Міністерство освіти і науки України
Харківський національний університет радіоелектриніки
Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Автоматизації проєктування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Синтез та верифікація цифрової схеми по логічному вектору
(тема)

Виконав: здобувач IV курсу, групи КІУКІ-21-8
Ряполов А.В.
(прізвище, ініціали)
Спеціальність *123 _ Комп'ютерна інженерія*
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник проф. Чумаченко С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри АПОТ _____ Чумаченко С.В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123_Комп'ютерна інженерія
(шифр і назва)

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

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри АПОТ
Чумаченко С.В.
(підпис)
«_06_»___05_____ 2025 р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ряполову Андрію Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Синтез та верифікація цифрової схеми за логічним вектором
затверджена наказом університету від 21 05 2025 р. № 403Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2025 р.
3. Вихідні дані до роботи
Математичний апарат дискретної математики, комп'ютерної логіки, векторно-логічні процедури, структури даних
4. Перелік питань, що потрібно опрацювати в роботі
Постановка задачі. Аналіз стану технологій. Огляд літератури.
Побудова логічного вектора для цифрової схеми
Декомпозиція логічного вектора.
Синтез схеми Шнейдера
Верифікація структур

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

19 слайдів

6. Консультанти розділів роботи (проекту)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми.	06.05.2025 – 06.05.2025	
2	Аналіз предметної області, постановка задачі, стан технологій, огляд литератури.	07.05.2025 – 15.05.2025	
3	Аналіз моделей, методів, технологій – математичного апарату для вирішення завдань.	16.05.2025 – 24.05.2025	
4	Розробка процедур синтезу та реалізація їх на окремих схемах	25.05.2025 – 05.06.2025	
5	Верифікація та тестування	06.06.2025 – 10.06.2025	
6	Оформлення пояснювальної записки.	11.06.2025 – 15.06.2025	
7	Перевірка проєкту науковим керівником, допуск до захисту роботи.	16.06.2025 – 20.06.2025	

Дата видачі завдання 06.05.2025 р.

Здобувач _____ Ряполов А.В.
(підпис)

Керівник роботи _____ _____ проф. Чумаченко С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка з кваліфікаційної містить: 45 с., 14 рисунків, 1 додаток, 18 джерел за переліком посилань.

СХЕМА, СИНТЕЗ, ЛОГІЧНИЙ ВЕКТОР, ВЕКТОРНО-ЛОГІЧНИЙ КОМП’ЮТИНГ, МЕТРИКА, КАРТА ТЕСТУВАННЯ, ВЕРИФІКАЦІЯ

Тематика роботи стосується питань синтезу та верифікації цифрової схеми по логічному вектору на основі математичного апарату та технологій векторно-логічного комп’ютингу.

Мета бакалаврської роботи – зниження витрат на тестування та верифікацію цифрових проектів за рахунок синтезу логічного вектору цифрової схеми на основі векторно-логічних обчислювальних механізмів.

Задачі: огляд стану технологій; аналіз наукових джерел; аналіз математичного апарату та технологічних рішень векторно-логічного комп’ютингу; синтез та верифікація цифрової схеми по логічному вектору на основі векторно-логічних обчислювальних механізмів.

REFERENCE

The explanatory note of the qualification work contains: 45 pages, 11 figures, 1 application, 18 references.

SCHEME, SYNTHESIS, LOGICAL VECTOR, VECTOR-LOGICAL COMPUTING, METRICS, TESTING MAP, VERIFICATION

The topic of the work concerns the issues of synthesis and verification of a digital circuit by a logical vector based on the mathematical apparatus and technologies of vector-logical computing.

The purpose of the bachelor's work is to reduce the costs of testing and verification of digital projects by synthesizing the logical vector of a digital circuit based on vector-logical computing mechanisms.

Tasks: review of the state of technologies; analysis of scientific sources; analysis of the mathematical apparatus and technological solutions of vector-logical computing; synthesis and verification of a digital circuit by a logical vector based on vector-logical computing mechanisms.

.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence (штучний інтелект);

ASIC – Application-Specific Integrated Circuit, (інтегральна схема спеціального призначення);

CNN – Convolutional Neural Nerwork (згорткова нейронна мережа);

CPU – Central Processing Unit (центральний процесор);

FAAS – Faults-as-address circuit simulation (моделювання схем з неисправностями як адресами);

FPGA – Field-Programmable Gate Array (програмована вентильна матриця);

GenAI – Generative AI (генеративний ШІ);

GPU – Graphics Processing Unit (графічний процесор);

IMC – in-memory computing (обчислення в пам'яті);

IoT – Internet of Things (інтернет речей);

PQC – post-quantum cryptography (постквантова криптографія);

SRAM – Static Random Access Memory (статична пам'ять з довільним доступом);

ДНФ – диз’юнктивна нормальна форма;

КНФ – кон’юнктивна нормальна форма;

ШІ – штучний інтелект.

ВСТУП

Тематика роботи стосується питань синтезу та верифікації цифрової схеми по логічному вектору на основі математичного апарату та технологій векторно-логічного комп’ютингу.

Мета бакалаврської роботи – зниження витрат на тестування та верифікацію цифрових проектів за рахунок синтезу логічного вектору цифрової схеми на основі векторно-логічних обчислювальних механізмів.

Задачі: огляд стану технологій; аналіз наукових джерел; аналіз математичного апарату та технологічних рішень векторно-логічного комп’ютингу; синтез та верифікація цифрової схеми по логічному вектору на основі векторно-логічних обчислювальних механізмів.

1 СТАН ТЕХНОЛОГІЙ

Розглядаються сучасні тенденції та технології, що визначають актуальність роботи. Наводиться огляд літератури.

1.1 Топ-10 стратегічних технологічних трендів 2025 року від Gartner

Стратегічні технологічні тренди відіграють ключову роль у формуванні майбутнього, сприяючи впровадженню інновацій [1, 2]. Проте одночасно вони постають джерелом дискусій щодо питань етики, відповідальності та довіри. Лідерські компетенції керівників у галузі інформаційних технологій піддаються перевірці їхньою здатністю передбачати майбутні виклики та відповідним чином до них готуватися. У цьому контексті огляд стратегічних технологічних трендів, запропонований компанією Gartner, може слугувати своєрідним орієнтиром для визначення напрямків розвитку.

Технологічні тренди можна розділити на три категорії.

1. Вимоги та ризики штучного інтелекту. Прогрес ШІ-агентів вимагатиме нових підходів до регулювання ШІ та нових технологій для боротьби з дезінформацією: агентський ШІ, платформи регулювання ШІ, захист від дезінформації.

2. Нові обчислювальні можливості. Квантові обчислення вимагатимуть нових методів шифрування, а недорогі сенсори стимулюватимуть розвиток інноваційних бізнес-моделей: пост-квантова криптографія, фоновий ШІ, енергоефективні обчислення; гібридні обчислення.

3. Синергія людини та машини. Треба бути готовими до інтенсивної взаємодії фізичного та віртуального світів, до робіт у повсякденному житті та до технологій, що безпосередньо впливають на процес пізнання та продуктивність праці: просторові обчислення; багатофункціональні роботи; нейроудосконалення.

Розглянемо кожен з технологічних трендів та сценарії їх використання.

1. *Агентський ШІ*. Агентський ШІ (Agentic AI) – програмне забезпечення, здатне самостійно приймати рішення та виконувати дії, спрямовані на досягнення певних цілей. Для цього він використовує такі функції ШІ, як запам'ятовування, планування, сприйняття навколишнього середовища, використання інструментів та дотримання правил безпеки.

Прогноз Gartner: частка повсякденних рішень, що приймаються автономним агентським ШІ, зросте з нуля у 2024 році до 15% у 2028 році.

Чому в топі: здатність агентського ШІ діяти автономно або напівавтономно може дозволити ІТ-директорам реалізувати ідеї підвищення продуктивності за рахунок генеративного ШІ.

Сценарії використання: допомога співробітникам у плануванні та реалізації технічно складних ініціатив, від мікроавтоматизації до масштабних проектів, за допомогою звичайної людської мови; автоматизація клієнтського досвіду шляхом прорахування рішень на кожному кроці на основі аналізу даних; поліпшення механізмів реагування на ситуацію, що складається, і прийняття рішень за рахунок більш швидкого аналізу даних та передиктивної аналітики.

2. *Платформи регулювання ШІ*. Платформи регулювання ШІ (AI Governance Platforms) забезпечують відповідальне та етичне використання систем на базі ШІ. Вони дають ІТ-керівникам впевненість у тому, що ШІ надійний, прозорий, чесний, підконтрольний, відповідає стандартам етики та безпеки. Така платформа – гарантія того, що ШІ не суперечить цінностям організації та очікуванням суспільства загалом.

Прогноз Gartner: до 2028 року корпорації, що застосовують платформи регулювання ШІ, отримають рейтинги довіри споживачів на 30% вищі, ніж у конкурентів, а рейтинги відповідності регуляторним вимогам – на 25% вищі.

Чому в топі: ШІ застосовується дедалі ширше, зокрема у галузях із жорстким регулюванням. У міру поширення ШІ зростають ризики, пов'язані з дискримінацією та персональними даними, дедалі актуальнішою стає необхідність відповідати загальнолюдським цінностям. Критично важливо,

щоб ШІ не шкодив певним соціальним групам, не маніпулював ринком і не мав контролю над ключовими системами.

Сценарії використання: оцінка можливих ризиків та збитків від використання ШІ, включаючи дискримінацію, негативні соціальні наслідки та витік персональних даних; процеси регулювання, що супроводжують ШІ-моделі протягом усього їх життєвого циклу, забезпечують виконання всіх належних контрольних процедур. Відстеження використання ШІ, моніторинг продуктивності ШІ-систем, аудит процесу прийняття рішень та забезпечення постійного дотримання стандартів та нормативів.

3. *Захист від дезінформації.* Технології захисту від дезінформації повинні показувати, чому можна вірити. Вони націлені на створення систем, що забезпечують точність та автентичність інформації, що запобігають підробці та відстежують поширення шкідливого контенту.

Прогноз Gartner: до 2028 року продукти, послуги або можливості запобігання дезінформації використовуватимуть 50% корпорацій порівняно з 5% у 2024 році.

Чому в топі: захист від дезінформації стає полем для гонки цифрових озброєнь: зловмисники вдаються до фішингу, кібератак, фейкових новин та соціальної інженерії, щоб сіяти страх і хаос і шахрайства. У міру розвитку та більшої доступності ШІ та машинного навчання компанії стикатимуться зі зростаючим потоком дезінформації. Якщо залишити це поза увагою, ризики будуть дуже високі.

Сценарії використання: виявлення контенту, створеного з використанням ШІ, у юридично значимих ситуаціях (наприклад, комунікації у режимі реального часу або перевірка скарг); моніторинг службою безпеки інформації, що розповсюджується через ЗМІ або соціальні мережі, яка може нашкодити бренду, топ-менеджменту, сприйняттю запропонованих товарів та послуг; запобігання підміні особи контрагентів – співробітників, підрядників, постачальників та клієнтів.

4. *Пост-квантова криптографія*. Постквантова криптографія (post-quantum cryptography, PQC) – це методи шифрування, спроектовані так, щоб бути захищеними від потенційних загроз з боку квантових комп'ютерів.

Прогноз Gartner: до 2029 року прогрес у галузі квантових обчислень зробить більшість загальноприйнятих методів асиметричного шифрування небезпечними.

Чому в топі: квантові обчислення скоро – мабуть, протягом наступного десятиліття – стануть реальністю. Очікується, що більшість існуючих методів шифрування будуть визнані застарілими, що створить серйозну загрозу безпеці даних. Зловмисники з нетерпінням чекають на прорив у квантових технологіях, а тим часом діють за принципом «добудь дані зараз – розшифруй потім». Ця загроза робить пост-квантові технології все більш актуальними і затребуваними, оскільки вони дозволять захиститися від дешифрування квантовими комп'ютерами.

Сценарії використання: захист систем від майбутніх загроз витоку конфіденційної фінансової інформації навіть у світі квантових комп'ютерів; захист інтелектуальної власності від кіберзагроз, включаючи майбутні атаки за допомогою квантових комп'ютерів, що не дозволяє хакерам або конкурентам розшифрувати конфіденційну інформацію; захист зашифрованих повідомлень, контрактів або операційних даних від перехоплення та подальшого розшифрування за допомогою квантових технологій.

5. *Фоновий ШІ*. Під фоновим ШІ (Ambient Invisible Intelligence) мається на увазі поширення невеликих і недорогих міток і сенсорів, що дозволяють відстежувати розташування та статус різних об'єктів або навколишнього середовища. Ця інформація потім відправляється в хмару для аналізу та зберігання. Така технологія вбудовуватиметься у предмети повсякденного побуту, часто без відома користувача.

Прогноз Gartner: у період до 2028 року ранні зразки фонового ШІ будуть застосовуватися для недорогого відстеження та виявлення предметів, що

дозволить вирішувати нагальні проблеми бізнесу у сфері зниження витрат та підвищення продуктивності.

Чому в топі: технологія недорогих міток та датчиків стала більш доступною та економічно виправданою. Вона забезпечує можливість контролю у реальному часі, що особливо цінно в ланцюгах поставок, а згодом може знайти застосування у більш масштабних екосистемах. Крім того, дані з сенсорів стануть основним джерелом аналітики для подальшого поліпшення продуктів і процесів. Швидкому розвитку та впровадженню фонового ШІ сприятиме прогрес у бездротових технологіях та Bluetooth, а також у таких нових технологіях, як зворотне розсіювання (backscatter) та друкована електроніка (printed electronics).

Сценарії використання: у роздробі – автоматичне налаштування світла та звуку, рекомендації на підставі переваг покупця; в офісі – моніторинг використання офісного простору співробітниками та автоматичне налаштування параметрів клімату та освітлення; у охороні здоров'я – безперервний моніторинг стану пацієнтів без використання гаджетів, що носять, що дозволить оперативно реагувати на екстрені ситуації.

6. *Енергоефективні обчислення*. Під енергоефективними обчисленнями (Energy-Efficient Computing) розуміється розробка та експлуатація комп'ютерів, центрів обробки даних та інших цифрових систем з метою мінімізації енергоспоживання та вуглецевого сліду.

Прогноз Gartner: вуглецевий слід – пріоритет для більшості IT-організацій.

Чому в топі: стійкий розвиток знаходиться в центрі уваги порад директорів компаній. Фінтех, ШІ та різноманітні IT-сервіси споживають велику кількість енергії і тим самим впливають на екологію. При поточному рівні розвитку технологій енергоефективність в IT досягла меж, але в найближчі десять років можливе її зростання за рахунок нових технологій, таких як графічні процесори, нейроморфні обчислення та квантові комп'ютери.

Сценарії використання: зниження витрат на центри обробки даних за рахунок скорочення енергоспоживання серверів та систем охолодження; стійка технологія – розробка енергоефективних продуктів з допомогою енергоефективних засобів розробки; зниження енергоспоживання в офісі за рахунок використання розумних систем керування електроенергією.

7. *Гібридні обчислення*. Гібридні обчислення (Hybrid Computing) – це комбінування різних обчислювальних технологій, таких як центральні процесори (CPU), графічні процесори (GPU), спеціалізовані мікросхеми (ASIC), нейроморфні та квантові комп'ютери, фотоніка, для вирішення складних обчислювальних завдань. Гібридне середовище дозволяє скористатися перевагами кожної технології.

Чому в топі: гібридні обчислення дозволять бізнесу зробити потужний ривок за рахунок нових обчислювальних технологій – фотонних, нейроморфних, квантових та біокомп'ютерів. Генеративний ШІ – яскравий приклад того, що вирішення складних проблем потребує масштабних обчислювальних ресурсів та інфраструктури для зберігання та передачі даних.

Сценарії використання: економічне масштабування за рахунок використання для пікових навантажень хмарної інфраструктури на додаток до більш захищених локальних серверів; підвищення захищеності даних та дотримання регуляторних вимог за рахунок зберігання конфіденційної інформації на локальних серверах та використання зовнішньої інфраструктури для аналітики та несекретних даних; прискорення розробки та інновацій за рахунок поєднання хмарного середовища для розробки та захищених внутрішніх ресурсів для експлуатації IT-систем.

8. *Просторові обчислення*. Просторові обчислення (Spacial Computing) доповнюють фізичний світ цифровим контентом, дозволяючи користувачам взаємодіяти з ним – реалістично, інтуїтивно-зрозуміло, з ефектом занурення.

Прогноз Gartner: до 2028 року 20% людей хоча б раз на тиждень звертатимуться до контенту з перманентною геопозиційною прив'язкою порівняно з менш як 1% таких людей у 2023 році.

Чому в топі: розвиток технологій доповненої реальності (Augmented Reality, AR), змішаної реальності (Mixed Reality, MR) та ШІ створюють цифрове середовище з ефектом занурення у комп'ютерних іграх, електронній торгівлі та охороні здоров'я. Поширення мереж 5G та таких нових пристроїв, як Apple Vision Pro та Meta Quest 3, стимулюють споживчий попит та відкривають дорогу новим бізнес-моделям. Завдяки ключовим гравцям Nvidia і Qualcomm, що вкладаються у створення екосистем, очікується зростання ринку зі $110 млрд у 2023 році до $1,7 трильйона у 2033 році.

Сценарії використання: взаємодія співробітників у цифровому тривимірному середовищі, що робить віддалені наради більш інтерактивними та продуктивними; створення максимально наближених до реальності симуляторів для навчання співробітників, що знижує ризики та витрати на навчання, допомагає придбанню та підтримці навичок; віртуальний похід по магазину з віртуальним помічником, що сприяє підвищенню задоволеності та залучення покупця та зростання продажів.

9. *Багатофункціональні роботи.* Багатофункціональні роботи (Polyfunctional Robots) – це гнучкі машини, здатні виконувати різноманітні операції, наслідуючи приклад або вказівку людини.

Прогноз Gartner: до 2030 року щодня взаємодіяти з розумними роботами будуть 80% людей, порівняно з менш ніж 10% сьогодні.

Чому в топі: за допомогою роботів можна впоратися з дефіцитом кадрів, зростанням витрат на персонал і падінням рентабельності в логістиці і на виробництві. Виробники роботів звертають увагу на дедалі доступніші ціни на них. Піонери впровадження робототехніки можуть обирати різний функціонал у різних цінових категоріях та оцінювати реальні переваги роботів, їх можливості та вплив на витрати бізнесу.

Сценарії використання: у логістиці для пакування, обробки та транспортування товарів; у охороні здоров'я для доставки медичних товарів, допомоги маломобільним пацієнтам або дезінфекції; на виробництві,

включаючи небезпечне та важкодоступне, для обслуговування та ремонту обладнання.

10. *Нейроудосконалення*. Нейроудосконалення (Neurlogical Enhancements) – це технології поліпшення когнітивних здібностей людини за рахунок зчитування та розшифрування сигналів мозкової активності або навіть надсилання сигналів у мозок.

Прогноз Gartner: нейроудосконалення, такі як двосторонній інтерфейс мозок-машина, що тільки з'явилися в 2024 році, до 2030 року охоплять 60% IT-фахівців.

Чому в топі: нейроудосконалення потенційно здатне зробити діяльність мозку прозорою, що зробить революцію у охороні здоров'я. У міру розвитку ШІ компанії вивчають можливості інтерфейсів мозок-машина, щоб допомогти співробітникам підвищувати кваліфікацію, покращувати когнітивні здібності та бути затребуваними професіоналами. У рамках маркетингу нового покоління вивчається можливість надання більш персоналізованого споживчого досвіду.

Сценарії використання: скорочення терміну інтернатури хірургів цілий рік; персоналізація навчальних матеріалів під студентів реального часу; зниження травматизму робочому місці, підвищення промислової безпеки; більш ефективне залучення та утримання персоналу завдяки виявленню нейросумісності співробітників.

1.2 Аналіз літератури

Впровадження в EDA-ринок інженерного підходу до вирішення завдань design and test на основі використання простих моделей векторної логіки, орієнтованої на економічний in-memory комп'ютинг [3, 4] на основі read-write транзакцій, вільний від інструкцій процесора. Векторна логіка представлена в основному апаратом матрично-векторного формалізму для логічних функцій та значень істинності-неправди [5-7].

В останні роки обчислення в пам'яті (IMC) є перспективним методом вирішення проблеми вузького місця переміщення даних у пристроях штучного інтелекту на периферії. Для виконання деяких простих обчислень у пам'яті, конструкції IMC часто використовують аналогові операції, що може призвести до неминучих помилок обчислень. Щоб відновити втрату точності, додавання деяких випадкових збурень до навчання CNN є простим підходом, щоб зробити його більш стійким до помилок обчислень. Однак випадкові значення можуть суттєво відрізнятися від реальних помилок під час виконання, спричинених неідеальними ефектами.

У роботі [3] автори пропонують ієрархічну структуру моделювання для систем IMC для підтримки навчання CNN з урахуванням помилок. Ця структура включає ефективний підхід до побудови точних моделей поведінки IMC, які відображають реальні неідеальні ефекти. Використовуючи поведінкову модель, можна ефективно побудувати точну модель помилок високого рівня, щоб забезпечити помилки під час виконання для навчання CNN та перевірки коефіцієнта помилок. Як показано в результатах моделювання, навчання CNN з урахуванням помилок за допомогою запропонованих моделей ефективно покращує точність CNN у реальних застосуваннях майже без втрати точності.

У роботі [4] представлено інструмент для створення прототипів, адаптований для ранньої фази проектування архітектур обчислень у пам'яті на основі SRAM. Використовуючи SystemVerilog та SystemC, запропонований інструмент забезпечує гнучке, спритне та незалежне від фреймворку моделювання. Крім того, представлено фреймворк для системного моделювання для оцінки правильності моделювання та потенційних переваг існуючих конструкцій IMC.

Робота [5] описує діаграматичну система пропозиційної логіки, в якій пропозиції розглядаються як вектори або зміщення в «логічному просторі». В [6] векторна логіка розглядається як матрично-векторне подання логічного числення, натхненного моделями нейронних мереж. У цьому алгебраїчному

формалізмі значення істинності відображаються на ортонормованих Q-вимірних векторах, монадні операції представлені квадратними матрицями, а діадні операції створюють прямокутні матриці, які діють на добуток кронекера векторних значень істинності. У цьому формалізмі теореми та тавтології класичної логіки демонструються за допомогою правил матричної алгебри. Автори аналізують тризначну векторну логіку, яка додає до векторів «так» і «ні» третій «невизначений» вектор, який представляє значення істинності, що відповідає нерозв'язним твердженням. Нечіткість створюється як лінійними комбінаціями векторів «так» і «ні», так і додатковою розмірністю логічного векторного підпростору. Описано основні матричні оператори та показано, що для цієї тризначної векторної логіки модальності «можливість» і «необхідність» є простими квадратними матрицями, а не нескінченними рекурсивними процесами.

В [7] автори досліджують представлення контрфактичних умовних операторів за допомогою векторної логіки, матрично-векторного формалізму для логічних функцій та значень істинності. В межах цього формалізму контрфактичні оператори можуть бути перетворені в комплексні матриці, попередньо обробляючи матрицю імплікації з одним із квадратних коренів NOT, комплексною матрицею. Цей математичний підхід підтверджує віртуальний характер контрфактичних операторів. Це відбувається тому, що таке представлення дає оцінку контрфактичного оператора, яка є суперпозицією двох протилежних значень істинності, зважених відповідно двома комплексно спряженими коефіцієнтами. Цей результат показує, що ця процедура дає невизначену оцінку, спроектовану на комплексну область. Після цього базового представлення, судження про правдоподібність заданого контрфактичного оператора дозволяє змістити рішення в бік прийняття або відмови. Цей зсув є результатом застосування вдруге одного з двох квадратних коренів NOT.

В статті [8] запропоновано векторний синтез карти тестування (моделювання) несправностей для логіки, який без моделювання дозволяє

визначити всі несправності, виявлені на тестових наборах, а також визначити тестові набори для виявлення заданих несправностей. Для синтезу використовується суперпозиція інтелектуальних структур даних, що містить: дедуктивну матрицю D, як похідну логічного вектора L, таблицю істинності тестів T та таблицю істинності несправностей F. Дедуктивна матриця розглядається як ген функціональності та основа механізму моделювання несправностей для вирішення всіх задач тестування та верифікації. У матричному синтезі використовується аксіома: всі згадані таблиці ідентичні за формою одна одній та завжди взаємодіють одна з одною згортково $T \oplus L \oplus F = 0$. Запропоновано універсальний дедуктивний реверсивний перетворювач «тест-несправності» та «несправності-тест» для логічних функціональностей будь-якої розмірності. Функції перетворювача: моделювання несправностей на тестових наборах $T \rightarrow F$ та синтез тестових наборів $F \rightarrow T$ для виявлення заданих несправностей. Перетворювач може бути використаний як сервіс генерації тестів та моделювання несправностей для систем на кристалі (SoC) з IP-ядро відповідно до стандарту IEEE 1500 SECT. На основі дедуктивної матриці будується карта тестування несправностей для логіки, де кожен тестовий набір зіставляється з логічними несправностями вхідних ліній.

В [9] запропоновано векторно-логічний механізм моделювання несправностей за адресами на інтелектуальних структурах даних, який виключає алгоритм моделювання тестових наборів для отримання карти тестування логічної функціональності. Інтелектуальні структури даних представлені логічним вектором та його похідними у вигляді таблиць істинності та матриць. Карта тестування являє собою матрицю, координати якої визначаються комбінаціями всіх логічних несправностей, виявлених на двійкових наборах вичерпного тесту. Побудова карти тестування орієнтована на архітектуру обчислень в оперативній пам'яті на основі транзакцій читання-запису, що робить механізм моделювання економічним щодо часу моделювання та енергоспоживання через відсутність центрального процесора.

Логічний вектор, як єдиний компонент вхідних даних, не потребує синтезу в технологічно дозволену структуру елементів. Синтез інтелектуальних структур даних на основі чотирьох матричних операцій створює карту тестування виявлених несправностей для будь-якої логіки. Запропонований механізм орієнтований на обслуговування IP-ядер SoC за стандартом IEEE 1500, що може бути позитивно сприйнято інженерами на ринку електронних інформаційних систем (EDA). З точки зору простоти та передбачуваності розміру структур даних і відсутності алгоритму моделювання тестових наборів, запропонований механізм не має аналогів у проектних та галузевих тестах.

В [10] розглядається нетрадиційне логічне тестування, засноване на використанні логічної константи, представленої у вигляді матриці відмов. Для вирішення задач тестування функціональності, заданої логічними векторами, пропонується константна матриця xor-відстаней між тестами та відмовами. У цьому випадку для тестів та відмов використовується одна модель у вигляді таблиці істинності n-змінних. Логічна константа є ядром механізму тестування логічних функцій та цифрових схем. Разом з логічною матрицею, яка є похідною логічного вектора елемента, логічна константа бере участь у проектуванні інтелектуальних структур даних, які без моделювання дозволяють побудувати карту тестів. Всі компоненти інтелектуальних структур даних є похідними від логічного вектора. Усі задачі тестування вирішуються у двовимірному просторі, який організований на основі операції xor над адресами таблиці істинності n-змінних. Векторно-логічні механізми тестування реалізовані на Python. Поточна навчальна реалізація системи дозволяє будувати тести та моделювати несправності для будь-якої функціональності та логічних схем, з розмірністю до 100 елементів. Логіка моделювання несправностей без моделювання є новизною та перевагою механізму швидкого тестування.

Робота [11] присвячена моделюванню несправностей як адрес (FAAS) – це механізм моделювання для тестування комбінацій несправностей ліній,

представлених бітовими адресами логічних векторів елементів. Зв'язок XOR між тестовим набором T та таблицею істинності L елемента утворює дедуктивний вектор для моделювання несправностей, використовуючи адреси таблиці істинності або біти логічного вектора. Адреси використовуються в матриці моделювання для позначення тих n-комбінацій вхідних несправностей, виявлених на виході елемента. Стовпці матриці моделювання обробляються як адреси n-рядків для генерації вихідного рядка елемента за допомогою дедуктивного вектора. Немає транспортування вхідних несправностей до виходу елемента, лише 1-сигнали записуються в координатах невхідного рядка матриці моделювання схеми. Матриця моделювання спочатку заповнюється 1-сигналами вздовж головної діагоналі. Несправності ліній, виявлені на тестовому наборі схем, визначаються інверсними значеннями хороших ліній, які мають 1-значення в рядку матриці, що відповідає вихідному елементу схеми. Дедуктивний вектор отримується за допомогою співвідношень XOR між тестовим набором та логічним вектором у трьох табличних операціях. Перевагою запропонованого механізму FAAS є передбачувана складність алгоритму та споживання пам'яті для зберігання структур даних під час моделювання тестового набору.

У стандарті [12] розглянуто способи, за допомогою яких розробники інтегральних схем можуть послідовно аналізувати синхронізацію та живлення мікросхем у широкому спектрі застосувань автоматизації проектування електротехніки (EDA). Розглянуто також методи, за допомогою яких постачальники інтегральних схем можуть виражати інформацію про синхронізацію та живлення один раз для кожної заданої технології. Крім того, обговорюються засоби, за допомогою яких постачальники EDA можуть задовольнити потреби своїх застосувань у продуктивності та потужності.

У роботі [13] йдеться про м'які помилки в інтегральних схемах (IC), що завжди були серйозною проблемою, особливо враховуючи, що вузли CMOS-технології продовжують зменшуватися, що призводить до вищої частоти, нижчої потужності та меншої площі, що посилює м'які помилки, викликані

радіацією. Тому перехідний процес внаслідок однієї події (SET) став вирішальним фактором при проектуванні сучасних радіаційно-стійких схем, оскільки він може спричинити збої у виходах схеми. У цій статті використовується концепція ймовірності сигналу для поширення перехідних несправностей у схемах. Розглядаючи проблему маскування перехідних несправностей, для кожного випадку маскування несправностей представлено модель поширення помилок. Крім того, запропоновано підходи як для ймовірнісних, так і для часових сценаріїв для врахування впливу шляхів повторної збіжності на поширення помилок. Оскільки розгляд шляхів повторної збіжності збільшує обчислювальну складність, у цій статті запропоновано три обчислювальні алгоритми, спрямовані на максимально можливе зменшення розміру матриці ймовірностей. Ми порівняли результати моделювання з методом Монте-Карло та моделюванням на основі HSPICE для перевірки запропонованого методу. Згідно з результатами моделювання на тестах ISCAS’85, запропонований підхід до оцінки частоти одиничних подій демонструє середній відносний відсоток похибки менше 5% порівняно з традиційною оцінкою впровадження несправностей.

1.3 Висновки та постановка завдання

Наведено тенденції розвитку сучасних технологій, що визначено консалтинговою компанією Gartner. Енергоефективні, гібридні та просторові обчислення вимагають нових підходів. Одним з них може виступати застосування нового напрямку – векторної логіки [8-11] в архітектурі in-memory комп’ютингу на основі read-write транзакцій, що дозволяє зменшити споживані ресурси у вигляді енергії, часу. Реалізація векторної логіки в обчисленнях у пам'яті робить її масовою та енергозберігаючою, вільною від складних алгоритмів аналізу великих даних.

Мета бакалаврської роботи – зниження витрат на тестування та верифікацію цифрових проектів за рахунок синтезу логічного вектору цифрової схеми на основі векторно-логічних обчислювальних механізмів.

Задачі: огляд сучасних технологій та публікацій, аналіз моделей, методів, технологій векторно-логічного комп’ютингу; синтез та верифікація цифрової схеми по логічному вектору на основі векторно-логічних обчислювальних механізмів.

2 МАТЕМАТИЧНИЙ АПАРАТ ТА ТЕХНОЛОГІЧНІ РІШЕННЯ ВЕКТОРНО-ЛОГІЧНОГО КОМП'ЮТИНГУ

Аналізується математичний апарат – моделі, методи, технології векторно-логічного комп'ютингу для подальшого синтезу та верифікації цифрової схеми по логічному вектору на основі векторно-логічних обчислювальних механізмів.

2.1 Векторна логіка

Векторно-логічний комп'ютинг (VLC) – це підхід до обчислень, який виключає використання традиційних процесорів, зосереджуючись на маніпуляціях з пам'яттю через read-write транзакції, виконані на логічних векторах. Основні характеристики та переваги векторно-логічного комп'ютингу включають:

1. Обчислення без процесорів забезпечують енергоефективність і значну економію ресурсів.

2. Відмова від класичної архітектури фон Неймана усуває необхідність передачі даних між пам'яттю і АЛП, що помітно скорочує час обробки даних.

3. Використання read-write транзакцій замість складних команд процесора дозволяє реалізовувати обчислювальні процеси практично на будь-якому пристрої пам'яті.

4. Логічні вектори легко розміщуються в пам'яті без необхідності трудомісткого синтезу в компоненти, від яких залежить технологічна реалізація.

5. Адреса пам'яті виступає не лише як простий ідентифікатор, але й як ефективний інструмент для обробки великих обсягів даних завдяки унітарному кодуванню патернів на множині базових елементів.

6. Лінійна складність алгоритмів, які розв’язують комбінаторні задачі через адреси таблиць істинності.

7. Використання інтелектуальних структур даних зі спрощенням програмування алгоритмів для їх аналізу.

8. Здатність вирішувати всі задачі тестування цифрової функціональності за допомогою логічних векторів та їх похідних.

9. Можливість моделювання як тестів, так і несправностей у вигляді адрес таблиць істинності.

10. Автоматичний синтез логічних функцій соціальних або фізичних явищ через аналіз двійково-кодованих великих даних. Цей підхід пропонує радикально нову парадигму обчислень, орієнтовану на швидкість, енергоефективність та масштабованість, використовуючи логіку і пам'ять як ключові компоненти.

Логічний вектор – це представлення процесу або явища, функції або структури послідовністю нулів і одиниць з довжиною $\mathbf{2^n}$, де n – кількість бітів двійкових змінних для формування адрес векторних бітів. Це визначення робить логічний вектор незалежним від таблиці істинності. Але обов'язковим неявним атрибутом вектора є стандартні адреси, які легко генеруються для будь-якого вектора. Декартовий добуток одного вектора сам з собою утворює рефлексивне відношення L= Y⊠Y, який породжує декартову матрицю, в даному випадку активність, або логічний вектор розмірності L $\mathbf{2^{n+1}}$ (рис. 2.1).

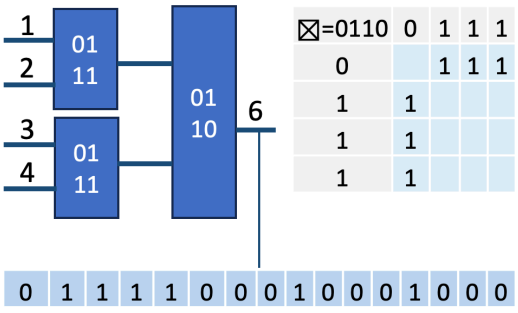


Рисунок 2.1 – Схема визначення рефлексивного xor-відношення вектора 0111

Векторна логіка – це представлення логічних операцій векторами станів таблиці істинності. Таблиця істинності має стандартизовані бітові адреси логічного вектора вихідних станів, що дає можливість використовувати логічний вектор як самостійну модель, поміщену в пам'ять. При необхідності завжди можна автоматично побудувати адреси кожного біта або згенерувати таблицю істинності для даної векторної моделі. Генерація логічного вектора передбачає розміщення його в пам'яті, де доступний кожен біт вектора, що забезпечується архітектурою адресних декодерів.

Логічний вектор, завдяки своїй компактній формі представлення таблиць істинності, є ефективним та універсальним засобом для тестування та верифікації будь-яких компонентів, від окремих функцій до складних шин даних. Він забезпечує повне покриття всіх елементів, що потребують перевірки. Крім того, логічний вектор може бути використаний як універсальна модель для аналізу та управління різними процесами, що цікавлять IT-сферу. Він відіграє важливу роль у розвитку інтелектуального комп'ютингу з in-memory архітектурою, а також є основою для створення дедуктивних матриць та карт тестування, що дозволяють комплексно перевіряти будь-яку функціональність.

2.2 Метрика логічного вектору

Векторний in-memory комп'ютинг – це спосіб обробки великих обсягів даних, який використовує транзакції запису-зчитування для обробки даних, представлених у вигляді логічного вектора. Доступ до даних здійснюється за допомогою адрес, що відповідають позиціям у цьому векторі.

Цей підхід, що відповідає сучасним вимогам до масових обчислень, характеризується двома ключовими перевагами: високою енергоефективністю та надзвичайно низькою затримкою (близько 0,6 наносекунд) [16, 17].

В основі цього методу лежить таблиця істинності. Запропоновано транспонувати цю таблицю (повернути на 90°, рис. 2.2) і використовувати

двійкові значення змінних як адреси для доступу до елементів логічного вектора, що дозволяє ефективно обробляти великі набори даних.

Truth table																
No		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	x_1									1	1	1	1	1	1	1	1
	x_2					1	1	1	1					1	1	1	1
	x_3			1	1			1	1			1	1			1	1
	x_4		1		1		1		1		1		1		1		1
Y			1	1		1				1	1			1		1	1

Рисунок 2.2 – Таблиця істинності й логічний вектор (Y) функціональності

Замість зберігання повних таблиць істинності з двійковими адресами, пропонується використовувати лише вектор вихідних значень. Цей вектор, по суті, є логікою обробки великих обсягів даних, де кожен біт представляє результат обробки певної комбінації вхідних даних. Адреса кожного біта у векторі неявно визначається комбінацією вхідних даних, які потрібно обробити.

Довжина цього логічного вектора 2^n визначає кількість можливих комбінацій вхідних даних n, які можна обробити. Більша довжина вектора означає можливість обробки більшої кількості вхідних даних, що дозволяє досягти більшого паралелізму в обробці великих даних.

На відміну від традиційних логічних елементів, функціональність цього логічного вектора не обмежена теоремою Поста [18]. Це дозволяє використовувати його для реалізації більш складних функцій.

Таким чином, логічний вектор може замінити арифметико-логічний пристрій (АЛП) та центральний процесор (ЦП) для більш ефективної обробки великих даних безпосередньо в місці їх зберігання, що значно зменшує енергоспоживання.

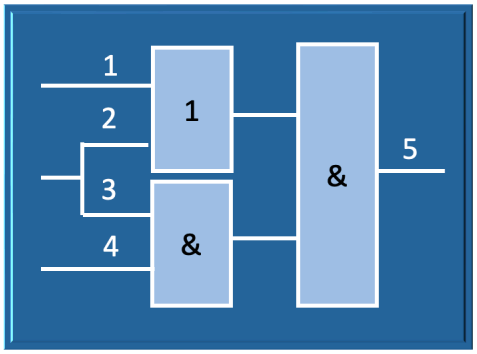
Для класифікації, кластеризації та ідентифікації патернів у великих даних, таблиця істинності (представлена логічним вектором) замінює методи штучного інтелекту, використовуючи обчислення подібності та відмінності.

Для ефективного використання логічного вектора та таблиці істинності необхідно застосовувати унітарне кодування елементів (патернів) потоку великих даних.

2.3 Структури даних

Структури даних для оперативного обчислення логічного вектора схеми: 1) два вхідних логічних вектора з довжиною, кратною двом; 2) числа для позначення вхідних змінних кожного вектора, які беруться з логічної схеми; 3) результуючий вектор схеми і вектор ідентифікаторів вхідних змінних схеми. Декартовий добуток двох векторів, що відповідає фрагменту діаграми двох елементів, виконується на основі двійкової логічної операції (AND, XOR, OR), яка дозволяє отримати нові, ще невідомі властивості в отриманому логічному векторі (матриці) для обробки даних.

Розгалуження в логічній схемі [12-15] генерують однойменні вхідні змінні (координати) на одному або декількох логічних елементах. Ця особливість схеми призводить до мінімізації результуючого вектора (координат матриці) фрагмента структури за рахунок однойменних логічних змінних, шляхом залишення координат, які відповідають однаковим значенням вхідних сигналів (адресних бітів) цих змінних, перекреслення координат з різними вихідними сигналами. В отриманому векторі L завжди є число координат, кратних степеню двох (рис. 2.3).



а

L	34	00	01	10	11
12	000 1	0	0	0	1
00	0	0	0	0	0
01	1	0	0	0	1
10	1	0	0	0	1
11	1	0	0	0	1

б

1	0	0	0	0	1	1	1	1
23	0	0	1	1	0	0	1	1
4	0	1	0	1	0	1	0	1
L	0	0	0	1	0	0	0	1

в

Рисунок 2.3 – Схема синтезу логічного вектора для фрагмента схеми

Побудова результуючого логічного вектора з двох взаємодіючих елементів на основі третього двійкового логічного вектора з двох змінних є універсальною процедурою, вільною від команд процесора, в якій використовуються тільки транзакції читання-запису, орієнтовані на обчислення в пам'яті. Тому при побудові тестової карти будь-якого логічного функціоналу можна використовувати логічні вектори і транзакції читання-запису.

Логічний вектор для будь-якої схеми може бути побудований шляхом рекурсивної суперпозиції завжди двох векторів: поточного дійсного вектора схеми і вектора наступного елемента. Побудову логічного вектора діаграми можна розглядати як паралельне пряме моделювання вичерпного тесту або вхідних множин, що є обов'язковою процедурою для перевірки проекту.

2.4 Елементи декартової логіки

Декартів добуток двох векторів різної розмірності створює новий вектор, розмірність якого визначається як 2^(n+m), де n та m – розмірності вихідних векторів. Цю операцію можна схематично представити трьома елементами (рис. 2.4). Збільшення розмірності логічного вектора відкриває нові можливості для аналізу та вирішення практичних задач.

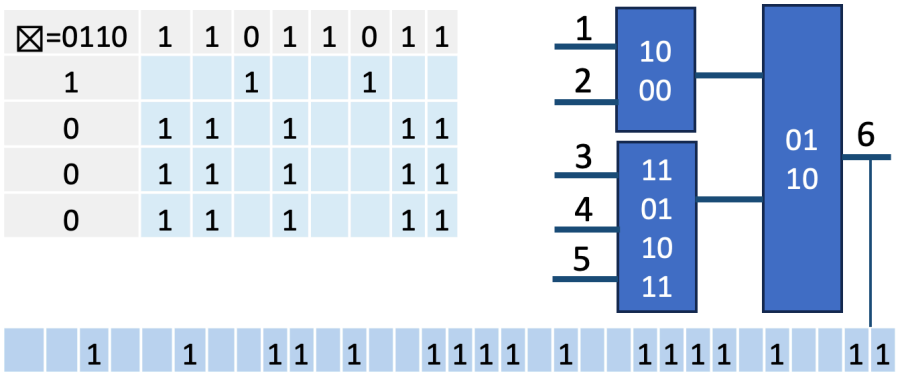


Рисунок 2.4 – Декартів добуток та його схемотехнічний аналог

Наприклад, вектор, отриманий в результаті декартового добутку, може вказати на конкретні позиції, в яких два вектори (що представляють різні функціональності) відрізняються один від одного, наприклад, при порівнянні 1000 та 11011011 в процесі повного тестування.

Ця схема ілюструє моделювання логічного вектора на основі векторів логічних елементів, що входять до схеми. Моделювання логічних векторів є важливим завданням для ринку EDA (Electronic Design Automation), оскільки

логічний вектор схеми є ефективним способом представлення інформації для вирішення задач проектування, тестування та інших.

Представлена схема є архітектурою для верифікації цифрових проектів, що базується на специфікації та поточній моделі цифрового пристрою. Результатом верифікації є логічний вектор, в якому одиничні значення вказують на розбіжності між результатами тестування в просторі верифікації проекту.

Декартова логіка – це система, що представляє собою матрицю відношень, сформовану за допомогою декартової операції (⊠-вектора) L=X⊠Y, яка застосовується до бінарних логічних векторів X та Y. Всі три вектори (X, Y та ⊠) є логічними, але вектор ⊠ (який також позначається як G) визначає відношення між X та Y. Важливо, що будь-який з цих векторів (X, Y або G) може виступати в ролі вектора, що визначає відношення між двома іншими.

Для коректного формування декартового відношення необхідно дотримуватися певних розмірностей векторів: X має розмірність 2^n, Y має розмірність 2^m, а G – 2^k, де n та m – довільні, а k дорівнює 2.

Декартова логіка є розширенням векторної логіки, що дозволяє ефективно вирішувати завдання, які є складними для звичайної векторної логіки. Вона використовує надмірність даних, що, як відомо, зменшує обчислювальну складність алгоритмів аналізу. Це пов'язано з тим, що модель даних та алгоритм обробки тісно пов'язані: збільшення складності моделі (наприклад, за рахунок надмірності) може спростити алгоритм, і навпаки.

Сутність полягає у використанні надлишкових моделей даних для вирішення складних комбінаторних задач за допомогою простих (лінійних або навіть 0-алгоритмів). Це досягається за рахунок збільшення надмірності моделі до експоненційної складності, що дозволяє спростити алгоритм обробки. Інтелектуальні надлишкові моделі стають ключем до вирішення широкого спектру обчислювальних задач ефективним чином.

2.5 Синтез логічного вектору схеми

Синтез логічного вектору схеми передбачає на початковому етапі розщеплення або розкладання багатовходових елементів на двовходові. Це дає можливість використовувати декартову логіку для отримання логічного вектору схеми або її фрагмента. Для звичайних функцій (AND, OR, XOR) та їх похідних завжди можна побудувати такий розподіл.

Для довільної векторної логіки такий розподіл також існує завжди (рис. 2.5), оскільки будь-який логічний вектор може бути представлений у вигляді ДНФ або КНФ, в яких регулярні логічні операції AND, OR, XOR.

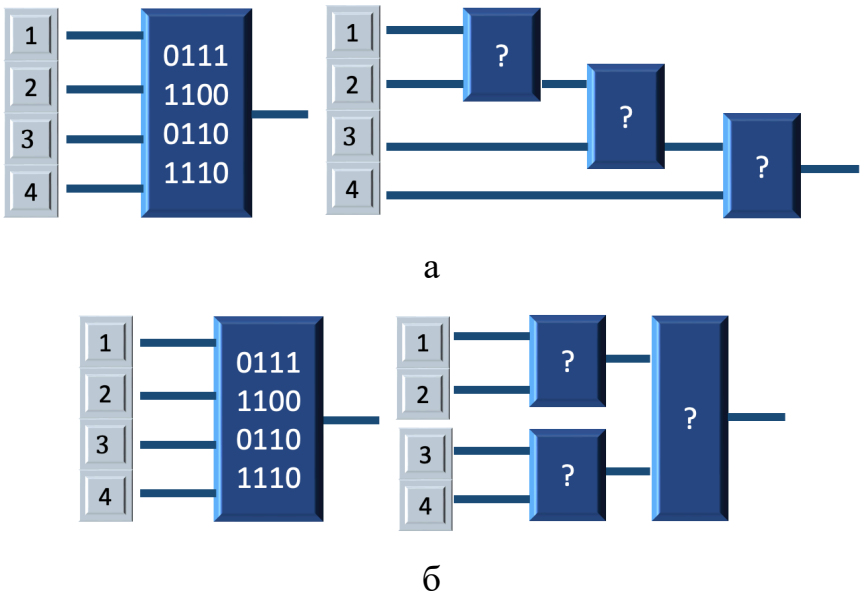


Рисунок 2.5 – Схеми розбиття складних елементів на двовходові

У загальному випадку для побудови логічного вектора з двох інших розмірностей будь-якого кратного потрібно використовувати наступний шаблон $\mathbf{2^n}$, степінь двійки і формуюча логічна функція двох змінних (рис. 2.6). Отже, завжди можна отримати логічний вектор з кількістю бітів $\mathbf{2^{n+m}}$, де n, m – довжина двох векторів, що утворюють матричне декартове відношення.

⊠	1	0	0	1	0	1	1	0
0								
1								
1								
0								

⊠=0111	1	0	0	1	0	1	1	0
0	1			1		1	1	
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
0	1			1		1	1	

⊠=0110	1	0	0	1	0	1	1	0
0	1			1		1	1	
1		1	1		1			1
1		1	1		1			1
0	1			1		1	1	

Рисунок 2.6 – Схема побудови логічного вектора-матриці фрагменту схеми на декартовій логіці

2.5 Висновки до розділу 2

Виконано аналіз математичного апарату та технологій векторно-логічного комп’ютингу для подальшого синтезу та верифікації цифрової схеми по логічному вектору.

3 СИНТЕЗ ТА ВЕРИФІКАЦІЯ ЦИФРОВОЇ СХЕМИ ПО ЛОГІЧНОМУ ВЕКТОРУ

Розглядаються прямі та зворотні процедури, що пов’язані із синтезом та верифікацією цифрової схеми по логічному вектору.

3.1 Побудова логічного вектора для цифрової схеми

Оскільки логічний вектор виступає як інструмент для вирішення завдань комп'ютингу, важливо будувати логічний вектор для цифрової схеми. Для цього застосовується рекурсивний алгоритм, що базується на декартовій логіці. Цей метод дозволяє отримати матрицю логічного вектору L схеми шляхом обчислення декартового добутку бітів двох векторів X і Y (X ⊠ Y) згідно з правилами логічної функції G (⊠). Вектор Y представляє собою сукупність результатів обробки попередніх елементів схеми, а вектор X – це вектор, що відповідає поточному елементу, який обробляється. В результаті отримується логічний вектор схеми, що відповідає вичерпному тестуванню, який можна розглядати як високопаралельне моделювання "good-value" для комбінаційної структури.

3.2 Декопмозиція логічного вектора

Декомпозиція логічного вектора, зображена на рис. 3.1, ґрунтується на рівнянні синтезу

$$L = X \boxtimes Y. \quad (3.1)$$

Суть задачі полягає в наступному: маючи відомий логічний вектор L та оператор ⊠, необхідно знайти вектори X та Y, які пов'язані між собою через

цей оператор. Важливим аспектом є визначення розмірності векторів X та Y, оскільки це впливає на структуру матриці, яку вони представляють. Для знаходження X та Y використовується правило

$$X \boxtimes L_{ij} = Y. \quad (3.2)$$

У цьому процесі також може бути застосована модель штучного інтелекту.

⊠=0001	1	0	0	1	0	1	1	0
0								
1	1			1		1	1	
1	1			1		1	1	
0								

L=X⊠Y

Рисунок 3.1 – Карта декомпозиції логічного вектора

Щоб розкласти вектор немонотонної функції, його представляють у вигляді матриці, отриманої множенням на відомий твірний G-вектор, як показано на рис. 3.2. Основна ціль – визначити два вектори, які, будучи об'єднаними (конкатенованими) побітово, утворюють цю матрицю.

Initial form for decomposition

⊠=0001								
00=0								
01=0		1			1		1	1
10=0		1			1		1	1
11=1								

L=X⊠Y→ 1=1⊠1

Resulting decomposition

⊠=0001	1	0	0	1	0	1	1	0
0								
1	1			1		1	1	
1	1			1		1	1	
0								

L=X⊠Y

Initial form for decomposition

⊠=0111								
00=0		1			1		1	1
01=1		1	1	1	1	1	1	1	1
10=1		1	1	1	1	1	1	1	1
11=1		1			1		1	1

L=X⊠Y→ 0=0⊠ 0

Resulting decomposition

⊠=0111	1	0	0	1	0	1	1	0
0	1			1		1	1	
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1			1		1	1	

L=X⊠Y

Рисунок 3.2 – Карти декомпозиції логічних векторів

Принцип дії полягає в наступному: конкатенація двох одиничних бітів (11) визначає адреси, що відповідають одиничним координатам у матриці логічного вектора. Аналогічно, конкатенація двох нульових бітів (00) визначає адреси нульових координат.

У загальному випадку, задача зводиться до пошуку двох логічних векторів, які, при побітовій декартовій конкатенації, генерують заданий інтегральний вектор, представлений у вигляді матриці.

Для вирішення цієї задачі необхідна дві умови:

1) відома твірна функція, алгоритмічна складність цього процесу оцінюється як $A=2^2 \times 2^n$.

2) якщо твірна функція невідома (рис. 3.3), алгоритмічна складність розв'язку задачі $A=2^2 \times 2^2 \times 2^n - 2 = 2^4 \times 2^n - 2$, що розв'язується повним перебором підстановки всіх 16–2 функцій від двох змінних, без двох функцій 0000 та 1111.

Initial form for decomposition									
⊠=$2^4 \times 2^n$-2									
00=0...1		1			1		1	1	
01=0...1		1	1	1	1	1	1	1	1
10=0...1		1	1	1	1	1	1	1	1
11=1...0		1			1		1	1	
L=X⊠Y									

Рисунок 3.3 – Карта декомпозиції логічного вектора на основі перебору

3.3 Декомпозиція логічного вектору шляхом синтезу диз'юнктивної нормальної форми

Існують двійкові вектори, які неможливо створити шляхом простого об'єднання (декартової суперпозиції) двох інших векторів. У таких випадках залишається лише один спосіб представити цей вектор як диз'юнктивну нормальну форму (ДНФ), побудовану на основі одиничних координат вектора.

Наприклад, вектор 10010100 можна розкласти в ДНФ як (рис. 3.4):

$$L=\bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_1x_2 \vee x_1\bar{x}_2x_2 \ . \qquad (3.3)$$

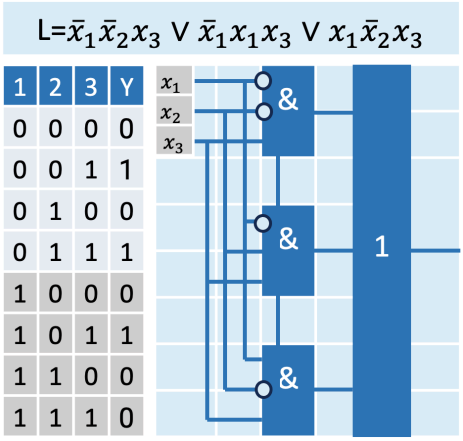


Рисунок 3.4 – Декомпозиція логічного вектора шляхом синтезу диз'юнктивної нормальної форми

Кожен елемент цієї схеми (ДНФ) є логічним вектором, що представляє монотонну логічну функцію (типу "І" або "АБО") з базису Поста. Це дозволяє легко побудувати логічний вектор для будь-якого фрагменту схеми, навіть якщо логічний вектор містить велику кількість $2^n - 1$ одиничних бітів.

Важливо, що монотонна функція характеризується тим, що її таблиця істинності має лише один перехід: або з 0 в 1, або з 1 в 0.

3.4 Розподіл логічного вектору на два

Для розділення логічного вектора на дві частини можна використовувати комутаційну схему. Мета полягає в створенні схеми, що складається з двовходових елементів, яка б обробляла монотонний логічний вектор. Цей алгоритм, хоч і передбачуваний, працює з векторами будь-якої структури (рис. 3.5). Однак, недоліком є можлива надмірність отриманої

схеми, яка може містити елементи з більшою кількістю входів, ніж потрібно для простого розділення вектора, наприклад, 11010101, на дві рівні частини.

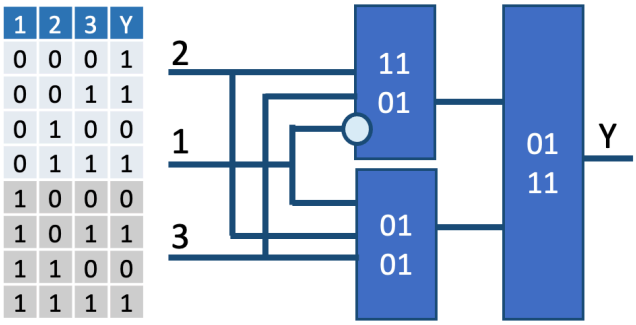


Рисунок 3.5 – Розподіл логічного вектору на два з застосуванням комутатору

3.5 Приклад синтезу схеми Шнейдера

Для порівняння процесів моделювання вектору та синтезу схеми розглядаються три варіанти схеми Шнейдера та таблиця логічного вектору (рис. 3.6):

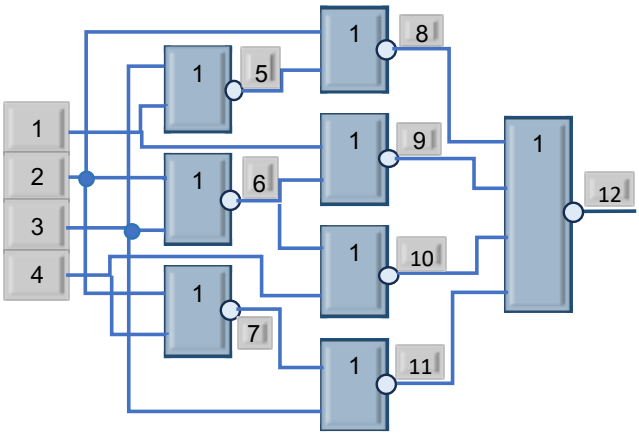
а) оригінал, синтезований евристично і представлений вісьмома компонентами і структурою з розгалуженнями, що сходяться;

б) схема, представлена одним елементом, векторно-логічна модель якого одержана за допомогою методу декартової логіки;

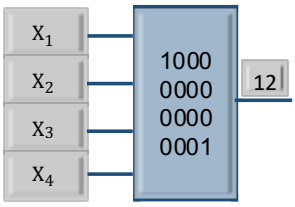
в) синтезована схема Шнейдера за отриманим логічним вектором шляхом побудови кон'юнктивних термів ДНФ за одиничними координатами вектора.

Всі розглянуті схеми є еквівалентними з точки зору функціональності. Однак, якщо говорити про їхню реалізацію в програмному або апаратному забезпеченні, то складність впровадження (алгоритму) оцінюється як 3:1:2.

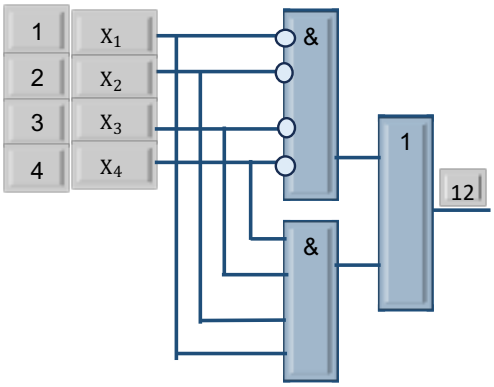
З точки зору діагностики, моделювання та тестування, складність алгоритмів обробки даних цими схемами характеризується співвідношенням 10:1:5.



а



б



в

Логічний вектор схеми Шнейдера (лінія 12)

123	234	000	001	010	011	100	101	110	111
	1000	0	1	1	0	1	1	1	0
000	0	1	0						
001	1			0	0				
010	1					0	0		
011	1							0	0
100	1	0	0						
101	1			0	0				
110	0					0	0		
111	0							0	1

г

Рисунок 3.6 – Три способи синтезу схеми Шнейдера та логічний вектор

Час затримки при формуванні вихідного сигналу для кожної схеми становить 3:1:2.

Енергоспоживання, необхідне для отримання результату, оцінюється як 8:1:3.

Оптимальним способом реалізації функціональності є використання логічного вектора, що зберігається в пам'яті. Моделювання та синтез можуть ефективно взаємодіяти, доповнюючи один одного. Моделювання передбачає створення інтелектуальної моделі в пам'яті, яку можна оптимізувати для підвищення швидкодії. Цю перевірену модель можна потім синтезувати для реалізації як функціональність системи на кристалі (SoC) з використанням швидких квантових, матричних механізмів або ASIC-чипів.

Моделювання – це процес одноразового створення моделі процесу або явища в пам'яті, призначеної для моделювання вхідних двійкових даних і отримання на виході двійкового результату, який представляє собою діагноз або прогноз.

Синтез – це одноразова процедура реалізації верифікованої моделі функціональності в систему елементів кристала ASIC, VLSI, FPGA, CPU, GPU, відповідно до можливостей технології.

Таким чином, моделювання та синтез, працюючи разом, дозволяють створити економічно вигідний обчислювальний пристрій з точки зору споживання енергії та часу.

3.6 Верифікація структур, що задані логічними векторами

Оскільки структура та функції мають ідентичну форму та зміст, логічний вектор є ефективним інструментом для їх опису.

У логічному векторі координати відповідають адресам, які в структурі (наприклад, графі) представляють собою переходи. Наявність переходу позначається одиницею у відповідній позиції логічного вектора, а відсутність – нулем. Отже, структуру (граф) можна представити у вигляді логічного

вектора, де двійкові коди переходів використовуються як адреси окремих бітів цього вектора.

З цього випливає, що тестування графа можна здійснювати як тестування логічної функції, представленої цим вектором. Це робить логічний вектор зручним і ефективним засобом для аналізу та перевірки структури графа.

Як протестувати граф, закодований у вигляді логічного вектора L = 11100111?

Вектор L описує наявність або відсутність переходів у графі. Кожен біт у векторі L відповідає певній адресі (вхідному набору) і вказує, чи повинен бути перехід за цією адресою (1) чи ні (0).

Несправності, пов'язані з адресацією, проявляються як помилки у переходах. Тому, кожен вхідний набір використовується для перевірки:

– якщо $L_i = 1$: чи дійсно є перехід за цією адресою?

– якщо $L_i = 0$: чи дійсно немає переходу за цією адресою?

Іншими словами, перевіряється, чи відповідає фактична поведінка графа очікуваній, закодованій у векторі L. Несправність виявляється, якщо перехід присутній там, де його не повинно бути (відповідно до L), або відсутній там, де він повинен бути. По суті, відшукуються випадки, коли реальний перехід суперечить значенню відповідного біта у векторі L.

Приклад 3.1. Двійковий набір 000 перевіряє несправність .11, яка отримана з адреси 011 який відповідає $L_{011} = 0$ вектора L=11100111.

Модель несправностей графа проста: несправності визначаються інверсією до розрядів логічного вектора. Дані дефекти корелюються з несправностями, що генеруються 1-розрядами адрес осередків логічного вектора.

3.7 Стратегія тестування графової структури

1. Створити тест, який забезпечить покриття всіх критичних точок функціональності, представленої вектором.

2. Створити тест, який перевіряє, чи виявлені всі можливі дефекти, пов'язані з вхідними змінними, що розглядаються як двійкові адреси, які визначають критичні стани логічного вектора.

Хоча 100% покриття поодиноких несправностей забезпечує тестову верифікацію цифрового виробу, не існує чіткого критерію функціонального покриття критичних станів, який би гарантував повну перевірку функціональності.

Тестування графа, представленого логічним вектором, можна звести до пошуку тесту, який виявляє несправності, що впливають на його функціональність. Для графа це означає перевірку наявності несправностей у всіх переходах, що визначаються метрикою станів логічного вектора.

Неможливо створити тест для графа (функції), який представлений логічним вектором, що складається лише з одиниць або лише з нулів, наприклад L=1111 (0000).

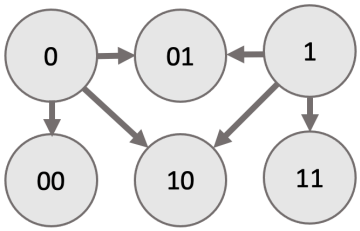
Мінімальний тест (011, 100) для графа L = 11100111 знаходиться шляхом покриття всіх поодиноких вхідних несправностей мінімальною кількістю двійкових наборів.

Перевіряються лише помилкові шляхи у метриці логічного вектора (рис. 3.7), правильні переходи не перевіряються.

Чотири вхідні комбінації (0011, 0100, 0101, 1010) утворюють тестовий набір, призначений для виявлення як одиночних, так і множинних дефектів, пов'язаних з переходами в графі, який описується таблицею істинності з чотирма змінними (рис. 3.8). Цей набір перевіряє, чи правильно відбуваються переходи, які впливають на вихідний логічний вектор L.

Зокрема, вхідні комбінації SE, для яких вихід L(S,E) має значення 0 (тобто, переходи, яких не повинно бути), можуть виявляти дефекти, що

призводять до помилкового формування переходів, які повинні давати вихід L(S,E) = 1.



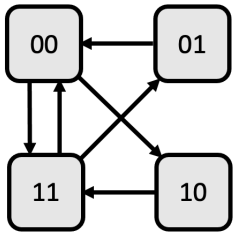
а

S	E	L
0	00	1
0	01	1
0	10	1
0	11	0
1	00	0
1	01	1
1	10	1
1	11	1

	000	001	010	011	100	101	110	111
000				.11	1..			
001			.1.			1.0		
010		..1					10.	
011		..0	.0.	.00	1..	1.0	10.	
100		..1	.1.	.11	0..	0.1	01.	
101		..0					01.	
110			.0.			0.1		
111				.00	0..			

б

Рисунок 3.7 – Карта тестування графа, синтезованого за логічним вектором



а

S	E	L
00	00	0
00	01	0
00	10	1
00	11	1
01	00	1
01	01	0
01	10	0
01	11	0
10	00	0
10	01	0
10	10	0
10	11	1
11	00	1
11	01	1
11	10	0
11	11	0

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000			..1.	..11	.1..							1.11	11..	11.1		
0001			..1.	..10		.1.0					1.1.		11..	11.0		
0010			..0.	..01	.1..	.1.1		.101	1...		1.0.	1.01	11..	11.1		
0011			..0.	..00	.1..	.1.0	.10.			1..0	1.0.	1.00	11..	11.0		
0100		...1	..1.	..11	.0..	.0.1					1.1.	1.11	10..	10.1	101.	
0101		...0					.01.	.010	1...	1..0					101.	
0110			..0.		.0..	.0.1					1.0.	1.01		10.1		
0111				..00	.0..	.0.0					1.0.	1.00	10..			
1000				..11	.1..	.1.1					0.1.	0.11	01..			
1001			..1.		.1..	.1.0					0.1.	0.10		01.0		
1010		...1					.10.	.101	0...	0..1					010.	
1011		...0	..0.	..00	.1..	.1.0					0.0.	0.00	01..	01.0	010.	
1100			..1.	..11	.0..	.0.1	.01.			0..1	0.1.	0.11	00..	00.1		
1101			..1.	..10	.0..	.0.0		.010	0...		0.1.	0.10	00..	00.0		
1110			..0.	..01		.0.1					0.0.		00..	00.1		
1111			..0.	..00	.0..							0.00	00..	00.0		

б

Рисунок 3.8 – Карта тестування для графа з чотирьох вершин

Аналогічно, вхідні набори, що відповідають правильним переходам L(S,E) = 1, перевіряють, чи не виникають помилкові переходи там, де їх не повинно бути L(S,E) = 0. Ці принципи враховуються при створенні карти тестування для графової структури, представленої у вигляді логічного вектора.

3.8 Висновки до розділу 3

Структура та функції еквівалентні поняття за формою та за змістом. Тому логічний вектор хороший для опису як функцій, і структур. Координати логічного вектора представлені адресами, які у структурі ототожнюються з переходами, зазначеними у логічному векторі одиничними координатами, якщо існує перехід. Якщо якогось переходу немає, йому відповідатиме нульове значення у відповідній позиції вектора.

Таким чином, модель структури (графа) – це логічний вектор, де двійково-кодовані переходи розглядаються як адреси поодиноких біт логічного вектора. Тому тестувати граф можна і потрібно як логічну функцію, представлену вектором. По карті тестування можна визначати суттєвість змінних та довжину тесту. Будь-які два взаємно-інверсні логічні вектори мають ідентичну карту тестування. За дедуктивною матрицею векторної моделі графа можна визначати булеві похідні. За логічним вектором графа можна моделювати справний стан будь-якої структури.

ВИСНОВКИ

1. Проаналізовано технологічні тенденції, що визначені компанією Gartner. Відзначено, що енергоефективні, гібридні та просторові обчислення вимагають нових підходів. Одним з них може виступати застосування нового напрямку – векторної логіки в архітектурі in-memory комп'ютингу на основі read-write транзакцій, що дозволяє зменшити споживані ресурси у вигляді енергії, часу.

2. Проаналізовано математичний апарат та технологічні рішення векторно-логічного комп'ютингу, значний внесок до розвитку теорії якого зробили вчені кафедри АПОТ, що відзначено у роботах [3-10].

3. Схема Schneider вже 50 років використовується для тестування механізмів моделювання справної поведінки, несправностей та генерації тестів. Вона хороша тим, що має всі структурні компоненти, які переводять її в розряд схеми з розгалуженнями, що сходяться. Ця схема була оброблена декартовою логікою, щоб отримати логічний вектор, який моделює справну поведінку цієї структури. Логічний вектор – це модель, яку використовують прості алгоритми синтезу тестів та моделювання несправностей. Цей вектор також був використаний для синтезу дворівневої схеми на основі ДНФ. Розглянуті в роботі процедури синтезу на основі векторно-логічних обчислювальних механізмів сприяють зниженню витрат на тестування та верифікацію цифрових проектів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 2025 Gartner Top 10 Strategic Technology Trends. Джин Альварес (Gene Alvarez), Gartner. October 10, 2024. [https://www.gartner.com/en/articles/top-technology-trends-2025]

2. Spotlight on 2024 Gartner Hype Cycle™ for Emerging Technologies Disruptive technologies hold great potential – for those businesses capable of overcoming the risks involved. Arun Chandrasekaran. October 10, 2024. [https://www.gartner.com/en/articles/hype-cycle-for-emerging-technologies]

3. S.-H. Chang, C. -N. J. Liu and A. Küster, "Behavioral Level Simulation Framework to Support Error-Aware CNN Training with In-Memory Computing," *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Villasimius, Italy, 2022, pp. 1-4, doi: 10.1109/SMACD55068.2022.9816307.

4. S. Li, N. Zhang, W. Zhang, R. Yang, Y. Chang and B. Xiong, "Framework Independent Modeling for SRAM-Based In-Memory Computing," *2024 2nd International Symposium of Electronic Design Automation (ISEDA)*, Xi'an, China, 2024, pp. 777-777, doi: 10.1109/ISEDA62518.2024.10617799.

5. J. Westphal and J. Hardy, "Logic as a Vector System," in *Journal of Logic and Computation*, vol. 15, no. 5, pp. 751-765, Oct. 2005, doi: 10.1093/logcom/exi040.

6. E. Mizraji, "Vector Logic: A Natural Algebraic Representation of the Fundamental Logical Gates," in *Journal of Logic and Computation*, vol. 18, no. 1, pp. 97-121, Feb. 2008, doi: 10.1093/logcom/exm057.

7. E. Mizraji, "Vector logic allows counterfactual virtualization by the square root of NOT," in *Logic Journal of the IGPL*, vol. 29, no. 5, pp. 859-870, May 2020, doi: 10.1093/jigpal/jzaa026.

8. Hahanov V. Vector Synthesis of Fault Testing Map For Logic / V. Hahanov, W. Gharibi, S. Chumachenko, E. Litvinova // IAES International Journal of Robotics

and Automation (IJRA). – 2024. – Vol. 13, No. 3. – Pp. 293-306. – DOI:10.11591/ijra.v13i3.pp293-306

9. Hahanov V. Vector-Logical In-Memory Simulation of Faults as Truth Table Addresses / V. Hahanov, E. Litvinova, H. Hahanova, S. Chumachenko, Z. Davitadze, I. Hahanova, H. Kulak, V. Ponomarova, V. H. Abdullayev // 2024 IEEE East-West Design & Test Symposium (EWDTS), Yerevan, Armenia, 2024, pp. 1-6, doi: 10.1109/EWDTS63723.2024.10873615.

10. Hahanov V. Prompt-Testing of Logic / V. Hahanov, D. Devadze, I. Hahanov, S. Chumachenko, E. Litvinova, V. Obrizan, P. Dmytro, A. Mishchenko, N. Maksymova // 2024 IEEE East-West Design & Test Symposium (EWDTS), Yerevan, Armenia, 2024, pp. 1-5, doi: 10.1109/EWDTS63723.2024.10873774.

11. Hahanov V. Faults-as-address simulation / V. Hahanov, S. Chumachenko, E. Litvinova, I. Hahanov, V. Ponomarova, H. Khakhanova, G. Kulak // IAES International Journal of Robotics and Automation. December 2024. Vol 13, No 4. P. 452-468. DOI: http://doi.org/10.11591/ijra.v13i4.pp452-468.

12. "IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)," in IEEE STD 1481-2009 , vol., no., pp.1-658, 11 March 2010.

13. E. Esmaieli, Y. Sedaghat and A. Peiravi, "Fanout-Based Reliability Model for SER Estimation in Combinational Circuits," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 72, no. 1, pp. 228-240, Jan. 2025, doi: 10.1109/TCSI.2024.3458864.

14. M.-E. Hwang, S.-O. Jung and K. Roy, "Slope Interconnect Effort: Gate-Interconnect Interdependent Delay Modeling for Early CMOS Circuit Simulation," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 7, pp. 1428-1441, July 2009, doi: 10.1109/TCSI.2008.2006217.

15. W. Huang, J. Du, W. Hua, K. Bi and Q. Fan, "A Hybrid Model-Based Diagnosis Approach for Open-Switch Faults in PMSM Drives," in *IEEE Transactions on Power Electronics*, vol. 37, no. 4, pp. 3728-3732, April 2022, doi: 10.1109/TPEL.2021.3123144.

16. P. Wang et al., "RC-NVM: Enabling Symmetric Row and Column Memory Accesses for In-memory Databases," 2018 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Vienna, Austria, 2018, pp. 518-530, doi: 10.1109/HPCA.2018.00051.

17. B. Ahn, J. Jang, H. Na, M. Seo, H. Son and Y. H. Song, "AI Accelerator Embedded Computational Storage for Large-Scale DNN Models," 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Incheon, Korea, Republic of, 2022, pp. 483-486, doi: 10.1109/AICAS54282.2022.9869991.

18. M. Davis, "Emil Post's contributions to computer science," Proceedings. Fourth Annual Symposium on Logic in Computer Science, 1989, pp. 134-136.