Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи та засоби генерації коду на VHDL

(тема)

Виконав:

студент II курсу, групи СПм-22-4

Івановський П.С.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Івановському Петру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи та засоби генерації коду на VHDL

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи

генерація коду

FPGA

VHDL

4. Перелік питань, що потрібно опрацювати у роботі

Аналіз предметної області

FPGA

Методи апаратного прискорення для згорткових нейронних мереж

Розробка інструменту генерації VHDL описів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 16 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	01.04.2024 – 06.04.2024	
2	Огляд існуючих рішень та методів	07.04.2024 – 12.04.2024	
3	Розробка моделі	13.04.2024 – 18.04.2024	
4	Вибір програмних засобів	19.04.2024 – 25.04.2024	
5	Програмна реалізація	26.04.2024 – 02.05.2024	
6	Аналіз отриманих результатів	03.05.2024 – 16.05.2024	
7	Оформлення записки	17.05.2024 – 11.06.2024	
8	Представлення роботи в ЕК	12.06.2024	

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 62 с., 23 рис., 2 дод., 28джерел.

ГЕНЕРАТОР КОДУ, VHDL, FPGA, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, CUDA.

Метою кваліфікаційної роботи є аналіз методів та засобів генерації коду на VHDL для існуючих рішень апаратного прискорення за рахунок використання штучних нейронних мереж.

У ході виконання кваліфікаційної роботи Проведено аналіз методів та засобві генерації кода VHDL. Розроблено метод апаратного прискорення згорткових нейронних мереж на FPGA з використанням VHDL. Проведено аналіз топологій згорткових нейронних мереж; аналіз існуючих алгоритмів навчання згорткових нейронних мереж. Досліджено існуючі рішення апаратного прискорення. Розроблений інструмент генерації VHDL. Використання методу значно зменшить час розробки, необхідний для впровадження згорткової нейронної мережі, також знащить недоліки, що виникають через складність розробки мов опису апаратних засобів.

ABSTRACT

Master’s thesis: 62 pages, 23 figures, 2 appendices, 28 sources.

CODE GENERATOR, VHDL, FPGA, CONVOLUTIONAL NEURAL NETWORKS, CUDA.

The major goal of this thesis is to analyze methods and means of VHDL code generation for existing hardware acceleration solutions through the use of artificial neural networks.

In order to the qualification work an analysis of methods and means of VHDL code generation was carried out. A method of hardware acceleration of convolutional neural networks on FPGA using VHDL has been developed. The topologies of convolutional neural networks were analyzed; analysis of existing learning algorithms for convolutional neural networks. Existing hardware acceleration solutions have been studied. Developed VHDL generation tool. The use of the method will significantly reduce the development time required for the implementation of a convolutional neural network, as well as the disadvantages arising from the complexity of the development of hardware description languages.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

ШНМ – штучна нейронна мережа

CNN – згорткова нейронна мережа

FPGA – програмована вентильна матриця

SGD – стохастичний градієнтний спуск

VHDL – мова опису апаратних засобів.

ВСТУП

Згорткова нейронна мережа (CNN), популярний алгоритм машинного навчання, зарекомендувала себе як високоточний і ефективний алгоритм, який використовується в різних програмах, таких як: розпізнавання рукописних цифр, візуальне розпізнавання та класифікація зображень.

Найсучасніші CNN є інтенсивними обчисленнями, але їх паралельна і модульна природа робить такі платформи, як Field Programmable Gate Array (FPGA), добре придатними для процесу прискорення. Як правило, згорткові нейронні мережі потребують дуже тривалого циклу розробки, щоб бути реалізованим або прискореним за допомогою FPGA, тому в цій кваліфікаційній роботі пропонується інструмент генерації VHDL (VGT), який за допомогою коду VHDL (архітектура CNN) може бути створений на льоту для різних моделей CNN. Згенерований код (або архітектура) оптимізований, модульний, паралельний, реконфігурований, масштабований, повністю конвеєрний і адаптований до різних моделей CNN.

У роботі проведено демонстрацію інструмента автоматичного генерування VHDL та його адаптивність. Були реалізувані дрібномасштабна модель CNN «LeNet-5» та великомасштабна модель «AlexNet». Згенерований код для дрібномасштабної моделі не містить жодного керування зовнішньою пам’яттю для параметрів CNN, тоді як параметри автоматично жорстко кодуються як константи, на відміну від того, як це зазвичай робиться для великомасштабних моделей. На Xilinx Virtex-7, що працює на частоті 200 МГц, система здатна обробляти до 125 тисяч зображень 28×28 в секунду для LeNet-5 і досягла максимальної продуктивності 611,52 GOP/s для AlexNet.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN), тип нейронних мереж і видатний алгоритм машинного навчання, натхненний зоровою корою мозку та математичною операцією під назвою згортка, наразі є найбільш життєздатним підходом до розуміння зображень. Справді, CNN завоювали популярність не тільки в класифікації зображень і відео [1,2,3,4], а й у багатьох інших програмах, таких як розпізнавання мовлення [5,6], аналіз тексту [1,2], а також візуальне розпізнавання об'єктів і самокеровані автомобілі [7]. Ідея нейронних мереж існує з 20-х років 19 століття, але останні покоління високопродуктивних обчислювальних платформ дозволили еволюцію CNN.

За останні пару років було представлено багато моделей CNN, таких як LeNet-5, AlexNet, VGG, GoogleNet, ResNet. Наприклад, модель AlexNet [8] перемогла ImageNet LargeScale Vision Recognition Challenge (ILS VRC) 2012, досягнувши точності в топ-5 84,7%. Виняткова продуктивність згорткових нейронних мереж є компромісом до величезних обчислювальних витрат, яких вони потребують, коли велика модель CNN вимагає понад мільярд операцій на зображення.

Завдяки наявності потужних платформ, таких як графічні процесори (GPU), цей рівень продуктивності може бути досягнутий, але через високе енергоспоживання графічних процесорів неможливо вбудувати такі рішення в невеликій портативній системі. Для ефективної реалізації CNN були розглянуті різні платформи, а FPGA були досліджені як найперспективніші [9]. Цікаво, що FPGA, здається, добре підходять для цієї роботи, оскільки вони піддаються реконфігурації, використовують переваги властивого CNN паралелізму та енергоефективні.

CNN відомі своїм частим доступом до даних, складністю обчислень і дуже тривалим циклом розробки FPGA, тому потрібна ефективна реалізація. У цій квалфікаційній роботі представлено інструмент генерації VHDL, який зменшує час і зусилля в процесі впровадження CNN на FPGA. Інструмент дозволяє користувачам легко налаштовувати модель CNN за допомогою графічного інтерфейсу користувача та генерувати для неї високо оптимізований код VHDL. Згенерований VHDL відображає модульну, дуже паралельну, масштабовану, реконфігуровану та повністю конвеєрну реалізацію цільової моделі CNN.

1.2 Машинне навчання

Машинне навчання – це підхід зі штучним інтелектом, за допомогою якого машини «комп'ютери» навчаються подібно до того, як навчаються люди. Машинне навчання стосується того, як програмні системи можуть автоматично навчатися та вдосконалюватися з досвідом. Навчання в цьому контексті є не навчанням на пам'ять, але розпізнавання складних моделей і прийняття розумних рішень на основі даних. Складність полягає в тому, що набір усіх можливих рішень з урахуванням усіх можливих вхідних даних занадто складний для опису. Для вирішення цієї проблеми в області машинного навчання розробляються алгоритми, які знаходять знання з конкретних даних і досвіду, засновані на надійних статистичних і обчислювальних принципах.

Сфера машинного навчання об'єднує багато різних підходів, таких як теорія ймовірностей, логіка, комбінаторна оптимізація, пошук, статистика, навчання з підкріпленням і теорія управління. Розроблені методи лежать в основі багатьох застосувань, починаючи від зору до обробки мови, прогнозування, розпізнавання образів, ігор, аналізу даних, експертних систем та робототехніки [10]. Машинне навчання зазвичай поділяють на два основних типи: предиктивне (без нагляду) або навчання з наглядом [11].

Існує третій тип машинного навчання, який широко не використовується, відомий як навчання з підкріпленням. Останній тип корисний для того, щоб навчитися діяти, коли періодично отримують сигнали нагороди чи покарання.

1.2.1 Контрольоване навчання

Під час навчання з наглядом надаються вхідні та вихідні дані, де вхідні дані обробляються мережею, а отримані результати порівнюються з бажаними результатами. Потім помилки поширюються назад через систему, змушуючи систему коригування вагів, які контролюють мережу. Цей процес відбувається знову і знову, оскільки вагові коефіцієнти постійно коригуються, щоб мінімізувати помилку. Набір даних, який забезпечує навчання, називається навчальним набором. Під час навчання мережі один і той самий набір даних обробляється багато разів, оскільки вагові показники з’єднань постійно уточнюються.

1.2.2 Неконтрольоване навчання

Під час навчання без нагляду мережа надається входами, але не бажаними виходами. Потім сама система повинна вирішити, які функції вона використовуватиме для групування вхідних даних. Це часто називають адаптацією. Навчання без нагляду зазвичай використовується для кластеризації.

1.2.3 Нейронні мережі

Розвиток нейронних мереж припадає на початок 19го століття. Моделі ШНМ створені на основі біологічних нейронних мереж, заснованих на функціональності нейронів. Зазвичай нейронні мережі складаються з безлічі

штучних нейронів, які взаємопов’язані один з одним. Нейрони влаштовані таким чином, щоб утворити нейронну мережу з прямим зв’язком. Нейрони є основним будівельним блоком нейронної мережі, де нейрон отримує низку вхідних сигналів xi від інших нейронів, і ці вхідні сигнали помножуються на ваги Wi для імітації синаптичної взаємодії на дендритах. Зважені вхідні дані підсумовуються, зміщуються зі значенням, що зазвичай дорівнює 1, і подається в нелінійну функцію активації, яка виробляє вихідний сигнал нейрона.

Чому саме нейронні мережі, а не звичайне комп’ютерне програмування? Ідея нейронних мереж полягає в тому, що вони не вимагають явного опису проблеми, а також програмування для виконання певного завдання. Нейронна мережа адаптується сама під час навчання фазі, на основі прикладів подібних задач. Коли мережа завершила свою фазу навчання, мережа може зв’язати дані проблеми з рішеннями, вхідні дані та вихідні дані та може запропонувати можливе рішення для нової проблеми.

Перед розгортанням нейронної мережі мережу необхідно навчити на певному наборі прикладів, де параметри (ваги та зміщення) в нейронній мережі не вибираються вручну, а вивчаються на цьому етапі навчання. Як згадувалося в навчанні з керівництвом, мережа надається набором позначених навчальних прикладів. Тренування починається з малих і випадково ініціалізованих ваг.

Вхідні дані помножуються на вагові показники та передаються до функції нелінійності, яка дає вихідні дані для порівняння з позначеними прикладами за допомогою функції втрат, яка вимірює різницю між істинним результатом (позначені приклади) та результатом функції нелінійності. Помилка мінімізується за рахунок оптимізації значень ваг. Використовуючи алгоритм зворотного поширення [4] , вихідні дані поширюються весь шлях назад у мережі. Зазвичай це вирішується за допомогою стохастичного градієнтного спуску (SGD) [12].

Алгоритм стохастичного градієнтного спуску є, мабуть, найбільш часто

використовуваною процедурою оптимізації для навчання глибоких нейронних мереж [13], в якій ваги мережі переміщуються вздовж від'ємного градієнта функції продуктивності. Термін зворотне поширення відноситься до способу обчислення градієнта для нелінійних багатошарових мереж. Алгоритм поширює помилку, яка обчислюється як різниця між результатом прямого проходу та очікуваним виводом, назад по всій мережі, щоб налаштувати значення ваг, щоб мінімізувати помилку.

1.3 Згорткові нейронні мережі

Згорткові нейронні мережі – це клас нейронних мереж із прямим зв'язком, які підходять для операцій з двовимірними даними, такими як зображення. CNN подібні до звичайних нейронних мереж, де вони складаються з нейронів, які мають вагу та зміщення. Нейрони в CNN отримують вхідні дані, виконують точковий добуток, за яким слідує нелінійність, а потім застосовують функцію втрат до шару класифікації. Основна відмінність між CNN і звичайними мережами з прямим зв'язком полягає в тому, що CNN краще справляються з двовимірними вхідними даними, і тому вони в основному використовуються в класифікації зображень. CNN зазвичай починають зі згорткового шару, де він приймає вхідні зображення та розкладає їх на різні карти об'єктів, такі як краї, лінії, криві тощо.

До витягнутих карт об'єктів по всій мережі застосовуються кілька процесів. Витягнуті карти об'єктів з останнього шару (як правило, повністю підключеного шару) класифікуються на вихідні класи за допомогою класифікатора, такого як класифікатор SoftMax [14].

Типова згортка нейронна мережа складається з наступних елементів:

- кількість згорткових і повнозв'язаних шарів, де виконується більшість операцій;
- об'єднання шарів, які використовуються для уникнення

переобладнання; шар класифікації, щоб класифікувати кінцеві результати за класами;

- інші шари за потребою.

Шар в CNN складається з тривимірних об'ємів нейронів, як показано на рисунку 1.1 (ширина, висота і глибина, а слово глибина відноситься до того, що називається «картами функцій або картами активації», а не кількістю шарів у CNN).

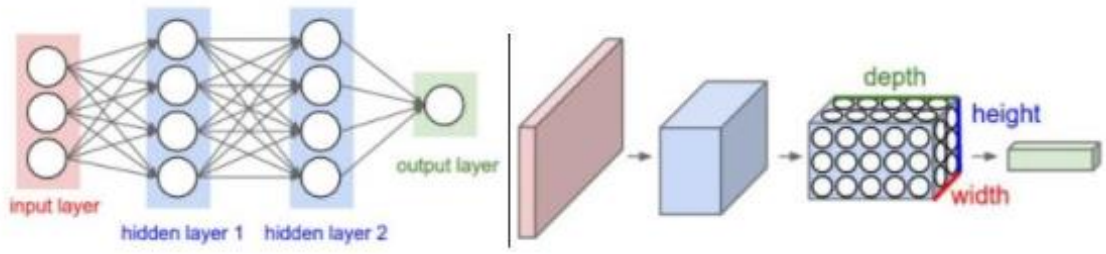

Рисунок 1.1 – Модель згорткової нейронної мережі

1.3.1 Шари CNN

Згортковий шар розглядається як основний будівельний блок CNN, і він містить більшість операцій у моделі CNN. Згортковий шар, по суті, виконує математичну операцію під назвою згортка, яка включає тривимірне множення накопичення (MACC) операції. На рисунку 1.2 показано ядро/фільтр (вибір значень фільтра залежить від призначених функцій, а вхідні зображення повинні ділитися на 2 багато разів) ваг, які помножуються на набір вхідних даних (приймальна область), а зважені вхідні дані підсумовуються разом.

Зміщення, значення якого зазвичай одиниця, додається до підсумованих зважених вхідних даних, щоб забезпечити спрацьовування нейронів. Функція активації, така як випрямлена лінійна одиниця (ReLU), застосовується до накопиченої суми, щоб внести нелінійність і обмежити

вихід до розумного діапазону. Результати функції активації передаються відповідним нейронам наступного шару.

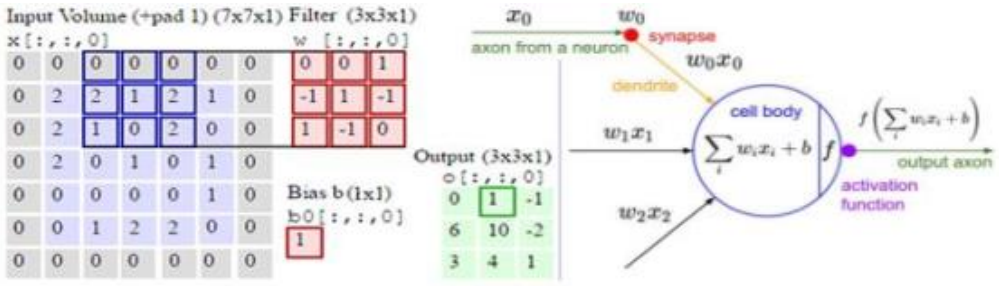

Рисунок 1.2 – Згортковий шар

Три гіперпараметри контролюють розмір вихідних даних: глибина, Stride, Zero-adding, де крок – це швидкість ковзання, з якою фільтр ковзає (найпоширенішим значенням слайда є 1, коли фільтр переміщується на один піксель вправо за раз), а нульове заповнення – це процес, який застосовується до межі вхідних даних, щоб допомогти контролювати просторовий розмір виводу та зберегти інформацію на межі. У CNN загальна кількість параметрів (ваги та зміщення) менша, ніж у звичайних мережах із прямим зв’язком, тоді як не всі нейрони підключені один до одного.

Загальна кількість параметрів зменшується, оскільки застосовуються рецептори локального поля (локальна зв’язність), де нейрони підключаються лише до відповідного локального поля без необхідності підключатися до всіх входів (пікселів на зображенні чи нейронів на карті ознак).

Рецептор поля є загальним для всіх нейронів наступного шару. Наприклад, якщо є N прихованих шарів і рецепторне поле 5×5×3, то загальне кількість параметрів дорівнює - (5×5×3×N) + (N зміщень). На рисунку 3.3 показано вхідне зображення, згорнуте за допомогою фільтра (зеленим), що створює відповідні карти активації (синім кольором). Інша карта активації (зелена) також була створена іншим фільтром з таким же розмір, але різні значення фільтра.

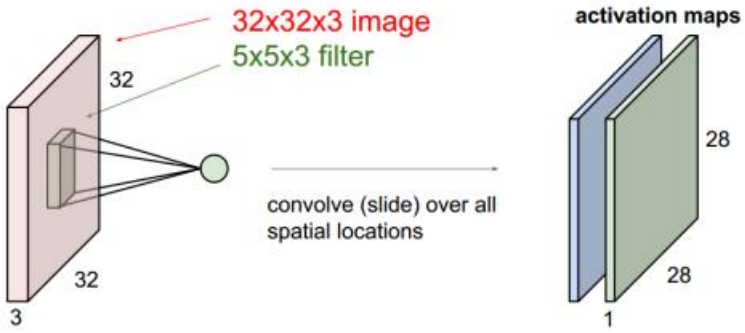

Рисунок 1.3 – CNN 5 x 5

Функція активації застосовується до кожного вхідного пікселя, щоб забезпечити нелінійність мережі, а також позбутися від непотрібної інформації. Серед різних функцій активації, Sigmoid, Tanh і ReLU є найбільш часто використовуваними функціями активації. Функція активації застосовується до кожного вхідного пікселя для забезпечення нелінійності в мережах, а також для позбавлення від непотрібної інформації. Серед різних функцій активації, Sigmoid, Tanh і ReLU є найбільш часто використовуваними функціями активації.

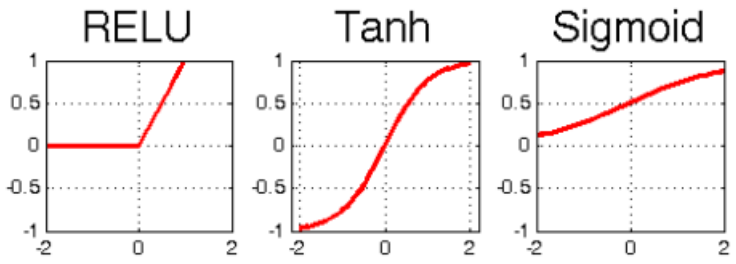

Рисунок 1.4 – Функції активації

Нормалізація або нормалізація локальної реакції (LRN) реалізує латеральне гальмування [16], демпфуючи відповіді, які є рівномірно великими в будь-якому локальному околиці. Перед відправкою зважених входів (виходів) згортки до нелінійності, рівень нормалізації нормалізує виходи залежно від сусідніх нейронів, щоб допомогти привести вхідні дані

до ReLU до загального масштабу. Рівень LRN був введений в архітектуру AlexNet [8], але менш поширений в останніх CNN.

1.3.2 Прихований шар

Важливість об’єднання шарів полягає в тому, що вони запобігають переобладнанню CNN [17]. По суті, просторове об’єднання є формою нелінійної підвибірки, яка використовується для зменшення розмірів об’єктів у міру заглиблення в мережу. Існує кілька методів об’єднання в пул, і найпоширенішими є середнє та максимальне об’єднання.

При максимальному об’єднанні набір нейронів відбирається на основі розміру фільтра об’єднання, тоді як максимальне значення нейрона в цьому фільтрі передається відповідному нейрону наступного шару, а решта нейронів вилучається. При об’єднанні середнього об’єднання значення, яке передається відповідному нейрону наступного шару, є середнім значенням усіх нейронів у використаному фільтрі, як показано на рисунку 1.5.

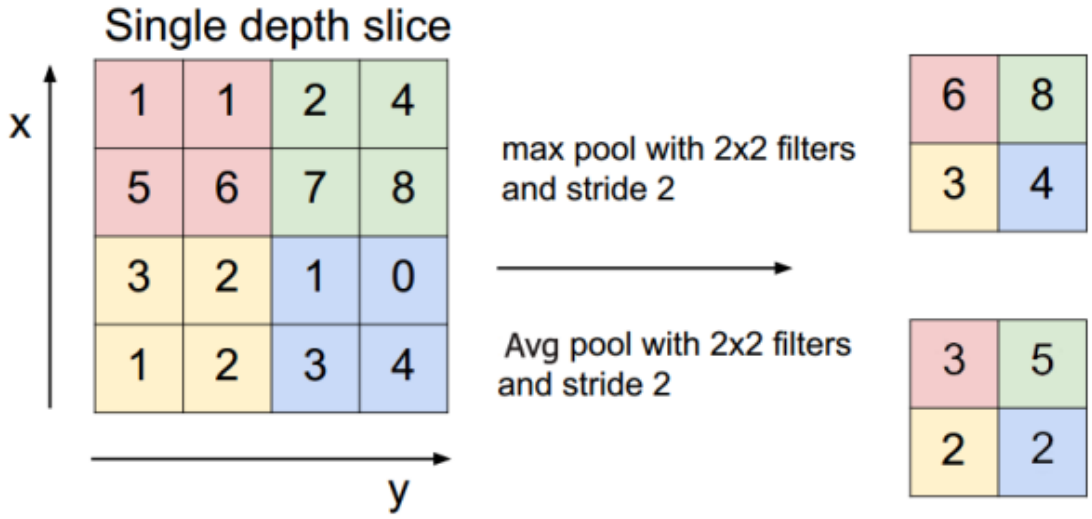

Рисунок 1.5 – Середня та максимальна продуктивність об’єднання для фільтра 2 x 2 із кроком у два

1.3.3 Повністю пов’язаний шар

Повністю зв’язаний (FC) шар зазвичай стоїть перед шаром класифікації, і він містить найбільшу кількість параметрів, оскільки кожен нейрон цього шару з’єднаний з усіма нейронами попереднього шару, а параметри транслюються на зв’язках між цими нейронами. Вхідні дані в цьому шарі помножуються на відповідні ваги, відповідно додаються зміщення, а нелінійність застосовується так само, як і згорткові шари.

Рівень випадання: метод, який використовується для уникнення переобладнання у великих CNN. Під час навчання цей рівень випадковим чином скидає вибирається відсоток своїх з'єднань, щоб запобігти тому, щоб мережа навчалася дуже точним відображенням, і змусити вбудовувати деяку надмірність у вивчені ваги.

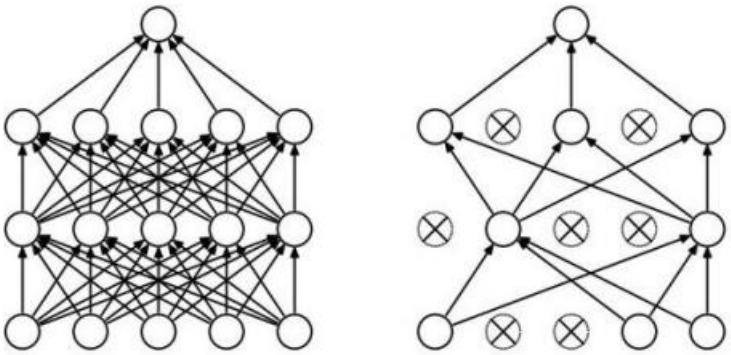

Рисунок 1.6 – Рівень випадання

Рівень класифікації: це останній рівень у CNN, і його основна функціональність полягає в тому, щоб класифікувати остаточний результат попереднього рівня на певні класи. На цьому рівні для виконання процесу класифікації використовується функція класифікації, наприклад SoftMax. В основному, класифікатор SoftMax перетворює вихідні оцінки класу zi нелінійності в попередньому шарі на ймовірність Pi в діапазоні (0, 1) відповідно до $Pi = ezi\sum k$ загальних результатів до $ezi\sum k$ до Pi в діапазоні (0,

Верхні ймовірності з класифікатора SoftMax потім порівнюються з фактичними мітками доступних класів, отже, оцінюють точність моделі.

1.4 Топології згорткових нейронних мереж

Процес навчання з використанням GPU дозволив значно підвищити кількість параметрів мережі. Отримана мережа отримала назву AlexNet (рисунок 1.7). Модель AlexNet стала переможцем конкурсу ILSVRC у 2012 році і була розроблена Алексом Крижевським та співавторами [8]. Архітектура містить загалом 5 згорткових шарів з трьома шарами об’єднання та двома повнозв’язаними шарами. AlexNet має близько 60 мільйонів параметрів і виконує приблизно 1,1 мільярда операцій MACC за один перехід вперед. У цій мережі прийняті класифікатор SoftMax і методика випадання, а також у цій архітектурі був запропонований рівень LRN. Ця модель досягла рейтингу помилок у топ-5 15,4%

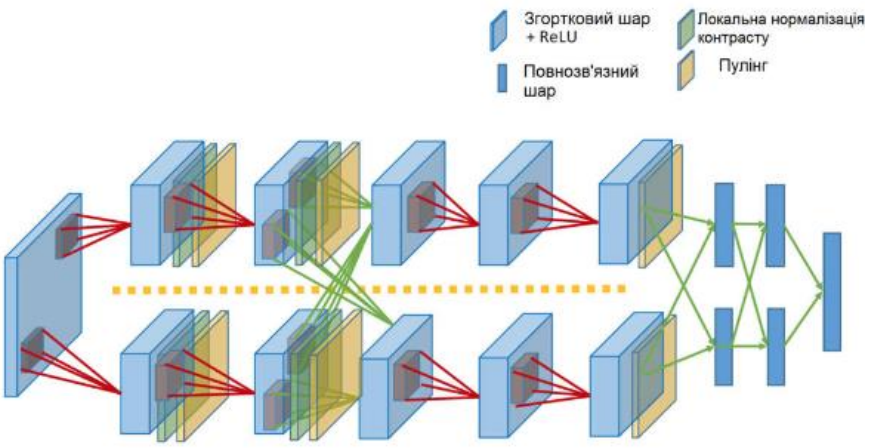

Рисунок 1.7 – AlexNet

З точки зору топології мережі це майже той же LeNet (рисунок 1.8), просто збільшений в тисячу разів. Додалися ще кілька згорткових шарів, а розмір ядер згортки зменшується від входу мережі до виходу. Серед ранніх

моделей CNN LeNet-5 був запропонований Yann LeCun et al [18] у 1998 році для виконання розпізнавання цифр на зображеннях, що містять цифри. Модель містила лише два згорткових шари та два об’єднаних шари разом із двома повнозв’язаними шарами.

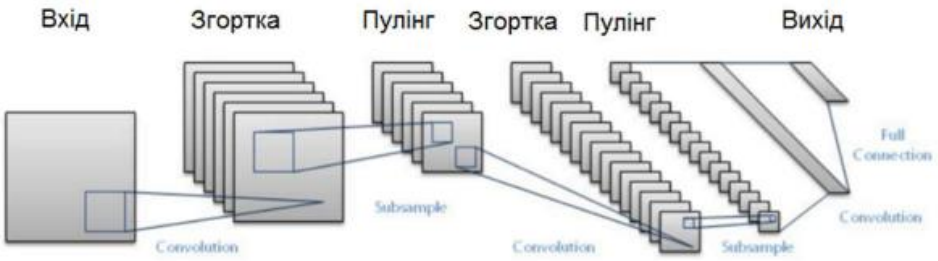

Рисунок 1.8 – LeNet

В 2013 було створено нову модель згорткових нейронних мереж МеттьюЗейлером та Робом Фергюсом з NYU. Названа ZF Net (рисунок 1.9), ця модель досягла 11,2% порогу похибки. Ця архітектура була більш точно налаштована ніж попередня структура AlexNet, але все-таки розроблена з деякими ключовими ідеями щодо підвищення продуктивності.

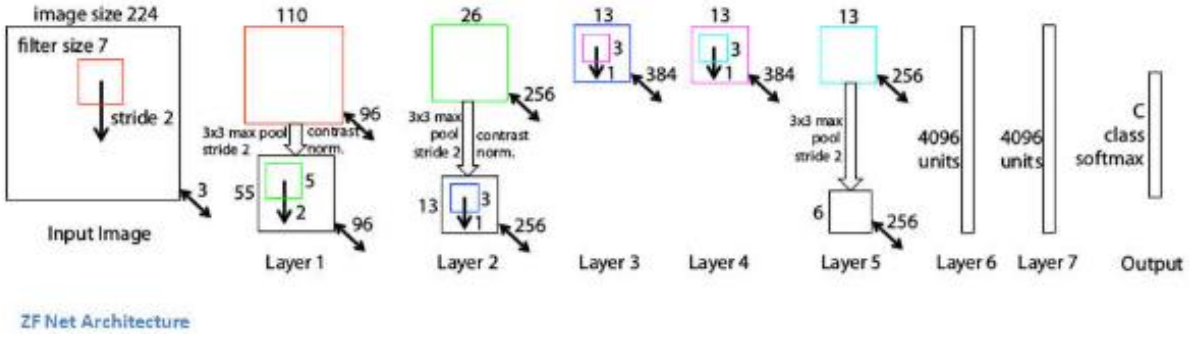

Рисунок 1.9 – ZF Net

GoogLeNet (рисунок 1.10) була однією з перших архітектур CNN, які дійсно відхилялися від загального підходу простого об’єднання згорткових і субдискретизуючих шарів в послідовній структурі.

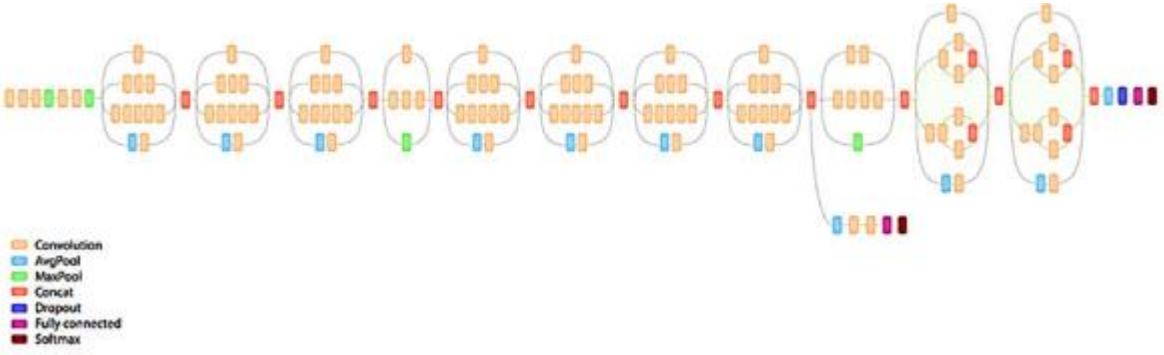

Рисунок 1.10 – GoogLeNet

За останні пару років було представлено багато топологій CNN для їх точності та продуктивності. Класифікація зображень є цікавим і значно найскладнішим завданням комп'ютерного зору, де розв'язувач повинен або позначати зображення, ідентифікувати об'єкти на зображеннях, або групувати зображення з однаковими характеристиками в подібні групи. Здається, що це завдання ефективно вирішується за допомогою згорткових нейронних мереж. Щорічно проводиться конкурс із класифікації зображень під назвою ImageNet Large Scale Visual Recognition Challenge (ILSVRC), де учасники змагаються з розробленими ними алгоритмами CNN за класифікацію зображень із бази даних ImageNet.

База даних ImageNet складається з понад 14 мільйонів зображень, кожне з яких позначено відповідним класом. Навчальний набір ILSVRC містить близько 1,2 мільйона зображень для 1000 різних класів. Таблиця I показує підсумок деяких алгоритмів CNN, які брали участь у ImageNet Challenge.

Розроблено Мін Ліном та співавт. [19] у 2013 році архітектура NIN складається з невеликого багатошарового персептрона, що працює як згорткові фільтри, які ковзають по відповідному входу. У цій мережі середній пул прийнято в класифікаторі замість повністю підключених шарів, отже, мережа має меншу кількість параметрів. Цю модель можна навчати на наборі даних ImageNet і досягати рівня точності Alex-Net [20].

Модель групи візуальної геометрії (VGG), розроблена Кареном Симоняном та Ендрю Зіссерманом [21], стала переможцем конкурсу ImageNet у 2014 році. Найглибша запропонована модель містила 19 згорткових шарів, тобто приблизно в 4 рази глибиною, ніж AlexNet. У згорткових шарах використовувалися виключно фільтри згортки 3 × 3 і 2 × 2 фільтри максимального об'єднання. Ця мережа має дуже велику кількість параметрів, близько 138 мільйонів, і один перехід вперед вимагає приблизно 16 мільярдів операцій MACC.

Оскільки тенденція щодо моделей CNN полягала в розвитку більш глибоких мереж, Крістіан Сегеді та інші [22] запропонували 22-шарову глибоку модель CNN під назвою GoogLeNet-5, яка виграла конкурс ImageNet у 2015 році з частотою помилок у топ-5 6,7%. Модель має лише 1,2 мільйона параметрів, що становить приблизно 0,86% параметрів VGG. Масове зменшення параметрів призвело до більш складної архітектури, яка використовує так звані початкові модулі. Початковий модуль є в основному це підархітектура мережа в мережі, яка використовує згортковий шар 1 × 1 для зменшення кількості вхідних каналів.

Запропонована Kaiming та іншими [23] з дослідження Microsoft, залишкова мережа (ResNet) з глибиною 152 виграла завдання ImageNet, досягнувши частоти помилок топ-5 3,6%. В ній були 152 шари, це означає важку проблему навчання. Щоб подолати цю проблему, дослідники включили обхід кожної партії наступних згорткових шарів. Цю топологію можна розглядати як $y = F(x) + x$, де мережа має вивчати лише функцію залишку F(x).

2 ПРОГРАМОВАНІ ЛОГІЧНИ МАСИВИ

Польові програмовані логічні масиви (FPGA) – це готові напівпровідникові пристрої, які складаються з 2D-масивів конфігурованих логічних блоків (CLB, або логічних фрагментів), які з’єднані за допомогою програмованої логіки. Взаємозв'язок нагадує мережу пучків проводів (рисунок 2.1), які проходять горизонтально і вертикально між логічними блоками з коробками комутаторів (матрицями комутаторів) на кожному перетині між горизонтальним і вертикальним пучками. Логічні блоки, блоки з фіксованою функцією, а також з'єднання програмуються в електронному вигляді шляхом запису бітового потоку конфігурації в пристрій для реалізації будь-якого цифрового дизайну.

Конфігурація зазвичай зберігається в осередках пам’яті SRAM, і FPGA можна перепрограмувати багато разів [24]. Перша FPGA на основі статичної пам’яті (SRAM) була запропонована Уолстромом у 1967 році. Ця архітектура дозволяла конфігурувати як логіку, так і конфігурацію взаємозв’язку за допомогою потоку бітів конфігурації. Перша комерційна FPGA сучасної епохи була представлена Xilinx у 1984 році. Вона містила масиви конфігурованих логічних блоків та входів/виходів (I/Os). Сучасні покоління FPGA високого класу мають сотні тисяч конфігурованих логічних блоків, і вони включають велику кількість загартованих функціональних блоків, які дозволяють швидко та ефективно реалізовувати загальні функції.

Програмовані логічні блоки в FPGA використовуються для забезпечення основних обчислювальних і запам'ятовуючих елементів, що використовуються в цифрових системах. Типовий базовий логічний елемент містить певну форму програмованої комбінаційної логіки, тригер або фіксатор, а також деяку логіку швидкого перенесення для зменшення вартість площі та затримки. Крім того, сучасні FPGA містять гетерогенну суміш різних блоків, деякі з яких можна використовувати для певних

функцій, таких як виділені блоки пам'яті, помножувачі (блоки DPS) або мультиплексори.

Незважаючи на те, що найкраще адаптувати алгоритми до паралельної природи графічних процесорів, архітектура FPGA спеціально розроблена для застосування, де користувацькі механізми обробки можуть бути створені з використанням програмованих логічних блоків для задоволення потреб алгоритму.

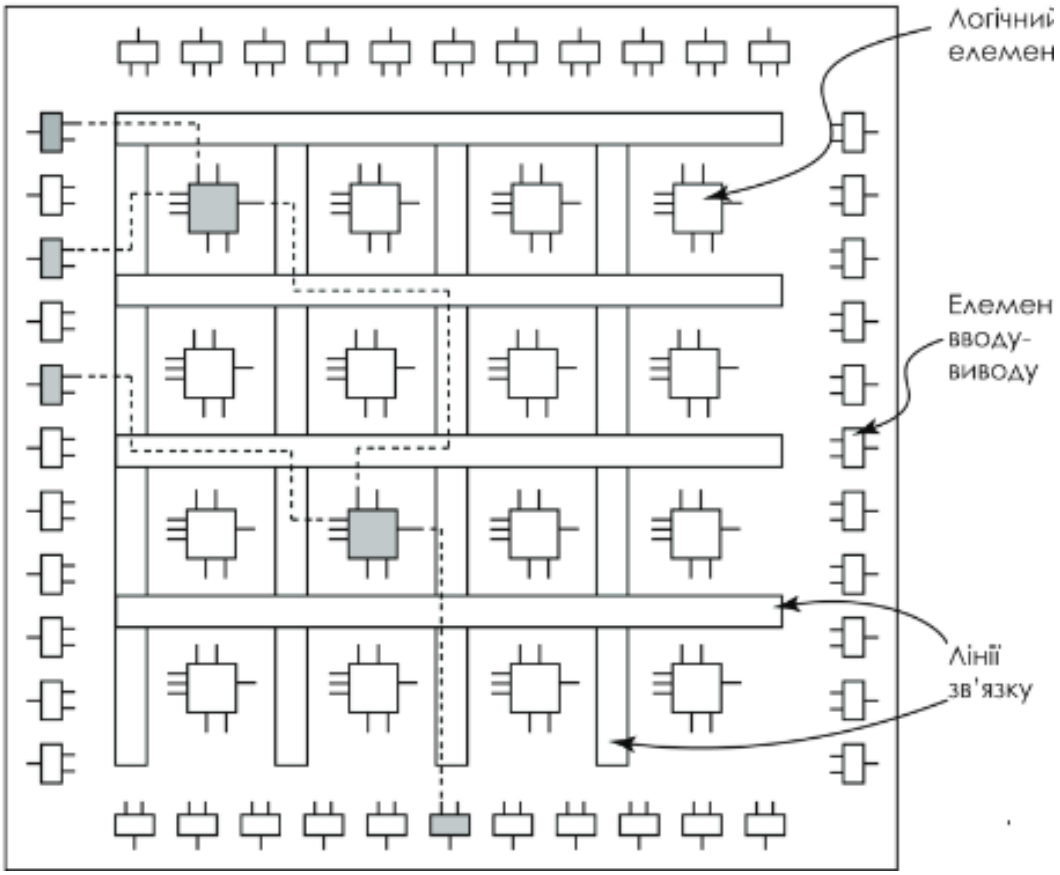

Рисунок 2.1 – Класична FPGA

Іншими словами, при розробці методів машинного навчання для FPGA приділяється менше уваги адаптації алгоритмів. Це дає більше свободи для вивчення оптимізації рівня алгоритму. Продуктивність розробки FPGA можна додатково підвищити, використовуючи формати даних із фіксованою точкою або точністю до половини точки. Оптимізації та методи, які

вимагають багатьох складних низькорівневих апаратних операцій, не можуть бути легко реалізовані на мовах програмного забезпечення високого рівня, тому більш привабливим є розгляд реалізації FPGA.

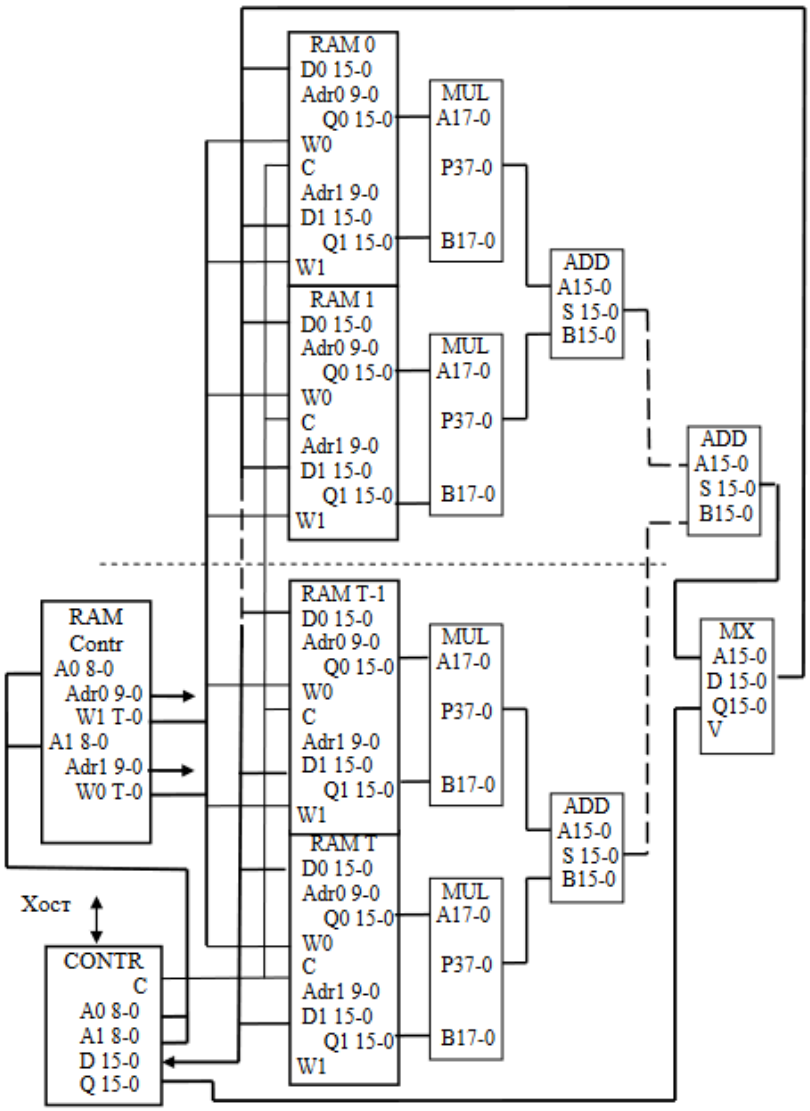

Рисунок 2.2 – Структурна схема нейромережі на FPGA з паралельним обчисленням ваг

Крім того, на додаток до адаптивності реалізації FPGA, FPGA є реконфігурованими та гнучкими, що пропонують широкий спектр моделей згорткових нейронних мереж, які можна реалізувати на одному чіпі без додаткових витрат на проектування, як це відбувається в ASIC (рисунок 2.2).

Поки що FPGA є найбільш підходящою платформою для розробляємого методу, але недоліком реалізацій на основі FPGA є те, що розробникам доводиться використовувати мови опису обладнання для виконання їхньої реалізації, які не дуже зручні для програмування та вимагають пристойного досвіду програмування.

Для лінійних базисних функцій реалізація розрахунків можлива з паралельною обробкою всіх нейронів чергового шару і послідовним накопиченням зважених сум ваг для кожного з них. Таке рішення передбачає застосування DSPIP-ядер, що входять до складу останніх серій FPGA Xilinx. Це забезпечує гнучке управління областями аргументів карт ознак, але є досить витратним.

Варіант з паралельним обчисленням входу функції активації (послідовнодля кожного нейрона чергового шару мережі) ефективно реалізується в більш доступних ПЛІС типу Spartan3. На структурній схемі на рисунку цей варіант показаний пірамідальним суматором ADD зважених за допомогою помножувачів MUL значень функцій активації всіх нейронів попереднього шару. Значення функцій активації і вагові коефіцієнти зберігаються в двопортовій блочній пам'яті RAM.

Програмовані логічні блоки в FPGA використовуються для забезпечення основних обчислювальних і запам'ятовуючих елементів, що використовуються в цифрових системах. Типовий базовий логічний елемент містить певну форму програмованої комбінаційної логіки, тригер або фіксатор, а також деяку логіку швидкого перенесення для зменшення площі та вартості затримки. Крім того, сучасні FPGA містять гетерогенну суміш різних блоків, деякі з яких можна використовувати для певних функцій, таких як виділені блоки пам’яті, помножувачі (блоки DPS) або мультиплексори [25].

З'єднання між логічними блоками та блоками вводу-виводу забезпечуються за допомогою програмованої маршрутизації в FPGA. Взаємоз'єднання складається з прохідних транзисторів, буферів із трьома

станами та мультиплексорів, які досягають бажаного з'єднання. Як правило, мультиплексори та проходні транзистори використовуються в логічному кластері для з’єднання логічних елементів разом, тоді як усі три використовуються для більш глобальних структур маршрутизації. Існує кілька глобальних структур маршрутизації, які використовувалися в FPGA, таких як острівні архітектури, стільникові, на основі шини та зареєстровані архітектури.

Логічні блоки та архітектури маршрутизації взаємодіють із зовнішніми компонентами FPGA за допомогою панелей введення/виводу або програмованого введення/виводу. Панель вводу/виводу та допоміжні логічні схеми утворюють важливі компоненти, які називають осередками введення/виводу. Через різницю стандартів напруги живлення та опорної напруги проектування програмованих блоків вводу/виводу є досить складним. Вибір підтримуваного стандарту є одним з найважливіших рішень у проектуванні архітектури вводу/виводу. Підтримка великої кількості стандартів може значно збільшити площу кремнію, необхідну для елементів вводу/виводу [25].

Архітектура FPGA була розроблена з плином часу шляхом додавання більш спеціалізованих програмованих функціональних блоків, таких як вбудована пам'ять (блок RAM), арифметична логіка (ALU), помножувачі (MUX), цифрові сигнальні процесори (DSP48) і вбудовані мікропроцесори. Це зробило FPGA гетерогенними платформами.

Перевагою систем на основі FPGA перед традиційними системами на базі процесорів, такими як настільні комп’ютери, смартфони та графічні процесори, є наявність вільно програмованих логічних блоків загального призначення. FPGA можуть бути організовані у високопродуктивні спеціалізовані прискорювачі для дуже конкретних завдань, що призводить до покращення швидкості обробки, більшої пропускної здатності. У порівнянні з графічними процесорами, FPGA вважаються дуже енергоефективними пристроями, де вони краще підходять для програм на базі мобільних

пристроїв. Ці переваги пов’язані з ціною збільшення складності та зниження швидкості під час розробки, коли дизайнерам необхідно ретельно враховувати доступні апаратні ресурси та ефективне відображення цільових алгоритмів на архітектурі FPGA. Крім того, FPGA перевищують обчислювальну потужність цифрових сигнальних процесорів (DSP), порушуючи парадигму послідовного виконання і досягаючи більше за такт, де вони повністю використовують переваги апаратного паралелізму. Управління входами та виходами (I/O) на апаратному рівні забезпечує швидший час відгуку та спеціалізовані функціональні можливості, які точно відповідають вимогам програми. FPGA зазвичай не використовують операційні системи, які фактично мінімізують проблеми з надійністю завдяки справжньому паралельному виконанню та детермінованим апаратним забезпеченням, призначеним для кожного завдання [26].

Спеціальні інтегральні схеми (ASIC) — це напівпровідникові пристрої, спеціально розроблені на замовлення. На відміну від FPGA, ASIC не мають накладних витрат площі або часу, які можуть бути викликані логікою конфігурації та загальними взаємоз'єднаннями, що призводить до найшвидших, найбільш енергоефективних і найменших систем. Однак складні процеси виготовлення для ASIC призводять до дуже тривалого та складного циклу розробки та дуже високих одноразових початкових витрат на інженерію, які вимагають вперше правильної методології проектування та дуже широкої перевірки дизайну. Тому ASIC здебільшого підходять для дуже великого обсягу, чутливих до витрат додатків, де одноразові витрати на інженерію та виготовлення можуть бути розподілені між великою кількістю пристроїв.

FPGA з їх можливостями перепрограмування краще підходять для створення прототипів і коротких циклів розробки, де концепції можна тестувати і перевіряти на апаратному рівні, не проходячи через довгий процес виготовлення спеціального дизайну ASIC.

Мікросхеми FPGA можна оновлювати на місцях і не вимагають часу та

витрат, пов’язаних з перепроектуванням ASIC. Цифровий наприклад, протоколи зв’язку мають специфікації, які можуть змінюватися з часом, а інтерфейси на основі ASIC можуть викликати проблеми з обслуговуванням і сумісністю. Будучи реконфігурованими, FPGA можуть не відставати від майбутніх модифікацій, які можуть знадобитися [25,27].

3 МЕТОДИ АПАРАТНОГО ПРИСКОРЕННЯ ДЛЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Згорткові нейронні мережі проходять кілька етапів, перш ніж вони будуть реалізовані в апаратному забезпеченні для виконання конкретних завдань для певної програми. У попередньому розділі ми проілюстрували різні топології CNN, де всі вони по суті базуються на одних і тих же концепціях проектування типової структури CNN. Варіації між вищезгаданими топологіями обумовлені параметрами, які контролюють поведінку мережі. У цьому розділі буде представлено навчання CNN, оптимізація та впровадження.

3.1 Тренування згорткових нейронних мереж

Щоб згортка нейронна мережа виконувала класифікацію зображень певного набору даних, мережу потрібно навчити виконувати класифікацію для цього набору даних. CNN, як правило, навчаються за допомогою алгоритму зворотного поширення [4]. Зазвичай це вирішується за допомогою стохастичного градієнтного спуску (SGD) [12]. SGD є, мабуть, найбільш часто використовуваною процедурою оптимізації для навчання глибоких нейронних мереж [13], в якій ваги мережі переміщуються вздовж від’ємного градієнта функції продуктивності.

Термін зворотне поширення відноситься до способу обчислення градієнта для нелінійних багатошарових мереж. Алгоритм поширює помилку, яка обчислюється як різниця між результатом прямого проходу і очікуваний вихід по всій мережі, щоб налаштувати значення ваг, щоб мінімізувати помилку. Зазвичай моделі CNN розробляються з використанням бібліотек, наданих відомими фреймворками, такими як TensorFlow, Keras або Caffe. Навчання в основному здійснюється за допомогою графічних

процесорів оскільки реалізувати CNN на графічному процесорі відносно легко, але графічні процесори забезпечують дуже високу швидкість навчання, хоча на навчання великої CNN, наприклад AlexNet [8], може знадобитися до тижня. Для швидкого та легкого початку навчання CNN рекомендується виконувати навчальний процес TensorFlow [28], де можна отримати докладний посібник із розробки та навчання CNN.

CNN – це природний паралельний алгоритм, і щоб у повній мірі скористатися цими природними явищами, краще використовувати доступний паралельність настільки, наскільки це необхідно для цільової програми. Паралелізм у CNN можна пояснити наступним чином:

- паралелізм всередині згортки, згортка матриці $n \times n$ за допомогою фільтра $m \times m$ може бути обчислена паралельно за один такт;

- паралелізм в об'єднанні, операцію об'єднання можна розпаралелювати шляхом підвибірки всіх окремих підматриць одночасно;

- паралелізм у вихідних картах об’єктів, вилучені карти об’єктів повністю незалежні одна від одної, отже, усі вони можуть обчислюватися паралельно.

Іншими словами, якщо ми дивимося на X-об’єкти на зображенні, то можна запускати X-паралельні процеси для вилучення цих функцій; Паралельність у вхідних картах об’єктів, вхідні карти об’єктів з попередніх шарів можна обробляти паралельно, оскільки їх можна об’єднати, щоб отримати один вихідний результат. Іншою методологією оптимізації, яку можна взяти до уваги, є обмеження точності даних. Нижча точність може заощадити багато апаратних ресурсів, отже ефективне зниження точності моделі з огляду на відповідність вимогам програми може досягти величезної оптимізації. Раніше був досліджен вплив обмеженої точності представлення даних і обчислень на навчання нейронної мережі. У контексті низькоточних обчислень із фіксованою точкою, вони помітили, що схема округлення відіграє вирішальну роль у визначенні поведінки мережі під час навчання. Їхні результати показують, що глибокі мережі можна навчати,

використовуючи лише 16-бітове представлення чисел із фіксованою точкою при використанні стохастичного округлення, і майже не зазнають деградації.

Паралелізм у CNN можна пояснити наступним чином: паралелізм всередині згортки, згортка матриці $n \times n$ за допомогою фільтра $m \times m$ може бути обчислена паралельно за один такт. Паралелізм в об'єднанні, операцію об'єднання можна розпаралелювати шляхом підвибірки всіх окремих підматриць одночасно; Паралелізм у вихідних картах об’єктів, вилучені карти об’єктів повністю незалежні одна від одної, отже, усі вони можуть обчислюватися паралельно. Іншими словами, якщо ми дивимося на X-об’єкти на зображенні, то можна запускати X-паралельні процеси для вилучення цих функцій;.

Паралельність у вхідних картах об’єктів, вхідні карти об’єктів з попередніх шарів можна обробляти паралельно, оскільки їх можна об’єднати, щоб отримати один вихідний результат. Іншою методологією оптимізації, яку можна взяти до уваги, є обмеження точності даних. Нижча точність може заощадити багато апаратних ресурсів, отже ефективне зниження точності моделі з огляду на відповідність вимогам програми може досягти величезної оптимізації.

Згорткові нейронні мережі можуть бути реалізовані за допомогою наступних платформ: центральних процесорів загального призначення (GPCPU), графічних процесорів (GPU) і FPGA. GPCPU є найменш популярними платформами для запуску CNN, оскільки вони недостатньо використовують CNN. Згорткові нейронні мережі, природно, паралельні, і їх кінцеві додатки в основному є додатками на основі обробки зображень, тому наявність ЦП в цьому рівнянні взагалі не підходить, оскільки ЦП є послідовно заснованими елементами обробки, які не придатні для обробки зображень і не використовують переваги успадкування. паралелізм у CNN. Хоча процесори не є хорошими для обробки CNN, графічні процесори є найулюбленішими платформами для навчання CNN, і це, очевидно, тому, що з CNN GPU обробляють те, для чого вони насправді були створені. Однак

графічні процесори не є енергоефективними через високе споживання енергії.

Оскільки процесори не використовують переваги доступного паралелізму в CNN, а графічні процесори є енергонеефективними пристроями, FPGA вдається збалансувати це рівняння. FPGA є типом реконфігурованих пристроїв, які можуть бути розроблені відповідно до конкретних вимог проектування. Зазвичай FPGA використовуються як прискорювачі, а у випадку CNN вони, здається, чудово підходять, оскільки використовують переваги успадкованого паралельності в CNN із набагато меншим споживанням енергії, ніж у графічних процесорів. Життя не завжди може бути легким, і це стосується FPGA.

Щоб прискорити CNN на FPGA, розробник/дослідник повинен пройти дуже тривалий і напружений процес розробки, використовуючи мову опису обладнання (HDL), наприклад VHDL або Verilog. Розробка з використанням HDL може призвести до найбільш оптимізованої реалізації прискорювача; однак деякі дизайнери вважають за краще поступитися певною продуктивністю, щоб спростити впровадження та скоротити час розробки. Інструменти синтезу високого рівня, такі як HLS-Vivado, пропонують альтернативну методологію для впровадження апаратних прискорювачів, де вони замінюють мови опису обладнання мовами високого рівня, такими як C, C++ або SystemC. Синтез високого рівня та мова опису обладнання будуть детально пояснені в наступному розділі.

3.2 Методика апаратного прискорення

Пропонується впровадження інструменту генерації VHDL на основі FPGA для реалізації згорткових нейронних мереж. Для розробки використовується мова Java. Метод призначений для полегшення процесу апаратного прискорення моделей згорткових нейронних мереж з використанням FPGA шляхом параметризації реалізації цих моделей.

Інструмент має містити графічний інтерфейс користувача, за допомогою якого можливе налаштування своєї цільової моделі згорткової нейронної мережі, надавши специфікації моделі. Використання методу значно зменшить час розробки, необхідний для впровадження згорткової нейронної мережі, також знащить недоліки, що виникають через складність розробки мов опису апаратних засобів, і має пом’якшити недостатню оптимізацію, викликану інструментами синтезу високого рівня. Інструмент повинен бути оптимізован для створення модульної, масштабованої, реконфігурованої та паралельної реалізації моделей згорткових нейронних мереж.

4 РОЗРОБКА ІНСТРУМЕНТУ ГЕНЕРАЦІЇ VHDL ОПИСІВ

4.1 Мова опису VHDL

VHDL є однією з найпоширеніших мов опису обладнання, яка використовується для розробки апаратних схем на рівні передачі регістра (RTL). У VHDL розробники зазвичай вказують деталі свого алгоритму, використовуючи ряд паралельних процесів, які описують деяку комбінаційну логіку, а також основні арифметичні операції та регістри. Ці процеси керуються наростаючим і спадним фронтами тактового сигналу, і вони працюють з векторами двійкових сигналів і простими цілочисельними типами даних, отриманими з них. Процес перетворення алгоритму в процеси, логічні блоки та кінцеві автомати на рівні передачі регістра дуже тривалий, виснажливий і схильний до помилок. Оскільки подальші зміни складні та дорогі в цьому процесі, дизайнерам доводиться розглянути та прийняти багато дизайнерських рішень, перш ніж спробувати написати будь-який рядок коду.

Розробка з використанням HDL вимагає від дизайнера гідного досвіду, щоб зменшити витрати на зміни та забезпечити задовільні результати проектування. Крім того, складність розробки в HDL перешкоджає ітераційній оптимізації, вимагає великої інтуїції, щоб мати повністю оптимізовану та функціональну реалізацію алгоритмів. Таким чином, розробка з використанням HDL не дуже віддає перевагу багатьом дослідникам, особливо тим, хто з цим не знайомий. Це фактично робить FPGA менш привабливими для прискорення алгоритму. Крім того, CNN є масивним алгоритмом, і реалізація навіть моделі невеликого розміру, як-от LeNet-5 [18], може зайняти місяці, що робить реалізацію великомасштабної моделі, наприклад AlexNet [8], непрактичною та нездійсненною.

4.2 Інструмент генерації VHDL

Інструмент генерації VHDL (рисунок 4.1) , заснований на мові Java, який пропонує параметризовану реалізацію для досягнення наступного:

- по-перше, подолати бар’єри, створені мовами високого опису та обмеження інструментів HLS;
- по-друге, досягти високої продуктивності та уникнути недостатнього використання згорткових нейронних мереж;
- по-третє, значно скоротити цикл розробки та забезпечити легку й ітераційну оптимізацію.

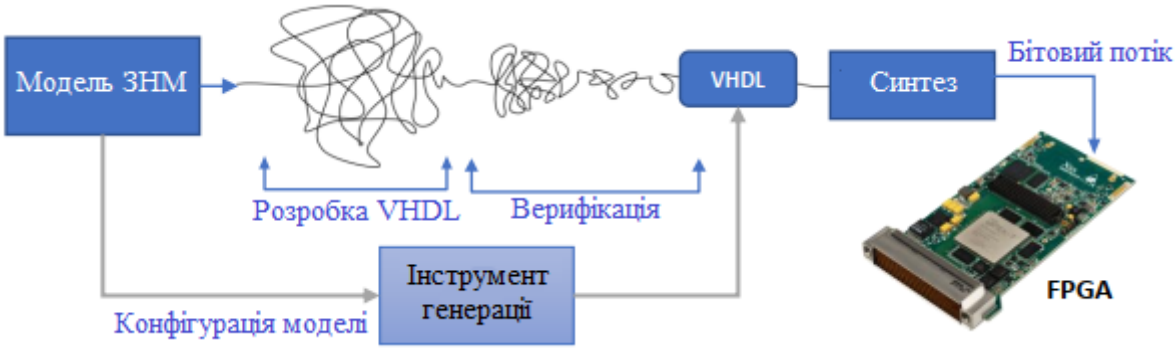

Рисунок 4.1 – Інструмент генерації VHDL

Користувач встановлює специфікації моделі за допомогою графічного інтерфейсу інструменту (ручна конфігурація) або за допомогою зовнішнього файлу специфікації/конфігурації. Після етапу налаштування користувачу пропонується перевірка своєї модель ЗНМ. Тільки після успішної перевірки користувач може перейти до другого процесу, де він може обробляти параметри (ваги) цільової моделі ЗНМ.

Після успішної конфігурації та перевірки моделі ЗНМ користувачу буде запропоновано надати параметри цільової моделі. У цьому процесі користувача просять надати представлення параметрів, а також бажану точність. Далі користувачу пропонується вибрати, що робити з параметрами

залежно від розміру цільової моделі.

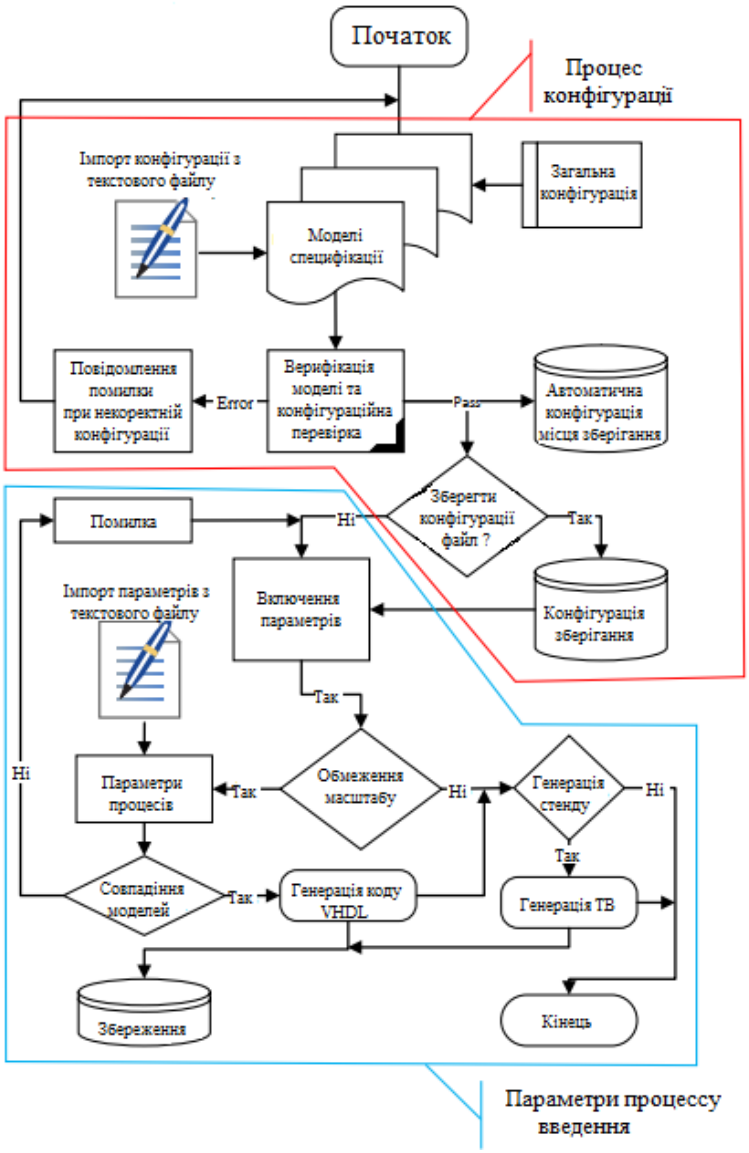

Рисунок 4.2 – Схема розробленого інструменту

Процес конфігурації: у цьому процесі користувачі встановлюють специфікації моделі за допомогою графічного інтерфейсу інструменту (ручна конфігурація) або за допомогою зовнішнього файлу специфікації/конфігурації. Після етапу налаштування користувачам пропонується перевірити свою модель CNN. Справді, існує набір правил, яких дизайнери повинні дотримуватися при створенні моделі CNN.

У VGT ці правила вбудовуються, щоб перевірити дійсність

налаштованої моделі. Тільки після успішної перевірки користувачі можуть перейти до другого процесу, де вони можуть обробляти параметри (ваги та зміщення) цільової моделі CNN. Процес включення параметрів: після успішної конфігурації та перевірки моделі CNN користувачам буде запропоновано надати параметри цільової моделі. У цьому процесі користувачів просять надати представлення параметрів, а також бажану точність.

Далі користувачам пропонується вибрати, що робити з параметрами залежно від розміру цільової моделі. Є два варіанти обробки параметрів моделі: або жорстко закодувати їх як частину програмованої логіки, і це стосується моделей невеликого розміру, або зберегти їх у зовнішньому джерелі пам’яті, і це стосується великомасштабних моделей. Після вибору бажаної точності, представлення параметрів і типу зберігання користувачам буде запропоновано включити параметри через графічний інтерфейс інструмента.

Перевірка запуску інструмента перевіряє імпортовані параметри, щоб перевірити, чи відповідають вони налаштованій моделі та представленню параметрів. Якщо вміст файлу параметрів порушує будь-яке з вищезгаданих правил, то файл не буде імпортовано/завантажено в інструмент, а користувачеві відобразиться інформаційне повідомлення із зазначенням помилок, які слід виправити. Після успішного включення параметрів модуль генерації коду вмикається, і користувачі можуть генерувати код VHDL для своєї моделі, а також тестовий стенд для симуляції та перевірки.

Після створення оптимізованого коду для своєї реалізації, моделюється згенерований код за допомогою наданого тестового стенду, щоб перевірити функціональність моделі, а також отримати уявлення про продуктивність моделі. Крім того, користувач може пропускати моделювання та безпосередньо синтезувати свою модель та генерувати бітовий потік для запуску на FPGA, як показано на рисунку 4.3.

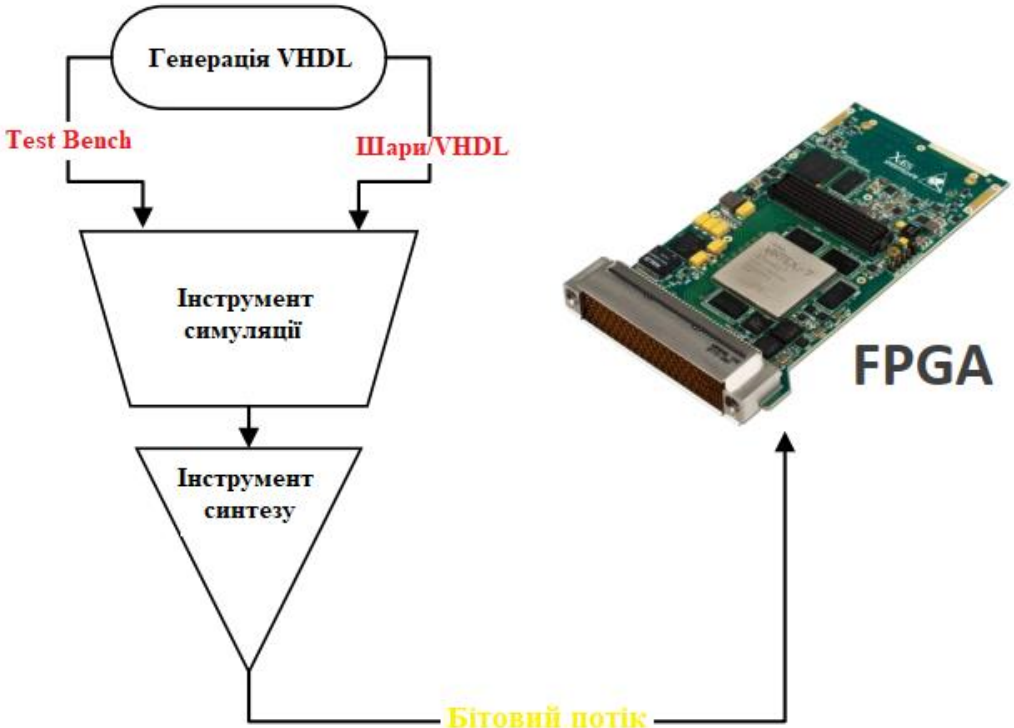

Рисунок 4.3 – Процес впровадження моделі згорткової нейронної мережі

Для невеликих моделей користувачам потрібно лише помістити зображення цільового набору даних у зовнішню пам’ять і встановити зв’язок з прискорювачем, не турбуючись про ваги та зміщення, оскільки вони жорстко закодовані як частина програмованої логіки. Для великомасштабних моделей користувачам потрібно буде зберігати як параметри, так і зображення цільового набору даних у відповідному джерелі пам’яті, наприклад, у зовнішній пам’яті або в зовнішній пам’яті та на мікросхемі. Наразі інструмент генерує код лише для шарів CNN, і користувачі повинні подбати про зв’язок прискорювача та пам’яті для завантаження/передавання зображень і параметрів.

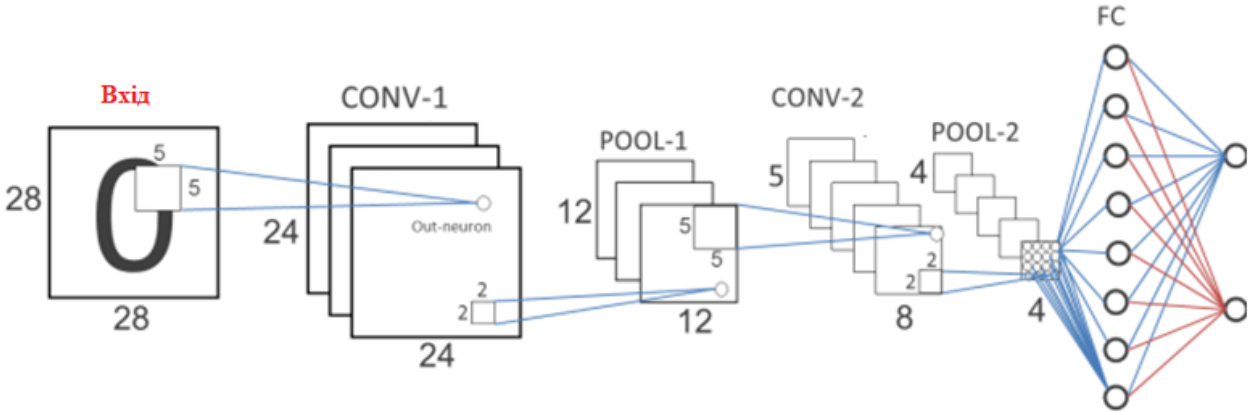

Рисунок 4.4 – Тестова модель згорткової нейронної мережі

Цільова модель використовується для класифікації зображень рукописних 0 і 1. Вхідне зображення має відтінки сірого розміру 28 × 28. У першому згортковому шарі (Conv-1) витягуються три різні карти ознак шляхом згортання вхідного зображення з ядрами/фільтрами розміром 5 × 5. Крок згорткового фільтра дорівнює одиниці, до вхідного зображення не застосовується заповнення, а застосована функція активації. Вихід шару PYV передається до першого шару об’єднання (Pool-1), де застосовується функція максимального об’єднання (max-pool) за допомогою розміру фільтра 2 × 2. В основному, в max-pool максимальне значення нейрона в фільтрі передається до відповідного нейрона в наступному шарі, а решта нейронів відкидається.

Другий згортковий шар приймає три вхідні карти розміром 12 × 12 і витягує 5 нових карт об’єктів для кожної вхідної карти об’єктів. Conv-2 використовує ядра розміром 5 × 5 з розміром кроку один і без заповнення. Результатом Conv-2 є 5 карт об’єктів розміром 8 × 8.

Повністю зв’язаний шар (FC) подібний до згорткового, але згортка замінена на множення матриці. Рівень FC складає більшість параметрів/з'єднань у мережі, які є більшими, ніж загальна кількість параметрів для обох згорткових рівнів разом узятих. У цьому шарі витягується 8 карт об’єктів для кожної вхідної карти об’єктів з другого шару об’єднання. Загальну кількість параметрів у шарі FC можна розрахувати, як

показано у рівнянні. Використовується функція активації ReLU.

VGT містить три етапи конфігурації, які користувач повинен пройти, щоб створити повний код VHDL для моделі CNN. У цьому підрозділі ми налаштуємо модель VGTEST за допомогою графічного інтерфейсу користувача VGT.

Блок конфігурації моделі У цьому блоці користувачі можуть налаштувати свою цільову мережу CNN вручну за допомогою графічного інтерфейсу VGT або завантажити конфігурацію попередньо налаштованої моделі із зовнішнього текстового файлу, як показано на рисунку 4.4.

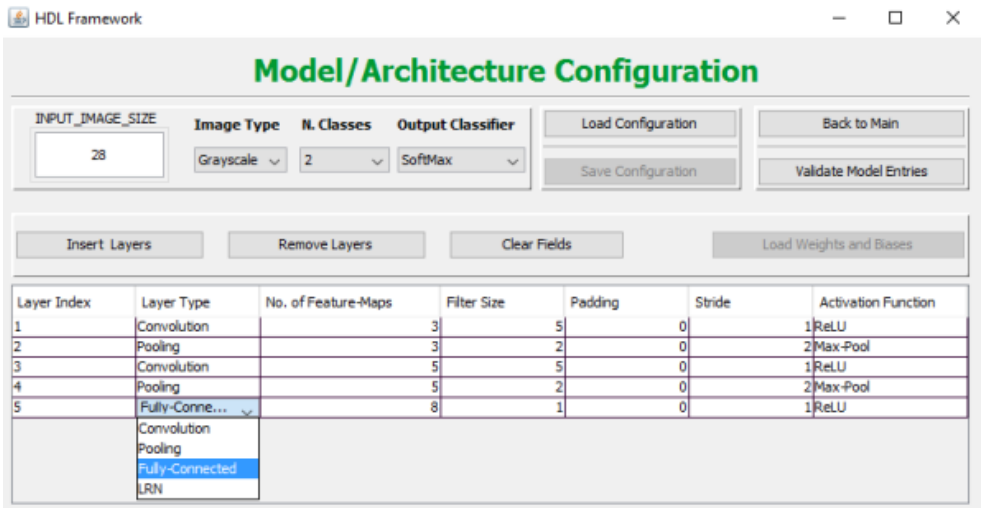

Рисунок 4.5 – Архітектура моделі

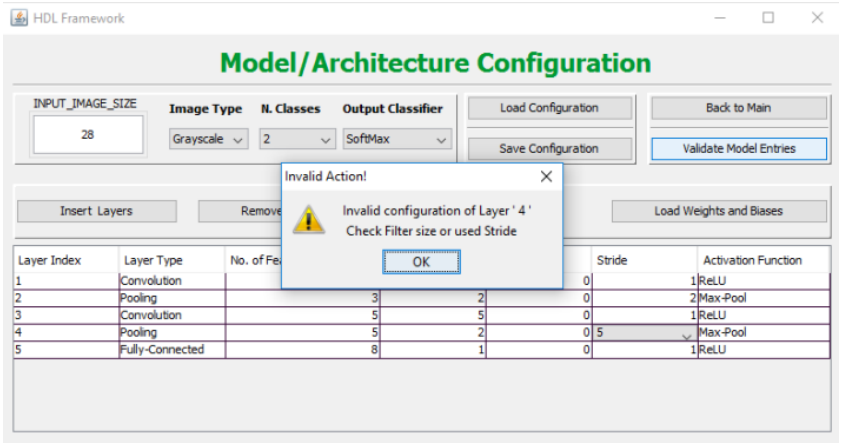

Рисунок 4.6 – Архітектура моделі

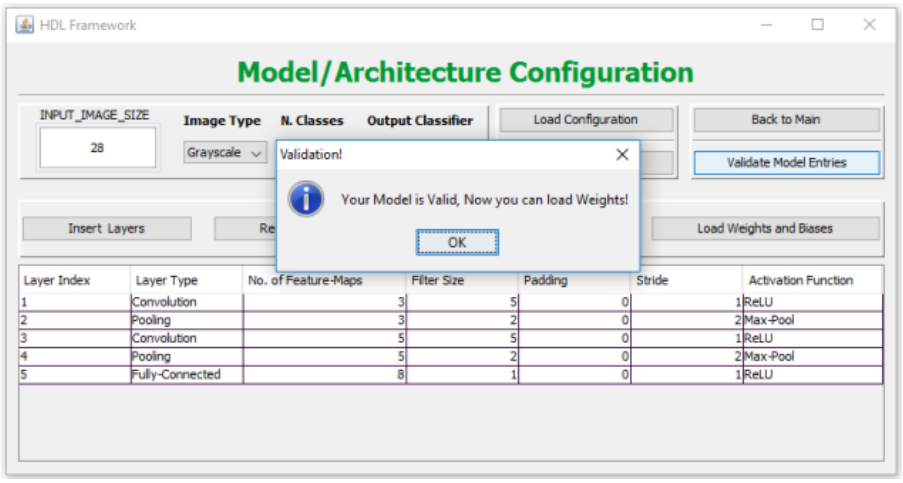

Рисунок 4.7 – Архітектура моделі

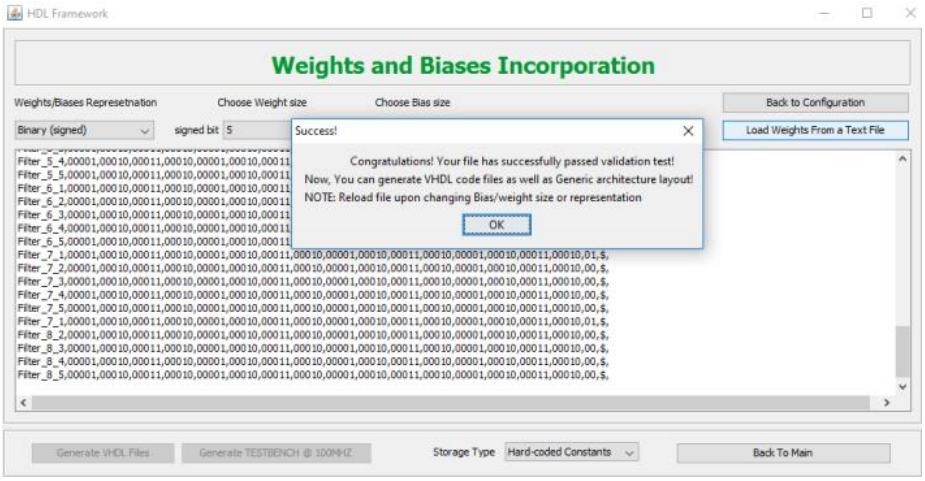

Рисунок 4.8 – Архітектура моделі

Етапи реалізації моделі представлені на рисунках 4.5-4.8. У разі невдалого включення параметрів із зовнішнього файлу користувачеві буде запропоновано повідомлення про помилку із зазначенням того, які виправлення йому слід зробити, щоб продовжити.

На рисунку 4.8 показані деталі архітектури апаратного забезпечення елемента обробки в згортковому шарі для прикладу фільтра розміром 3×3. PE можна масштабувати до різних розмірів фільтрів. INM2-OM1-ядро – це ядро, яке включає вагові коефіцієнти для вилучення карти підфункцій 1 з карти функцій 1 з вхідної карти 2.

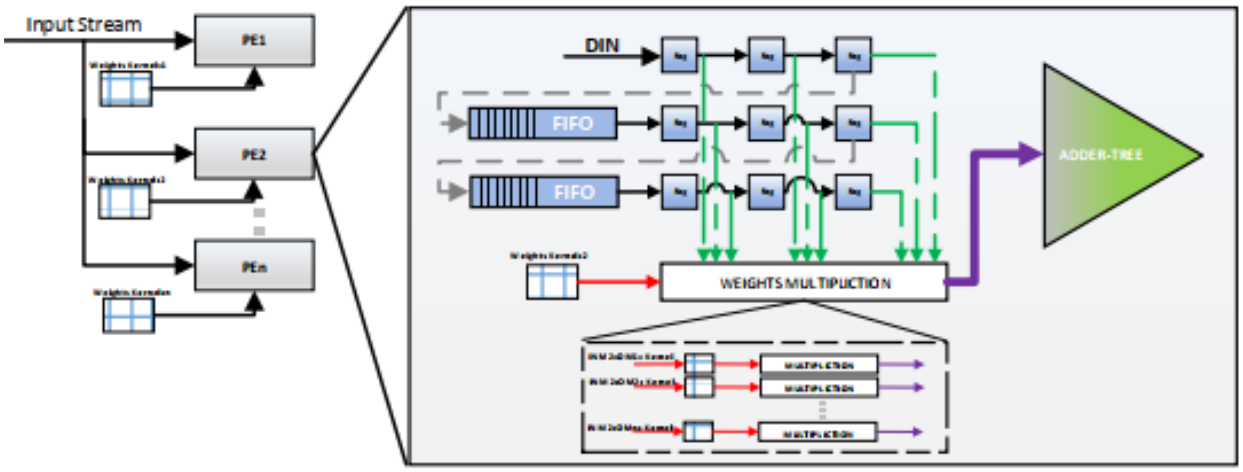

Рисунок 4.9 – Деталі архітектури

Деталі архітектури повного згорткового шару на скриншоті нижче. Різниця між першим згортковим шаром і пізнішими згортковими шарами полягає в тому, що перший шар може мати лише один елемент обробки, якщо зображення має відтінки сірого.

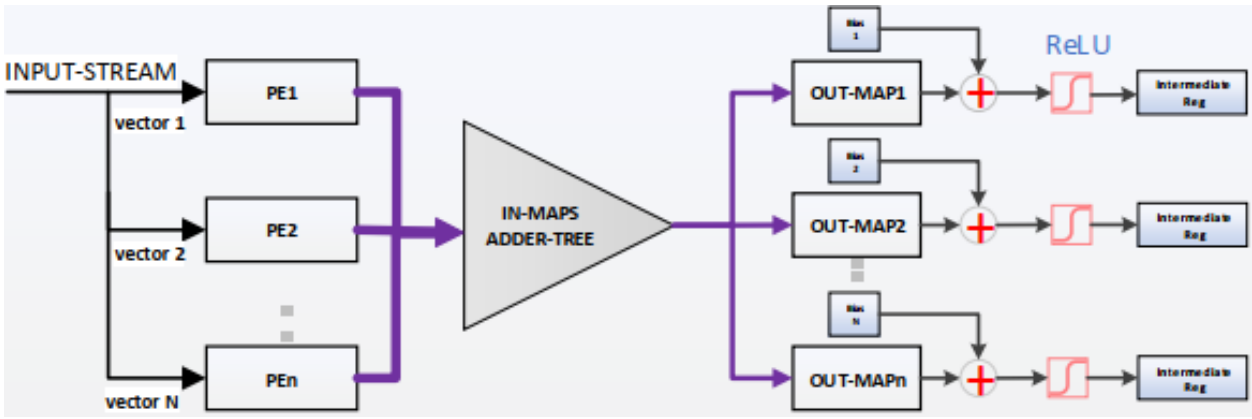

Рисунок 4.10 – Деталі архітектури

Розробка і побудова нейронних мереж у FPGA дозволяє значно прискорити процеси їх функціонування і навчання за рахунок використання можливості паралельної обробки від сотень до декількох тисяч обчислювальних потоків.

Процес генерації VHDL залежить від двох зовнішніх файлів, один є обов'язковим, а інший необов'язковим. Додатковий файл – це зовнішній файл конфігурації, а необхідний – файл параметрів.

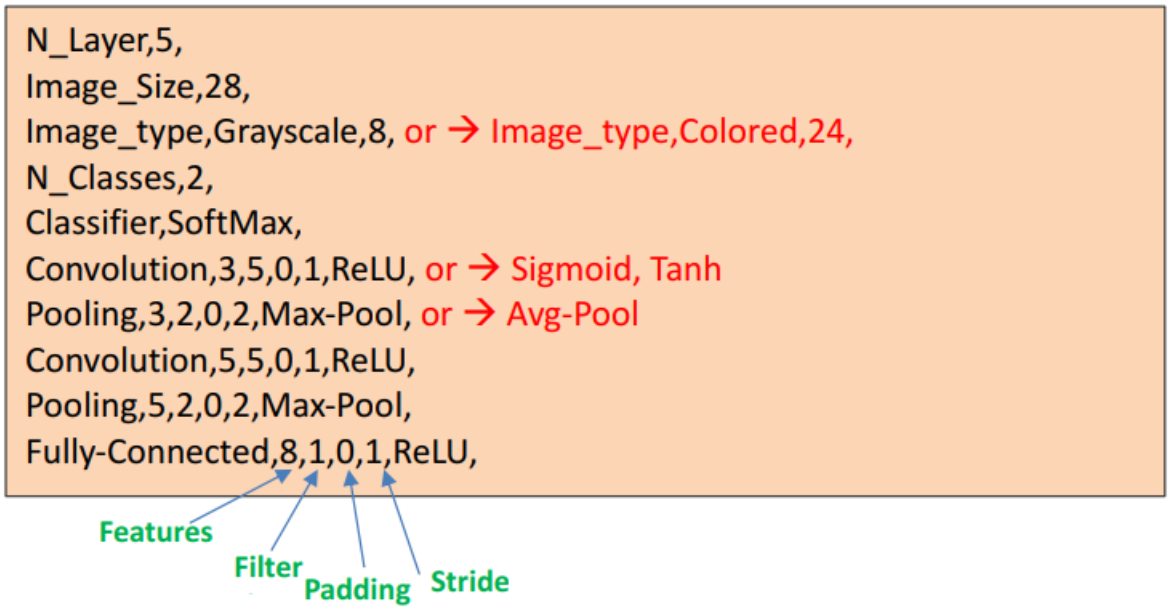

Рисунок 4.11 – Синтаксис файлу конфігурації прикладної моделі

На рисунку 4.11 показано синтаксис конфігурації для VGTEST, де N_Layer представляє кількість шарів у мережі; Image_Size – розмір вхідного зображення; Image_type визначає тип зображення, якщо воно кольорове або відтінки сірого, і 8 представляє ширину вхідних даних (пікселів), де 24 – кольорове, а 8 – відтінки сірого; N_classes представляє кількість вихідних класів, а Classifier - це використовувана функція класифікатора; Convolution,2,2,0,2,Max pool відповідно представляють назву шару, кількість вихідних карт об’єктів, розмір фільтра, заповнення, розмір кроку та використану функцію активації; той самий синтаксис застосовується до об'єднаних і повністю підключених шарів.

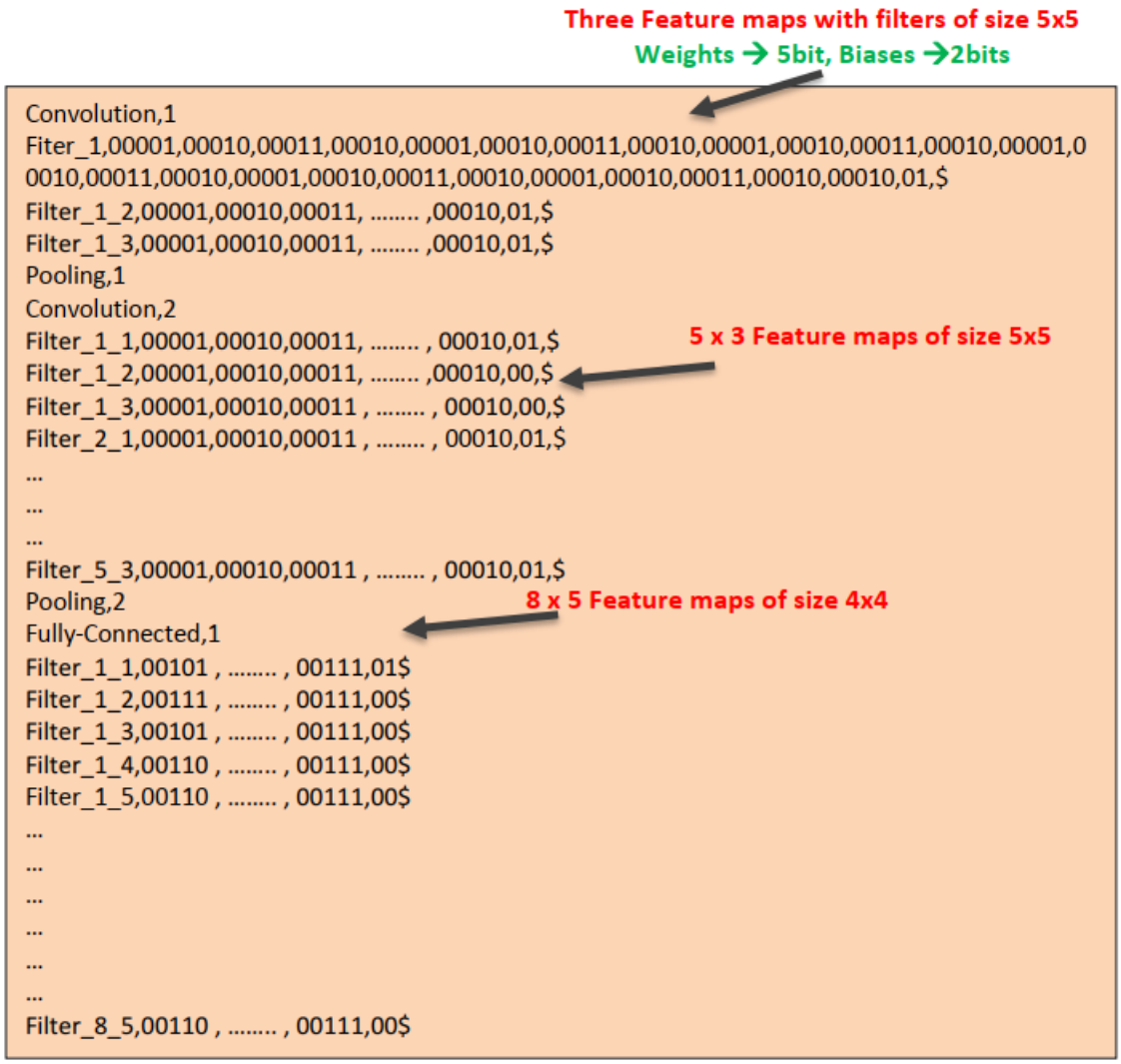

Рисунок 4.12– Синтаксичний файл випадкових параметрів прикладу

На рисунку 4.14 показано синтаксис файлу параметрів нашої моделі. Користувачі повинні структурувати цей файл наступним чином: Почніть з назви шару, а потім його карт об’єктів і фільтрів їх ваги та упередження. Назви чутливі до регістру шарів, але не для фільтрів. Рядки повинні закінчуватися знаком долара. Цей файл має відповідати налаштованій моделі в конфігурації етапі, а також на етапі включення параметрів, інакше він не буде прийнятий інструментом і призведе до повідомлення про помилку. Можливими причинами неприйняття файлу параметрів можуть бути; недійсне розширення файлу або невідповідний вміст; невідповідний розмір фільтра, кількість шарів/фільтрів або представлення даних; відсутні

упередження або знак долара в кінці кожного рядка.

Цей модуль обробляє шаблони шарів і підтримувані функції в кожному шаблоні. Основними шарами в CNN є згорткові, об'єднані та повністю зв'язані шари, і є три шаблони, по одному для кожного шару, які містять необхідні функції для реалізації цих шарів. Наприклад, згортковий шар складається з операцій згортки з подальшим додаванням зміщення та операцій функції активації. Операція згортки може бути реалізована різними засобами, які включають, але не обмежуються ними, систолічний масив або ковзне вікно. Ці дві функції зберігаються в шаблоні згорткового шару і на основі вхідних специфікацій, які передаються конструктором архітектури, окремі функції вибираються та структуруються певним чином. Аналогічно, існують різні функції активації, які включають, але не обмежуються ними, випрямлену лінійну одиницю, Tanh або сигмовидну, які можуть бути сформовані/структуровані на основі специфікацій, переданих конструктором архітектури. Цей процес параметризації також застосовується до об'єднаних і повністю пов'язаних шарів і це відповідно до їхніх відповідних функцій.

Генерація та зберігання VHDL є останнім етапом процесу генерації. Після передачі специфікацій моделі до менеджера бібліотеки параметризації фактичні шари CNN формуються на основі доступних шаблонів, отже, може відбуватися процес генерації. Після завершення процесу параметризації генератор коду записує код VHDL і зберігає його у визначеній папці відносно розташування інструмента на жорсткому диску.

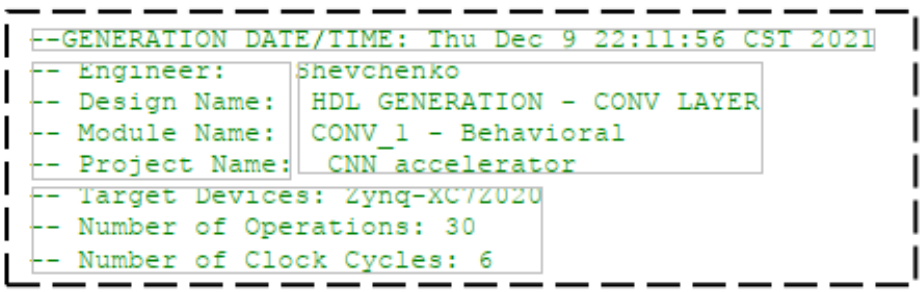

Рисунок 4.13 – Знімок розділу заголовка згенерованого коду VHDL

Загальний модуль визначає параметри, які можуть зробити проект реконфігурованим без необхідності вносити суттєві зміни в сам дизайн. Наприклад, оскільки розмір FIFO є параметризованим, розмір можна змінити, змінивши константу розміру FIFO в загальному модулі без повторного запису коду VHDL. Розроблено модуль для забезпечення можливості переналаштування дизайну без використання VGT для переналаштування мережі шляхом створення нового коду VHDL.

ВИСНОВКИ

Проведено аналіз методів та засобві генерації кода VHDL. Розроблено метод апаратного прискорення згорткових нейронних мереж на FPGA з використанням VHDL. Проведено аналіз топологій згорткових нейронних мереж; аналіз існуючих алгоритмів навчання згорткових нейронних мереж. Досліджено існуючі рішення апаратного прискорення. Розроблений інструмент генерації VHDL. Використання методу значно зменшить час розробки, необхідний для впровадження згорткової нейронної мережі, також знащить недоліки, що виникають через складність розробки мов опису апаратних засобів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Залеський В. Д., Івановський П. С., Федорченко В. М. Сучасні інструменти оркестрації даних для побудови конвеєрів автоматичної обробки даних // Системи управління, навігації та зв'язку. 2024. № 3. С.95-98.

2. Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A Neural Probabilistic Language Model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

3. R. Collobert and J. Weston, “A unified architecture for natural language processing,” *Proc. 25th Int. Conf. Mach. Learn. - ICML ’08*, vol. 20, no. 1, pp. 160–167, 2008.

4. A. Coates, A. Arbor, and A. Y. Ng, “An Analysis of Single-Layer Networks in Unsupervised Feature Learning,” *Aistats 2011*, pp. 215–223, 2011.

5. Q. V Le, A. Coates, B. Prochnow, and A. Y. Ng, “On Optimization Methods for Deep Learning,” *Proc. 28th Int. Conf. Mach. Learn.*, pp. 265–272, 2011.

6. G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 20, no. 1, pp. 30–42, 2012.

7. G. Hinton *et al.*, “Deep Neural Networks for Acoustic Modeling in Speech Recognition,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

8. J. L. F. Pereira and R. J. F. Rossetti, “An integrated architecture for autonomous vehicles simulation,” *Proc. 27th Annu. ACM Symp. Appl. Comput. - SAC ’12*, pp. 286– 292, 2012.

9. A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” pp. 1–9, 2012.

10. G. Lacey, G. W. Taylor, and S. Areibi, “Deep Learning on FPGAs: Past, Present, and Future,” 2016.

11. “MACHINE LEARNING.” [Online]. Available: http://www.mlplatform.nl/what-ismachine-learning/.

12. R. Jing та Y. Zhang, «A View of Support Vector Machines Algorithm on Classification Problems,» в International Conference on Multimedia Communications, 2010.

13. P. H. Swain та H. Hauska, «The Decision Tree Classifier: Design and Potential,» в IEEE Transactions on Geoscience Electronics, 1977.

14. M. Aquino, «Fake BACS Remittance Emails Delivers Dridex Malware,» 2014. [Електронний ресурс]. Available: https://blog.cyren.com/articles/fake-bacs-remittance-emails-delivers-dridex-malware.html.

15. Kaspersky Lab, "Kaspersky Security Bulletin 2015. Overall statistics for 2015," 2016. [Електронний ресурс].

16. J. Horton та J. Seberry, «Computer Viruses. An Introduction,» University of Wollongong, Wollongong, 1997.

17. C. Smith, A. Matrawy, S. Chow та B. Abdelaziz, «Computer Worms: Architectures, Evasion Strategies, and Detection Mechanisms,» Journal of Information Assurance and Security, № 4, pp. 69-83, 2009.

18. M. Moffie, W. Cheng, D. Kaeli та Q. Zhao, «Hunting Trojan Horses,» в Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability, San Jose, USA, 2006.

19. E. Chien, «Techniques of Adware and Spyware,» 2005.

20. A. Chuvakin, "An Overview of Unix Rootkits," 2003.

21. W. Lopez, H. Guerra, E. Pena, E. Barrera та J. Sayol, «Keyloggers,» 2013.

22. K. Savage, P. Coogan та H. Lau, «The Evolution of Ransomware,» 2015.

23. Juniper Research, «Cybercrime will cost businesses over $2 trillion by 2019,» 2016. [Електронний ресурс]. https://www.juniperresearch.com /press/press- releases/cybercrime-cost-businesses-over-2trillion.

24. V. Aliyev, «Using honeypots to study skill level of attackers based on the exploited vulnerabilities in the network,» Göteborg, 2010.

25. D. Harley and A. Lee, "Heuristic Analysis – Detecting Unknown

Viruses," 2009.

26. M. Abadi et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous
Distributed Systems,” 2016.

27. “2015- Deep Learning with Limited Numerical Precision.” LightNN : Filling the Gap between Conventional Deep Neural Networks and Binarized Networks,” pp. 2–7.

27. Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, and Lingli Wang, “A high performance FPGA-based accelerator for large-scale convolutional neural networks,” 2016 26th Int. Conf. F. Program. Log. Appl., pp. 1–9, 2016.

28. M. K. Hamdan and D. T. Rover, “VHDL generator for a high performance convolutional neural network FPGA-based accelerator,” in 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 2017, pp. 1–6.