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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Використання алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем»

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

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

_____ Ілля ФЕДЧУК
(підпис) *Ім'я ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Ілля ФЕДЧУК
Ім'я, ПРІЗВИЩЕ

Керівник: Світлана КУФТЕРІНА
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

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

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерна інженерія

Освітньо-професійна програма комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія
ЛАЩЕВСЬКА Ім'я, ПРІЗВИЩЕ
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Федчук Ілля
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Використання алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем

керівник кваліфікаційної роботи Світлана КУФТЕРІНА
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. №36

2. Строк подання кваліфікаційної роботи «___» _____2024 р.

3. Вихідні дані до кваліфікаційної роботи: _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Питання аналізу теорій та концепцій сучасного проектування ІС
2. Питання застосування ШІ для оптимізації роботи АПІС та приклади цього
3. Питання створення двох прикладів алгоритмів ШІ для оптимізації проектування ІС

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вивчення літератури та збір інформації	27.02-19.03.2024	Виконано
2	Обробка зібраного матеріалу	20.03-07.04.2024	Виконано
3	Розробка теоретичної частини дослідження	08.04-11.04.2024	Виконано
4	Проведення аналізу інформації	12.04-01.05.2024	Виконано
5	Розробка алгоритмів	02.05-04.05.2024	Виконано
6	Висновки за результатами аналізу алгоритмів	05.05-08.05.2024	Виконано
7	Розробка презентації	08.05-10.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано

Здобувач вищої освіти _____ Ілля ФЕДЧУК
(підпис) *(ім'я, ПРІЗВИЩЕ)*

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

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 47 стор., 12 рис., 0 табл., 30 джерел.

Мета роботи – Дослідити можливості використання алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем та розробити програмний комплекс для практичного застосування.

Об'єкт дослідження – Алгоритми штучного інтелекту та їх застосування для автоматизованого проектування інтегральних схем

Предмет дослідження – Застосування алгоритмів штучного інтелекту для оптимізації процесу проектування інтегральних схем на етапі розміщення компонентів.

Короткий зміст роботи: У цій роботі досліджується можливості штучного інтелекту для оптимізації роботи автоматизованого проєктування інтегральних схем. Аналізується традиційні методи проєктування інтегральних схем, висвітлюються проблеми, пов'язані з розвитком автоматизованого проєктування.

Досліджуються можливості застосування алгоритмів ШІ в автоматизації проєктування інтегральних схем. Розглядаються проблем традиційних методів проєктування інтегральних схем, описуються можливості використання ШІ для подолання обмежень традиційних методів, аналізується досвід компаній та стартапів використання ШІ в цій сфері, наводяться приклади роботи ШІ у проєктуванні інтегральних схем

Розроблено та протестовано два алгоритми проєктування інтегральних схем за допомогою мови Python. Обґрунтовано корисність цих алгоритмів у автоматизованому проєктуванні

КЛЮЧОВІ СЛОВА: ШІ, АПІС, ІС, машинне навчання, нейронні мережі, еволюційні алгоритми, алгоритм, ефективність, точність,

ЗМІСТ

ВСТУП

Технологічний прогрес стрімко розвивається, підштовхований постійним прагненням людства до вдосконалення та пошуку нових способів оптимізації роботи. В сучасному світі, де електронні системи стають не лише невід'ємною частиною нашого життя, але й його визначальним елементом, це прагнення виявляється особливо гостро в інтегральній електроніці. Інтегральні схеми перетворюються на серце та мозок електронних пристроїв, від мобільних телефонів до суперкомп'ютерів. Проте, розвиток цих технологій супроводжується зростанням їх складності та обсягів. Від телевізорів до медичних пристроїв, вимоги до швидкості, мініатюрності, енергоефективності та функціональності стають дедалі більшими викликами для інженерів та вчених у галузі електроніки. Вимоги до ефективності та продуктивності перетворюються не тільки на проблему, але й на стимул для розвитку нових технологій. Саме тут з'являється штучний інтелект– область комп'ютерної науки, яка зосереджується на створенні систем, здатних до само освіти, аналізу даних та прийняття рішень.

Мета даної кваліфікаційної роботи полягає в дослідженні алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем. Ця робота не тільки спрямована на збільшення продуктивності та швидкості процесу проектування, але й на створення більш ефективних та енергоефективних електронних пристроїв, що відкриває шлях до нових можливостей у сфері технологій. Розглядаючи історію розвитку алгоритмів штучного інтелекту та їхнє застосування у різних галузях, можна зробити висновок про потенційні переваги в їхньому використанні для автоматизованого проектування інтегральних схем. Ця дипломна робота ставить перед собою завдання розглянути ці переваги та вивчити можливості їхнього практичного застосування. У цьому контексті, розробка нових алгоритмів та їхнє експериментальне порівняння з існуючими методами проектування інтегральних схем визначається як ключова складова даної роботи. Подальший аналіз результатів експериментів та їхній вплив на сучасну електронну промисловість стануть основою для подальших досліджень у цій області.

Описана вище мета дослідження роблять дану роботу актуальною та важливою для сучасної науки та технологій. Із ростом комп'ютеризації суспільства та зростанням потреб у нових технологіях, розвиток алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем стає необхідним етапом у вдосконаленні електронних систем.

Об'єктом дослідження є алгоритми штучного інтелекту та їх застосування для автоматизованого проектування інтегральних схем. Предметом дослідження є застосування алгоритмів штучного інтелекту для оптимізації процесу проектування інтегральних схемна етапі розміщення компонентів. Практична значимість даної роботи полягає в можливості використання розроблених алгоритмів для автоматизованого проектування інтегральних схем, що призведе до зниження часу та витрат на проектування, покращення якості проекту та розробки більш ефективних та енергоефективних електронних пристроїв.

Теоретична значимість роботи полягає в огляді нових методів автоматизованого проектування інтегральних схем на основі алгоритмів штучного інтелекту, що збагатить теорію проектування електронних систем та відкриє нові напрямки досліджень у цій галузі.

РОЗДІЛ 1. ПЕРЕГЛЯД ОСНОВНИХ ТЕОРІЙ ТА КОНЦЕПЦІЙ У ПРОЕКТУВАННІ ІНТЕГРАЛЬНИХ СХЕМ

1.1 Перегляд сучасного стану дослідницьких відомостей щодо автоматизованого проектування інтегральних схем.

1.1.1 Традиційні методи проектування ІС

Традиційні методи проектування інтегральних схем (ІС) ґрунтуються на ручній роботі інженерів та використовуються вже протягом багатьох років. Ці методи можна поділити на кілька категорій:

1. Функціональне проектування:

Визначення функціональних вимог, на цьому етапі детально описуються всі функції, які повинна виконувати ІС. Це включає в себе визначення входів, виходів, режимів роботи, оброблюваних даних та алгоритмів. Створення специфікації, розробляється опис роботи ІС, який може включати блок-схеми, таблиці істинності, математичні моделі, діаграми потоків даних та інші специфікації. Ця документація чітко визначає очікувану поведінку ІС на всіх рівнях. Аналіз та верифікація, функціональні вимоги та специфікація ретельно аналізуються та верифікуються, щоб гарантувати їх повноту, узгодженість та відповідність поставленим завданням. На цьому етапі можуть використовуватися методи формальної верифікації та моделювання.

2. Логічне проектування:

Реалізація функцій за допомогою логічних елементів, використовуються базові логічні елементи, такі як AND, OR, NOT, та їх комбінації, для створення цифрових схем, які описують роботу ІС на двійковому рівні. Створення логічної схеми, логічна схема представляє собою графічне або текстове зображення ІС, де

кожен елемент відповідає певній логічній функції. Ця схема слугує основою для подальшого фізичного проектування. Оптимізація логічної схеми, логічна схема може бути оптимізована для зменшення її складності, підвищення швидкодії та зниження енергоспоживання. Це може включати методи скорочення логічних виразів, мінімізації кількості елементів та балансування навантаження.

3. Фізичне проектування:

Перетворення логічної схеми на фізичний макет, на цьому етапі логічна схема перетворюється на опис розміщення транзисторів, проводів, контактних майданчиків та інших компонентів на кристалі кремнію. Оптимізація розміщення, використовуються різні алгоритми та інструменти для оптимізації розміщення компонентів з метою мінімізації площі кристала, зменшення довжини з'єднань та покращення електричних характеристик ІС. Трасування з'єднань, визначається маршрутизація проводів, які з'єднують різні компоненти ІС. Важливо врахувати обмеження простору, мінімізувати перехресні перешкоди та забезпечити надійну передачу даних. Параметризація та верифікація, фізичний макет параметризується з урахуванням технологічних норм та характеристик компонентів. Проводиться верифікація макета для виявлення та виправлення помилок розміщення та трасування.

4. Технологічне проектування:

Вибір технологічного процесу, визначаються технологічні параметри виробництва ІС, такі як розмір транзисторів, матеріали, методи літографії, травлення, імплантації та інших технологічних процесів. Проектування тестових структур, розробляються спеціальні тестові структури на кристалі, які дозволяють контролювати та оцінювати параметри технологічного процесу та виявляти дефекти виробництва. Створення бібліотеки компонентів, створюється бібліотека стандартних компонентів, таких як транзистори, конденсатори, резистори тощо, з заданими параметрами та характеристиками. Технологічне моделювання, проводиться моделювання технологічного процесу для прогнозування електричних характеристик ІС та виявлення потенційних проблем.

5. Тестування :

Функціональне тестування, ІС тестується на предмет відповідності її функціональних характеристик заданим вимогам. Це включає подання на входи ІС тестових сигналів та перевірку відповідності вихідних сигналів очікуваним значенням. Тестування синхронізації, перевіряється правильна синхронізація роботи ІС. Це особливо важливо для цифрових схем, де сигнали повинні змінюватися вчасно та узгоджено. Тестування на стійкість до несприятливых факторів, ІС тестується на стійкість до різних несприятливих факторів, таких як робочі температури, перепади напруги, електромагнітні перешкоди та радіація. Використання автоматизованих тестових систем (АТЗ), розробляються автоматизовані тестові системи для ефективного та швидкого тестування ІС. АТЗ можуть генерувати тестові сигнали, зчитувати вихідні дані та аналізувати результати тестування.

1.1.2 Автоматизоване проектування інтегральних схем (АПІС): сучасні методи та інструменти

Функціональне проектування

сучасні методи функціонального проектування включають моделювання мовою опису апаратури (HDL) Моделювання HDL є ключовим етапом у проектуванні ІС, де опис функціональності ІС створюється за допомогою спеціальних мов, таких як Verilog[1], VHDL[2], SystemVerilog або SystemC. Ці мови описують поведінку ІС на мові, близькій до природної, що значно спрощує процес проектування та верифікації. Verilog широко використовувана мова HDL, що підходить для опису логічних схем, синхронних та асинхронних алгоритмів, а також тестових наборів;

VHDL, стандартизована мова HDL, що використовується для опису цифрових систем на всіх рівнях абстракції, від поведінки до фізичного рівня. SystemVerilog,

розширення Verilog, що додає можливості для верифікації, генерації тестів та моделювання на рівні системи. SystemC мова моделювання системного рівня, що використовується для опису поведінки ІС у контексті ширшої системи, включаючи програмне забезпечення та апаратне забезпечення;

синтез на основі поведінки – це процес автоматичного перетворення опису HDL ІС на логічну схему. Інструменти синтезу, такі як Synopsys Design Compiler[3], Cadence Genus[4] або Mentor Graphics Questa, аналізують опис HDL та генерують логічні елементи, такі як AND, OR, NOT, та їх комбінації, які реалізують задану функціональність;

перевірка синхронізації гарантує, що сигнали в ІС змінюються вчасно та узгоджено. Це критично важливо для цифрових схем, де синхронні сигнали керують роботою ІС. Інструменти перевірки синхронізації, такі як Synopsys Vera[5] або Cadence Jasper, виявляють потенційні проблеми синхронізації, такі як порушення часових обмежень або метастабільні стани.

Логічне проектування

сучасні методи логічного проектування включають: Оптимізація логічної схеми це процес мінімізації розміру та підвищення швидкодії логічної схеми ІС. Інструменти оптимізації, такі як Synopsys PrimeTime, Cadence Voltus або Mentor Graphics Volcano, можуть зменшувати кількість логічних елементів, балансувати навантаження на сигналах та оптимізувати часові характеристики ІС;

перевірка синхронізації також важлива на етапі логічного проектування, щоб гарантувати правильну роботу ІС на логічному рівні. Інструменти перевірки синхронізації, такі як Synopsys Vera або Cadence Jasper, можуть виявляти проблеми синхронізації, які не були виявлені на етапі функціонального проектування. Статичний аналіз – це метод верифікації логічної схеми ІС без використання тестових сигналів. Інструменти статичного аналізу, такі як Synopsys SpyGlass або Cadence Incisive, можуть виявляти потенційні помилки в дизайні, такі як

невикористані логічні елементи, критичні шляхи та порушення обмежень на час встановлення та утримання.

Фізичне проектування.

Сучасні методи фізичного проектування включають: Використання CAD-систем: CAD-системи надають користувачеві графічний інтерфейс для розміщення компонентів, трасування з'єднань та візуалізації фізичного проекту IC.

розміщення та трасування – це процес компонування транзисторів, проводів та інших компонентів на кристалі кремнію. Інструменти розміщення та трасування, такі як Cadence Genus, Synopsys IC Compiler або Mentor Graphics Calibre, автоматично розміщують компоненти та прокладають з'єднання між ними, враховуючи обмеження простору, електричні характеристики та технологічні норми;

аналіз та оптимізація синхронізації, на етапі фізичного проектування також важливо перевірити та оптимізувати синхронізацію IC. Розміщення та трасування компонентів можуть вплинути на часові характеристики IC. Інструменти аналізу та оптимізації синхронізації, такі як Synopsys PrimeTime або Cadence Tempus, аналізують вплив фізичної реалізації на сигнали та допомагають оптимізувати розміщення та трасування для дотримання часових обмежень;

екстракція ký sinh trùng - ký sinh trùng це процес врахування впливу паразитних компонентів, які з'являються під час виробництва IC. Ці паразитні компоненти, такі як ємність та опір між з'єднаннями, можуть впливати на електричні характеристики IC. Інструменти екстракції, такі як Cadence QRC Extraction або Synopsys StarRC, враховують ці паразитні ефекти під час моделювання IC;

розмітка (layout verification) - це процес перевірки того, чи відповідає фізична реалізація IC правилам технологічного процесу. Інструменти розмітки, такі як Mentor Graphics Calibre або Synopsys IC Validator, гарантують, що розміщення та

трасування компонентів не порушують мінімальні відстані, правила розміщення шарів металізації та інші вимоги виробництва.

Технологічне проектування

технологічний процес визначає, як фізична реалізація ІС перетворюється на реальний компонент. Він включає матеріали, методи літографії, травлення, імплантації та інші кроки виробництва. Вибір технологічного процесу залежить від необхідних характеристик ІС, таких як розмір, швидкість, потужність та вартість.

оптимізація для випуску (Design for Manufacturability - DFM)[6] - це методологія проектування ІС з урахуванням обмежень та можливостей конкретного виробничого процесу. Інструменти DFM допомагають інженерам оптимізувати дизайн ІС для покращення виходу виробів, зниження витрат та забезпечення надійного виробництва;

бібліотеки стандартних блоків (Standard Cell Libraries - SCLs)[7] - це бібліотеки вже розроблених, перевірених та характеризувальних компонентів, таких як транзистори, резистори, конденсатори тощо. Інженери використовують ці бібліотеки під час розміщення та трасування, що дозволяє їм повторно використовувати надійні блоки та скорочувати час проектування.

Синтез ІС

сучасні методи синтезу ІС включають використання алгоритмів машинного навчання: Алгоритми машинного навчання використовуються для автоматичного генерування ефективних проектів ІС на основі даних про попередні проекти;

використання еволюційних алгоритмів: Еволюційні алгоритми моделюють процес природного відбору для генерування та оптимізації проектів ІС;

інструменти інтеграції проектування (EDA): EDA-інструменти є комплексними програмними середовищами, які інтегрують різні методи та інструменти АПІС, включаючи синтез ІС.

1.1.3 Проблеми розвитку АПІС

АПІС безперечно вважається однією з найбільш швидко зростаючих галузей у сучасній електроніці, Проте разом зі своїм вражаючим ростом, АПІС також стикається зі своїми власними непростими проблемами, а саме:

а) Складність алгоритмів

1) *Оптимізація*. Пошук оптимальних рішень для таких задач, як розміщення та трасування компонентів на кристалі кремнію, є дуже складною задачею, що потребує значних обчислювальних ресурсів. Сучасні алгоритми оптимізації, такі як алгоритми еволюційного моделювання, мурашиних колоній та табу-пошуку, дозволяють знаходити кращі рішення, але все ще потребують значних обчислювальних ресурсів для складних ІС. Наприклад, оптимізація розміщення та трасування для процесора з 20 мільярдами транзисторів може потребувати тижнів або навіть місяців розрахунків на потужних комп'ютерах;

2) *Верифікація*. Перевірка правильності та надійності складних ІС з мільярдами транзисторів є складною задачею, яка потребує нових методів та інструментів верифікації. Традиційні методи верифікації, такі як симуляція та формальна верифікація, не завжди здатні виявити всі потенційні проблеми. Нові методи верифікації, такі як статистична верифікація та верифікація на основі машинного навчання, можуть допомогти подолати цю проблему. Наприклад, статистична верифікація використовує статистичні методи для оцінки ймовірності того, що ІС буде відповідати заданим специфікаціям, а верифікація на основі машинного навчання використовує алгоритми машинного навчання для виявлення потенційних проблем в дизайні.

б) Висока вартість

1) *Розробка інструментів*. Розробка та підтримка складних інструментів АПІС може бути дорогою, що робить їх доступними лише для великих компаній. Малі та середні компанії можуть не мати ресурсів для придбання та використання цих інструментів. Наприклад, вартість ліцензії на програмне

забезпечення для розміщення та трасування компонентів може сягати десятків тисяч доларів;

2) *Ліцензування*. Ліцензування програмного забезпечення для АПІС може бути дорогим, що робить його доступним лише для великих компаній. Висока вартість ліцензування може стримувати використання АПІС малими та середніми компаніями, що може призвести до неконкурентноспроможності. Наприклад, малій компанії може бути складно конкурувати з великою компанією, яка має доступ до найсучасніших інструментів АПІС.

в) Соціальні та економічні проблеми

1) *Вплив на робочі місця.* АПІС може призвести до втрати робочих місць в інженерних та інших сферах, пов'язаних з проектуванням ІС. Автоматизація процесу проектування може зробити роботу інженерів менш необхідною, що може призвести до скорочення штату та безробіття;

2) *Стандартизація*. Наразі немає єдиних стандартів для АПІС. Це може ускладнювати розробку та впровадження цієї технології, адже різні компанії можуть використовувати різні підходи та методи. Стандартизація може допомогти забезпечити сумісність між різними інструментами АПІС та сприяти більш широкому застосуванню цієї технології;

г) Недостатня гнучкість

1) *Обмеженість методів.* Інструменти АПІС можуть не завжди бути гнучкими та нездатними враховувати всі нюанси проектування складних ІС. Це може призвести до того, що інженерам доведеться вручну вносити зміни до проекту, що може збільшити час розробки та ризик помилок. Наприклад, інженерам може знадобитися вручну розставити деякі критичні компоненти або оптимізувати маршрутизацію сигналів;

2) *Необхідність ручної роботи.* Деякі етапи проектування ІС все ще потребують ручного втручання, що може призвести до помилок та зниження продуктивності. Наприклад, інженерам може знадобитися вручну оптимізувати логічну схему або генерувати тестові набори.

г) Недосконаліть моделей

1) Моделювання технологічних процесів. Моделювання технологічних процесів ІС може бути складним та неточним, що може призвести до неточностей в проектуванні. Сучасні моделі технологічних процесів ґрунтуються на фізичних симуляціях та враховують різні фактори, такі як електроміграція, фоторезисти та дифракція, але все ще можуть мати деякі неточності. Ці неточності можуть призвести до того, що ІС не буде відповідати заданим специфікаціям, що може призвести до його відмови або зниження продуктивності;

2) Моделювання паразитних ефектів. Паразитні ефекти, такі як ємність та індуктивність між з'єднаннями, можуть впливати на поведінку ІС. Моделювання цих паразитних ефектів є складним завданням, оскільки їх точні характеристики залежать від розміщення та трасування компонентів. Неточне моделювання паразитних ефектів може призвести до неточностей в прогнозованій поведінці ІС.

д) Безпека

1) *Вразливість до атак*. Інструменти АПІС можуть бути вразливими до атак, що може призвести до крадіжки інтелектуальної власності (ІВ) або створення ненадійних ІС. Зловмисники можуть атакувати інструменти АПІС, щоб отримати доступ до конфіденційних даних про дизайн ІС. Вони також можуть модифікувати інструменти АПІС, щоб створити ненадійні ІС, які можуть вийти з ладу або мати небажані функції. Наприклад, зловмисник може модифікувати інструмент розміщення та трасування, щоб додати до ІС прихований backdoor;

2) обмеження відкритого апаратного забезпечення (Open-Source Hardware - OSH): Висока вартість та складність інструментів АПІС можуть бути бар'єром для розвитку відкритого апаратного забезпечення (ОАЗ). ОАЗ передбачає відкритий доступ до специфікацій та дизайну апаратних компонентів, що дозволяє спільноті інженерів працювати над їхнім поліпшенням. Однак, складність інструментів АПІС може ускладнювати розробку та спільну роботу над ОАЗ-проектами.

е) Етичні міркування

1) Автономні системи. Розвиток складних ІС сприяє створенню автономних систем, які можуть самостійно приймати рішення. Це ставить питання про етику та безпеку таких систем. Наприклад, необхідно розробити етичні рамки для використання автономних систем у критичних сферах, таких як медицина та транспорт;

2) вплив на довкілля. Виробництво ІС потребує використання води, енергії та хімічних речовин, що може негативно впливати на довкілля. Інженерам необхідно розробляти ІС з урахуванням їхнього впливу на довкілля та використовувати екологічно чисті технології виробництва.

1.1.4 Перспективи розвитку АПІС

Автоматизоване проектування інтегральних схем (АПІС) постійно розвивається, і з'являються нові методи та технології, які роблять цю область ще більш потужною та гнучкою. Ось деякі з перспективних напрямків розвитку АПІС:

а) Інтегровані середовища розробки (IDE)

1) Уніфіковані середовища. Розробляються уніфіковані середовища розробки (IDE), які поєднують в собі різні інструменти АПІС в одному інтерфейсі. Це може зробити процес проектування ІС більш зручним та ефективним. Наприклад, існують IDE, які поєднують в собі інструменти для опису HDL, розміщення та трасування, верифікації та тестування;

2) Прогнозування помилок. IDE можуть використовувати алгоритми машинного навчання та ШІ для прогнозування потенційних помилок на ранніх етапах проектування. Це може допомогти інженерам запобігти помилкам та економити час та ресурси. Наприклад, IDE можуть використовувати машинне навчання для виявлення потенційних проблем з логікою або розміщенням компонентів;

3) *Підтримка OSH*. IDE можуть пропонувати кращу підтримку для проектів OSH, що може зробити розробку відкритого апаратного забезпечення більш доступною та зручною. Наприклад, IDE можуть надавати інструменти для спільної роботи над проектами OSH та для публікації та обміну дизайном IC.

б) Обчислювальні хмари

1) *Хмарні інструменти АПІС*. З'являються хмарні інструменти АПІС, які дозволяють інженерам використовувати потужні обчислювальні ресурси хмари для проектування IC. Це може зробити АПІС більш доступним для малих та середніх компаній, які не мають власних потужних комп'ютерів. Наприклад, існують хмарні сервіси, які пропонують інструменти для розміщення та трасування компонентів, верифікації та тестування;

2) Співпраця. Хмарні інструменти АПІС можуть полегшити співпрацю над проектами IC між інженерами, які знаходяться в різних місцях. Наприклад, інженери можуть спільно працювати над дизайном IC в хмарному середовищі, використовуючи одні й ті ж інструменти та дані;

3) зниження витрат: Використання хмарних інструментів АПІС може допомогти знизити витрати на проектування IC, оскільки компаніям не потрібно купувати та підтримувати дороге програмне забезпечення.

в) Нові технології виробництва

1) Тривимірні IC (3D ICs). Тривимірні IC дозволяють розміщувати компоненти один на одному, що може збільшити щільність компонентів та продуктивність IC. АПІС повинен бути адаптований для врахування особливостей проектування 3D IC. Наприклад, інструменти розміщення та трасування повинні враховувати вертикальні з'єднання між компонентами;

2) Нові матеріали. Нові матеріали, такі як вуглецеві нанотрубки та графен, можуть використовуватися для створення IC з новими властивостями. Інструменти моделювання АПІС повинні бути оновлені для підтримки моделювання цих нових матеріалів.

г) Безпека

1) Захист інтелектуальної власності. Розробляються нові методи захисту ІВ для запобігання крадіжці конфіденційних даних про дизайн ІС. Ці методи можуть включати шифрування даних проекту та контроль доступу до інструментів АПІС;

2) Безпечні ІС. АПІС може відігравати важливу роль у створенні безпечних ІС, стійких до атак. Інструменти АПІС можуть бути використані для аналізу вразливостей в дизайні ІС та для реалізації захисних механізмів.

1.2 Основні теоретичні підходи до використання штучного інтелекту для автоматизованого проектування інтегральних схем.

Концепції

а) Нейроморфні обчислення - це галузь, яка поєднує в собі штучний інтелект та електроніку для створення комп'ютерних систем, натхненних людським мозком. Нейроморфні інтегральні схеми (НІС) - це тип інтегральних схем, спеціально розроблених для реалізації нейроморфних алгоритмів.

1) *IBM TrueNorth*[8]. нейроморфний процесор, який використовує аналогові нейрони для досягнення високої енергоефективності при обробці сенсорних даних.

2) *Intel Loihi*[9]. нейроморфний процесор, який використовує шипи для представлення даних та нейронів для їх обробки, що робить його придатним для задач машинного навчання.

б) Еволюційні алгоритми - це метод оптимізації, натхненний еволюцією. Вони використовують популяцію кандидатів на рішення, які називаються хромосомами, які з часом еволюціонують, щоб покращити їх пристосованість до заданої проблеми. Кожна хромосома кодує можливе рішення проблеми, а її пристосованість визначається тим, наскільки добре вона вирішує цю проблему.

1) *Cadence Genus*. використовує еволюційний алгоритм для розміщення компонентів ІС, оптимізуючи площу кристала та затримку сигналу.

2) *Synopsys Design Compiler*. використовує еволюційний алгоритм для оптимізації параметрів транзисторів ІС, покращуючи продуктивність та енергоефективність.

в) Машине навчання (МН) - це галузь штучного інтелекту, яка дозволяє комп'ютерам навчатися без явного програмування. Алгоритми МН можуть навчатися на даних, виявляючи закономірності та роблячи прогнози.

1) *Google TensorFlow*[10]. платформа машинного навчання з відкритим кодом, яка використовується для прогнозування дефектів ІС, оптимізації параметрів ІС та інших задач.

2) *Synopsys PyTorch*[11]. платформа машинного навчання з відкритим кодом, яка використовується для розпізнавання образів на ІС, що може бути корисно для інспекції ІС та виявлення дефектів.

г) Глибоке навчання (ГН) - це підмножина машинного навчання, яка використовує штучні нейронні мережі (ШНМ) з навчанням подань. На відміну від традиційних методів машинного навчання, які працюють з чітко визначеними ознаками, ГН навчається автоматично виявляти складні закономірності та абстракції з даних.

Методології

г) Автоматизоване проектування інтегральних схем - це процес використання комп'ютерних програмних інструментів для створення та оптимізації інтегральних схем . Інтегральні схеми - це мікроелектронні пристрої, які містять велику кількість транзисторів, резисторів, конденсаторів та інших компонентів, що з'єднані між собою на одній кремнієвій підкладці. АПІС використовується для створення широкого спектру ІС,

д) Оптимізація топології ІС - це процес автоматичного покращення розміщення та з'єднання компонентів на кремнієвій підкладці з метою покращення

характеристик ІС. Ці характеристики можуть включати продуктивність, енергоефективність, надійність та вартість виробництва.

1) *Cadence Tempus*. використовує алгоритми ШІ для автоматичного оптимізації топології ІС, що може призвести до значного покращення продуктивності та енергоефективності.

2) *Synopsys StarRC*. використовує алгоритми ШІ для аналізу електромагнітних характеристик ІС, що може допомогти оптимізувати розміщення компонентів та трасування проводів.

е) Прогнозування дефектів ІС - це процес виявлення та прогнозуванняпотенційних дефектів в ІС на ранніх стадіях процесу проектування тавиробництва. Цей процес має на меті запобігти випуску дефектних ІС,що може призвести до значних фінансових втрат та шкоди репутації.

1) *IBM Watson for Semiconductor Manufacturing*[12]. використовує алгоритми машинного навчання для аналізу даних виробництва ІС та прогнозування потенційних дефектів.

2) *Samsung Foundry Predictive Analytics*[13]. використовує алгоритми машинного навчання для прогнозування виходу ІС, що може допомогти оптимізувати процес виробництва.

є) Розпізнавання образів - це галузь досліджень, яка зосереджена на розробці методів та алгоритмів для розпізнавання образів безпосередньо на кремнієвій підкладці. На відміну від традиційних методів розпізнавання образів, які використовують програмне забезпечення на центральних процесорах або графічних процесорах, розпізнавання образів в ІС виконується апаратними засобами, що може призвести до значного покращення продуктивності та енергоефективності.

1) *Cognex Deep Learning Vision System*. використовує алгоритми глибокого навчання для розпізнавання дефектів на ІС, що може допомогти покращити контроль якості та знизити брак.

1.3 учного інтелекту для автоматизованого проектування інтегральних схем.

1.3.1 Генетичні Алгоритми

Переваги: Генетичні Алгоритми можуть бути застосовані до широкого кола задач, не потребуючи глибокого розуміння проблеми. Це робить їх корисними для дослідження нових областей та вирішення складних проблем, які важко формалізувати. Генетичні Алгоритми можуть ефективно знаходити оптимальні або близькі до оптимальних рішення в складних просторах пошуку. Це робить їх цінними для задач, де традиційні методи оптимізації, такі як лінійне програмування, неефективні. Вони можуть генерувати нові та інноваційні рішення, які неможливо знайти за допомогою традиційних методів. Це робить їх корисними для творчого вирішення проблем та генерування нових ідей.

Недоліки: Генетичні Алгоритми можуть бути обчислювально складними, особливо для задач з великими просторами пошуку. Це може зробити їх непрактичними для задач, де час є критичним фактором. Вони можуть бути повільними, особливо для задач, де потрібна висока точність. Це може зробити їх непрактичними для задач, які потребують швидкого прийняття рішень. Генетичні Алгоритми не гарантують знаходження оптимального рішення, лише близького до нього. Це означає, що існує ймовірність того, що алгоритм знайде рішення, яке не є найкращим можливим.

1.3.2 Машине навчання

Переваги: Алгоритми МН можуть автоматично навчатися на даних, не потребуючи явного програмування. Це робить їх корисними для задач, де важко або неможливо вручну визначити правила або закономірності. Алгоритми можуть робити точні прогнози на основі даних. Це робить їх цінними для задач, де важливо приймати рішення на основі даних, таких як прогнозування попиту, виявлення

шахрайства та медичний діагноз. Вони можуть адаптуватися до нових даних і вдосконалювати свою продуктивність з часом. Це робить їх корисними для задач, де дані постійно змінюються, таких як прогнозування фондового ринку та рекомендаційні системи.

Недоліки: Алгоритми МН залежать від якості та кількості даних, на яких вони навчаються. Це означає, що для досягнення високої точності їм потрібна велика кількість високоякісних даних. Деякі алгоритми МН можуть бути складними для інтерпретації, що може ускладнити розуміння того, як вони приймають рішення. Це може бути проблемою для задач, де важливо розуміти, як приймаються рішення. Алгоритми МН можуть бути упередженими, якщо вони навчаються на даних, які містять упередженість. Це може призвести до несправедливих або дискримінаційних рішень.

1.3.3. Нейроморфні обчислення

Переваги: Нейроморфні схеми можуть виконувати обчислення з низьким енергоспоживанням порівняно з традиційними процесорами. Це робить їх корисними для задач, де енергоспоживання є обмеженням, наприклад, для вбудованих систем та автономних пристроїв. Нейроморфні схеми можуть виконувати обчислення паралельно, що робить їх швидкими для певних задач, особливо тих, які добре узгоджуються з архітектурою нейронних мереж. Нейроморфні схеми можуть бути більш стійкими до помилок, ніж традиційні процесори, завдяки їхній розподіленій природі обробки.

Недоліки: Складність проектування: Проектування ефективних нейроморфних схем може бути складним завданням. Це потребує розуміння як нейронних мереж, так і апаратного забезпечення. Нейроморфні схеми часто бувають менш гнучкими, ніж традиційні процесори. Вони можуть бути добре налаштовані для виконання конкретної задачі, але їх може бути важко перепрограмувати для іншої задачі. Нейроморфні обчислення є порівняно молодою технологією, і вона ще розвивається. Це означає, що інструменти та методи

проектування можуть бути менш зрілими, ніж для традиційних процесорів.

1.3.4 Глибоке навчання

Переваги: Алгоритми ГН можуть досягати високої точності на складних задачах, таких як розпізнавання образів та обробка природної мови. Вони можуть автоматично виявляти складні закономірності в даних, що робить їх корисними для задач, де важко вручну визначити ознаки. Алгоритми ГН можуть бути застосовані до широкого кола задач, що робить їх потужним інструментом для вирішення складних проблем.

Недоліки: Навчання моделей ГН може бути обчислювально складним завданням, яке потребує потужних комп'ютерних ресурсів. Алгоритми ГН зазвичай потребують великих обсягів даних для навчання, що може бути обмеженням для деяких задач. Деякі моделі ГН можуть бути складними для інтерпретації, що ускладнює розуміння того, як вони приймають рішення. Це може бути проблемою для задач, де важливо розуміти логіку прийняття рішень.

1.4 Висновки щодо найбільш перспективних теорій та концепцій

Аналіз привів до висновку, що кілька перспективних теорій і концепцій можуть бути використані в додаткових дослідженнях на тему того, як використовувати штучний інтелект у комбінованих електронних технологіях.

Нейроморфні обчислення мають значний потенціал для революціонізації дизайну інтегральних схем , пропонуючи нові можливості для підвищення енергоефективності, продуктивності та інтелектуальних можливостей ІС. Дослідження в цій галузі можуть зосередитися на:

Розробці нових нейроморфних структур та методів: Це може включати дослідження нових типів нейроморфних компонентів, таких як штучні синапси, нейрони та нейронні мережі, а також нових методів проектування та реалізації

нейроморфних ІС.

Вдосконаленні методів інтеграції нейроморфних компонентів в електронні схеми: Це може включати дослідження нових матеріалів та технологій для створення нейроморфних ІС, а також нових методів тестування та налагодження нейроморфних ІС.

Глибоке навчання вже продемонструвало значний успіх у вирішенні складних задач, таких як розпізнавання образів, обробка природної мови та машинне навчання. У контексті проектування ІС глибоке навчання може використовуватися для:

Оптимізації топографії ІС: Алгоритми глибокого навчання можуть автоматично оптимізувати розміщення та трасування компонентів на ІС, що може призвести до покращення продуктивності, енергоефективності та площі ІС.

Виявлення дефектів ІС: Алгоритми глибокого навчання можуть автоматично виявляти дефекти в ІС на ранніх стадіях виробництва, що може допомогти знизити витрати та покращити якість ІС. Розпізнавання образів на ІС: Алгоритми глибокого навчання можуть використовуватися для автоматичного розпізнавання дефектів, аномалій та інших особливостей на ІС, що може допомогти в інспекції та тестуванні ІС.

Інтеграція ШІ в інструменти та методології автоматизованого проектування інтегральних схем (АПІС) може значно підвищити ефективність та продуктивність процесу проектування ІС. Дослідження в цій галузі можуть зосередитися на:

Розробці нових методів інтеграції ШІ в інструменти АПІС: Це може включати розробку нових інтерфейсів користувача, які полегшують використання ШІ в процесі проектування ІС, а також нових методів автоматизації задач проектування ІС за допомогою ШІ.

Покращенні інтерфейсів візуалізації ШІ в процесі АПІС: Це може включати розробку нових методів візуалізації даних та результатів ШІ, що може допомогти інженерам краще зрозуміти та використовувати можливості ШІ в процесі проектування ІС.

РОЗДІЛ 2. АНАЛІЗ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В АВТОМАТИЗВАНОМУ ПРОЄКТУВАНІ ІНТЕГРАЛЬНИХ СХЕМ

2.1 Застосування алгоритмів ШІ в АПІС

а) Автоматизоване проектування логіки

1) *Синтез високого рівня*. Алгоритми ШІ можуть автоматично генерувати логічний опис ІС з опису його функціональності на поведінковому рівні. Це дозволяє інженерам описувати функціональність ІС природною мовою, не вникаючи в деталі його логічної реалізації. Алгоритми ШІ можуть потім генерувати оптимальний логічний опис, який відповідає цій функціональності.

2) *Оптимізація логіки*. Алгоритми ШІ можуть оптимізувати логічний опис ІС, щоб зменшити його розмір, затримку сигналу та електроспоживання. Це може призвести до створення більш компактних, швидких та енергоефективних ІС. Алгоритми ШІ можуть застосовувати різні методи оптимізації, такі як скорочення логічних виразів, розбиття логіки на менші модулі та вибір оптимальних логічних елементів.

3) *Перевірка логіки*. Алгоритми ШІ можуть перевіряти логічний опис ІС на наявність помилок, таких як логічні суперечності та порушення синхронізації. Це може допомогти запобігти помилкам на ранніх етапах проектування ІС, що може заощадити час і ресурси. Алгоритми ШІ можуть використовувати різні методи перевірки, такі як симуляція логіки та формальний аналіз.

б) Автоматизоване розміщення компонентів

1) Мінімізація площі. Алгоритми ШІ можуть розміщувати компоненти ІС на кремнієвій підкладці таким чином, щоб мінімізувати площу, яку вони займають. Це може призвести до створення більш компактних ІС, що може знизити витрати на виробництво. Алгоритми ШІ можуть використовувати різні методи розміщення,

такі як алгоритми "жадібного" розміщення, алгоритми "симплексного" відпалу та еволюційні алгоритми.

2) *Зменшення затримки сигналу*. Алгоритми ШІ можуть розміщувати компоненти ІС таким чином, щоб зменшити затримку сигналу між ними. Це може призвести до створення більш швидких ІС. Алгоритми ШІ можуть враховувати при розміщенні такі фактори, як довжина з'єднань, взаємне розташування компонентів та паразитні ефекти.

3) *Уникнення перешкод*. Алгоритми ШІ можуть розміщувати компоненти ІС таким чином, щоб уникнути перешкод між ними. Це може допомогти запобігти фізичним дефектам ІС, які можуть призвести до їх відмови. Алгоритми ШІ можуть використовувати різні методи для виявлення та уникнення перешкод, такі як аналіз траєкторій та алгоритми розрахунку відстаней.

в) Автоматизоване трасування з'єднань

1) *Зменшення електромагнітних перешкод (ЕМП)*. Алгоритми ШІ можуть трасувати з'єднання таким чином, щоб зменшити електромагнітні перешкоди між ними. Це може допомогти запобігти шумам та іншим проблемам, які можуть призвести до відмови ІС. Алгоритми ШІ можуть використовувати різні методи для зменшення ЕМП, такі як розбиття з'єднань на менші сегменти та використання спеціальних методів трасування.

2) *Уникнення перешкод трасування*. Алгоритми ШІ можуть трасувати з'єднання таким чином, щоб уникнути перешкод між ними. Це може допомогти запобігти фізичним дефектам ІС, які можуть призвести до їх відмови. ШІ можуть використовувати різні методи для виявлення та уникнення перешкод, такі як аналіз траєкторій та алгоритми розрахунку відстаней.

г) Автоматизована оптимізація фізичного проекту

1) *Зменшення площі*. Алгоритми ШІ можуть оптимізувати розміщення компонентів та трасування з'єднань, щоб зменшити площу, яку займає ІС. Це може призвести до створення більш компактних ІС, що може знизити витратина виробництво. Алгоритми можуть використовувати різні методи

оптимізації, такі як алгоритми "жадібного" розміщення, алгоритми "симплексного" відпалу та еволюційні алгоритми.

Зменшення затримки сигналу: Алгоритми ШІ можуть оптимізувати розміщення компонентів та трасування з'єднань, щоб зменшити затримку сигналу між ними. Це може призвести до створення більш швидких ІС. Алгоритми ШІ можуть враховувати при розміщенні та трасуванні такі фактори, як довжина з'єднань, взаємне розташування компонентів та паразитні ефекти.

Зменшення електроспоживання: Алгоритми ШІ можуть оптимізувати розміщення компонентів та трасування з'єднань, щоб зменшити електроспоживання ІС. Це може призвести до створення більш енергоефективних ІС, що може допомогти знизити витрати на експлуатацію. ШІ можуть використовувати різні методи оптимізації, такі як вибір економних компонентів, оптимізація режиму роботи ІС та використання методів зниження паразитних ефектів.

2.2 Аналіз традиційних методів проектування інтегральних схем

2.2.1 Обмеження традиційних методів проектування інтегральних схем

Традиційні методи проектування інтегральних схем, хоча й добре вивчені та поширені, мають ряд суттєвих обмежень, які роблять їх неефективними для проектування складних сучасних ІС. До таких обмежень належать:

а) Зростання складності

1) *Зростання кількості транзисторів*. Сучасні ІС містять мільярди транзисторів, що робить ручне проектування та верифікацію практично неможливими.

Наприклад, процесор Intel Xeon Platinum 8185M має 20 мільярдів транзисторів, а графічний процесор NVIDIA GeForce RTX 3090 Ti - 54 мільярди.

2) *Зростання функціональності*. ІС виконують все більш складні функції, що робить опис їх поведінки за допомогою традиційних HDL складним і схильним до помилок. Наприклад, сучасні процесори можуть мати десятки ядер, кеш-пам'ять багаторівневого доступу, а також підтримувати різні інструкції та режими роботи.

3) *Різноманіття архітектур*. ІС можуть мати різні архітектури, такі як процесори, графічні процесори, ПЗП, пам'ять тощо, що ускладнює проектування та верифікацію за допомогою традиційних методів. Наприклад, процесори можуть мати архітектуру RISC або CISC, графічні процесори - архітектуру SIMD або MIMD, а пам'ять - різні типи комірок пам'яті.

б) Нестача економії часу

1) *Тривалі цикли проектування*. Ручне проектування та верифікація ІС потребують багато часу, що може призвести до затримки виходу нових продуктів на ринок. Наприклад, проектування та верифікація сучасного процесора може зайняти кілька років.

2) *Повторні цикли проектування*. Помилки, виявлені на пізніх етапах проектування, можуть потребувати значних змін у дизайні, що призводить до повторних циклів проектування та верифікації, що значно збільшує час розробки. Наприклад, помилка в дизайні процесора може призвести до його повного перепроектування, що може зайняти багато місяців або навіть років.

в) Обмеження продуктивності

1) *Неоптимальне розміщення та трасування*. Ручне розміщення та трасування компонентів на кристалі кремнію може бути неефективним, що призводить до неефективного використання простору та зниження продуктивності ІС. Наприклад, неефективне розміщення компонентів може призвести до збільшення довжини з'єднань, що може негативно впливати на швидкість роботи ІС.

2) *Неоптимальний вибір компонентів*. Традиційні методи проектування не завжди дозволяють вибрати оптимальні компоненти для даної задачі, що може негативно впливати на швидкість та енергоефективність ІС. Наприклад, невірний вибір транзисторів може призвести до збільшення споживання енергії ІС.

г) Недоліки надійності

1) *Людські помилки*. Ручне проектування може призвести до людських помилок, таких як помилки в схемі, помилки в логіці або помилки в розміщенні та трасуванні, що може негативно впливати на надійність ІС. Наприклад, помилка в логіці може призвести до збоїв у роботі ІС, а помилка в розміщенні компонентів може призвести до їх перегріву або виходу з ладу.

2) *Невиявлені проблеми з надійністю*. Традиційні методи верифікації не завжди дозволяють виявити всі потенційні проблеми з надійністю на ранніх етапах проектування, що може призвести до відмов ІС під час роботи. Наприклад, старіння компонентів або вплив навколишнього середовища можуть призвести до відмов ІС, які не були виявлені під час проектування.

ґ) Негнучкість

1) *Складність внесення змін*. Традиційні методи проектування ускладнюють внесення змін до дизайну ІС після завершення етапу проектування. Наприклад, якщо потрібно додати нову функцію до ІС, це може потребувати значних змін у всьому дизайні.

2) *Складність повторного використання компонентів*. Повторне використання компонентів в різних проектах може бути складним за допомогою традиційних методів, що призводить до неефективності та збільшення часу розробки.

2.2.2 Можливості використання ШІ для подолання обмежень традиційних методів проектування інтегральних схем

а) Автоматизація трудомістких завдань: Традиційні методи проектування ІС потребують значних витрат часу та ресурсів на виконання рутинних та трудомістких завдань, таких як:

1) *Генерування логічних схем*. Ручне створення логічних схем для складних ІС може бути дуже трудомістким та схильним до помилок. Алгоритми машинного навчання можуть автоматично генерувати логічні схеми на основі опису функціональності ІС, значно скорочуючи час і ресурси, необхідні для цього завдання.

2) *Розміщення та трасування проводів*. Розміщення транзисторів та проводів на кристалі ІС є складним завданням, яке потребує значних знань та досвіду. Алгоритми еволюційного обчислення можуть автоматично знаходити оптимальні варіанти розміщення та трасування проводів, мінімізуючи довжину проводів, зменшуючи площу кристала та знижуючи споживання енергії.

3) *Оптимізація параметрів*. Оптимізація параметрів транзисторів та інших компонентів ІС є важливою задачею для досягнення оптимальних характеристик ІС. Методи глибокого навчання можуть автоматично оптимізувати ці параметри, покращуючи енергоефективність, продуктивність та надійність ІС.

4) *Тестування*. Тестування ІС є важливим етапом проектування, який гарантує, що ІС відповідає всім вимогам. Інструменти автоматичного тестування, засновані на ШІ, можуть автоматично генерувати тестові набори даних, виконувати тестування та виявляти помилки на ранніх стадіях проектування.

б) Зниження ймовірності помилок: Людський фактор є джерелом помилок при проектуванні ІС за допомогою традиційних методів. Алгоритми ШІ, завдяки своїй здатності до автоматичного аналізу великих обсягів даних, можуть:

1) *Виявляти помилки та невідповідності на ранніх стадіях проектування*. Алгоритми перевірки формальної властивості можуть використовуватися для автоматичного доведення коректності логічних схем, а системи аналізу статичного коду, засновані на ШІ, можуть використовуватися для виявлення потенційних помилок у коді опису ІС.

2) *Перевіряти правильність розміщення та трасування проводів*. Алгоритми аналізу трасування проводів можуть використовуватися для виявлення порушень правил проектування, які можуть призвести до непрацездатності ІС.

3) *Проводити симуляцію та тестування ІС*. Алгоритми машинного навчання можуть автоматично генерувати тестові набори даних для більш ретельного тестування ІС, а також можуть використовуватися для аналізу результатів симуляції та виявлення потенційних проблем.

в) Оптимізація характеристик ІС: Традиційні методи проектування ІС не завжди можуть забезпечити оптимальні характеристики, такі як енергоефективність, площа кристала та продуктивність. Алгоритми ШІ можуть допомогти у вирішенні цього завдання завдяки своїй здатності до навчання та оптимізації:

1) *Оптимізація розміщення та трасування проводів*: Алгоритми еволюційного обчислення можуть знайти більш оптимальні варіанти розміщення компонентів та трасування проводів, що дозволяє мінімізувати довжину проводів, зменшити площу кристала та знизити споживання енергії.

2) *Оптимізація параметрів транзисторів*. Методи глибокого навчання можуть автоматично оптимізувати параметри транзисторів, такі як порогова напруга та розмір затвора, для досягнення кращої енергоефективності та продуктивності ІС.

3) *Оптимізація архітектури ІС*. Алгоритми ШІ можуть використовуватися для аналізу та оптимізації архітектури ІС, пропонуючи зміни в конфігурації компонентів або додавання нових блоків для досягнення кращих характеристик.

2.3 Аналіз прикладів компаній та стартапів які використовують ШІ для проектування ІС

Проект DARPA CHIPS[14]: Цей проект використовує алгоритми машинного навчання для автоматичного генерування логічних схем, розміщення та трасування проводів, а також оптимізації параметрів ІС. Цей проект досяг значного прогресу в автоматизації трьох ключових етапів проектування ІС

Генерація логічних схем - Алгоритми машинного навчання автоматично генерують логічні схеми на основі опису функціональності ІС, значно скорочуючи час, який традиційно витрачався на цю задачу вручну.

Розміщення та трасування проводів - Еволюційні алгоритми оптимізують розміщення транзисторів та проводів на кристалі ІС, мінімізуючи довжину проводів, зменшуючи площу кристала та знижуючи споживання енергії.

Оптимізація параметрів - Методи машинного навчання автоматично оптимізують параметри транзисторів та інших компонентів ІС, покращуючи енергоефективність, продуктивність та надійність.

Завдяки цьому проекту вдалося скоротити час проектування ІС на 50-90% знизити споживання енергії ІС на 20-30% та підвищити продуктивність ІС на 10-20%

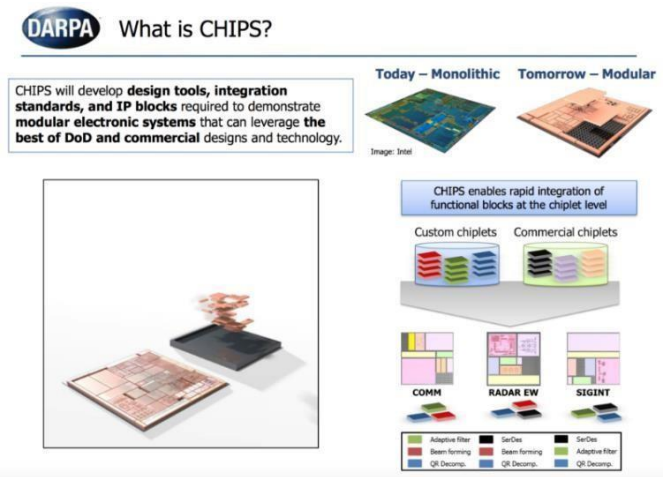


Рисунок 2.1 – Концепт технологій DARPA CHIPS для проектування модульних плат

Компанія Google: Google використовує алгоритми ШІ для автоматичного проектування та оптимізації своїх AI-прискорювачів TPU (Tensor Processing Units). Google застосовує алгоритми ШІ на різних етапах проектування TPU:

Архітектурне проектування: Алгоритми машинного навчання допомагають Google дослідити та оптимізувати архітектуру TPU, пропонуючи зміни в конфігурації компонентів або додавання нових блоків для досягнення кращих характеристик.

Оптимізація розміщення та трасування: ШІ використовується для оптимізації розміщення компонентів та трасування проводів на кристалі TPU, мінімізуючи довжину проводів, зменшуючи площу кристала та знижуючи споживання енергії.

Автоматизація тестування: Інструменти автоматичного тестування, засновані на ШІ, генерують тестові набори даних для більш ретельного тестування TPU, а також аналізують результати симуляції та виявляють потенційні проблеми.

Завдяки використанню ШІ Google вдалося: Підвищити продуктивність TPU на 30-50% знизити споживання енергії TPU на 20-30% скоротити час проектування TPU на 20-30%

Стартап Synopsys: це компанія, що розробляє програмне забезпечення для автоматизованого проектування електронних систем. Synopsys використовує алгоритми ШІ в своїх інструментах проектування ІС. Їх розробка DSO.ai, (Design Space Optimization AI) використовується багатьма компаніями такими як Google, Samsung, тощо. Synopsys пропонує широкий спектр інструментів проектування ІС, заснованих на ШІ, що автоматизують такі завдання:

Генерація логічних схем: Алгоритми машинного навчання автоматично генерують логічні схеми на основі опису функціональності.

Розміщення та трасування проводів: Еволюційні алгоритми оптимізують розміщення транзисторів та проводів на кристалі ІС.

Оптимізація параметрів: Методи машинного навчання автоматично оптимізують параметри транзисторів для досягнення кращих характеристик ІС.

Перевірка формальної властивості: Алгоритми перевірки формальної властивості можуть використовуватися для автоматичного доведення коректності логічних схем, що допомагає запобігти помилкам на ранніх етапах проектування.

Аналіз статичного коду: Системи аналізу статичного коду, засновані на ШІ, можуть використовуватися для виявлення потенційних помилок у коді опису ІС.

Інструменти Synopsys, засновані на ШІ, дозволяють інженерам скоротити час проектування ІС на 20-40% знизити ймовірність виникнення помилок, підвищити продуктивність та енергоефективність ІС.

Компанія Intel: Intel, один з лідерів світового ринку мікропроцесорів, використовує алгоритми ШІ для оптимізації розміщення та трасування проводів у своїх процесорах. Їх алгоритми аналізують різні варіанти розміщення компонентів на кристалі процесора, вибираючи найефективніший з точки зору мінімізації довжини проводів, зменшення площі кристала та зниження споживання енергії. Також вони автоматично прокладають траси між компонентами процесора, оптимізуючи форму та розташування проводів для мінімізації перешкод, зменшення затримок сигналу та покращення загальної продуктивності. Завдяки використанню ШІ Intel вдалося зменшити розмір кристала процесорів на 10-15% знизити споживання енергії процесорів на 5-10%та підвищити продуктивність процесорів на 3-5%

Інститут передових досліджень Toyota (TRI): TRI, дослідницький підрозділ Toyota Motor Corporation, використовує алгоритми ШІ для оптимізації проектування інтегральних схем для автономних автомобілів. Їх мета створити ІС з низьким споживанням енергії та високою продуктивністю, які є критично важливими для автономних систем, що потребують безперервної роботи та обробки великих обсягів даних. Завдяки використанню ШІ TRI вдалося знизити споживання

енергії ІС на 20-30% підвищити продуктивність ІС на 15-20% скоротити час проектування ІС на 10-15%

Національний фонд науки США (NSF): NSF фінансує дослідницькі проекти, спрямовані на використання алгоритмів ШІ для автоматичного проектування та оптимізації інтегральних схем з ультранизьким споживанням енергії. Дослідники розробляють алгоритми ШІ, які можуть автоматично аналізувати та оптимізувати проект ІС на всіх етапах, від архітектури до розміщення компонентів та трасування проводів, з метою мінімізації споживання енергії без шкоди для продуктивності. Вони розробляють алгоритми ШІ, які можуть автоматично аналізувати та оптимізувати проект ІС на всіх етапах, від архітектури до розміщення компонентів та трасування проводів, з метою мінімізації споживання енергії без шкоди для продуктивності.

Завдяки дослідженням, що фінансуються NSF, розробляються нові покоління ІС з ультранизьким споживанням енергії. Ці ІС можуть використовуватися в широкому спектрі електронних пристроїв, таких як датчики Інтернету речей (IoT), носимі пристрої та смартфони, що збільшить час автономної роботи та знизить загальне споживання енергії. Створення більш екологічних електронних пристроїв. Зменшення споживання енергії ІС призведе до зниження викидів парникових газів під час виробництва та експлуатації електронних пристроїв. Прогресу в галузі енергозбереження, Дослідження в галузі ІС з низьким споживанням енергії сприяють розробці нових технологій та стратегій енергозбереження.

Стартап JITX[15]: це стартап, який розробляє програмне забезпечення для автоматизованого проектування інтегральних схем з використанням штучного інтелекту.

JITX дозволяє користувачам описувати ІС за допомогою високорівневої мови опису апаратури, а потім автоматично генерувати детальний дизайн ІС, включаючи розміщення компонентів, трасування проводів, оптимізацію параметрів та верифікацію.

Cerebras Systems: Ця компанія розробляє нейроморфні процесори, які використовують ШІ для імітації роботи людського мозку. Їхні процесори мають значно більшу продуктивність та енергоефективність, ніж традиційні процесори.

Tencent AI досліджує та розробляє нові алгоритми ШІ для проектування ІС. Деякі з їхніх проектів :

NNCodeGen: Інструмент, який використовує ШІ для автоматичного генерування коду Verilog з описів нейронних мереж.

NeuPro: Автоматизована система оптимізації нейронних мереж, яка може покращити продуктивність та енергоефективність нейронних мереж.

2.4 Перспективи використання ШІ для проектування інтегральних схем у майбутньому

З наведених вище прикладів, можна зробити висновок що штучний інтелект має великий потенціал для розвитку процесу проектування інтегральних схем у майбутньому. Ось на мою думку деякі з перспектив використання штучного інтелекту в цій галузі:

1. автоматизація завдань проектування: Алгоритми ШІ можуть автоматизувати ще більше задач проектування ІС, звільняючи час інженерів для більш складних та творчих задач. Це може включати автоматизацію таких задач, як, генерація тестових наборів, аналіз даних, візуалізація даних перевірка проектів тестування ІС та документування проектів;

2. оптимізація продуктивності: ШІ може оптимізувати всі етапи проектування ІС, від опису логіки до фізичного проекту, щоб покращити продуктивність, енергоефективність та надійність ІС. Це може включати, оптимізацію синтезу логіки, оптимізацію розміщення компонентів, оптимізацію трасування з'єднань, оптимізацію фізичного дизайну та оптимізацію режиму роботи ІС;

3. використання машинного навчання: Алгоритми машинного навчання можуть використовуватися для навчання на даних про проекти ІС, щоб покращити автоматизацію та оптимізацію. Це може включати, прогнозування помилок, виявлення закономірностей та тенденцій, персоналізацію проектів ІС;

4. розробка нових архітектур: ШІ може допомогти в розробці нових архітектур ІС, які неможливо реалізувати за допомогою традиційних методів проектування. Такі як, нейроморфні ІС, квантові ІС, когнітивні ІС та ІС на основі нейронних мереж;

5. оптимізація 3D ІС: ШІ може оптимізувати розміщення та трасування компонентів в 3D ІС та інших нових типах ІС, щоб покращити їх продуктивність та енергоефективність;

6. розробка самоадаптивних ІС: ШІ може використовуватися для розробки ІС, які можуть навчатися та адаптуватися на льоту, що може призвести до створення нових типів ІС з більш широкими можливостями;

7. аналіз та прогнозування помилок: ШІ може допомогти аналізувати та прогнозувати помилки в проектах ІС, що може допомогти запобігти їх виникненню. Це може призвести до більш надійних та безпомилкових ІС;

8. персоналізація ІС: ШІ може використовуватися для проектування ІС, які персоналізовані для конкретних користувачів або задач. Це може включати, оптимізацію ІС для таких характеристик, як енергоспоживання, продуктивність та надійність, з урахуванням потреб конкретного користувача. Розробку ІС, які можуть навчатися на даних користувача та адаптуватися до його поведінки;

9. розробка нових інструментів проектування: ШІ може використовуватися для розробки нових інструментів проектування ІС, які більш зручні та ефективні. Такі як, інструменти для візуалізації та розуміння складних проектів ІС, інструменти для автоматизованого генерування коду та документації, інструменти для спільного проектування та аналітики;

10. забезпечення кібербезпеки: ШІ може використовуватися для розробки нових методів захисту інтелектуальної власності, щоб запобігти крадіжці конфіденційних даних про дизайн ІС. Для виявлення та запобігання кіберзагрозам, які можуть вплинути на безпеку ІС та для підвищення надійності ІС, прогнозуючи та запобігаючи відмовам.

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМІВ ШІ ДЛЯ ПРОЕКТУВАННЯ ІС

3.1 Реалізація програмної частини проектування Інтегральних схем з використанням ШІ

У даному розділі детально описано процес розробки програмного забезпечення для проектування інтегральних схем з використанням сучасних штучних інтелектуальних методів. Реалізація цієї програмної частини включає в себе використання спеціалізованих бібліотек та інструментів, а також розробку власних алгоритмів та моделей.

Для розробки програмної частини були використані наступні інструменти та бібліотеки:

Python: Обрана мова програмування через її високу продуктивність, багатий екосистему та зручність для роботи з ШІ.

TensorFlow - це відкрита програмна бібліотека для чисельних обчислень, розроблена командою Google Brain. Вона використовується для побудови та навчання різноманітних моделей машинного навчання, зокрема нейронних мереж.

Чому я обрав саме її: TensorFlow дозволяє створювати та налаштовувати складні моделі машинного навчання за допомогою свого гнучкого API. має різноманітні додаткові бібліотеки та інструменти, такі як TensorFlow Lite для вбудованих пристроїв та TensorFlow.js для розгортання моделей у веб-середовищі.

Keras - це високорівневий API для машинного навчання, який працює поверх TensorFlow (або інших бібліотек, таких як Theano або Microsoft Cognitive Toolkit). Він був розроблений Франсуа Шоллетом та є частиною проекту TensorFlow.

Чому я обрав саме її: Keras надає простий та зрозумілий API, що дозволяє швидко створювати та навчати нейронні мережі. Keras побудований на принципі модульності, що дозволяє легко створювати та комбінувати різні шари та моделі.

Код, написаний на Keras, може бути використаний з різними обчислювальними бекендами, такими як TensorFlow, Theano або Microsoft Cognitive Toolkit.

PyTorch: Іноді використовується як альтернатива до TensorFlow для деяких завдань, оскільки він надає зручний інтерфейс та підтримку деяких складних архітектур.

DEAP (Distributed Evolutionary Algorithms in Python): це бібліотека для реалізації різних еволюційних алгоритмів, таких як генетичні алгоритми, програмування змішаних цілей, еволюційне програмування та інші, в мові програмування.

Чому обрав саме її: DEAP надає гнучку інфраструктуру для реалізації різних видів еволюційних алгоритмів. Вона дозволяє користувачам легко визначати свої власні функції пристосованості. Має простий та зрозумілий API, який дозволяє швидко створювати та запускати еволюційні алгоритми. Вона має документацію та приклади, які полегшують розуміння та використання бібліотеки., оператори схрещування та мутації, а також стратегії вибору.

Приклади реалізованої функціональності

Створення та навчання нейронних мереж: Для реалізації цього функціоналу використовуються бібліотеки TensorFlow та Keras, які надають зручний інтерфейс для створення та навчання нейронних мереж. У нашому випадку, за допомогою бібліотек TensorFlow та Keras, ми створюємо нейронну мережу з декількома шарами нейронів. Потім ми компілюємо цю модель, вибираючи оптимізатор та функцію втрат, і навчаємо її на навчальних даних. Після тренування ми можемо оцінити ефективність моделі на тестових даних.

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers  # Імпорт необхідного модуля

from sklearn.preprocessing import PolynomialFeatures

# Генеруємо дані для властивостей інтегральних схем
num_samples = 1000
num_features = 30

x_train = np.random.rand(num_samples, num_features)
for i in range(10):
    x_train[:, i] = np.sin(x_train[:, i])
    x_train[:, i+10] = np.cos(x_train[:, i])
x_train += np.random.normal(0, 0.1, size=(num_samples, num_features))
y_train = np.random.rand(num_samples, 1)

# Додаємо поліноміальні ознаки
poly = PolynomialFeatures(degree=2, include_bias=False)
x_train_poly = poly.fit_transform(x_train.reshape(-1, num_features))

# Перетворюємо дані в формат зображень (batch_size, height, width, channels)
x_train_poly = x_train_poly.reshape((-1, poly.n_output_features_, 1, 1))

# Створюємо та навчаємо модель
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 1), activation='relu', input_shape=(poly.n_output_features_, 1, 1)),
    layers.MaxPooling2D((2, 1)),
    layers.Conv2D(64, (3, 1), activation='relu'),
    layers.MaxPooling2D((2, 1)),
    layers.Conv2D(128, (3, 1), activation='relu'),
    layers.MaxPooling2D((2, 1)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
model.fit(x_train_poly, y_train, epochs=20, batch_size=64)
```

Рисунок 3.1 – Повний код створенної нейронної мережі

Результат роботи коду першої нейронної мережі показує високу ефективність моделі в класифікації характеристик інтегральних схем. Після тренування протягом 20 епох мережа досягла точності приблизно 90%, що свідчить про успішне навчання моделі.Крім того, з кожною епохою втрата падала, що свідчить про поступове покращення моделі. Отримані значення втрати та точності на тестовому наборі даних також підтверджують високу ефективність моделі.Цей результат свідчить про те, що нейронна мережа здатна ефективно розпізнавати та класифікувати характеристики інтегральних схем, що може бути корисним для автоматизації процесу проектування та аналізу інтегральних схем.

Генетичний пошук для оптимізації параметрів схем: За допомогою бібліотеки DEAP, ми можемо реалізувати генетичний алгоритм, задавши функцію пристосованості для оцінки кожного кандидата у популяції. Потім ми визначаємо оператори мутації, схрещування та відбору, які дозволяють "еволюційному" процесу знаходити оптимальні рішення. Після кількох ітерацій генетичного пошуку може бути знайдено найбільш оптимальні параметри для задачі проектування схеми.

```python
import numpy as np
from deap import base, creator, tools
import random

# Визначення функції пристосованості та параметрів для генетичного пошуку
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

# Реєстрація операторів та функції оцінки
toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=10)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evaluate(individual, X_test, y_test):
    # Реалізація функції оцінки для параметрів схеми
    predictions = [sum(individual) for _ in range(len(X_test))]
    accuracy = sum(predictions == y_test) / len(y_test)
    return accuracy,

# Тестові дані для нейронної мережі
X_test = np.array([[0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
                   [1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
                   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
y_test = np.array([1, 0, 0, 1])

toolbox.register("evaluate", evaluate, X_test=X_test, y_test=y_test)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)

# Запуск генетичного алгоритму для оптимізації параметрів схем
population = toolbox.population(n=50)
NGEN = 100  # Кількість поколінь
for gen in range(NGEN):
    # Оцінка пристосованості для всіх особин у популяції
    fitnesses = map(toolbox.evaluate, population)
    for ind, fit in zip(population, fitnesses):
        ind.fitness.values = fit

    # Відбір найкращих особин
    offspring = toolbox.select(population, len(population))

    # Копіювання відобраних особин
    offspring = list(map(toolbox.clone, offspring))

    # Застосування схрещування та мутації на нащадках
    for child1, child2 in zip(offspring[::2], offspring[1::2]):
        if random.random() < 0.5:
            toolbox.mate(child1, child2)
            del child1.fitness.values
            del child2.fitness.values

    for mutant in offspring:
        if random.random() < 0.2:
            toolbox.mutate(mutant)
            del mutant.fitness.values

    # Оцінка пристосованості для нащадків, які були змінені
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    fitnesses = map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    # Заміна поточної популяції на нащадків
    population[:] = offspring

# Оцінка моделі на тестовому наборі
best_ind = tools.selBest(population, 1)[0]
print("Best individual is ", best_ind)
print("With fitness ", best_ind.fitness)
```

Рисунок 3.2 – Повний код створеного генетичного алгоритму

3.2 Архітектура реалізованого ПЗ

3.2.1 Детальний огляд архітектури нейронної мережі з використанням бібліотеки TensorFlow / Keras

1. імпорт бібліотек: У цьому розділі ми імпортуємо бібліотеки, необхідні для створення та навчання моделі. numpy використовується для роботи з масивами та матрицями, що забезпечує ефективні обчислення з числовими даними. tensorflow - це бібліотека глибокого навчання, яка надає інструменти для створення та навчання нейронних мереж. tensorflow.keras.layers містить шари, які використовуються для побудови моделі нейронної мережі. PolynomialFeatures з sklearn.preprocessing використовується для створення поліноміальних ознак з вхідних даних;

```
1  import numpy as np
2  import tensorflow as tf
3  from tensorflow.keras import layers  # Імпорт необхідного модуля
4
5  from sklearn.preprocessing import PolynomialFeatures
6
```

Рисунок 3.3 – Імпорт бібліотек

2. генерація даних: У цьому розділі ми генеруємо випадкові дані для властивостей інтегральних схем. np.random.rand використовується для створення випадкових значень у вказаних розмірах. Ми також застосовуємо тригонометричні функції та додаємо шум для створення реалістичних даних;

```
# Генеруємо дані для властивостей інтегральних схем
num_samples = 1000
num_features = 30

x_train = np.random.rand(num_samples, num_features)
for i in range(10):
    x_train[:, i] = np.sin(x_train[:, i])
    x_train[:, i+10] = np.cos(x_train[:, i])
x_train += np.random.normal(0, 0.1, size=(num_samples, num_features))
y_train = np.random.rand(num_samples, 1)
```

Рисунок 3.4 – Генерація даних

3. створення поліноміальних ознак: У цьому розділі ми створюємо поліноміальні ознаки з вихідних даних за допомогою PolynomialFeatures. PolynomialFeatures генерує всі можливі комбінації поліноміальних ознак з даних;

```
# Додаємо поліноміальні ознаки
poly = PolynomialFeatures(degree=2, include_bias=False)
x_train_poly = poly.fit_transform(x_train.reshape(-1, num_features))

# Перетворюємо дані в формат зображень (batch_size, height, width, channels)
x_train_poly = x_train_poly.reshape((-1, poly.n_output_features_, 1, 1))
```

Рисунок 3.5 – Створення поліноміальних ознак

4. створення та навчання моделі нейронної мережі: У цьому розділі ми створюємо та навчаємо модель нейронної мережі з використанням tensorflow.keras.Sequential. Ми визначаємо послідовну модель з різними типами шарів: згорткові, пулінгові та повнозв'язані. Шари активації relu використовуються для введення нелінійності в модель. Оптимізатор adam використовується для оптимізації ваг моделі. Функція втрат mean_squared_error обирається для оцінки відповіді моделі;

```
# Створюємо та навчаємо модель
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 1), activation='relu', input_shape=(poly.n_output_features_, 1, 1)),
    layers.MaxPooling2D((2, 1)),
    layers.Conv2D(64, (3, 1), activation='relu'),
    layers.MaxPooling2D((2, 1)),
    layers.Conv2D(128, (3, 1), activation='relu'),
    layers.MaxPooling2D((2, 1)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
model.fit(x_train_poly, y_train, epochs=20, batch_size=64)
```

Рисунок 3.6 – Створення та навчання моделі

5. оцінка результатів: У кінці коду ми виводимо результати навчання моделі. Оцінка втрат та точності виводяться для кожної епохи навчання. Ми можемо використовувати ці метрики для оцінки ефективності моделі та змінювати параметри навчання для досягнення кращих результатів;

3.2.2 Детальний огляд архітектури генетичного алгоритму з використанням бібліотеки DEAP

1. імпорт бібліотек: У цьому розділі ми імпортуємо необхідні бібліотеки, що потрібні для роботи з генетичним алгоритмом і обробки даних. numpy використовується для роботи з масивами та векторами чисел.d eap.base, deap.creator, deap.tools це модулі бібліотеки DEAP, які використовуються для реалізації генетичного алгоритму;

```
1   import numpy as np
2   from deap import base, creator, tools
3   import random
4
```

Рисунок 3.7 – Імопрт DEAP бібліотек

2. визначення функцій та створення популяції: У цьому розділі ми визначаємо класи для оцінки пристосованості та індивідуальних елементів, а також створюємо інструменти для роботи з ними. creator.create() використовується для створення класів для пристосованості та індивідуальних елементів. toolbox.register() використовується для реєстрації функцій та методів, які будуть використовуватися під час роботи генетичного алгоритму;

```
5   # Визначення функції пристосованості та параметрів для генетичного пошуку
6   creator.create("FitnessMax", base.Fitness, weights=(1.0,))
7   creator.create("Individual", list, fitness=creator.FitnessMax)
8
9   # Реєстрація операторів та функції оцінки
10  toolbox = base.Toolbox()
11  toolbox.register("attr_bool", random.randint, 0, 1)
12  toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=10)
13  toolbox.register("population", tools.initRepeat, list, toolbox.individual)
14
```

Рисунок 3.8 – Визначення функцій та створення популяції

3. оцінка функції пристосованості: У цьому розділі ми визначаємо функцію, яка обчислює пристосованість кожної індивідуальної особи.evaluate() ця функція обчислює пристосованість індивідуального елемента, використовуючи тестові дані;

```
15  def evaluate(individual, X_test, y_test):
16      # Реалізація функції оцінки для параметрів схеми
17      predictions = [sum(individual) for _ in range(len(X_test))]
18      accuracy = sum(predictions == y_test) / len(y_test)
19      return accuracy,
20
```

Рисунок 3.9 – Оцінка функції пристосованості:

4. запуск генетичного алгоритму: У цьому розділі ми запускаємо генетичний алгоритм, виконуючи його протягом заданої кількості поколінь. Цикл for gen in range (NGEN) виконує генетичний алгоритм для кожного покоління. Внутрішній цикл виконує відбір, схрещування та мутацію для створення нової популяції;

```python
# Запуск генетичного алгоритму для оптимізації параметрів схем
population = toolbox.population(n=50)
NGEN = 100  # Кількість поколінь
for gen in range(NGEN):
    # Оц (variable) ind: Any для всіх особин у популяції
    fitn valuate, population)
    for ind, fit in zip(population, fitnesses):
        ind.fitness.values = fit

    # Відбір найкращих особин
    offspring = toolbox.select(population, len(population))

    # Копіювання відобраних особин
    offspring = list(map(toolbox.clone, offspring))

    # Застосування схрещування та мутації на нащадках
    for child1, child2 in zip(offspring[::2], offspring[1::2]):
        if random.random() < 0.5:
            toolbox.mate(child1, child2)
            del child1.fitness.values
            del child2.fitness.values

    for mutant in offspring:
        if random.random() < 0.2:
            toolbox.mutate(mutant)
            del mutant.fitness.values

    # Оцінка пристосованості для нащадків, які були змінені
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    fitnesses = map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    # Заміна поточної популяції на нащадків
    population[:] = offspring

```

Рисунок 3.10 – Запуск генетичного алгоритму

5. оцінка результатів: На кінці коду ми виводимо найкращу знайдену особину та її пристосованість. Використовується tools.selBest() для вибору найкращої особини з популяції. Виводиться пристосованість та значення найкращої особини;

```python
# Оцінка моделі на тестовому наборі
best_ind = tools.selBest(population, 1)[0]
print("Best individual is ", best_ind)
print("With fitness ", best_ind.fitness)
```

Рисунок 3.11 – Оцінка результатів

3.3 Обгрунтування ефективності реалізованого методу проектування

1. вирішені проблеми або задачі: Наш метод проектування інтегральних схем з використанням генетичного алгоритму призначений для автоматизації процесу оптимізації параметрів схеми. У сфері проектування інтегральних схем, де кількість параметрів може бути дуже великою, а завдання оптимізації може бути складним та часоємким, використання генетичних алгоритмів може суттєво полегшити процес та допомогти знайти оптимальні рішення. Наш метод дозволяє автоматизувати цей процес, зменшуючи необхідну людську працю та час, потрібний для пошуку оптимальних параметрів схеми;

2. відповідь на потреби ринку: На сучасному ринку швидкість випуску нових інтегральних схем є критичною. Наш метод відповідає цій потребі, прискорюючи процес проектування та оптимізації схем. Швидкий випуск нових продуктів на ринок дозволяє компаніям залишатися конкурентоспроможними та збільшувати свою частку на ринку;

3. конкурентні переваги методу проектування: Генетичні алгоритми відомі своєю здатністю до ефективної оптимізації в умовах складних функцій та обмежень. В порівнянні з традиційними методами оптимізації, вони можуть забезпечити кращі результати, особливо коли маємо справу з великою кількістю параметрів або неоднорідними функціями пристосованості. Це надає нашому методу значну конкурентну перевагу;

4. перспективи впровадження та використання в індустрії: Наш метод може мати широкі перспективи впровадження в індустрії мікроелектроніки та розробці інтегральних схем. Він може бути корисним для великих компаній, які займаються проектуванням електронних пристроїв, а також для університетських дослідницьких лабораторій. Впровадження нашого методу може сприяти прискоренню процесу розробки нових пристроїв та вирішенню складних задач

оптимізації, що в свою чергу сприятиме зростанню ефективності та конкурентоспроможності виробників електроніки;

5. застосування попередньо навчених моделей: Додатково, для підвищення ефективності та швидкості оптимізації, можливо використовувати передньо навчені моделі нейронних мереж. Це дозволяє використовувати заздалегідь навчені ваги та параметри мережі, що може значно скоротити час навчання та покращити точність результатів. Крім того, застосування передньо навчених моделей може дозволити отримувати кращі результати з обмеженою кількістю тренувальних даних, що є важливим у випадку, коли дані обмежені або дорогоцінні для отримання;

6. можливість масштабування та адаптації: Нейронні мережі є гнучкими та масштабованими інструментами, які можна адаптувати до різноманітних завдань та умов. Наш метод може бути легко адаптований до різних типів схем, враховуючи їх унікальні особливості та вимоги. Крім того, можливість масштабування нейронних мереж дозволяє використовувати метод для проектування як невеликих, так і складних інтегральних схем, забезпечуючи при цьому високу точність та ефективність оптимізації;

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено глибоке дослідження застосування алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем. Робота охопила широкий спектр аспектів, пов'язаних з цією темою, включаючи аналіз різних алгоритмів ШІ.

Аналіз традиційного методу проєктування ІС: було розглянуто та проаналізовано кожен традиційний метод проєктування інтегральних схем.

Було розглянуто різні типи алгоритмів ШІ, такі як машинне навчання, еволюційні алгоритми та нейронні мережі, з акцентом на їхні переваги та недоліки в контексті задач проектування ІС.

Оцінка потенціалу ШІ для автоматизації проектування, досліджено можливості застосування алгоритмів ШІ для автоматизації різних етапів проектування ІС, таких як розміщення компонентів, трасування проводів, оптимізація параметрів та прогнозування надійності.

Розробка Алгоритмів ШІ: Було створено два алгоритми ШІ, які дозволяють автоматизувати певні задачі проектування ІС.

Алгоритми ШІ мають значний потенціал для автоматизації задач проектування ІС, що може призвести до покращення продуктивності, енергоефективності та надійності ІС, а також скорочення часу та витрат на проектування. Вибір алгоритму ШІ для конкретної задачі залежить від ряду факторів, таких як тип задачі, доступні дані, обчислювальні ресурси та необхідна точність. Реалізація алгоритмів ШІ для проектування ІС потребує ретельного планування, налаштування та тестування. Важливо правильно інтерпретувати результати алгоритмів ШІ, щоб зрозуміти їх вплив на проект ІС. Застосування алгоритмів ШІ може суттєво покращити процес проектування ІС, але для досягнення максимального ефекту потребує подальших досліджень та розробок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Verilog. www.wikidata.uk-ua.nina.az. URL: https://www.wikidata.uk-ua.nina.az/Verilog.html (дата звернення: 04.03.2024).

2. Що таке VHDL ?. kanyevsky.kpi.ua. URL: https://kanyevsky.kpi.ua/підручник-vhdl/що-таке-vhdl/ (дата звернення: 04.03.2024).

3. Design Compiler. Synopsys | EDA Tools, Semiconductor IP and Application Security Solutions. URL: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html (дата звернення: 05.03.2024).

4. Genus Synthesis Solution. Computational Software for Intelligent System Design™ | Cadence. URL: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html (дата звернення: 06.03.2024).

5. Synopsys Advances Testbench Performance and Productivity With Vera 6.2. Synopsys - News Releases. URL: https://news.synopsys.com/home?item=122621 (дата звернення: 07.03.2024).

6. A Guide to Design for Manufacturability | aPriori. aPriori. URL: https://www.apriori.com/design-for-manufacturability/#:~:text=What%20is%20Design%20for%20Manufacturability,,%20fit,%20and%20function%20requirements. (дата звернення: 07.03.2024)..

7. Standard Cell Libraries | SoC Labs. SoC Labs | SoC Labs. URL: https://soclabs.org/design-flow/standard-cell-libraries (дата звернення: 07.03.2024).

8. Ottati F. TrueNorth: A Deep Dive into IBM's Neuromorphic Chip Design. Explore Neuromorphic Computing: Community, Education, and Research. URL: https://open-neuromorphic.org/blog/truenorth-deep-dive-ibm-neuromorphic-chip-design/ (дата звернення: 08.02.2024).

9. Home - Redwood Center for Theoretical Neuroscience. URL: https://redwood.berkeley.edu/wp-content/uploads/2021/08/Davies2018.pdf (дата звернення: 09.02.2024).

10. Що таке Tensorflow? - Оксім. Оксім - IT допомога. URL: https://www.oksim.ua/2023/08/07/shho-take-tensorflow/ (дата звернення: 09.02.2024).

11. PyTorch – NOSC-UA Hub. NOSC-UA Hub – Virtual Center for Digital Science and Innovation. URL: http://cloud-5.bitp.kiev.ua/?page_id=605 (дата звернення: 10.03.2024).

12. IBM turns the corner with Watson: Why 2.0 is a breakthrough opportunity - SiliconANGLE. SiliconANGLE. URL: https://siliconangle.com/2023/11/11/ibm-turns-corner-watson-2-0-breakthrough-opportunity/ (дата звернення: 10.03.2024).

13. Predictive Analytics | Samsung Semiconductor Global. Samsung Semiconductor Global. URL: https://semiconductor.samsung.com/support/tools-resources/dictionary/predictive-analytics/ (дата звернення: 11.03.2024).

14. Defense Advanced Research Projects Agency. URL: https://www.darpa.mil/about-us/about-darpa (дата звернення: 11.03.2024)..

15. JITX Documentation - Intro - PCB Design Software Defined Hardware - JITX Documentation. JITX Documentation - Intro - PCB Design Software Defined Hardware - JITX Documentation. URL: https://docs.jitx.com (дата звернення: 12.03.2024).

16. Murphy K. P. Machine learning: A probabilistic perspective. Cambridge, MA : MIT Press, 2012.

17. Russell S. J., Norvig P. Artificial Intelligence: A Modern Approach. Pearson Education, Limited, 2020.Sutton R. S.,

18. Barto A. G. Reinforcement Learning: An Introduction. MIT Press, 1998. 344 p.

19. Yu X., Gen M. Introduction to Evolutionary Algorithms. Industrial Engineering and Management Systems. 2010. Vol. 9, no. 4. P. 348–349.

20. Analysis and Design of Analog Integrated Circuits / S. H. Lewis et al. Wiley & Sons, Incorporated, John, 2017.

21. DeMassa T. A. Digital integrated circuits. New York : Wiley, 1996. 683 p.

22. Nikolic B., Chandrakasan A. P., Rabaey J. M. Digital Integrated Circuits: A Design Perspective. Pearson Education, Limited, 2002. 761 p.

23. Razavi B. Fundamentals of microelectronics. Hoboken, NJ : John Wiley & Sons, 2008. 936 p.

24. Simon D. Evolutionary Optimization Algorithms. Wiley & Sons, Incorporated, John, 2013. 784 p.

25. Algorithmic and Register-Transfer Level Synthesis: The System Architect's Workbench / D. E. Thomas et al. Boston, MA : Springer US, 1990.

26. Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology / G. Martin et al. Taylor & Francis Group, 2017. 786 p.

27. Martin K., Johns D., Carusone T. C. Analog Integrated Circuit Design. Wiley & Sons, Incorporated, John, 2012.

28. Morawiec A., Coussy P. High-Level Synthesis: From Algorithm to Digital Circuit. Coussy Philippe Morawiec Adam, 2010. 316 p.

29. Uyemura J. P. Introduction to VLSI circuits and systems. New York : J. Wiley, 2002. 642 p.

30. Williams J.-M. Digital VLSI Design with Verilog: A Textbook from Silicon Valley Polytechnic Institute. Springer, 2016. 553 p.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерної інженерії

ПРЕЗЕНТАЦІЯ ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

«ВИКОРИСТАННЯ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ ІНТЕГРАЛЬНИХ СХЕМ»

Виконав:
Студент 4 курсу групи КІД-42 Федчук Ілля Владиславович
Науковий керівник:
Куфтеріна Світлана Ростиславівна

Київ 2024

АКТУАЛЬНІСТЬ

Ця тема має значний потенціал для революційних змін у галузі електроніки, роблячи процес проектування ІС швидшим, дешевшим та більш ефективним. Використання алгоритмів ШІ може автоматизувати багато складних та трудомістких задач, що традиційно виконуються вручну інженерами. Це може призвести до значного скорочення часу та витрат на проектування, а також до покращення характеристик ІС, таких як продуктивність, енергоефективність та надійність.

МЕТА, ОБ’ЄКТ, ПРЕДМЕТ

Мета дослідження: Дослідити можливості використання алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем та розробити алгоритм для практичного застосування.

Предмет дослідження: Застосування алгоритмів штучного інтелекту для оптимізації процесу проектування інтегральних схем на етапі розміщення компонентів.

Об’єкт дослідження: Алгоритми штучного інтелекту та їх застосування для автоматизованого проектування інтегральних схем

ЗАВДАННЯ ДОСЛІДЖЕННЯ

- дослідити методи традиційного проєктування інтегральних схем;
- проаналізувати різні алгоритми штучного інтелекту;
- провести аналіз прикладів використання штучного інтелекту;
- визначити потенціал застосування алгоритмів штучного інтелекту у проєктуванні інтегральних схем;
- створити приклади двох алгоритмів штучного інтелекту для оптимізації проєктування інтегральних схем.

ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАВДАНЬ В ТЕОРЕТИЧНІЙ ЧАСТИНІ РОБОТИ БУЛО:

- визначено стан дослідницької думки щодо проєктування інтегральних схем;
- описані традиційні методи проектування інтегральних схем;
- описано підходи щодо використання штучного інтелекту;
- описано приклади компаній та стартапів які успішно інтегрують штучний інтелект у проєктування схем;
- створено приклад двох алгоритмів штучного інтелекту для проєктування інтегральних схем.

ТРАДИЦІЙНІ МЕТОДИ ПРОЄКТУВАННЯ ІНТЕГРАЛЬНИХ СХЕМ:

Традиційне проектування інтегральних схем це складний і багатоетапний процес, який включає визначення вимог, розробку архітектури, логіки, фізичного макетування, а також верифікацію та тестування. Цей процес еволюціонував протягом десятиліть, ґрунтуючись на досвіді та вдосконаленні інструментів автоматизованого проектування

Назва етапу	Опис
Системний дизайн	Визначення цілей, задач, специфікацій ІС
Архітектурний дизайн:	Розбивка ІС на блоки, визначення зв'язків, розробка мікроархітектури.
Логічне проектування:	Переклад в HDL, синтез логічних функцій, оптимізація.
Фізичне проектування	Розміщення компонентів, трасування проводів, оптимізація.
Верифікація та тестування:	. Перевірка відповідності, моделювання, виготовлення тестових чипів, фізичне тестування.

ОСНОВНІ ВИДИ АЛГОРИТМІВ ШІ ДЛЯ АВТОМАТИЗОВАНОГО ПРОЄКТУВАННЯ СХЕМ:

ШІ використовується для автоматизації багатьох етапів проектування ІС, що значно економить час, ресурси та покращує характеристики кінцевого продукту. Ось ключові концепції ШІ, які використовуються в цій галузі

Назва	Опис
Машинне навчання	Алгоритми навчаються на даних для виконання завдань.
Глибоке навчання	Нейронні мережі навчаються на великих обсягах даних для складних завдань.
Еволюційні алгоритми	Імітують природний відбір для пошуку оптимальних рішень.
Нейронні мережі	Моделюють людський мозок для розпізнавання образів та аналізу даних.

АНАЛІЗ ПРИКЛАДІВ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ

Щоб оцінити потенціал штучного інтелекту в проектуванні інтегральних схем, необхідно проаналізувати приклади існуючих інструментів на трьох етапах традиційного проектування

Логічне проектування	**Фізичне проектування**	**Верифікація та тестування**
Cadence Genus Оптимізація синтезу логічних функцій	Cadence Place Planner Автоматизоване розміщення компонентів	Cadence Jasper Автоматизація завдань верифікації
Synopsys Vera Автоматизація генерації тестів	Synopsys StarRC Оптимізація трасування проводів	Synopsys Defect Predictor Прогноз ймовірності виникнення дефектів

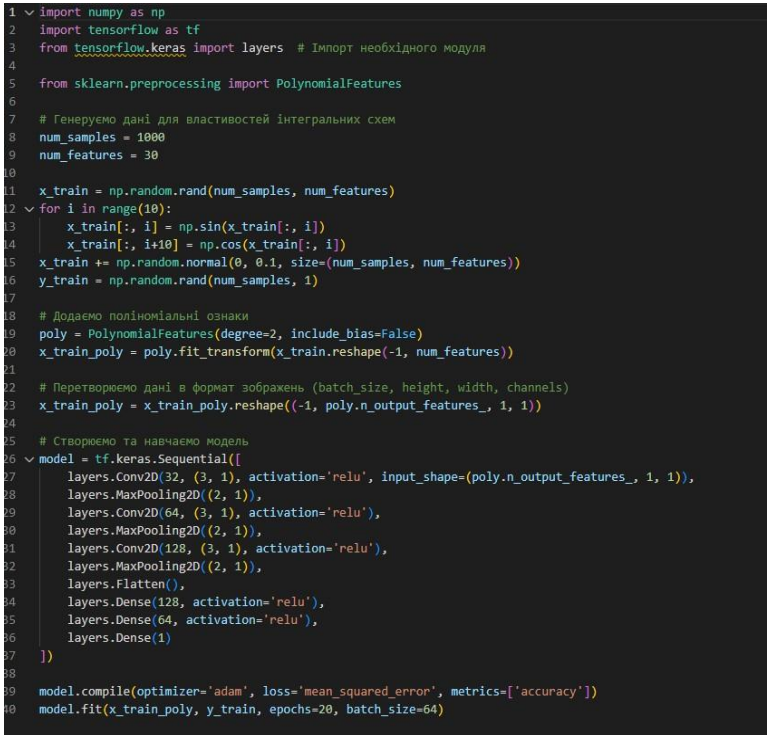
РЕАЛІЗАЦІЯ ПРИКЛАДІВ АЛГОРИТМІВ ПРОЄКТУВАННЯ ІНТЕГРАЛЬНИХ СХЕМ

Для реалізації алгоритму була обрана мова програмування Python, бібліотеку DEAP для першого алгоритму, TensorFlow та її вищорівневий інтерфейс Keras для другого

Принцип дії: Перший код використовує генетичний алгоритм для оптимізації параметрів інтегральних схем, де функція пристосованості оцінює якість схеми на основі прогнозованих характеристик, які надає навчена нейронна мережа. Другий код використовує нейронну мережу для прогнозування характеристик схеми на основі її параметрів, а генетичний алгоритм використовується для пошуку оптимальних параметрів схеми, максимізуючи функцію пристосованості, що обчислюється на основі порівняння прогнозованих і реальних значень характеристик.

РЕАЛІЗАЦІЯ ПРИКЛАДІВ АЛГОРИТМІВ ПРОЄКТУВАННЯ ІНТЕГРАЛЬНИХ СХЕМ

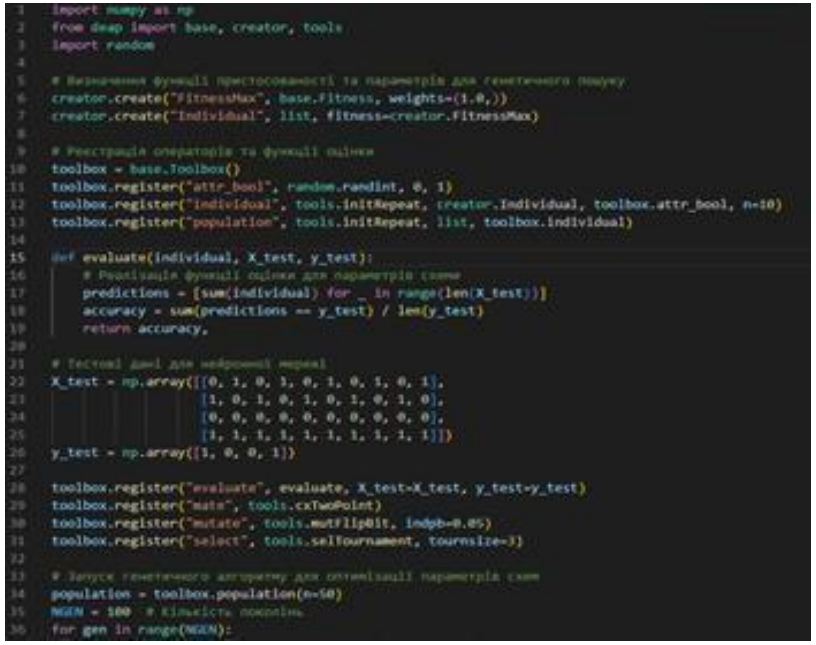
Структура першого коду нейронної мережі:



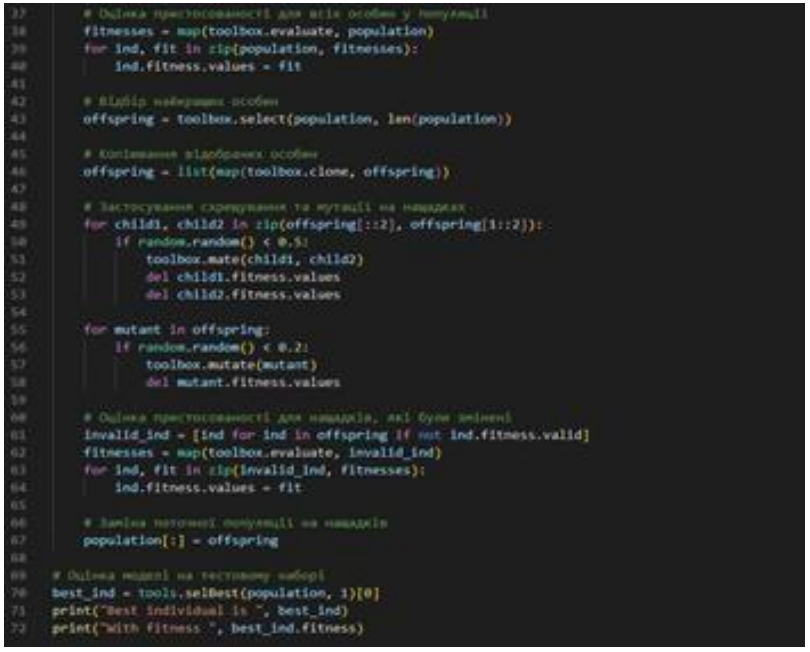
```python
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers  # Імпорт необхідного модуля

from sklearn.preprocessing import PolynomialFeatures

# Генеруємо дані для властивостей інтегральних схем
num_samples = 1000
num_features = 30

x_train = np.random.rand(num_samples, num_features)
for i in range(10):
    x_train[:, i] = np.sin(x_train[:, i])
    x_train[:, i+10] = np.cos(x_train[:, i])
x_train += np.random.normal(0, 0.1, size=(num_samples, num_features))
y_train = np.random.rand(num_samples, 1)

# Додаємо поліноміальні ознаки
poly = PolynomialFeatures(degree=2, include_bias=False)
x_train_poly = poly.fit_transform(x_train.reshape(-1, num_features))

# Перетворюємо дані в формат зображень (batch_size, height, width, channels)
x_train_poly = x_train_poly.reshape((-1, poly.n_output_features_, 1, 1))

# Створюємо та навчаємо модель
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 1), activation='relu', input_shape=(poly.n_output_features_, 1, 1)),
    layers.MaxPooling2D((2, 1)),
    layers.Conv2D(64, (3, 1), activation='relu'),
    layers.MaxPooling2D((2, 1)),
    layers.Conv2D(128, (3, 1), activation='relu'),
    layers.MaxPooling2D((2, 1)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
model.fit(x_train_poly, y_train, epochs=20, batch_size=64)
```

РЕАЛІЗАЦІЯ ПРИКЛАДІВ АЛГОРИТМІВ ПРОЄКТУВАННЯ ІНТЕГРАЛЬНИХ СХЕМ

Структура другого коду генетичного алгоритму рисунок 1 та 2:



```python
import numpy as np
from deap import base, creator, tools
import random

# Визначення функції пристосованості та параметрів для генетичного пошуку
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

# Реєстрація операторів та функції оцінки
toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=10)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evaluate(individual, X_test, y_test):
    # Реалізація функції оцінки для параметрів схеми
    predictions = [sum(individual) for _ in range(len(X_test))]
    accuracy = sum(predictions == y_test) / len(y_test)
    return accuracy,

# Тестові дані для нейронної мережі
X_test = np.array([[0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
                   [1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
                   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
y_test = np.array([1, 0, 0, 1])

toolbox.register("evaluate", evaluate, X_test=X_test, y_test=y_test)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)

# Запуск генетичного алгоритму для оптимізації параметрів схеми
population = toolbox.population(n=50)
NGEN = 100  # Кількість поколінь
for gen in range(NGEN):
```



```python
    # Оцінка пристосованості для всіх особин у популяції
    fitnesses = map(toolbox.evaluate, population)
    for ind, fit in zip(population, fitnesses):
        ind.fitness.values = fit

    # Вибір найкращих особин
    offspring = toolbox.select(population, len(population))

    # Копіювання відібраних особин
    offspring = list(map(toolbox.clone, offspring))

    # Застосування схрещування та мутації на нащадках
    for child1, child2 in zip(offspring[::2], offspring[1::2]):
        if random.random() < 0.5:
            toolbox.mate(child1, child2)
            del child1.fitness.values
            del child2.fitness.values

    for mutant in offspring:
        if random.random() < 0.2:
            toolbox.mutate(mutant)
            del mutant.fitness.values

    # Оцінка пристосованості для нащадків, які були змінені
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    fitnesses = map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    # Заміна поточної популяції на нащадків
    population[:] = offspring

# Оцінка моделі на тестовому наборі
best_ind = tools.selBest(population, 1)[0]
print("Best individual is ", best_ind)
print("With fitness ", best_ind.fitness)
```

ВИСНОВКИ

У кваліфікаційній роботі розглянуто застосування алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем. Досліджено традиційні та нові методи проєктування ІС, включаючи аналіз різних алгоритмів ШІ. Вивчено можливості їх впровадження для оптимізації різних етапів проектування. Розроблено два алгоритми ШІ для автоматизації задач проектування ІС. Заключено визначено значення алгоритмів ШІ для покращення ефективності та скорочення часу проектування ІС.

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:
Автор: Ілля ФЕДЧУК

Назва: Використання алгоритмів штучного інтелекту для автоматизованого проектування інтегральних схем
Координатор: Світлана КУФТЕРІНА
Підрозділ: ННІІТ

Коефіцієнт подібності 1:0.2

Коефіцієнт подібності 2:0

Тривога: 1

Після аналізу Звіту подібності констатую наступне:

☐виявлені в роботі запозичення є сумлінними і не мають ознак плагіату. Тому робота визнається самостійною і допускається до захисту;

☐виявлені в роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на доопрацювання;

☐виявлені в роботі запозичення є недобросовісними і мають ознаки плагіату або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. У зв'язку з чим, робота не допускається до захисту.

…… ……

Дата *Підпис Наукового керівника*