УДК 681.326

ВЕКТОРНО-ЛОГІЧНЕ МОДЕЛЮВАННЯ НЕСПРАВНОСТЕЙ

Хаханов В. І. − д-р техн. наук, професор кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Чумаченко С. В. – д-р техн. наук, професор, завідувач кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Литвинова Є. І. – д-р техн. наук, професор кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Хаханова І. В. – д-р техн. наук, професор кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Хаханова Г. В. − канд. техн. наук, доцент кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Шкіль О. С. − канд. техн. наук, доцент кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Рахліс Д. Ю. − канд. техн. наук, доцент кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Хаханов І. В. − аспірант кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

Шевченко О. Ю. − канд. техн. наук, асистент кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки, Україна.

АНОТАЦІЯ

Актуальність. Основна ідея – створення векторно-логічного in-memory комп'ютингу (ВЛК), який використовує лише read-write транзакції на адресної пам'яті для моделювання несправностей, як адрес. Традиційна логіка відсутня. ВЛК вільний від команд процесора та АЛП для організації обчислень і тому орієнтований на імплементацію у кристали SoC і FPGA. Пропонується векторно-логічний метод синтезу дедуктивних матриць для транспортування вхідних несправностей, який має квадратичну обчислювальну складність.

Мета. Розробка векторного дедуктивного методу моделювання несправностей на основі примітивних read-write-транзакцій для аналізу логічних схем.

Метод. Використовується вхідний тестовий набір та логічний вектор функціональності. Метод, що розробляється, є розвитком алгоритму синтезу дедуктивних векторів на основі таблиці істинності. Дедуктивна матриця призначена для синтезу та верифікації тестів за допомогою паралельного моделювання комбінацій несправностей, як адрес, на основі read-write-транзакцій над бітами дедуктивних векторів, що знаходяться в пам'яті.

Результати. Запропоновано векторний метод синтезу дедуктивних матриць для транспортування вхідних несправностей векторів на вихід елемента. Розроблено структури даних для паралельного моделювання несправностей цифрових схем на основі примітивної read-write-транзакції в матричній пам'яті, де поєднання несправностей є стовпцями-адресами. Запропоновано секвенсор із п'яти блоків, що складають векторно-логічний комп'ютинг, пов'язаний з дедуктивним моделюванням несправностей на основі read-write транзакцій. Виконано верифікацію моделей та методів на тестових прикладах.

Висновки. Наукова новизна полягає у розробці наступних інноваційних рішень: 1) вперше запропоновано векторно-логічний метод синтезу матриці дедуктивних векторів для паралельного моделювання комбінацій вхідних несправностей як адрес; 2) вперше запропоновано автомат векторно-дедуктивного моделювання несправностей, як адрес, на основі read-write транзакцій, орієнтований для імплементації в FPGA LUT, вбудований online симулятор SoC, як ядро для моделювання несправностей цифрових систем RTL-рівня; 3) демонстрація технологічних переваг векторно-логічного синтезу дедуктивних матриць виконана на численних прикладах традиційної та RTL-логіки, що підкреслює технологічність векторів у порівнянні з аналітичними дедуктивними формулами для побудови симуляторів; 4) матриця дедуктивних векторів, як сукупність вектор-стовпців булевих похідних використовується для побудови мінімальних тестів для логічних елементів; 5) рекурсивна формула синтезу матриці перестановки координат у логічному векторі активності дозволяє суттєво спростити отримання дедуктивної матриці для моделювання несправностей як адрес. Практичне значення полягає в тому, що in-memory simulator дозволить отримати швидкодію моделювання несправностей реальних цифрових блоків SoC на рівні сотень наносекунд. Наводяться оцінки складності відповідних алгоритмів.

КЛЮЧОВІ СЛОВА: векторний комп'ютинг, векторна форма логіки, матриця дедуктивних векторів, векторний метод синтезу дедуктивної матриці, read-write транзакції, векторна модель дефектів, векторно-логічне дедуктивне моделювання несправностей.

АБРЕВІАТУРИ

АЛП – аріфметико-логічний пристрій;

ВЛК – векторно-логічний комп'ютинг;

ДДНФ – досконала диз'юнктивна нормальна форма;

BIST – built-in self-test (вбудована самоперевірка);

CIM – compute in-memory (обчислення безпосередньо у пам'яті);

DOI 10.15588/1607-3274-2023-2-5

CPU – central processing unit (центральний процесор);

CS – computational storage (обчислювальне сховище);

DNN – deep neural networks (глубокі нейронні мережі);

EDA – electronic design automation (система автоматизованого проектування);

FPGA − field-programmable gate array (програмована користувачем вентильна матриця);

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

IP-core – intellectual property core (складний функціональний блок);

LUT – look-up table (таблиця пошуку);

NLP – natural language processing (обробка природної мови);

NMC – near-memory computing (обчилення близько до пам'яті);

PIM-computing – processing-in-memory computing (обчислення з обробкою в пам'яті);

RTL – register transfer level (рівень регістрових передач);

SoC – system on chip (системи на кристалі).

НОМЕНКЛАТУРА

C – обчислювальна складність;

d_i – відстань;

D_{ij} – елемент D-матриці дедуктивних векторів;

D_x – дедуктивний вектор стану виходів таблиці істинності несправностей, що перевіряються на тестових двійкових наборах x ;

F – функціональність;

F_{ij} – елемент F-матриці несправностей;

H^i – i-елемент H-матриці;

L – несправність;

L_f – вектор несправностей, що перевіряються;

L_H – L-матриця, що перекодована за допомогою матриці H;

L_i – i-строка L-матриці;

n − кількість входів;

Q_i –i-елемент Q-вектора;

T – тест;

tD – час генерації дедуктивного вектора;

tF – час обробки вхідних векторів несправностей;

ΔT – час обробки одного логічного елемента;

U – номер лінії;

x – вхідний тестовий набір;

X – таблиця істинності одиночних та кратних вхідних константних несправностей;

X_i – вхідна змінна;

X_i' – похідна за змінною X_i;

Y – вихідна змінна;

Z – значення несправності 0 або 1.

ВСТУП

Мотивація дослідження продиктована такими факторами: застосування елементарних транзакцій читання-запис (read-write transactions) для обчислень на основі пам'яті (memory-driven computing) з векторним описом логіки, наявність пам'яті для зберігання гнучких моделей логіки, використання даних як адрес для підвищення швидкодії дедуктивного моделювання, перехід комп'ютингу на нижчий рівень обчислювальних процесів (read-write transaction), де можна не дотримуватися архітектури фон Неймана (Neuman architecture) та теорему Поста-Яблонського про функціональну повноту [2–3]. Сутність векторного комп'ютингу – транзакції читання-запис на векторних структурах даних в адресній пам'яті. Актуальність цього напряму видно з останнього дослідження компанії Gartner, яка опублікувала як тригерний тренд обчислювальне сховище (computational storage, CS), який переносить обробку даних із центрального процесора у пам'ять, де вони знаходяться [4]. Великі дані мають оброблятись за місцем їх зберігання [5]. Одне з таких рішень – векторний комп'ютинг – обчислювальний процес на основі read-write транзакцій над бітами двійкового вектора функціональності, що формує обчислювальне сховище, де вхідні дані є адресами бітів вектора логіки. Векторно-логічний комп'ютинг розглядається як квазіоптимальний варіант вирішення протиріччя між потужною системою команд процесора (Neuman architecture) та бітових структур великих даних, що підлягають обробці. Суть пропонованого комп'ютингу – обробка великих даних, як адрес, за місцем їх зберігання за допомогою read-write транзакцій над бітами векторної логіки, що становить обчислювальну пам'ять (computational memory). Актуальність застосування In-Memory Computing (IMC), Near-Memory Computing (NMC), Processing-In-Memory (PIM) комп'ютингу підтверджується наступними публікаціями [6–10]. Тут можна виділити наступні метрики ефективного використання IMC.

1. Створення моделі глибоких нейронних мереж (DNN) особливо в галузі обробки природної мови (NLP), що скорочує енергетичну потужність використання програми на 36.3%, і підвищує продуктивність алгоритму великих даних 22.6% [6] .

2. Основна мета IMC-архітектур максимально знизити енергоспоживання підвищення автономності з допомогою скорочення передачі між пам'яттю і обчислювальним блоком з допомогою побітових логічних операцій (NOT, AND, OR, XOR) всередині масиву пам'яті та поруч із ним, просуваючи концепцію обчислень у пам'яті (CIM) [7].

3. Пропускна здатність передачі даних та пов'язане з нею енергоспоживання стали найбільш критичним вузьким місцем у обчислювальній архітектурі фон Неймана через поділ процесора та пам'яті. Усвідомлення єдності обчислень та пам'яті в одному місці відкрило перспективний напрямок досліджень обчислень у пам'яті [8].

4. Сучасні програми повинні бути орієнтовані на обробку все більших даних.

DOI 10.15588/1607-3274-2023-2-5

Тенденція у технологіях пам'яті не зростає так швидко, як продуктивність обчислень, що призводить до так званої стіни пам'яті. В даний час вивчаються нові архітектури для вирішення цієї проблеми як для вбудованої, так зовнішньої пам'яті. Останні методи, що максимально наближають обчислення до масиву пам'яті, такі як обчислення в пам'яті (IMC), обчислення ближньої пам'яті (NMC), обробка пам'яті (PIM), дозволяють знизити вартість переміщення даних між обчислювальним ядром і пам'яттю. Для вбудованих обчислень схема In-Memory Computing забезпечує вигідні обчислення та виграш енергії до 78% для певного класу додатків [9]. Даному виду IMC-комп'ютингу присвячено більше 64 тисяч публікацій в IEEE Xplore, що свідчить про актуальність проблеми аналізу великих даних за допомогою простих CPU-free моделей, розміщених у пам'яті. Разом з цим запропоновані рішення, спрямовані на створення обчислювача у пам'яті, де зберігаються великі дані. Наступним кроком буде створення In-Data-Computing, без логіки та схем обробки у метриці операції зчитування-запису (read-write transaction) над бітами великих даних.

Об'єкт дослідження – in-memory комп'ютинг, який знижує енергетичні та часові витрати при обробці великих даних.

Предмет дослідження – in-memory моделювання логічних компонентів SoC будь-якої розмірності за допомогою read-write транзакцій на логічних векторах.

Мета дослідження – розробка векторного дедуктивного методу моделювання несправностей на основі примітивних read-write транзакцій для аналізу логічних схем.

1 ПОСТАНОВКА ЗАДАЧІ

Вирішується проблема перенесення архітектури фон Неймана до пам'яті та заміни потужного процесора read-write транзакціями на логічних векторах для зниження енергетичних та часових витрат при моделюванні логічних функціональностей будь-якої розмірності.

Нехай задані моделі вхідних даних, що можуть бути представлені (рис. 1):

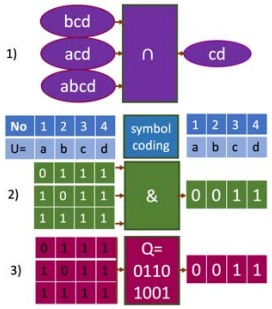

Рисунок 1 – Структури вхідних даних

1) множинами – компактні дані, що вимагають складного та послідовного за входами алгоритму їх обробки; 2) векторами – унітарне кодування (symbol coding) даних на універсумі примітивів, що використовує паралельний регістровий алгоритм їхньої обробки, послідовний по входах; 3) адресами – компактне кодування унітарних даних на універсумі примітивів та послідовний алгоритм їх обробки read-write транзакціями на пам'яті без логіки та процесора з паралельністю за стовпцями-адресами.

Нехай вектор Q = 01101001 являє собою xor-функцію від трьох змінних, які формують адреси даних, що підлягають аналізу.

Тут і далі Q-вектор переходить з частини таблиці істинності у самостійну компактну форму для завдання функціональності або структури з метою створення векторно-логічного комп'ютингу обробки великих даних, що розміщені у адресній пам'яті (CS).

Задача полягає у: 1) розробці векторного методу синтезу дедуктивних матриць для транспортування вхідних несправностей векторів на вихід елемента; 2) розробці структури даних (дедуктивної матриці D, логічного вектору Q, матриці перекодування H, комбінацій несправностей на входах елементу X), для паралельного моделювання несправностей цифрових схем на основі примітивної read-write-транзакції в матричній пам'яті, де поєднання несправностей служать стовпцями-адресами; 3) розробці п'яти блоків секвенсора (рис. 2), що становлять векторно-логічний комп'ютинг, пов'язаний з дедуктивним моделюванням несправностей на основі read-write транзакцій.

Рисунок 2 – Структура дослідження

2 ОГЛЯД ЛІТЕРАТУРИ

Основні проблеми в галузі моделювання несправностей слід визначити таким чином.

1. Висока обчислювальна складність алгоритмів комбінаторного аналізу на регістровому рівні опису моделі [10–12].

2. Складність алгоритмів моделювання та симуляції послідовних схем, пов'язана з непередбачуваною кількістю ітерацій [11].

3. Значний обсяг структур даних для аналізу цифрових систем на кристалі, що негативно позначається на продуктивності методів моделювання несправностей та синтезу тестів [15–21].

4. Складність аналізу та синтезу тестів для логіки великої розмірності та їх структурна і комбінаторна складність, пов'язана з розгалуженнями, що збігаються [10].

DOI 10.15588/1607-3274-2023-2-5

5. Паралельне розв'язання завдань тестування та імітації цифрових пристроїв [9–12].

У design and test використовуються три основні форми опису процесів та явищ: таблична, аналітична, графова [10–21]. При цьому матриця (таблиця) і вектор є дві форми опису моделей, що переходять один до одного. Матриця, при необхідності, розпускається в одновимірний вектор для зручності паралельної обробки даних регістрової пам'яті. Природно, досить просто відновити таблицю чи матрицю з векторної форми опису процесу чи явища. Вектор Q (двійковий, багатозначний) є компактним видом таблиці істинності як упорядкованої послідовності станів виходу, якщо вхідні компоненти-адреси впорядковані за зростанням [11–20].

Криза сучасного комп'ютингу пов'язана з двома проблемами: обробкою великих даних неспроможною за часовими витратами парою процесор-пам'ять, а також катастрофічним збільшенням споживання електроенергії глобальними обчислювальними процесами на сучасній мікроелектроніці. Вирішення першої проблеми грунтується на аксіомах: "Всі дані знаходяться у пам'яті". Немає логіки, яку не можна було б реалізувати на пам'яті. Немає даних, які не можна було б використовувати як адреси для обробки на пам'яті, де вміщена логіка. Не існує логіки або функцій, які не можна було б реалізувати за допомогою read-write транзакції на пам'яті. Найтехнологічніша структура даних для пам'яті є вектор або матриця, доступні для швидкодіючих read-write транзакцій. Де є великі дані, там необхідно будувати простий обчислювач для їх аналізу.

3 МАТЕРІАЛИ І МЕТОДИ

Розглянемо основні визначення.

Логічний елемент – будь-яка двійкова функціональність від n змінних, з невизначеною внутрішньою структурою, що задається логічним вектором, розмірністю 2^n.

Логічний вектор – явна форма завдання функціональності за допомогою впорядкованої послідовності 2^n бітів, де кожен біт має свою двійкову адресу в метриці n змінних. Логічний вектор разом із впорядкованою сукупністю явних двійкових адрес утворюють таблицю істинності.

Вхідний двійковий набір – вхідна тестова послідовність з n двійкових бітів, які забезпечують транспортування комбінації вхідних несправностей елемента з його вихід.

Дедуктивний вектор – логічна функціональність, задана на вхідному двійковому наборі, що забезпечує транспортування комбінації вхідних несправностей на вихід елемента.

Таблиця істинності несправностей – явна комбінаторна форма завдання всіх можливих варіантів вхідних несправностей одиничними значеннями координат на множині n інверсних значень змінних вхідного двійкового набору.

Дедуктивна матриця – явна форма завдання 2^n дедуктивних векторів у форматі логічного вектора 2^n, кожен із яких на вхідному двійковому наборі, довжиною n, формує функціональність транспортування комбінації вхідних несправностей на вихід елемента.

Матриця перекодування бітів – сукупність номерів-індексів у метриці декартового добутку логічного вектора $2^n \times 2^n$ призначена для синтезу дедуктивної матриці з логічного вектора або його інверсії.

Розглянемо векторний метод синтезу матриці дедуктивної для моделювання несправностей.

На початку слід розглянути побудову H-матриці перестановок, яка є основою синтезу дедуктивних матриць. Для генерації H-матриці необхідно знати лише n – кількість вхідних змінних будь-якої функціональності. Матриця H відстежує закономірності зміни кожної вхідної змінної у просторі 2^n рядків таблиці істинності не враховуючи станів виходів. Іншими словами, H-матриця забезпечує потенційні умови активізації вхідних змінних (несправностей). Реальні умови активізації буде згенеровано шляхом суперпозиції матриці з логічним вектором конкретної функціональності (рис. 3). Тому H-матриця є ядерною незмінною структурою для технологічного синтезу D-матриці дедуктивних векторів.

Рисунок 3 – Роль H-матриці у формуванні дедуктивної моделі

Синтез H-матриці перестановки координат використовує технологічно просту рекурсивну формулу $H^i = \begin{bmatrix} H_1^{i-1} & H_2^i \\ H_3^i & H_4^{i-1} \end{bmatrix}$, де $i = 1,2,3,....$ – кількість вхідних змінних.

Алгоритм синтезу представлений трьома пунктами (рис. 4):

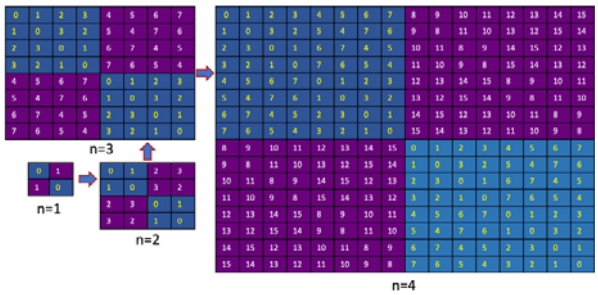

Рисунок 4 – Синтез L-матриці перекодування векторів

1) перша та четверта чверть матриці перестановок береться від попереднього рекурсивного обчислення матриці для $n = i - 1$ змінних, тут виконуються рівно-

DOI 10.15588/1607-3274-2023-2-5

сті: $H_1^{i-1} = H_4^{i-1}, H_2^{i-1} = H_3^{i-1}$, при цьому $H_0^0 = 0, H_1^1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$;

2) друга чверть матриці знаходиться на підставі виразу $H_{i,j+2^{n-1}} = (2^n - 1) - H_{i,j}$, $j = \overline{1,2^{n-1}}$, $i = \overline{1,2^{n-1}}$;

3) третя чверть матриці знаходиться шляхом копіювання другої частини матриці до третьої області $H_3^i = H_2^i$.

Метод синтезу матриці дедуктивних векторів за Q-вектором має лише два пункти.

1. Отримання матриці модифікованих L-векторів функціонального елемента (рис. 4) на вхідному i-наборі за правилом: $L_i = Q \oplus Q_i$. Для виконання цього пункту Q-вектор записується по горизонталі та вертикалі L-матриці. Потім аналізується кожна координата Q-вектора, записаного по вертикалі Q_i. Якщо Q_i – координата вертикального Q-вектора дорівнює одиниці, виконується інверсія горизонтального вектора Q і його запис в поточний рядок матриці $L_i = \overline{Q}$. В іншому випадку в поточний рядок L-матриці записується Q-вектор без зміни: $L_i = Q$.

2. Визначення всіх бітів дедуктивної матриці за формулою: $D_{ij} = L_{H_{IJ}}$, $j = \overline{1,2^n}$.

Розглянемо векторний метод синтезу дедуктивної матриці традиційної логіки.

Математична основа дедуктивного моделювання несправностей полягає у транспортуванні на вихід двійкових комбінацій вхідних дефектів через функціональність F на заданому вхідному тестовому наборі T за формулою $L = T \oplus F$. Сутність дедуктивного моделювання полягає у зміні логіки елемента F залежно від вхідних умов T. Дедуктивне моделювання, запропоноване рівно 50 років тому Армстронгом [22], досі є найвитонченішим та найефективнішим засобом аналізу якості тестів та синтезу таблиць для пошуку дефектів, простежування шляху розповсюдження несправності. Далі пропонується його реалізація на основі векторної форми опису логіки [11–21], яка виключає логічні аналітичні форми, що дає можливість суттєво спростити алгоритми синтезу дедуктивних моделей та їх застосування для інтерпретативного моделювання цифрових елементів та схем великої розмірності. Мета – прибрати всі аналітичні вирази дедуктивної логіки, що фігурують у більшості робіт [6–9, 20], присвячених дедуктивному моделюванню. Можна і потрібно використовувати лише вектор дедуктивного моделювання несправностей. Дуже часто форма моделі визначає зміст, компактність та швидкодія процедур аналізу. Базові методи дедуктивного моделювання [10, 20, 22] зводилися до синтезу аналітичних дедуктивних форм (ДНФ). Інформація про логічний елемент була представлена таблицею істинності чи логічним виразом [1]. Але у будь-якому випадку, явно чи не явно, використовувалося рівняння технічної діагностики $L = T \oplus F$ для отримання дедуктивних форм моделювання несправностей.

Далі пропонується новий метод синтезу дедуктивних векторів, який зводиться до аналізу Q-вектора вихідних станів логічного n-входового елемента без таблиці істинності. Метод синтезу дедуктивних матриць за Q-вектором для традиційних логічних елементів на операторі $D = (Q \oplus Q_i)_H$ продемонстровано на рис. 5.

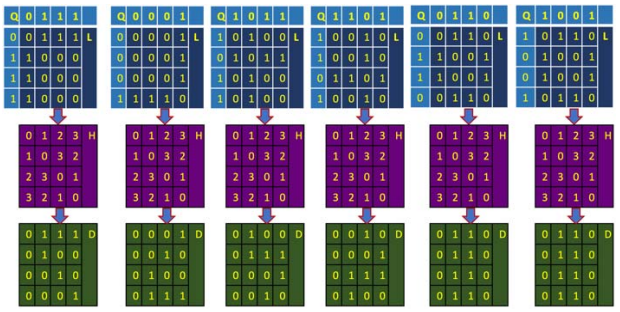

Рисунок 5 – Синтез дедуктивних векторів для базових логічних елементів

Спочатку записується логічний вектор Q по горизонталі та вертикалі першої Q-матриці синього кольору. Потім здійснюється запис вектора $Q_i = Q$ без зміни якщо по вертикалі $Q_i = 0$. Якщо $Q_i = 1$, то виконується інверсія логічного вектора $Q_i = \overline{Q}$. Матриця H є індексами для перестановки координат матриці Q. Зелена матриця D являє собою результат перестановки вмісту координат матриці Q за допомогою матриці H: $D = (Q \oplus Q_i)_H$.

Слід зазначити, що синтез дедуктивних формул для взаємно інверсних елементів: $Q = 0110$ і $Q = 1001$ дає однакові значення матриці дедуктивних векторів, які вироджуються в один вектор 0110 на всіх вхідних наборах. Обчислювальна складність (computational complexity) виконання цього оператора дорівнює: $C = 2 \times 2^n \times 2^n = 2^{2n+1}$. У разі паралельного виконання регістрових операцій над векторами обчислювальна складність даного оператора дорівнюватиме: $C = 2 \times 2^n$. Найбільш примітивними елементами є інвертор ($Q = 10$) та повторювач $Q = 01$). Незважаючи на те, що це різні елементи, вони мають однакові дедуктивні вектори, які дозволяють проводити цифрову логічну активність з транспортування вектора несправностей від входу до виходу, не спотворюючи його (рис. 6).

DOI 10.15588/1607-3274-2023-2-5

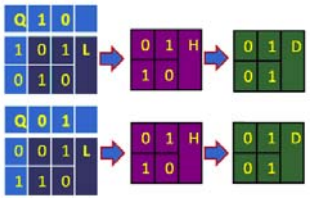

Рисунок 6 – Синтез дедуктивних векторів для примітивних взаємно інверсних елементів: 10 та 01

Розглянемо метод синтезу дедуктивних векторів RTL-логіки.

На рис. 7 представлено синтез дедуктивних векторів для трьох-входового хог-елемента, заданого вектором Q = 01101001. Отриманий результат має унікальну властивість: всі 2^n дедуктивних векторів хог-функціональності рівні між собою D = 01101001 і дорівнюють вихідному вектору Q = 01101001 цієї функції. Аналітична форма хог-функції дорівнює її дедуктивній функції, яка представлені наступною ДДНФ:

$$Y = \overline{X_1}\,\overline{X_2}X_3 \vee \overline{X_1}X_2\overline{X_3} \vee X_1\overline{X_2}\,\overline{X_3} \vee X_1X_2X_3\,.$$

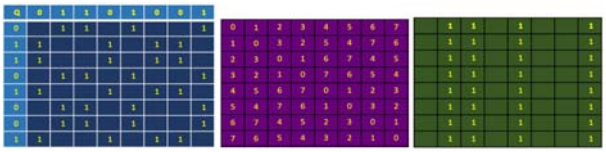

Рисунок 7 – Синтез дедуктивних векторів для трьох-входового елемента XOR

ДДНФ може бути записана диз'юнкцією двійкових кодів $Y = 001 \vee 010 \vee 100 \vee 111$. Перші три терми виконують роль фільтрів для визначення відмінностей на кожному з трьох входів. Останній терм визначає схожість даних на всіх трьох входах. Для двох-входового хог-елемента $Y = \overline{X_1}X_2 \vee \overline{X_1}X_2$ працює тільки фільтр відмінності вхідних даних, а вектор $Q = 0110$ складений конкатенацією векторів інвертора 10 і повторювача 01, які мають однакові дедуктивні вектори, рівні 01. Чотирьох-входова хог-функція має ДДНФ

$$Y = \overline{X_1}\,\overline{X_2}\,\overline{X_3}X_4 \vee \overline{X_1}\,\overline{X_2}X_3\overline{X_4} \vee \overline{X_1}X_2\overline{X_3}\,\overline{X_4} \vee$$
$$\vee X_1\overline{X_2}\,\overline{X_3}\,\overline{X_4} \vee \overline{X_1}X_2X_3X_4 \vee X_1\overline{X_2}X_3X_4 \vee$$
$$\vee X_1X_2\overline{X_3}X_4 \vee X_1X_2X_3\overline{X_4},$$

або

$$Y = 0001 \vee 0010 \vee 0100 \vee 1000 \vee$$
$$\vee 0111 \vee 1011 \vee 1101 \vee 1110,$$

яка відповідає наступному вектору $Q = 0110100110010110$. Перші чотири терми визначають фільтр відмінності вхідних даних, інші терми – подібності на всіх комбінаціях з трьох входів. Рекурсивна схема синтезу хог-векторів для входів n = 1,2,3,4,5,6,7,8 показано на рис. 8.

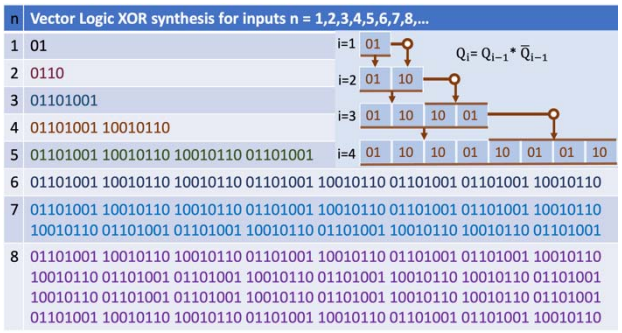

Рисунок 8 – Синтез Q-векторів для n-вхідного XOR

Тут кожен наступний вектор обчислюється з урахуванням рекурсивного висловлювання $Q_i = Q_{i-1} * \overline{Q_{i-1}}$, що містить конкатенацію двох векторних компонентів попереднього рівня. Вектор 01, при $n = 1$ слід розглядати як вироджену хог-функцію від однієї змінної. Метрика хог-функції: 1) хог-функція не підлягає мінімізації; 2) число одиниць та нулів у Q-векторі однакове; 3) всі дедуктивні вектори рівні між собою та рівні Q-вектору; 3) є єдиною функцією для вимірювання будь-яких процесів та явищ у кіберпросторі $\bigoplus_{i=1}^{n} d_i = 0$; 4) формує рівняння технічної діагностики $T \oplus F \oplus L = 0$ для вирішення завдань комп'ютингу; 5) здатна вимірювати подібності та відмінності вхідних даних як адрес; 6) з будь-якого входу пропускає будь-яку активність до виходу, не вимагаючи жодних умов активізації; 7) хог-функція утворює транзитивне замикання з функціями and, or: $\wedge \oplus \vee = \oplus$ або $\wedge \oplus \vee \oplus \oplus = 0$, або у векторній формі: $0001 \oplus 0111 \oplus 0110 = 0000$; 8) для хог-функції Q-вектор є компактною формою.

Розглянемо матрицю дедуктивних векторів, як вектор-стовпець булевих похідних.

Два поняття векторно-булева похідна та дедуктивна матриця мають деякі загальні метричні властивості: 1) обидва використовують характеристичне рівняння технічної діагностики на основі хог-операції: $L = T \oplus F$; 2) мають одну і ту ж матрицю перекодування бітів $H^i = \begin{bmatrix} H_1^{i-1} & H_2^i \\ H_3^i & H_4^{i-1} \end{bmatrix}$; 3) обидві спрямовані на транспортування будь-якої вхідної активності на виходи логічної схеми. Залишається вирішити питання, як взаємодіють між собою ці два компоненти. Для цього використовуємо дедуктивну матрицю логічної вектор-функції 10000001 від трьох змінних, яка наведена на рис. 9. Тут порожні клітини у таблицях позначають нульові стани.

DOI 10.15588/1607-3274-2023-2-5

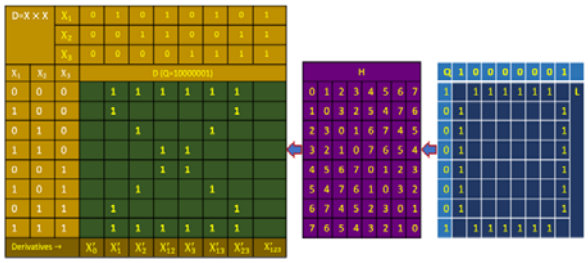

Рисунок 9 – Синтез дедуктивної матриці для симетричної функції $Q = 10000001$

Матриця D є декартовим добутком множини вхідних змінних $D = X \times X$, яке отримано на основі наступної формули $D = (Q \oplus Y)_H$. Рядки матриці є дедуктивними векторами транспортування вхідних активностей, а стовпці матриці формують векторні похідні або умови транспортування вхідних активностей. Крім того, стовпці та рядки дедуктивної матриці відзначені двійковими кодами вхідних змінних.

Булева векторна похідна по змінній X_i – це функція від n–1 змінної, яка виключає змінну, за якою береться похідна:

$$\frac{dF}{dX_i} = X_i^{'} = f(X_1, X_2, ..., X_i = 0, ..., X_n) \oplus f(X_1, X_2, ..., X_i = 1, ..., X_n).$$

Фактично булева похідна – це є умова активізації змінної X_i, яка повинна бути виставлена на всіх n–1 вхідних змінних, за винятком змінної X_i. Дотримуючись цього визначення, можна записати ДНФ булевих похідних по стовпчиках дедуктивної матриці:

$$X_0^{'} = X_{123}^{'} = 0.$$

$$X_1^{'} = X_{23}^{'} = 000 \vee 100 \vee 011 \vee 111 = X00 \vee X11 = \overline{X_2}\,\overline{X_3} \vee X_2X_3.$$

$$X_2^{'} = X_{13}^{'} = 000 \vee 010 \vee 101 \vee 111 = 0X0 \vee 1X1 = \overline{X_1}\,\overline{X_3} \vee X_1X_3.$$

$$X_3^{'} = X_{12}^{'} = 000 \vee 110 \vee 001 \vee 111 = 00X \vee 11X = \overline{X_1}\,\overline{X_2} \vee X_1X_2.$$

Дедуктивні матриці для трьох-входової логіки представлені такими таблицями (рис. 10). Тут порожніми клітинами позначені нульові координати.

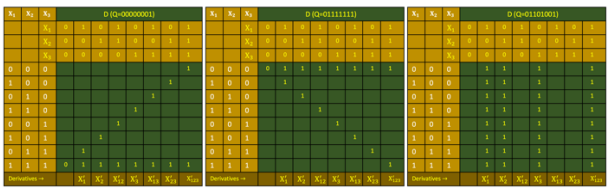

Рисунок 10 – Дедуктивні матриці для трьох-входової логіки

Стовпці дедуктивної матриці є похідними по одній, двом і трьом змінним. Йдеться про умови, які забезпечують активізацію виходу елемента залежно від кількості активних: одного, двох або трьох входів.

ДНФ булевих похідних по стовпчиках дедуктивних матриць мають вигляд:

$$X_1^{'} = (Q = 00000001) = 011 \vee 111 = X11 = X_2X_3.$$

$$X_2^{'} = (Q = 00000001) = 101 \vee 111 = 1X1 = X_1X_3.$$

$$X_3^{'} = (Q = 00000001) = 110 \vee 111 = 11X = X_1X_2.$$

$$X_{12}^{'} = (Q = 00000001) = 001 \vee 111 = \overline{X_1}\,\overline{X_2}X_3 \vee X_1X_2X_3.$$

$$X_{13}^{'} = (Q = 00000001) = 010 \vee 111 = \overline{X_1}X_2\overline{X_3} \vee X_1X_2X_3.$$

$$X_{23}^{'} = (Q = 00000001) = 100 \vee 111 = X_1\overline{X_2}\,\overline{X_3} \vee X_1X_2X_3.$$

$$X_{123}^{'} = (Q = 00000001) = 000 \vee 111 = \overline{X_1}\,\overline{X_2}\,\overline{X_3} \vee X_1X_2X_3.$$

$$X_1^{'} = (Q = 01111111) = 000 \vee 100 = X00 = \overline{X_2}\,\overline{X_3}.$$

$$X_2^{'} = (Q = 01111111) = 000 \vee 010 = 0X0 = \overline{X_1}\,\overline{X_3}.$$

$$X_3^{'} = (Q = 01111111) = 000 \vee 001 = 00X = \overline{X_1}\,\overline{X_2}.$$

$$X_{12}^{'} = (Q = 01111111) = 000 \vee 110 = \overline{X_1}\,\overline{X_2}\,\overline{X_3} \vee X_1X_2\overline{X_3}.$$

$$X_{13}^{'} = (Q = 01111111) = 000 \vee 101 = \overline{X_1}\,\overline{X_2}\,\overline{X_3} \vee X_1\overline{X_2}X_3.$$

$$X_{23}^{'} = (Q = 01111111) = 000 \vee 011 = \overline{X_1}\,\overline{X_2}\,\overline{X_3} \vee \overline{X_1}X_2X_3.$$

$$X_{123}^{'} = (Q = 01111111) = 000 \vee 111 = \overline{X_1}\,\overline{X_2}\,\overline{X_3} \vee X_1X_2X_3.$$

$$X_1^{'} = X_2^{'} = X_3^{'} = X_{123}^{'}(Q = 01101001) = 000 \vee 001 \vee 010 \vee 011 \vee 100 \vee 101 \vee 110 \vee 111 = XXX = 1.$$

В останньому випадку похідна за будь-якою вхідною змінною дорівнює одиниці. Це означає інваріантність стану входів для активізації будь-якої вхідної змінної.

Цікавим є той факт, що якщо отримані похідні або умови активізації всіх вхідних змінних – це означає, що практично побудований повний квазімінімальний тест для поодиноких несправностей вхідних та вихідних змінних цього функціонального елемента. Формальний або аналітичний запис такого тесту має вигляд: $T = (0 \vee 1) * X_i^{'}$, де $i = \overline{1, n}$, а символ $*$ – операція конкатенації. Як приклад $(0 \vee 1) * (X_2X_3) = 011 \vee 111$ – тест для перевірки першого входу елемента 3and.

Розглянемо векторно-табличне дедуктивне моделювання несправностей логіки для підвищення продуктивності та якості моделювання несправностей логічних схем шляхом суперпозиції таблиці істинності одиночних та кратних константних несправностей, що перевіряються на двійковому тестовому наборі та матриці дедуктивних векторів. Тут відокремлюються такі задачі: 1) синтез структур даних та алгоритму отримання дедуктивної матриці за логічним Q-вектором; 2) розробка таблиці істинності як моделі вхідних несправностей логіки для визначення якості тесту; 2) and-суперпозиція таблиці істинності та векторів дедуктивної матриці для аналізу несправностей, що перевіряються на двійкових тестових наборах; 3) алгоритм спільного аналізу таблиць істинності пе-

DOI 10.15588/1607-3274-2023-2-5

ревірки вхідних одиночних та кратних константних несправностей на повному тесті.

Метод синтезу дедуктивної матриці. Вихідні дані: логічний вектор функціональності Q = 10000001 (рис. 11). Синтез дедуктивної матриці використовує таку формулу: $D = L_H = (Q \oplus Q_x)_H$. Логічний вектор Q спочатку записується в матрицю по горизонталі та вертикалі, створюючи метрику матриці. Заповнюються рядки матриці $L_i = Q \oplus Q_i$ шляхом виконання правила: якщо вертикальна координата вектора $Q_i = 1$, вектор Q записується в i-рядок з інверсією своїх біт. В іншому випадку $Q_i = 0$ без інверсії. Потім виконується перекодування бітів матриці L за допомогою матриці H з метою отримання дедуктивної матриці $D = L_H$.

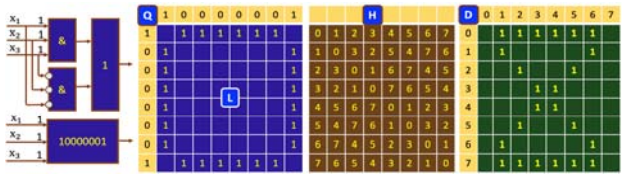

Рисунок 11 – Синтез дедуктивної матриці елемента за логічним вектором

Метод моделювання несправностей за дедуктивною матрицею. Вихідні дані, однакові в метриці 2^n дедуктивного вектора: матриця дедуктивних векторів D та таблиця істинності X одиночних та кратних вхідних константних несправностей (рис. 12). Вхідні дані – двійковий тестовий набір x, який визначає номер-адресу дедуктивного вектора D_x для моделювання несправностей, а також знаки несправностей, що перевіряються в таблиці істинності X.

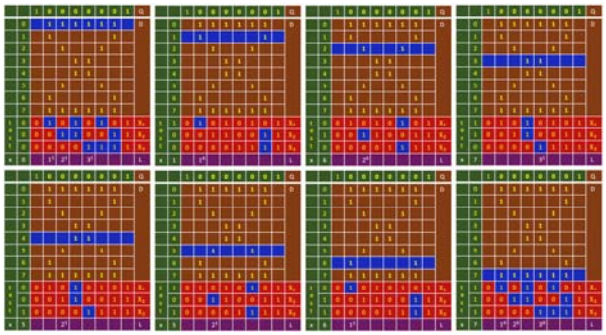

Рисунок 12 – Векторно-табличне дедуктивне моделювання несправностей логіки

Тут представлена суперпозиція однакових за розмірами матриці дедуктивних векторів і таблиці істинності всіх комбінацій константних несправностей на входах логічного елемента, заданого вектора Q = 10000001. Стовпці таблиці істинності x одиничними координатами формують вичерпні комбінації одиночних та кратних константних дефектів, які є похідними від двійкового вхідного тестового набору x. Символ $x_{ij} = 1$ означає перевірку одиночної несправності $j^{\overline{Xi}}$ на вхідному сигналі X_i. Кратна несправність ідентифікована декількома 1-координатами в стовпці матриці x. Наприклад, у синіх клітинах одиницями зазначені несправності, які перевіряються на тестових наборах $x = 101\,(2^1), 011\,(1^1), 111\,(1^0, 2^0, 3^0)$.

Алгоритм формування координат вектора дефектів, що перевіряються, містить всього один оператор логічного множення x-матриці на вектор D_x $L = X \wedge D_x$. При цьому дедуктивний вектор D_x (x = 0,1,2,3,4,5,6,7) розглядається як вектор стану виходів таблиці істинності несправностей, що перевіряються на тестових двійкових наборах x = (000, 001, 010, 011, 100, 101, 011, 111). Результат множення своїми одиничними координатами формує сукупність несправностей, що перевіряються. Вони ідентифікуються в останніх рядках таблиць за загальноприйнятим позначенням.

Визначення знаків несправностей у матриці X заснована на суперпозиції двійкового тестового x-вектора зі стовпцями X-матриці: $X_j = \overline{x} \wedge X_j$. Знак несправності, що перевіряється, на активних 1-координат X-матриці, які виділені синім кольором (рис. 13), визначається за правилами: $X_{ij} = \overline{x_i} \wedge X_{ij}$. Наприклад: $\overline{0} \wedge 1 = 1, \overline{1} \wedge 1 = 0$. Процес визначення знаків несправностей представлений першим та другим стовпцями. Аналіз якості тесту використовує метрику структури даних, представлених таблицями несправностей, що перевіряються, для їх об'єднання (суперпозиції) в процесі моделювання в одну компактну таблицю істинності. Підсумкова таблиця несправностей, визначена у трьохзначному алфавіті {0, 1, X} за допомогою координатної операції об'єднання $0 \cup 1 = X$. Позначення координат таблиці несправностей: 0 – stack-at-0, 1 – stack-at-1, X={0,1} – перевірка несправностей обох знаків вхідної лінії. Процес об'єднання таблиць представлений у правій частині.

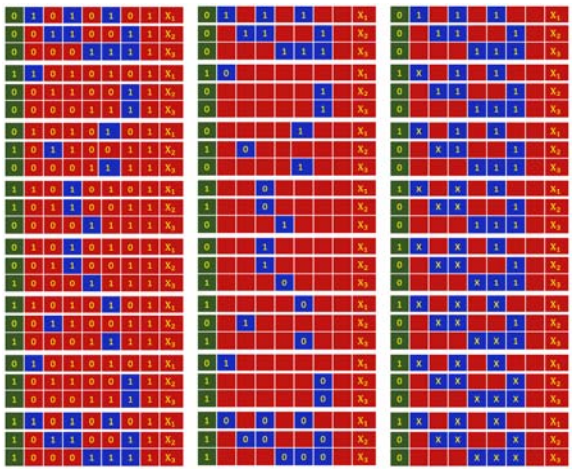

Рисунок 13 – Інтеграція таблиць перевірки несправностей

DOI 10.15588/1607-3274-2023-2-5

4 ЕКСПЕРИМЕНТИ

1. Верифікацію запропонованого векторного методу синтезу дедуктивної матриці для моделювання несправностей виконано на прикладі синтезу дедуктивної матриці для трьох-входового логічного елемента, заданого вектором 10000001 (рис. 14).

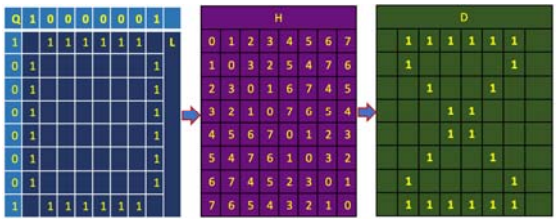

Рисунок 14 – Синтез матриці дедуктивних векторів

$$D = L_H$$

Такий елемент слід розглядати як чорний ящик або RTL-рівень представлення функцій стосовно його структури, яка може бути по-різному виконана при векторному завданні його поведінки. Тут інтерес є результатом транспортування списків активностей від входу до виходу цього елемента. При цьому нецікаво, які шляхи задіяні всередині конкретної реалізації логічного елемента. Проте синтез дедуктивних векторів цього елемента показав, що у всіх вхідних впливах вектори транспортування активностей мають одне й теж значення на парах наборів: 0–15 і 5–6. Інші набори, володіючи симетрією, не повторюються у матриці дедуктивних векторів. Нульові координати матриць L та D представлені порожніми клітинами з метою візуального сприйняття інформації. Таким чином, D-матриця дедуктивних векторів здобувається на основі виконання наступного оператора $D = (Q \oplus Q_i)_H$, отриманого внаслідок суперпозиції операторів: $L = Q \oplus Q_i$ та $D = L_H$.

2. Розглядається процес синтезу дедуктивних формул для чотирьох-входової логічної схеми, вентильна структура якої відома під ім'ям «схема Schneider» [13], яка представлена на рис. 15.

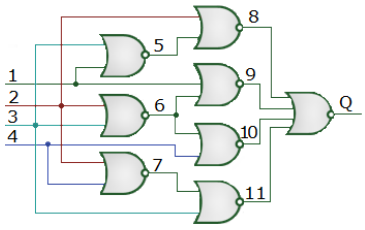

Рисунок 15 – Вентильна реалізація схеми Шнейдера

RTL-модель схеми задана векторним покриттям Q = 1000000000000001 (рис. 16).

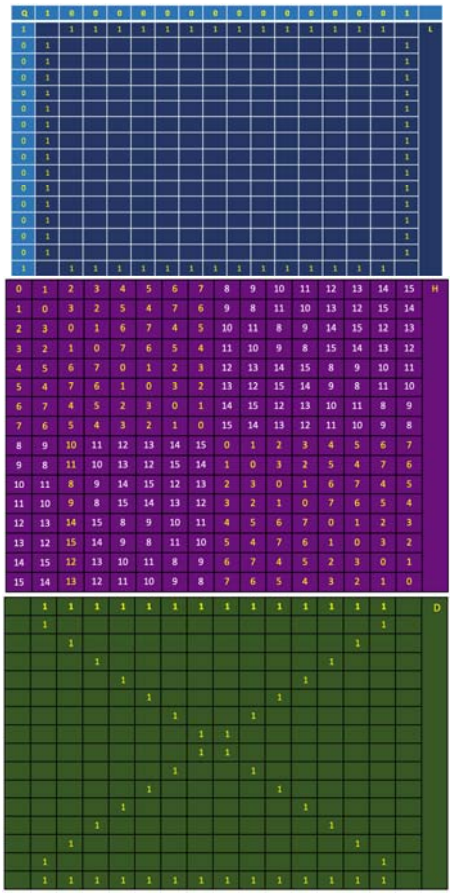

Рисунок 16 – Синтез дедуктивних векторів для чотирьох-входової схеми Q = 1000000000000001

Природно, що для кожного дедуктивного вектора можна отримати аналітичну форму у вигляді ДНФ для транспортування несправностей на конкретному наборі. Але цей шлях є технологічно складним і обчислювально-витратним, тому він не прийнятний для ринку електронних технологій.

Приклад моделювання вхідних несправностей чотирьох-входового елемента на дедуктивному векторі 0000001100000000 (7-й рядок наведеної вище матриці дедуктивних векторів) представлений на рис. 17. За чотири автоматні такти вектори вхідних несправностей були транспортовані на вихід з отриманням результату у вигляді одного вектора вихідних несправностей 0110. Розв'язання даного завдання на вентильній структурі цієї схеми потребувало б тридцять два автоматних такти.

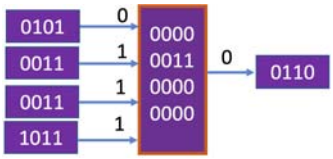

Рисунок 17 – Фрагмент моделювання чотирьох-входової схеми Q = 1000000000000001

DOI 10.15588/1607-3274-2023-2-5

Збільшення кількості входів функціонального RTL-елемента призводить до підвищення продуктивності моделювання за рахунок того, що більше векторів вхідних несправностей моделюються паралельно для отримання вихідного вектора дефектів. Використання чотирьох-входового елемента підвищує продуктивність моделювання в 11 разів, порівняно з його структурним вентильним еквівалентом. У загальному випадку, підвищення продуктивності моделювання для RTL-схем, що мають n входів, порівняно з аналізом ДНФ-структури двох-входових вентилів, визначається такою формулою: $Q = \frac{n}{2} + n - 1$, n = 4, 8, 16 …

3. Розглянемо автомат векторно-дедуктивного моделювання несправностей логіки.

Мета – створення швидкодіючого ефективного вбудованого механізму для векторно-дедуктивного моделювання несправностей логічних елементів цифрових схем (рис. 18).

Рисунок 18 – Секвенсор векторно-дедуктивного моделювання несправностей логіки

Структура даних: H – універсальна матриця перекодування бітів для отримання дедуктивного вектора за вхідними параметрами: $H(x,2^n)$, x – десятковий еквівалент вхідного двійкового слова, індекс, що визначає рядок таблиці перекодування, 2^n – довжина цього рядка в таблиці. H-матриця будується один раз для всіх n-вхідних елементів. $X = (X_1 X_2 X_3)$ – матриця векторів вхідних несправностей для набору x, U – універсум ідентифікаторів одиночних несправностей ліній схеми. $L = Q \oplus Q_x$ – інверсія логічного вектора у разі одиничної реакції елемента $Q_x = 1$ на вхідне слово. $D = (Q \oplus Q_x)_{H\,i} = H$ – матриця отримання дедуктивного вектора для моделювання несправностей як адрес на основі логічного вектора Q.

Далі наводиться приклад векторно-дедуктивного моделювання вхідних двійкових наборів 101 та 111 для трьох-входової схеми Шнейдера (рис. 15).

Алгоритм моделювання несправностей як адрес на дедуктивному векторі має три команди (рис. 19), наведені нижче.

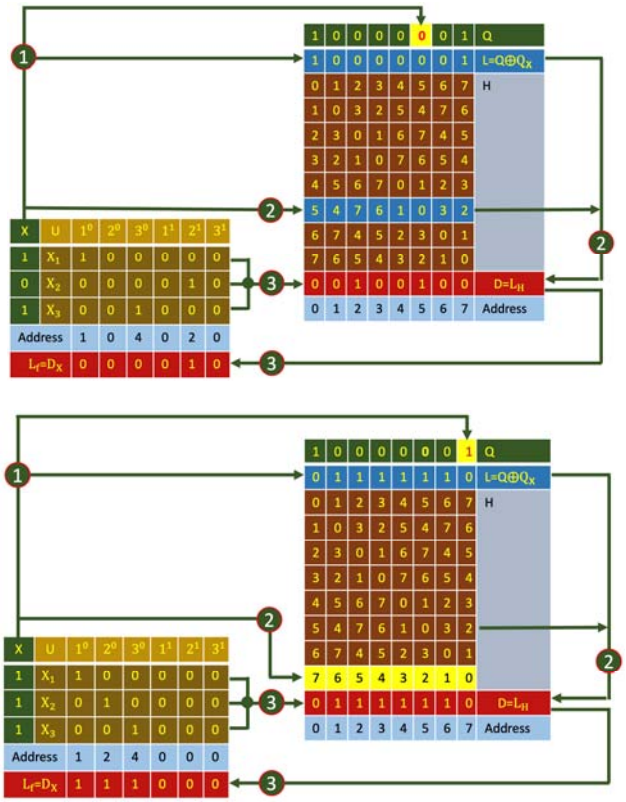

Рисунок 19 – In-memory автомат векторно-дедуктивного моделювання несправностей логіки

1. Генерація активного вектора $L = (Q \oplus Q_x) = 10000001 \oplus 0 = 10000001$. Вектор $Q = 10000001 \oplus 1 = 01111110$ інвертується (права схема), якщо вхідне слово $x(111)$ адресує комірку $Q_x = 111 = 1$.

2. Генерація дедуктивного вектора на основі координатної переадресації бітів активного вектора $D = L_{Hx} = (10000001)_{54761032} = 00100100$.

3. Моделювання несправностей як адрес:

$$L_f = D_x = (00100100)_{010=1} = 1, (00100100)_{100=4} = 0,$$
$$(00100100)_{101=5} = 1, ...$$

Вхідна інформація представлена двома векторами: вхідним $x = 101(5)$, довжиною n (3): логічним $Q = 10000001$, розмірністю 2^n (8). Для правої схеми: $x = 111(7)$, решта параметрів однакові з першою схемою.

Структура даних представлені двома блоками або матрицями. Перший блок-матриця формує вектори несправностей як матриці $F = X \times U$ для кожного вхідного двійкового набору у форматі $n \times 2n$, де $2n$ – кількість одиночних константних несправностей на n-входах. Координата матриці несправностей визначається за такою формулою: $F_{ij} = 1 \leftarrow U_j^{z=\overline{x_i}}$. Напри-

DOI 10.15588/1607-3274-2023-2-5

клад, 1^0 – 0-константна несправність на лінії 1, а 2^1 – 1-константна несправність на лінії 2. Інакше $F_{ij} = 0$. Іншими словами, вхідні несправності, що підлягають моделюванню, є інверсними до справного стану входів.

Стовпці *F*-матриці несправності формують десяткові адреси клітин дедуктивного вектора за правилом: $X_1 \times 2^0 + X_1 \times 2^1 + X_1 \times 2^2 + ...$. Результат моделювання несправностей на вхідному тестовому наборі *x* формується у (червоному) векторі $L_f = D_x$, де одиничними координатами відзначені стовпці, які відповідають несправностям, що перевіряються. *X* – стовпці матриці несправностей, що розглядаються як адреси комірок дедуктивного *Q*-вектора для формування L_f -векторів несправностей, що перевіряються на основі read-write транзакцій. Другий блок призначений для генерації дедуктивного *D*-вектора за вихідними даними: логічний вектор *Q*, двійкове вхідне слово *x*, матриця перекодування бітів *H*. Для синтезу дедуктивного вектора використовується формула: $D = L_H = (Q \oplus Q_x)_H$. Блакитні рядки адрес (Address) використовуються для візуалізації процесу моделювання. Вони не потрібні для написання програмного коду.

5 РЕЗУЛЬТАТИ

У роботі запропоновано векторний метод синтезу дедуктивних матриць для транспортування вхідних векторів несправностей на вихід елемента.

Обчислювальна складність $C = \frac{1}{2} \times k \times n^2$, де k – час зчитування з пам'яті (< 10 ns), *n* – кількість ліній у схемі (<1000), $\frac{1}{2}$ – половина таблиці моделювання одного тест-набору. Оброблено 50 комбінаційних схем різної розмірності. Швидкодію методу підтверджено експериментально: схема на 10 входів обробляється на повному тесті за 10 мкс, 20 входів – 18 мкс. Використовувався 4-х ядерний процесор Intel Core i5, 3.8 GHz. Результати експериментів узагальнено на рис. 20.

Circuit	c17	c53	C 432	c865	c953	c1023	c4532	c4378
Test length, sets	32	64	100	255	784	1024	1056	1056
Modeling, ns	22	32	57	59	63	71	88	87
Simulation, ns	21	64	531	855	867	875	1033	1091

Рисунок 20 – Обчислювальна складність моделювання у порівнянні з аналогами

Розроблено структури даних для паралельного моделювання несправностей цифрових схем на основі примітивної read-write транзакції в матричній пам'яті, де поєднання несправностей є стовпцями-адресами.

Запропоновано секвенсор із п'яти блоків, що складають векторно-логічний комп'ютинг, пов'язаний з дедуктивним моделюванням несправностей на основі read-write транзакцій.

6 ОБГОВОРЕННЯ

Незважаючи на технологічну простоту запропонованого методу синтезу матриці дедуктивних векторів, даний підхід має очевидний недолік, пов'язаний з розмірністю таблиць при великій кількості вхідних змінних. Його можна усунути, якщо на вхідному тестовому наборі оперативно генерувати лише один дедуктивний вектор для моделювання несправностей. Для цього потрібно використовувати лише оператор $D_i = (Q \oplus Q_i)_{Hij}$, описаний раніше. Обчислювальна складність цієї процедури дорівнює $C = 2^n$, де *n* – число змінних логічного елемента, яка визначається перестановками бітів в *Q*-векторі по *H*-матриці для отримання *D*-вектора. У цьому випадку дедуктивне моделювання несправностей не відрізнятиметься скоріш за все від справного моделювання цифрової схеми. Дельта час обробки одного логічного елемента $\Delta T = tD + tF = 2^{n+1}$.

Можна дійти висновку, що дедуктивна матриця довільної логіки має таку метрику на вирішення завдань технічної діагностики: 1) матриця дедуктивних векторів-рядків вирішує питання точного моделювання дефектів шляхом транспортування вхідних векторів несправностей на вихід елемента; 2) матриця є впорядкованою сукупністю векторних похідних по вхідним змінним, які визначаються стовпцями матриці; 3) сукупність $X_i^{'}$ – похідних по вхідним змінним у матриці є тестом для перевірки одиночних константних несправностей вхідних і вихідних ліній; 4) одиничні координати матриці дедуктивних векторів є умови активізації вектора вхідних даних як адрес для транспортування на вихід схеми; 5) матриця може бути активною моделлю для паралельного пошуку несправностей у безтестовому режимі online діагностування.

Переваги запропонованої технології моделювання несправностей можна описати так (рис. 21): 1) відсутність обмежень на розмір та складність логічних елементів, що підлягають моделюванню; 2) відсутність традиційно складного алгоритму моделювання, тільки операція and-суперпозиції таблиці істинності та дедуктивного вектора; 3) паралельно і одночасно моделюються одиночні та всі комбінації кратних константних несправностей, представлених у вигляді таблиці істинності; 4) відсутня процедура генерації списків вхідних несправностей, оскільки таблиця істинності для будь-якої функціональності на *n*-входів та ж сама; 5) метод використовує лише одну паралельну логічну and-операцію і має лінійну обчислювальну складність виконання векторних операцій на множині несправностей; 6) для реалізації методу не потрібні потужні процесори та блоки АЛП, достатньо лише read-write транзакції на пам'яті; 7) метод слід розглядати як про-

DOI 10.15588/1607-3274-2023-2-5

цесорне ядро для моделювання несправностей IP-core SoC і може бути корисним для ринку електронних технологій; 8) метод орієнтований на технології сучасного in-memory computing обробки великих даних без блоків АЛП та процесора.

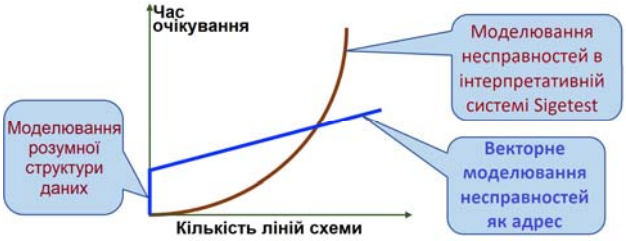

Рисунок 21 – Обчислювальна складність моделювання у порівнянні з аналогами

Переваги автомата векторно-дедуктивного моделювання несправностей полягають у наступному:

1) технологічність синтезу дедуктивного вектора за два автоматні цикли для моделювання несправностей.

2) простота структур даних та алгоритму моделювання несправностей. Чим більше вхідних змінних у логічній схемі, як елементі, тим вищий рівень паралелізму обробки несправностей як адрес. Використовуються тільки read-write транзакції та жодної традиційної логіки CPU.

3) обчислювальна складність алгоритму дедуктивно-векторного моделювання несправностей як адрес: $C = 2^n + 2^n + 2F$, де 2^n – складність виконання координатних операцій для генерації вектора активності $L = Q \oplus Q_x$, 2^n – складність перекодування координат вектора активності з метою отримання дедуктивного вектора $D = L_{Hx}$, $2F$ – число автоматних тактів для моделювання комбінацій вхідних несправностей як адрес.

4) структура даних і алгоритм моделювання несправностей як адрес, орієнтовані на імплементацію як вбудованого BIST SoC, а також для реалізації як програмних за стосунків (software aps) для оцінки якості синтезованого тесту.

ВИСНОВКИ

У роботі розглянуто завдання перенесення архітектури фон Неймана у пам'ять та заміни потужного процесора read-write транзакціями на логічних векторах для зниження енергетичних та часових витрат при моделюванні логічних функціональностей будь-якої розмірності.

В результаті дослідження зроблено крок на шляху створення векторно-логічного in-memory комп'ютингу, що використовує лише read-write транзакції на адресній пам'яті. Використана метрика управління на основі відмов (failure-driven management) $T \oplus F \oplus L = 0$, яка формалізує на макрорівні всі відомі процеси створення комп'ютингу, включаючи проектування та верифікацію (design and test).

Наукова новизна полягає у розробці наступних інноваційних рішень:

1) вперше запропоновано векторно-логічний метод синтезу матриці дедуктивних векторів для паралельного моделювання комбінацій вхідних несправностей як адрес;

2) вперше запропоновано автомат векторно-дедуктивного моделювання несправностей як адрес на основі read-write транзакцій, орієнтований для імплементації в FPGA LUT, вбудованних online симуляторах SoC, як ядро для моделювання несправностей цифрових систем RTL-рівня;

3) демонстрація технологічних переваг векторно-логічного синтезу дедуктивних матриць виконана на численних прикладах традиційної та RTL-логіки, що підкреслює технологічність векторів у порівнянні з аналітичними дедуктивними формулами для побудови симуляторів;

4) матриця дедуктивних векторів, як сукупність вектор-стовпців булевих похідних використовується для побудови мінімальних тестів для логічних елементів;

5) рекурсивна формула синтезу матриці перестановки координат у логічному векторі активності дозволяє суттєво спростити отримання дедуктивної матриці для моделювання несправностей як адрес.

Практична значимість дослідження полягає у тому, що in-memory симулятор дозволить отримати швидкодію моделювання несправностей реальних цифрових блоків SoC лише на рівні сотень наносекунд [23].

Перспективи дослідження пов'язані з імплементацією даної технології моделювання цифрових пристроїв на ринку EDA, оскільки аналоги таких простих алгоритмів обробки великих проектів відсутні навіть у великих компаніях (Synopsis, Cadence, Mentor Graphics, Aldec).

ПОДЯКИ

Автори вдячні колегам за їх активну участь в обговоренні та підтримці наукової школи «Проектування та технічна діагностика цифрових систем на кристалах, комп'ютерах та мережах», розуміння важливого значення розвитку фундаментальних та прикладних досліджень теорії in-memory computing.

ЛІТЕРАТУРА

1. Vector-deductive memory-based transactions for fault-as-address simulation / [W. Gharibi, A. Hahanova, V. Hahanov et al.] // Electronic Modeling. – 2023. – V. 45, №1. – P. 3–26. DOI: 10.15407/emodel.45.01.003.
2. Shannon Claude E. Von Neumann's contributions to automata theory / Claude E. Shannon // Bulletin American Mathematical Society. – 1958. – V.64, №3. – P. 123–129. DOI: 10.1090/S0002-9904-1958-10214-1.
3. Davis M. Emil Post's contributions to computer science / M. Davis // Fourth annual symposium on logic in computer

DOI 10.15588/1607-3274-2023-2-5

science, Pacific Grove, CA, USA, 5–8 June 1989. – P. 134–136. DOI: 10.1109/LICS.1989.39167.
4. What's new in the 2022 Gartner hype cycle for emerging technologies [Electronic resource]. – Access mode: https://www.gartner.com/en/articles/what-s-new-in-the-2022-gartner-hype-cycle-for-emerging-technologies. – Access data: 02.08.2023.
5. RC-NVM: Enabling Symmetric Row and Column Memory Accesses for In-memory Databases / [P. Wang et al.] // IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, 24–28 February 2018. – P. 518–530. DOI: 10.1109/HPCA.2018.00051.
6. AI Accelerator Embedded Computational Storage for Large-Scale DNN Models / [B. Ahn, J. Jang, H. Na, et al.] // IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Incheon, Korea, Republic, 13–15 June 2022. – P. 483–486. DOI: 10.1109/AICAS54282.2022. 9869991.
7. Reliable ReRAM-based logic operations for computing in memory / [M. Moreau et al.] // IFIP/IEEE International Conference on Very Large-Scale Integration (VLSI-SoC), Verona, Italy, 8–10 October 2018. – P. 192–195. DOI: 10.1109/VLSI-SoC.2018.8644780.
8. Kang W. Spintronic memories: from memory to computing-in-memory / W. Kang, H. Zhang, W. Zhao // IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), Qingdao, China, 17–19 July 2019. – P. 1–2. DOI:10.1109/NANOARCH47378.2019.181298.
9. Memory sizing of a scalable SRAM in-memory computing tile based architecture / [R. Gauchi et al.] // IFIP/IEEE 27th International Conference on Very Large-Scale Integration (VLSI-SoC), Cuzco, Peru, 6–9 October 2019. – P. 166–171. DOI: 10.1109/VLSI-SoC.2019.8920373.
10. Pomeranz I. Forward-looking fault simulation for improved static compaction / I. Pomeranz, S. M. Reddy // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2001. – Vol. 20, No. 10. – P. 1262–1265. DOI: 10.1109/43.952743.
11. Deductive qubit fault simulation / [V. Hahanov, S. Chumachenko, I. Iemelianov et al.] // 14th International Conference: The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, Ukraine, 21–25 February 2017. – P. 256–259. DOI: 10.1109/CADSM.2017.7916129.
12. Qubit-driven Fault Simulation / [V. Hahanov, W. Gharibi, E. Litvinova et al.] // IEEE Latin American Test Symposium (LATS), Santiago, Chile, 11–13 March 2019. – P. 1–7. DOI: 10.1109/LATW.2019.8704583.
13. Qubit Test Synthesis Processor for SoC Logic / [W. Gharibi, D. Devadze, V. Hahanov et al.] // IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 13–16 September 2019. – P. 1–5. DOI: 10.1109/EWDTS.2019.8884476.
14. Vector-qubit models for SOC logic-structure testing and fault simulation / [V. Hahanov et al.] // 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Lviv, Ukraine, 22–26 February 2021. – P. 24–28. DOI: 10.1109/CADSM52681.2021.9385266.
15. Quantum models and method for analysis and testing computing systems / [V. I. Hahanov, S. M. Hyduke, W. Gharibi et al.] // 11th International Conference on Information Technology: New Generations, Las Vegas, 7–9 April 2014. – P. 430–434. DOI: 10.1109/ITNG.2014.125.
16. Qubit fault detection in SoC logic / [M. Karavay, V. Hahanov, E. Litvinova et al.] // IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 13–16 September 2019. – P. 1–7. DOI: 10.1109/EWDTS.2019.8884475.
17. Qubit-driven fault simulation / [V. Hahanov, W. Gharibi, E. Litvinova et al.] // IEEE Latin American Test Symposium (LATS), Santiago, Chile, 11–13 March 2019. – P. 1–7. DOI: 10.1109/LATW.2019.8704583.
18. Qubit fault detection in SOC logic / [M. Karavay, V. Hahanov, E. Litvinova et al.] // IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 13–16 September 2019. – P. 1–7. DOI: 10.1109/EWDTS.2019.8884475.
19. Hahanov V. Cyber physical computing for iot-driven services / V. Hahanov. – New York : Springer, 2018. – 279 p. DOI: 10.1007/978-3-319-54825-8
20. Fast RTL fault simulation using decision diagrams and bitwise set operations / [U. Reinsalu, J. Raik, R. Ubar et al.] // IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, Vancouver, Canada, 3–5 October 2011. – P. 164–170. DOI: 10.1109/DFT.2011.42.
21. Pomeranz I. A synthesis procedure for flexible logic functions / I. Pomeranz, S. M. Reddy // Design, Automation and Test in Europe, Paris, France, 23–26 Februry 1998 . – P. 973–974. DOI: 10.1109/DATE.1998.655995.
22. Armstrong D.B. A deductive method for simulating faults in logic circuits / D.B. Armstrong // IEEE Transactions on Computers. – 1972. – Vol. C-21, No. 5. – P. 464–471. DOI: 10.1109/T-C.1972.223542.
23. Performance Evaluation of LUTs in FPGA in Different Circuit Topologies / [N. Vinod et al.] // International Conference on Communication and Signal Processing (ICCSP), 28–30 July 2020. – P. 1511–1515. DOI: 10.1109/ICCSP48568.2020.9182074.

Стаття надійшла до редакції 27.04.2023.
Після доробки 19.05.2023.

UDC 681.326

VECTOR-LOGICAL FAULT SIMULATION

Hahanov V. – Doctor of science, Professor of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

Chumachenko S. – Doctor of science, Professor Head of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

Litvinova Y. – Doctor of science, Professor of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

Hahanova I. – Doctor of science, Professor of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

DOI 10.15588/1607-3274-2023-2-5

Khakhanova A. – PhD, Associated Professor of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

Shkil A. − PhD, Associated Professor of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

Rakhlis D. − PhD, Associated professor of Design Automation Department, Kharkiv National University of Radio Electronics, Ukraine.

Hahanov I. – Post-graduate student of Kharkiv National University of Radio Electronics, Ukraine.

Shevchenko O. – PhD, Assistant of Kharkiv National University of Radio Electronics, Ukraine.

ABSTRACT

Context. The main idea is the creation of vector-logical in-memory computing (VLC), which uses only read-write transactions on the address memory for faults-as-addresses simulation. There is no traditional logic. VLC is free from processor commands and ALU for computing organization and is therefore focused on implementation in SoC and FPGA. A vector-logical method of deductive matrix synthesis for the transportation of input faults, which has a quadratic computational complexity, is proposed. An in-memory simulator-automata for vector-deductive faults-as-addresses simulation, which based on read-write transactions for implementation in SoC is proposed.

Objective. Development of a vector deductive method of fault simulation based on primitive read-write transactions for the analysis of logic circuits.

Method. An input test set and a logical functionality vector are used. The proposed method is a development of the deductive vectors' synthesis algorithm based on the truth table. The deductive matrix is intended for the synthesis and verification of tests using parallel simulation of faults-as-addresses combinations, based on read-write transactions over bits of deductive vectors in memory.

Results. A vector method of the deductive matrices synthesis for the transportation of input faults vectors to the output of the element, was proposed. Data structures have been developed for parallel faults simulation of digital circuits based on a primitive read-write transaction in matrix memory, where combinations of faults serve as address-columns. A sequencer of five blocks, that constitute a vector-logic computing, connected with deductive faults simulation based on read-write transactions, is proposed. Verification of models and methods on test examples has been performed.

Conclusions. The scientific novelty consists in the development of the following innovative solutions: 1) a vector-logic method of synthesis of the deductive vectors matrix for parallel simulation of combinations of input faults-as-addresses, is proposed for the first time; 2) an automata for vector-deductive faults-as-addresses simulation, on the basis of read-write transactions, which is oriented for implementation in FPGA LUT, embedded online simulator SoC, as a core for faults simulation of RTL-level digital systems, was proposed for the first time; 3) the demonstration of the technological advantages of the vector-logic synthesis of deductive matrices is performed on numerous examples of traditional and RTL-logic, which accentuate the manufacturability of vectors in comparison with analytical deductive formulas during simulators construction; 4) a matrix of deductive vectors, as a set of vector-columns of Boolean derivatives is used to construct minimal tests for logical elements; 5) the recursive formula for the synthesis of the permutation of coordinates matrix in the logical activity vector makes it possible to significantly simplify the obtaining of the deductive matrix for faults-as-addresses simulation. The practical significance lies in the fact that the in-memory simulator will allow to obtain the speed of faults simulation of real digital blocks for SoC at the level of hundreds of nanoseconds. Complexity estimates of the corresponding algorithms are given.

KEYWORDS: vector computing, vector form of logic, matrix of deductive vectors, vector method of deductive matrix synthesis, read-write transaction, vector model of faults, vector-logical deductive faults simulation.

REFERENCES

1. Gharibi W., Hahanova A., Hahanov V. et al. Vector-deductive memory-based transactions for fault-as-address simulation, *Electronic Modeling,* 2023, V. 45, №1, pp. 3–26. DOI: 10.15407/emodel.45.01.003.
2. Shannon Claude E. Von Neumann's contributions to automata theory, *Bulletin American Mathematical Society*, 1958, V.64, №3, pp. 123–129. DOI: 10.1090/S0002-9904-1958-10214-1.
3. Davis M. Emil Post's contributions to computer science, *Fourth annual symposium on logic in computer science.* Pacific Grove, CA, USA, 5–8 June 1989, pp. 134–136. DOI: 10.1109/LICS.1989.39167.
4. What's new in the 2022 Gartner hype cycle for emerging technologies [Electronic resource]. Access mode: h-ttps://www.gartner.com/en/articles/what-s-new-in-the-2022-gartner-hype-cycle-for-emerging-technologies. Access data: 02.08.2023.
5. Wang P. et al. RC-NVM: Enabling Symmetric Row and Column Memory Accesses for In-memory Databases, *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Vienna, Austria, 24–28 February 2018, pp. 518–530. DOI: 10.1109/HPCA.2018. 00051.
6. Ahn B., Jang J., Na H. et al.] AI Accelerator Embedded Computational Storage for Large-Scale DNN Models, *IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. Incheon, Korea, Republic, 13–15 June 2022, pp. 483–486. DOI: 10.1109/AICAS54282.2022. 9869991.
7. Moreau M. et al. Reliable ReRAM-based logic operations for computing in memory, *IFIP/IEEE International Conference on Very Large-Scale Integration (VLSI-SoC).* Verona, Italy, 8–10 October 2018, pp. 192–195. DOI: 10.1109/VLSI-SoC.2018.8644780.
8. Kang W., Zhang H., Zhao W. Spintronic memories: from memory to computing-in-memory, *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. Qingdao, China, 17–19 July 2019, pp. 1–2. DOI:10.1109/NANOARCH47378. 2019.181298.
9. Gauchi R. et al. Memory sizing of a scalable SRAM in-memory computing tile based architecture, *IFIP/IEEE 27th International Conference on Very Large-Scale Integration*

DOI 10.15588/1607-3274-2023-2-5

(VLSI-SoC). Cuzco, Peru, 6-9 October 2019, pp. 166–171. DOI: 10.1109/VLSI-SoC.2019.8920373.

10. Pomeranz I., Reddy S. M. Forward-looking fault simulation for improved static compaction, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 2001, Vol. 20, No. 10, pp. 1262–1265. DOI: 10.1109/43.952743.
11. Hahanov V., Chumachenko S., Iemelianov I. et al. Deductive qubit fault simulation, *14th International Conference: The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*. Lviv, Ukraine, 21–25 February 2017, pp. 256–259. DOI: 10.1109/CADSM.2017.7916129.
12. Hahanov V., Gharibi W., Litvinova E. et al. Qubit-driven Fault Simulation, *IEEE Latin American Test Symposium (LATS)*. Santiago, Chile, 11–13 March 2019, pp. 1–7. DOI: 10.1109/LATW.2019.8704583.
13. Gharibi W., Devadze D., Hahanov V. et al. Qubit Test Synthesis Processor for SoC Logic, *IEEE East-West Design & Test Symposium (EWDTS).*, Batumi, Georgia, 13–16 September 2019, pp. 1–5. DOI: 10.1109/EWDTS.2019. 8884476.
14. Hahanov V. et al. Vector-qubit models for SOC logic-structure testing and fault simulation, *16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*. Lviv, Ukraine, 22–26 February 2021, pp. 24–28. DOI: 10.1109/CADSM52681.2021. 9385266.
15. Hahanov V. I., Hyduke S. M., Gharibi W. et al. Quantum models and method for analysis and testing computing systems, *11th International Conference on Information Technology: New Generations*. Las Vegas, 7–9 April 2014, pp. 430–434. DOI: 10.1109/ITNG.2014.125.
16. Karavay M., Hahanov V. , Litvinova E. et al. Qubit fault detection in SoC logic, *IEEE East-West Design & Test Symposium (EWDTS)*. Batumi, Georgia, 13–16 September 2019, pp. 1–7. DOI: 10.1109/EWDTS.2019. 8884475.
17. Hahanov V., Gharibi W., Litvinova E. et al. Qubit-driven fault simulation, *IEEE Latin American Test Symposium (LATS)*. Santiago, Chile, 11–13 March 2019, pp. 1–7. DOI: 10.1109/LATW.2019.8704583.
18. Karavay M., Hahanov V. , Litvinova E. et al. Qubit fault detection in SOC logic, *IEEE East-West Design & Test Symposium (EWDTS)*. Batumi, Georgia, 13–16 September 2019, pp. 1–7. DOI: 10.1109/EWDTS.2019. 8884475.
19. Hahanov V. Cyber physical computing for iot-driven services. New York, Springer, 2018, 279 p. DOI: 10.1007/978-3-319-54825-8
20. Reinsalu U., Raik J., Ubar R. et al. Fast RTL fault simulation using decision diagrams and bitwise set operations, *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*. Vancouver, Canada, 3–5 October 2011, pp. 164–170. DOI: 10.1109/DFT.2011.42.
21. Pomeranz I, Reddy S. M. A synthesis procedure for flexible logic functions, *Design, Automation and Test in Europe*. Paris, France, 23–26 Februry 1998, pp. 973–974. DOI: 10.1109/DATE.1998. 655995.
22. Armstrong D. B. A deductive method for simulating faults in logic circuits, *IEEE Transactions on Computers*, 1972, Vol. C-21, No. 5, pp. 464–471. DOI: 10.1109/T-C.1972.223542.
23. Vinod N. et al. Performance Evaluation of LUTs in FPGA in Different Circuit Topologies, *International Conference on Communication and Signal Processing (ICCSP)*, 28–30 July 2020, pp. 1511–1515. DOI: 10.1109/ICCSP48568.2020.9182074.

DOI 10.15588/1607-3274-2023-2-5