Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління___________________
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_________
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____________________
(рівень вищої освіти)

Структури даних для дедуктивного моделювання умовних операторів HDL
(тема)

Виконав: студент 2 курсу, групи СКСм-22-1

________Тріфанов_ О.В.________________
(прізвище, ініціали)

Спеціальність *123 _ Комп'ютерна інженерія*

Тип програми ____*освітньо-професійна*_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_________________________

*Спеціалізовані комп'ютерні системи*_______
(повна назва освітньої прграми)

Керівник роботи доц. Шкіль О.С.__
(посада, прізвище, ініціали)

Допускається до захисту

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

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123_ Комп’ютерна інженерія

(шифр і назва)

Тип програми Освітньо-професійна

(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп’ютерні системи

(повна назва

ЗАТВЕРДЖУЮ:

Зав. кафедри ____________

(підпис)

« 07 » 11 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Тріфанову Олегу Віталійовичу

(прізвище, ім’я, по батькові)

1. Тема роботи (проекту)_ Структури даних для дедуктивного моделювання умовних операторів HDL

затверджена наказом по університету від « 03 » 11 2023р. № 1282 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10.01.2024

3. Вихідні дані до роботи (проекту)

Кубітні моделі цифрових схем

Мова опису апаратури VHDL

САПР XILINX ISE

Програма моделювання несправностей DCP

4. Перелік питань, що потрібно опрацювати у роботі

Моделі цифрових пристроїв на мовах опису апарватури

Дедуктивне моделювання несправностей

Кубічне моделювання несправностей

Моделювання несправностей по дедуктивним формулам

Метод моделювання несправностей по дедуктивним векторам

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (слайдів) 17 слайдів

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання 02.09.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми	02.09.2023-08.09.2023	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів	09.09. 2023-15.09. 2023	
3	Аналіз стилів опису цифрових пристроїв мовами опису апаратури	16.09.2023-29.09.2023	
4	Аналіз методів дедуктивного моделювання несправностей	30.09. 2023-13.10.2023	
5	Розробка методів взяття булевих похідних для кубічного та аналітичного стилів опису	14.10. 2023-31.10. 2023	
6	Проектування кубітних описів цифрових пристроїі, які реалізують умовні конструкції	01.11. 2023-30.11. 2023	
7	Розробка методів моделювання несправностей по дедуктивним векторам	01.12. 2023-20.12. 2023	
8	Оформлення пояснювальної записки	21.12. 2023-10.01. 2024	
9	Захист проекту	11.01. 2023-25.01. 2024	

Студент ____________________
(підпис)

Керівник роботи (проекту) ________________ доц. Шкіль О.С..
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 66 сторінок, 37 рисунків, та 13 джерел за переліком посилань.

МОВА ОПИСУ АПАРАТУРИ, УМОВНІ ОПЕРАТОРИ, ТАБЛИЦІ ІСТИННОСТІ, ДЕДУКТИВНЕ МОДЕЛЮВАННЯ НЕСПРАВНОСТЕЙ

Метою роботи є підвищення швидкості та якості створення діагностичного забезпечення цифрових пристроїв за рахунок створення оптимальних структур даних та процедур дедуктивного моделювання несправностей на основі структурно-функціональних моделей комбінаційних схем.

В ході виконання роботи розглянуті різні стилі опису цифрових пристроїв мовами опису апаратури. Показана еквівалентність паралельних та послідовних умовних операторів а також їх схемна реалізація у вигляді мультиплексорів. Запропонований спосіб отримання таблиць істинності синтезованої схемної структури за допомогою TestBench Xilinx ISE.

Розглянуті різні технології та структури даних дедуктивного моделювання несправностей для табличного, аналітичного та кубітного способів опису цифрових схем. Розглянута програмна реалізація кубічного дедуктивного моделювання несправностей та показна еквівалентність отриманих результатів для схем мультиплексорів MX 2-в-1 та MX4-в-1 з використанням програмного продукту DCP.

ABSTRACT

The explanatory note contains: 66 pages, 37 figures, 13 sources according to the list of links.

HARDWARE DESCRIPTION LANGUAGE, CONDITIONAL OPERATORS, TRUTH TABLES, DEDUCTIVE FAULT SIMULATION

The purpose of the work is to increase the speed and quality of creating diagnostic support for digital devices due to the creation of optimal data structures and procedures for deductive faults simulation based on structural-functional models of combinational circuits.

In the course of the work, various styles of digital device description in hardware description languages were considered. The equivalence of concurrent and sequential conditional operators is shown, as well as their circuit implementation in the form of multiplexers. A method is proposed for obtaining truth tables of the synthesized circuit structure using TestBench Xilinx ISE.

Various technologies and data structures of deductive fault simulation for tables, analytical and qubit description digital circuits are considered. The software implementation of the cubic deductive fault simulation and the demonstrable equivalence of the obtained results for the circuits of the MX 2-in-1 and MX 4-in-1 multiplexers using the DCP software product are considered.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГСА – граф-схеми алгоритмів;

ДФ – дедуктивна формула;

ДНФ – диз'юнктивна нормальна форма;

КП – кубічне покриття;

ОКН – одиночні константні несправності;

ПЕ – примітивний елемент;

ПЗ – програмне забезпечення;

ПЛІС – програмовані логічні інтегральні схеми;

РЕА – радіоелектронна апаратура;

САПР – система автоматизованого проектування;

СФ – скобкова форма;.

ТІ – таблиця істинності;

ТН – таблиця несправностей;

ЦП – цифровий пристрій;

CPLD – Complex Programmable Logic Device (різновид ПЛІС);

FPGA – field-programmable gate array (різновид ПЛІС);

FSM – finite state machine (кінцевий автомат);

HDL – hardware description language (мова опису апаратури);

RTL – register transfer level (рівень регістрових передач);

SoC – System of Chip (система на кристалі);

UUT – Unit Under Test (об'єкт діагностування);

VHDL – very high speed integrated circuits HDL (одна з мов опису апаратури).

ВСТУП

Ускладнення програмного та апаратного забезпечення комп'ютерних систем злічує кількість відмов несправностей, що викликають значущість ні матеріальні та моральні збитки. Навіть адитивне нарощування в комп'ютері програмних продуктів (апаратури) призводить до конфліктних ситуаціям, збоям, відмовам, що пояснюватиметься відсутністю верифікації системної працездатності та сумісності встановлених компонентів Суттєва тенденція обмеження інвестицій на сервісне діагностичне обслуговування комп'ютерних систем на стадіях проектування та експлуатації.

Вона стосується необхідності розробки та супроводження тестопридатного програмно-апаратного забезпечення комп'ютера на функціональному, алгоритмічному та системному рівнях. Для цього потрібно вирішувати завдання проектування тестів, алгоритмів контролю та пошуку дефектів, системної верифікації сумісності компонентів Рішення згаданих задач вимагає подальшого розвитку та удосконалення методів та засобів моделювання справної поведінки та несправностей комп'ютерної структури на різних рівнях абстракції. Що стосується моделювання несправностей, зараз необхідні нові технології визначення якості тестів перевірки та дефектів компонентів комп'ютерної системи. Такі технології мають відповідати вимогам швидкодії, уніфікації, простоти програмної реалізації, Цим визначається актуальність нових технологій моделювання несправностей комп'ютерних систем.

1 МОДЕЛІ ЦИФРОВИХ ПРИСТОЇВ НА МОВАХ ОПИСУ АПАРАТУРИ

1.1 Подвійність моделей мовами опису апаратури

Апаратні системи складніші за «чисті» програмістські, тому що це hard-realtime системи. На кожну функцію є жорстке обмеження часу, за яке вона повинна відпрацювати за будь-яких вхідних даних. На затримки впливає геометрія та взаємне становище елементів – ще один фактор, який у принципі відсутній у розробці програмного забезпечення (ПЗ). Це не надто сильно впливає на системи, що вбудовуються на етапі проектування, але це береться до уваги, тому що може впливати на архітектуру. Кожен зайвий міліметр площі кристала – близько 6 центів для техпроцесу при 0,13мкм. Одна ітерація із прототипом – це мінімум 4 місяці. Перевипуск фотошаблону – це від сотень тисяч доларів до близько мільйона (або вище), якщо йдеться про сучасні тонкі технічні процеси. Мови опису апаратури служать для формального опису дискретних пристроїв обчислювальної техніки і можна використовувати всіх етапах розробки цифрових електронних систем. VHDL може використовуватися на етапах проектування, верифікації, синтезу та тестування апаратури так само, як і для передачі даних про проект, модифікації та супроводу. Спочатку HDL призначався для моделювання (що і пояснює його велику універсальність), але пізніше з нього було виділено синтезоване підмножина. Написання алгоритмічної моделі на підмножині, що синтезується, гарантує автоматичний синтез за цією моделлю алгоритмічної схеми. У HDL було введено поняття модельного часу, сигналу, події, компонента та інші. Використання HDL дозволяє не прив'язувати проект заздалегідь до конкретного фізичного способу реалізації, та ж логічна HDL-реалізація є джерелом генерації різних фізичних реалізацій. У основі мови лежать такі принципи побудови: підтримка функціональної декомпозиції (функціональна ієрархія та рекурсія); підтримка структурної декомпозиції (структурна ієрархія); подання системи у вигляді паралельно функціонуючих взаємодіючих процесів; використання абстрактних типів даних; використання подієвого моделювання; підтримка різних рівнів абстракції та деталізації подання проекту. Тобто головна особливість HDL – паралелізм та наявність сигналів у описі. У програмній моделі всі оператори виконуються один за одним (не враховуючи переходи та виклики функцій). У певний час виконується один оператор. В апаратної моделі на HDL всі призначення сигналів виконуються одночасно в той самий момент часу. Розглянемо простий фрагмент коду: a = b; c=a. Якщо це фрагмент програмної моделі (де оператори виконуються послідовно), то ясно, що 'c' та 'a' будуть містити значення 'b'. Якщо це фрагмент апаратної моделі (де оператори виконуються паралельно), то 'c' міститиме старе значення 'a', та 'a' буде вміщено значення 'b'.

В основі структурно-функціональних підходів, що розробляються, до проектування та верифікації HDL-моделей лежить дуалізм HDL-коду. З одного боку HDL-модель – це опис алгоритмічною мовою з формалізацією мовних конструкцій та наявністю спеціалізованого середовища розробки. З іншого боку, HDL-модель – це опис цифрового пристрою для САПР РЕА. Даний опис за формальними правилами перетворюється (синтезується) на структурно-функціональні схемні моделі різного рівня ієрархії. Стандартизовані (шаблонні) фрагменти HDL-коду перетворюються на стандартні схемні реалізації. Синтезовані моделі строго формально перетворюються на схемні реалізації пристрою (ПЛІС, замовні НВІС, мікропроцесорні структури). Таким чином, HDL-модель - це фактично схема і до неї застосовні методи схемного аналізу, синтезу тестів, побудови алгоритмів пошуку дефектів, які досить добре розроблені для цифрових схем.

Між кодом мовою описи апаратури та відповідної йому цифрової схемою у вибраній технологічній платформі «перебуває» система синтезу, тобто. комплекс програмних засобів, що виконує перетворення кодів, складених на HDL, на деяку схемну реалізацію за визначеними правилами.

Система синтезу однозначно перетворює вихідний VHDL опис у схемну реалізацію рівня регістрових передач незалежно від обраної технології. Рівень реєстрових передач ґрунтується на бібліотечних елементах, які не залежать від обраної реалізації. Правила перетворення операторних конструкцій стандартизовані (прикладом може бути стандарт 1076.6-1999 IEEE Standard for VHDL RTL Synthesis та 1364.1-2002 IEEE Standard for Verilog RTL Synthesis та їх пізніші версії). Можливі установки систем синтезу не впливають на реалізацію вихідного стандарту. Отже, можна припустити, що в рамках стандарту кожен оператор мови перетворюється на єдину можливу схемну реалізацію.

1.2 Стилі описів цифрових пристроїв мовами опису апаратури

В мовах опису апаратури використовується в три стилі опису проектів – потоковий («стиль потік даних»), поведінковий та структурний. Але базовими є потоковий та поведінковий.

В потоковому стилі використовуються паралельні оператори. Паралельні оператори визначають паралельну (у часі) поведінку схем. Паралельні оператори активізуються сигналами, які використовуються зв'язку паралельних операторів. Порядок виконання паралельних операторів пов'язаний з порядком їх появи всередині архітектурного тіла. Всі паралельні оператори виконуються одночасно під час одного і того самого циклу моделювання. Результати паралельних операторів доступні для інших операторів після закінчення поточного циклу моделювання. Всі паралельні оператори можуть бути перетворені в еквівалентні процеси

У потокових проектах використовують оператори паралельного призначення сигналів. Паралельне призначення визначено у трьох різних формах:

– безумовне паралельне призначення;
– умовне паралельне призначення;

– паралельне призначення на вибір.

Синтаксис безумовного паралельного призначення:

signal-name <= expression;

Синтаксис умовне паралельне призначення:

signal-name <= expression when boolean-expression else expression;

Його можна прочитати так: «Сигнал з ім'ям «signal-name» набуває значення виразу «expression» за умови або безумовно. Призначення значення сигналу задають складовим оператором "<=". Оскільки в мові VHDL необхідно суворо дотримуватись типів, тип виразу expression повинен бути сумісний з типом сигналу signal-name.

Наприклад y1 <= '1'; y1 <= '1' when State=a1 else '0';

Паралельне призначення на вибір – це вибіркове (на вибір) призначення сигналу його значення (selected signal-assignment statement).

```
with expression select
signal-name <= signal-value when choices,,
. . .
signal-value when choices;
```

Цей оператор обчислює заданий вираз expression і надає сигналу з ім'ям signal-name значення сигналу (signal-value), що відповідає тій з альтернатив (choices), значення якої дорівнює expression. Альтернативою в кожному реченні може бути одиночне можливе значення expression або список значень, розділених вертикальною рисою (|). Альтернативи choices в даному операторі повинні бути взаємно виключними і разом включати всі можливі випадки. В останньому реченні, коли можна скористатися ключовим словом інших як вказівки на всі значення expression, які ще не були згадані.

Наприклад :

```
with condition select
 y <= xl when "00",  x2 when "01",  x3 when  "10",  x4  when others;
```

Ключовим елементом поведінкового опису (behavioral description) мови VHDL є «процес». Процес (process) - сукупність «послідовних» операторів, які виконуються одночасно з іншими паралельними операторами та іншими процесами. За допомогою процесу можна задати складну взаємодію сигналів і подій таким способом, що при моделюванні ця взаємодія реалізується практично за нульовий час в моделі, а результатом синтезу стає комбінаційна або послідовна схема, яка виконує операцію, що моделюється безпосередньо.

VHDL-процес завжди або виконується (running process), або зупинений (suspended process). Переліком сигналів у визначенні процесу, який називається списком чутливості (sensitivity list), визначаються умови, коли процес виконується. Спочатку процес зупинено; коли змінюється значення будь-якого з сигналів у його списку чутливості, виконання процесу відновлюється, починаючи з першого послідовного оператора, і воно триває, поки не буде досягнутий кінець. Якщо будь-який сигнал зі списку чутливості змінює своє значення в результаті виконання процесу, процес знову виконується. Це триває доти, доки запуск процесу не перестане призводити до зміни значення будь-якого з цих сигналів. При моделюванні все це відбувається за нульовий час у моделі.

В процесі використовуються послідовний сигнальний оператор призначення та оператор присвоювання значення змінній. У мові VHDL є послідовні оператори кількох видів. Перший – це послідовний сигнальний оператор присвоювання (sequential signal-assignment statement); у нього той самий синтаксис, що й у паралельного аналога (signal-name <= expression;), але послідовний оператор перебуває у тілі процесу, а не у тілі архітектури. Аналогічний оператор для змінних оператор надання значення змінній (variable-assignment statement), синтаксис якого має вигляд "variable-name: = expression;". Зауважимо, що для змінних використовується інший оператор присвоювання " := ".

Крім того в процесі використовуються послідовні умовні оператори `if` та `case`.

Найпростіший умовний оператор – це оператор if (if statement),. У першій і найпростішої формі цього оператора перевіряється булеве вираження boolean-expression і, якщо вона має значення true, то виконується послідовний оператор sequential statement. У другій формі додається пропозиція "else" з іншим послідовним оператором sequential-statement, який виконується, якщо булевий вираз має значення false.

Для утворення вкладених операторів if-then-else у мові VHDL використовують спеціальне ключове слово elsif, яке вводить середні пропозиції. Послідовний оператор sequential-statement пропозиції elsif виконується в тому випадку, коли булеве вираз boolean-expression в цій пропозиції істинно, а всі попередні булеви вирази boolean expressions виявляються помилковими. Послідовний оператор sequential-statement заключної необов'язкової пропозиції else виконується лише тоді, коли всі попередні вирази boolean-expressions мають значення false.

Наведемо три форми синтаксису оператора `if`:

```
if boolean-expression  then sequential-statement end if;
if boolean-expression then sequential-statemen
 else sequential-statement
end if;
if boolean-expression then sequential-statement
elsif boolean-expression then sequential-statement
...
eisif boolean-expression then sequential-statement
else sequential-statement
end if;
```

Послідовний оператор багато альтернативного вибору `case` використовується коли потрібно вибирати серед кількох альтернатив на підставі значення лише одного сигналу або виразу, зазвичай більш читабельним і дає кращий результат синтезу

Синтаксис оператора case:

```
case expression is
when choices => sequential-statements
...
when choices => sequential-statements
end case;
```

У цьому операторі обчислюється заданий вираз expression, за його значенням вибирається одна з альтернатив choices і виконуються відповідні послідовні оператори sequential-statements. Зауважте, що в кожному з наборів альтернатив choices можна записати один або більше послідовних операторів. Самі альтернативи choices можуть мати форму одного значення чи кількох значень, розділених вертикальною рисою (|). Альтернативи choices повинні бути взаємно виключаючими та містити всі можливі значення типу виразу expression; в останній альтернативі choices можна скористатися ключовим словом others для вказівки всіх значень, які ще не були згадані раніше.

Паралельні та послідовні умовні оператори еквівалентні та використовуються в залежності від уподобань проектувальника [1].

Листинг 1 це приклад, який демонструє еквівалентність оператора умовного призначення сигналу `(architecture first)` та послідовного умовного оператора в конструкції `process (architecture second)`.

Лістинг 1.1

```
entity example_condition is
port (
      x1, x2, x3, x4 : in bit;
      condition      : in bit_vector (1 downto 0);
      F              : out bit);
end example_condition;

architecture first of example_condition is
begin
     F <= x1 when condition = "00" else
          x2 when condition = "01" else
          x3 when condition = "10" else
          x4;
end first;
```

```
architecture second of example_condition is
begin
 process (x1, x2, x3, x4, condition )
 begin
  if (condition = "00")    then F <= x1;
  elsif (condition = "01") then F <= x2;
  elsif (condition = "10") then F <= x3;
  else F <= x4;
  end if;
 end process;
end second;
```

Листинг 1.2 це приклад, який демонструє еквівалентність паралельного оператора вибіркового призначення сигналу `(architecture first1)` та оператора багатопозиційного вибору `case` в конструкції `process` `(architecture second1)`.

Лістинг 1.2

```
entity example1_condition is
port (
      x1, x2, x3, x4 : in bit;
      condition      : in bit_vector (1 downto 0);
      F              : out bit);
end example1_condition;

architecture first1 of example1_condition is
begin
  with condition select
     F <= xl when "00",
          x2 when "01",
          x3 when "10",
          x4 when others;
end first1;

architecture second1 of example1_condition is
begin
 process (x1, x2, x3, x4, condition )
 begin
  case condition is
     when "00" => F <= x1;
     when "01" => F <= x2;
     when "10" => F <= x3;
     when others => F <= x4;
  end case;
 end process;
end second1;
```

1.3 Схемні структури поведінкових описів умовних операторів

Розглянемо синтез та моделювання умовного послідовного оператора if з складним умовним виразом (лістинг 1.3). На рисунку 1.1 наведений результат синтезу вказаної мовної конструкції, результатом синтезу є мультиплексор MX 2-в-1 та комбінаційна схема, яка реалізує складний умовний вираз.

Лістинг 1.3 − VHDL-модель умовного оператора `if`

```
architecture Behavioral of test1 is
begin
process (a0,a1,a2,a3,A,B)
begin

   if ((a3 and a2 and a1 and a0)='1') then y<=A;
   else                                    y<=B;
   end if;

end process;
end Behavioral;
```

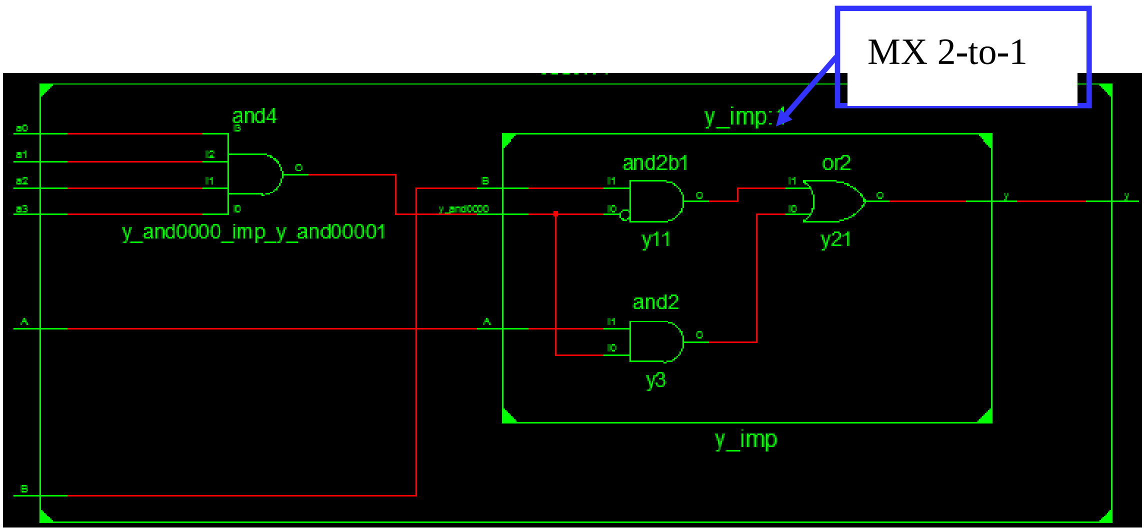
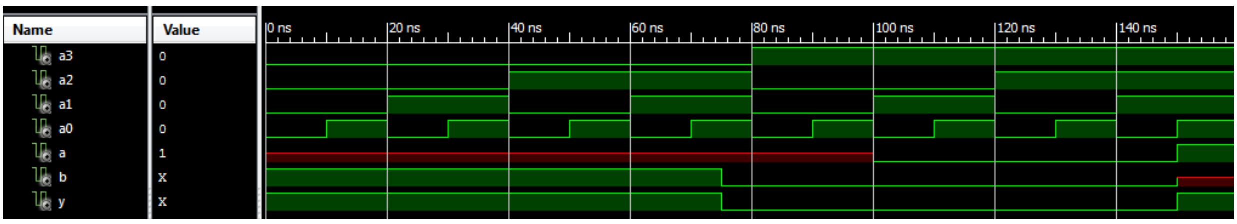


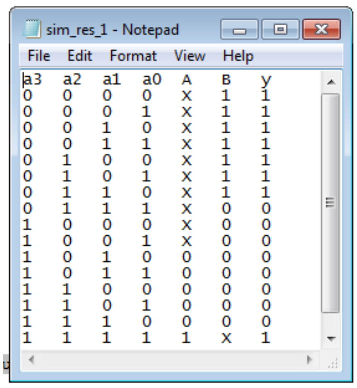
Рисунок 1.1 − Результат синтезу RTL-схема для оператора if

На лістингу 1.4 наведений фрагмент TestBench для VHDL-моделі умовного оператора `if` на вичерпному тесті (64 набора). Результат моделювання виводиться на waveform та у файл для подальшого використання (рис. 1.2).

Лістинг 1.4 – Фрагмент TestBench для VHDL-моделі умовного оператора `if`

```
Instantiate the Unit Under Test (UUT)
 uut: test1 PORT MAP ( a3 => a3, a2 => a2, a1 => a1, a0 => a0, y => y, A => A, B =>B  );

process
begin
B<='1'; A<='X'; wait for 75 ns;
B<='0';  wait for 25 ns;
A<='0';  wait for 50 ns;
B<='X'; A<='1';  wait;
end process;

process
FILE out_file : TEXT OPEN WRITE_MODE IS "sim_res_1.txt";
VARIABLE out_buff : LINE;
begin
     WRITE(out_buff, "a3  a2  a1  a0  A   B   y");
           WRITELINE(out_file, out_buff);
wait;
end process;

process
FILE out_file : TEXT OPEN APPEND_MODE IS "sim_res_1.txt";
VARIABLE out_buff : LINE;
begin
           WRITELINE(out_file, out_buff);                    --write line to file
wait for period;
end process;
```





a3	a2	a1	a0	A	B	y
0	0	0	0	X	1	1
0	0	0	1	X	1	1
0	0	1	0	X	1	1
0	0	1	1	X	1	1
0	1	0	0	X	1	1
0	1	0	1	X	1	1
0	1	1	0	X	1	1
0	1	1	1	X	0	0
1	0	0	0	X	0	0
1	0	0	1	X	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	1	X	1

Рисунок 1.2 – Результат моделювання умовного оператора if

Повний результат синтезу та моделювання умовного оператора if наведений у Додатку А.

Розглянемо синтез та моделювання умовного послідовного оператора багатопозиційного вибору `case` з складним умовним виразом (лістинг 1.5). На рисунку 1.3 наведений результат синтезу вказаної мовної конструкції, результатом синтезу є мультиплексор MX 4-в-1 та комбінаційна схема, яка реалізує складний умовний вираз.

Лістинг 1.5 – VHDL-модель оператора вибору `case`

```
architecture Beh of MuX is
begin
process(A,S)
begin
  case S is
    when "00" => y<= A(0);
    when "01" => y<= A(1);
    when "10" => y<= A(2);
    when "11" => y<= A(3);
    when others => y<='X';
  end case;
end process;
end Beh;
```

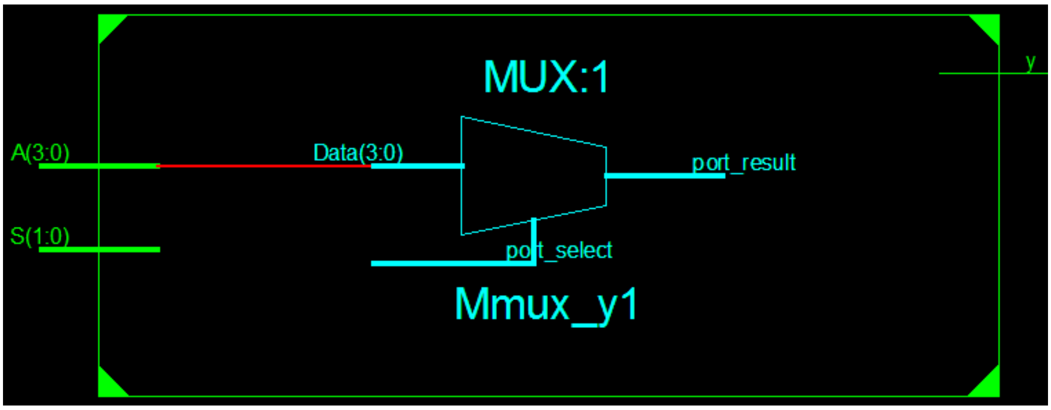
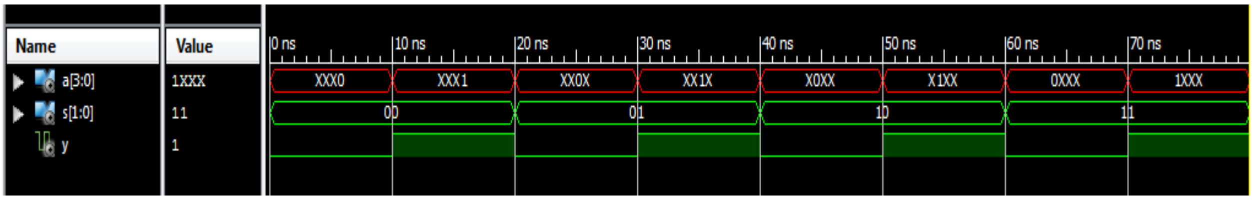


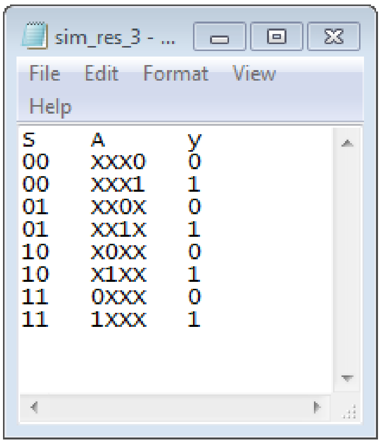
Рисунок 1.3 – Результат синтезу RTL-схема для оператора `case`

На лістингу 1.6 наведений фрагмент TestBench для VHDL-моделі умовного оператора case на вичерпному тесті (64 набора). Результат моделювання виводиться на waveform та у файл для подальшого використання (рис. 1.4). Повний результат синтезу та моделювання умовного оператора `case` наведений у Додатку Б.

Лістинг 1.6 – Фрагмент TestBench для VHDL-моделі оператора `case`

```
   --Inputs
   signal A : std_logic_vector(3 downto 0) ;
   signal S : std_logic_vector(1 downto 0) ;

  --Outputs
   signal y : std_logic;
   constant period:time:= 10 ns;

BEGIN
    uut: MuX PORT MAP (A => A, S => S,y => y);

process
begin
S<="00";
A<="XXX0"; wait for period;
A<="XXX1"; wait for period;
S<="01";
A<="XX0X"; wait for period;
A<="XX1X"; wait for period;
S<="10";
A<="X0XX"; wait for period
A<="X1XX"; wait for period;
S<="11";
A<="0XXX"; wait for period;
A<="1XXX";   wait;
end process;

process
FILE out_file : TEXT OPEN WRITE_MODE IS "sim_res_3.txt";
VARIABLE out_buff : LINE;
begin
      WRITE(out_buff, "S    A      y");       WRITELINE(out_file, out_buff);
wait;
end process;

process
FILE out_file : TEXT OPEN APPEND_MODE IS "sim_res_3.txt";
VARIABLE out_buff : LINE;
begin
                       WRITE(out_buff, S); WRITE(out_buff, "   ");
                        WRITE(out_buff, A); WRITE(out_buff, "   ");
                        WRITE(out_buff, y);
                        WRITELINE(out_file, out_buff);
      wait for period;
end process;
END;
```





S	A	y
00	XXX0	0
00	XXX1	1
01	XX0X	0
01	XX1X	1
10	X0XX	0
10	X1XX	1
11	0XXX	0
11	1XXX	1

Рисунок 1.4 – Результат моделювання оператора `case`

1.4 Постановка задачі

Виходячи з отриманих результатів можна зробити висновки, що умовні оператори мови опису апаратури перетворюються у схемні конструкції мультиплексорів MX 2-в-1 та MX 4-в 1. Таким чином, для моделювання несправностей умовних операторів мов опису апаратури слід розглянути різні методи моделювання несправностей для схем MX 2-в-1 та MX 4-в 1.

Ще одною особливістю запропонованої методики є можливість отримання повної таблиці істинності всієї схеми, що дозволяє ефективно використовувати кубітні методи аналізу цифрових схем.

Метою даного дослідження є підвищення швидкості та якості створення діагностичного забезпечення цифрових пристроїв за рахунок створення оптимальних структур даних для процедур дедуктивного моделювання несправностей на основі структурно-функціональних моделей комбінаційних схем.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) провести аналіз паралельних та послідовних умовних операторів мов опису апаратури тата схемних структур, в які вони синтезуються;

2) розробити процедури формування таблиць істинності схемних структур, поданих мовами опису апаратури;

3) розробити універсальні структури даних для кубічного та аналітичного дедуктивного моделювання несправностей;

4) удосконалити векторні моделі кубітного представлення структур і компонентів цифрових систем на основі адресного кодування вхідних сигналів;

5) розробити процедури отримання булевих похідних шляхом перестановок розрядів таблиць істинності та використанням операції XOR;

6) розробити структури даних для дедуктивного моделювання несправностей на основі кубітного представлення компонентів цифрових схем.

Об'єкт дослідження – процеси побудови діагностичного забезпечення цифрових систем на основі використання інтерпретативних моделей даних.

Предмет дослідження – табличні та кубітно-векторні моделі опису комбінаційних схем та процедури дедуктивного моделювання несправностей на основі цих моделей.

2 ДЕДУКТИВНІ МЕТОДИ МОДЕЛЮВАННЯ НЕСПРАВНОСТЕЙ

2.1 Дедуктивний алгоритм моделювання несправностей

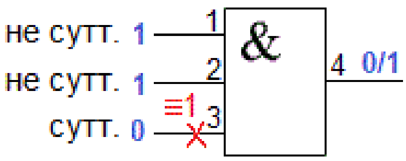
Всі методи моделювання несправностей за способом реалізації можна поділити на дві групи: явні та неявні. При реалізації явних методів моделювання виконується математичне "внесення" до схеми несправностей (по одній або групам) та моделювання схеми із цими несправностями. До явних методів моделювання несправностей відносяться одиночний та паралельний методи. При реалізації явних методів кількість циклів моделювання схеми на одному наборі дорівнює числу несправностей (або числу внесених груп несправностей). При реалізації неявних методів виконується обчислення списку несправностей, що перевіряються (виявлені) на одному наборі. У цьому число циклів моделювання схеми дорівнює кількості наборів плюс один. При неявних методах моделювання несправностей "внесення" несправностей замінюється спеціальними формулами або процедурами обробки списків несправностей на примітивах схеми. Надалі у всьому викладенні матеріалу будуть розглядатися одиночні константні несправності (ОКН), якщо не буде обумовлене інше.

Для отримання формул неявного моделювання несправностей використовується поняття суттєвість змінної на наборі (двійковому векторі). Змінна функціонала примітивного елементу (ПЕ) є суттєвою на наборі, якщо її зміна на протилежне призводить до зміни на протилежне значення виходу функціоналу ПЕ. Порівняємо це визначення з основним правилом перевірки (моделювання) несправностей на наборі.

Існує правило, що несправність перевіряється на тесті (двійковому наборі), якщо вона змінює стан хоча б одного виходу, що спостерігається.

У такий спосіб можна сформулювати основне правило неявного моделювання несправностей :несправність перевіряється на виході ПЕ, якщо вона з'являється на суттєвому вході на даному двійковому наборі. Крім того, на виході ПЕ перевіряються і всі несправності попередніх рангів схеми, пов'язані з цим входом.

Існує загальна формула отримання списків несправностей на виходах ПЕ для неявних методів у термінах суттєвості (рис. 2.1).



$$S_{вих ПЭ} = \left(\bigcap S_{вх}^{сутт} / \bigcup S_{вх}^{несутт}\right) + S_{вих}$$

Рисунок 2.1 – Формула отримання списків несправностей на виходах ПЕ для неявних методів

До неявних методів належать дедуктивний, спільний та кубічний та інші методи моделювання несправностей. У кожному із зазначених методів ця формула конкретизується залежно від механізму отримання списків несправностей.

Найпростішим неявним методом моделювання несправностей є дедуктивний метод (deductive method) або метод Армстронга (на ім'я американського професора, який розробив даний метод) [2]. Даний метод орієнтований на обробку схем вентильного рівня булевого (І, АБО, НІ) або універсального базисів. Суть дедуктивного моделювання полягає у заміні обробки законів функціонування ПЕ обробкою списків несправностей, що надходять на його входи. Для відповідних типів ПЕ розроблено формули отримання списків несправностей на виходах ПЕ залежно від виду двійкових наборів на входах ПЕ, які є окремим випадком формули (рис 2.1).

Алгоритм дедуктивного моделювання несправностей складається з наступних кроків.

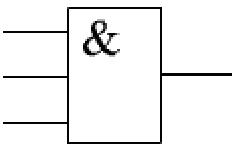
1. Побудова структурно-функціональної моделі схеми.

2. Виконання справного моделювання заданого вхідного набору.

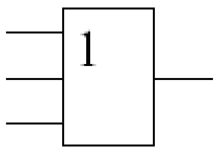
3. Складання початкових списків несправностей на зовнішніх входах

схеми, що моделюється. У початкові списки входять константні несправності, які є інверсними від справних значень сигналів зовнішніх входах.

4. Вибираються по порядку ПЕ схеми і для кожного з них виконується формування вихідних списків несправностей на підставі результатів справного моделювання та дедуктивних формул для елементів логічного базису (І, АБО, НІ, І-НІ, АБО-НІ). Варто зазначити, що інверсія виходу ПЕ не впливає на вигляд формул дедуктивного моделювання. Нижче наведено дедуктивні формули для елементів І та АБО.



S^0 - вхід дорівнює 0;
$S^1 \to$ дорівнює 1
(0 - суттєве значення)
$S_{вих} = \cap S^0 \setminus \cup S^1$
$S_{вих} = \cup S^1$, якщо всі 1



S^1 - вхід дорівнює 1; S^0
$\to$ дорівнює 0
(1 - суттєве значення)
$S_{вих} = \cap S^1 \setminus \cup S^0$
$S_{вих} = \cup S^0$, якщо всі 0

Рисунок 2.2 – Формули дедуктивного моделювання для булевого базису

5. Пункт 4 виконується до тих пір, поки не будуть складені списки перевірених несправностей для всіх ПЭ.

6. На основі результуючого списку несправностей на зовнішньому виході (виходах) схеми формується рядок таблиці несправностей.

Дедуктивний алгоритм моделювання несправностей ефективно працює для схем вентильного рівня, а також зручний при вивченні ручних методів моделювання несправностей. Для схем вищого рівня ієрархії складання та використання дедуктивних формул досить складно.

Перевагою дедуктивного моделювання несправностей є простота, наочність і число циклів моделювання, яке дорівнює числу наборів, що моделюються. Основним недоліком є наявність дедуктивних формул обробки списків несправностей лише для елементів вентильного рівня. Наслідком цього недоліку є необхідність перетворення схеми будь-якого рівня складності у вентильний еквівалент (базису І, АБО, НІ, І-НІ, АБО-НІ). Зазначений недолік робить дедуктивне моделювання несправностей суто "ручним" методом, який дуже зручний для застосування у навчальному процесі при вивченні методів моделювання несправностей.

Дедуктивне моделювання несправностей для комбінаційних схем вентильного рівня розглянемо на прикладі мультиплексора MX 2-в-1, функція якого має вигляд $y = x1x3 \vee x2\overline{x3}$ (y(2, 5, 6, 7) = 1) На рис. 2.3 наведено схемне позначення, вентильна схема, таблиця істинності (ТІ) та кубічне покриття зазначеного пристрою.

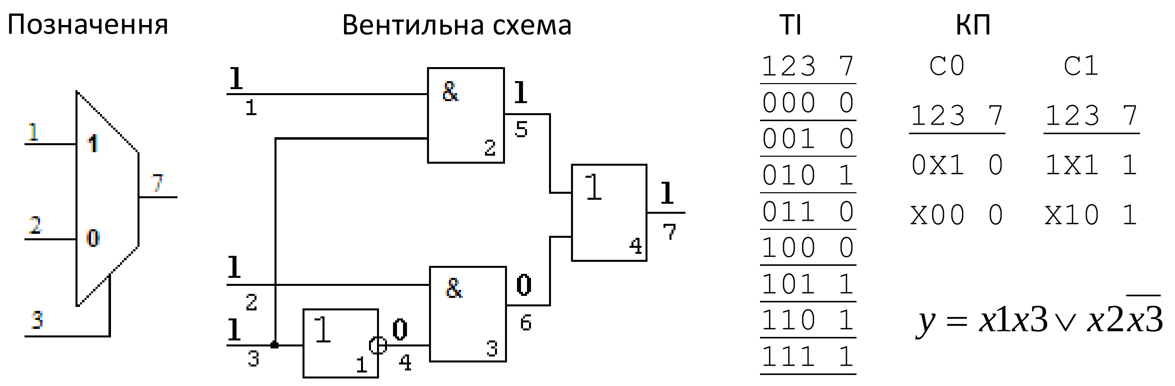


Рисунок 2.3 – Мультиплексор MX 2-в-1,

Для отриманої схеми виконаємо ранжирування шляхом нумерації її ліній, а потім виконаємо дедуктивне моделювання несправностей для вичерпного тесту – $2^n = 8$ наборів. На рис. 2.2 наведені результати простого дедуктивного моделювання несправностей для вичерпного тесту.

Результати дедуктивного моделювання несправностей можуть представлятися як у вигляді списків несправностей, так і у вигляді таблиць несправностей (ТН), рядки яких відповідають наборам, а стовпці –

На підставі результатів моделювання несправностей на всіх наборах заповнюємо багатозначну (поєднану) таблицю несправностей. Число рядків даної ТН дорівнює числу наборів (8) , а число стовпців - числу ліній (7) . Якщо на розглянутому наборі (рядок ТН) перевіряється несправність типу "0" на лінії, то у відповідному стовпці ставиться 0, а якщо перевіряється несправність типу "1" на лінії, то у відповідному стовпці ставиться 1.

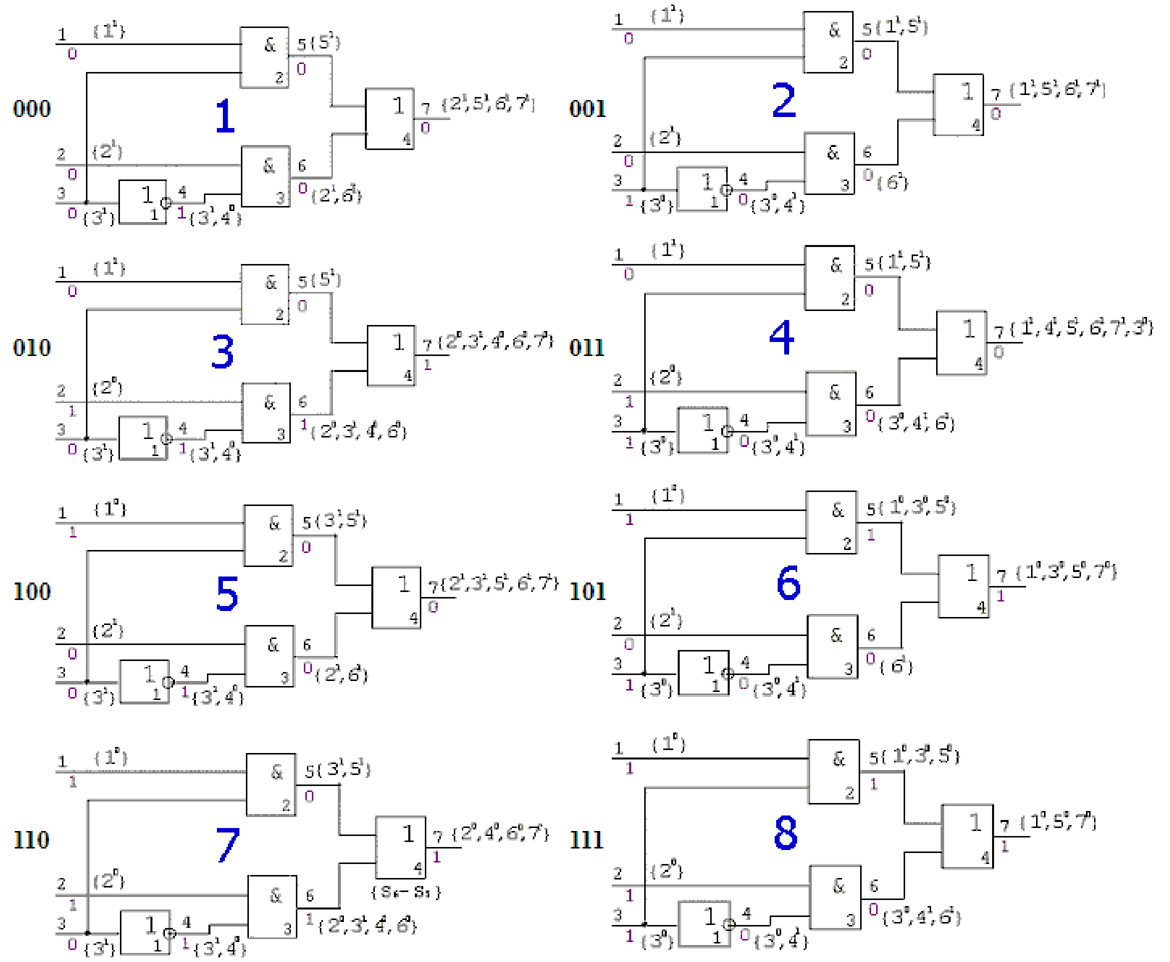
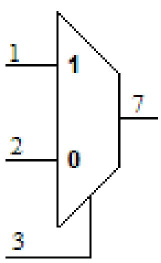


Рисунок 2.4 – Результати дедуктивного моделювання несправностей для вентильної схеми мультиплексора МХ 2-в-1,

Викреслимо із ТН внутрішні лінії та представимо різну послідовність наборів в ТІ за умови, який розряд є молодшим у двійковому наборі (рис.2.5).



ТН для down to

	123 7	1	2	3	7
0	000 0	-	1	-	1
1	001 0	1	-	-	1
2	010 1	-	0	1	0
3	011 0	1	-	0	1
4	100 0	-	1	1	1
5	101 1	0	-	0	0
6	110 1	-	0	-	0
7	111 1	0	-	-	0

ТН для to

	321 7	1	2	3	7
0	000 0	-	1	-	1
4	001 0	-	1	1	1
2	010 1	-	0	1	0
6	011 1	-	0	-	0
1	100 0	1	-	-	1
5	101 1	0	-	0	0
3	110 0	1	-	0	1
7	111 1	0	-	-	0

ТН для КП

	123 7	1	2	3	7
0	000 0	-	1	-	1
1	001 0	1	-	-	1
3	011 0	1	-	0	1
4	100 0	-	1	1	1
2	010 1	-	0	1	0
5	101 1	0	-	0	0
6	110 1	-	0	-	0
7	111 1	0	-	-	0

Рисунок 2.5 – Варіанти ТН для мультиплексора МХ 2-в-1

2.2 Кубічне моделювання несправностей комбінаційних схем

Кубічне моделювання несправностей є машинно-орієнтованим алгоритмом в інтерпретативних системах моделювання. Воно ґрунтується на поданні опису функціоналів структурно-функціональної моделі кубічними покриттями (мінімізованими таблицями істинності) для комбінаційних схем алфавіту $A^1 = \{0, 1, U, X\}$, а для послідовних схем в алфавіті A^2 (16).

Розглянемо табличний спосіб опису комбінаційного пристрою та суттєвість змінних в ньому. Будь-яка комбінаційна схема $Y = f(x_1,x_2, ...x_n)$, де x_i - вхідні змінні, а Y – вихідна змінна може бути описана таблицею істинності або кубічним покриттям (КП).

Булева змінна x_i суттєва на двійковому наборі $(x_1, x_2, ..., x_n)$, якщо зміна її значення на протилежне призведе до зміни на протилежне значення функції Y на цьому наборі.

Булева змінна (2.1) x_i є суттєвою, якщо булева похідна даної функції для змінної x_i дорівнює 1, тобто інші змінні функції мають такі значення, щоб похідна дорівнювала 1.

$$\frac{dY}{dX_i} = f(x_1 x_2 \cdot x_{i-1}, 0, x_{i+1} \cdots x_n) \oplus f(x_1 x_2 \cdot x_{i-1}, 1, x_{i+1} \cdots x_n) \qquad (2.1)$$

Суттєва змінна x_i на двійковому наборі $(x_1,x_2, ... x_n)$ визначається значенням булевої похідної на цьому наборі. Якщо булева похідна по x_i на двійковому наборі $(x_1,x_2, ... x_n)$ приймає значення 1, то вона суттєва на цьому наборі. В іншому випадку змінна x_i несуттєва на даному наборі. Група булевих змінних $x_i...x_j$ суттєва на двійковому наборі $(x_1,x_2, ... x_n)$, якщо одночасна зміна всіх значень зазначених змінних на протилежні призведе до зміни значення функції Y.

Кубічне покриття (КП) елемента є мінімізованою таблицею істинності, значення координат якої представлені в алфавіті {0, 1, X}. З погляду умов суттєвості кожен куб КП є однією з умов суттєвості змінних аналізованої функції. Якщо змінна в кубі КП не дорівнює X, вона вважається суттєвою (передбачається, що КП елемента є мінімальним). Кубічне покриття в цілому є повним переліком всіх умов суттєвості вхідних змінних примітивного елемента (ПЕ). З погляду виявлення константних несправностей входів ПЕ, що реалізує функцію $Y = f(x_1, x_2, \ldots\ldots x_n)$, на наборі $(x_1, x_2, \ldots\ldots x_n)$, ці несправності виявлятимуться на виході ПЕ, якщо вони змінюють значення сигналів на входах, суттєвих на цьому вхідному наборі [3, 4].

Виходячи з вищезазначеного, для отримання списку несправностей ПЕ, що виявляються на наборі $(x_1, x_2, \ldots\ldots x_n)$, необхідно визначити входи ПЕ (групи входів), суттєві на даному наборі, і потім за формулою (2.2) отримати список несправностей, що виявляються.

$$S_{\text{вихПЭ}} = \bigcup^{\text{гр.сутт}} \left(\bigcap S_{\text{вх}}^{\text{сутт}} / \bigcup S_{\text{вх}}^{\text{несутт}} \right) + S_{\text{вих}} \qquad (2.2)$$

Тут $S_{\text{вих}}$– власна несправність виходу (інверсія від справного значення).

За аналогією із взяттям булевої похідної (2.1) функції, заданої булевим рівнянням, для визначення суттєвості змінних на наборі, якщо функція ПЕ задана у вигляді КП, використовується операція суми за модулем 2 вхідного набору T з усіма кубами КП C_{ij} (i-кількість кубів у КП, j-число координат у КП). Бінарна координатна операція $\oplus$ (XOR), яка визначає взаємодію координат вхідного набору T з координатами кубів покриттів C_{ij} у троїчному алфавіті наведена на рис. 2.6.

$$R_{ij} = T \oplus C_{ij}$$

$\oplus$	0	1	X
0	0	1	X
1	1	0	X
X	X	X	X

Рисунок 2.6 – Кубічна операція XOR

Суттєвими вважаються ті змінні, на координатах яких R_{ij}=1. Якщо R_{ij}=0, координата вважається несуттєвою, а якщо R_{ij}=X, координата індиферентна (не впливає на вихід). Список вхідних несправностей проявляється на виході ПЕ, якщо R_{ij} для вихідної координати дорівнює 1, при цьому формула отримання списку несправностей L_i для одного і-го куба КП має вигляд (2.3).

$$L_i = \begin{cases} \varnothing \;, & \text{якщо } R_{i\,\text{вих}} = 0 \text{ або } X; \\ \bigcap\limits_j S_j^{R_{ij}=1} - \bigcup\limits_j S_j^{R_{ij}=0}, & \text{якщо } R_{i\,\text{вих}} = 1. \end{cases} \tag{2.3}$$

Побудова структурно-функціональної моделі здійснюється шляхом ранжування ліній схеми та нумерації її ПЕ.

1. Виконується нумерація в порядку зовнішніх входів схеми, тобто. ліній не мають попередників.

2. Наступними номерами нумеруються виходи примітивів, входи яких занумеровані, і які є зовнішніми виходами, тобто. мають наступників.

3. Пункт 2 виконується доти, доки не виявляться занумерованими всі внутрішні лінії схеми.

4. Наступними порядку номерами нумеруються зовнішні виходи схем, тобто. лінії не мають наступників.

5. Нумеруються примітивні елементи у порядку проходження номерів вихідних ліній даних елементів.

6. Складається список різних типів примітивних елементів для кожного типу ПЕ будуватися таблична модель його функціонування у формі кубічного покриття в алфавіті {0, 1, X}.

Алгоритм кубічного моделювання несправностей складається з наступних кроків.

1. Побудова структурно-функціональної моделі схеми.

2. Виконання справного моделювання заданого вхідного набору.

3. Складання початкових списків несправностей на зовнішніх входах схеми, що моделюється. У початкові списки входять константні

несправності, що визначаються інверсними від справних значеннями сигналів на зовнішніх входах.

4. Вибираються по порядку ПЕ схеми, і для кожного з них виконується формування вихідних списків несправностей на підставі результатів справного моделювання та вхідних списків поточного ПЕ:

– виконується сума за модулем 2 вхідного набору ПЕ з кубами його КП;

– за результатами п.4.1 на підставі формули (5) формується частковий список несправностей для одного куба КП;

– загальний список несправностей ПЕ формується виходячи з підсумовування часткових списків для кожного куба плюс власна несправність виходу ПЕ, $S_{ПЭ} = \sum_{i=1}^{m} L_i + S_{вых}$, де m – кількість кубів КП, $S_{вых}$ – власна несправність виходу ПЕ (значення, протилежне справному значенню на виході ПЕ).

5. Пункт 4 виконується доти, доки не будуть складені списки несправностей, що перевіряються, для всіх ПЕ.

6. На підставі результуючого списку несправностей на зовнішньому виході формується рядок таблиці несправностей.

Побудова структурно-функціональної моделі комбінаційної схеми MX 2-в-1 наведено на рис. 2.7.

Розглянемо виконання кроків алгоритму.

1. Дана комбінаційна схема MX 4-в-1, що складається з функціональних елементів MX 2-в-1. Виконаємо побудову структурно-функціональної моделі схеми.

2. Виконаємо справне моделювання. Для демонстрації справного моделювання подамо на схему набір 111110. Результатом справного моделювання будуть значення сигналів в алфавіті {0,1} на всіх лініях схеми, які представлені на рис. 2.7 .

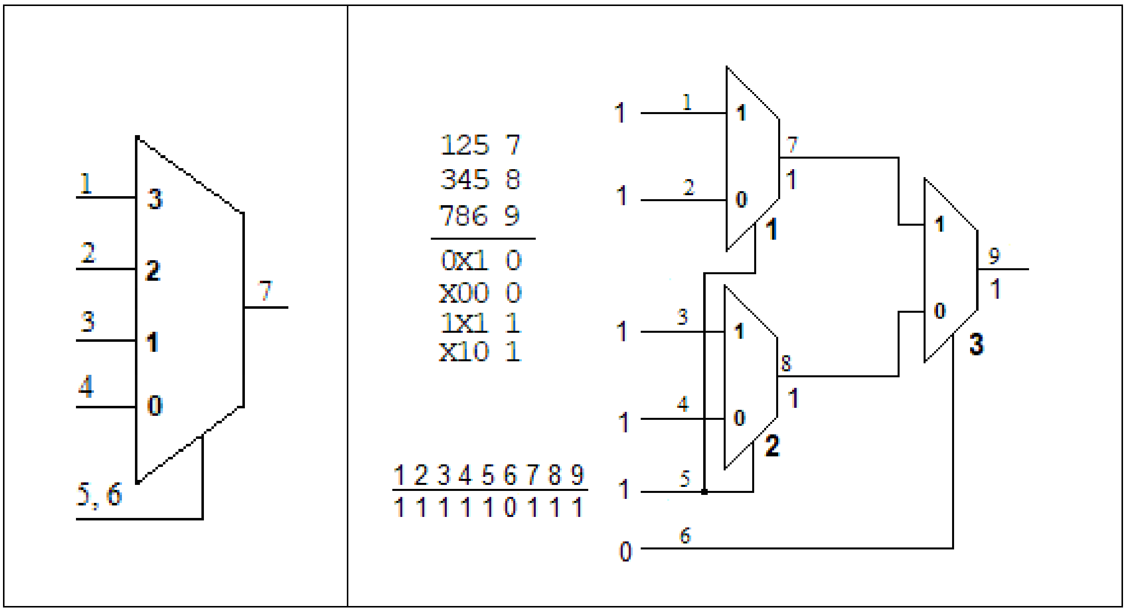


Рисунок 2.7 – Структурно-функціональна модель та справне моделювання схеми мультиплексора MX 4-в-1

3. Складаємо початкові списків несправностей:

$S_1 = \{1^0\}$ $S_2=\{2^0\}$ $S_3=\{3^0\}$ $S_4=\{4^0\}$ $S_5=\{5^0\}$ $S_4=\{6^1\}$

4. Для кожного ПЕ схеми виконується формування вихідних списків несправностей, а саме:

– виконується сума за модулем 2 вхідного набору ПЕ з кубами його КП;

– за результатами ⊕ та на підставі формули (2.3) формується частковий перелік несправностей;

– формується загальний перелік несправностей ПЕ.

5. Пункт 4 виконується доти, доки не будуть складені списки несправностей, що перевіряються, для всіх ПЕ.

6. Формується рядок багатозначної (суміщеної) таблиці несправностей.

Результат проведення кубічного моделювання несправностей наведений на рис. 2. 8.

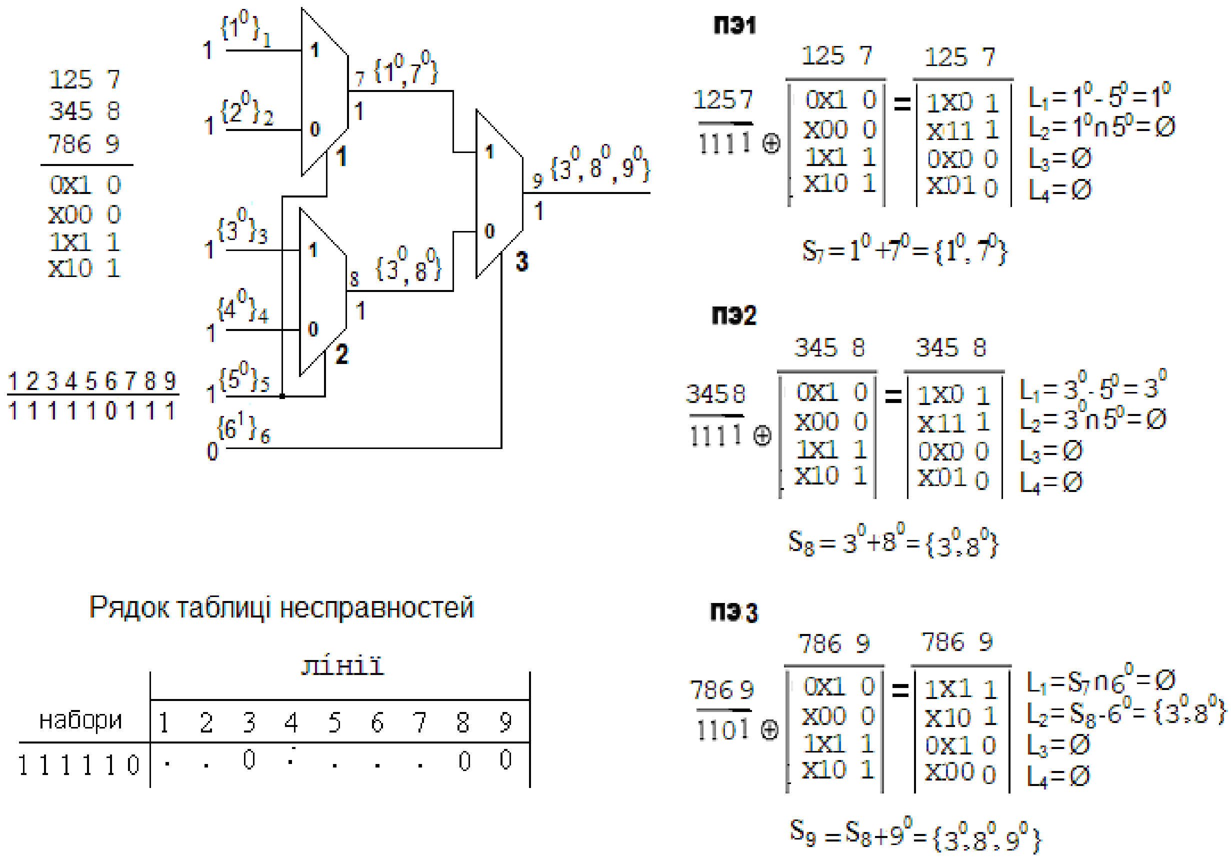
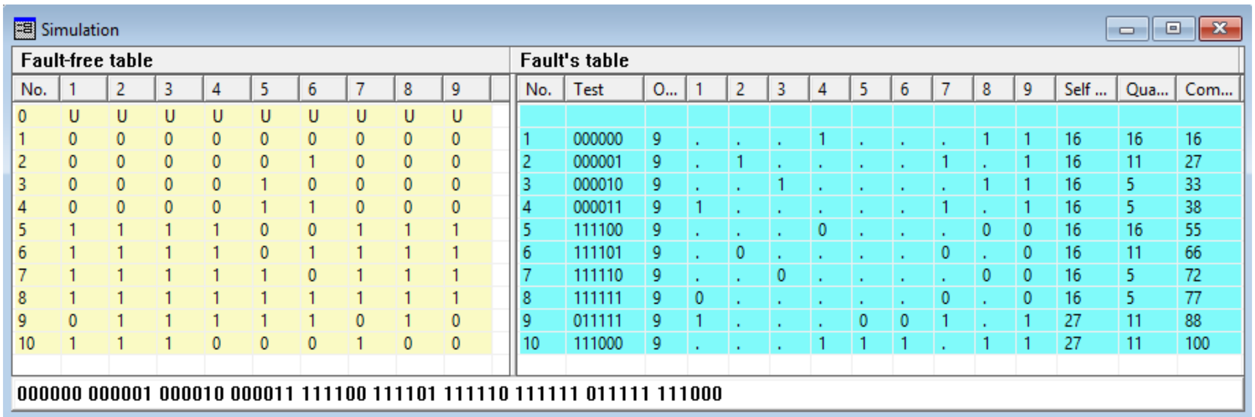


Рисунок 2.8 – Справне та несправне кубічне моделювання

У суміщеній таблиці несправностей, кількість стовпців якої дорівнює кількості ліній схеми, в стовпці ставиться 1 - якщо на даній лінії перевіряється несправність константа 1, 0 - якщо перевіряється константа 0. Якщо жодна несправність на лінії не перевіряється, то у відповідному стовпці ставиться точка "."

Алгоритм кубічного моделювання несправностей реалізований у програмі DCP (Deductive Circuit Processor), багаторічне використання якої у навчальному процесі ХНУРЕ підтверджує працездатність зазначеного метода. На рис.2.9 представлені результати моделювання фрагменту вичерпного тесту схеми MX 4-в-1 в програмі кубічного моделювання DCP. Слід відмітити, що повнота наведеного фрагменту тесту дорівнює 100%. Для набору 111101 результати співпадають з результатами ручного моделювання.


Simulation

Fault-free table

No.	1	2	3	4	5	6	7	8	9
0	U	U	U	U	U	U	U	U	U
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0	0
4	0	0	0	0	1	1	0	0	0
5	1	1	1	1	0	0	1	1	1
6	1	1	1	1	0	1	1	1	1
7	1	1	1	1	1	0	1	1	1
8	1	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	0	1	0
10	1	1	1	0	0	0	1	0	0

Fault's table

No.	Test	O...	1	2	3	4	5	6	7	8	9	Self ...	Qua...	Com...
1	000000	9	.	.	.	1	.	.	.	1	1	16	16	16
2	000001	9	.	1	.	.	.	.	1	.	1	16	11	27
3	000010	9	.	.	1	.	.	.	.	1	1	16	5	33
4	000011	9	1	.	.	.	.	.	1	.	1	16	5	38
5	111100	9	.	.	.	0	.	.	.	0	0	16	16	55
6	111101	9	.	0	.	.	.	.	0	.	0	16	11	66
7	111110	9	.	.	0	.	.	.	.	0	0	16	5	72
8	111111	9	0	.	.	.	.	.	0	.	0	16	5	77
9	011111	9	1	.	.	.	0	0	1	.	1	27	11	88
10	111000	9	.	.	.	1	1	1	.	1	1	27	11	100

000000 000001 000010 000011 111100 111101 111110 111111 011111 111000

Рисунок 2.9 – Результати моделювання фрагменту тесту схеми MX 4-в-1 в програмі кубічного моделювання DCP

2.3 Метод моделювання несправностей по дедуктивним формулам

Подальшим удосконаленням методів дедуктивного моделювання несправностей є дедуктивно-паралельний метод [5], який використовується як у програмному, так і в апаратному аналізі якості тестів.

Отримання рівнянь дедуктивних функцій (ДФ) ґрунтується на взаємодії тесту та функції справної поведінки. Далі розглядається модель дедуктивно-паралельного синхронного аналізу несправностей, яка дозволяє за одну ітерацію обробки схеми обчислювати дефекти, що перевіряються на двійковому тест-векторі. Вона заснована на рішенні D-рівняння: $L = T \oplus F$, де $F = (F_{m+1}, F_{m+2}, ..., F_i\ , ...F_n\ )$ $(i = m+1,n)$ – сукупність функцій справної поведінки пристрою, m - кількість його входів; $Y_i = F_i\ (X_{i1}\ , ..., X_{ij}\ , ..., X_{in}\ )$ де-n_i - вхідний i-й елемент схеми, що реалізує F_i для визначення стану лінії (виходу) Y_i на тест-векторі T_t; X_{ij} - j-й вхід i-го елемента; двійковий тест $T = (T_1, T_2, ..., T_t\ , ..., T_k\ )$ – упорядкована сукупність двійкових векторів, довизначена в процесі справного моделювання на множині вхідних, внутрішніх та вихідних ліній, де тест-вектор задається у вигляді $T_t = (T_{t1}, T_{t2}, ..., T_{ti}\ , ..., T_{tn}\ )$, для якого невхідні координати обчислюються моделюванням функції $Y_i = F_i\ (X_{i1}\ , ..., X_{ij}\ , ..., X_{in}\ )$ на тест-векторі T_t; $L = (L_1, L_2, ..., L_t\ , ..., L_k\ )$ – множина дедуктивних схем або моделей, де

$L_t = L_{t1},L_{t2},...,L_{ti}\ ,...,\ L_{tn})$; $L_{ti} = T_t \oplus F_i$ – дедуктивна функція паралельного моделювання несправностей на тест-векторі T_t , що відповідає справному елементу F_i, та яка дає можливість обчислювати список вхідних несправностей, що транспортуються на вихід елемента F_i. З урахуванням розбиття тесту на складові вектори рівняння отримання ДФ для $T_t \in T$ набуває наступного вигляду: $L_t = T_t \oplus F_i$

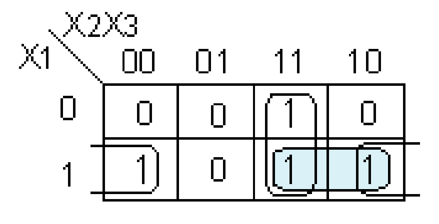
Якщо функціональний опис цифрового пристрою представлений компонентами (примітивами), що формують стан всіх ліній схеми, то як формула перетворення справної моделі примітиву F_i на тест-векторі T_t в дедуктивну функцію L_{ti} виступає L_{ti}-рівняння:

$$L_{ti} = T_t \oplus F_i = F_{ti}[(X_{i1} \oplus T_{t1}),(X_{i2} \oplus T_{t2}),...,(X_{ij} \oplus T_{tj}),...,(X_{ini} \oplus T_{tni})] \oplus T_{ti}\,. \qquad (2.4)$$

В основі реалізації формули (4) лежить властивість функції XOR, а саме $F \oplus 1 = \overline{F}$. Якщо рівняння функції $Y_i = F_i\ (X_{i1}\ ,...,X_{ij}\ ,...,X_{in}\ )$ задано у вигляді ДНФ, то для побудови дедуктивної функції інвертуємо змінні, де X_{ij}=1, інвертуємо всю функцію, якщо Y_i=1, і видаляємо надлишкові терми. На рис. 2.10 показана побудова дедуктивної функції для набора (111) з видаленням надлишкового терму x_1x_2 для дедуктивної функції L(111).

$$f(x_1,x_2,x_3) = (x_1x_3 \vee x_2\overline{x_3})(111) = 1$$

$$L(111) = (\overline{x_1}\,\overline{x_3} \vee \overline{x_2}\,\overline{\overline{x_3}}) \oplus 1 = \overline{\overline{x_1}\,\overline{x_3} \vee \overline{x_2}x_3} = (\overline{\overline{x_1}\,\overline{x_3}})(\overline{\overline{x_2}x_3}) =$$
$$= (x_1 \vee x_3)(x_2 \vee \overline{x_3}) = x_1x_2 \vee x_2x_3 \vee x_1\overline{x_3}$$



$$L(111) = x_2x_3 \vee x_1\overline{x_3}$$

Рисунок 2.10 – Побудова дедуктивної функції для набора 111

Таким чином дедуктивна функція для мультиплексора MX 2-в-1 буде мати вигляд :

L = (x2^x3 v x1x3(000)) V (x2x3 v x1^x3 (001)) V (x2^x3 v ^x1x3(010)) V (^x2x3 v x1^x3(011)) V V (x2^x3 v ^x1x3(100)) V (^x2x3 v x1^x3(101)) V (x2^x3 v x1x3(110)) V (x2x3 v x1^x3(111))

Перевіряємо отриману ДФ для вхідних списків a b c за умови $a \cap b \cap c = \varnothing$:

- набор 000 списки abc L=b^c v ac=b-c=b;
- набор 010 списки abc L=b^c v ^ac=b-c v c-a=b v c;
- набор 100 списки abc L=b^c v ^ac=b-c v c-a=b v c;
- набор 111 списки abc L=bc v a^c=a-c =a .

Отримані результати співпадають з результатами простого дедуктивного моделювання вентильної схеми (рис. 2.5).

3 СТРУКТУРИ ДАНИХ ДЛЯ КУБІТНОГО МЕТОДУ ДЕДУКТИВНОГО МОДЕЛЮВАННЯ НЕСПРАВНОСТЕЙ

3.1 Кубітна форма опису цифрових схем

В теорії методів автоматизованого проектування діагностичного забезпечення використовується поняття кубіту як двійкового або багатозначного вектора для спільного і одночасного завдання булеана станів в дискретній області кіберпростору на основі лінійної суперпозиції унітарних кодів, орієнтованих на паралельне використання методів аналізу та синтезу компонентів кіберпростору. Теоретично квантових обчислень вектори станів утворюють квантовий регістр з n кубітів, що формують унітарний або гільбертовий простір, що запропоновано проф.Хахановим В.І.[6].

Будь-який компонент функціональності, як і структура системи, представляється векторної формою, впорядкованої за адресами, таблицею істинності, реалізованої з допомогою пам'яті. Логічні функції у традиційному виконанні не розглядаються. Від цього частково зменшується швидкодія, але, враховуючи, що 94% SoC-кристалу становить пам'ять, 6%, що залишилися, слід реалізовувати на пам'яті, що не буде критичним для більшості хмарних сервісів. Практично для створення ефективних комп'ютингових структур слід використовувати теорію, засновану на обчислювальних компонентах високого рівня абстракції: пам'ять, що адресується, і транзакція.

Особливість організації даних у класичному комп'ютері полягає у адресації біта, байта чи іншого компонента. Адресованість створює проблему обробки асоціації неадресованих елементів множини, які не мають порядку визначення. Рішенням може бути процесор, де образ універсуму з n унітарно кодованих примітивів використовує суперпозицію формування булеана $|B(A)|=2^n$ всіх можливих станів.

Коректність використання прикметника «кубітна» для моделей цифрових пристроїв заснована на порівнянні лінійної та булевої алгебри Кантора серветом (алфавітом) $A^1 = \{0, 1, X, U\}$. Тут перші два символи – примітиви. Третій визначається суперпозицією: $X = 0 \cup 1$. Символ U є доповненням до універсуму.

Отже, структуру даних типу «булеан» можна розглядати як детермінований образ квантового кубіту в алгебрі логіки, елементи якої унітарно кодуються двійковими векторами і мають властивості суперпозиції, паралелізму і переплутування. Це дає можливість використовувати пропоновані моделі для підвищення швидкодії аналізу цифрових пристроїв на класичних обчислювачах, а також без модифікації – на квантових комп'ютерах, які з'являться через кілька років на ринку електроніки.

Кубит (n-кубіт) є векторною формою унітарного кодування універсуму з n примітивів для завдання булеану станів 2^{2^n} за допомогою 2^n двійкових змінних. Якщо n=2, то 2-кубіт задає 16 станів за допомогою чотирьох змінних, при n=1, кубит задає чотири стани на універсумі з двох примітивів (10) і (01) за допомогою двох двійкових змінних (00,01,10, 11). При цьому допускається суперпозиція у векторі станів 2^n, позначених примітивами. Кубіт дає можливість використовувати паралельні векторні логічні операції замість поелементних теоретико-множинних для істотного прискорення процесів аналізу дискретних систем. Далі за текстом кубіт ототожнюється з n-кубітом або двійковим вектором, якщо це не заважає розумінню матеріалу, що викладається. Синонімом кубіту при заданні двійкового вектора логічної функції є Q-покриття (Q-вектор) як уніфікована векторна форма суперпозиційного завдання вихідних станів, що відповідають адресним кодам вхідних змінних функціонального елемента.

Кубіт у цифровій системі виступає як форма завдання структурного примітиву, інваріантної до технологій реалізації функціональності (hardware, software). Понад те, «квантовий» синтез цифрових систем з урахуванням кубітних структур не прив'язаний жорстко до теореми Посту, визначальною умови існування функціонально повного базису. Формат структурного кубітного компонента цифрової схеми включає інтерфейс (вхідні та вихідні змінні), а також кубит-вектор (quantum vector) Q , що задає функцію Y = Q(X) , розмірність якого визначається ступеневою функцією від числа вхідних ліній $k = 2^n$. Новизна кубітної форми полягає в заміні невпорядкованих рядків ТІ функціональних елементів векторами упорядкованих станів виходів.

Практично орієнтована новизна кубитного моделювання полягає у заміні таблиць істинності компонентів цифрового пристрою на вектори станів виходів. Досить легко можна продемонструвати такі перетворення, стосовно логічного елементу. Якщо функціональний примітив NAND має відповідне двійкове покриття, йому можна поставити у відповідність кубіт або Q-покриття, в тому числі у десятковому еквіваленті (рис. 3.1).

$$P = \begin{array}{|cc|c|} X_1 & X_2 & Y \\ \hline 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array} \rightarrow Q = (1110) = (14).$$

Рисунок 3.1 – Принцип формування Q-вектора

Переваги n-кубіта полягає в здатності паралельно виконувати логічні операції над векторним форматом теоретико-множинних даних. Наприклад, xor-операція над A = {a, b, c, d, e, f} B={a,c,f,g,h,k} виконується паралельно за один такт, якщо кожен елемент буде представлений унітарним кодом, а підмножини - векторами, які є кубіт-операндами (рис.3.2).

9 – qubit	a	b	c	d	e	f	g	h	k
A =	1	1	1	1	1	1	0	0	0
B =	1	0	1	0	0	1	1	1	1
$A \oplus B$ =	0	1	0	1	1	0	1	1	1

Рисунок 3.2 – Логічні операції над кубітами

Процедура моделювання на Q-векторі функціональності зводиться до запису вихідну змінну Y стану біта, адреса якого сформований на основі конкатенації значень вхідних змінних: $Y = Q(X) = Q(X_1 * X_2 ... * X_j ... * X_k)$. Для моделювання цифрових систем, де компонентами виступають взаємозалежні з урахуванням М-вектора еквіпотенційних ліній Q-примитиви, процедура обробки останніх визначається виразом: $M(Y) = Q[M(X)] = Q[M(X_1 * X_2 ... * X_j ... * X_k)]$. З урахуванням наскрізної нумерації Q-примітивів, універсальна процедура моделювання поточного і-елемента матиме формат: $M(Y_i) = Q_i[M(X_i)] = Q_i[M(X_{i1} * X_{i2} ... * X_{ij} ... * X_{ik_i})]$. У разі істотно спрощується алгоритм аналізу цифрової системи та у 2^n раз підвищується швидкодія інтерпретативного моделювання з допомогою збільшення обсягу пам'яті для описи функціональності схемної структури [7].

Синтез Q-покриття цифрової системи зводиться до виконання операції суперпозиції над Q-векторами функціональностей, що входять до неї. Наприклад, для трьох примітивів (елементи AND, NAND, NAND), що становлять схему, операція суперпозиції формує Q-вектор всієї функціональності (розряд “a” старший), де його розмірність буде більшою, ніж сума Q-покриттів вихідних примітивів (рис.3.3).

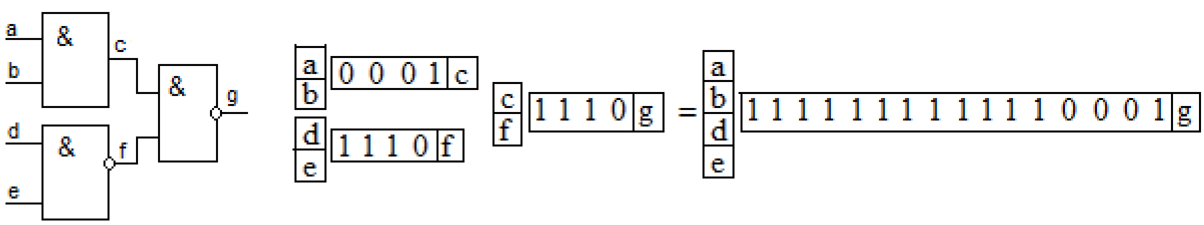


Рисунок 3.3 – Суперпозиція кубітних функціональностей

Але при цьому процедура моделювання Q-вектора структури матиме більш високу швидкодію, оскільки вона представлена лише одним зверненням до Q-покриття для вилучення вмісту з комірки замість трьох, коли система представлена трьома примітивами.

Триелементна схема, представлена вище Q-векторами, може бути задана схемотехнічно (зручною формою для людини), де замість векторів фігуруватимуть відповідні десяткові еквіваленти (рис. 3.4).

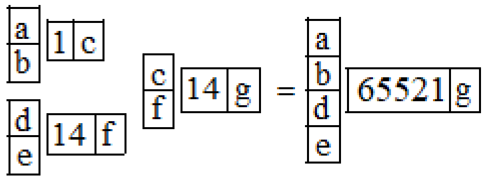


Рисунок 3.4 – Представлення кубітів десятковими еквівалентами

При обробці такої форми функціональних покриттів необхідно «розгорнути» десятковий код у двійковий вектор і обчислити адресу комірки, вміст якої визначатиме стан вихідної змінної, в даному випадку g. Природно, що десятковий код існує на папері, а в комп'ютері це уявлення завжди двійковий вектор. Насправді «м'яка» схемотехніка ідентифікації (нумерації) міжз'єднань має майбутнє, оскільки не пов'язана зі сполучними проводами, які замінюються адресами або номерами ліній, що створюють структуру цифрового виробу.

Кубітне представлення функціональних елементів дає можливість ввести нові схемотехнічні позначення, пов'язані з десятковим номером Q-вектора, що задає функціональність. Якщо система логічних елементів має n=2 входу, то число всіх можливих функцій дорівнює $k = 2^{2^n}$, де типи або номери функціоналів представлені в нижньому рядку таблиці (рис 2.5).

00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
f =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Рисунок 3.5 – Кубіти двовходових логічних елементів

Більш того, на основі множини кубітів першого рівня, що задають функції від двох змінних, можна ввести кубіт другого рівня, що унітарно кодує двовходові функції, що дає можливість створювати структуру одночасного завдання та аналізу всіх невпорядкованих станів дискретної системи, де вхідними змінними виступають вже функціонали першого рівня (рис. 3.6)

00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Q =	0	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0

Рисунок 3.6 – Кубіти другого рівня

У цій таблиці представлені чотири вектори-примітиви вхідних змінних (00,01,10,11), що утворюють повну множину всіх можливих функцій, які розглядаються як примітиви другого рівня. Потім – вектори-примітиви вихідних змінних (16 стовпців від 0000 до 1111), формують вже функціональних примітивів, що входять до складу складнішої дискретної системи, які можна аналізувати паралельно! Далі можна екстраполювати створення складнішої системи кубитів, де вектор Q=0111110000001100, представлений нижньою рядком, буде розглядатися як один з примітивів третього рівня ієрархії. У кожному рівні ієрархії кубитів кількість або булеан станів експоненційно залежить від числа примітивів-векторів. Якщо вектор Q має всі поодинокі значення Q=1111111111111111, він одночасно визначає простір, що містить 16 символів двотактного алфавіту, які відповідають булеану на універсумі з чотирьох примітивів.

Основна інноваційна ідея квантових обчислень у порівнянні з машиною фон Неймана полягає в переході від обчислювальних процедур над байт-операндом, що визначає в дискретному просторі одне рішення (точку) до квантових паралельних процесів над кубіт-операндом, що одночасно формує булеан рішень. У цій тезі сформульовано майбутнє всіх високопродуктивних комп'ютерів для паралельного нецифрового аналізу структур та сервісів дискретного кіберпростору. Інакше, обчислювальна складність виконання процедури обробки множини з n елементів у «квантовому» процесорі та одного в машині фон Неймана рівні між собою за рахунок відповідного n-кратного підвищення апаратної складності «квантової» структури.

3.2 Булеві похідні для графічного представлення логічних функцій

Якщо розглядати схемну реалізацію булевих функцій, представлених у вигляді ДНФ або КНФ, то, без урахування інверсій, вони є дворівневими схемами I-АБО (АБО-I). Але в реальній схемотехніці подібні схеми трапляються досить рідко. Зазвичай загальні частини різних імплікант (імпліцент) на основі дистрибутивного закону "виносяться за дужки" в результаті чого виходить багаторівнева схема, аналітичний запис якої прийнято називати скобковою формою (СФ). Іноді у літературі операція " винесення за дужки " називається факторизацією чи виділенням загальних частин. У цьому підрозділі описується структура та одержання скобкових форм булевих функцій на основі ДНФ, але абсолютно аналогічні викладки застосовні і до КНФ. Наприклад, для функції трьох змінних y(2, 5, 6, 7) = 1) ДНФ має вигляд $y = x1x3 \vee x2\overline{x3}$

Для безпосереднього запису СФ булевої функції зазвичай використовують першу формулу розкладання :

$$f(x_1, x_2, ...x_i, ... x_n) = \overline{x}_i\, f(x_1, x_2, ... 0, ... x_n) \vee x_i\, f(x_1, x_2, ... 1, ... x_n);$$

Для простоти подальшого викладу $f(x_1, x_2, ... x_i, ... x_n)$ можна записати, як $f(x_i)$, $f(x_1, x_2, ... 0, ... x_n)$ – як $f^{\overline{x}_i}$, а $f(x_1, x_2, ...1, ... x_n)$ – як f^{x_i}. В результаті отримаємо першу формулу розкладання можна записати:

$$f(x_i) = \overline{x}_i \cdot f^{\overline{x}_i} \vee x_i \cdot f^{x_i}.$$

Для наочності часто використовують графічну форму подання розкладання змінної x_i. При цьому кожне розкладання змінної x_i являє собою вузол граф-схеми, всередині якого записується змінна, по якій проводиться розкладання. Нижнє ліве ребро (позначається "0") відповідає функції $f^{\overline{x}_i}$, а нижнє праве ребро ("1") - функції f^{x_i}. Вихід вузла (верхнє ребро) відповідає результуючій функції $f(x_i)$. При цьому якщо f є деяким булевим виразом, то його доцільно укладати в дужки. Звідси і випливає результуючий вираз у вигляді СФ.

Змінна x_i називається суттєвою функції f, якщо $f^{\overline{x}_i} \neq f^{x_i}$.. В іншому випадку змінна вважається несуттєвою. На рис. 3.7 показано графічну інтерпетацію розкладання функції f по змінній x_i та запис результуючого виразу. Нижні ребра вузла (вершини) графа (на рисунку помічені 0 і 1) зазвичай називаються вхідними, а верхнє ребро - вихідним.

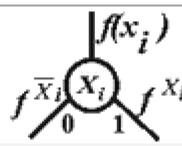
	$f(x_i) = \overline{x}_i\,(f^{\overline{x}_i}) \vee x_i \cdot (f^{x_i})$

Рисунок 3.7 – Формування результуючого виразу у вершині графа

Наприклад якщо $f^{\overline{x}_i} = x1x2$, а $f^{x_i} = x1 \vee x2$, то розкладання по x3 буде мати вигляд : $f(x3) = x1x2\,\overline{x}3 \vee x3(x1 \vee x2)$.

Правила отримання вихідних виразів у вершині графа.

1. Якщо на правому (прямому) і лівому (інверсному) ребрах знаходяться однакові значення функцій або однакові булеві вирази, то змінна x_i (за формулою розкладання) є несуттєвою і у вихідному виразі у цій вершині відсутня.

2. Якщо на одному з ребер стоїть 0, а на іншому булевий вираз f , то результуючий вираз представляє собою кон'юнкцію виразу f і змінної x_i з інверсією, якщо f знаходиться на лівому 0-ребрі, або без інверсії, якщо f знаходиться на правому 1-ребрі (рядки 6,7 у таблиці 3.1).

3. Якщо на одному з ребер стоїть 1, а на іншому булевий вираз f , то результуючий вираз представляє собою диз'юнкцію виразу f і змінної x_i з інверсією, якщо f знаходиться на правому 1-ребрі, або без інверсії, якщо f знаходиться на лівому 0 -ребрі (рядки 8,9 у таблиці 3.1).

У таблиці 3.1 представлені правила отримання вихідних виразів у вузлі графа і наведені різні значення вихідних виразів f (x_i) у вершині x_i залежно від значень виразів на вхідних ребрах.

Таблиця 3.1 – Правила отримання вихідних виразів у вузлі графа

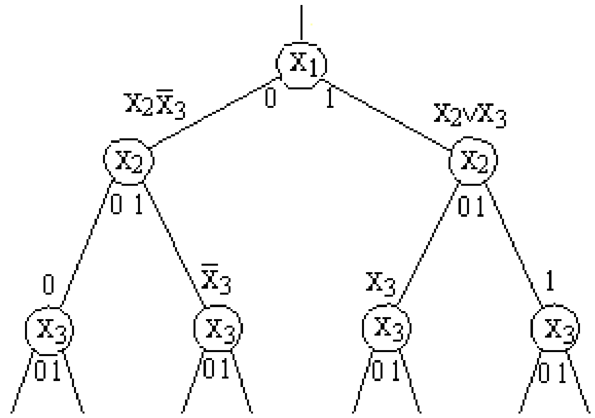
$f^{\bar{x}_i}$	f^{x_i}	$f(x_i)$
0	0	0
1	1	1
0	1	x_i
1	0	$\bar{x}_i$
f	f	f
f	0	$\bar{x}_i$ f
0	f	$x_i \cdot f$
f	1	$x_i \vee f$
1	f	$\bar{x}_i \vee f$

Граф-схема отримання скобкової форми для логічної функції n змінних складається з n ярусів. Яруси нумеруються знизу вгору числами від 1 до n, у своїй кожен ярус (ряд) відповідає змінної x_i, i = (1, n.) У кожному ряду міститься 2^{n-1} вершин. Число можливих шляхів від входів першого ряду до вершини граф-схеми дорівнює числу вхідних наборів розглянутої функції (2^n).

Граф-схема для функції f називається регулярною, якщо всередині кожного ряду всі вузли мають однакові номери змінних. Інакше граф-схема називається нерегулярною.

На рисунку 3.1 показана графічна інтерпретація отримання скобкової форми булевої функції трьох змінних f (2,5,6,7) = 1. Змінні в ярусах графа розташовуються в природному порядку (x1, x2, x3) від старших розрядів до молодших. У нижній частині рисунку в рамці показано таблицю істинності цієї функції (блакитному полі), а нижче показано відповідні номери наборів у десятковому еквіваленті. Двійковий еквівалент шляху (складений з 0 і 1, що помічають вхідні ребра вузлів графа) від вершини верхнього ярусу графа до відповідного значення функції (блакитне поле) вказує номер відповідного двійкового набору (порядок розрядів - x1, x2, x3). Наприклад, шлях 0 1 1 (через вершини x1, x2, x3) відповідає номеру набору 3, що підтверджується його десятковим еквівалентом (рис. 3.8).

$$У = x_1x_2 \vee x_1x_3 \vee \bar{x}_1x_2\bar{x}_3 = x_1x_3 \vee x_2\bar{x}_3$$



0	0	1	0	0	1	1	1
0	1	2	3	4	5	6	7

Рисунок 3.8 – Графова форма функції мультиплексора MX 2-в-1

Залежно від порядку розміщення змінних у ярусах графа можна отримати різні варіанти скобкових форм для однієї і тієї ж булевої функції.

На рисунку 3.9 показана графічна інтерпретація перестановки вершин у ярусах графа для функції f (2,5,6,7) = 1. На цьому рисунку переставлені змінні в другому і третьому ярусах графа (тепер порядок прямування розрядів у двійковому еквіваленті x1, x3, x2). Але, незважаючи на перестановку, вагомозначність розрядів у двійковому еквіваленті не змінюється, тобто x2 відповідає вазі 2^1, x1 - 2^2, x3 - 2^0. Таким чином, двійковий еквівалент шляху 001 (через вершини x1, x3, x2) відповідає набору з десятковим еквівалентом 2. Виходячи з цього можна сформулювати правило перестановки значення функції в таблиці істинності : для кожної вершини верхнього з ярусів, що переставляються, змінюються місцями між собою значення функцій в підграфах на "внутрішніх" ребрах вершин нижнього з ярусів , що переставляються. У прикладі при перестановці ярусів 2 і 3 змінюються місцями набори (1, 2) і (5, 6).

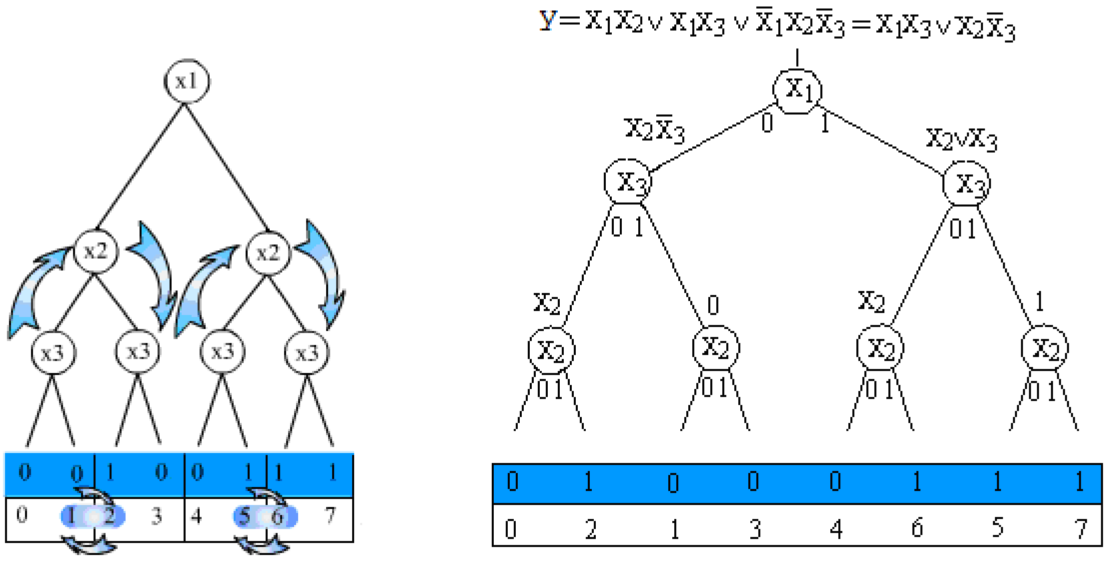


Рисунок 3.9 – Перестановка вершин у ярусах графа

Булева змінна x_i цілком визначеної функції f $(x_1,x_2, ..., x_n)$ суттєва, якщо є пара наборів, у яких зміна значення змінної x_i на протилежне (при постійних інших змінних) призводить до зміни значення функції.

Булева змінна x_i функції f $(x_1, x_2, ..., x_n)$ є суттєвою, якщо булева похідна даної функції змінної x_i дорівнює 1, тобто інші змінні функції мають такі значення, щоб похідна дорівнювала 1.

Булева змінна x_i суттєва на двійковому наборі $(x_1, x_2, ..., x_n)$, якщо зміна її значення протилежне призведе до зміни на протилежне значення функції f цьому наборі.

Суттєвість змінної x_i на двійковому наборі $(x_1, x_2, ..., x_n)$ визначається значенням булевої похідної на цьому наборі. Якщо булева похідна змінної x_i

на двійковому наборі (x_1,x_2, ..., x_n) приймає значення 1, то дана змінна суттєва цьому наборі. Інакше змінна x_i несуттєва цьому наборі.

Група булевих змінних x_i...x_j суттєва на двійковому наборі (x1, x2, ...x_n), якщо одночасна зміна всіх значень зазначених змінних на протилежні призведе до зміни значення функції f. Групова суттєвість змінних визначається кратною булевою похідною за змінними x_i...x_j.

Аналітичне взяття булевої похідної визначається формулою

$$f_3 = \frac{dy}{dx_3} = (x_1 \cdot 0 \vee x_2 \cdot 1) \oplus (x_1 \cdot 1 \vee x_2 \cdot 0) = x_2 \oplus x_1 = x_1\overline{x_2} \vee \overline{x_1}x_2 \; .$$

Графічне взяття булевої похідної для змінної нижнього яруса графа визначається порівнянням парних та непарних позицій у переставленій таблиці істинності для. Якщо відповідна змінна стоїть у нижньому ярусі графа. Якщо парні та непарні позиції у таблиці істинності розрізняються (0 1 або 1 0), то відповідна гілка дерева впливає на булеву змінну (на рисунку помічається "+"). Далі на другому ярусі відповідна змінна записується з інверсією, якщо "+" іде з лівого ребра, та без інверсії, якщо з правого. Далі застосовуються загальні правила отримання вихідних виразів у вузлі графа.

Розглянемо аналітичне і графічне взяття булевих змінних для функції $y(x_1,x_2,x_3) = (x_1x_3 \vee x_2\overline{x_3})$.

$$f_1 = \frac{dy}{dx_1} = (0 \cdot x_3 \vee x_2\overline{x_3}) \oplus (1 \cdot x_3 \vee x_2\overline{x_3}) = x_2\overline{x_3} \oplus (x_2 \vee x_3) =$$

$$= x_2\overline{x_3} \cdot \overline{(x_2 \vee x_3)} \vee \overline{x_2\overline{x_3}} \cdot (x_2 \vee x_3) = x_2\overline{x_3} \cdot \overline{x_2}\,\overline{x_3} \vee (\overline{x_2} \vee x_3) \cdot (x_2 \vee x_3) = 0 \vee x_3 = x_3$$

$$f_2 = \frac{dy}{dx_2} = (x_1x_3 \vee 0 \cdot \overline{x_3}) \oplus (x_1x_3 \vee 1 \cdot \overline{x_3}) = x_1x_3 \oplus (x_1x_3 \vee \overline{x_3}) = x_1x_3 \oplus (x_1 \vee \overline{x_3}) =$$

$$= x_1x_3 \cdot \overline{(x_1 \vee \overline{x_3})} \vee \overline{x_1x_3} \cdot (x_1 \vee \overline{x_3}) = x_1x_3 \cdot \overline{x_1}x_3 \vee (\overline{x_1} \vee \overline{x_3}) \cdot (x_1 \vee \overline{x_3}) = 0 \vee \overline{x_3} = \overline{x_3}$$

$$f_3 = \frac{dy}{dx_3} = (x_1 \cdot 0 \vee x_2 \cdot 1) \oplus (x_1 \cdot 1 \vee x_2 \cdot 0) = x_2 \oplus x_1 = x_1\overline{x_2} \vee \overline{x_1}x_2$$

Графічне взяття булевих похідних наведено на рис. 3.10.

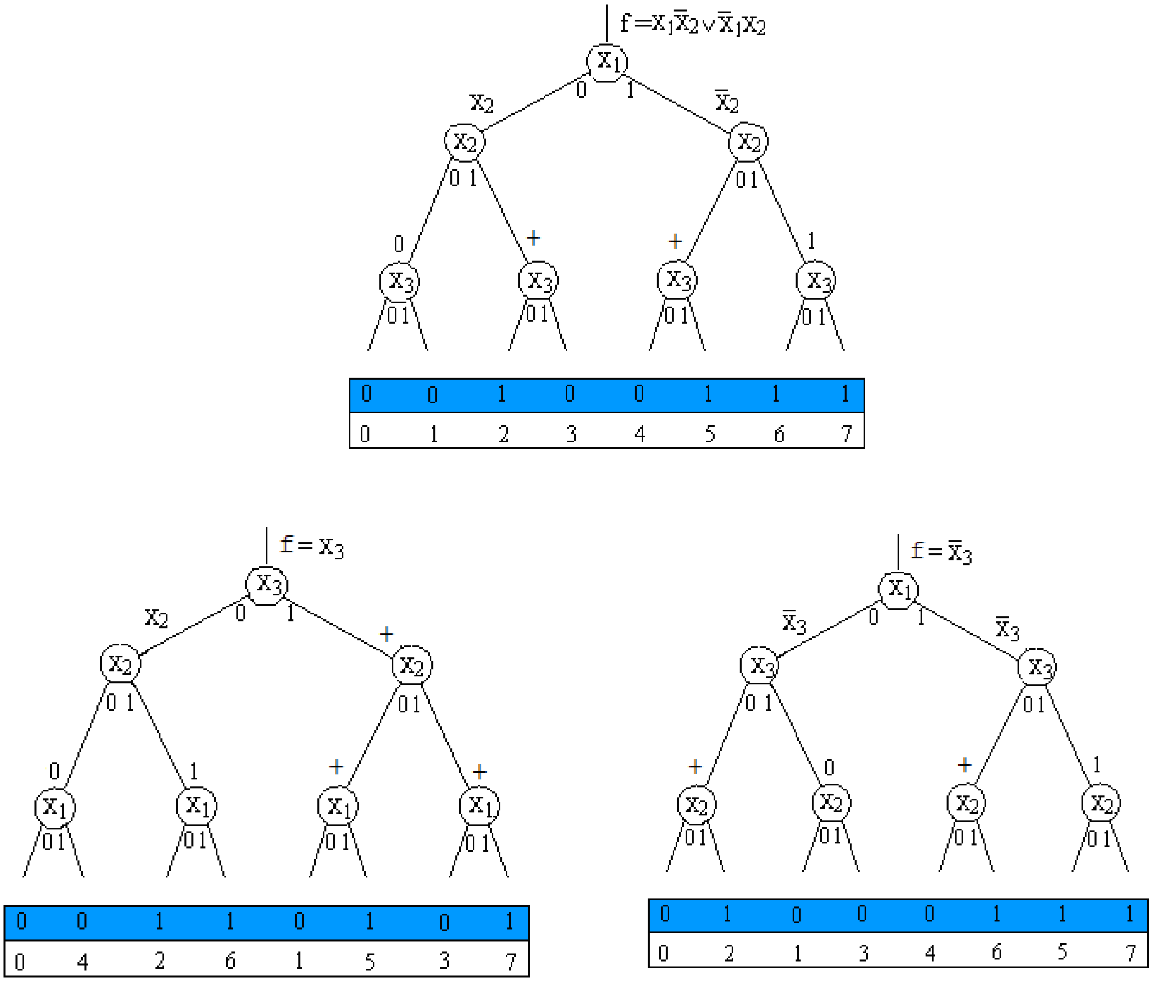


Рисунок 3.10 – Графічне взяття булевих похідних

Для трьох змінних отримано чотири умови активізації, що відповідають чотирьом логічним шляхам у схемній структурі диз'юнктивної форми даної функції. Результати отримання булевих похідних для графічного та аналітичного способу представлення булевих функцій повністю співпадає, що підтверджує ефективність застосування запропонованих методів.

Таким чином, взяття булевої похідної фактично складається з перестановки розрядів таблиці істинності (Q-вектора) відповідно до обраного суттєвого входу, а застосування формули (2.4) дозволяє застосовувати переставлений Q-вектор для дедуктивного моделювання несправностей.

3.3 Булеві похідні для кубітного представлення логічних функцій

Розглянемо спосіб взяття булевих похідних по кубитному покриттю для визначення умов суттєвості вхідних змінних при моделюванні несправностей. Покажемо аналогію між двома формами булевих функцій для взяття похідних: аналітичної та векторної.

Визначити всі похідні першого порядку за кубітною формою логічної функції $f(x) = x_1 \vee x_1 \bar{x}_2$. Розглянемо перетворення для кубітного покриття функції, заданого вектором:

$$\begin{array}{ccc} X_1 & X_2 & Y \\ 0 & 0 & 0 \\ 0 & 1 & 0 = (0011) \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}.$$

Похідну за таблицею істинності можна обчислювати за допомогою послідовного завдання всіх нулів і одиниць у координатах стовпців, що відповідають кожній змінній (рис. 3.11). Тут позначення $Y_i^0 = Y($при $X_i=0)$, $Y_i^1 = Y($при $X_i=1)$, $Y_i' = Y_i^0 \oplus Y_i^1$

X_1	X_2	Y	Y_1^0	Y_1^1	Y_1'	Y_2^0	Y_2^1	Y_2'
0	0	0	0	1	1	0	0	0
0	1	0	0	1	1	0	0	0
1	0	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1	0

Рисунок 3.11 – Кубітне взяття булевих похідних

Таким чином, похідні за першою та другою змінною, записані у форматі кубитного покриття, дорівнюють 1111 і 0000. Це означає, що похідна за першою змінною дорівнює одиниці, а за другою – нулю. Однак такий результат можна отримати більш формально та технологічно, не розглядаючи вхідні набори таблиці істинності, використовуючи лише логічні операції зустрічного зсуву та подальшої xor-операції над розрядами кубитного покриття {a,b} = a ⊕ b, де a,b – сусідні підвектори кубіту Q=(a,b):

Y	Y_1'	Y_2'
0	1	0
0	1	0
1	1	0
1	1	0

Інакше, для першої змінної необхідно xor-скласти зрушені один щодо одного дві половинки першого стовпця, а результат записати в обидві симетричні області, що зрушуються: {a,b} = a ⊕ b, (11,11)=00⊕11. Для другої змінної слід розглядати пари сусідніх координат стовпця, а загальний результат записувати в кожні симетричні області-біти, що зсуваються: {a,b}=a ⊕ b: (0,0)=0 ⊕ 0=0, (0,0)= 1 ⊕ 1=0. Таким чином, результат підсумовування буде загальним для кожної пари підвекторів, що взаємодіють, розмірність яких визначається номером аналізованої змінної від нуля до .

У загальному випадку формулу обчислення булевих похідних для формування відповідної кубитної матриці black box функціональності можна представити як xor операцію над розрядами сусідніх груп бітів:

$$D_{(ij,\ ij+1)}(Q, X_i) = Q_j(X_i) \bigoplus_{j=1}^{2^{n-i}} Q_{j+1}(X_i),\ \ i = \overline{1, n}.$$

Тут n – кількість змінних у black box функціональності; i – номер змінної, за якою береться похідна; (j, j+1) – номери сусідніх груп у кубітному векторі, які підлягають xor-порівнянню, де число та потужність таких груп функціонально залежить від номера змінної.

$D_{(ij,\ ij+1)}(Q, X_i)$ –сусідні групи похідної по і-й змінній, що формуються xor-операцією. Під групою розуміється сукупність бітів, кратна ступеня двійки, що підлягає порозрядному xor-складання з відповідними бітами сусідньої групи кубітного вектора.

Тепер обчислимо всі похідні першого порядку за кубітною формою логічної функції мультиплексора MX 2-в-1 $y(x_1,x_2,x_3)=(x_1x_3 \vee x_2\overline{x_3})$ Обчислення трьох похідних першого порядку за таблицею істинності дає такий результат (рис. 3.12).

X_1	X_2	X_3	Y	Y_1^0	Y_1^1	Y_1'	Y_2^0	Y_2^1	Y_2'	Y_3^0	Y_3^1	Y_3'
0	0	0	0	0	0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0	0	0	0	0	0
0	1	0	1	1	1	0	0	1	1	1	0	1
0	1	1	0	0	1	1	0	0	0	1	0	1
1	0	0	0	0	0	0	0	1	1	0	1	1
1	0	1	1	0	1	1	1	0	1	0	1	1
1	1	0	1	1	1	0	0	1	1	1	1	0
1	1	1	1	0	1	1	1	0	1	1	1	0

Рисунок 3.12 – Кубітне взяття булевих похідних для MX 2-в-1

Якщо виключити з розгляду таблицю істинності, а використовувати кубітне покриття, то на змістовному рівні процес обчислення похідних матиме такий вигляд:

Y	Y_1'	Y_2'	Y_3'
0	0	1	0
0	1	0	0
1	0	1	1
0	1	0	1
0	0	1	1
1	1	0	1
1	0	1	0
1	1	0	0

Для отримання похідної по першій змінній необхідно xor-скласти зсунуті назустріч один одному два сусідні тетради кубит-вектора Y, а результат записати в обидві тетради: {a,b}=a⊕b, (0110,0110)=0101⊕0011 .

Для отримання похідної за другою змінною слід послідовно xor-скласти сусідні пари кубит-вектора Y а загальний результат записувати в кожну пару: {a,b}=a⊕b: (00,00)=01⊕01, (11, 11) = 00⊕11. Для отримання похідної по третій змінній слід послідовно xor-скласти сусідні біти кубит-вектора Y а загальний результат записати в кожен сусідній біт: {a,b} = a⊕b: (1,1) = 0⊕1, (1 ,1) = 0⊕1, (0,0) = 0⊕0, (0,0) = 0⊕0. Природно, що кубит-похідна за будь-якою вхідною змінною як вектор має відносну симетрію рівності підвекторів по побудові: похідна третьої змінної має симетричну рівність двох тетрад, похідна другої змінної має симетричну рівність кожних сусідніх пар, похідна першої змінної має симетричну рівність кожних сусідніх бітів.

Таким чином $Y_1'(01010101) = X_3$, $Y_2'(10101010) = \overline{X}_3$, $Y_3'(00111100) = X_1 \oplus X_2$. Результати отримання булевих похідних для кубічного, графічного та аналітичного способу представлення булевих функцій на прикладі MX 2-в-1 повністю співпадають, що підтверджує працездатність та ефективність застосування запропонованих методів.

3.4 Метод моделювання несправностей по дедуктивним Q-векторам

Математична основа дедуктивного моделювання несправностей полягає в транспортування бінарних комбінацій вхідних несправностей на вихід на заданому вхідному набору за формулою L = T ⊕F. Дедуктивне моделювання полягає у зміні логіки елемента F залежно від вхідних умов T. Дедуктивне моделювання є найефективнішим засобом аналізу якості тестів та синтезу таблиць несправностей для пошуку дефектів, простежування шляху поширення несправності. Пропонується його реалізація на основі векторної форми опису логіки що виключає логічні аналітичні форми, що дає можливість істотно спростити алгоритми синтезу дедуктивних моделей та їх застосування для інтерпретативного моделювання цифрових елементів та схем великої розмірності.

Суть методу моделирования неисправностей по дедуктивним Q-векторам, який запропонований проф.Хахановим [10, 11. 12], полягає в побудові дедуктивних векторів, які є впорядкованої сукупності векторних похідних по вхідним змінним, що визначаються стовпцями матриці. Сукупність похідних по вхідним змінним у матриці представляє собою тест для перевірки одиночних константних несправностей вхідних та вихідних ліній. Одиничні координати дедуктивних векторів є умовами активізації вектора вхідних даних як адреси для транспортування на вихід схеми. Будується дедуктивний вектор шляхом перестановки бітів Q-вектора та операції XOR між вхідним набором для функціонального елемента та його таблицею істинності (Q-вектором).

Розглянемо метод синтезу дедуктивних векторів за Q-вектором на прикладі мультиплексору MX 2-в-1.

1. Дано таблицю істинності (TI) n змінних у порядку вагомозначності позиційного коду вхідних наборів (від 000 ... 0 до 2^{n-1}). Представимо вихідний стовпець TI як Q-вектор довжиною 2^n . Для MX 2-в-1 Q = (00100111).

2. Отримання матриці L-векторів (взяття булевих похідних за i-ю змінною). Для кожного розряду Q_i обчислюється $L_i = Q \oplus Q_i$, де Q_i – стан i-біта Q-вектора. Для цього складається матриця (2^{n+1} x 2^{n+1}) у перший рядок і перший стовпець якої записується Q-вектор, за умови, що нульова комірка порожня (жовті рядок і стовпець на рис. 3.13 а)).

3. У кожен i-й рядок матриці L записується вектор Q без інверсії, якщо в стовпці Q_i=0 та з інверсією, якщо Q_i=1 (рис. 13 а)).

4. Виконується формування дедуктивної матриці D (2^n x 2^n, рис. 3.13 в)) шляхом перестановок елементів рядків матриці L за правилами матриці H (2^n x 2^n, рис. 3.13 б)). Це фактично відповідає встановленню по черзі відповідних змінних у молодший розряд і перестановці відповідно до цього елементів таблиці істинності (Q-вектора).

L-матрица

	0	0	1	0	0	1	1	1
0	0	0	1	0	0	1	1	1
0	0	0	1	0	0	1	1	1
1	1	1	0	1	1	0	0	0
0	0	0	1	0	0	1	1	1
0	0	0	1	0	0	1	1	1
1	1	1	0	1	1	0	0	0
1	1	1	0	1	1	0	0	0
1	1	1	0	1	1	0	0	0

а)

H-матрица

0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0

б)

D-матрица

0	0	1	0	0	1	1	1
0	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0
0	1	0	0	1	1	1	0
0	1	1	1	0	0	0	0
0	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1
0	0	0	1	1	0	1	1

в)

Рисунок 3.13 – Формування дедуктивних векторів для MX 2-в-1

Запропонована інша процедура перестановок Q-вектора для отримання D-матриці без використання H-матриці, по аналогії з графовим способом отримання булевих похідних. Принцип перестановок наступний: аналізуються розряди вхідного набора починаючи зі старшого і для тих розрядів, вхідного набору, які дорівнюють 1 виконується перестановка елементів Q-вектора групами по $2^{(n-1)}$ розрядів, де n-номер розряду, при умові, що молодший розряд (правий) вважається першим. Після всіх перестановок елементи отриманого дедуктивного векторам інвертуються, якщо значення Q-вектора на цьому наборі дорівнює 1 (у відповідності до формули 2.4).

Розглянемо приклад перестановок Q-вектора 5 змінних (n=5) для вхідного набору (11010). Початковий Q-вектор представлений у табл. 3.2 за природним порядком змінних “down to” (другий рядок у таблиці є продовженням першої).

Таблиця 3.2 – Початковий Q-вектор

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Починаємо аналіз зі старших розрядів вхідного набору і переставляємо групи елементів у Q-векторі для тих розрядів, які визначаються одиницями у вхідному наборі. Для першого розряду **1**1010 переставляємо групи по $2^{(n-1)}$, тобто. по 16 елементів (табл. 3.3).

Таблиця 3.3 – Перша перестановка

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Для другого розряду 1**1**010 переставляємо групи по $2^{(n-2)}$, тобто по 8 елементів (табл. 3.4).

Таблиця 3.4 – Друга перестановка

24	25	26	27	28	29	30	31	16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7

Для четвертого розряду 110**1**0 переставляємо групи по $2^{(n-4)}$, тобто по 2 елементи (табл. 3.5).

Таблиця 3.5 – Третя перестановка

26	27	24	25	30	31	28	29	18	19	16	17	22	23	20	21
10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5

Ця процедура не залежить від кількості змінних, а залежить виключно від числа 1 у вхідному наборі і повністю співпадає з перестановками у графовому поданні булевих функцій [8] та з ітеративною процедурою у [10].

Певний інтерес представляє результат перестановок для вхідного набору 11111 (табл. 3.6).

Таблиця 3.6 – Перестановка для набору 11111

31	30	29	28	27	26	26	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Процедура моделювання несправностей по дедуктивним векторам складається з наступних кроків.

1. Існує шаблон обчислення адрес у дедуктивному векторі для моделювання вхідних несправностей. Число стовпців дорівнює подвоєному числу несправностей, спочатку розташовуються нульові несправності, а потім одиночні. Кожен рядок має вагу, кратну ступені двійки, причому X1 (перший рядок) відповідає молодшому розряду. Спочатку шаблон та вектор моделювання (L-вектор) заповнюються нулями (або мають порожні комірки).

2. Вхідні несправності мають інверсні значення від справних значень сигналів на входах ПЕ (вхідному набору).

3. У комірках, що відповідають вхідним несправностям у шаблоні ставиться 1.

4. По стовпцям, де стоять 1, обчислюється адреса (індекс) координати дедуктивного вектора, з урахуванням ваги рядків (десятковий еквівалент двійкового числа стовпця шаблону 1. 2. 4. 8. … тощо).

5. У вектор моделювання (L) у комірки, що відповідають стовпцям шаблону з 1, записуються значення розрядів відповідного дедуктивного вектора за обчисленими адресами (індексами). Відмітимо, що вихідні несправності ПЕ не моделюються і відповідні (вихідні) позиції L-вектора не обчислюються. Слід враховувати, що дедуктивний вектор для одиничних значень вихідного сигналу на цьому наборі інвертується.

6. Якщо за одиничними значеннями вектора моделювання у шаблоні стоїть 1, то несправність, що визначає стовпець шаблону, перевіряється.

Розглянемо приклади моделювання несправностей по дедуктивним векторам для ПЕ MX 2-в-1 для декількох вхідних наборів.

Набор (0 0 0). Початковий список несправностей $\{1^1, 2^1, 3^1\}$. Дедуктивный вектор $(0\ \underline{0}\ \underline{1}\ 0\ \underline{0}\ 1\ 1\ 1)$. Результат моделювання несправностей на рис. 3.14.

	3^0	2^0	1^0	3^1	2^1	1^1
X1				1		
X2					1	
X3						1
Адреси стовпців початкових несправностей						
				1	2	4

	0	1	2	3	4	5	6	7
					1	1	1	1
			1	1			1	1
		1		1		1		1
L	0	0	0	0	1	0	0	0

Це правильний результат

3^0	2^0	1^0	3^1	2^1	1^1
			0	1	0

Результат $\{2^1, 7^1\}$

Рисунок 3.14 – Моделювання несправностей для набору 0 0 0 ПЕ МХ 2-в-1

Набор (1 1 1). Початковий список несправностей $\{1^0, 2^0, 3^0\}$. Дедуктивный вектор (0 $\underline{0}$ $\underline{0}$ 1 $\underline{1}$ 0 1 1). Результат – рис. 3.15.

	3^0	2^0	1^0	3^1	2^1	1^1
X1	1					
X2		1				
X3			1			

	0	1	2	3	4	5	6	7
					1	1	1	1
			1	1			1	1
		1		1		1		1
L	0	0	1	0	0	0	0	0

Це правильний результат

3^0	2^0	1^0	3^1	2^1	1^1
0	0	1			

Рисунок 3.15 – Моделювання несправностей для набору (1 1 1) ПЕ МХ 2-в-1

Розглянемо, як впливає послідовність змінних у вхідному наборі на результат моделювання по дедуктивним Q-векторам. Зазначимо, що в мовах опису апаратури використовуються два способи опису векторних змінних:

– std_logic_vector (3 downto 0); – молодший розряд правий;

– std_logic_vector (3 to 0); – молодший розряд лівий.

Для прикладу розглянемо моделювання симетричного набору (0 1 0). Початковий список несправностей $\{1^1, 2^0, 3^1\}$. Послідовність змінних у вхідному наборі впливає тільки на формування шаблону обчислення адрес. Дедуктивный вектор (0 $\underline{1}$ $\underline{1}$ 1 $\underline{0}$ 0 1 0).

Спочатку розглянемо шаблон для "downto". Тут молодший розряд правий (третій за порядком), що відповідає загальноприйнятій системі позиційного коду. Результати моделювання несправностей наведені на рис. 3.16.

	3^0	2^0	1^0	3^1	2^1	1^1
X1						1
X2		1				
X3				1		

	0	1	2	3	4	5	6	7
					1	1	1	1
			1	1			1	1
		1		1		1		1
L	0	1	0	1	0	0	0	0

Це правильний результат

3^0	2^0	1^0	3^1	2^1	1^1
	1		1		0

Рисунок 3.16 – Моделювання несправностей набору (0 1 0) для шаблону ‟downto″

Далі розглянемо шаблон для ‟to″. Тут молодший розряд лівий (перший за порядком), що не відповідає загальноприйнятій системі позиційного коду. Результати моделювання несправностей наведені на рис. 3.17.

	1^0	2^0	3^0	1^1	2^1	3^1
X1				1		
X2		1				
X3						1

	0	1	2	3	4	5	6	7
					1	1	1	1
			1	1			1	1
		1		1		1		1
L	0	1	0	1	0	0	1	0

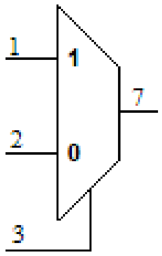
Це результат с точністю до навпаки

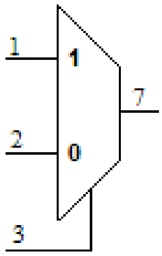
1^0	2^0	3^0	1^1	2^1	3^1
	1		1		0

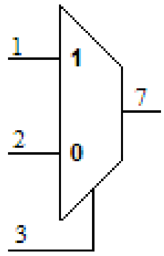
Рисунок 3.17 – Моделювання несправностей набору (0 1 0) для шаблону ‟to″

Таким чином результати моделювання для шаблону ‟downto″ повністю співпадають з результатами простого дедуктивного моделювання (рис. 2.5) і не співпадають для шаблону ‟to″. Таким чином, при використанні позиційного коду номерів розрядів ПЕ рекомендується користуватися шаблоном обчислення адрес ‟downto″.

Результати моделювання представимо зведеною таблицею (рис. 3.18). При цьому слід мати на увазі, що вихідні несправності ПЕ просто підсумуються до списку у відповідності до формули (2.2). Результати повністю співпадають з іншими способами дедуктивного моделювання.



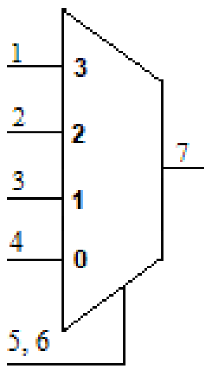




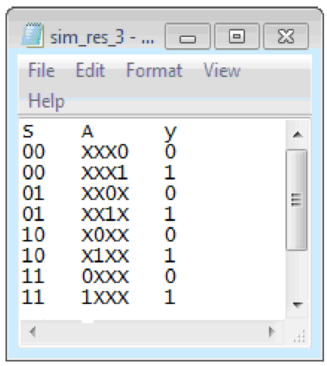
1 2 3 7 0 0 0 0 Вхідні списки$\{1^1, 2^1, 3^1\}$ набор (адреса) 000 Дедуктивний вектор 0 0 1 0 0 1 1 1 Результат $\{2^1, 7^1\}$	1 2 3 7 1 1 1 1 Вхідні списки$\{1^0, 2^0, 3^0\}$ набор(адреса) 111 Дедуктивный вектор 0 0 0 1 1 0 1 1 Результат $\{1^0 ,7^0\}$	1 2 3 7 0 1 0 1 Вхідні списки$\{1^1, 2^0, 3^1\}$ набор(адрес) 010 Дедуктивный вектор 0 1 1 1 0 0 1 0 Результат $\{2^0, 3^1 ,7^0\}$

Рисунок 3.18 – Результати моделювання несправностей ПЕ МХ 2-в-1 по дедуктивним Q-векторам

Розглянемо приклади моделювання несправностей по дедуктивним векторам для більш складного ПЕ МХ 4-в-1. На рис. 3.19 представлено його позначення, закон функціонування та TI, отримані в системі Xilinx. Останній стовпець на правому рисунку це фактично Q-вектор.



1	2	3	4	5	6	7
X	X	X	0	0	0	0
X	X	X	1	0	0	1
X	X	0	X	0	1	0
X	X	1	X	0	1	1
X	0	X	X	1	0	0
X	1	X	X	1	0	1
0	X	X	X	1	1	0
1	X	X	X	1	1	1



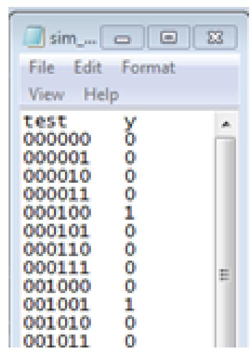


Рисунок 3.19 – Позначення та закон функціонування ПЕ МХ 4-в-1

Повна таблиця істинності для МХ 4-в-1 ($2^6 = 64$) набора, розбита на 4 частини по16 наборів, представлена на рис. 3.20. Останній стовпець в таблиці це Q-вектор, він же переставлений D-вектор, помічений кольором. Зелений колір – для набора (0 0 0 0 0 0), а синій – для набора (1 1 1 1 1 1).

1	2	3	4	5	6	7
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	0
0	0	0	1	0	0	1
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	0
0	0	1	0	0	0	0
0	0	1	0	0	1	1
0	0	1	0	1	0	0
0	0	1	0	1	1	0
0	0	1	1	0	0	1
0	0	1	1	0	1	1
0	0	1	1	1	0	0
0	0	1	1	1	1	0

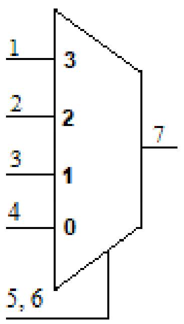
1	2	3	4	5	6	7
0	1	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	0	1	1	0
0	1	0	1	0	0	1
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	1	1	0
0	1	1	0	0	0	0
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	1
0	1	1	1	0	1	1
0	1	1	1	1	0	1
0	1	1	1	1	1	0

1	2	3	4	5	6	7
1	0	0	0	0	0	0
1	0	0	0	0	1	0
1	0	0	0	1	0	0
1	0	0	0	1	1	1
1	0	0	1	0	0	1
1	0	0	1	0	1	0
1	0	0	1	1	0	0
1	0	0	1	1	1	1
1	0	1	0	0	0	0
1	0	1	0	0	1	1
1	0	1	0	1	0	0
1	0	1	0	1	1	1
1	0	1	1	0	0	1
1	0	1	1	0	1	1
1	0	1	1	1	0	0
1	0	1	1	1	1	1

1	2	3	4	5	6	7
1	1	0	0	0	0	0
1	1	0	0	0	1	0
1	1	0	0	1	0	1
1	1	0	0	1	1	1
1	1	0	1	0	0	1
1	1	0	1	0	1	0
1	1	0	1	1	0	1
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	0	0	1	1
1	1	1	0	1	0	1
1	1	1	0	1	1	1
1	1	1	1	0	0	1
1	1	1	1	0	1	1
1	1	1	1	1	0	1
1	1	1	1	1	1	1

Рисунок 3.20 – Повна таблиця істинності для МХ 4-в-1

На рис.3.21 наведені результати моделювання несправностей для ПЕ МХ 4-в-1 по дедуктивним Q-векторам на наборах (0 0 0 0 0 0) та (1 1 1 1 1 1).



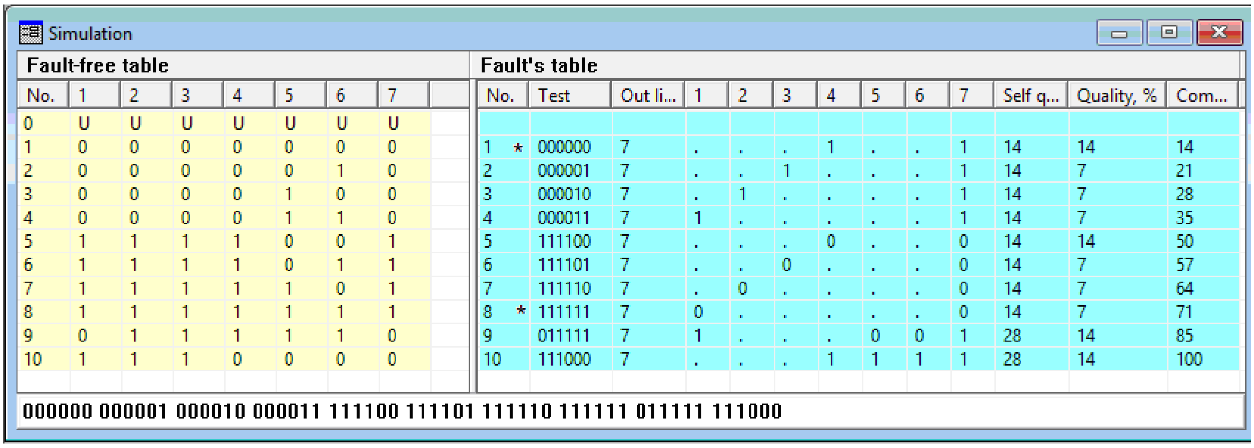
Набор	0	0	0	0	0	0	
Несправності	6^1	5^1	4^1	3^1	2^1	1^1	4^1
Позиції в D-векторі	1	2	4	8	16	32	
Значення в D-векторі	0	0	1	0	0	0	
Набор	1	1	1	1	1	1	
Несправності	6^0	5^0	4^0	3^0	2^0	1^0	1^0
Позиції в D-векторі	62	61	59	55	47	31	
Значення в D-векторі. (інв)	0	0	0	0	0	1	

Рисунок 3.21 – Результати моделювання несправностей ПЕ МХ 4-в-1 по дедуктивним Q-векторам

Таким чином для набора (0 0 0 0 0 0) результат моделювання { 4^1, 7^1}, а для набора (1 1 1 1 1 1) результат моделювання { 1^0, 7^0}.

Перевіримо отримані результати шляхом моделювання несправностей на програмі DCP (кубічне моделювання, рис. 3.22) повного тесту 000000

000001 000010 000011 111100 111101 111110 111111 011111 111000, в якому присутні зазначені набори. Результат повністю співпадає, що підтверджує працездатність та ефективність методу отримання та перестановки Q-векторів і застосування їх в дедуктивному адресному моделюванні несправностей.



Simulation

Fault-free table

No.	1	2	3	4	5	6	7
0	U	U	U	U	U	U	U
1	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0
3	0	0	0	0	1	0	0
4	0	0	0	0	1	1	0
5	1	1	1	1	0	0	1
6	1	1	1	1	0	1	1
7	1	1	1	1	1	0	1
8	1	1	1	1	1	1	1
9	0	1	1	1	1	1	0
10	1	1	1	0	0	0	0

Fault's table

No.	Test	Out li...	1	2	3	4	5	6	7	Self q...	Quality, %	Com...
1 *	000000	7	.	.	.	1	.	.	1	14	14	14
2	000001	7	.	.	1	.	.	.	1	14	7	21
3	000010	7	.	1	.	.	.	.	1	14	7	28
4	000011	7	1	.	.	.	.	.	1	14	7	35
5	111100	7	.	.	.	0	.	.	0	14	14	50
6	111101	7	.	.	0	.	.	.	0	14	7	57
7	111110	7	.	0	.	.	.	.	0	14	7	64
8 *	111111	7	0	.	.	.	.	.	0	14	7	71
9	011111	7	1	.	.	.	0	0	1	28	14	85
10	111000	7	.	.	.	1	1	1	1	28	14	100

000000 000001 000010 000011 111100 111101 111110 111111 011111 111000

Рисунок 3.22 – Результати моделювання програмою DCP повного тесту для ПЕ MX 4-в-1

Розглянемо співвідношення витрат пам’яті для кубічного моделювання несправностей та моделювання по дедуктивним Q-векторам

Для кодування символів алфавіту A^2 (16 символів) використовується кубітне кодування по 4 біта (полубайт) на кожен символ. Таким чином для MX 4-в-1 розмір КП буде 56 полубайтових комірок (56x4 = 224 біти), Q-вектор – 64 біт, тобто Q-вектор займає менше пам’яті. Якщо умовно взяти n = 8 (MX 5-в-1), то КП буде 90 полубайтових комірок (360 біт), а Q-вектор – 256 біт, що близько по витратам пам’яті одне до одного.

Тобто при n > 8 (а в середньому таке співвідношення буде для всіх регулярних КП основних логічних елементів та мультиплексорів) пам'ять для зберігання Q-вектора перевищуватиме пам'ять для зберігання КП. І чим більше n тим більша різниця не на користь кубітних Q-векторів. Тобто для ПЕ з числом входів n > 8 використовувати Q-вектора недоцільно виходячи з витрат пам’яті.

І що далі? А далі лише декомпозиція. Наведемо приклад декомпозиції МХ 4-в-1 на ПЕ МХ 2-в-1 та моделювання несправностей декомпозованої структури. На рис. 3.23 наведена декомпозована структура МХ 4-в-1 через МХ 2-в-1 та результати моделювання несправностей на наборі (0 0 0 0 0 0) з урахуванням результатів, отриманих при моделюванні набора (0 0 0) по дедуктивним векторам для МХ 2-в-1 (рис.3.14).

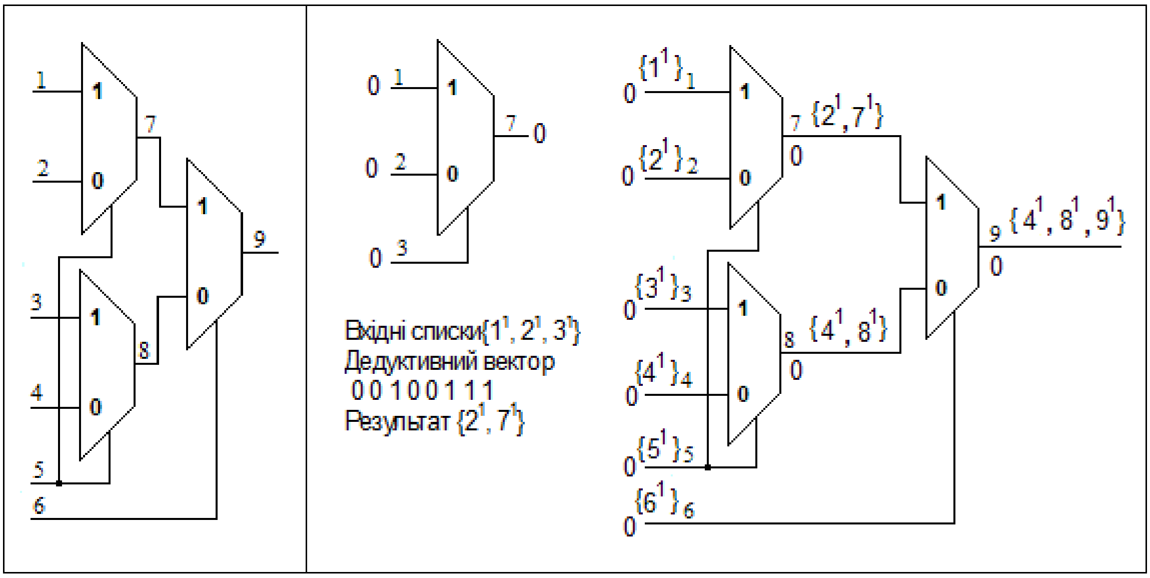


Рисунок 3.23 – Результати моделювання МХ 4-в-1 після декомпозиції

Отримані результати повністю співпадають з результатами кубічного моделювання зазначеного набора програмою DCP (рис. 2.9). Це повністю підтверджує працездатність методу моделювання несправностей по дедуктивним Q-векторам за результатами декомпозиції.

ВИСНОВКИ

В ході виконання роботи розглянуті різні стилі опису цифрових пристроїв мовами опису апаратури. Показана еквівалентність паралельних та послідовних умовних операторів а також їх схемна реалізація у вигляді мультиплексорів. Запропонований спосіб отримання таблиць істинності синтезованої схемної структури за допомогою TestBench Xilinx ISE на основі опису цифрової схеми мовами опису апаратури.

Розглянуті різні технології та структури даних дедуктивного моделювання несправностей для табличного та аналітичного способу опису цифрових схем. Розглянута програмна реалізація кубічного дедуктивного моделювання несправностей та показна еквівалентність отриманих результатів для схем мультиплексорів MX 2-в-1 та MX 4-в-1 з використанням програмного продукту DCP.

Отримали подальший розвиток методи побудови кубитних моделей опису структури цифрових систем і функцій компонентів, які характеризуються компактністю опису таблиць істинності у формі Q-покриттів, що дозволяє підвищити швидкодію програмних та апаратних засобів для справного та несправного моделювання, завдяки адресній реалізації аналізу кубітних векторів. Показана можливість використання отриманих кубітних структур даних в Q-методі дедуктивного адресного моделювання несправностей цифрових схем, який характеризується використанням компактних Q-покриттів замість таблиць істинності компонентів.

Отримані результати дедуктивного моделювання з використанням програмного продукту DCP для різних структур даних схем мультиплексорів MX 2-в-1 та MX 4-в-1 повністю співпадають, що підтверджує працездатність різних моделей даних для дедуктивного моделювання несправностей, отриманих на основі описів цифрових схем мовами опису апаратури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бибило П.Н. Синтез логических схем с использованием языка VHDL / П.Н. Бибило – М.: СОЛОН-Р, 2009. – 384 с.

2. Armstrong D. B. A deductive method of simulating faults in logic circuits // IEEE Trans. on Computers. – 1972. – Vol. C-21, – N 5. – P. 464-471.

3. Шкиль А.С. Дедуктивный метод кубического моделирования неисправностей цифровых устройств/ В.И. Хаханов, А.С. Шкиль, В.В Ханько // Радиоэлектроника и информатика. – 1999. – № 1(6). – С. 77-84.

4. Хаханов В.И. Кубическое моделирование неисправностей цифровых проектов на основе FPGA, CPLD / В.И. Хаханов, Е.В. Ковалев, Хак Х.М. Джахирул, Масуд М.Д. Мехеди // Радиоэлектроника и информатика. – 1999. – № 4. – С. 64-71.

5. Хаханов В.И. HFS-процессор аппаратного моделирования неисправностей цифровых проектов / В.И. Хаханов, Хассан Ктейман, А.Н. Парфентий, И.В. Хаханова // АСУ и приборы автоматики. –2007. – № 1(134). – С. 93-108.

6. Чумаченко С.В. Кубитная форма описания вычислительных структур / С.В. Чумаченко, T. Bani Amer, И.В. Емельянов// Радиоэлектроника и информатика. – 2016. – № 1. – С. 47-52.

7. Хаханов В. И. Кубитные структуры данных вычислительных устройств / В.И. Хаханов, Багхдади Аммар Авни Аббас, Е.И. Литвинова, А.С. Шкиль // Электронное моделирование. – 2015. – Т. 37, № 1. – С. 49-76.

8. Шкиль А.С. Автоматизация получения булевых разностей / Г.Ф. Кривуля, А.С. Шкиль // АСУ и приборы автоматики. – 1981.– вып.59. – С. 73-78.

9. Хаханов В.И. Кубитный метод дедуктивного анализа неисправностей для логических схем / В.И. Хаханов, И.В. Емельянов, М.М Любарский, С.В. Чумаченко, Е.И. Литвинова, Тамер Бани Амер //

Электронное моделирование. – 2017. – Т. 39, № 6. – С. 59-91.

10. Hahanov V., Vector-deductive Memory-based Transactions for Fault-as-address Simulation / W. Gharibi, A. Hahanova, V. Hahanov, S. Chumachenko, E. Litvinova, I. Hahanov // Èlektronic modeling. – 2023. – Т. 45, № 1. – С. 3-26.

11. Gharibi, Wajeb & Hahanova, A. & Hahanov, Vladimir & Chumachenko, Svetlana & Litvinova, Eugenia & Hahanov, Ivan. Vector–Logic Synthesis of Deductive Matrices for Fault Simulation. Èlektronic modeling. 2023, tom 45, No 2. 16-33. https://doi.org/10.15407/emodel.45.02.016.

12. Wajeb Gharibi, Vladimir Hahanov, Svetlana Chumachenko, Eugenia Litvinova, Ivan Hahanov, Irina Hahanova. Vector-logic computing for faults-as-address deductive simulation // IAES International Journal of Robotics and Automation (IJRA), vol. 12, P. 274-288, no. 3 September 2023. DOI:10.11591/ijra.v12i3.pp274-288.

13. Тріфанов О. Структури даних для дедуктивного моделювання умовних операторів HDL / О. Шкіль, М. Мірошник, Д. Рахліс, О. Тріфанов, // Сучасний стан наукових досліджень та технологій в промисловості. – 2023. – № 3 (25). – С. 27-42.