчислень використовується пакет з фіксованими числами, що дуже нагадує за своєю специфікою мову програмування С. Це робить розрахунки надійними та точними, а в даній роботі це одне з найважливіших характеристик.

Варто сказати про те що всі методи та операції даного пакету дуже добре оптимізовані та дозволяють їхнє виконання в паралельному режимі, що робить їх ще кращим інструментом для розробки.

На рисунку 6.2 зображено розбиття сигмоїдальної функції на маленькі шматочки для подальшої апроксимації та уточнення результатів. Для цього ми використовуємо похідну першого ступеня і виконуємо розбивку для кожної частини лінійного рівняння. Таким чином ми визначили усі потрібні коефіцієнти.

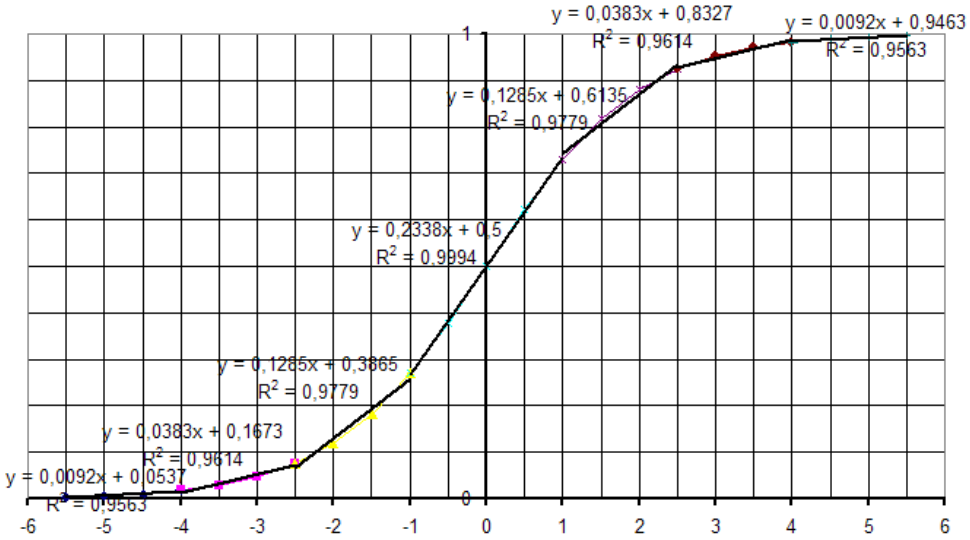

Рисунок 6.2 – Кусково-лінійна апроксимація сигмоїдальної функції

Визначаємо коефіцієнти лінійних рівнянь:

$$k, b = [0.234, 0.500], \text{якщо } 0 \leq x' < 1 \quad (6.2)$$
$$k, b = [0.129, 0.605], \text{якщо } 0 \leq x' < 2.5$$

Визначаємо змінну *f*, обчислюємо значення за формулою:

$$f = k * x' + b \quad (6.3)$$

Розрахунки локальних мінімумів відбувається якщо змінна менша нуля:

$$f = 1 - f \quad (6.4)$$

Далі відбувається присвоєння значення результату роботи функції та передача цього значення далі у вигляді сигналу іншим шарам нейронів. Це робить використання такого методу багаторазовим, що свідчить про його значимість та використання його багатьма частинами програми. Звісно лише такий маленький прийом може зробити підтримку рішення не тільки зручнішою, але й пришвидшити його виконання за рахунок внутрішніх оптимізацій.

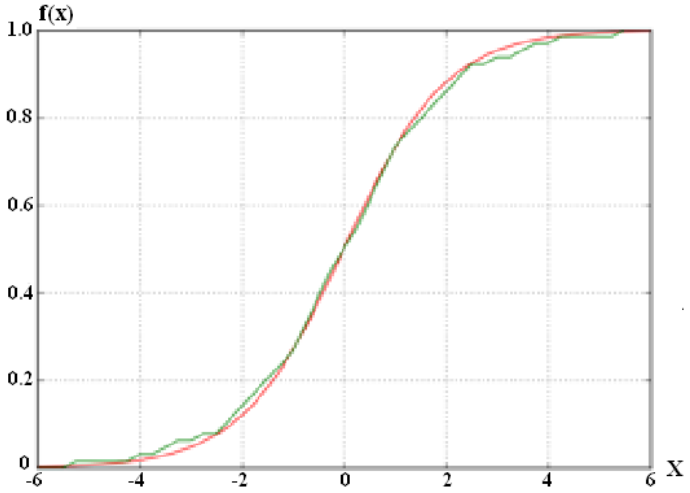

Рисунок 6.3 – Сигмоїдальна функція, реалізована на ПЛІС

На рисунку 6.3 показано стандартну сигмоїдальну функцію та функцію реалізовану на ПЛІС. Даний обчислювальний блок, що реалізує сигмоїдальну функцію в ПЛІС точно відображає її значення.

6.2 Розробка блоку генетичного алгоритму на ПЛІС

Для розробки блоку генетичного алгоритму, рисунок 6.4, потрібно було розробити його структуру. Він складається з вхідних портів які приймають сигнали – in1, in2, in3, також було відведено місце під порт для збереження результуючого значення – expectedOutput, та змінної результуючого виходу, що показує поточне значення виходу після кожної ітерації.

FinishTeaching – сигнал, що відповідає за закінчення навчання мережі.

Start – сигнал, що відповідає за процес формування вихідних імпульсів. Кожного разу змінюючи значення цієї змінної змінюються ваги та перезаписуються інші сигнали в паралельних процесах для ініціалізації інших функцій.

chrIndex – індекс хромосоми, на основі якої формуються вагові коефіцієнти перед запуском процеса з перетворення виходу нейронної мережі(перед установкою сигнала start в 1).

NoutsForming – сигнал, який приймає значення 1 поки відбувається навчання хромосом нейронної мережі. Зміна значення відбувається коли всі дані було переглянуто при переході з 1 в 0. Далі відбувається аналіз результатів: чи завершення алгоритму чи створення нової популяції. Нова ітерація починається коли значення змінної міняється з 0 на 1.

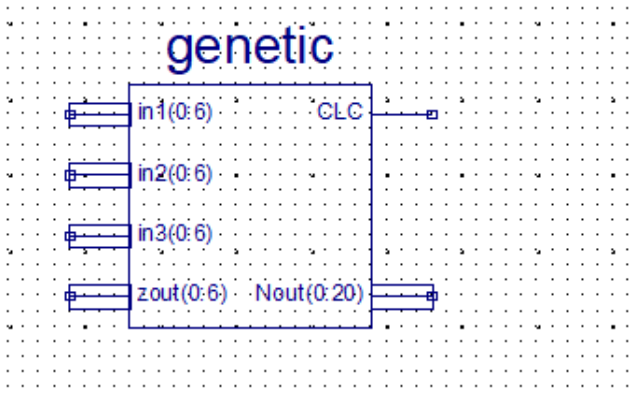

Рисунок 6.4 – Блок генетичного алгоритму на ПЛІС

Два різних процеса керують основними функціями про аналізування частини даних та перехід до оцінки цих даних, щоб дати адекватну відповідь і почати новий цикл або ж завершити виконання.

Зміна значення одного сигналу в двох різних процесах не передбачена в мові VHDL, тому був введений додатковий сигнал – analysis.

Після навчання 12-ти хромосом ШНМ значення змінної змінюється із 0 на 1, а при початку нових ітерацій алгоритму – із 1 в 0. При переході сигналу Analysis із 1 в 0, також міняється NoutsForming, але в іншому процесі.

Analysis – це сигнал, що має значення 1, доти доки досліджуються дані щодо правильності отриманого рішення та перехід до наступної ітерації.

За формування затримок відповідальна змінна CLC, вона формує затримки в 1 пікосекунду, потрібно для того щоб краще проаналізувати та опрацювати результати навчання. Це дозволяє проанаізувати отримані результати оскільки алгоритм відпрацьовує досить швидко та не дає такої змоги. Тому був уведений сигнал затримки процесу навчання.

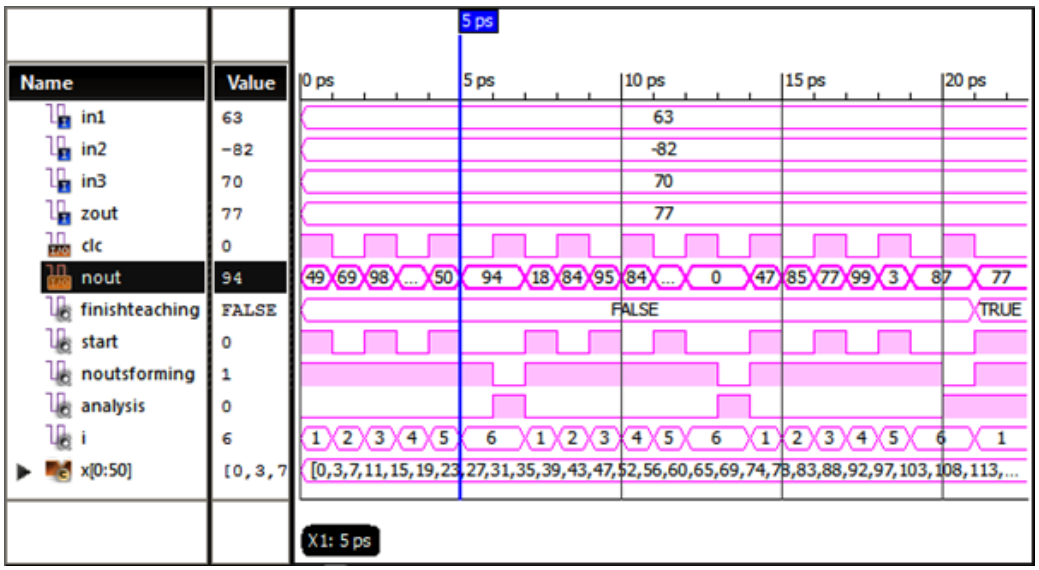

Рисунок 6.5 – Успішне знаходження результату генетичним алгоритмом на ПЛІС

Навчання хромосом відбуваються в перші 6 пікосекунд, рисунок 6.5, коли сигнал NoutsForming приймає значення 1. Значення сигналу i-ї хромосоми міняється кожної пікосекунди, таким чином це дає поштовх на зміну значення змінної start де відбувається продовження формування результату для певного масиву з особин коли змінюється значення сигнала Nout.

Аналіз інформації починається на 7-ій пікосекунді, так як сигнал Analysis=1 отримано при навчанні хромосом через нейронну мережу. Так як на даній ітерації не досягнуто необхідне значення виходу (Nout не дорівнює Zout), відбувається формування нового покоління хромосом і розпочинається нова ітерація генетичного алгоритму.

В період між 8-14 пікосекунд сигнали NoutsForming, Analysis, i, start, змінюються точно так само як і на попередній ітерації семи пікосекунд, але значення сигналу Nout змінилось, тому що алгоритм виконується для іншого покоління хромосом.

Під час третьої ітерації алгоритм відбувається отримання результату який задовільняє наші очікування, таким чином після аналізу інформації на 21-ій пікосекунді сигнал FinishTeaching приймає позитивне значення і алгоритм закінчує своє виконання, а сигнали: i, start, NoutsForming, Analysis припиняють

змінюватися, відбувається обнулення значень вагів, після отримання кінцевого значення результуючої хромосоми на останній ітерації циклу, а вихід мережі встановлюється в необхідне значення 77.

На рисунку 6.6 зображені характеристики виконання даної програми на ПЛІС. На ньому видно скільки ресурсів було використано, час швидкодії та інші показники характерні для логічних елементів ПЛІС.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Latches	220	15,360	1%
Number of 4 input LUTs	11,306	15,360	73%
Number of occupied Slices	6,096	7,680	79%
Number of Slices containing only related logic	6,096	6,096	100%
Number of Slices containing unrelated logic	0	6,096	0%
Total Number of 4 input LUTs	11,701	15,360	76%
Number used as logic	11,306		
Number used as a route-thru	395		
Number of bonded IOBs	52	173	30%
Number of MULT18X18s	13	24	54%
Number of BUFGMUXs	2	8	25%
Average Fanout of Non-Clock Nets	2.53		

Рисунок 6.6 – Технічні характеристики використання пристрою

Алгоритм знайшов правильний результат за 3 ітерації, тому що були підібрані такі значення для портів in1, in2, in3, zout.

В загальному якщо не підбирати спеціально значення, алгоритм навчання завершується за 20-70 ітерацій, приблизний час 200-500 пікосекунд, із затримками сигналу CLC, при відсутності затримок, цей алгоритм виконується менш як за 1 пікосекунду.

6.3 Реалізація блоку для навчання нейрону з використанням генетичного алгоритму

Був створений окремий модуль для блоку навчання та компоненту нейрону, який називається neuron_with_trainer.

Це потрібно було для компоновки модулю нейрону та функції навчання за допомогою генетичного алгоритму, це включає функцію активації, а також оцінку критерія закінченості алгоритму. Даний блок зображений на рисунку 6.7.

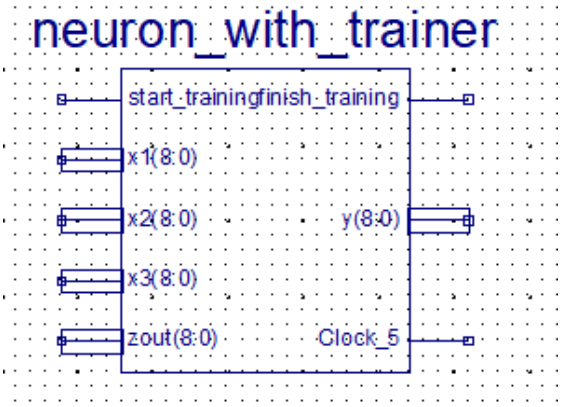

Рисунок 6.7 – Апаратний блок нейрону на ПЛІС

Модуль навчання нейрону приймає на вхід цілі числа, які не прив’язані до певного знаку. Різницю між модулем генетичного алгоритму та нейронним блоком можна знайти у значеннях входу.

Вектор вхідних сигналів, що складається із булевих змінних:

```
input1, input2, input3, input4: in STD_LOGIC_VECTOR (15 downto 0);
```
(6.5)

де входи input1, input2, input3 та input4 складаються з булевих змінних.

Вхідні порти на який надходять імпульси:

```
inside1:in integer range -50 to 50
inside2:in integer range -50 to 50
inside3:in integer range -50 to 50
```
(6.6)

де входи inside1, inside2, inside3 містять в собі значення масиву цілих чисел від -50 до 50.

Для того, щоб алгоритм розумів коли починати працювати ми повинні задати значення початку навчання. Запускаючи процес з назвою startTraining ми

розпочинаємо процес навчання, це в свою чергу трігерить роботу низки інших процесів.

Коли значення сигналу start змінюються процес neuron починає виконуватись, таким чином гарантується виконання навчання.

Далі з використанням змінної LocalOut ми отримуємо вихід нейрона для поточної ітерації, цей результат буде прирівнюватися до змінної expectedOut, що представлятиме очікуваний результат. Це дасть нам шанс сформувати похибку обчислення та на наступній ітерації зробити її меншою приблизивши значення потрібної на змінно до очікуваного результату.

В той момент коли починається навчання і кожен раз коли міняється сигнал і починає своє виконання процес CntrInI. Таким чином міняється адреса поточного синапсу в який записується отримані значення вагів після кожної ітерації.

Зміна значення лічильника супроводжується декількома іншими процесами, один із них це запис значення адресу в зміну synaddr.

Перш за все відбувається перевірка чи подія start_training наступила, якщо так, то їй присвоюється початкове значення адреси. В той час якщо ж ваги вже ініціалізовано, то ми збільшуємо значення адреси.

Використовуючи змінну r_synwren ми маємо змогу міняти ваги нейронної мережі, коли саме нам потрібно.

Коли процес навчання розпочинається відбувається ініціалізація усіх методів та полів алгоритму, в той час коли справджуються умови завершення зміни припиняються.

Спочатку потрібно виконати ініціалізацію синаптичних вагів. Ініціалізація вагів відбувається просто призначенням для них дефолтних значень які дорівнюють нулям. Після того як сходження алгоритму відбулося і лічильник нам дав про це знати ваги отримують свої граничні значення.

Виконання присвоєння значень для вагів відбувається в процесі з назвою SetWeightLen. Перезаписуючи адресу змінної randomCounterInputAddress ми отримуємо новий масив об’єктів який відповідатиме на наступні ітерації новим значенням в своїй комірці пам’яті.

Перетворення значень об’єктів хромосом у значення синаптичних вагів виконує один із важливих процесів ChrsToW. В свою чергу кожна хромосома представлена у вигляді багатовимірного масиву, що містить у своєму складі двійкові числа.

Після того як відбулося перетворення значень усіх хромосом на ціле число ми використовуємо їх далі як ваги для оптимізації значень за допомогою генетичного алгоритму.

На рисунку 6.8 зображено структурну схему на якій відбувається процес тренування апаратних блоків за допомогою одношарової мережі:

Крок 1. Завантаження портів input1, input2, input3 та expectedOut до необхідного виходу. Порти можуть бути представлені як беззнаковий вектор довжиною 9 біт. Такий вектор може представляти числа в діапазоні від 0 до 255.

Крок 2. Отримання значення сигналом start_training протягом 5 пікосекунд, а потім зробити напругу низькою, тобто онулити значення сигналу.

Крок 3. Запуск навчання доти, поки finish_training не набуде значення true.

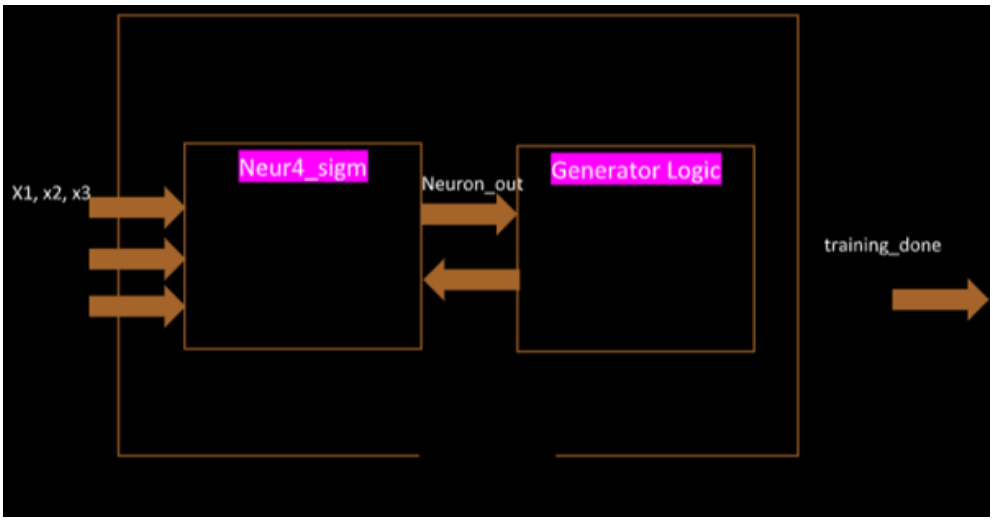

Рисунок 6.8 – Структурна схема роботи модуля навчання

Нижче, на рисунку 6.9, зображено складний випадок, в якому входи та ваги було обрано випадковим шляхом і не підбирались спеціально для найшвидшого знаходження відповіді.

Протягом кожної зміни значення таймера clock_5 також відбувається переналаштування параметрів та запуск наступних процесів щодо оцінки та продовження навчання.

В даному випадку змінна r_training_start що встановлена в 1 показує, що процес навчання розпочався. В свою чергу відбулася зміна змінної яка позначає адресу синапсиса в який відбувається запис значення вагів синапса.

Для позначення адреси синапсису та адреси вагів були створені дві змінні r_synaddr та r_synsetw. Вони змінюють свої значення, під час кожної ітерації як видно в емуляції.

Встановлюючи значення '1' в сигнал synwren ми даємо розуміти що починаємо запис вагів, це робиться для того щоб розуміти в який момент часу ми можемо це робити.

Певну умову для роботи адресу сигналу syndaddr було зроблено для покращення роботи програми, це інкрементне збільшення значення вагів у двійковій системі.

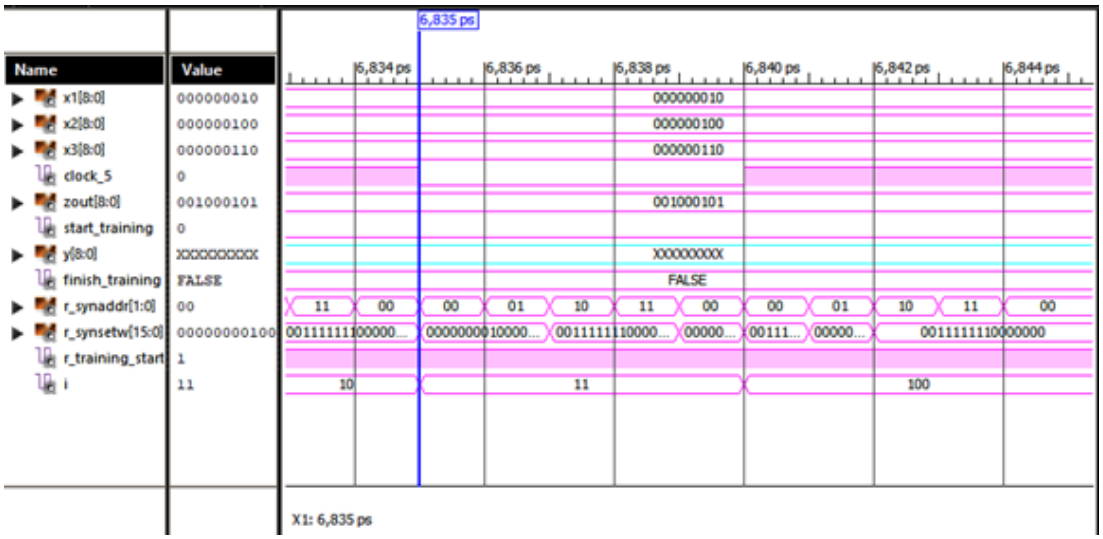

Рисунок 6.9 – Результати емуляції навчання нейрона на ПЛІС

Для виконання емуляції було використано програмне середовище ISim VHDL, на якому працював та емулював результати пристрій QC9B200L сімейства Virtex9 із застосування пакету RFA741.

Детальніше про характеристики пристрою можна дізнатися на рисунку 6.10.

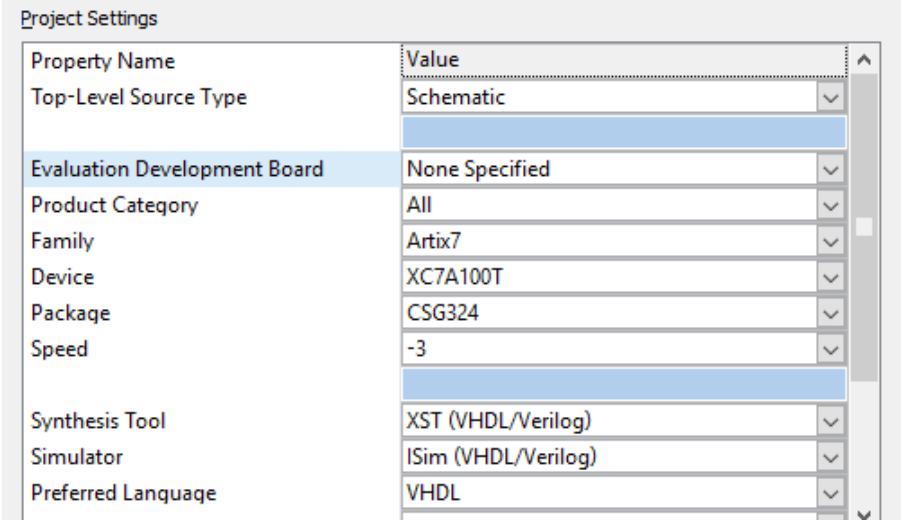

Рисунок 6.10 – Налаштування емуляції для виконання алгоритму

Для виконання емуляції потрібно було наступні ресурси: 55508КБ пам’яті та 1874мс процесорного часу(рисунок 6.11).

```
Time Resolution for simulation is 1ps.
Waiting for 1 sub-compilation(s) to finish...
Compiled 26 VHDL Units
Built simulation executable D:/Donwload/One_neuron/One_neuron/One_neuron/tb_neuron_with_trainer_isim_beh.exe
Fuse Memory Usage: 55508 KB
Fuse CPU Usage: 1874 ms
Launching ISim simulation engine GUI...
```

Рисунок 6.11 – Використання пам’яті та процесорного часу

В наступному розділі було розширені межі застосування алгоритму навчання. Було поєднано блоки разом і створено повнозв’язні багатошарові нейронні мережі, як показано на рисунку 6.12.

Було протестовано низку ПЛІС різних виробників та проаналізовано їх швидкодію та витрату ресурсів.

Поєднуючи блоки разом в мережу дає певну затримку при навчанні, але в подальшому збільшується точність прийняття рішення та швидкодія прийняття правильного рішення.

Як результат, досліди проводилися по навчанню штучного нейрону з 3-ма входами і фітнес функцією що описує сигмоїду з використанням ПЛІС для

знакових чисел з фіксованою точкою зайняв 10627 LUTs (Look Up Table - вентилі логічної матриці).

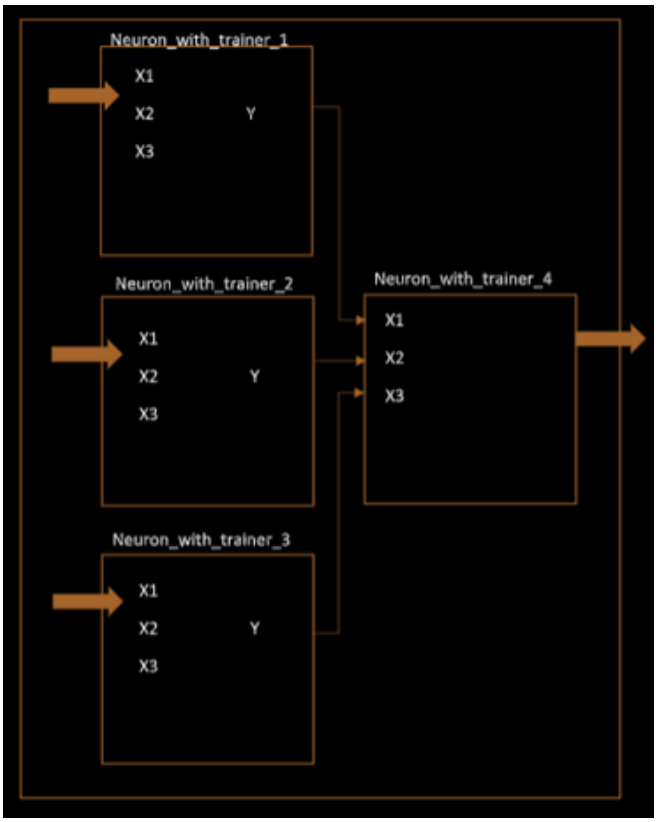

Рисунок 6.12 – Апаратна реалізація генетичного алгоритму для багатошарових нейронних мереж

Апаратний блок було розроблено на основі попередніх досліджень штучного нейрону. Було створено математичну модель та її апаратну версію. Це все дало змогу змоделювати та запустити процес навчання апаратного блоку що складався лише з одного нейрону. Було зроблено заміри швидкості навчання що склали приблизно 65.7 наносекунд використовуючи невеликий масив тестових даних.

Так як апаратний блок має певні обмеження в своїй роботі потрібно контролювати кількість використаних ресурсів. Ресурси в свою чергу залежать від тих структур даних з якими буде працювати блок, які потрібно робити з цими даними операції та конвертації. Від цього також залежатиме швидкість роботи блоку та безперебійність отриманого результату.

Досліження апаратного блоку велося в середовищі яке було створено для емуляції ПЛІС та створення на їх основі нові апаратні засоби. Тому результати потрібно сприймати як деякий еквівалент що може не відповідати дійсності.

Як результат, після досліджень було спроектовано та створено першу версію апаратного блоку навченого за допомогою генетичного алгоритму та протестованого на ПЛІС фірми Xilinx.

Було покращено швидкодію навчання нейрону за допомогою паралельних обчислень на ПЛІС.

Апаратний блок складається з декількох важливих сутностей. Це сутність самого нейрону, сутність зв’зяків та інтерфейс щодо навчання цього нейрону. Це все в сумі дає нам робоче рішення для подальши досліджень.

Таким чином цей апаратний блок енкапсулює в собі створені компоненти, він сам може існувати як окремий компонент. Що дає повну свободу створення на його основі повнозв’язних мереж та елементів комунікації.

Як результат, було спроектовано апаратний блок для використання в проектуванні штучних нейронних мереж з кількома шарами.

Поєднуючи окремі блоки навчання і формуючи повнозв’язну мережу ми маємо змогу навчати її вирішувати різні задачі.

7 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ АПАРАТНОГО БЛОКУ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

7.1 Результати навчання штучного нейрону на ПЛІС різних серій

Таблиця 7.1 – Результати моделювання штучного нейрону на ПЛІС різних серій

FPGA		Швидкодія, нс	Ресурс FPGA (LUT)
Серія	Speed		
	-5	100.155	
	-4	115.657	
	-2	7.136	
	-3	6.427	
	-3	0.99	
	-2	1.457	
	-1	1.627	
	-1	1.654	
	-2	1.271	
	-3	0.93	
	-2	1.333	
	-1	1.573	
	-2	50.137	
	-1	58.59	
	-10	71.110	
	-11	62.2	
	-12	54.015	
	-3	0.91	
	-2	1.267	
	-1	1.369	
	-1	1.987	
	-2	1.661	
	-3	1.650	

В даному розділі представлені результати навчання штучного нейрону генетичним алгоритмом для ПЛІС різних виробників.

Для цього було використано віртуальні схеми різних виробників ПЛІС у середовищі ISE Design Suite 14.7.

В даному випадку один нейрон навчався дуже швидко(таблиця 7.1) і використовував не дуже багато ресурсів для свого навчання, на відміну від багатошарових мереж які ми будемо тестувати та порівнювати далі.

7.2 Апаратні блоки штучних нейронних мереж з різною архітектурою

Головна функція апаратного блоку це обробка сигналів в паралельному середовищі виконанні. Такі мережі можуть представляти значну швидкодію та базуватися на основних принципах роботи апаратного забезпечення.

По термінології нейроінформатики це універсальні паралельні обчислювальні структури, призначенні для вирішення різноманітних класів задач.[10]

Основним принципом яким керуються дані блоки це відпрацювання різної кількості операцій одночасними темпами. Це підводить нас до викриття принципу за яким працюють конвеєри на заводах, що в свою чергу дозволяє випускати великі партії за короткий термін.

Поділивши нейрон на певну кількість конвеєрів ми можемо виконувати декілька дій одночасно та не перейматися про швидкість виконання. Таким чином кожен апаратний блок виступає у ролі обробника даних, своєрідного процесора що може працювати незалежно від інших.

Кожен нейрон представляється окремим блоком, такі блоки займаються обробкою сигналів одночасно та дають відповіді на кожен сигнал у відповідний потік який відкривається саме для того щоб передати та отримати якісь дані.

На рисунку 7.1 зображено апаратний блок з двома входами. Було отримані результати по швидкодії навчання та кількість використаного ресурсу.

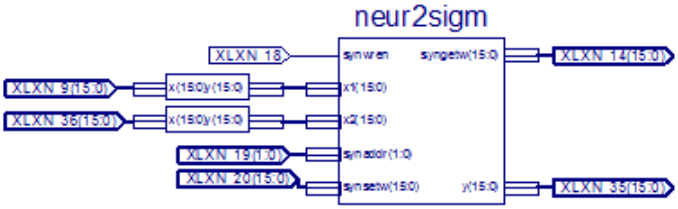

Рисунок 7.1 – Апаратний блок з двома входами

Дана мережа з одним апаратним блоком зайняла 655 вентилів логічної матриці LUTs та час максимальної затримки 92.043 наносекунди. На основі цього блоку були побудовані наступні мережі з двома нейронами (рисунок 7.2), трьома (рисунок 7.3) і чотирма (рисунок 7.4) блоками відповідно.

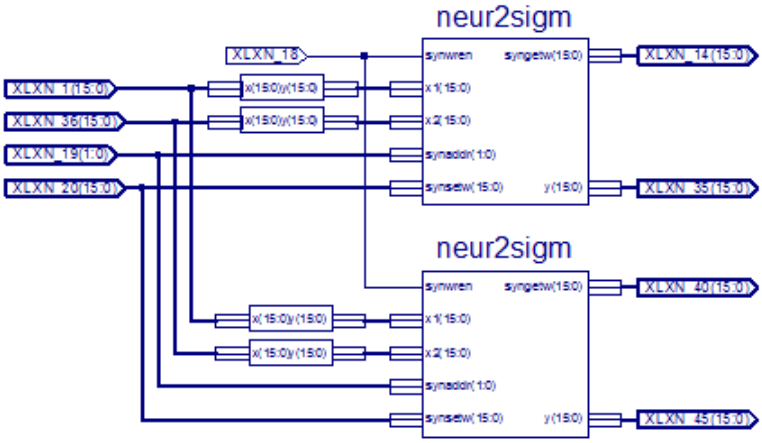

Рисунок 7.2 – Два апаратних блока з двома входами

Мережа з двома апаратними блоками використала 1210 LUTs, що майже вдвічі більше за попередню.

Таким чином демонструючи що чим більша стає мережа тим більше їй потрібно ресурсів для успішного та швидкого навчання. Далі на основі експериментальних даних ця теорія буде підтверджена.

Застосування до вхідних векторів «генетичних операторів» (у більшості випадків «схрещування»- crossover і «мутація»-mutation), створює наступне «покоління».

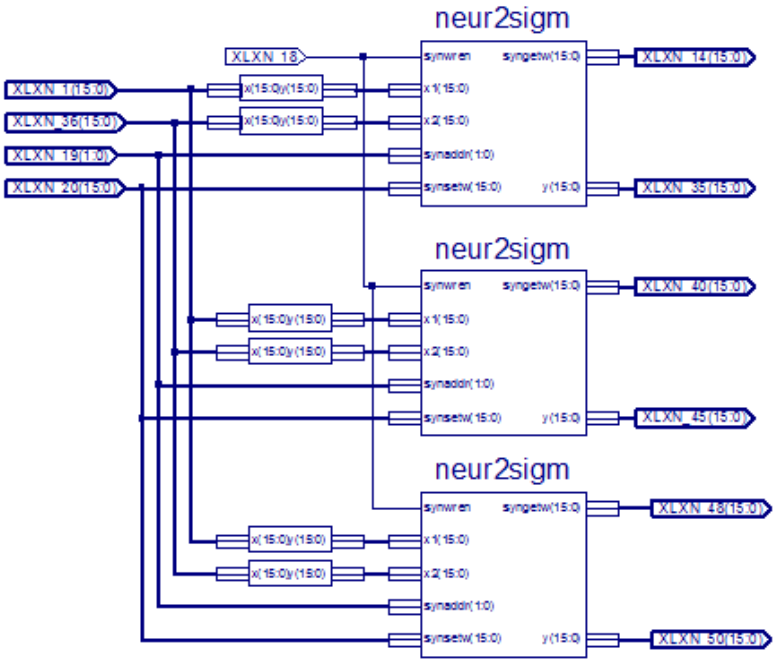

Рисунок 7.3 – Три апаратних блока з двома входами

Мережі з трьома і чотирма блоками займають 1485 LUTs і 1950, це вказує на те що ресурси залежать лінійно від кількості підключених до мережі блоків. В той час їхня швидкість майже не відрізнялась 91.133 і 91.141нс відповідно.

Як бачимо, швидкодія та кількість ітерацій для мереж зі схожою архітектурою які відрізняються лише одним нейроном, в даному випадку, три та чотири нейрони.

Дуже схожі і майже не відрізняються, лише на долі наносекунд та максимум на одну ітерацію навчання, проте ресурс який споживають ці мережі може значно відрізнятися, в даному випадку майже 25% LUTs.

Для запуску нового процесу виконання, у мові програмування передбачені збудження за допомогою сигналів. Механізм сигналів дозволяй будь-яким чином створювати та використовувати вже створені процеси та їхні дані.

Було промодельовано навчання нейрону та нейронних мереж на різних серіях ПЛІС. Зроблено заміри швидкодії, витраченого ресурсу LUTs та кількості ітерацій.

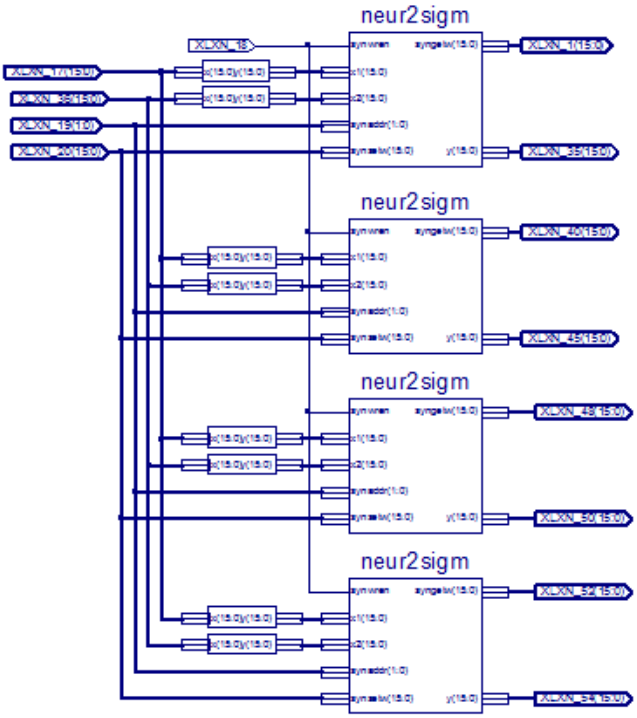

Рисунок 7.4 – Чотири нейрона з двома входами

7.3 Аналіз швидкодії та використаних ресурсів LUTs апаратних блоків

В минулому розділі ми переконалися, що чим більше нейронів містить мережа в апаратному блоці тим більше ресурсів вона потребую для свого навчання і тим менше часу вона при цьому може витрачати, або більше, це все залежить від конкретного випадку.

В цьому розділі ми побудуємо графіки залежностей швидкодії навчання нейронних мереж до використаних ресурсів, а саме до кількості використаних вентилів логічної матриці.

Далі ми побудуємо графіки які продемонструють залежність ресурсу ПЛІС від кількості входів на один та два нейрони. Також ми порівняємо швидкодію від кількості вхідних векторів.

Це дасть нам розуміння того як залежить швидкодія від кількості нейронів та вхідних векторів, та чи пришвидшить виконання навчання мережею, а також кількість витраченого ресурсу при різних архітектурних моделях, що дасть нам

розуміння того як можна реалізувати найбільш швидшу мережу із найменшими затратами ресурсів.

На рисунку 7.5 показано яким чином відбувається зростання використаного ресурсу у мережах що містять різну кількість апаратних блоків. При цьому швидкодія залишається на тому ж місці та не змінюється.

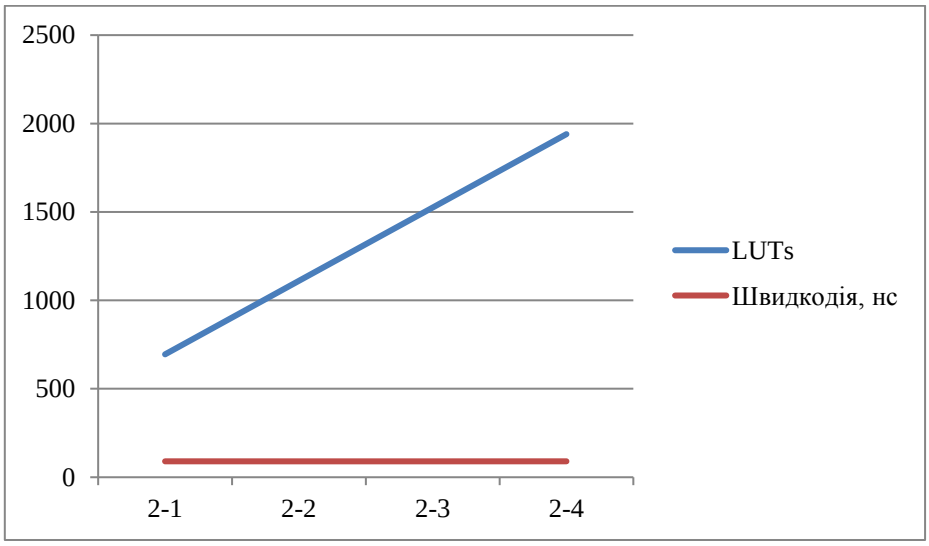

Рисунок 7.5 – графік що демонструє залежність використаного ресурсу до швидкодії навчання

На рисунку 7.5 гарно показана залежність використаного ресурсу апаратних блоків та їхньої швидкодії. Цей дослід показує наскільки відчутна мережа до нових елементів та скільки потрібно буде витратити ресурсів щоб додати певну кількість блоків.

На рисунку 7.6 показано схожі результати але вже для компонента що складається з чотирьох входів.

В таблиці 7.2 зображено порівняння нейронних мереж які складаються з таких апаратних блоків. В даній таблиці всі мережі використовують схему прямого розповсюдження. В таблиці основними заміряними параметрами є швидкодія навчання блоків та займаний ресурс кожної такої мережі.

В цьому розділі було проведено реалізацію та дослідження апаратного блоку навчання нейронних мереж.

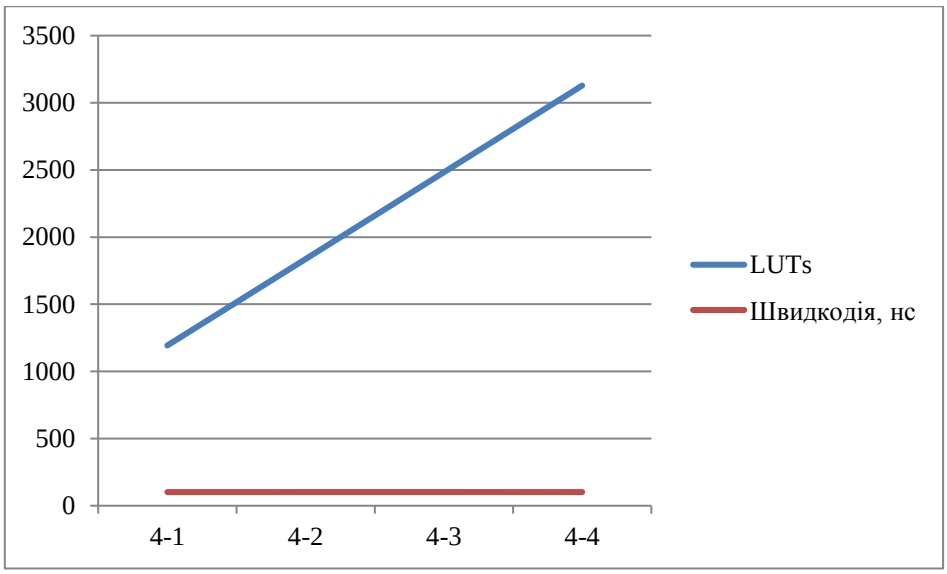

Рисунок 7.6 – графік залежності LUTs та швидкодії мереж з чотирма вхідними векторами

7.4 Результати навчання штучних нейронних мереж з різною архітектурою на ПЛІС Spartan 3

Згадані методи за допомогою яких було спроектовано дані апаратні блоки мають за основу принцип розширення. Коли регулювання мережі відбувається шляхом додавання або видалення блоку. Це все робить таку систему легко масштабованою та передбачати займаний ресурс якщо потрібно будувати складні мережі з багатьма прихованими шарами.

Це дає нам право робити висовок що оптимальними були мережі що складалися з трьох апаратних блоків. Вони мали цілком гарні результати по швидкодії та займали невелику кількість ресурсу. За допомогою них можна було вирішувати задачі різного спектру та не боятися сходження.

Як видно одні з найкращих результатів по навчанню нейрону показують чіпи Xilinx Zinq, Kintex 7, Artix 7 та Virtex 7. Непогані результати показав чіп Virtex 6, що дозволяє його використовувати як більш дешеву заміну Virtex 7.

А найгірші результати показали Xilinx Spartan 3 та Virtex 4 вони є найдешевшеми в своїй категорії, але якщо порівнювати з графічними процесорами то вони помітно їх обходять за швидкодією навчання.

Таблиця 7.2 – Результати моделювання нейронних мереж на ПЛІС

Нейронна мережа Архітектура	Кількість синапсів	Ресурс FPGA (LUT)	Кількість ітерацій	Швидкодія, нс
1-1	1	440	10	88.264
1-2	2	743	11	88.875
1-3	3	1020	11	88.321
1-4	4	1399	11	88.985
2-1	2	700	12	90.654
2-2	4	1210	12	90.984
2-3	6	1518	12	90.999
2-4	8	1890	12	91.347
3-1	3	965	15	93.541
3-2	6	1570	15	94.141
3-3	9	1998	16	95.103
3-4	12	2316	16	95.212
4-1	4	1177	18	102.698
4-2	8	1898	19	103.497
4-3	12	2383	19	103.874
4-4	16	3158	22	103.991
5-1	5	1445	29	107.371
5-2	10	2192	30	107.498
6-1	6	1679	35	109.669
6-2	12	2751	37	109.864
8-1	8	2496	42	113.297
8-2	16	3096	44	113.982

Далі на рисунках 7.7 та 7.8 зображено залежності займаного ресурсу та швидкодії від входів на апаратні блоки.

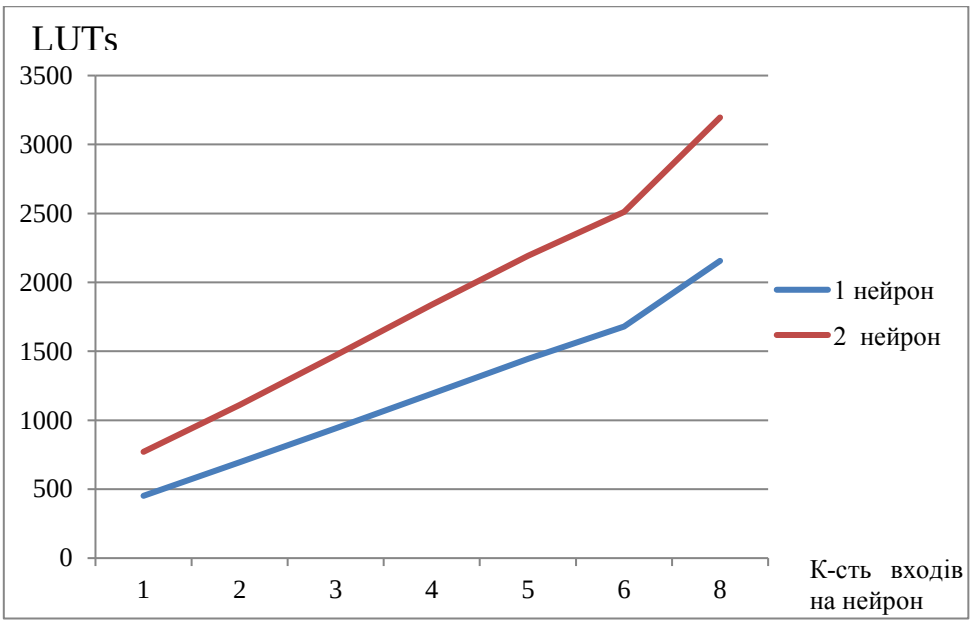

Рисунок 7.7 – Графік залежності LUTs від кількості вхідних векторів

Дані графіки дають нам розуміння того що, чим більше вхідних даних отримує апаратний блок тим більше потрібно ресурсів для їхньої обробки. При цьому чим більше кількість блоків з’єднана між собою тим більше ресурсів потребує мережа. З іншого боку швидкодія ніяким чином не змінюється від кількості вхідних елементів. Це може пояснюватись тим що обробка даних ведеться паралельно і витрати часу незначні.

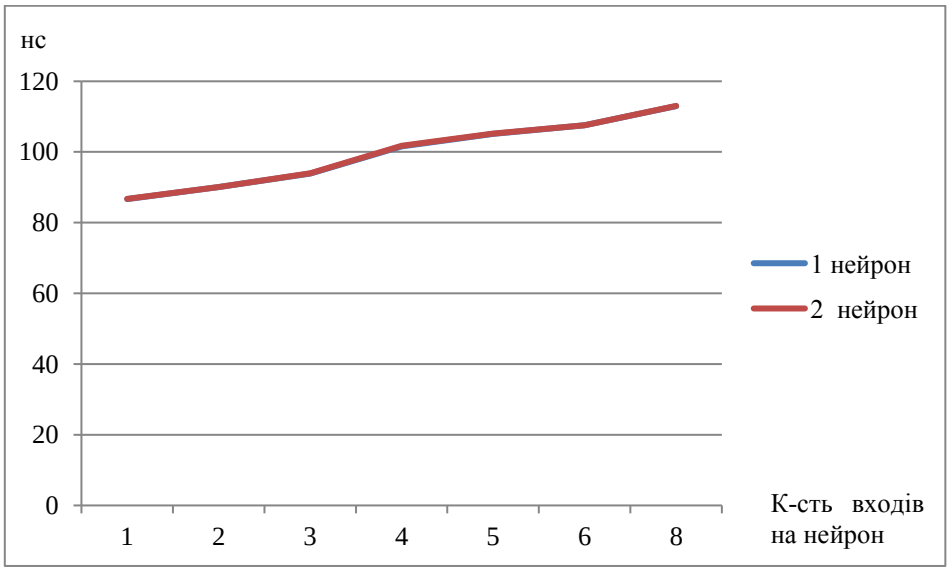

Рисунок 7.8 – Графік залежності швидкодії від кількості вхідних векторів

В даній роботі було проведено дослідження щодо створення мереж з апаратних блоків які навчалися за допомогою генетичного алгоритму. Було заміряно швидкодію їхнього навчання та витрачений ресурс. Всі дані було занесено до таблиці.

Далі було досліджено як змінюються характеристики ПЛІС різних виробників під час навчання апаратних блоків. Були заміряні ключові параметри витрати ресурсів та швидкості навчання. Всі дані були занесені до таблиці та побудовані відповідні графіки.

Апаратні блоки обробляють вхідну інформацію в паралельних потока. Це демонструє нам графік зображений вище. Там показано як зі збільшенням вхідних потоків швидкість навчання не змінюється. Завдяки такому пристосуванню апаратний блок стає універсальним компонентом для побудови складних багатошарових мереж.

В свою чергу кожен апаратний блок як окремий компонент потребує певного обсягу затрачених на себе ресурсів, які називаються логічними вентилями. Після проведення дослідження та побудови графіка чітко видно, що потреба у вентилях лінійно зростає з кожним новим доданим блоком. Цей ресурс також зростає і тоді коли потрібно додати вхідний порт, але не так сильно як при додаванні до мережі нового блоку. Все це накладає певні обмеження на використання та робить мережу дорожчою у використанні з кожним додаванням нового блоку.

Таким чином, зі збільшенням входів на нейрон, а відповідно зі збільшенням кількості синапсів лінійно збільшується кількість LUTs та кількість ітерацій.

8 РОЗРОБКА СТАРТАП-ПРОЄКТУ

8.1 Опис та аналіз ідеї проєкту

За даною роботою був розроблений стартап-проєкт. Опис змісту ідеї та напрямки використання були описані у таблиці 8.1. Ця таблиця дає нам розуміння вигоди для користувача.

Таблиця 8.1 – Опис ідеї та застосування стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. Паралельна обробка даних	Висока швидкість навчання
	2. Заміна стандартних апаратних засобів по навчанню ШНМ	Низька ціна та висока швидкодія з тією ж точністю

У таблиці 8.2 показано ріні сторони проекту від слабких до сильних. Було розглянуто основні характеристики продукту, а також його порівняння з ринковими конкурентами.

Так як основними елементами стартап-проекту є ідея то в таблиці 8.1 було описано декілька ключових факторів цієї ідеї, а саме: було описано зміст ідеї та її суть, надано інформацію по напрямкам застосування та сформовано перелік з вигодами для користувача.

Ці всі елементи дають основу для визначення сильних, слабких та нейтральних сторін ідеї проекту, що в свою чергу допомагає сформувати таблицю 8.2 в якій було надано дані щодо техніко-економічних характеристик ідеї, сформовані основні концепції потенційних конкурентів та проаналізовано кожну зі сторін.

Надалі буде проведено аудит ідеї та проаналізовано на реальних прикладах потенційних конкурентів.

Таблиця 8.2 – Аналіз сильних, слабких та нейтральних властивостей ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	потенційні конкуренти на ринку			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Реалізація штучних нейронних мереж на багатоядерних процесорах SEAFORTH	Реалізація апаратного блоку навчання нейронних мереж на масивно-паралельній архітектурі графічного процесора			
1.	Швидкодія	80-120нс	700нс	580нс	-	-	+
2.	Ціна	50-100$	50$	50-500$	-	+	-
3.	Похибка	0.5%	2%	0.2-2%	-	-	+

8.2 Технологічний аудит ідеї проекту

В таблиці 8.3 було проаналізовано можливі технології для реалізації ідеї проекту. Для цього було розглянути усі можливі варіанти технології та її реалізації. Було порівняно між собою декілька різних технологічних стеків за іхньою наявністю та доступністю.

В таблиці 8.3 було описано основні аудиторні складові проєкту. Визначення технологічної можливості реалізації проєкту. Це дасть нам уяву про те, наскільки складно та можливо можна реалізувати дану ідею.

Таблиця 8.3 – Визначення можливості реалізації ідеї технологічним стеком

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
		FPGA (Field-Programmable Gate Array) – Програмована користувачем вентильна матриця, ПКВМ	Наявна	Доступна
		CPLD – запрограмована із заводу логічна схема	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: ПЛІС та мова VHDL				

8.3 Аналіз ринкових можливостей запуску стартап-проекту

Так як аналіз ринкових можливостей продукту грає основну роль, бо він має показати чи є в нашої ідеї право на існування в реалізованом вигляді, то було досліджено основні показники стану ринку та їхні значення для нашого проекту.

Це в свою чергу дозволило нам зрозуміти динаміку ринку, його основних гравців, наявність певних обмежень для входу на ринок, загальний обсяг доходів які можна отримати від продажів, специфічні вимоги до сертифікації продукції, зрозуміти середню норму рентабельності в галузі та вцілому по ринку.

Таким чином, це продемонструвало чіткі цифри, що мають позитивну тенденція та можуть бути використані як еталон для реалізації та впровадження проєкту.

У таблиці 8.4 проаналізовано ринок продукту. Було досліджено попит на нього, обсяг та динаміку розвитку ринку.

Визначено середню рентабельність продукту та вимогі які ставить ринок перед продуктом. Все це дає повну уяву про те як розвивається та працює ринок.

Таблиця 8.4 – Основні показники стану та розвитку потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	2800
3	Динаміка ринку (якісна оцінка)	Зростання
4	Наявність обмежень для входу (вказати характер обмежень)	Недискримінаційні якісні
5	Специфічні вимоги до стандартизації та сертифікації	-
6	Середня норма рентабельності в галузі (або по ринку), %	60%

Потенційний ринок виглядає лакомим та готовий до нових гравців.

У таблиці 8.5 відображені дані по клієнтам. Була підібрана цільова аудиторія, уточнені вимоги можливих користувачів продуктом та сформовані ринком потреби.

Користувачами продукту є різного роду наукові організації, дослідницькі інститути, а також військові організації та підрозділи.

Зрозумівши базові потреби потенційних клієнтів проекту, було проаналізовано цільову аудиторію проекту, був створений список вимог до товару від споживачів, серед яких основною потребою була надійність та точність результатів обчислення.

Розібравши основні відмінності у поведінці різних цільових груп клієнтів, було отримано дані про те що більшість компаній мають фіксований бюджет і налаштовані на низьку цінову політику це в основному маленькі та середні компанії які шукають рішення для пришвидшення своїх продуктів.

Потребу яку сформував ринок це реалізація не дорогих та надійних апаратних блоків для користування всередині компанії та вирішення проблем компанії.

Таблиця 8.5 – Цільова аудиторія та вимоги споживачів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Створення швидких та пришвидшення існуючих продуктів компаній. Досліди в різних сферах	Військові департаменти, дослідницькі інститути, наукові організації	Швидкість роботи існуючих рішень компаній; Доступна ціна та підтримка рішення; Проведення частих та не дорогих експериментів	Безперебійна робота рішення; Зрозуміла інструкція з використанн я

Всі фактори розділені на фактори загроз(таблиця 8.6) та фактори можливостей(таблиця 8.7).

Таблиця 8.6 – Фактори загроз та реакцій

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Недовіра до нового гравця на ринку	Неперевірене рішення яке не дає чіткого розуміння чи варто пробувати	Запропонувати пробний період та тестовий зразок. Після чого зібрати зворотній зв’язок
2	Задачі, не для апаратних блоків навчання нейронних мереж	Задачі які не під силу виконати апаратний блок або використання не за призначенням	Створення документації з використання та прикладів для повного розуміння призначення продукту

Зрозумівши фактори які впливають на ринок та попит, було сформовано таблицю 8.7 в якій показано різні можливості та їхні реакції на можливі події які можуть відбутися з продуктом.

Таблиця 8.7 – Фактори можливостей та реакцій компанії

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Масштабування кількості клієнтів	Створення партнерських відносин з ЦА	Заключати контракти та створювати корисні ділові зв’язки з потенційними партнерами
2	Зростання темпів виробництва	Збільшення потреби в продукті в один момент часу	Розширення виробництва. Створення нових робочих місць

В таблиці 8.8 проаналізовано ринок на фактор конкурентності, розглянуто різні впливи на діяльність підприємства та можливі прояви і реакції на ці прояви.

Таблиця 8.8 – Аналіз конкурентного середовища ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції – олігополія	Декілька потужних компаній, які мають велику ринкову силу	Можна співпрацювати з конкурентами
2. За рівнем конкурентної боротьби – міжнародний	Розміщення споживачів на земній кулі не має значення	Надавати пропозицію у більшості країн світу

3. За галузевою ознакою – міжгалузева	Боротьба відокремлених товаровиробниками різних галузей економіки	Займати галузі з меншим прибутком, бо там легша конкуренція у поєднанні з ціновою перевагою
4. Конкуренція за видами товарів – товарно-родова	Конкуренція між різними товарами, що виконують схожі функції	Це краща стратегія, бо у товарно-видовій конкуренції складніше
5. За характером конкурентних переваг – цінова	Боротьба за споживачів шляхом пониження ціни	Здешевлення всіх етапів впровадження
6. За інтенсивністю – марочна	Марка не має ролі	Відсутній

В таблиці 8.9 описано конкретні конкуренти та їхні основні плюси. В даній таблиці наголос зроблено на ключові характеристики конкуренції.

Таблиця 8.9 – Дослідження конкурентів в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Xilinx, Intel, MicroChip	Тверді гравці в галузі	Різні дочірні компанії та партнери	Базова та преміум задоволення потреб	Довіра та статус
Висновки:	Індекс концентрації $CR_3 = 50\%$	Прямих можливих конкурентів	Відсутні	Клієнти диктують умови.	Обмежень немає

	Ринок помірно концентрований	не було знайдено		Ціна, точність роботи	

Розвиток в галузі сприяє отриманню конкурентів як основних клієнтів та партнерів. Це сильно спрощує виживання на ринку.

В таблиці 8.10 було зібрано усю інформацію щодо конкурентоспроможності. Було проаналізовано попередні дані та надано чіткі відповіді на питання конкурентів.

Таблиця 8.10 – Аналіз конкурентоспроможності в галузі

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Швидкодія роботи	Наявність створення паралельних потоків для пришвидшення обчислення різних процесів та збільшення швидкості навчання мереж
2	Цінова політика	Ціна апаратних блоків буде значно нижчою за існуючі аналоги, тому це привабить більше потенційних клієнтів
3	Точність обчислень	Точність забезпечується за допомогою генетичного алгоритму

Після того як ми визначили та проаналізували фактори які впливають на ринок варто оцінити ризики та можливі реакції компанії на такі фактори. Це важливо адже це може допомогти адаптуватися до нових умов.

Для формування точного рейтингу товарів-конкурентів було проаналізовано головні фактори конкурентноспроможності: швидкодія роботи, цінова політика та точність обчислень.

За даними факторами була побудована порівняльна таблиця 8.11, де знаком «+» показано відношення рейтингу товарів-конкурентів до запропонованого.

Таблиця 8.11 – Рейтинг факторів конкурентоспроможності проєкту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			–3	–2	–1	0	+1	+2	+3
1	Швидкодія роботи	16	-	-	-+	-	-	-	-
2	Цінова політика	18	-	+	-	-	-	-	-
3	Точність обчислень	14	-	-	-	+	-	-	-

Для повного розуміння усіх можливостей та загроз які спідкають компанію потрібно створити таблицю зі SWOT-аналізом, в даному випадку вона у нас таблиця 8.12 в якій чітко описані головні фактори компанії. Ці чинники дадуть нам повне розуміння та передбачення щодо дій компанії в різних ситуаціях.

Таблиця 8.12 – SWOT-аналіз стартап-проєкту

Сильні сторони: Ціна виробництва продукту. Ціна реалізації продукту. Висока швидкість виконання. Необмежене коло для експериментів	Слабкі сторони: Недовіра до нового гравця та тиск з боку конкурентів. Недостатня кількість відгуків клієнтів
Можливості: Партнерство з уже відомими компаніями та отримання клієнтів серед знайомих	Загрози: Копіювання продукту незважаючи на права інтелектуальної власності та різного роду ліцензії

В таблиці 8.13 було досліджено альтернативні кроки з впровадження продукту та ймовірні строки реалізації. До важливих спостережень варто віднести строки реалізації проекту так як при створенні альянсу з партнерів та створення

мінімальної версії проекту яку можна буде давати користувачам на деякий проміжок часу, це все може значно пришвидшити реалізацію проекту та дати краще розуміння продукту від потенційних клієнтів.

Таблиця 8.13 – Орієнтовний комплекс заходів з впровадження стартап-проєкту

№ п/п	Альтернатива ринкової поведінки	Імовірність отримання ресурсів	Строки реалізації
1	Заключення партнерських відносин з суміжними фірмами	Досить імовірне	4 місяці
2	Пробний період та впровадження	Імовірне	3 місяці

Альтернативною поведінкою було обрано створення пробного періоду для продукту щоб у користувачів був час ознайомитись з продуктом та отримати довіру.

Також було обрано заключення партнерських відносин з суміжними компаніями яким було б цікаво використати апаратний блок у своїх напрацюваннях що збільшить клієнтську базу та дасть час на розвиток продукту.

8.4 Розробка стратегії проєкту

В таблиці 8.14 було проаналізовано різні групи потенційних користувачів. Це дасть змогу розробити проєкт під особливості та потреби кожного кому цікавий буде апаратний блок. Таким чином це впливає на подальшу стратегію та розвиток компанії.

Таблиця 8.14 – Потенційні групи клієнтів на ринку

№	Опис	Готовність	Орієнтовний	Інтенсивність	Простота

п/п	профілю цільової групи потенційних клієнтів	споживачів сприйняти продукт	попит в межах цільової групи (сегменту)	конкуренції в сегменті	входу у сегмент
1	Наукові організації	Середня	50%	Низька	Низька складність
2	Дослідницькі інститути	Середня	50%	Середня	Середня складність
3	Військові департаменти	Висока	70%	Низька	Висока складність
Які цільові групи обрано: Дослідницькі інститути Наукові організації					

Після дослідження цільової аудиторії були винесені певні висновки, які дозволили обрати базову стратегію розвитку продукту, що в свою чергу дало поштовх на розуміння своїх сильних та слабких сторін.

В таблиці 8.15 сформовано стратегію охоплення та завоювання ринку. Було досліджено основні позиції розвитку та росту проєкту.

Таблиця 8.15 – Аналіз стратегії розвитку продукту

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Співпраця з науковими та дослідницькими	Концентрація усіх зусиль на одному	Швидкість роботи апаратного блоку та низька ціна	Стратегія спеціалізації та розвитку

	організаціями на початковому етапі з можливістю подальшого розвитку у військовій сфері	напрямку галузі з подальшим масштабуванням	впровадження та підтримки даного рішення	

Так як стратегія розвитку та розуміння конкурентоспроможньої позиції відповідно до обраної альтернативи є ключовимим факторами для визначення базової стратегії конкурентної поведінки на ринку, то було проведено дослідження на цю тему та сформовано базові критерії поведінки по відношенню до конкурентів на ринку.

Для вибору правильної стратегії був використаний попередній досвід створення продуктів і було чітко сформовані межі за якими буде відбуватись розвиток продукту. В таблиці 8.16 представлено вибір стратегії конкурентної поведінки.

Таблиця 8.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Шукати нових і забирати існуючих	Можливо	Утримання менш конкурентної ніші

У таблиці 8.17 було визначено позиціонування компанії відносно базової стратегії та асоціацій що формують позицію проєкту.

Таблиця 8.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Швидкість роботи; Низька ціна	Стратегія утримання	Ціна на підтримку та впровадження. Швидкість роботи	Апаратні блоки як головні схеми нейромережевих зв’язків та інтерфейсів

8.5 Розробка маркетингової програми стартап-проєкту

В таблиці 8.18 було сформовано головні концепції маркетингової стратегії та просування товару.

Таблиця 8.18 – Основні переваги та вигоди проєкту для клієнта перед конкурентами

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидко ставити експерименти та отримувати результат	Ціна на підтримку та впровадження. Швидкість роботи	Використання ресурсів ПЛІС та час роботи

В таблиці 8.19 виконаний опис трьох рівнів товару.

Таблиця 8.19 – Трирівнева модель товару

Рівні товару	Сутність та складові		
І. Товар за задумом	Створення не дорогих та апаратних блоків навчання нейронних мереж генетичним алгоритмом		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Швидкодія 2. Використаний ресурс 3. Ціна		
	Якість: похибка до 0.01%		
	До продажу: документація з використання та допомога у впровадженні		
	Після продажу: збір зворотнього зв’язку від усіх клієнтів		
Захист продукту від копіювання за рахунок патенту та ліцензій			

Після визначення переваг та сильних сторін продукту, було сформовано список з переваг та визначення основних відмінностей над конкурентами. Це в свою чергу визначило подальший маркетинговий план дій для отримання більшого попиту продукту на ринку. В таблиці 8.20 було проаналізовано границі по встановлення ціни на продукт та послуги.

Таблиця 8.20 – Границі встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	-	50-500$	1000-1000000$	40-300$

В таблиці 8.21 було описано один із важливих елементів компанії це система продаж та постачальників.

Таблиця 8.21 – Системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Через знайомих. Через існуючі платформи та онлайн тендери. Через партнерські відносини з іншими компаніями	Розповсюдження та налагодження ділових зв’язків	Канал декількох рівнів	Рівний розподіл

При формуванні системи збуту було отримано дані про специфіку закупівельної поведінки цільової аудиторії що дає нам розуміння потреби користуватися відкритими тендерами та вести подальші перемовини з можливими користувачами там.

Остання складова маркетингової програми знаходяться в таблиці 8.22. Тут представлена концепція комунікацій. Один з найважливіших аспектів маркетингу це комунікація.

Так як компанія планує свою діяльність на світовому рівні, то будуть використовуватись прямі офіційні канали комунікації. Для позиціонування було обрана послідовна реалізація даної позиції та унікальність послуги, таким чином завданням рекламного повідомлення буде заявити про позицію доступності для широкого кола споживачів.

Для того щоб концепція рекламного звернення до цільового користувача носила характер прямого звернення то було обрано саме раціоналістичний тип рекламного звернення що дасть користувачу розуміння про сам продукт та його способи використання.

Таблиця 8.22 – Стратегія комунікацій проєкту

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Місцевий та світовий ринки	Офіційні канали та реклама	Утримання позиції в галузі та пропозиція унікального рішення	Розказати про продукт та його використання, навіщо він споживачам	Використовувати різного роду рекламу та ЗМІ

8.6 Висновки

Після дослідження цільової аудиторії були винесені певні висновки, які дозволили обрати базову стратегію розвитку продукту, що в свою чергу дало поштовх на розуміння своїх сильних та слабких сторін.

Було виявлено ключеві характеристики та можливих користувачів системи. Маркетингова програма підлаштована до відповідної цільової аудиторії продукту.

Рентабельність ринку висока, попит є, це дає можливість ринкової комерціалізації проекту та подальшого розвитку і масштабування проекті на різні галузі та сфери.

Для формування точного рейтингу товарів-конкурентів було проаналізовано головні фактори конкурентноспроможності: швидкодія роботи, цінова політика та точність обчислень.

ВИСНОВКИ

В даній магістерській дисертації вирішена проблема збільшення швидкості навчання ШНМ за рахунок розробки апаратного блоку та його реалізації і навчання генетичним алгоритмом на ПЛІС багатошарової нейронної мережі при мінімальному використанні обчислювального ресурсу.

Реалізований за даним алгоритмом апаратний блок може формувати нейронні мережі та використовує сигмоїду як передатну функцію. Були використані стандартні бібліотеки для роботи з числами з фіксованою точкою. Такий апаратний блок зайняв 1089 LUTs ресурсів(Look Up Table - вентилі логічної матриці).

Кожен нейрон прихованого шару мережі розроблений на ПЛІС як окремий обчислювальній блок займає 93 LUTs. Похибка абсолютна ± 0.005. Швидкодія, як сумарна затримка комбінаційної схеми блоку нейронної мережі, склала 114.1 нс. Додавання 4-х нейронів прихованого шару до нейрону з сигмоїдальною функцією активації збільшила кількість LUTs всього на 62%, а час на 50%.

Згадані методи за допомогою яких було спроектовано дані апаратні блоки мають за основу принцип розширення. Коли регулювання мережі відбувається шляхом додавання або видалення блоку. Це все робить таку систему легко масштабованою та передбачати займаний ресурс якщо потрібно будувати складні мережі з багатьма прихованими шарами.

Апаратний блок було розроблено на основі попередніх досліджень штучного нейрону. Було створено математичну модель та її апаратну версію. Це все дало змогу змоделювати та запустити процес навчання апаратного блоку, що складався не лише з одного нейрону.

В даній роботі було проведено дослідження щодо створення мереж з апаратних блоків які навчалися за допомогою генетичного алгоритму. Було заміряно швидкодію їхнього навчання та витрачений ресурс.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нейросетевые системы управления, Терехов В.А., Ефимов Д.В., Тюкин И.Ю. 2002. - 183с.

2. Миркес Е. М.,Нейрокомпьютер. Проект стандарта. – Новосибирск: Наука, 1999. – 337 с.

3. Анализ принципов построения и свойств устройств для моделирования нейрона, О. К. Колесницкий, к. т. н., доц.; И. В. Бокоцей; А. А. Коренной, 2012.

4. Volodymyr Shymkovych. Method and technology of synthesis of neural network models of object control with their hardware implementation on FPGA. / P.I., Kravets, V., Samotyy, V.M., Shymkovych//2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, 2017, pp. 947-951.

5. Нейронні структури в виробничих технологіях цивільної авіації [Електронний ресурс] : Режим доступу: http://lecture.in.ua/lekciya-2-nejronni-strukturi-v-virobnichih-tehnologiyah-civile.html — Назва з екрану. 04.05.2017 р.

6. Integrated Circuits (ICs). Embedded - FPGAs (Field Programmable Gate Array). Xilinx Inc. XC7K160T-2FFG676C [Електронний ресурс] : Режим доступу: https://www.digikey.com/product-detail/en/xilinx-inc/XC7K160T-2FFG676C/122-1836-ND/3671575— Назва з екрану.

7. Integrated Circuits (ICs). Embedded - FPGAs (Field Programmable Gate Array). Xilinx Inc. XC7V2000T-CES9937 [Електронний ресурс] : Режим доступу: https://www.digikey.com/product-detail/en/xilinx-inc/XC7V2000T-1FLG1925CES9937/122-1802-ND/3585744— Назва з екрану.

8. Нейроні мережі і генетичні алгоритми – К.:«Корнійчук», . 2008. – 446 с.

9. Volodymyr Shymkovych. Software means of modeling of the vector type of reactive engine control system. / Doroshenko, A., Shymkovych, V., Fedorenko, V. // CEUR Workshop Proceedings, 2139, pp. 296-305

10. Volodymyr Shymkovych. A real time control system for balancing a ball on a platform with FPGA parallel implementation. /Samotyy, V., Telenyk, S., Kravets, P., Symkovych, V., Posvistak, T.// Czasopismo Techniczne, Poland, 2018, 109-117.

11. Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks, 2012.

12. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка, 2012: Кравець П.І., Шимкович В.М., Зубенко Г.А., Технологія апаратно-програмної реалізації штучного нейрона та штучних нейронних мереж засобами FPGA.

13. Метод оптимизации весовых коэффициентов нейронных сетей с помощью генетического алгоритма при реализации на программируемых логических интегральных схемах / П.И. Кравец, В.Н. Шимкович // Электронное моделирование. — 2013. — Т. 35, № 3. — С. 65-74. — Бібліогр.: 12 назв. — рос.

14. Fletcher R., Reeves C.M. Function minimization by conjugate gradients // Computer Journal. — 1964. — Vol. 7. — P. 149—157.

15. Головко В. А. Нейронные сети: обучение, организация и применение — ИПРЖР, 2001.

16. A Genetic Algorithm Tutorial by Darrell Whitley Computer Science Department Colorado State University.

17. Volodymyr Shymkovych. Neural Network Control System with Direct and Inverse Model of Control Object Hardware and Software Realization of in FPGA / Petro Kravets, Volodymyr Shymkovych./ Information Technology, Computational and Experimental Physics. Kulczycki P., Kowalski P.A., Lukasik S. (eds.) AGN-UST. Krakow, Poland - 2016. - pp. 180-183

18. Шимкович В.М. Розробка та дослідження технології оцінювання показників нейромережевих моделей MIMO-об'єктів керування./ Кравець П.І., Лукіна Т.Й., Шимкович В.М., Ткач І.І. // Вісник НТУУ «КПИ». Інформатика, керування та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – №57. – С. 144–149.

19. Stephen D. S., Sharad S., and Ashok S. A hardware engine for genetic algorithms, Technical report UNLCSE-97-001, University of Nebraska Lincoln, July, USA, 4, 1997.

20. L.-M. Ionescu, A. Mazare, A.-I. Lita, G. Serban, Fully integrated artificial intelligence solution for real time route tracking, in: 2015 38th International Spring Seminar on Electronics Technology (ISSE), IEEE, 2015, pp. 536– 540.

21. В.В. Гудилов, Л.А. Зинченко. Аппаратная реализация вероятностных генетических алгоритмов с параллельным формированием хромосомы

22. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. — 1-е. — М.: Гаряча лінія - Телекому, 2001. — С. 382.

23. В. А. Терехов, Д. В. Єфімов, И. Ю. Тюкин Нейромережні системи керування. — 1-е. — Высшая школа, 2002. — С. 184.

24. Ф. Уоссермен. Нейрокомп’ютерна техніка. Теорія і практика. — 1-е. — М.: Мир, 1992. — С. 240.

25. Саймон Хайкин. Нейроні мережі: повний курс = Neural Networks: A Comprehensive Foundation. — 2-е. — М.: «Вильямс», 2006. — С. 1104.

26. Роберт Каллан. Основні концепції нейроних мереж = The Essence of Neural Networks First Edition. — 1-е. — «Вильямс», 2001. — С. 288.

27. Л.Н. Ясницкий. Введення в штучний інтелект. — 1-е. — Издательский центр "Академия", 2005. — С. 176.

28. Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергєєв. Генетичні алгоритми, штучні нейроні мережі і проблеми віртуальної реальності. — Замовне. — Х.: ОСНОВА, 1997. — С. 112.

29. Д. Рутковская, М. Пилиньский, Л. Рутковский. Нейронные сети, генетические алгоритмы и нечеткие системы —1-е. — М.: Гарячая линия – Телеком, 2007. — С. 384.

30. Кравець П.І., Шимкович В.М., Ференс Д.А. Метод и алгоритмы реализации на ПЛИС функции активации для искусственных нейронных сетей. Международный научно-технический журнал Электронное моделиро-вание. – 2015. – 37, №4. – С. 63-73.

31. Petro Kravets, Volodymyr Shymkovych. Neural Network Control System with Direct and Inverse Model of Control Object Hardware and Software Realization of in FPGA/ Information Technology, Computational and Experimental Physics. Kulczycki P., Kowalski P.A., Lukasik S. (eds.) AGN-UST. - 2016. pp. 180-183.

32. S.F., Telenyk, P.I., Kravets, V.M., Shymkovych, T.V., Posvistak, FPGA Implementation of the PID Algorithm for Real Time Ball Balancing on the Platform, Proceedings of The Fourth International Conference on ‘Automatic Control and Information Technology’ (ICACIT’17) December 14-16, 2017, Cracow, Poland, pp. 160-169.

33. P., Kravets, V., Shymkovych, Z., Yurchenko. Algorithm for the Implementation of Radial-Basic Neural Networks and their Activation Functions on the FPGA. Proceedings of The Fourth International Conference on ‘Automatic Control and Information Technology’ (ICACIT’17) December 14-16, 2017, Cracow, Poland, pp. 122-129.