УДК 519.711; 004.421; 519.68; 004.89(004.896); 681.5(681.518.5)
КП
№ держреєстрації 0112U000209
Інв. №

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
(ХНУРЕ)

61166, м. Харків, пр. Леніна, 14
тел. (057) 70-21-425

ЗАТВЕРДЖУЮ
Проректор з наукової роботи ХНУРЕ
д-р фіз.-мат. наук, проф.

_____М.І.Сліпченко
«___»_____2015 р.

ЗВІТ
ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ № 268

ПЕРСОНАЛЬНИЙ ВІРТУАЛЬНИЙ КІБЕРКОМП'ЮТЕР ТА ІНФРАСТРУКТУРА АНАЛІЗУ КІБЕРПРОСТОРУ

(заключний)

Керівник НДР
Завідувач кафедри АПОТ ХНУРЕ, д.т.н., проф.  С.В. Чумаченко

2015

Рукопис закінчено 15 лютого 2015 р.

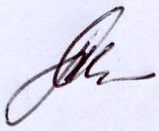
Результати роботи розглянуті науково-методичною радою ХНУРЕ,
Протокол № __ від «__»_____2015 р.

СПИСОК АВТОРІВ

*Керівник НДР, завідувач кафедри АПОТ, д.т.н., проф.	С.В. Чумаченко (розділи 7,9,10)
*Відповідальний виконавець НДР, д.т.н., проф. каф. АПОТ	Є.І. Литвинова (розділи 1-4, 8, 10)
*Проф. каф. АПОТ, декан факультету КІУ, д.т.н.	В.І. Хаханов (розділи 1-10)
*С.н.с. каф. АПОТ, к.т.н.	В.Б. Таранов (розділ 7, 10)
*Н.с. каф. АПОТ, к.т.н.	Д.Є. Шуклін (розділ 10)
*Проф. каф. АПОТ, к.т.н.	В.П. Немченко (розділ 7)
*Проф. каф. АПОТ, д.т.н.	І.В. Хаханова (розділи 5, 10)
*Доц. каф. АПОТ, к.т.н., с.н.с.	О.С. Шкиль (розділ 1,2,10)
*Доц. каф. АПОТ, к.т.н.	Г.П. Фастовець (розділ 7,10)
Проф. каф. АПОТ, к.т.н.	М.Я. Какурін (розділ 8, 10)
Проф. каф. АПОТ, д.т.н.	Г.Ф. Кривуля (розділ 3)
Доц. каф. АПОТ, к.т.н.	І.В. Філіпенко (розділ 10)
Ст. викл. каф. АПОТ	В.І. Обрізан (розділи 6,7)
Ас. каф. АПОТ	О.С. Адамов (розділ 7)
Доц. каф. АПОТ, к.т.н.	Г.В. Хаханова (розділ 10)
Аспірант каф. АПОТ	О.С. Міщенко (розділи 7,9, 10)
Аспірант каф. АПОТ	В.О. Мізь (розділи 3, 10)
Аспірант каф. АПОТ	А.Н. Зіарманд (розділ 7, 10)
Аспірант каф. АПОТ	С.О. Дементьєв (розділ 10)
Аспірант каф. АПОТ	О.Ю. Мова (розділ 3, 10)
Аспірант каф. АПОТ	Д.Ю. Кучеренко (розділ 10)
Аспірант каф. АПОТ	Дахірі Фарід (розділ 6, 7,10)
Аспірант каф. АПОТ	О.С. Приймак (розділ 10)
Аспірант каф. АПОТ	Д.О. Щербін (розділ 10)
Пошукач каф. АПОТ	К.О. Сорудєйкін (розділ 10)
Пошукач каф. АПОТ	І.В. Ємельянов (розділ 10)
Пошукач каф. АПОТ	О.О. Горобець (розділ 10)
Магістрант каф. АПОТ	М.В. Скоробогатий (розділ 10)

Магістрант каф. АПОТ		А.С. Сірокурова (розділ 10)
Магістрант каф. АПОТ		В.В. Білоус (розділ 10)
Аспірант		Багдаді Аммар Авні Аббас (розділи 1,3,4, 10)
Аспірант		Альмадхоун С.М.М. (розділ 10)
Аспірант		Мурад Алі Аббас (розділ 2, 10)
Аспірант		Тамер Бані Амер (розділ 10)
Пошукач, докторант		О.О. Гузь (розділ 5,10)

* - з повною або частковою оплатою в межах фінансування

Нормоконтролер		Т.Г. Силантьєва

РЕФЕРАТ

Звіт про НДР: 351 с., 61 рис., 17 табл., 32 лістинги, 120 джерел.

Публікації: монографії 1, навчальні посібники та підручники 1, статті 40, тези 108, патенти 2.

Мета НДР – створення індивідуального та віртуального комп'ютера в кіберпросторі для виконання інтелектуальних транзакцій з даними і сервісами, орієнтованими на кожну людину.

Об'єкт дослідження – кіберпростір як сукупність взаємозв'язаних сервісів та даних, що доставляються користувачу.

Предмет дослідження: віртуальний персональний кіберкомп'ютер як новий тип інтелектуального інтерфейсу між кіберпростором та користувачем.

Методи дослідження – булева алгебра, векторно-асоціативна логіка, теорія множин, теорія графів, теорія цифрових автоматів – для побудови моделей тестування; логічний аналіз, теорія алгоритмів, методи проектування та моделювання цифрових систем – для синтезу тестів, структур даних і сервісного обслуговування; методи аналізу якості моделей та ефективності сервісного обслуговування цифрових систем – для досягнення заданої глибини пошуку дефектів запропонованими методами; internet of things, big data analytics, кубітні структури даних, квантові технології обчислення, моделі кіберфізичних систем – для управління даними.

За завданням НДР проведено аналіз стану проблеми, висвітлені основні задачі та проблемні питання, означені шляхи їх реалізації, наведено характеристику основних наукових та практичних результатів роботи.

Науково-практичним результатом, на одержання якого була спрямована НДР, є розробка та впровадження методів та моделей, орієнтованих на створення персонального віртуального кіберкомп'ютера та інфраструктури аналізу кіберпростора, який потенційно використовує всі структурні, обчислювальні та інтелектуальні потужності кіберпростору планети.

Ключові слова: віртуальний кіберкомп'ютер, кіберфізична система, цифрові кубітні структури, моделювання, діагностування та ремонт цифрових систем, великі дані, Мультипроцесор, векторно-логічний анализ і простір, критерій якості, діагностування несправностей пам'яті, процес-модель.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CNOT – Контрольоване НІ

SWAP – Оператор обміну станами кубітів

CCNOT – Трикубітовий вентиль Тоффолі

ETG – Extended Toffoli Gate

VLSI – Very Large Scale Integrated

RTL – Register Transfer Level

LAMP – Logical Associative MultiProcessor

НВІС – Надвелика інтегральна схема

SoC – System-on-Chip

ATPG – Automation Test Part of Generation

CDFG – Control-Data Flow Graph

КА – Кінцевий автомат

ТФН – Таблиця функцій несправностей

ПЛІС – Програмована логічна інтегральна схема

ВК – Віртуальний комп'ютер

LUT – Look Up Table

HES – Hardware Embedded Simulator

SSV – System State Vector

TAB – Tests – Assertions –Blocks

CFT – Code-Flow Transaction

АВС-граф – Assertion Based Coverage Graph

HS-система – Hardware-Software система

UUT – Unit Under Test

MUT – Model Under Test

ДНФ – Диз'юнктивна нормальна форма

КНФ – Кон'юнктивна нормальна форма

QHP – Quantum Hasse Processor

GUI – графічний інтерфейс користувача

КС – кібернетична система

HVS – Hardware Vulnerability Simulator

ДП – Деструктивні проникнення

CFTG-граф – Code-Flow Transaction Graph

VLAS – Vector Logical Algebra Space

slc – Shift Left Bit Crowding

PLD – Programmable Logic Device

CPU – Central Processor Unit

LP – логічний процесор

ВСТУП

Перспективи розвитку віртуального комп'ютінгу - створення метрики-семантичної моделі взаємодії технологічної культури big data з інтелектуальними сенсорами моніторингу реального світу для управління кіберфізичними процесами, підвищення якості життя людей і збереження екосистеми планети. Задачі аналізу джерел [55, 56, 58, 59, 107-116, 117-120]: 1) Визначення актуальних ринково-орієнтованих напрямів науково-освітніх досліджень і дизрапторного розвитку кіберекосистеми планети. 2) Ринково орієнтовані структури кіберфізичних систем управління неприродними процесами. 3) Використання технологічної культури big data для створення кіберфізичних систем моніторингу та управління.

Наука - сфера людської діяльності, спрямована на процес збору та аналізу фактів для отримання об'єктивних знань про навколишню дійсність з метою прогнозування природних явищ та активного управління творчими процесами розвитку людства. Мета наукової діяльності - еволюційний перехід науки зі стадії пасивного моніторингу та аналізу до фази активного керування суспільними і кіберфізичними процесами для забезпечення якості життя людей і збереження екології планети.

Освіта - безперервний процес формування (духовної, фізичної, емоційної, інтелектуальної та професійної) культури людини шляхом придбання та накопичення загальнолюдських цінностей, знань, умінь і навичок за допомогою існуючої у часі і у просторі багаторівневої системи виховання і навчання, що має за мету придбання соціальної значущості кожним індивідуумом у творчому процесі розвитку людства, направленому на підвищення якості життя людей і збереження екосистеми планети. Дані визначення істотні для викладу дослідження, яке спрямоване на раціональну зміну кіберфізичної екосистеми планети шляхом вилучення людини з управління неприродними процесами.

Індекси числа наукових публікацій [107] в IEEE Xplore формують модні тренди в напрямку створення кіберфізичної екосистеми: біотехнології (13907), нанотехнології (26176), Internet of Things (4200), Smart Everything (46439), Cloud Computing (19865), Big Data and Analytics (7709), Cyber Physical System (CPS) (2348), мобільні технології (220000) та соціальні мережі (19782), технології кіберзахисту (124917), штучний інтелект (149397). Цікавий також рейтинг посилань в IEEE Xplore для комп'ютерних напрямків ринку науково-освітніх послуг: Program Engineering (79783), Computer Engineering (196748), System Engineering (376614, Electronics - 566496), Social Engineering (10012), Radio Engineering (38546, 151094 - Radiosystems), Electronical Engineering (119856), Production Engineering (45293), Telecommunications (228278), Applied Mathematics (10034), Computer Science (181916), Media Devices and TV (28126), System Programming (159495), Computer Networks and Systems (44853), Internet (118157), Control Systems (563336), Computers (702123), Computer Science and Engineering (61369).

Проривними системо-утворюючими дізрапторами для інвестицій часових, фінансових і людських ресурсів в найближчі 8 років будуть: Crowd-sourcing / open-sourcing of hardware development (419240), 2) Changes in educational structure / design (MOOCs) (387777), 3) Virtual / alternative currencies (Bitcoin) (71), 4) Smartphone for payment (216), 5) Cloud computing (20291), 6) Robots as source of labor (281), 7) Nonvolatile memory influencing big data accessibility and portability (2308), 8) Quantum / nondeterministic computing (7653), 9) 3D printing (1335), 10) Green computing (5827), 11) New user interfaces (Siri, Kinect) (11051).

Кіберфізічна система покликана зробити активною концепцію big data, розглядаючи великі дані у взаємодії з киберсистемами (хмарами) управління, орієнтованими на пошук, розпізнавання і прийняття рішень. Структурний зміст CPS (рис. А) - сукупність комунікаційно пов'язаних реальних і віртуальних компонентів з вираженими функціями адекватного фізичного цифрового моніторингу та оптимального хмарного комп'ютерного

кіберуправленія для забезпечення якості життя, продукції, процесів або сервісів в заданих умовах обмежень на час і ресурси. CPS включає компоненти: Cyber Control, Internet of Things або Cloud, Security, Intelligence, Big Data and Services, Digital Monitoring, Cyber Managing, Physical Smart Everything, Nature, Social, and Tech World. Регуляторні відношення (Relationship) між компонентами CPS формуються законами, статутами підприємств і організацій, морально-етичними правилами поведінки всередині соціальної групи. Напрямок руху RoadMap - Harmony of Human, Nature and Tech кіберфізичної системи людства можна визначити як досягнення такого інтегрального рівня розвитку кіберфізичних компонентів, який забезпечить гармонію життя людини з природою і технікою (створеним світом - Created World).

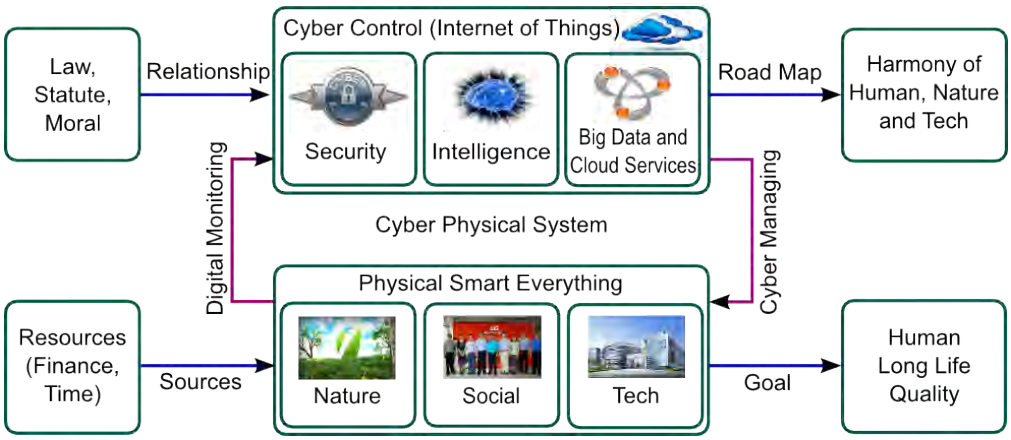


Рисунок А - Кіберфізична система управління неприродними процесами

Гармонія передбачає створення кібер-інтелекту, який до 2050 року повинен позиціонуватися як мозок людства (Humanity Brain); цифрову ідентифікацію всіх фізичних процесів, об'єктів і тривимірного простору за допомогою технологій Internet of Things, Smart Everything і Big Data.

Особливе значення тут набуває досить нова «частково рекламна» парадигма Big Data, для критики якої існує не менше фахівців, ніж тих, хто

вважає її дізраптором або проривною технологією зберігання та аналізу даних 21 століття. На жаль, практично не існує точного визначення даного достатньо модного ринкового тренда.

Big Data - технологічна культура кіберпростору, спрямована на формування інфраструктури, що динамічно розвивається, кіберфізичної екосистеми планети шляхом метрики-семантичної структуризації великих потоків (обсягів) гетерогенних даних на основі використання інтелектуальних швидкодіючих спеціалізованих хмарних фільтрів паралельного моніторингу та метричного аналізу витягуваної інформації для online управління фізичними та віртуальними процесами.

При цьому можна виділити декілька принципів, що характеризують Big Data [107-115,118-120]: 1) Замість причинно-наслідкових зв'язків, пропонується використовувати домінування кореляції інформаційних об'єктів. 2) Замість вибірки даних (максимум користі з мінімуму інформації) - пропонується використовувати повну множину матеріалів. 3) Замість зберігання даних - інноваційно декларується, що цінність даних полягає в рівні їх багаторазового або масового використання вчора, сьогодні і завтра для прогнозування та / або управління дійсністю. 4) Замість традиційних знань для розуміння минулого пропонується здобувати здатність прогнозувати майбутнє. 5) Замість структур даних з жорсткими зв'язками - адресна організація фізичних і віртуальних об'єктів і процесів. 6) Замість ручного введення даних - використання Інтернету як входу до кіберсистеми: smart everything + internet of everything. 7) Замість винесення даних за межі кіберпростору - використання як виходу кіберсистеми Інтернету і керуючих регуляторних впливів для cyber physical systems. 8) Замість технологій пасивного відображення реального та віртуального світу - кіберфізичні системи моніторингу та аналізу даних для управління фізичними та віртуальними процесами. 9) Замість універсальних та великовагових систем збору та аналізу інформації - спеціалізовані віртуальні паралельні мультипроцесори моніторингу та управління фізичними та віртуальними

процесами. 10) Замість хаосу статичних даних і знань в кіберпросторі Інтернету - поступова семантична структуризація динамічних потоків великих даних кіберфізичних процесів і явищ для їх ефективного моніторингу, аналізу та управління. 11) Замість невпорядкованих даних, важких для розуміння і використання людиною або кіберсистемою - розумні, метрично ранжовані інформаційні структури, орієнтовані на прийняття оптимального рішення. 12) Замість відокремленого розвитку реального та віртуального просторів - поступове створення замкнутої кіберфізіческой екосистеми планети для спільного гармонійного розвитку реального та віртуального світів. На додаток до метриці big data, що диференціює, можна ще додати зовсім унікальну характеристику VVVV: volume - велика розмірність даних; velocity - висока швидкодія надання сервісу; variety - потужна просторово-часова семантика і онтологія даних; veracity - висока валідність і точність корисної інформації.

Одним з можливих споживачів big data культури є CPS, яка орієнтована на заміну людини-менеджера кіберхмарними сервісами управління соціальними групами, біологічними, технічними та віртуальними об'єктами. Мета кіберфізіческой системи - Human Long Life Quality - визначається якістю життя людства, соціальних груп і кожної людини в гармонії із зеленою планетою і штучним світом. Мета, як функціональний вихід CPS, залежить від точного цифрового моніторингу та оптимального кіберуправління віртуальними та фізичними ресурсами, що включають час, гроші, кадри і матеріали. Головною відмінністю масштабованої кіберфізичної системи є відсутність людського фактора в блоці управління (Cyber), що робить її, при конструктивній і гуманній законотворчості, справедливою, ефективною, оптимальною, надійною та захищеною від суб'єктивних помилок менеджера.

Ринково-привабливі глобальні проекти сьогодні виконуються під егідою об'єднання фізичного і віртуального простору в єдине ціле. Кібер-фізичний простір (Cyber Physical Space) - метрика телекомунікаційної

взаємодії фізичних, біологічних і соціальних об'єктів, процесів і явищ з віртуальними або хмарними (комп'ютерними) технологіями моніторингу та управління на основі використання Internet of Things & Smart Everything для досягнення соціально значущих цілей. Воно покликане інтегрувати найбільш перспективні кіберфізічні технологічні рішення: 1) Вбудований інтерфейс безпосереднього зв'язку мозку людини з комп'ютером та/або киберпростором шляхом заміни послідовних мовних інтерфейсів на паралельні образні відношення. 2) Створення штучного інтелекту для самонавчання і самовдосконалення кіберфізичних структур, програм і процесів. 3) Нановиращування комп'ютера шляхом адитивного структурування атомів. 4) Найцікавіше рішення пов'язане з невідворотністю природної відмови людства від функцій управління біологічними, соціальними та технічними об'єктами і процесами на користь неупереджених кіберфізичних систем!

Дев'ять технічних лідерів IEEE Computer Society [107] об'єднали зусилля для прогнозування майбутнього планети, до якого включені 23 комп'ютерні технології 2022 року. Сформована кібермода на найближчі 8 років: 3D printing, big data and analytics, open intellectual property movement, massively online open courses, security cross-cutting issues, universal memory, 3D integrated circuits, photonics, cloud computing, computational biology and bioinformatics, device and nanotechnology, sustainability, high-performance computing, internet of things, life sciences, machine learning and intelligent systems, natural user interfaces, computer vision and pattern recognition, networking and interconnectivity, quantum computing, software-defined networks, multicore, and robotics for medical care.

Як підсумок сказаному вище можна констатувати, що кіберфізичні системи, великі дані і квантоподібні паралельні мультипроцесори формують сегмент ринку, призначений для пошуку, розпізнавання та прийняття ефективних управлінських рішень.

Пропонується big data driven кіберфізіческая система (рис. Б) online управління фізичними та/або віртуальними процесами, інваріантними по відношенню до сфер людської діяльності.

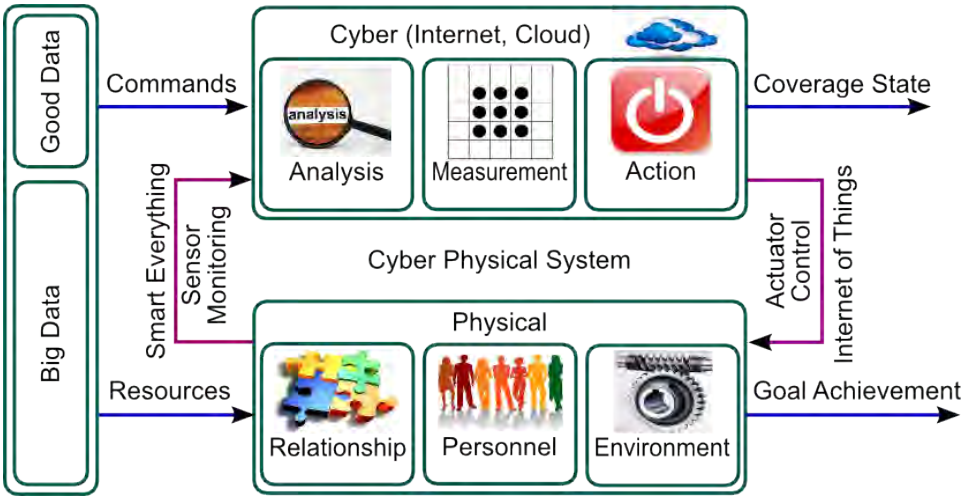


Рисунок Б – Кіберфізична система управління процесами

Окремі задачі, які позиціонуються як процеси, визначені в часі і просторі, можна сформулювати у вигляді наступного списку: 1) цілевказування при стрільбі, коли мішень управляє ракетою; 2) пошук контенту за ключовими словами, які ототожнюються з булевими примітивами; 3) управління світлофором з боку машин, які формують в online-режимі матрицю актюатора; 4) навчальний план управляє освітнім процесом студента; 5) злітно-посадкова смуга керує посадкою літака; маршрут транспортного засобу керує його пересуванням; 6) roadmap, відносини і розумні сенсори керують соціальними інститутами, виробничими підприємствами і компаніями.

Представлена big data driven кіберфізична система управління фізичними процесами має інноваційні відмінності від існуючих рішень: 1) матричне подання мети, процесу, маршруту руху, компетенцій об'єктів і суб'єктів, 2) online виконання всіх процесів, 3) використання нецифрової булеан-метрики для структурного і скалярного оцінювання процесів і явищ, 4) використання метричного дискретного булевого простору для

ідентифікації процесів і явищ, застосування big data технологічної культури для організації та активного управління кіберфізичними процесами, 5) кіберсистемное управління процесами на основі жорсткої взаємодії виконавчого та керуючого механізмів, сенсорів і актюаторів.

Взаємодія інтелектуальних засобів (фільтрів) управління та big data в рамках кіберфізичної системи фільтрів пошуку корисної інформації подібно до процесу видобутку золота з піску за допомогою драги, налаштованої на виділення важкого металу, коли легкі фракції вимиваються водою. Слід зауважити, що баланс екосистеми не порушується, а здобуте золото залишається на планеті, тільки в строго визначеному місці. Аналогічні процеси протікають і в кіберекосистемі планети, коли з хаосу кіберпростору важкою працею розумних фільтрів видобувається корисна інформація, щоб її потім покласти в осередок вже структурованої частини кіберпростору, яка стає reusable і може бути легко доступна всім бажаючим. Людство прагне порядку і структуризації, але воно також відповідально за сміття і хаос в кіберпросторі, обсяги якого ще більшою мірою підпорядковуються закону Мура. Кожні 2 роки інформація подвоюється, а до 2020 року її обсяг складе 40 зеттабайт. Тому процес упорядкування інформації завжди буде відставати від мусоризациі кіберпростору. Сьогодні людство фільтрує і використовує близько 0,4 відсотка вже корисної інформації. Далі не буде більше. Це означає, що в найближчі 100 років актуальність технологічної культури big data для створення «правильної» інфраструктури кіберпростору буде тільки зростати. Прогнозується, що фінансові і кадрові інвестиції для створення інфраструктури кіберекосістеми до 2020 року зростуть на 40%. Інвестиції в збереження та захист інформації, Big Data і Cloud Computing будуть зростати значно швидше [108].

Сьогодні більше 60% компаній роблять інвестиції в технології Big Data, Cloud Services та аналітичні продукти, щоб мати big data-driven кіберуправління кадровими та матеріальними ресурсами. Порядку 60% компаній у світі, за оцінками журналу «Форбс», готові купити програмні

системи управління ресурсами. Компанії замотивовані сімома аргументами: продукти дорослішають і розумнішають, їх стає легко купити, з'явився зручний користувальницький інтерфейс, системи здатні інтегрувати численні програмні засоби компанії, Big Data реально дозволяє управляти кадрами - шляхом playing "Moneyball" with their people data, хмарні технології дозволяють легко перемикатися на нові сервіси управління кадрами, а талант став назавжди стратегічним товаром і головним питанням кожного керівника. Людський капітал, за оцінкою журналу «Форбс», має індекс важливості для вирішення проблем компанії, організації, держави - 2,44; управління і виконання операцій 2,10; інновації 1,99; решта 7 мають індекси: відносини зі споживачами 1,72; глобальна політика 1,68; урядове регулювання 1,55; глобальна експансія 1,31; корпоративний бренд і репутація 0,92; стійкість 0,82 і віра в бізнес 0,46. Дуалізм управління на основі Big Data і Cloud Services включає 1) детермінізм - технології (правильні дані) керують нами і 2) волюнтаризм - ми керуємо технологіями. Обидва варіанти в своєму комплексному розвитку призводять ринок хмарних технологій управління до детермінізму на основі використання концепції кіберфізичних систем, де фігурують величезні масиви даних, не завжди достовірної інформації. Але розумна аналітика движків по Big Data просторам повинна навчитися формувати правильне рішення. Leon Trotsky: "Tell me anyway - maybe I can find the truth by comparing the lies". Скажи мені що ти думаєш в будь-якій формі, а я зумію знайти правду порівнянням навіть неправдивих висловлювань. За даними журналу Форбс технологія Big Data згенерує в 2015 році 3,7 трильйона прибутку в продуктах і сервісах, що означає появу на ринку 4,4 мільйона нових робочих місць. Якщо врахувати, що у всіх компаніях світу заробітна плата становить 40% доходів, то управління персоналом та ресурсами сьогодні є найважливіша проблема бізнесу. Головний висновок із сказаного - людство настільки геніально і одночасно недосконале, що воно не може об'єктивно керувати самим собою! Людина геніальна у творчості і бездарна у самоврядуванні. Таким чином, світовий

ринок безальтернативно приходить до необхідності використання кіберхмарного управління ресурсами і кадрами без участі людини, але на основі витягу правильних даних з кіберінформаційного «сміття».

Мета ринкового бренду big data - вирощування в кіберпросторі культурного шару інфраструктури метрико-семантично впорядкованої легко доступної корисної reusable інформації за рахунок розробки віртуальних хмарних сервісів на основі паралельних мультипроцесорів, що виконують роль швидкодіючих інтелектуальних фільтрів при пошуку, розпізнаванні та прийнятті рішень.

Задачі технологічної культури big data:

1) Зберігання неструктурованої різнорідної інформації в надійних розподілених системах, що обслуговуються Hadoop-сервісами.

2) Аналіз big data в реальному часі за допомогою низькорівневих паралельних інтелектуальних швидкодіючих процесорів середовища Map-Reduce, поміщених в хмарні сервіси.

3) Створення нових метрик вимірювання відстаней між процесами та явищами в кіберпросторі для побудови швидкодіючих метрико-семантичних фільтрів пошуку корисної інформації.

4) Розробка і впровадження інфраструктурних рішень для кіберпростору з метою компактного зберігання і швидкого семантико-метричного витягу корисної класифікованої інформації за допомогою хмарних сервісів і спеціалізованих процесорів.

5) Створення типового шаблону кіберфізичної системи управління big data analytics, що використовує структури хмарних фільтрів для вилучення корисної інформації з великих обсягів неструктурованих даних з метою отримання прибутку шляхом big-data-driven управління фізичними та віртуальними процесами.

6) Побудова масштабованої кіберінформаційної системи масової метрики-семантичної переробки в реальному масштабі часу великих обсягів

даних в корисну структуровану інформацію, що використовується для управління фізичним світом.

7) Розробка big data-driven аналітичної кіберфізичної системи прогнозування (планування) та управління неприродними (соціальними) процесами та явищами (катаклізмами) шляхом точного і вичерпного моніторингу громадської думки для вироблення регуляторних інформаційних управляючих впливів з метою забезпечення якості життя соціальних груп та усунення конфліктів. Наприклад, при бажанні можновладців можна без фінансових і часових витрат усунути всі конфлікти в Україні шляхом законодавчого врахування інтересів усіх верств населення в часі і просторі на основі толерантного об'єднання історичних, мовних і національних культур. Курс будь-якої системи влади на перетин інтересів соціальних груп, очевидно всім і не тільки знаючим математику, неодмінно призводить до порушення цілісності та / або загибелі державних утворень.

Структура кіберінформаційної системи (рис. В) з сигналами моніторингу та управління містить компоненти: 1) «big data - good data», які повинні прагнути процентного співвідношення 90:10. 2) хмарні сервіси-фільтри, призначені для формування 10 відсотків структурованої, легко доступної, корисної і багаторазово затребуваної інформації. 3) Матеріальні та часові ресурси для трансформування кіберпростору в семантико-метричну інфраструктуру правильних даних. 4) Метрики класифікації та оцінювання інформаційних об'єктів, необхідні для створення фільтрів, аналізу big data і синтезу структур правильних даних. 5) Вектор стану кіберпростору, що визначає фактичне співвідношення між «big data і good data» в реальному часі. 6) Мета - досягнення високого рівня правильних даних (Good Data = PureData [118-120]) по відношенню до «інформаційного сміття» - big data і подальшого використання вже метрики-семантично-структурованих даних для оптимального управління кіберфізичними процесами планети.

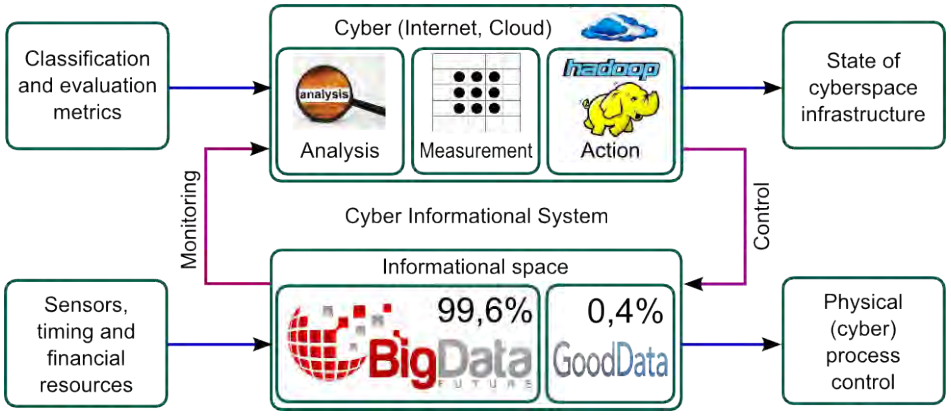


Рисунок В – Кіберінформаційна система трансформування big data

За оцінкою журналу «Форбс» сьогодні 36 відсотків компаній вкладають ресурси в технологічну культуру big data. Однак тільки 13 відсотків з них займаються прогнозуванням бізнесів в своєму сегменті ринку. Тім не менш, вже 16 відсотків компаній намагають використовувати здобуті з big data правильні дані для управління кіберфізичними процесами. Таким чином, можна зробити висновок, что натомість статистичному аналізу даних за частковою представніцькою вібіркою приходить точний и повний аналіз великих даних за наперед заданою тематикою, де експертне формулювання проблеми є мистецтво попадання в ціль.

1 МЕТОДИ АНАЛІЗУ ТА ДІАГНОСТУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ

Представлені основні напрямки технологій проектування, валідації та забезпечення якості (quality assurance) систем на кристалах і в пакетах кристалів.

Мета – аналітичний огляд квантових методів обчислень, технологій тестування і діагностування цифрових систем на кристалах при їх проектуванні та верифікації, орієнтований на підвищення швидкодії програмних і апаратних засобів аналізу цифрових пристроїв, а також істотне збільшення виходу придатної продукції і зменшення часу її виходу на ринок мікроелектроніки.

Завдання розділу: 1) Особливості класичних і квантових обчислень. 2) Побудова моделей цифрових систем. 3) Моделювання справного поведінки цифровий систем. 4) Моделювання несправностей цифрової системи. 5) Генерація тестів. 6) Методи діагностування несправностей. 7) Побудова кубітних моделей цифрових систем.

Джерела дослідження: класичні і квантові обчислення [1-20]; квантові вентилі [6, 15, 27]; види несправностей цифрових схем [28-32, 36, 40-44]; генерація тестів [36, 37, 45]; моделювання цифрових систем [28-35, 38, 39]; технології тестування і діагностування систем на кристалах [28-34, 36-39].

1.1 Класичний і квантовий біт інформації

Класичний і квантовий комп'ютери оперують числами. Двійкова система зчислення класичного комп'ютера ґрунтується на використанні двох чисел: 0 і 1. Кількість інформації, що зберігається в одному двійковому осередку дорівнює 1 біту. У двох двійкових осередках зберігається 2 біти інформації, які можуть становити наступні комбінації: 00, 01, 10. 11. У регістрі, що

складається з n двійкових осередків, зберігається n біт інформації, кількість комбінацій можливих значень яких дорівнює 2^n.

У квантовому комп'ютері [1-20] інформація також може бути представлена у вигляді двійкових чисел. Однак в комірці, званої квантовим бітом (кубітом) може зберігатися не одна з двох чисел довічного числення (0 або 1), а одночасно обидві ці цифри [1,2]. Так наприклад, у двох кубітах можуть зберігатися одночасно 4 довічних числа 00, 01, 10, 11. У n кубітах можуть зберігатися одночасно 2^n двійкових чисел довжини n. Кубіти можуть бути пов'язані один з одним (заплутані): зміна одного з них призводить до змін інших. Сукупність заплутаних кубітів являє собою заповнений квантовий регістр, який може не тільки перебувати у всіляких комбінаціях складових його бітів, але і реалізовувати всілякі залежності між ними [2]. У процесі обчислень на квантовому комп'ютері одночасно обробляються всі біти квантового регістра (має місце квантовий паралелізм).

У процесі обчислень квантові комп'ютери виконують логічні операції над квантовими станами шляхом унітарних перетворень, що не порушують квантові суперпозиції, відповідно до алгоритму [3]:

1) запис (підготовка, ініціалізація) початкового стану;

2) обчислення (унітарні перетворення початкових станів);

3) вивод результату (вимір, проекція кінцевого стану).

Класичний комп'ютер оперує бітами - булевими змінними, що приймають значення 0 і 1 - і на будь-якому етапі обчислень в кожному біті зберігаються певні значення, використовувані для обчислень. На першому етапі обчислень в кожен біт записуються вихідні дані -визначення значення (0 або 1).

Квантовий комп'ютер оперує станами квантового регістра. Дані в процесі обчислень являють собою квантову інформацію, яка після закінчення обчислювального процесу перетворюється в класичну шляхом вимірювання кінцевого стану квантового регістра. Виграш за швидкодією в квантових алгоритмах досягається за рахунок того, що при застосуванні однієї квантової

операції велика кількість коефіцієнтів суперпозиції квантових станів, які у віртуальній формі містять класичну інформацію, перетворюється одночасно (квантовий паралелізм).

1.2 Квантові вентилі

1.2.1 *Прості квантові вентилі*. За аналогією зі звичайним електронним цифровим комп'ютером, вентиль квантової обчислювальної системи, може обробляти один або кілька кубітів [6, 15]. Квантова система має вхідні і вихідні стани, що визначаються (задаються) одним або декількома кубітами. Система перетворює їх відповідно до аксіомам квантової механіки. Стани квантової системи перетворюються за допомогою унітарних, оборотних (що інвертуються) операторів. Отже, квантові вентилі повинні мати однакову кількість вхідних і вихідних кубітів.

Простий квантовий вентиль А, що перетворює один вхідний кубіт в один вихідний кубіт, зображено на рис. 1.1.

Рисунок 1.1 – Квантовый вентиль

Тут передбачається, що в квантовому вентилі потік інформації направлений зліва направо, тому стрілки на рис. 1.1 не показані. Дане спрощення графічного зображення квантового вентиля стало можливим завдяки тому факту, що квантові вентилі перетворюють кубіти за допомогою унітарних оборотних операторів. У цьому випадку графічне зображення логічних вентилів, що перетворюють класичні біти в електронному цифровому комп'ютері, не є зручним для квантових вентилів.

На рис. 1.1 $|\psi>, |\chi> \in C^2$ – кубіти; A: $C^2 \to C^2$ – унітарний лінійний оборотний оператор. Зручно використовувати матричне зображення квантових вентилів:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

де для кубитів $|\psi> = \alpha|0> + \beta|1>, |\chi> = \gamma|0> + \delta|1>$, для яких $A|\psi> = |\chi>$, справедливо:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}. \qquad (1.6)$$

Приклад квантового вентиля NOT, що задається унітарною матрицею:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Можна відмітити, що в цьому випадку вираз (1.6) перетвориться до виду:

$$X(\alpha|0> + \beta|1>) = \beta|0> + \alpha|1>.$$

Іншими словами, вентиль NOT являє собою перемикач між станами |0> і |1>.

Інший приклад квантових вентилів, що описуються унітарними матрицями:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

які діють на даний кубіт у відповідності з виразами:

$$Y(\alpha\,|\,0> + \beta\,|\,1>) = -i(-\beta\,|\,0> + \alpha\,|\,1>),$$
$$Z(\alpha\,|\,0> + \beta\,|\,1>) = \alpha\,|\,0> - \beta\,|\,1>,$$

X, Y і Z називаються матрицями Паулі.

Вентиль Адамара здійснює самооборотну операцію формування суперпозиції станів і визначається унітарноюї матрицею [6, 15]:

$$H = (1/\sqrt{2})\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Справедлив наступний вираз

$$X^2 = Y^2 = Z^2 = H^2 = I,$$

який означає, що кожен вентиль X, Y, Z і H визначається за допомогою кореня квадратного одиничної матриці і відповідного квантового вентиля I.

1.2.2 *Багатокубітні квантові вентилі*. У багатокубітному вентилі має бути однакова кількість кубітів на вході і виході [6].

Двохкубітні вентилі відповідають операціям повороту в гільбертовому просторі двох взаємодіючих кубітів, які не можуть бути подані у вигляді прямого добутку незалежних однокубітових операцій [15].

Основним двухкубітовим вентилем є оборотний контрольований інвертор або оператор «контрольоване HI» (CNOT) з двома вхідними і двома вихідними кубітами (рис. 1.2), який функціонує відповідно до виразів:

$$|00>\mapsto|00>,\quad |01>\mapsto|01>,$$
$$|10>\mapsto|11>,\quad |11>\mapsto|10>.$$

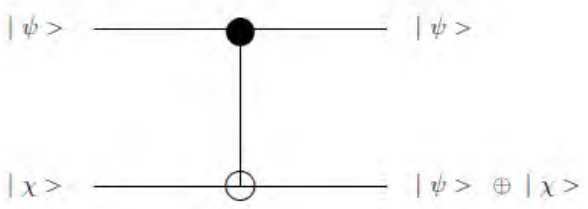


Fig. 2.4.1

Рисунок 1.2 - Контрольоване НІ

Коли $|\psi>=|0>$, то $|\psi>\oplus|\chi>=|\chi>$; для $|\psi>=|1>$ має місце $|\psi>\oplus|\chi>=X|\chi>$.

Вентиль CNOT описується матрицею:

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}\alpha\\\beta\\\gamma\\\delta\end{pmatrix}=\begin{pmatrix}\alpha\\\beta\\\delta\\\gamma\end{pmatrix},$$

де $|\psi>=\alpha|0>+\beta|1>, |\chi>=\gamma|0>+\delta|1>$.

Кубіт $|\psi>$ є контролюючим, кубіт $|\chi>$ – контрольованим, над яким виконується операція NOT за умови, що перший кубіт знаходиться в стані $|1>$.

Двохкубітовий оператор обміну станами кубітів SWAP може бути реалізований шляхом послідовного виконання трьох операцій CNOT (рис. 1.3) [15] та описується матрицею:

$$\mathrm{SWAP}=\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}$$

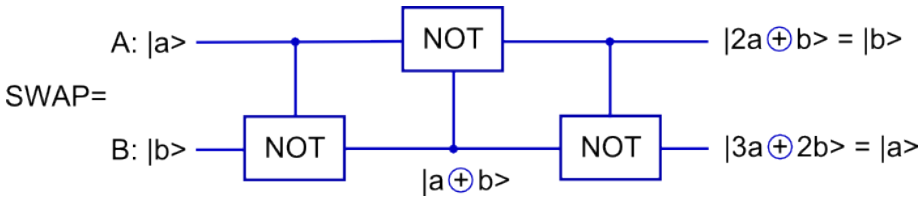


Рисунок 1.3 – Оператор SWAP

Трикубітовий вентиль Тоффолі (CCNOT) зображений на рис. 1.4 і містить два керуючих кубіта A і B, а також один керований C.

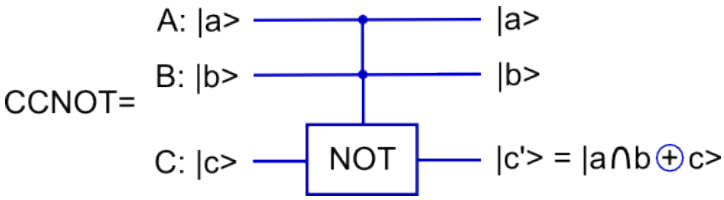


Рисунок 1.4 – Вентиль Тоффолі

Вентиль Тоффолі описується матрицею 8x8 в базісних станах |0,0,0>, |0,0,1>, |0,1,0>, |1,0,0>, |0,1,1>, |1,0,1>, |1,1,0>, |1,1,1>:

$$CCNOT = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Дана операція може бути також реалізована у вигляді п’яти двокубітових операцій.

N-бітний узагальнений вентиль Тоффолі описується як $(k_1, k_2, ..., k_n) \to (k_1, k_2, ..., (k_1, k_2, ..., k_{n-1}) \oplus k_n)$ [20]. Вентиль NOT є окремим випадком вентиля Тоффолі, для якого $n = 1$, вентиль CNOT – окремим випадком, коли $n = 2$.

Расширений $n+1$-біитний вентиль Тоффолі (extended Toffoli gate, ETG) з двома керованими лініями (o_n, o_{n+1}) показано на рис. 1.5. ETG має вхідний вектор $(k_1, k_2, ..., k_n, k_{n+1})$ і вихідний вектор $(o_1, o_2, ..., o_n, o_{n+1})$, де $o_j = k_j$ для $j = \overline{1,(n-1)}$, $o_n = k_1, k_2, ..., k_{n-1} \oplus k_n$, $o_{n+1} = k_1, k_2, ..., k_{n-1} \oplus k_{n+1}$. Перші $n-1$ бітів є керуючими, останні два біта – керованими.

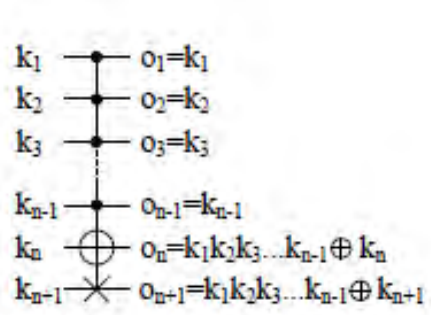


Рисунок 1.5 - $n+1$-бітний вентиль Тоффолі

1.3 Види дефектів

Підходи до моделювання несправностей, діагностування, тестування та забезпечення відмовостійкості цифрових схем мають важливе значення для створення надійних виробів [40]. З розвитком VLSI технології різко істотно збільшилася кількість компонентів, що розміщуються на одному кристалі і, як результат - зросла ступінь інтеграції і кількість несправностей.

У процесі проектування цифрового виробу необхідно періодично перевіряти (шляхом тестування) коректність функціонування схеми і відсутність несправностей, а також забезпечити коректну роботу схеми в разі появи несправностей (відмовостійкість) [36].

Під несправністю схеми розуміють фізичний дефект одного або більше компонентів схеми [28, 29, 32]. Розрізняють постійні та нерегулярні несправності. Постійні (жорсткі) несправності можуть бути наслідком руйнування або зносу компонента. Нерегулярні (м'які) дефекти виявляються

в певні проміжки часу і можуть бути короткочасними (transient) і перемежованими (intermittent) [40-44].

Несправності можуть бути логічними і параметричними. Логічна несправність змінює булеву функцію, яка реалізується схемою. Параметрична - змінює значення параметра схеми (струм, напруга). До параметричних дефектів відносять несправності затримки, пов'язані з різним часом проходження сигналу через логічні вентилі, що призводить до перегонів (змагань) сигналів [46].

Моделювання великої кількості фізичних дефектів може бути засноване на використанні однієї моделі логічної несправності, що дозволяє істотно зменшити складність моделювання. Модель логічної несправності не залежить від технології імплементації проекту, а тести, розроблені для виявлення логічної несправності, можуть застосовуватися також і для виявлення фізичних дефектів.

Модель логічної несправності може бути явною або неявною. Явна модель несправності визначає простір несправностей, в якому кожна несправність може бути ідентифікована, а несправності, підлягають аналізу, можуть бути явно описані. Явна модель несправності практично застосовна, якщо її розмірність не є надто великою. Неявна модель описує простір несправностей шляхом сукупної ідентифікації несправностей певного типу їх характеристичними ознаками. Моделювання несправностей тісно пов'язане з моделюванням системи. Несправності, які визначаються в поєднанні зі структурною моделлю, відносяться до структурних несправностей, що виявляється у зміні структури міжз’єднань компонентів. Функціональні несправності визначаються в поєднанні з функціональною моделлю. Так, наприклад, функціональна несправність може проявлятися у зміні таблиці істинності компонента або спотворенні RTL операції.

Класи несправностей. Існує три класи логічних несправностей: константна несправність (stuck-at-fault), мостикова несправність (bridging fault) і несправність затримки (delay fault).

Найбільш поширена модель константної несправності - одиночна константна несправність. Сутність моделі полягає в тому, що несправність логічного вентиля призводить до «залипання» логічного 0 (константа 0, sa-0) або логічної 1 (константа 1, sa-1) на одному з його входів або виходів [42, 44].

Модель константної несправності також використовується для зображення кратних несправностей у схемі. При цьому передбачається, що більш, ніж на одній лінії схеми є константна несправність sa-0 або sa-1. Іншими словами, сукупність константних несправностей існує в схемі в один і той же час. Різновидом кратної несправності є односпрямована несправність - коли всі констітуенти (складові частини) несправності являють собою sa-0 або sa-1, але не обидві одночасно.

Модель константної несправності не є ефективною при моделюванні HBIC (Very Large Scale Integrated, VLSI), побудованих за CMOS технологією. Несправності в CMOS схемах не обов'язково являють собою логічні дефекти, які можуть бути описані моделлю константної несправності. Дефекти CMOS-схем відображаються також моделями стійких обривів транзисторів SOP (stuck-open) і стійких замикань транзисторів SON (stuck-on) [42].

Мостикові несправності типу «коротке замикання» являють собою постійні дефекти, які не можуть бути змодельовані константною несправністю. Коротке замикання виникає, коли дві або більше сигнальних ліній схеми електрично пов'язані одна з одною. Мостикові несправності вентильного рівня класифікуються наступним чином: вхідні - викликані коротким замиканням входів логічного елемента, несправності типу зворотного зв'язку - викликані замиканням вхідної і вихідної ліній, а також несправності без зворотного зв'язку, які не відносяться до перших двох типів. У теорії моделювання мостикових несправностей робиться припущення, що ймовірність замикання більше двох ліній є низькою і логіка міжз’єднань реалізується у вигляді зв'язків. Мостикова несправність в позитивній логіці виникає в тому випадку, коли її поведінка описується проводовим AND (0 є

домінантним логічним значенням), і в негативній логіці - коли її поведінка описується проводовим OR (1 є домінантним логічним значенням).

Несправності затримки [41, 43]. Невелика кількість дефектів, які можуть викликати розриви і короткі замикання в схемі, мають досить високу ймовірність появи через наявність відхилень параметрів виробничого процесу. Дефекти можуть також призводити до порушень часових параметрів схеми без зміни логіки її роботи: затримка перемикання сигналу з 0 в 1 і навпаки. Існує два види несправностей затримки: несправність затримки вентиля і несправність затримки шляху. Затримка вентиля використовується для моделювання дефектів, при яких час проходження сигналу через вентиль перевищує гранично-допустимий. Дана модель може бути використана тільки для ізольованих, не транспортованих дефектів, наприклад, кілька малих затримок. Модель затримки шляху може бути використана як для ізольованих, так і для транспортованих дефектів. При цьому передбачається, що несправність проявляється у випадку, якщо затримка поширення сигналу уздовж лінії схеми перевищує припустиме значення.

Перемежовані несправності розглядаються як часові дефекти. Основна частина несправностей цифрових схем викликана саме часовими дефектами, які характеризуються складністю виявлення та усунення. Перемежовані дефекти є неповторяюваними і викликаються, як правило, флуктуаціями напруги живлення або впливом радіаційного випромінювання. Вони є основною причиною відмови елементів пам'яті систем на кристалах.

Перемежовані несправності можуть з'являтися в результаті порушення міжз’єднань, застосування дефектних компонентів, впливу зовнішніх факторів (температура, вологість, вібрація) або бути наслідком помилок проектування. Перемежовані несправності виникають випадковим чином і моделюються за допомогою імовірнісних методів (Марківські моделі).

1.4 Генерація тестів

Безперервне вдосконалення технологій проектування і виробництва цифрових виробів призводить до збільшення щільності компонування і складності пристроїв, досягнення необхідного рівня надійності яких забезпечується тестуванням. Для вирішення завдання тестування сучасних надскладних електронних пристроїв необхідні нові, більш ефективні методи побудови тестів [47-52]. Виробництво систем на кристалах (system-on-chip, SoC) з використанням технології глибокого субмікронного (Deep Submicron, DSM) дозволяє знизити витрати, але при цьому вартість тестування залишається незмінною і являє собою значну частину загальної вартості проекту. Зменшити витрати на тестування виробу можна шляхом повторного використання блоків інтелектуальної власності (IP cores), а також розробки моделей і методів тестування SoC на високому рівні ієрархії [53]. Високорівневі модулі системи на кристалі описуються в термінах поведінки функціональних компонентів, що не дозволяє використовувати для їх тестування готові технічні рішення вентильного рівня. Класична модель одиночної константної несправності (stuck-at fault), що представляє внутрішні логічні вентилі або їх міжз'єднання, не може бути застосована для використання на системному рівні. Структурне високорівневе тестування не може бути виконано з використанням готових тестових рішень, оскільки генерація тесту виконується після структурного синтезу. Реалізація тестування залежить від технології виготовлення SoC і змінюється в процесі життєвого циклу виробу [36].

Для забезпечення можливості багаторазового використання тестів у нових проектах необхідно розробити таку модель несправності, яка є незалежною від реалізації системи на кристалі. Слід також знайти відповіді на запитання: 1) Чи може тест, побудований на базі функціональної моделі несправності, бути ефективно використаний для непокриваємих тестом фізичних дефектів? 2) Як ефективність тесту залежить від синтезованої

структури? Ці запитання є важливими не тільки з точки зору повторного використання тестів, але також через те, що програмні модулі можуть бути синтезовані за допомогою існуючих систем автоматизованого проектування і збережені в бібліотеках IP модулів.

Для успішного проектування і виробництва виробу необхідні методологія тестування і моделі несправностей, які забезпечують високорівневу валідацію проекту [36].

Дефект-орієнтоване тестування, засноване на генерації тестів на транзисторному рівні і використанні струмових моделей (IDDQ), ефективно використовується в технології глибокого субмікронного. IDDQ метод тестування заснований на вимірюванні струму і добре працює, коли середній струм схеми з несправністю більше, ніж струм справного пристрою [37]. Дефект-орієнтоване тестування починається після реалізації етапу розміщення і трасування (place and route) (рис. 1.6) [36].

Сутністю константної моделі несправностей є абстракція реального дефекту. Модель є основою для автоматичної генерації тестових наборів і формування алгоритмів моделювання несправностей. Умовою досягнення високого рівня покриття несправностей є те, що тест повинен транспортувати деяке конкретне значення (або їх сукупність) в дефектну область від керованих точок введення і далі до спостережуваних виходів з метою виявлення несправної поведінки. Дана модель найбільш ефективна для тестування на кристалі (post-silicon testing). Генерація тестів для константних несправностей виконується перед розміщенням і трасуванням проекту (рис. 1.6). Тести для константних несправностей покривають реальні дефекти топології тільки примітивних вентилів. У цьому випадку говорять про генерації тестів, не залежних від топології кристала.

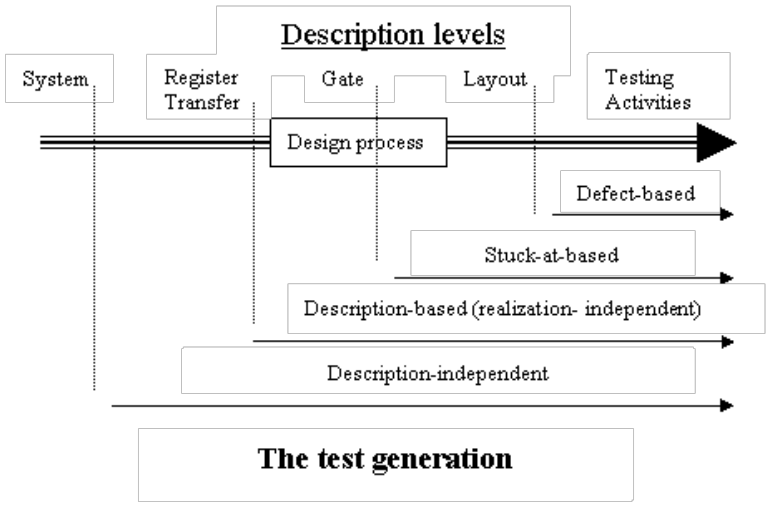


Рисунок 1.6 – Генерація тестів в процесі проектування

Відомі кілька підходів до генерації тестових наборів на рівні регістрових передач (Register Transfer Level, RTL) [36]: використання двійкових дерев рішень, виявлення спотворень в RTL описі схеми, об'єднання статичного аналізу з симуляцією. Більшість з них дозволяють генерувати тестові послідовності задовільної якості, що сумісні із засобами ATPG вентильного рівня. Головною перевагою тестування пристрою на RTL рівні є те, що розмірність опису схеми тут набагато менше, ніж на логічному рівні. При генерації тестів на рівні регістрових передач набір тестових послідовностей створюється для всіх можливих реалізацій, і можна говорити про генерацію тестів, не залежних від імплементації. Завдання генерації тестів може виконуватися паралельно з синтезом схеми на вентильному рівні (див. верхню частину рис. 1.7).

Генерація тестів на системному рівні залежить від використовуваної моделі несправностей. У цьому випадку не тільки реалізація, а й синтезований поведінковий опис не відомі. Задача генерації тестів в цьому випадку є більш складною, але може вирішуватися одночасно з формуванням синтезованого опису та синтезом схеми на вентильному рівні (див. рис. 1.7) [36].

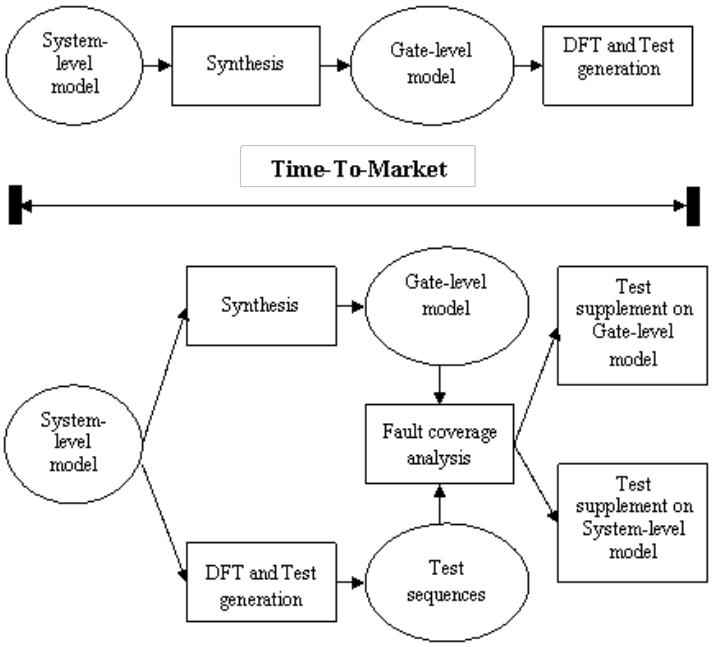


Рисунок 1.7 – Час time-to-market на різних етапах процесу проектування

Перед відправкою проекту у виробництво необхідно сформувати тестові набори. Верифікація пристрою передбачає запуск тестової програми на робочій моделі кристала. Тестери є дорогими компонентами і можуть використовуватися протягом тривалого часу. Початок розробки тестів наприкінці процесу проектування значно збільшує час виходу виробу на ринок. Якщо використовується методологія проектування зверху вниз, то системна модель виробу на кристалі формується на самому початку процесу проектування і може бути використана при розробці тестової програми. Таким чином, інженер-тестувальник може стати учасником процесу проектування на ранніх стадіях і використовувати віртуальний прототип пристрою у вигляді системної моделі. Це дозволить суттєво зменшити час проектування і вартість виробу.

Методологія проектування зверху вниз орієнтована на автоматичний синтез списку з'єднань вентильного рівня з використанням поведінкового опису або системної моделі. Час виходу виробу на ринок (time-to-market) залежить від тривалості процедури логічного синтезу, тривалості тестопридатного проектування і генерації тестів (див. рис. 1.7).

Тестопридатність проектування і генерація тестів на основі системної моделі дозволить скоротити час виходу на ринок [36, 45]. Аналіз тестового покриття на вентильному рівні не є часовитратною процедурою і дає можливість зменшити довжину тестових послідовностей, отриманих на системному рівні.

Генерація тестів на системному рівні не може гарантувати 100% покриття несправностей вентильного рівня для кожної можливої імплементації. Формування тестових послідовностей необхідно виконувати паралельно на системному і на вентильному рівнях, оскільки ймовірність генерації тестів для несправностей, що важко виявляються, може бути різною на кожному рівні опису.

1.5 Моделі функціональних несправностей

Все більш широке використання програмно-апаратних систем в критичних додатках привело до підвищення значущості верифікації та тестування програмних і апаратних модулів. В даний час існує ряд проблем верифікації, пов'язаних з високою складністю програмно-апаратних додатків і їх гетерогенною структурою [35, 48-53]. Вартість верифікації системи збільшилася до такої міри, що іноді навіть перевищує вартість проекту. Формальні методи верифікації дозволяють перевірити функціональність за допомогою формальних методів (перевірка моделей, перевірка еквівалентності, автоматичне доведення теорем). Для управління складністю задачі верифікації, запропоновані методи, що засновані на симуляції (емуляції) опису системи заданої вхідної послідовністю.

Функціональні несправності спотворюють простір станів цифрового виробу, що представлено специфікацією. Дефект проектування являє собою неправильну деталь проекту, сформовану розробником. Дефекти проектування є наслідком синтаксичних (семантичних) помилок в описі пристрою або фундаментального нерозуміння функціональності, описаної

проектною специфікацією. Кількість потенційних дефектів проектування може бути занадто великим, щоб з ними можна було боротися автоматично або вручну, тому необхідно застосовувати способи зменшення складності проекту без шкоди для точності результатів. Модель проектної несправності описує поведінку деякої множини дефектів проектування. Модель функціональної несправності описує фізичні та проектні дефекти апаратних і програмних модулів. Модель функціональної несправності можна оцінити точністю моделювання проектних дефектів і ефективністю (кількістю виявлених несправностей схеми).

Більшість апаратних систем розробляються на основі методології проектування зверху вниз, яка починається з поведінкового опису системи. Як результат, більшість моделей функціональних несправностей є моделями поведінкового або алгоритмічного рівня. Існуючі моделі функціональних несправностей можуть бути класифіковані за стилем поведінкового опису, на якому вони базуються. Системна поведінка описується на мовах програмування (System C) або опису апаратури (VHDL, Verilog) і перетворюється у внутрішній формат для використання в процесі симуляції.

1.5.1 *Текстові (семантичні) моделі неисправностей.* Текстова (семантична) модель несправностей використовується для вихідного текстового поведінкового опису проекту [38]. Найпростішою текстовою моделлю є метрика покриття інструкцій (statement coverage metric), яка використовується при тестуванні програмного забезпечення, що пов'язує потенційну помилку з кожним рядком коду, і вимагає, щоб кожен оператор поведінкового опису виконувався під час тестування [38]. Ця модель не дуже ефективна, оскільки кількість можливих несправностей дорівнює числу рядків коду. Обмеження точності покриття інструкцій дозволяє в поєднанні з іншими моделями несправностей підвищити ефективність тестування.

Ряд моделей функціональних несправностей базується на обході шляхів графа потоків управління (Control-Data Flow Graph, CDFG), що описує поведінку системи [38]. Ранні моделі несправностей CDFG

грунтувалися на покритті гілок і шляхів графа. Покриття гілок припускає, що багато всіх перевірених шляхів графа CDFG охоплює два напрямки реалізації всіх бінарних умов. Покриття гілок широко використовується при тестуванні програмного і апаратного забезпечення, однак використання тільки даної моделі не дозволяє отримати повну гарантію коректності коду.

Метрика покриття шляхів є більш ефективною у порівнянні з метрикою покриття гілок, оскільки вона відображає кількість шляхів графа потоків управління. Передбачається, що дефект пов'язаний з деяким шляхом графа потоків управління і, отже, для гарантованого виявлення всіх несправностей повинні бути виконані всі шляхи потоку управління. Кількість шляхів управління може бути нескінченною, якщо граф CDFG містить цикл. Тому, метрика обходу шляхів може бути обмежена довжиною шляху. Оскільки загальне число шляхів потоку управління зростає експоненціально з кількістю умовних операторів, можна вибрати підмножину всіх шляхів потоку управління, необхідну і достатню для тестування. Одним з критеріїв вибору шляху може бути базисний набір шляхів або підмножина шляхів, які лінійно незалежні і можуть утворювати будь-який інший шлях. При тестуванні потоків даних поява кожної змінної розглядається або як опис змінної, або як її використання. При виборі шляху розглядаються такі, які пов'язують визначення змінної з її використанням. Критерії тестування потоків також застосовуються для перевірки поведінкового опису апаратних модулів.

Більшість моделей несправностей графа потоків управління розглядають шляхи без обмеження значень змінних і сигналів. На противагу їм, існують моделі несправностей, орієнтовані на змінні/сигнали, які включають більш жорсткі обмеження на величину сигналу з метою виявлення несправностей. Методика аналізу домену при тестуванні програмного забезпечення розглядає не тільки шлях потоку управління, але і значення змінних і сигналів під час виконання. Домен являє собою підмножину простору вхідних елементів програми, в якому кожен елемент

активізує виконання програми за деяким шляхом. Несправність домену викликає виконання програми, наслідком якого є перехід в неправильну область. Даний метод може бути також застосований для тестування апаратного забезпечення.

Багато моделей несправностей графа потоків управління містять вимоги до активізації несправностей не залежно від значення спостережуваності. Для усунення цього недоліку запропоновані поведінкові моделі несправностей, застосовні для тестування software і hardware модулів. Підхід OCCOM [38, 56] базується на додаванні несправностей, званих тегами, до кожного визначення змінної, що представляють позитивне чи негативне зміщення від правильного значення сигналу. Знак помилки відомий, але величина - ні. Аналіз спостережуваності уздовж шляху потоку управління робиться вероятностно, з використанням алгебраїчних властивостей операцій і даних моделювання. Тег буде поширюватися за допомогою поведінкової операції, якщо будуть виконані дві умови: 1) співпаде знак; 2) інші входи в процесі виконання операції не контролюються. Розроблено також точний метод визначення спостережуваності, в якому константна несправність вводиться на внутрішніх змінних, і її поширення забезпечується поведінкою об'єкта. Оскільки аналіз спостережуваності є точним, обчислювальна складність при цьому зростає.

1.5.2 *Мутаційні моделі несправностей*. Мутаційне тестування засноване на штучному внесенні несправностей в код програми і застосовується для тестування програмного і апаратного забезпечень [38]. Основна ідея полягає в імітації типових помилок програміста і створення спеціальних тестів для їх виявлення (тестів, які б виявляли несправності, якби вони були присутні). Несправності вводяться в оригінальну програму і створюється багато несправних версій програми. Кожна з них містить одну помилку. Помилкові програми називаються мутантами оригінальної програми. Метою генерації тестів є розмежування оригінальної програми та всіх її мутантів. Оригінальна програма і всі її мутації тестуються на одному і

тому ж наборі тестів. Якщо на цьому наборі підтверджується правильність програми і виявляються всі помилки в програмах-мутантів, то оригінальна програма оголошується правильною. Якщо в деяких мутантів були виявлені всі помилки, то тестовий набір вважають не повним, і він підлягає доопрацюванню. Набір мутаційних операторів мови VHDL включає зміни наступних об'єктів: арифметичні оператори, визначення і зміна абсолютного значення і константи, логічні оператори, реляційні оператори, додавання унарного оператора. Кожен оператор зображує певний клас несправностей. Всі можливі зміни в програмі не можуть розглядатися через їх непомірно велику кількості. Зміни можуть бути обмежені до прийнятного набору на основі двох гіпотез: ефект зчеплення і компетентний програміст. Ефект зчеплення свідчить, що складні несправності можуть поєднуватися з простими несправностями, таким чином, тестовий набір, який виявляє всі прості помилки в програмі, буде виявляти також і складні несправності. Гіпотеза «компетентний програміст» стверджує, що компетентний програміст прагне писати програми, які практично є правильними. Іншими словами, програма, написана компетентним програмістом може бути неправильною, але вона буде відрізнятися від правильної версії відносно простими помилками. Недоліком мутаційних моделей є локальний характер мутацій, що обмежує застосування моделей для опису великого набору дефектів проектування.

1.5.3 *Моделі несправностей кінцевого автомата*. Кінцеві автомати (КА) є класичним способом опису поведінки послідовних схем, і для них визначені моделі несправностей [38]. Найбільш поширеною є модель покриття станів, заснована на вимозі покриття всіх можливих станів і виконання всіх можливих переходів в процесі тестування. Дефекти КА не виводять його за простір станів, заданий специфікацією. Проблема використання моделей несправностей кінцевого автомата полягає у високій складності вирішення завдання тестування, яка обумовлена великою розмірністю простору станів типової обчислювальної системи. Рішенням

зазначеної проблеми може бути виявлення підмножини станів кінцевого автомата, що мають вирішальне значення для його коректного функціонування. Моделі розширеного кінцевого автомата (Extended Finite State Machine, EFSM [60, 77]) і машини контролю (Extracted Control Flow Machine, ECFM) дозволяють зменшити кінцевий автомат шляхом його розподілу на простір станів і простір даних. Зменшений кінцевий автомат генерується шляхом проектування оригінального кінцевого автомата на множину станів, що мають найбільше значення для процесу валідації.

1.6 Модель справної поведінки об’єкта

Модель об'єкта діагностування – це сукупність гетерогенних компонентів, взаємопов'язаних у часі та просторі, що із заданою адекватністю описують певний процес або явище. Модель може бути подана в аналітичній, табличній, векторній, графічній або іншій формі і задана в явному або неявному вигляді [28, 29, 32].

Явна модель об'єкта діагностування складається з описів його справної і всіх несправних модифікацій. Неявна модель містить опис справного об'єкта, моделі його фізичних несправностей і правила отримання за ними всіх несправних модифікацій об'єкта. Універсальною математичною моделлю об'єкта діагностування є таблиця функцій несправностей (ТФН). Кожний несправний стан об'єкта діагностування відповідає одній несправності (одиночній або кратній) із заданого класу несправностей. Недоліком ТФН є її великі розміри. Модель дискретної системи може бути зображена у вигляді таблиці істинності, логічної мережі, альтернативного графа, еквівалентної нормальної форми подання булевих функцій, таблиці переходів-виходів багататактної схеми. Вибір моделі впливає на глибину і трудомісткість процесу діагностування.

1.7 Логічне моделювання

Логічне моделювання є формою верифікаційного тестування проекту з використанням моделі проектованої системи. На вхід моделі подаються вхідні стимули, виконується побудова та аналіз часових діаграм для зовнішніх входів, виходів схеми і внутрішніх ліній [35].

Верифікація проекту дозволяє перевірити функціональні режими щодо специфікації. Перевірка здійснюється на кожній стадії перетворення моделі від системного рівня до рівня імплементації проекту в кристал шляхом порівняння результатів, отриманих в процесі моделювання, і еталонних результатів, передбачених специфікацією.

Завдання, які вирішуються в процесі логічного моделювання: 1. Перевірка правильності функціонування цифрової схеми. 2. Дослідження часових параметрів схеми (швидкодія, час виконання операцій, тактова частота). Виявлення змагань, ризиків збою, аналіз затримок. 3. Оптимізація проектних рішень. 4. Генерація часових діаграм. 5. Генерація тестових послідовностей. 6. Моделювання несправностей.

Для верифікації проекту необхідний прототип пристрою, що функціонує на заданій робочій частоті, однак створення прототипу є дорогим і трудомістким процесом. Заміна прототипу програмної моделлю називається симуляцією. Верифікація проекту з використанням симулятора має такі переваги:

– перевірка помилкових умов (наприклад, конфлікти шин);

– можливість зміни затримок в моделі для перевірки граничних часових параметрів;

– перевірка заданих користувачем значень параметрів схеми;

– можливість початку моделювання схеми на будь-якому етапі проектування;

– точний контроль таймінгу асинхронних подій (наприклад, переривань);

– можливість формування автоматизованого тестового оточення модельованої схеми, використання RTL моделі для управління і спостереження за поведінкою схеми в процесі моделювання.

Технології використання апаратного прототипування за допомогою PLD мають ряд істотних переваг в порівнянні з програмним симулятором: швидкодія і можливість корекції проекту.

Методи логічного моделювання можна класифікувати за такими ознаками: 1) залежно від способу обліку часу поширення сигналу - синхронні (без урахування затримок в елементах схеми) і асинхронні (з урахуванням затримок); 2) залежно від способу подання сигналів – двійкові і багатозначні (троїчні, п'ятизначні); 3) за способом організації роботи програми – компілятівні та інтерпретативні; 4) залежно від організації черговості моделювання – покрокові і подієві.

Синхронне моделювання призначене для аналізу перехідних процесів в цифрових пристроях вентильного та функціонального рівнів опису на основі моделей елементів, представлених їх логічними функціями без урахування затримок сигналів. У процесі моделювання обчислюють значення сигналів на виходах логічних елементів схеми за заданими вхідними сигналами. При цьому передбачається, що час існування перехідного процесу набагато більше номінальної затримки схеми. Синхронне моделювання найбільш ефективно використовується для аналізу роботи комбінаційних схем в сталому режимі. Результат моделювання в цьому випадку найбільш точно відповідає реальному режиму роботи пристрою. До методів синхронного моделювання відносять:

– Метод Ейхельбергера, призначений для синхронного аналізу перехідних процесів в цифрових пристроях вентильного рівня опису;

– Багатозначне синхронне моделювання, що дозволяє виявляти всі реальні змагання в схемі. Цей метод іноді вказує на помилкові змагання, що призводить до додаткових витрат при логічній верифікації цифрових систем.

Рішенням зазначеної вище проблеми є асинхронні методи аналізу цифрових схем. Їх різноманітність визначається значністю алфавіту моделювання та ступенем адекватності моделей за реальними часовими параметрами.

Асинхронне моделювання застосовується для аналізу перехідних процесів в логічних схемах з урахуванням часу поширення сигналів в елементах і сполучних ланцюгах схеми. Кожен компонент схеми характеризується деякою середньої затримкою, значення якої може змінюватися залежно від режиму роботи компонента, комбінації вхідних сигналів, температури, відхилень в технології виготовлення.

До методів асинхронного моделювання відносять:

– Δ-Троїчне моделювання, яке усуває недоліки двійкового асинхронного і троїчного синхронного методів;

– Асинхронне троїчне моделювання з наростаючою невизначеністю, яке усуває детермінізм в модельній затримці компонента схеми, укладаючи її в деякий інтервал.

1.8 Моделювання несправностей

Сутність моделювання несправностей полягає у визначенні впливу одного або декількох дефектів на стани ліній об'єкта при подачі тестових послідовностей [39, 54]. Методи моделювання несправностей можна класифікувати наступним чином: одиночне, паралельне, дедуктивне, кубічне і спільне моделювання.

Одиночне моделювання несправностей базується на внесенні однієї одиночної константної несправності еквіпотенційної лінії до схеми. При подачі тестових послідовностей виконується аналіз прояву несправності на

зовнішніх виходах об'єкта діагностування. Метод орієнтований на обробку схем нерегістрового рівня і не вимагає значних часових витрат.

Паралельне моделювання несправностей ґрунтується на використанні машинних команд паралельної обробки слів (регістрів): логічне додавання, множення, інверсія, виключне АБО. Метод відноситься до компілятивного моделювання, оскільки поведінка примітивів схеми описується за допомогою алгоритмічних мов або асемблерів. У процесі моделювання одночасно виконується аналіз P несправностей на вхідному наборі, де P – розрядність машинного слова, доступного для паралельної обробки. До недоліків методу відносять складність проектування моделей і їх орієнтацію на конкретну обчислювальну платфрму. Швидкодія методу в P разів вище одиночного моделювання несправностей. Ідея паралельної обробки бінарного вектора за допомогою тільки логічних операцій може бути використана для істотного збільшення швидкості моделювання.

Дедуктивне моделювання несправностей полягає в одночасній обробці всіх одиночних константних несправностей схеми на одному вхідному наборі і виділенні при цьому підмножини перевірюваних дефектів. Метод орієнтований на вентильний рівень опису моделі проектованого об'єкта в базисі І-АБО-НЕ. Необхідність отримання аналітичних формул для кожного типу примітивного елемента і великі витрати пам'яті для зберігання списків несправностей ускладнюють практичну реалізацію методу.

У спільному (конкурентному) моделюванні, як і в дедуктивному, виявляються відразу всі перевірювані несправності для даного вхідного набору. Метод орієнтований на обробку різних типів моделей схем, несправностей, затримок і сигналів. На відміну від дедуктивного методу, де дефекти моделюються неявно, конкурентний алгоритм аналізує явно справну роботу і ті несправності, які модифікують стани входів або виходів схеми, що використовуються ефективні моделі елементів, такі як табличні та функціональні.

Дедуктивно-паралельне моделювання несправностей цифрових систем ґрунтується на використанні переваг дедуктивного і паралельного алгоритмів [39] і дозволяє за одну ітерацію обробки модельованої схеми виявити всі несправності, що перевіряються на тест-векторі. Метод дозволяє істотно підвищити швидкодію моделювання одиночних константних несправностей для оцінки якості синтезованих тестів цифрових систем, імплементованих в ПЛІС, що містять мільйони вентилів.

1.9 Методи діагностування несправностей

Задачами технічного діагностування є: визначення технічного стану об'єкта, пошук місця і визначення причин відмови. Визначення технічного стану об'єкта здійснюється за допомогою спеціальної тестової послідовності вхідних впливів. Методи формування тестових послідовностей для діагностування несправностей можна умовно розділити на кілька типів, описаних нижче.

1. Розподіл, використання дерев рішень – полягає в моделюванні поведінки справної системи і систем з N заздалегідь визначеними несправностями. Відгук кожної з них на вхідний вплив використовується для формування (N+1) систем. При цьому повинні виконуватися умови:

– справна система повинна бути швидко відокремлена від несправних (виявлення несправностей);

– всі системи є однозначно ідентифікованими (помітними) (виявлення несправностей і визначення їх місця розташування).

Результуючий розподіл або побудова дерева рішень визначає діагностичну тестову послідовність, яка дозволяє однозначно визначити належність тестованої системи однієї з (N+1) категорій.

Для пошуку несправності застосовують послідовний, комбінаційний і послідовно-комбінаційний методи. Послідовний метод полягає в такій побудові процедури пошуку несправностей, при якому інформація про стан

окремих тестованих систем вводиться і логічно обробляється послідовно. Реалізація методу полягає в основному у визначенні черговості контролю. Програма пошуку при цьому може бути жорсткою або гнучкою. Жорстка програма передбачає наявність заздалегідь визначеної послідовності контролю. При гнучкій програмі зміст і порядок подальших перевірок залежать від попередніх результатів. Комбінаційний метод полягає в тому, що на вхід тестованої системи подається фіксований набір тестів. Діагноз формується тільки після того, як будуть отримані відгуки на всі тестові впливи.

2. Активізація одновимірного шляху. Даний метод грунтується на введенні відомої несправності в схему і її транспортуванні на один з первинних виходів за активізованим шляхом. При цьому будь-яка зміна логічного значення в місці несправності призводить до зміни значення на відповідному первинному виході. Описана процедура носить назву прямої фази. Зворотня фаза полягає у визначенні значень інших первинних входів і виходів, таких, щоб задана несправність виявлялася на первинному виході. Метод простий і зручний у використанні, проте у схемі можуть існувати несправності, для перевірки яких необхідно активізувати кілька шляхів (у разі наявності збіжних розгалужень).

3. Використання таблиці функцій несправностей і таблиці несправностей. Таблиця функцій несправностей є спеціальною формою подання поведінки об'єкта діагностування у справному та несправному станах. Таблиця несправностей пов'язує набір тестів і перевірюваних ними несправностей. Обмеженням даного методу є розмірність зазначених таблиць.

4. Метод булевих похідних. Булева похідна визначається шляхом виконання операції OR над двома булевими функціями, які представляють справний і несправний об'єкт. Якщо булева похідна дорівнює 1, вважається, що проявляється помилка і визначається відповідна тестова послідовність.

Тестові набори визначаються шляхом формування булевої похідної для кожної несправності.

5. Метод еквівалентної нормальної форми заснований на зображенні булевої функції у вигляді еквівалентної нормальної форми, яка описує конкретну реалізацію схеми. Еквівалентна нормальна форма може бути обчислена методом підстановки, з тією різницею, що надлишкові терми не виключаються, так як вони характеризують конкретну реалізацію схеми.

1.10 Висновки до розділу 1

Двійкове моделювання або емуляція квантових схем на класичних комп'ютерах не відображає всіх властивостей квантових схем. Емуляція квантових компонентів на класичних комп'ютерах не дає суттєвих результатів щодо збільшення швидкодії [21-25]. У той же час моделювання цифрових класичних схем на основі використання окремих властивостей кубіта може збагатити «бідні» алгоритми класичних машин. Наприклад, існує ще не вирішена проблема, до якої можна підступитися: як за один такт промоделювати 2 в ступені n вхідних наборів, поданих на цифрову схему, якщо ці набори представлені одним двійковим вектором-кубітом?

2 КУБІТНІ МОДЕЛІ ЦИФРОВИХ ПРИСТРОЇВ

Пропонуються кубітні моделі та методи підвищення швидкодії програмних і апаратних засобів аналізу цифрових пристроїв за рахунок збільшення розмірності структур даних і пам'яті. Вводяться основні поняття, терміни та визначення, необхідні для імплементації квантових обчислень в практику аналізу віртуальних комп'ютерів. Представляються результати досліджень, що стосуються проектування і моделювання комп'ютерних систем в кіберпросторі на основі використання двокомпонентного автомата <пам'ять, транзакції>.

2.1 Моделі віртуальних обчислювачів для аналізу кіберпростору

Ринкова привабливість емуляції квантових методів обчислень при створенні віртуальних (хмарних) комп'ютерів (ВК) в кіберпросторі ґрунтується на використанні кубітних моделей даних, орієнтованих на паралельне вирішення задач дискретної оптимізації за рахунок істотного підвищення витрат пам'яті. Тут не розглядаються фізичні основи квантової механіки, що стосуються недетермінованої взаємодії атомних частинок [10, 55-57], але використовується поняття кубіта як двійкового або багатозначного вектора для спільного і одночасного завдання булеана станів дискретної області кіберпростору на основі лінійної суперпозиції унітарних кодів, орієнтованих на паралельне виконання методів аналізу і синтезу компонентів кіберпростору.

Мотивація нового підходу для проектування ВК пов'язана з появою хмарних сервісів, які представляють собою спеціалізовані і розосереджені в просторі віртуальні комп'ютерні системи, інваріантні по відношенню до реалізації в апаратурі або в програмному продукті. Інакше, сьогодні вже не завжди цікаво програмісту як і куди імплементуються коди програмних додатків, так само як і немає необхідності знати теорію проектування

цифрових автоматів на рівні вентилів, реєстрових передач і використовувати специфічні компоненти обчислювальної техніки (тригери, регістри, лічильники, мультиплексори). Будь-який компонент функціональності покривається векторною формою таблиці істинності, реалізованою за допомогою пам'яті. Логічні функції в традиційному виконанні тут виключені з розгляду. Від цього частково страждає швидкодія, але враховуючи, що 94 відсотки SoC-кристала сьогодні є пам'ять, то решта 6 відсотків також можна реалізувати на пам'яті, що не буде критичним для більшості програмних і апаратних додатків. Тому практично корисною для програмування ефективних віртуальних комп'ютерів буде теорія, заснована на двох, більш високого рівня абстракції, компонентах: пам'ять і транзакція.

Особливість організації даних в класичному комп'ютері полягає в тому, що кожен біт, байт або інший компонент має свою адресу. Тому існує проблема в ефективній обробці асоціації (кінцевого алфавіту символів) рівних за значимістю елементів, які не мають порядку за визначенням, наприклад, множина всіх підмножин (булеан). Рішенням може бути процесор, де елементарною коміркою служить образ або шаблон універсуму з n унітарно кодованих примітивів, які використовують суперпозицію для формування булеана $Q = 2^n$ всіх можливих станів такої комірки у вигляді множини всіх підмножин [58, 59]. Простежується певна аналогія за структурами даних з квантовим комп'ютером, де поняттю кубіта в класичному комп'ютері можна поставити у взаємно однозначну відповідність чотири стани: {10,01,11,00} [60] векторного завдання булеана примітивів Кантора $A^k = \{0,1,X,\varnothing\}$, де X – описує «переплутаний», одночасно суперпозиціонуючий стан двох рівнів сигналів 0 и 1. Більшість робіт, зокрема [10, 55-57], розглядає можливість емуляції класичних обчислювальних процесів на квантових комп'ютерах, але враховуючи «реверсність» або оборотність згаданої відповідності, далі пропонується зворотне перетворення – емуляція деяких переваг квантових обчислень на класичних процесорах.

На ринку електронних технологій існує конкуренція між базами імплементації ідеї [61]:

1) Гнучка (м'яка) реалізація проекту пов'язана з синтезом інтерпретативної моделі програмної реалізації пристрою або в апаратному виконанні програмованих логічних пристроїв на основі FPGA, CPLD. Тут переваги полягають в технологічності модифікації проекту, недоліки – у невисокій швидкодії функціонування цифрової системи;

2) Тверда реалізація має орієнтацію на використання компілятивних моделей при розробці програмних додатків або на імплементацію проекту в кристалах VLSI. Переваги та недоліки жорсткої реалізації інверсної по відношенню до м'якого виконання проектів: високу швидкодію і неможливість модифікації.

З урахуванням чотирьох викладених базових варіантів для реалізації ідеї нижче пропонуються квантові структури даних, орієнтовані на підвищення швидкодії гнучких моделей програмного або апаратного виконання проекту.

2.2 Квантові структури опису цифрових систем

n-Кубіт є векторна форма унітарного кодування універсуму з n примітивів для завдання булеана станів 2^{2^n} за допомогою 2^n двійкових змінних. Наприклад, якщо n = 2, то 2-кубіт задає 16 станів за допомогою 4-х змінних. Якщо n = 1, то кубіт задає 4 стани на універсумі з двох примітивів за допомогою 2-х двійкових змінних (00,01,10,11) [55, 61]. При цьому припускається суперпозиція (одночасне існування) у векторі 2^n станів. Кубіт (n-кубіт) дає можливість використовувати логічні операції замість теоретико-множинних для істотного прискорення процесів аналізу дискретних систем. Далі кубіт ототожнюється з n-кубітом або вектором, якщо це не заважає розумінню викладеного матеріалу. Оскільки квантові обчислення пов'язані з аналізом кубітних структур даних, то далі частково експлуатується

визначення «квантовий» для ідентифікації технологій, що використовують дві властивості квантової механіки: паралелізм обробки та суперпозицію станів. Синонімом кубіта при завданні логічної функціональності є: Q-покриття (Q-вектор) [60], як уніфікована векторна форма суперпозіційного завдання вихідних станів, що відповідають унітарним кодам адрес вхідних змінних будь-якої логічної функції.

Кубіт в цифровій системі виступає як форма завдання структурного примітиву, інваріантної до технологій реалізації функціональності (hardware, software). Більш того, «квантовий» синтез цифрових систем на основі кубітних структур вже не прив'язаний жорстко до теореми Посту, яка визначає умови існування функціонально повного базису. На пропонованому рівні абстракції n-кубіт дає вичерпні і більш широкі можливості для векторного завдання будь-якої функції з множини $\beta(f) = 2^n$. Формат структурного кубітного компоненту цифрової схеми $Q^* = (X, Q, Y)$ містить інтерфейс (вхідні і вихідну змінні), а також кубіт-вектор Q, який задає функціональність $Y = Q(X)$, розмірність якого визначається степеневою функцією від числа вхідних ліній $k = 2^n$.

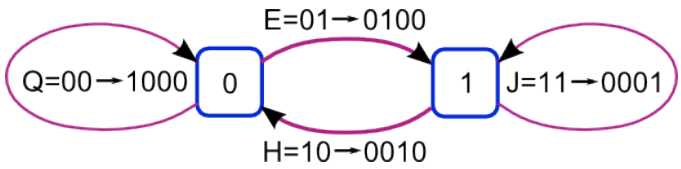
Практично орієнтована новизна кубітного моделювання полягає в заміні таблиць істинності компонентів цифрового пристрою векторами станів виходів. Досить просто можна продемонструвати такі перетворення, стосовно до логічного елементу. Нехай функціональний примітив має наступне двійкове покриття:

$P =$

X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0

яке можна трансформувати шляхом унітарного кодування вхідних векторів на основі використання двотактного алфавіту [61, 62]. Спочатку він був

призначений для компактного опису всіх можливих переходів автоматних змінних, що ілюструється на рис. 2.1 відповідним графом і інтерпретацією символів.



Q = 00 → 1000	E = 01 → 0100	H = 10 → 0010	J = 11 → 0001
O = {Q, H} = = {00,10} → 1010	I = {E, J} = = {01,11} → 0101	A = {Q, E} = = {00, 01} → 1100	B = {H, J} = = {01,11} → 0101
S = {Q, J} = = {00,11} → 1001	P = {E, H} = = {01,10} → 0110	C = {E, H, J} = = {01,10,11} → 0111	F = {Q, H, J} = = {00,10,11} → 1011
L = {Q, E, J} = = {00, 01,11} → 1101	V = {Q, E, H} = = {00, 01,10} → 1110	Y = {Q, E, H, J} = = {00, 01,10,11} → 1111	∅ = 00 → 0000

Рисунок 2.1 – Двохтактний алфавіт автоматних змінних

Тут зображено: символи, їх виконавчі та унітарні коди (наприклад, Q = 00 - 1000), призначені для опису двох сусідніх станів автоматних змінних. Структурно алфавіт являє собою булеан (множину всіх підмножин) станів на універсумі з чотирьох примітивів Y = {Q, E, H, J}. Унітарний код відповідає формату вектора, що містить два кубіта, за допомогою яких формуються 16 символів двотактного алфавіту. Використовуючи останнє, будь-яке покриття функціонального двохвходового логічного примітиву можна представити двома кубами або навіть одним, враховуючи, що вони взаємно інверсні:

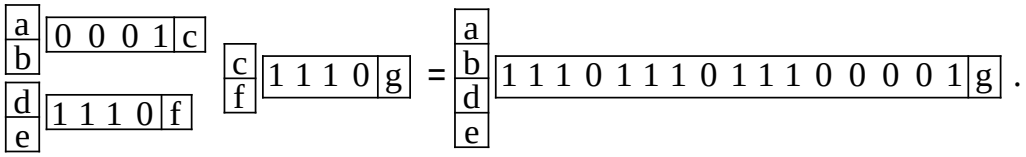
$$P = \begin{array}{|c|c|}\hline 00 & 1 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline Q & 1 \\ E & 1 \\ H & 1 \\ J & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline V & 1 \\ J & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1110 & 1 \\ 0001 & 0 \\ \hline \end{array} \rightarrow \boxed{1\ 1\ 1\ 0}$$

Тут спочатку кодуються всі пари символами двотактного алфавіту, потім виконується об'єднання перших трьох кубів за правилом оператора співграней [58]: вектори, що відрізняються за однією координатою, мінімізуються в один. Далі отримане покриття з двох кубів кодується

відповідними даним символам кубітними векторами. Для моделювання справної поведінки досить мати тільки один куб (нульовий або одиничний), оскільки другий завжди є доповненням до першого. Отже, орієнтуючись, наприклад, на одиничний куб, що формує на виході 1, можна прибрати біт стану виходу примітиву, що зменшить розмірність куба або моделі примітиву до кількості адресованих станів елемента, де адреса є вектор, складений з двійкових значень вхідних змінних, за яким визначається стан виходу примітиву. Зважаючи на тривіальність, немає сенсу показувати, що за аналогією будь-яку таблицю істинності можна привести до кубітної функціональності у формі вектора вихідних станів логічного елементу, що має n входів.

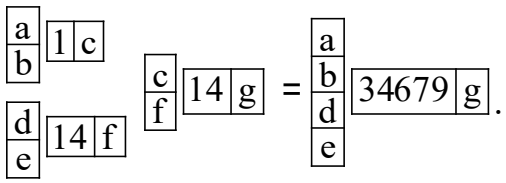
Процедура моделювання на Q-векторі функціональності зводиться до запису у вихідну змінну Y стану біта, адреса якого сформована на основі конкатенації значень вхідних змінних: $Y = Q(X) = Q(X_1 * X_2 ... * X_j ... * X_k)$. Для моделювання цифрових систем, де компонентами виступають взаємопов'язані на основі M-вектора еквіпотенціальних ліній Q-примітиви, 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-елементу буде мати формат: $M(Y_i) = Q_i[M(X_i)] = Q_i[M(X_{i1} * X_{i2} ... * X_{ij} ... * X_{ik_i})]$. В даному випадку істотно спрощується алгоритм аналізу цифрової системи і в 2^n разів підвищується швидкодія інтерпретативного моделювання за рахунок збільшення обсягу пам'яті для опису функціональності схемної структури.

Синтез Q-покриття цифрової системи зводиться до виконання операції суперпозиції над Q-векторами функціональностей, що входять до неї. Наприклад, для трьох примітивів (елементи and, and-not, and-not), що складають схему, операція суперпозиції формує Q-вектор всієї функціональності, де його розмірність буде більшою, ніж сума Q-покриттів вихідних примітивів:



Але при цьому процедура моделювання Q-вектора структури буде мати більш високу швидкодію, оскільки вона подана тільки одним зверненням до Q-покриття для виїмки вмісту з осередку замість трьох, коли система зображена трьома примітивами.

Трьохелементна схема, що подана вище Q-векторами, може бути задана схемотехнично (зручно для людини), де замість векторів будуть фігурувати відповідні десяткові номери:



При обробці такої форми функціональних покриттів необхідно «розгорнути» десятковий код в двійковий вектор і обчислити адресу комірки, вміст якої буде визначати стан вихідної змінної, в даному випадку g. Природно, що десятковий код існує на папері, а в комп'ютері це подання – 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, яка має гнучкість заміни примітивів в разі виявлення помилок проектування або дефектів.

Кубітне подання функціональних елементів дає також можливість запровадити нові схемотехнічні позначення, пов'язані з десятковим номером Q-вектора, що задає функціональність. Якщо система логічних елементів має

n = 2 входи, то число всіх можливих функцій дорівнює $k = 2^{2^n}$, де типи або номери функціоналів зображені в нижньому рядку наступної таблиці:

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

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

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

У даній таблиці подані чотири вектори-примітиви вхідних змінних (00,01,10,11), що утворюють $k = 2^{2^2} = 2^4$ повну множину всіх можливих функцій, які розглядаються як примітиви другого рівня. Потім – вектори-примітиви вихідних змінних (16 стовпців від 0000 до 1111), формують вже $k = 2^{2^4} = 2^{16}$ функціональних примітивів, що входять до більш складної дискретної системи, які можна аналізувати паралельно! Далі можна екстраполювати створення більш складної системи кубітів, де вектор Q=0111110000001100, поданий нижнім рядком, буде розглядатися як один з $k = 2^{2^{16}}$ примітивів третього рівня ієрархії. У кожному рівні ієрархії кубітів кількість або булеан станів експоненціально залежить від числа примітивів-векторів $k = 2^{2^n}$. Якщо вектор Q має всі одиничні значення

Q=1111111111111111, то він одночасно визначає простір, що містить 16 символів двотактного алфавіту, які відповідають булеану на універсумі з чотирьох примітивів [62].

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

2.3 Графові структури опису цифрових схем

Дещо відмінна модельна схемотехніка, не прив'язана безпосередньо до транзисторів, може бути представлена графовими структурами, де кожна вершина (дуга) ототожнюється з функціональним перетворенням, яке задається Q-вектором. Тоді дуга (вершина) визначає взаємозв'язки між функціональними Q-покриттями, а також вхідні і вихідні змінні. Питання реалізації таких структур спочатку прив'язано до осередків пам'яті (LUT FPGA), які здатні зберігати інформацію у вигляді Q-вектора, де кожен біт або розряд має свою адресу, що ототожнюється з вхідним словом. Тим не менш, програмна реалізація таких структур стає конкурентоспроможною за швидкодією на ринку промислових систем проектування цифрових систем на кристалах за рахунок адресної реалізації процесів моделювання функціональних примітивів. Крім того, апаратна підтримка систем

проектування у вигляді Hardware Embedded Simulator (HES, Aldec) [63] набуває нової мотивації на системному рівні проектування цифрових виробів, коли програмні і апаратні рішення мають один і той же кубітний формат. Далі для розгляду пропонується комбінаційна схема (рис. 2.2), що містить 6 примітивів і три різних логічних елемента. Даній схемі відповідають три універсальні графові форми цифрової функціональності (рис. 2.3), що використовують Q-вектори для завдання поведінки логічних примітивів.

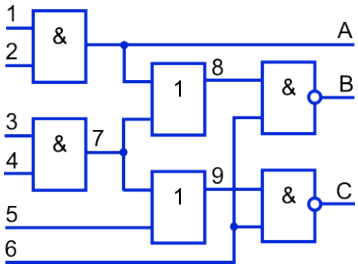


Рисунок 2.2 – Комбінаційна структура логічних примітивів

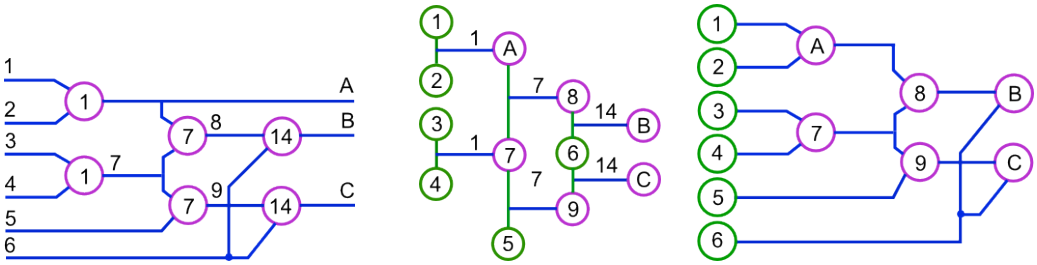


Рисунок 2.3 – Графові форми квантових функціональностей

Структура, що представлена на рис. 2.3,а містить 12 ліній (дуг), навантажених на квантові функціональності (1 = 0001, 7 = 0111, 14 = 1110). Вона подібна традиційній структурно-функціональній моделі комбінаційної схеми. Граф на рис. 2.3,б нагадує модель реєстрових передач С.Г. Шаршунова [64], який є зворотним по відношенню до першої структури. Тут горизонтальні сині дуги ототожнюються з функціональностями, а вершини – з групами вхідних для функціональностей ліній, об'єднаних в реєстрові змінні

допомогою зелених вертикальних дуг, стани яких утворюють двійковий вектор, що використовується як адреса для обчислення стану логічного елемента або більш складного функціоналу. Змінні, що беруть участь у формуванні адреси для Q-вектора функціональності, можна об'єднати в одну вершину з вказівкою всіх ідентифікаторів ліній, що створюють вектор-адресу. Регістровий граф комбінаційної схеми є ранжируваним за рівнями формування вхідних сигналів, що забезпечує умови паралелізму обробки елементів одного рівня і здійсненності ітерацій Зейделя [5,9], що підвищують швидкодію алгоритмів справного моделювання цифрових систем. Структура на рис. 2.3,б цікава своєю реєстровою реалізацією, що можна використовувати для формалізації як програмних, так і апаратних моделей вентильного, реєстрового і системного рівнів. Таке уявлення важко сприймається людиною, але воно технологічно і легко розуміється комп'ютером для автоматичного створення програмних систем аналізу обчислювальних структур і сервісів кіберпростору. Таким чином, регістровий граф цифрової схеми являє собою розривну за гальванічними зв'язками, а тому гнучку систему взаємопов'язаних адресних примітивів для формування функціональної структури будь-якої складності, насамперед у масштабах PLD, де всі комбінаційні примітиви представлені постійними запам'ятовують пристроями (LUT), що забезпечує високу швидкодію функціонування та ремонт логічних модулів в режимі online.

Одновімірній Q-вектор опису функціональності можна прив'язати до вихідної (внутрішньої) лінії пристрою, стан якої формується в процесі моделювання розглядуваного Q-покриття. Тоді реєстрова реалізація комбінаційного пристрою може бути представлена вектором моделювання M, на невхідні лінії якого навантажена функціональність з дугами, що йдуть від вхідних змінніх, значення яких задають адресою біта Q-вектора, що формує стан розглянутої невхідної лінії (рис. 2.3,в). Інакше, якщо функціональності описуються одновиходовими примітивами, то кожен з них можна ототожніти або ідентифікувати номером чи координатою невхідної

лінії, на яку навантажений даний елемент. Якщо функціональність багатовходова, то Q-покриття вже представляється матрицею з кількістю рядків, що дорівнює кількості виходів. Ефект від такого примітиву полягає у паралелізмі одночасного обчислення станів кількох виходів за одне звернення до матриці за поточною адресою! Дана обставина є істотним аргументом на користь синтезу узагальнених кубітів для фрагментів цифрового пристрою або всієї схеми з метою їх паралельної обробки в одному часовому такті. Близько до ідеальної за компактністю та часом обробки структурою даних, де Q-вектори функціональностей та номери вхідних змінніх прив'язані до невхідних ліній пристрою, є наступна таблиця:

L	1	2	3	4	5	6	7	8	9	A	B	C
M	1	1	1	1	1	0	0	1	1	1	0	1
X	.	.	.	.	.	.	34	A7	75	12	86	96
Q	.	.	.	.	.	.	0	0	0	0	1	1
	.	.	.	.	.	.	0	0	1	1	1	1
	.	.	.	.	.	.	0	0	1	1	1	1
	.	.	.	.	.	.	1	1	1	1	0	0

Вона дає уявлення про те, які змінні цифрової схеми є зовнішніми, скільки функціональних примітивів мається у структурі, а також які входи навантажені на кожен Q-вектор. Перевагою таблиці є відсутність вектора номерів виходів для кожного примітиву, але при цьому зберігається необхідність мати номери вхідних змінних для формування адрес, маніпулювання якими є достатньо витратним за часом процесом. Модель аналізу схемної структури спрощується до обчислення двох адрес (!). При формуванні вектора моделювання: $M_i = Q_i[M(X_i)]$ шляхом виключення складної адреси виходу примітиву в процесі запису станів виходів у координати M-вектора.

Кубітно-регістровий граф з рис. 2.3,в може бути поданий у вигляді матриці $\mu = \left|\mu_{ij}\right|, i = \overline{1,p}; j = \overline{1,q}$ паралельно-послідовної обробки логічних примітивів:

μ_{ij}	1	2	3
1	1/2 | 1 | A	A/7 | 7 | 8	8/6 | 14 | B
2	3/4 | 1 | 7	7/5 | 7 | 9	6/9 | 14 | C
3	X/X | 1 | X	X/X | 7 | X	X/X | 14 | X

який відображає взаємодію Q-покриттів за трьома рівнями спрацьовування щодо формату (X-Q-Y) входи- Q-вектор -вихід кожного примітиву: [(1,2-1-A), (3,4-1-7)], [(A, 7-7-8), (7,5-7-9)], [(8,6-14-B), (6,9-14-C)]. Щоб коректно відпрацювала функціональність примітиву, необхідно до розглянутого моменту сформувати всі його вхідні змінні. Тому кубітно-регістровий граф розділений на рівні спрацьовування, де всі примітиви всередині одного рівня можуть оброблятися паралельно, а самі рівні – послідовно один за одним. Кубітна матриця своєю регулярною структурою орієнтована на вирішення завдань:

1) Ремонт логічних примітивів в процесі функціонування за рахунок переадресації дефектних елементів на примітиви із запасу (рядок 3) [65], подібно до того, як це робиться в матричної пам'яті;

2) Індексна адресація кожного кванта матриці $\mu_{ij} \in \mu, \mu_{ij} = (X_{ij}, Q_{ij}, Y_{ij})$ для оперативного ремонту примітивів, що відмовили, (у прикладі можна замінити три дефектних примітиву, по одному з кожного шару);

3) Забезпечення високої швидкодії прототипу комбінаційного пристрою, реалізованого на основі кубітних примітивів, які імплементуються на кристалі PLD в LUT-елементи [61], за рахунок надання можливості паралельної обробки елементів одного шару;

4) Створення матричного кубітного мультипроцесора, орієнтованого на аналіз апаратних прототипів комбінаційних пристроїв великої розмірності, що дозволяють істотно прискорити процеси тестування та верифікації

цифрових систем на кристалах, як це робить Hardware Embedded Simulator (HES), компанії Aldec [63];

5) Розробка методів аналізу комбінаційних схем, орієнтованих на матричне виконання кубітних структур логічних елементів шляхом їх імплементації в елементи пам'яті кристалів PLD;

6) Створення генератора коду для масштабованого синтезу квантових матриць комбінаційних схем на основі використання структур схемотехнічних примітивів кристалів PLD;

7) Проектування керуючого автомата для функціональної обробки та сервісного обслуговування (відновлення працездатності) кубітної матриці комбінаційного пристрою, імплементованого в PLD структуру.

Модель керуючого автомата для симулювання кубітної структури комбінаційної схеми вкладається в три пункти:

1) Ініціювання чергового вхідного впливу для комбінаційного пристрою.

2) Вибір чергового шару (стовпця матриці) с номером і для паралельної обробки кубітних примітивів Q з метою формування станів виходів за адресою вхідного слова, представленого вектором $M(X_{ij})$, де X_{ij} – вектор номерів вхідних змінних для примітиву Q_{ij}, M – вектор моделювання всіх ліній комбінаційного пристрою: $M(Y_{ij}) = Q_{ij}[M(X_{ij})]$, $j = \overline{1,q}$

3) Приростання індексу стовпця i = i + 1 і перехід до пункту 2 обробки чергового шару кубітних примітивів. Після закінчення аналізу всіх стовпців матриці $i = p$ виконується інкремент індексу чергового вхідного впливу t=t+1 з наступним переходом на пункт 1. При досяжності кінцевого числа вхідних наборів $t = n_{max}$ цикл обробки тесту для кубітної матриці закінчується.

2.4 Автоматна структура MQT-процесора

Основою процес-моделі функціонування обчислювача (комбінаційної схеми), представленого у формі кубітної матриці, є операція транзакції (зчитування-запису) двійкової інформації на LUT-елементах пам'яті структури PLD: $M(Y) = Q[M(X)]$, що формує як завгодно складні функціональності і сервіси. Інакше, всі обчислювальні процеси в комп'ютерних системах, мережах та у кіберпросторі можна звести до однієї операції транзакції на будь-якій структурі, що здатна зберігати інформацію. Всі технологічні та схемотехнічні вузли, на яких реалізуються апаратні і програмні продукти, можна не брати до уваги, щоб синтезувати віртуальні інформаційні сервіси, де слід використовувати тільки операцію запис-зчитування, як базову процедуру над Q-покриттями, які своєю гнучкою універсальністю поглинають всі конструктиви обчислювальної техніки для синтезу та аналізу об'єктів, процесів і явищ в кіберпросторі. З'являється нова MQT-модель «м'якого» кубітного віртуального комп'ютера, заснована на транзакції (Transaction) між компонентами пам'яті Q (Qubit), об'єднаними в систему за допомогою вектора моделювання M (Memory). Природно, що будь-які логічні функції реалізуються шляхом зчитування бітів Q-вектора $Q = (q_1, q_2, ..., q_i, ..., q_n)$, $q_i \in \{0,1\}$. Але при цьому стає дещо надмірною теорема Поста, що формує умови у вигляді математичного функціонального базису логічних функцій, необхідно і достатньо повного для створення будь-якої обчислювальної системи, оскільки Q-вектор є універсальна, гнучка і компактна форма опису як простої, так і складної функціональності кіберпростору. Транзакція може існувати тільки за наявності заданих відношень на компонентах пам'яті, число яких повинно бути не менше одного. Циклічність транзакції $M \xleftarrow{Y_i} Q_i \xleftarrow{X_i} M$, (рис. 2.4,а,б) представленої Read–Write операціями $M(Y_i) = Q_i[M(X_i)]$ в MQT-структурі, визначається використанням вектора M взаємних зв'язків компонентів, який

утворює «м'яку», адресно орієнтовану і гальванічно розірвану автоматну модель, де головним компонентом обчислювальної системи фігурує кубіт-пам'ять:

$$\begin{cases} A =< M, Q, f, g, X, Y >, \\ M(t+1) = f[X(t), Q(t), M(t)]; \\ Y(t) = g[X(t), Q(t), M(t)]; \\ M(Y_i) = Q_i[M(X_i)]. \end{cases}$$

Дані вирази підходять під класичне визначення автомата Муру, але тут рівність $M(Y_i) = Q_i[M(X_i)]$ задає основну і єдину процедуру аналізу адресованих Q-компонентів цифрової системи для формування координат вектора стану M. Останній являє собою адресно обчислювані реакції Q-примітивів пристрою на вхідні впливи, отримані конкатенацією координат вектора M, адреси яких відповідають номерам вхідних змінних розглянутого примітиву. Автомат має вхідну реєстрову змінну X, вектор вихідних сигналів Y, який будучи підмножиною змінних вектора M, може бути виключений з розгляду. Адресні реєстрові змінні Y_i, X_i кожного примітиву дають можливість формувати стан вектора M шляхом моделювання структури квантових примітивів Q.

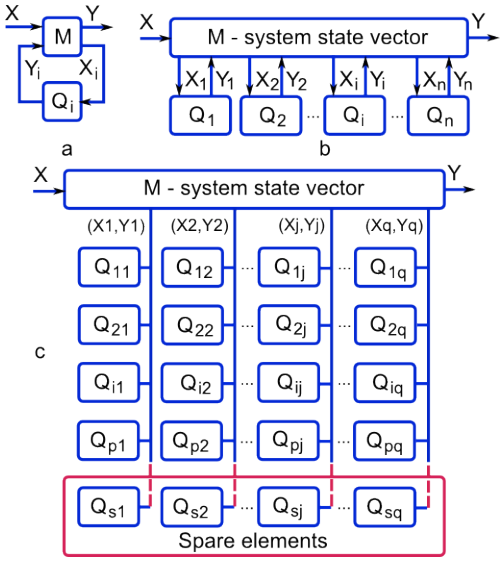


Рисунок 2.4 – Бітова, векторна та матрична структури кубітних примітивів

Тут (рис. 2.4) модель цифрового виробу у вигляді структурного MQT-автомата містить вектор моделювання або стану системи, який об'єднує і «м'яко» структурує всі кубітні примітиви для формування функціональності, заданої специфікацією. При цьому цикл обробки кожного Q-елементу полягає у транзакційній взаємодії між собою пари компонентів M-Q (рис. 2.4,a), які в сукупності реалізують універсальну функціональність $M(Y_i) = Q_i[M(X_i)]$ за допомогою двох транзакцій $M \xleftarrow{Y_i} Q_i \xleftarrow{X_i} M$. Обробка всіх кубітних елементів на вхідній дії X дає можливість сформувати вектор M стану цифрової системи (System State Vector - SSV), який при необхідності має і вихідні змінні Y (рис. 2.4,b) для управління іншими компонентами обчислювальної структури. Швидкодія аналізу (запис-зчитування) векторної (лінійної) структури з $n = p \times q$ примітивів має оцінку: $\gamma = (R + W) \times n$. Строго послідовний характер обробки упорядкованих за зростанням номерів виходів квантових примітивів можна вдосконалити в бік зменшення часу аналізу схеми. Для цього необхідно побудувати вже двовимірну структуру - матрицю Q-елементів, що ранжовані за рівнями паралельної обробки груп примітивів, оформлених в стовпці (рис. 2.4,c). Швидкодія такої структури в порівнянні з лінійною підвищується в q разів, що стає вже сумірним з часом обробки комбінаційної схеми на основі жорстко заданих гальванічних зв'язків $\gamma = \frac{1}{q}(R + W) \times p$. Але головна перевага матриці «м'яких» зв'язків компонентів – наявність нижнього рядка запасних примітивів для ремонту в режимі online, що робить будь-який проект, що містить комбінаційну логіку, тестопридатним, завдяки оперативній переадресації Q-примітиву, який відмовив, на елемент з ремонтного запасу. Для цього бажано при синтезі такої двовимірної структури розташовувати в одному стовпці однотипні примітиви, що зменшить витрати на кількість запасних елементів.

Інноваційна ідея матриці кубітних примітивів характеризується реалізацією комбінаційних Q-покриттів на адресованих: координатах кубіта і

елементах пам'яті, м'яко з'єднаних в цифрову схему за допомогою вектора станів ліній, що дає можливість ремонтувати логічні примітиви, які відмовили, в реальному часі за допомогою їх переадресації на запасні компоненти при досить високій швидкодії функціонування обчислювального пристрою. Платою за такі переваги є суттєва апаратна надлишковість в порівнянні з жорсткою комбінаційною схемою, яка полягає в додаванні наступних компонентів (рис. 2.5): CU – пристрій управління; Q-ADC – дешифратор адрес примітивів; M-SSV – вектор стану цифрової системи; X – ADC – дешифратор адрес входів; X-M – пам'ять входів; Y-ADC – дешифратор адресів виходів; Y-M – пам'ять виходів.

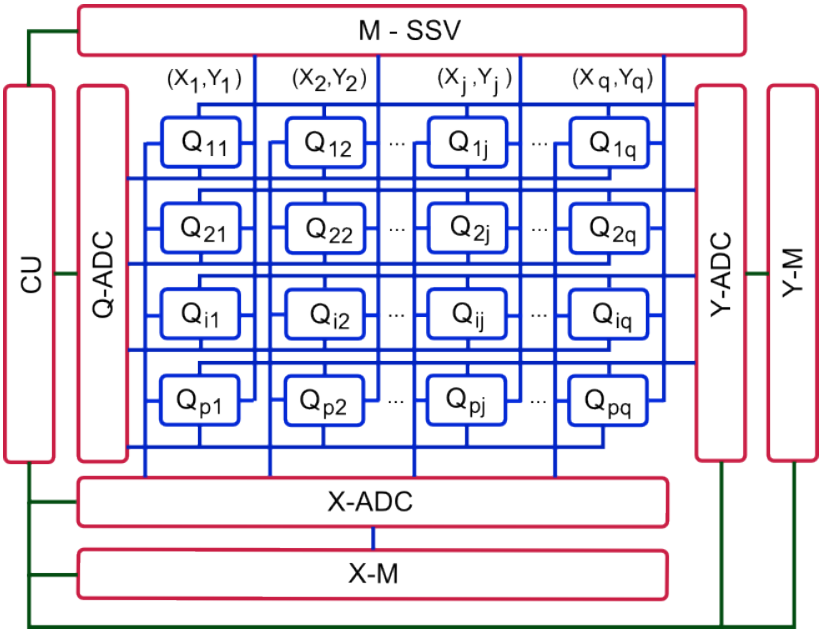


Рисунок 2.5 – Матрична структура операційного пристрою

Спрощення автомата Мура до пари транзакційно взаємодіючих компонентів на основі M-вектора моделювання, адресно структуруючого елементи, призвело до тривіальної моделі MQT-автомата:

$$\begin{cases} A =< M, Q, X, Y >, \\ M(Y_i) = Q_i[M(X_i)]. \end{cases}$$

Прив'язка ідентифікаторів функціональних Q-примітивів до номеру (адреси) вихідної лінії цифрової системи дає можливість спростити MQT-автомат управління ще на одну транзакційну операцію, пов'язану з формуванням адреси стану виходу примітиву Y:

$$\begin{cases} A =< M, Q, X >, \\ M_i = Q_i[M(X_i)]. \end{cases}$$

Тут уже немає системи вихідних змінних, які замінюються координатами М-вектора моделювання ліній цифрового пристрою. Представлений MQT-автомат має особливості:

1) Відсутність жорстких гальванічних зв'язків між елементами, що формують комбінаційну логіку;

2) Наявність «м'яких» або гнучких зв'язків Q-примітивів, які формуються за рахунок використання М-вектора (моделювання) стану цифрової системи (System State Vector - SSV);

3) Часткове зменшення швидкодії обчислювальної структури за рахунок усунення гальванічних зв'язків;

4) Забезпечення нової можливості комбінаційної структури, яка полягає у відновленні працездатності системи у разі виникнення відмов у примітивних елементах;

5) Інваріантність реалізації MQT-комп'ютерів по відношенню до середовища використання (віртуальний простір, критичні об'єкти) і субстанції імплементації (тверді кристали і матеріали, рідкі, газоподібні та плазмові форми існування матерії);

6) Структура адресної взаємодії Q-компонентів пам'яті між собою за допомогою М-вектора моделювання (зосередження входів і виходів цифрової системи) нагадує технологію використання всіх пристроїв компанії Apple, коли входом і виходом будь-якого гаджета є кіберпростір (інтернет або хмара даної компанії). Інакше, можна зробити висновок - як на макро, так і на мікро

рівні середовища застосування автоматів вже немає необхідності використовувати зовнішні входи і виходи цифрової системи, яка вже сьогодні орієнтована на жорстку взаємодію з кіберпростором і не становить особливого інтересу як автономний обчислювальний виріб. Замість входів і виходів існує єдиний адресний простір, в даному випадку вектор M, звідки обчислювальна система бере вхідні дані, а після перетворень вона записує туди ж результати або вихідні дані:

$$\begin{cases} A =< M, Q >; \\ M_i = Q_i(M_j). \end{cases}$$

Тому MQT-модель системного опису структур даних подібна відносинам між обчислювальними пристроями (Q-компоненти) та інтернетом (M-пам'ять). Мабуть, це є сьогодення і майбутнє технологічної культури, коли комп'ютер (гаджет) зв'язується з кіберпростором, який є входом і виходом для будь-якого обчислювального пристрою.

7) Що стосується обробки кубітних обчислювальних MQT-структур нижчого рівня, то тут працює проста і ефективна транзакційна процедура – кубіт функціональності, що бере дані з адресного простору M, а потім туди ж записує результат перетворення: $M_i = Q_i[M_j]$.

8) Інакше, пропонована MQT-модель обчислювального осередку кіберпростору містить структурний MQ-автомат адресної м'якої організації взаємодії функціональних примітивів $MQ = \{M, Q, [(M \times Q) \to M] \vee [M_Y = Q(M_X)]\}$ шляхом M-вектора моделювання і MT-автомату управління обробкою кубітних примітивів на основі використання єдиної операції транзакції, яка регламентується характеристичним виразом $M_i = Q_i[M_j]$, що визначає взаємозв'язок адресованих компонентів і процедуру їх транзакційної обробки. Таким чином, запропонований MQT-автомат, використовуючи тільки елементи пам'яті і єдину операцію транзакції, дає можливість

системно проектувати високонадійні обчислювальні й інформаційні сервіси як в реальному, так і у віртуальному світі.

Як приклад далі пропонується аналіз цифрової схеми на одному тестовому наборі, для якої вектор моделювання та кубітна структура подані нижче:

M =

1	2	3	4	5	6	7	8	9	A	B	C
0	0	0	1	1	1	0	0	1	0	1	0

Q_{ij}	1	2	3
1	1 / 2 — 0001 — A	A / 7 — 0111 — 8	8 / 6 — 1110 — B
2	3 / 4 — 0001 — 7	7 / 5 — 0111 — 9	6 / 9 — 1110 — C

Тут сутність аналізу цифрової структури полягає в заповненні вектора моделювання за всіма координатами: 1) Запис у вектор моделювання M за шістьома координатами вхідних змінних двійкового набору 000111. 2) Обробка примітиву з номером Q_{11}, який має вхідні змінні 1 і 2, а вихід, позначений символом A. Для цього формується адреса 00 шляхом конкатенації вмісту комірок 1 і 2 вектора M. Застосовуючи дану адресу до Q-вектору 0111, визначається вміст нульового осередку, рівне нулю, який записується у вектор M за адресою A вихідної змінної оброблюваного кубіта. 3) Повторення процедури аналізу, описаної в пункті 2, до всіх Q-примітивів дає можливість повністю визначити двійковими сигналами координати вектора моделювання: 000111001010.

Апаратна імплементація наведеного та масштабованого прикладу кубітної структури цифрового пристрою на основі використання елементів пам'яті, представлена на рис. 2.6.

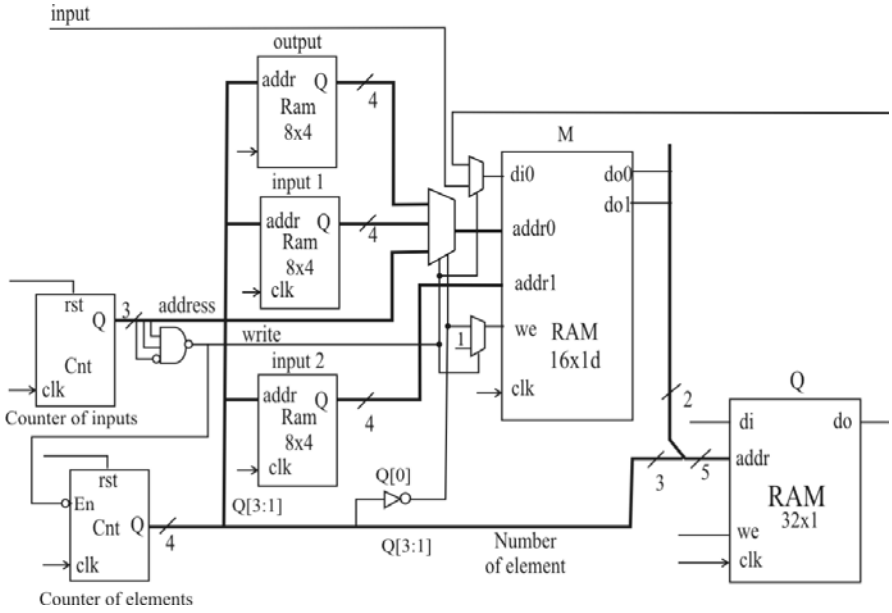


Рисунок 2.6 – Апаратна реалізація кубітної структури комбінаційної схеми

Структура схемної реалізації містить змінні та функціональні елементи, що мають наступне призначення: input – вхід для послідовного занесення вхідних значень вектора M; rst – загальний скид системи, в даному випадку лічильників; clk – вхід синхронізації; counter of inputs – лічильник заповнення вхідних координат вектора M; counter of element – лічильник номера оброблюваного примітиву, який надає два такти для зчитування вхідного набору з двох координат вектора M; Q [3: 1] – шина номера оброблюваного примітиву; Q [0] – змінна режиму зчитування вхідних значень з вектора M або запису результату в M. Блоки пам'яті: Ram 8x4 output – зберігає номери вихідних ліній примітивів; Ram 8x4 input 1 і Ram 8x4 input 2 – зберігають номери вхідних ліній примітивів. Ram 16x1d – двопортова пам'ять зберігання вектора моделювання M, де addr0 - адреса входу 1 при значенні 00 на входах управління мультиплексора, адреса запису результату при значенні 01 на входах управління мультиплексора, адреса для ініціалізації вхідних даних при значенні 1X на входах управління мультиплексора; addr1 – адреса входу 2 оброблюваного примітиву; di0 – вхід даних пам'яті при обробці примітиву (MUX = 1) або зовнішній вхід input при ініціалізації вхідних даних (MUX = 0); we – дозвіл запису в вектор M; do0 – вихід, відповідний входу addr0; do1 –

вихід, відповідний входу addr1. Компонент RAM 32x1 призначений для зберігання Q-векторів завдання функціональностей комбінаційної схеми: di – вхід даних, може бути використаний для ініціалізації (запису) структури кубітів; addr – [4: 0]: addr [4: 2] – номер елементу, addr [1: 0] – вхідний набір для примітиву. Складність апаратної реалізації прикладу комбінаційної схеми складає 150 вентилів, що включають 20 LUT системи елементів компанії Xilinx Spartan 3E. Швидкодія функціонування або формування вектора моделювання дорівнює 180 ns.

У загальному випадку модель функціонування цифрової структури спрощується до обчислення двох адрес при формуванні вектора моделювання

$$M_i = Q_i[M(X_i)]$$

шляхом вилучення складної адреси виходу примітиву в процесі запису станів виходів у координати М-вектора. Інакше, першою виконується процедура конкатенації станів бітів М-вектора, відповідних номерам вектора вхідних змінних Xi. Потім, за допомогою бінарного вектору зконкатенованих бітів, який є адресою, зчитується відповідний біт інформації з функціонального кубіт-вектора Qi. Зчитаний з кубіта біт заноситься у вектор моделювання М за адресою i. М-вектор може мати координати з символами X, що дає можливість виконувати троїчне моделювання цифрових пристроїв для вирішення задач тестування і верифікації. Сказане вище ілюструється наступним аналітичним виразом (k - число вхідних змінних i-примітиву, * - операція конкатенації бітів, A - адреса біта Q-вектора):

$$\left\langle M_i = Q_i(A) \right\rangle \leftarrow \left\langle A = \overset{k}{\underset{j=1}{*}} M(X_{ij}) \right\rangle \leftarrow \left\langle \begin{matrix} M \\ X_i \\ Q_i \end{matrix} \right\rangle.$$

Виходячи з характеристичного рівняння адресно-автоматної моделі цифрової системи, можна зробити висновок, що сучасний (віртуальний) комп'ютер <MQT> слід представляти як адресну організацію структури функціональних примітивів пам'яті без гальванічних або дротових зв'язків, на яких визначені адресні транзакції даних в часі і просторі для досягнення поставленої мети.

Що стосується опису послідовних примітивів (тригери, регістри, лічильники), то їх моделі також можна представляти Q-покриттями або кубітнимі векторами, які мають псевдозмінні для завдання внутрішнього стану. Наприклад, функціональний опис SR-тригера трансформується в квантовий примітив, заданий Q-покриттям, а потім реалізується на адресованому елементі пам'яті FPGA з діаграмами перевірки, що зображено на рис. 2.7.

S	R	Q^X	Q
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$\begin{pmatrix} S = a \\ R = b \\ Q^X = c \end{pmatrix} (11110001)(Q = f)$$

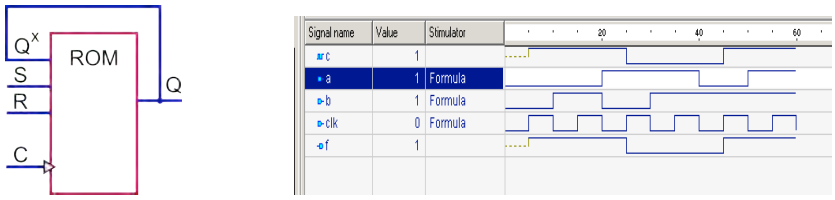


Рисунок 2.7 – Реализація SR-тригера на елементі пам’яті

Таблиця істинності тригера може бути зображена у формі вектора вихідних станів, який записується в елемент постійної пам'яті, що має три адресних входи, сигнал синхронізації, а також зворотний зв'язок, який з'єднує

вихід елемента пам'яті з одним адресним входом. HDL-код моделі тригера для синтезу та верифікації представлений наступним лістингом:

```
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_unsigned.all;
entity model_RS_flip-flop is
port(
a : in STD_LOGIC;
b : in STD_LOGIC;
clk : in STD_LOGIC;
f : out STD_LOGIC
);
end model_RS_psevdo;

architecture model_RS_psevdo of model_RS_psevdo is
constant func: std_logic_vector(0 to 7):= "11110001";
-- DV-trigger constant func: std_logic_vector(0 to 7):= "01000111";
signal c: STD_LOGIC;
begin
process(clk)
variable temp:integer;
begin
temp:=conv_integer(a&b&c);
if clk='1' and clk'event then
c <= func(temp);
end if;
end process;
f <= c;
end model_RS_psevdo;
```

HDL-реалізація в системі проектування Active HDL 9.1 (Aldec Inc.), а також результати верифікації синтезованого SR-тригера (див. рис. 2.7) підтверджують коректність схемотехнічного розв’язку.

Інший приклад пов'язаний з синтезом на елементі постійної пам'яті синхронного DV-тригера. Таблиця істинності тригера трансформована в вектор вихідних станів $Q(D,V,Q^X) = (01000111)$, який записується в елемент пам'яті, що має три адресних входи, сигнал синхронізації, а також зворотний зв'язок, який з'єднує вихід примітиву пам'яті з одним адресним входом. Всі згадані компоненти, включаючи часові діаграми верифікації HDL-коду моделі DV-тригера, представлені на рис. 2.8.

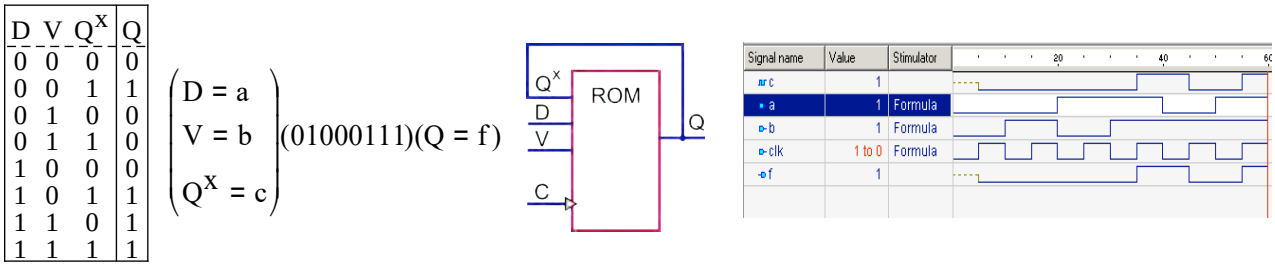
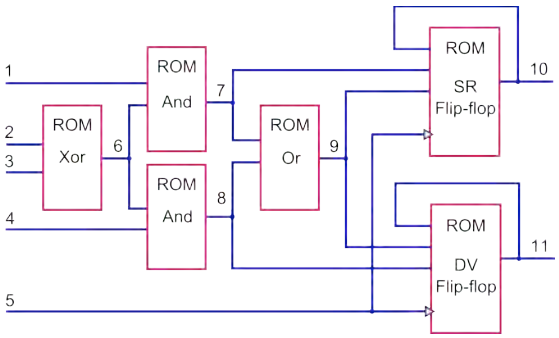


Рисунок 2.8 - Реализація DV-тригера на елементі пам’яті

На рис. 2.9 представлена схема з тригерами і комбінаційною логікою, яка також описана у вигляді елементів пам'яті, куди занесені вихідні стани таблиці істинності кожного логічного елемента. Структури даних, необхідні для моделювання цифрового пристрою, зведені в таблицю, де основними компонентами є: M - вектор моделювання або стану занумерованих ліній, який в даному випадку має 5 вхідних, 6 внутрішніх та вихідних ліній, стани яких підлягають визначенню; X - вектор номерів вхідних ліній для кожного примітиву, які необхідні для формування адреси з метою отримання по ньому стану виходу елемента Qi, функціональність якого задається Q-вектором.



L	1	2	3	4	5	6	7	8	9	10	11
M	0	0	1	1	1	1	0	1	1	1	1
X	.	.	.	.	.	(2,3)	(1,6)	(6,4)	(7,8)	(10,7,9)	(11,9,8)
Q	.	.	.	.	.	0	0	0	0	1	0
	.	.	.	.	.	1	0	0	1	1	1
	.	.	.	.	.	1	0	0	1	1	0
	.	.	.	.	.	0	1	1	1	1	0
	.	.	.	.	.	.	.	.	.	0	0
	.	.	.	.	.	.	.	.	.	0	1
	.	.	.	.	.	.	.	.	.	0	1
	.	.	.	.	.	.	.	.	.	1	1

Рисунок 2.9 – Memory-based комбінаційна схема з тригерами

У процесі моделювання адресно витягнутий стан осередку Q-покриття заноситься до розряду вектора моделювання M за адресою i. Результати обробки всіх Q-векторів схемної структури дозволили сформувати стани ліній M-вектора (6 - 11). Початкові стани невизначеностей на псевдовходах функціональних примітивів довизначаються сигналами нуля або одиниці у залежності від внутрішньої технологічної культури компанії, що надає промислові засоби моделювання та верифікації. Кількість вхідних змінних примітиву q пов'язано з довжиною Q-вектора співвідношенням: $\mathrm{card}(Q) = 2^q$. Правильність роботи пристрою на основі його HDL-опису була також верифікована за допомогою засобів моделювання Active HDL 9.1 (Aldec Inc.). Особливість структурно-функціонального завдання цифрової системи полягає в поданні всіх примітивів елементами пам'яті, куди записуються Q-вектори вихідних станів.

Висновки: 1) Будь-які структурні компоненти обчислювальних пристроїв, комбінаційних і/або послідовних, а також системи в цілому можна описувати кубітними Q-векторами і реалізовувати в елементах пам'яті FPGA, CPLD або VLSI. Це надасть ринку електронних технологій можливість не використовувати reusable логіку при синтезі обчислювальних пристроїв, яка складає сьогодні всього 6 відсотків апаратури SoC (решта 94% - пам'ять), але доставляє розробникам 90 відсотків проблем, пов'язаних з тестуванням, верифікацією і вбудованим дистанційним ремонтом жорсткої провідної

реалізації цифрових виробів. 2) Memory-based інтерпретативне адресно-орієнтоване моделювання комбінаційних і послідовних примітивів цифрових пристроїв стає порівнянним за швидкодією з компілятивним аналізом дискретних об'єктів. Крім того, стає можливим реалізовувати на програмованих логічних пристроях апаратне моделювання цифрових систем, де комбінаційні і послідовні функціональні примітиви будуть подані стандартними елементами пам'яті, в які зашиваються Q-вектори або кубітні покриття.

Q-метод проектування кубітного покриття комбінаційної схеми (без таблиць істинності логічних елементів). Синтез Q-покриття схемної структури на основі примітивів, заданих Q-векторами зводиться до отримання узагальненого покриття-вектора шляхом декартово-адресного координатного виконання логічної операції над розрядами кубітних векторів. Декартова процедура для двох чотирьохрозрядних кубітів, які суперпозіціонуються логічною операцією (or, and, xor), представлена в наступній таблиці:

∨,∧,⊕	b(0)	b(1)	b(2)	b(3)
a(0)	c(0) = a(0) ∨ b(0)	c(1) = a(0) ∨ b(1)	c(2) = a(0) ∨ b(2)	c(3) = a(0) ∨ b(3)
a(1)	c(4) = a(1) ∨ b(0)	c(5) = a(1) ∨ b(1)	c(6) = a(1) ∨ b(2)	c(7) = a(1) ∨ b(3)
a(2)	c(8) = a(2) ∨ b(0)	c(9) = a(2) ∨ b(1)	c(10) = a(2) ∨ b(2)	c(11) = a(2) ∨ b(3)
a(3)	c(12) = a(3) ∨ b(0)	c(13) = a(3) ∨ b(1)	c(14) = a(3) ∨ b(2)	c(15) = a(3) ∨ b(3)

Приклади, що використовують логічні суперпозиції двох кубітів для отримання Q-покриття схемних структур

$$c_1 = (a_1 \wedge a_2) \vee (b_1 \vee b_2), c_2 = (a_1 \wedge a_2) \wedge (b_1 \vee b_2), c_3 = (a_1 \wedge a_2) \oplus (b_1 \vee b_2),$$

подані наступною таблицею:

a(and) =	0001
b(or) =	0111
$c_1 = a(and) \vee b(or)$	0111011101111111
$c_2 = a(and) \wedge b(or)$	0000000000000111
$c_3 = a(and) \oplus b(or)$	0111011101111000

Тут побудовані Q-покриття трьох схем, що складаються з трьох елементів кожна, де два логічних примітиви суперпозиціно об'єднуються третім елементом (or, and, xor). В результаті виходять три вектора, кожен з яких має розмірність в 16 біт. Обчислювальна складність процедури синтезу Q-покриття комбінаційної схеми дорівнює добутку довжин Q-векторів p примітивів, що входять до неї: $\eta = \prod_{i=1}^{p} \mathrm{card}(Q_i)$.

Більш складною є проблема синтезу Q-покриття схеми, вхідні лінії якої мають гальванічні або дротяні з'єднання (тут за змінної a_2): $c = (a_1 \wedge a_2) \vee (a_2 \vee a_3)$. В даному випадку після синтезу Q-покриття схеми необхідно виконати його верифікацію щодо існування суперечливих адрес на змінних a_2 з метою мінімізації Q-вектора шляхом подальшого вилучення згаданих адрес з розгляду, що зменшує розмірність Q-покриття до $\mathrm{card}(Q) = 2^q$ координат, де q – загальне число вхідних змінних схеми:

Q =	0111 0111 0111 1111
a_1 =	0000 0000 1111 1111
a_2 =	0000 1111 0000 1111
a_2 =	0011 0011 0011 0011
a_3 =	0101 0101 0101 0101

=

Q =	0111 0111 0111 1111
a_1 =	0000 0000 1111 1111
a_2 =	00xx xx11 00xx xx11
a_2 =	00xx xx11 00xx xx11
a_3 =	0101 0101 0101 0101

=

Q =	0111 0111
a_1 =	0000 1111
a_2 =	0011 0011
a_2 =	0011 0011
a_3 =	0101 0101

=

Q =	0111 0111
a_1 =	0000 1111
a_2 =	0011 0011
a_3 =	0101 0101

Процедура синтезу Q-покриття: 1) будується таблиця відповідності адрес розрядам Q-вектора схеми, 2) далі суперечливі координати за двома рядками a2 відзначаються символами x, 3) потім всі стовпці з даними символами вилучаються з таблиці, 4) після чого виходять два ідентичних

рядки a2, які об'єднуються в один, 5) що дає в результаті Q-вектор комбінаційної схеми, але вже істотно меншої розмірності. Переваги запропонованого Q-методу синтезу обчислювальних пристроїв, які полягають в компактності їх опису Q-векторами і високій швидкодії адресного моделювання логічних елементів, що утворюють умови для розвитку ринково привабливої «квантової» теорії проектування цифрових систем на кристалах, яка використовує векторно-кубітну форму завдання структурних компонентів.

Аналіз кодів адресного простору. Питання - чи можна синтезувати кубіт-вектор схеми без явного завдання адресного простору? Щоб відповісти на дане запитання необхідно навчитися мінімізувати або зменшувати розмірність Q-вектора залежно від гальванічних з'єднань вхідних ліній (суттєвості вхідних змінних). У такій постановці стану Q-вектора не впливають на формування нового зменшеного адресного простору, а отже, на розмірність самого Q-вектора. Вона залежить тільки від фактичного гальванічного з'єднання вхідних змінних між собою, які накладають обмеження, пов'язані з непротиріччям сигналів на з'єднаних змінних. Тому правило мінімізації адресного простору полягає в усуненні адресних кодів, які створюють протиріччя за з'єднаними змінними. Нехай є Q-вектор схеми і його адресний простір, де змінні b, c, d (a, b, c) з'єднані гальванічно. Нижче наведені таблиці перетворення або мінімізації адресного простору з метою отримання зменшеного Q-вектора:

Q =	0111 0111 0111 1111
a =	0000 0000 1111 1111
b =	0000 1111 0000 1111
c =	0011 0011 0011 0011
d =	0101 0101 0101 0101

Q =	0xxx xxx1 0xxx xxx1
a =	0000 0000 1111 1111
b =	0xxx xxx1 0xxx xxx1
c =	0xxx xxx1 0xxx xxx1
d =	0xxx xxx1 0xxx xxx1

=

Q	0101
a =	0011
b =	0101

Q =	0111 0111 0111 1111
a =	00xx xxxx xxxx xx11
b =	00xx xxxx xxxx xx11
c =	00xx xxxx xxxx xx11
d =	0101 0101 0101 0101

=

Q	0111
a =	0011
d =	0101

По-перше, тут слід зазначити, що в таблицях спостерігається дзеркальна осьова симетрія з інверсією сигналів на координатах адресного простору, яка створює властивість, що описується наступним виразом: $L \oplus R = 1 \rightarrow L_{ij} \oplus R_{ij} = 1$. Дана обставина слід використовувати для зменшення розмірності аналізованого простору в два рази і відповідного зниження обчислювальної складності задачі синтезу квантової вектор-функціональності цифрової схеми. По-друге, кількість різних варіантів взаємодій на q вхідних змінних, пов'язаних з гальванічним з'єднанням різних сполучень вхідних ліній, визначається функціональною залежністю, значення якої знаходяться в інтервалі: $card(Q) = [2^q - 3^q]$. Тим не менш, є ефективна процедура для мінімізації розмірності Q-вектора шляхом виявлення суперечностей в кодах-стовпчиках, на координатах (Aij), відповідних гальванічно пов'язаним w-змінним за j-параметром. Таку процедуру досить виконати на половині адресного простору $card(Q) = 2^q / 2$, а інша частина суперечливих стовпців віддаляється відповідно до дзеркального відображення номерів тих стовпців, які були видалені з першої половини таблиці кодів адрес:

$$\{Q_i, Q_{2^q - i}\} = \varnothing \leftrightarrow (\bigwedge_{j=1}^{w} A_{ij}) \oplus (\bigvee_{j=1}^{w} A_{ij}) = 1,\ i \le 2^q / 2.$$

Якщо в стовпці Ai на групі з w пов'язаних змінних зафіксовано, що кон'юнкція їх станів дорівнює нулю, а диз'юнкція має значення одиниці, то i-стовпець і його дзеркальне відображення $2^q - i$ відаляються з адресного простору A, что автоматично призводить до вилучення з Q-вектора двох отриманих $\varnothing$-координат (в таблицях позначені символами x), відповідним даним стовпцям.

Природно, що також спостерігається симетрія простору векторів-відстаней за Хемінгом, отриманих шляхом xor-взаємодії між сусідніми рядками таблиці адресного простору, для яких суперпозиція лівої і правої частин дає результат $L \oplus R = 0 \rightarrow L_{ij} \oplus R_{ij} = 0$:

Q =	0111011101111111
a ⊕ b	0000111111110000
b ⊕ c	0011110000111100
c ⊕ d	0110011001100110
d ⊕ a	0101010110101010

= (L,R); (L ⊕ R) =

Q =	0111011101111111
a ⊕ b	00000000
b ⊕ c	00000000
c ⊕ d	00000000
d ⊕ a	00000000

↔ $(L = \overline{R})$

Чи доцільно мінімізувати логічну функцію, описану квант-вектором? Відповідь: мінімізація Q-векторів для отримання нормальних або дужкових форм не має практичного значення, істотно тільки зменшення розмірності вектора функціонального опису, що може бути лише наслідком визначення неістотності деяких вхідних (адресних) змінних. Тим не менш, існує проблема розбиття квант-вектора на складові частини меншої розмірності, що пов'язано з імплементацією функціональності в конструктивні компоненти LUT FPGA. У цьому випадку виконується розбиття Q-вектора на два рівних підвектора Q = (L, R), які з'єднуються в структурно-адресну організацію функціональності за допомогою мультиплексора $Q = (\bar{a} \wedge L) \vee (a \wedge R)$. Якщо змінна мультиплексування a=0, то функціональність Q формується за допомогою осередків лівого L-вектора, в іншому випадку, коли a=1, значення функції Q формується бітами правого R-вектора. Алгоритми розбиття та імплементації складних логічних функцій є в кожній промисловій системі синтезу, моделювання та верифікації компонентів SoC.

Отже, запропоновано нову модель технології проектування комп'ютера, яка характеризується: 1) використанням елементів пам'яті для реалізації транзакційної взаємодії всіх компонентів операційного та керуючого

автоматів, 2) концепцією синтезу та аналізу, заснованою на суперпозиції кубіт-векторних примітивів завдання функціональностей, імплементуємих в елементи пам'яті, що дає можливість суттєво підвищити швидкодію засобів моделювання та верифікації, а також значно спростити процедури створення віртуальних хмарних комп'ютерів.

2.5 MQT–структури для майбутніх досліджень

Введення MQT-автомата для синтезу та аналізу обчислювальних сервісів дозволяє зробити і більш загальні висновки. Світ має дуалізм матеріально-інформаційної цілісності, а отже детермінованої відтворюваності об'єктів, процесів і явищ. Будь-який матеріальний об'єкт здатний зберігати інформацію, отже, його можна використовувати для запису і зчитування, а відтак - для створення обчислювальної системи, нехай навіть в деякій мірі специфічною. Людина є біологічний комп'ютер із заданою програмою функціонування (життєдіяльності) всіх його органів в часі. Як будь-яку програму, код (геном) функціонування людини можна вважати-записати, а значить – можна і скорегувати. Будь-які об'єкти на планеті мають свої програми існування в часі – інформаційні геноми. Процеси і явища на планеті також мають свої програми розвитку, але більш високого рівня ієрархії в управлінні. Всесвіт також має власний геном – інформаційну програму розвитку. Природно, що будь-яку програму можна вважати-записати, а отже скорегувати – це оптимістична нота інформаційної моделі об'єктів, процесів і явищ матеріального світу! Декодувати програму, як і аналізувати двійковий код, складно, але можливо. Для цього потрібно визначити місце – носій коду і спосіб шифрування кожного об'єкта, процесу або явища природи. Можна припустити, що геном біологічного об'єкта має форму, подібну структурі багатозначних кубіт-векторів, коли кожен з них $Q=(q_1,q_2,...,q_i,...,q_n)$, $q_i \in \{a_1,a_2,...,a_j,...,a_k\}$ може бути трансформований в двійкову матрицю, де символ алфавіту опису розрядів кванта може бути

розгорнутий в двійковий вектор або код-стовпець. Багатозначність функціональності досить поширена в природі, зручна для сприйняття оком людини, але для комп'ютера символи будь-якого алфавіту поки трансформуються в двійкові вектори або коди. Геном людини з позиції комп'ютерної інженерії є структура даних, записана в біопамять, компактна програма розвитку та існування в часі і просторі білкових конструктивів, яка має апарат управління-моніторингу (зчитування-запису) без існування жорстких гальванічних зв'язків. Такі принципи біоінженерії слід застосовувати і до об'єктів кіберпростору, щоб забезпечити їх надійність, ремонтопридатність, відновлюваність на мікро і макрорівнях.

Таким чином, кубітні структури дають можливість зробити з комбінаційної схеми найпростіший автомат (інтегруюча пам'ять, кванти функціональностей, операція транзакції) і перейти від програмного моделювання цифрових систем до апаратної емуляції структур і процесів, що становить вже користувальницькі функції комп'ютера, інваріантні по відношенню до технологій імплементації. Показовим аналогом і прообразом служить апаратний прискорювач процесів моделювання PRUS [60] доктора Stanley Hyduke, Aldec Inc., орієнтований на зменшення часу проектування та верифікації цифрових систем на кристалах. Але в порівнянні з [60] тут пропонується використовувати за прямим функціональним призначенням процесор «м'якого» адресно орієнтованого апаратного моделювання кубітних структур як обчислювального виробу, що доставляє сервіси споживачеві. При цьому зберігається висока швидкодія функціонування пристрою, доповнена істотною для критичних систем можливістю відновлення працездатності в режимі реального часу.

Технологічна сингулярність (Вернор Віндж) – вибух в розумінні законів всесвіту на короткому проміжку часу і створення кібермозоку людства з метою підвищення якості життя очевидно станеться завдяки розвитку трьох компонентів: біоінженерії, штучного інтелекту кіберпростору і нанотехнологій проектування. Тут мається на увазі: 1) вбудована і

безпосередня інтеграція людини з комп'ютером (кіберпростором), шляхом усунення мовних бар'єрів і інтерфейсів між ними; 2) створення штучного інтелекту для самонавчання, по-друге, і самовдосконалення, по-перше, коли комп'ютери і кіберпростір будуть здатні до самовідтворення більш досконалих моделей, структур і процесів; 3) «нано-вирощування» комп'ютера шляхом структурування атомів. Однією з головних умов досягнення даної точки розвитку технологічної культури є створення принципово нового комп'ютера (обчислювального процесу) інваріантного по відношенню до будь-якої точка простору або його субстанції, яку можна структурувати і використовувати замість кремнію. Просторовий паралелізм існуючих комп'ютерів і точкова одночасна багатозначність – два шляхи в розвитку інтелектуального кіберпростору. Інакше, комп'ютер можна зробити на чому завгодно, що здатне зберігати в часі та / або в просторі не менше двох керованих і моніторюваних станів. Регістр або пам'ять, як основа для зберігання даних і виконання обчислювальних процедур, являє собою кілька фізичних точок або тригерів, до яких можна застосувати паралельну операцію, рознесену в просторі. При цьому підвищення продуктивності цифрового пристрою сьогодні здійснюється тільки за рахунок розширення субстанції обчислювальної структури. Запитання - чи можна об'єднати згадані точки простору в одну для виконання паралельних операцій над сукупністю даних, зосереджених в одній матеріальної точці? Можливо так і ставилося практичними вченими питання, яке передбачило появу квантового комп'ютера. Безсумнівно, що майбутнє комп'ютерної індустрії має бути пов'язане з багатоваріантністю фізичної точки простору, як примітиву структур даних, яка повинна мати можливість зберігання булеана станів як операнд для реалізації обчислювальних процедур. Але тут виникає тільки одне рішення, варте уваги. Все, що фізично є стабільним в часі, може мати тільки один стан у кожний конкретний момент часу, включаючи цифрові дискретні автомати, пристрої та системи. Рішення очевидно – дуалізм частки і хвилі на всьому спектрі електромагнітного випромінювання має шукані

властивості просторової точки: 1) Кожна точка простору може мати в конкретний момент часу множину хвиль або квантів з відомих і невідомих людству діапазонів; 2) Це означає можливість наявності в точці простору множини часток відповідних діапазонів, які формують сукупність інформаційних станів даної точки; 3) Невизначеність (багатоваріантність) у часі процесів і явищ є ключ до створення принципово нових комп'ютерів, які можна називати як квантовими, так і електромагнітними, враховуючи дуалізм електродинаміки відносно носія (частка або хвиля); 4) На кожній стадії розвитку технологічної культури необхідно вибирати підходящу точку простору або субстанції (газоподібна, рідка, тверда, плазмова), яку можна використовувати для моніторингу та управління її багатозначних станів за допомогою існуючих технологій. Пророцтва фантаста Станіслава Лема і вченого В.І. Вернадського про можливості існування розуму в рідкій (океан Соляріс) і газоподібному середовищі (ноосфера) сьогодні вже близькі в часі до створення глобального штучного розуму кіберпростору, який ми повинні отримати до 2050 року; 5) Дискретизація спектру частот у кожній точці простору може служити основою для кодування як примітивів (двійкових, багатозначних), так і більш складних асоціацій, наприклад, булеана станів. Інакше, 8-розрядний регістр, має 256 станів, можна уявити однією точкою в просторі для подальшої обробки на основі моніторингу та управління відповідними квантами або спектром частот. 6) Якщо припустити, що кванти існують у всіх діапазонах безперервного і нескінченного спектра, то питання полягає в тому, який діапазон частот в точці простору чи субстанції буде освоєно перший для створення нових як потужних, так і простих квантових електродинамічних обчислювачів на основі економічно прийнятних технологій моніторингу та управління (атомними примітивами) частками або хвилями. 7) Ринкова привабливість «зелених» нанотехнологій («висхідне» проектування) – побудова або вирощування обчислювача (квантового комп'ютера) шляхом структурування атомів – полягає в безвідходності, мікроміниатюрності, наднизькому енергоспоживанні, абсолютно мінімальної

витратності матеріалів, а в майбутньому і вартості, надвисокій швидкодії і необхідної (за принципом розумної достатності) масштабованості, сумірною з класом доставлюваних сервісів. Мікротехнології («низхідне» проектування) – суть яких: взяти заготовку і відсікти все зайве, стають все менш привабливими для ринку, через недосконалість якості та надвитратності створюваних виробів.

Вчені навчилися сьогодні не тільки сканувати атомні структури, а й послідовно будувати чи вирощувати їх. Однак на шляху вирішення проблеми нано-технологічного напрямку розробки квантового комп'ютера на ринку є три привабливих пункти: 1) Відкриття технологій з високою швидкодією вирощування необхідних гетерогенних атомних структур відповідно до заданої програмної специфікації обчислювача; 2) Створення ефективного транзакційного механізму для реалізації простого моніторингу та управління квантовими станами вирощених атомних обчислювальних структур з адресованими компонентами; 3) Забезпечення необхідної стабільності в часі станів компонентів атомної структури, що реалізує пам'ять; 4) Для вирішення проблеми необхідно об'єднати досягнення і зусилля вчених, що працюють в областях: квантової фізики атомних взаємодій, біоінженерії, мікроелектроніки, електрохімії та комп'ютерної інженерії.

2.6 Висновки до розділу 2

1. Запропоновано кубітні моделі опису цифрових систем і компонентів, які характеризуються компактністю опису таблиць істинності у формі Q-покриттів завдяки унітарному кодуванню вхідних станів, що дає можливість підвищити швидкодію програмних і апаратних засобів інтерпретативного моделювання обчислювальних пристроїв за рахунок адресної реалізації аналізу логічних примітивів.

2. Представлена матрична модель кубітних примітивів для реалізації комбінаційних схем, яка характеризується адресним об'єднанням Q-покриттів на елементах пам'яті, «м'яко» з'єднаних в цифрову схему за допомогою вектора станів ліній, що дає можливість ремонтувати логічні примітиви, що відмовили, в реальному часі за допомогою їх переадресації на запасні компоненти при досить високій швидкодії функціонування обчислювального пристрою.

3. Введена гнучка автоматна MQT-модель комп'ютера, яка характеризується використанням тільки адресованих структур пам'яті і операції транзакції для програмної та апаратної реалізації комбінаційних і послідовних функціональностей, що дає можливість створювати швидкодіючі і надійні обчислювачі для створення сервісів кіберпростору на основі паралельних логічних операцій і ремонту несправних адресованих функціональних примітивів.

4. Описана інноваційна ідея квантових обчислень, яка характеризується переходом від обчислювальних процедур над байт-операндом, визначальним у дискретному просторі одне рішення (точку) до логічних регістрових паралельних процесів над кубіт-операндом, одночасно формує булеан рішень, що дає можливість визначити нові перспективи на шляху створення високопродуктивних комп'ютерів паралельного аналізу і синтезу структур і сервісів дискретного кіберпростору.

5. Представлена концепція системного проектування програмних і апаратних хмарних сервісів кіберпростору на основі MQT-автомата, яка характеризується застосуванням транзакцій на кубітних адресно пов'язаних компонентах обчислювача за допомогою вектора станів (моделювання), що дає можливість програмувати логічні функціональності в кубітних примітивах іінтерпретативної структури цифрового пристрою.

6. Ринкова привабливість MQT-комп'ютера визначається: примітивізмом його реалізації в програмному і апаратному виконанні; високим рівнем використання пам'яті в структурі сучасних обчислювачів, що досягає 94% від площі кристала; і як наслідок - зменшенням впливу комбінаційної логіки на швидкодію системи в цілому; підвищенням надійності комп'ютерів за рахунок онлайн-ремонту адресованих елементів пам'яті, включаючи логічні примітиви. Існуюче неприйняття MQT-комп'ютера ринком пов'язано з лобіюванням обчислювачів на основі жорсткої комбінаційної логіки в VLSI-проектах з боку компаній, що розробляють процесори на основі технологій reusable logic.

7. Напрями майбутніх досліджень: синтез кубітних моделей цифрових систем на основі використання Q-покриттів, а також розкладання (аналіз) функциональностей на кубітние компоненти; моделювання несправностей і синтез тестів на основі використання Q-покриттів компонентів цифрових систем; розробка спеціалізованих програмних додатків для вирішення завдань синтезу комбінаційних пристроїв на основі використання матричної моделі MQT-процесора, орієнтованого на високий паралелізм вирішення практичних завдань та відновлення працездатності пристрою в реальному часі.

3 «КВАНТОВІ» МОДЕЛІ ДІАГНОСТУВАННЯ ТА МОДЕЛЮВАННЯ ЦИФРОВИХ ПРИСТРОЇВ

Пропонуються теорія і приклади реалізації кубітних моделей, методів і алгоритмів для підвищення швидкодії існуючих програмних і апаратних засобів аналізу цифрових обчислювальних пристроїв за рахунок збільшення розмірності структур даних і пам'яті для одночасного зберігання оброблюваних станів. Подаються результати досліджень, що стосуються моделей і методів діагностування цифрових систем, моделювання справної поведінки, відновлення працездатності примітивів, що відмовили.

3.1 Тенденції розвитку кіберпростору

Еволюція кіберпростору планети умовно розподіляється на такі періоди: 1) 1980-і роки – формування парку персональних комп'ютерів; 2) 1990-і роки – впровадження Інтернет-технологій у виробничі процеси і побут людини; 3) 2000-і роки – підвищення якості життя за рахунок впровадження мобільних пристроїв і хмарних сервісів; 4) 2010-і – створення цифрової інфраструктури моніторингу, управління і взаємодії між собою стаціонарних і рухомих об'єктів (повітряний, морський, наземний транспорт і роботи); 5) 2015-і роки – створення глобальної цифрової інфраструктури кіберпростору, де всі процеси, явища ідентифікуються в часі і в тривимірному просторі і стають інтелектуальними. У зв'язку з необхідністю розвитку паралельних обчислень для неупорядкованих даних в останні роки стають все більш значущими кубітні структури для створення хмарних Інтернет сервісів, завдяки їх позитивної альтернативності існуючим витратним за часом класичним моделям послідовної обробки теоретико-множинних структур за рахунок істотного розширення пам'яті [55]. Однак така плата в даний час цілком припустима, оскільки ринок нано-електронних технологій надає сьогодні розробникам цифрових систем до 1 мільярда вентилів на кристалі

розмірністю 2х2 см при товщині пластини в 5 мікрон. При цьому сучасні технології допускають створення пакета або «сендвіча», що містить до 7 кристалів. Практично «бездротове» з'єднання таких пластин ґрунтується на технологічній можливості свердління порядку 10 тисяч наскрізних отворів (vias) на 1 квадратному сантиметрі. Крім того, поява тривимірних FinFETs транзисторів і заснованих на них 3D-технологій реалізації об'ємних цифрових систем надають нові можливості для створення більш швидкодіючих за рахунок зменшення затримок паралельних обчислювальних пристроїв [60, 62, 63, 66, 67]. Тому можна і потрібно використовувати «жадібні» до апаратури моделі і методи для створення швидкодіючих засобів паралельного вирішення практичних завдань. Маючи на увазі дискретність і багатозначність алфавітів опису інформаційних процесів, властивість паралелізму (одночасності процесів), що закладене до квантових обчисленнь, є затребуваним при створенні ефективних та інтелектуальних «движків» для кіберпростору, хмарних структур і сервісів Інтернету; підвищення надійності цифрових пристроїв; тестування і моделювання дискретних систем на кристалах. Тут, як і вище, не розглядаються фізичні основи квантової механіки, що стосуються недетермінірованої взаємодії атомних частинок, але використовується поняття кубітної структури як векторної форми спільного або одночасного завдання булеана станів в кінцевій і дискретній області кіберпростору, орієнтованого на паралелізм і суперпозицію обробки пропонованих кубітних моделей і методів.

Квантові емулятори на класичних комп'ютерах досить ефективно застосовуються для вирішення оптимізаційних завдань, пов'язаних з повним перебором варіантів розв’язків на основі використання теорії множин [55,58]. Особливість в тому, що множина елементів в комп'ютері завжди є впорядкованою, оскільки кожен біт, байт або інший компонент має свою адресу. Тому всі теоретико-множинні операції, так чи інакше, зводяться до повного перебору адрес примітивних елементів. Адресний порядок структур даних добрий для завдань, де компоненти моделей можна строго ранжувати,

що дає можливість виконувати їх аналіз за один прохід або одну ітерацію. Там, де немає порядку в структурі, наприклад, множина всіх підмножин, класична модель пам'яті і обчислювальних процесів завдає шкоди часу аналізу асоціації рівних за рангом примітивів, або, в кращому випадку, обробка асоціативних груп не є ефективною. Що можна запропонувати для неупорядкованих даних замість строгого порядку? Процесор, де елементарною клітинкою служить образ або шаблон універсуму з n примітивів, який генерує $Q = 2^n$ всіх можливих станів такої комірки у вигляді булеана або множини всіх підмножин. Прямий розв’язок, орієнтований на створення такого осередку, використовує унітарне позиційне кодування станів примітивів, яке за допомогою суперпозиції останніх утворює універсум примітивів, що формує у межі – булеан або множину всіх підмножин [58, 68].

3.2 Кубітний метод діагностування цифрових систем

Пропонується метод діагностування функціональних порушень і константних несправностей в програмних або апаратних блоках, які використовують «кубітні» або багатозначні структури даних для завдання діагностичної інформації, що дає можливість істотно зменшити обчислювальну складність процесів моделювання та діагностування за рахунок введення паралельних логічних операцій над матричними даними. Представлений кубітний метод справного моделювання цифрових пристроїв з відновленням працездатності компонентів цифрової системи в режимі online, який має істотно більш високу швидкодію за рахунок адресної реалізації процедури обробки функціональних примітивів, заданих кубітними векторами станів виходів.

Модель об'єкта діагностування представлена у формі графа цифрової системи, яка має функціональні елементи, з'єднані лініями зв'язків. Серед них є асерція – точки спостереження або моніторингу, необхідні для верифікації,

тестування та діагностування несправностей [61]. Діагностична інформація подана компонентами: 1) Тест перевірки або діагностування несправностей заданого класу, в даному випадку розглядаються поодинокі константні дефекти $\{\equiv 0, \equiv 1\}$ ліній схеми. 2) Таблиця несправностей [62], рядки якої задають вектори, що перевіряються на кожному тестовому наборі дефектів, прив'язаних до ліній схеми. 3) Матриця досяжності, яка визначає досяжність кожної асерційної точки з боку множини попередніх ліній [58]. 4) Матриця стану асерційного механізму або матриця експериментальної перевірки, що задає стан кожної асерції на тестових наборах шляхом порівняння еталонної реакції в даній точці з реальним сигналом в процесі виконання діагностичного експерименту [61, 67].

Базова модель діагностування цифрового виробу, дискретного процесу або явища подана компонентами, які створюють 4 виміри у просторі ознак:

$$\begin{cases} D_b =< S, A, F, T > \\ D = \{< S, A >, < F, T >\}; \end{cases}$$

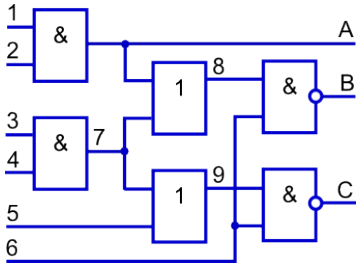
$$\begin{cases} V_b = (|S| \times |A| \times |F| \times |T|); \\ V = (|S| \times |A|) + (|F| \times |T|); \\ V_b >> V; \end{cases}$$

$$\begin{cases} S^* = f(S, A, T); \\ A^* = g(T, A); \\ F^* = h(S, A, F, T); \end{cases}$$

При цьому в моделі обсяг діагностичної інформації V формується декартовим добутком (потужностей) чотирьох компонентів в порядку проходження, зазначеному вище: 1) структура об'єкта; 2) механізм асерцій або моніторингу; 3) сукупність несправностей або модулів, схильних до функціональних порушень; 4) тестові набори або сегменти для діагностування несправностей або сукупності згаданих модулів. Актуально і суттєво зменшити обсяг діагностичної інформації можна шляхом зниження

розмірності простору ознак за рахунок розділення базової моделі на два непересічних підмножини $< S, A >, < F, T >$. У цьому випадку оцінка обсягу діагностичної інформації стає не мультипликативною, а адитивною по відношенню до потужності отриманих в результаті розбиття підмножин без будь-якого зменшення глибини діагностування. Тут перший компонент моделі діагностування зображений матрицею досяжностей, яка дозволяє мінімізувати маску можливих дефектів на основі аналізу структури схеми шляхом порівняння істинних і реальних результатів моделювання вихідних сигналів на кожному тестовому наборі або сегменті. Число рядків такої матриці дорівнює кількості спостережуваних виходів або асерцій.

У процесі виконання методу діагностування створюється двійкова матриця структурної активізації несправностей, яка служить маскою для істотного зменшення множини підозрюваних дефектів при спільному аналізі таблиці несправностей. При цьому символи одиночних константних дефектів $\{0, 1, X, \varnothing\}, X = \{0, 1\}$ у комірках таблиці несправностей [62] кодуються відповідними станами кубіта (10,01,11,00) багатозначного алфавіта Кантора $A^k = \{0, 1, X, \varnothing\}$, що дає можливість виключити з обчислювальних процесів теоретико-множинні процедури, замінивши їх на векторні логічні операції. Для розгляду сутності пропонованого методу використовується фрагмент цифрової схеми, зображеної на рис. 3.1. Тут є три асерційних точки A, B, C для спостереження за станом усіх ліній схеми в процесі тестування (виконання діагностичного експерименту) шляхом подачі п'яти тестових впливів, заданих в таблиці несправностей F(T). Координати даної таблиці задають перевірювані на тест-векторах несправності 0 і 1, а також є стани координат: $\varnothing$ (.) – відсутність перевірюваних дефектів і X – перевірка на лінії константи 0 і 1 одночасно. Права частина таблиці є матриця станів асерційного механізму у вигляді результатів порівняння еталонної і реальної реакцій цифрового пристрою на тестові набори. Значення 1 означає незрівняння, 0 – збіг згаданих реакцій.



T \ F	1	2	3	4	5	6	7	8	9	A	B	C	A_a	A_b	A_c
111101	0	0	0	0	.	0	0	0	0	0	1	1	1	0	0
010101	1	.	1	.	1	.	1	1	1	1	0	0	0	1	1
101001	.	1	.	1	1	.	1	1	1	1	0	0	0	0	0
000011	.	.	.	.	0	0	1	1	0	1	0	1	0	0	0
111110	0	0	.	.	.	1	.	.	.	0	0	0	1	1	1

Рисунок 3.1 – Фрагмент цифрової схеми і таблиця несправностей

У таблиці несправностей не враховується структура схеми для підвищення глибини діагностування на основі обчислення реальної матриці станів асерційного механізму, яка спільно з матрицею досяжності створює структурну маску, що мінімізує множину підозрюваних дефектів. Для фрагмента цифрової схеми, поданої на рис. 3.1, матриця досяжності має наступний вигляд:

$S=\|S_{ij}\|$	1	2	3	4	5	6	7	8	9	A	B	C
1	1	1	.	.	.	.	.	.	.	1	.	.
2	1	1	1	1	.	1	1	1	.	.	1	.
3	.	.	1	1	1	1	1	.	1	.	.	1

Тут виходи-асерції A,B,C є моніторами технічного стану об’єкту діагностування. Кожний з них може мати два значення: $A_{ij}=\{0,1\}$, які визначають матрицю експериментальної перевірки $A=\|A_{ij}\|$ шляхом порівняння еталонних $T=\|T_{ij}\|$ і реальних $U=\|U_{ij}\|$ станів спростережуваних або вихідних ліній: $A_{ij}=T_{ij}\oplus U_{ij}$, які формують маску можливих дефектів за допомогою наступного виразу:

$$S_i=S(T_i)=(\bigvee_{A_{ij}=1} S_{ij})\wedge(\overline{\bigvee_{A_{ij}=0} S_{ij}}).$$

Кожен тест-вектор (-сегмент) активізує власну структуру можливих дефектів, яка функціонально залежить від маски, асерцій (стану спостережуваних виходів) і тестових наборів: $S = f(S, A, T_i)$. Якщо припустити, що в матриці $S = |S_{ij}|$ стани асерційних виходів на першому тест-векторі дорівнюють $A_{1A} = 0; A_{1B} = 1; A_{1C} = 1$, де значення 1 ідентифікує прояв дефекту в пристрої, то маска можливих дефектів, згідно з функціоналом

$$S_1 = S(T_1) = (\underset{A_{1j}=1}{\vee} S_{1j}) \wedge (\overline{\underset{A_{1j}=0}{\vee} S_{1j}}),$$

буде мати наступний вигляд:

$$S_1 = S(T_1) = (S_2 \vee S_3) \wedge (\overline{S_1}) = (111101110010 \vee 001111101001) \wedge (\overline{110000000100}) =$$
$$= (111111111011) \wedge (001111111011) = (001111111011).$$

Отримана маска накладається на перший рядок таблиці несправностей, що визначає множину підозрюваних дефектів $F_i = T_i \wedge S|_{i=1} \rightarrow F_1 = T_1 \wedge S_1$, що формують асерційну вихідну реакцію $A_{1(A,B,C)} = (011)$ пристрою на перший тест-вектор:

Faults	1	2	3	4	5	6	7	8	9	A	B	C
T_1	0	0	0	0	.	0	0	0	0	0	1	1
S_1	0	0	1	1	1	1	1	1	1	0	1	1
$F_1 = T_1 \wedge S_1$	.	.	0	0	.	.	0	0	0	.	1	1

Відповідно до запропонованої процедури отримання маски одного рядка виконується побудова матриці структурної активізації несправностей S(T) на основі використання таблиці експериментальної перевірки $A = |A_{ij}|$,

що задає стан асерційного механізму в процесі виконання тестування $S(T) = S \otimes A$:

$S = \|S_{ij}\|$	1	2	3	4	5	6	7	8	9	A	B	C
1	1	1	.	.	.	.	.	.	.	1	.	.
2	1	1	1	1	.	1	1	1	.	.	1	.
3	.	.	1	1	1	1	1	.	1	.	.	1

$\otimes \rightarrow$

$A = \|A_{ij}\|$	A	B	C
T1	1	0	0
T2	0	1	1
T3	0	0	0
T4	0	0	0
T5	1	1	1

=

S(T)	1	2	3	4	5	6	7	8	9	A	B	C
T1	1	1	0	0	0	0	0	0	0	1	0	0
T2	0	0	1	1	1	1	1	1	1	0	1	1
T3	0	0	0	0	0	0	0	0	0	0	0	0
T4	0	0	0	0	0	0	0	0	0	0	0	0
T5	1	1	1	1	1	1	1	1	1	1	1	1

З метою формування структур даних, зручних для комп'ютерної обробки, необхідно перевести символи таблиці несправностей в двохрозрядні коди згідно з правилами $\triangleright$-кодування: $\triangleright = \{0 = 10; 1 = 01, X = 11, \varnothing = 00\}$, застосування яких до таблиці несправностей F(T) дає наступний результат:

F(T)	1	2	3	4	5	6	7	8	9	A	B	C
T1	0	0	0	0	.	0	0	0	0	0	1	1
T2	1	.	1	.	1	.	1	1	1	1	0	0
T3	.	1	.	1	1	.	1	1	1	1	0	0
T4	.	.	.	.	0	0	1	1	0	1	0	1
T5	0	0	.	.	.	1	.	.	.	0	0	0

$\triangleright \longrightarrow$

F(T)	1	2	3	4	5	6	7	8	9	A	B	C
T1	10	10	10	10	00	10	10	10	10	10	01	01
T2	01	00	01	00	01	00	01	01	01	01	10	10
T3	00	01	00	01	01	00	01	01	01	01	10	10
T4	00	00	00	00	10	10	01	01	10	01	10	01
T5	10	10	00	00	00	01	00	00	00	10	10	10

Після отримання структурної матриці S(T), призначеної маскувати реальні дефекти в таблиці несправностей, і її кодованої форми, необхідно виконати # -суперпозіцію двох матриць: $F(T) = S(T) \# F(T)$, яка зводиться до виконання #-операції над однойменними координатами $F_{ij} = \overline{F}_j \leftarrow (F_j = 00) \vee (S_{ij} = 0)$, що означає модифікацію кодів координат таблиці F (T) при виконанні заданих умов. Інакше, дана операція зводиться до інверсії осередків матриці кодів несправностей, маскованих нульовими сигналами структурної матриці активізації, а також всіх нульових кодів таблиці несправностей. Таблиця істинності даної # -операції в символьному і кодованому вигляді подана нижче:

#= $S_{ij} \setminus F_{ij}$	∅	1	0	X
0	X	0	1	∅
1	X	1	0	X

#= $S_{ij} \setminus F_{ij}$	00	01	10	11
0	11	10	01	00
1	11	01	10	11

Таблиця істинності скоригована щодо інверсії стану 00 в 11 при одиничному значенні сигналу активізації несправності, тому що такий код (00) означає присутність в схемі на лінії порожньої множини перевірюваних дефектів, що неможливо. Але код 00 ще блокує всі обчислення кон'юнкції за стовпцем, перетворюючи результат в 00. Інверсія коду дає можливість не маскувати при логічному множенні дійсно присутніх дефектів, довільних знаків. При цьому передбачається, що неможливо тест-вектором перевірити на одній лінії схеми дефекти різних знаків.

Виконання процедури суперпозиції структурної матриці з кодованою таблицею несправностей $F(T) = S(T) \# F(T)$ дає наступний результат:

S(T)	1	2	3	4	5	6	7	8	9	A	B	C
T1	1	1	0	0	0	0	0	0	0	1	0	0
T2	0	0	1	1	1	1	1	1	1	0	1	1
T3	0	0	0	0	0	0	0	0	0	0	0	0
T4	0	0	0	0	0	0	0	0	0	0	0	0
T5	1	1	1	1	1	1	1	1	1	1	1	1

$\#$
$\rightarrow$

F(T)	1	2	3	4	5	6	7	8	9	A	B	C
T1	10	10	10	10	00	10	10	10	10	10	01	01
T2	01	00	01	00	01	00	01	01	01	01	10	10
T3	00	01	00	01	01	00	01	01	01	01	10	10
T4	00	00	00	00	10	10	01	01	10	01	10	01
T5	10	10	00	00	00	01	00	00	00	10	10	10

=

F(T)	1	2	3	4	5	6	7	8	9	A	B	C
$\wedge$	10	10	01	01	11	01	01	01	01	10	10	10
	10	11	01	11	01	11	01	01	01	10	10	10
	11	10	11	10	10	11	10	10	10	10	01	01
	11	11	11	11	01	01	10	10	01	10	01	10
	10	10	11	11	11	01	11	11	11	10	10	10
$F(T) = \bigwedge_{i=1}^{n} F_i$	10	10	01	00	00	01	00	00	00	10	00	00
F =	0	0	1	.	.	1	.	.	.	0	.	.

На заключній стадії діагностування виконується єдина і векторна операція логічного множення всіх рядків кодованої модифікованої таблиці істинності F(T):

$$F(T) = (\bigvee_{A_i=1} F_i) \wedge (\overline{\bigvee_{A_i=0} F_i}) = (\bigwedge_{A_i=1} F_i) \wedge (\overline{\bigvee_{A_i=0} F_i}) = (\bigwedge_{A_i=1} F_i) \wedge (\bigwedge_{A_i=0} \overline{F}_i) = (\bigwedge_{i=1}^{n} F_i).$$

Це дає можливість точно визначити всі дефекти, присутні в об'єкті діагностування, які представлені в двох нижніх рядках наведеної вище кодованої таблиці несправностей F(T): $F = \{1^0, 2^0, 3^1, 6^1, A^0\}$.

Теоретичний доказ матричного діагностування одиночних і кратних дефектів подано нижче у вигляді двох теорем.

Теорема 3.1. *Одиночні константні дефекти цифрової схеми, задані кубітами на тестових наборах багатозначної таблиці несправностей, визначаються за допомогою векторної and-операції, маскованої по рядках матрицею (вектором) експериментальної перевірки* $A = |A_{ij}|$ *всіх асерційних точок*:

$$F(T) = (\bigvee_{A_i=1} F_i) \wedge (\overline{\bigvee_{A_i=0} F_i}) = (\bigwedge_{A_i=1} F_i) \wedge (\overline{\bigvee_{A_i=0} F_i}) = (\bigwedge_{A_i=1} F_i) \wedge (\bigwedge_{A_i=0} \overline{F}_i) = (\bigwedge_{i=1}^{n} F_i).$$

Вираз є вірним, тому що: 1) Другий співмножник – чиста математика – заперечення диз'юнкції є кон'юнкція заперечень, що означає множення кодів таблиці з їх попередніми запереченням. 2) Перший співмножник орієнтований на пошук несуперечливих дефектів, тому він замінюється на $(\bigwedge_{A_i=1} F_i)$. Дійсно, на одній лінії або змінній не можуть бути присутніми одночасно дві протилежних за знаком перевірюваних несправності. Тому в базовій формулі диз'юнкція дефектів $(\bigvee_{A_i=1} F_i)$ більшою мірою орієнтована на пошук кратних несправностей, але не пов'язаних з однією лінією. Кратність суперечливих дефектів на одній лінії, так само як і інверсія порожньої множини несправностей, теоретично створює умови безперешкодного множення інших комірок стовпчика з метою формування на кожній лінії

результату у вигляді дефекту одного знака або порожньої множини несправностей.

Теорема 3.2. *Кратні константні дефекти цифрової схеми, задані кубітами на тестових наборах багатозначної таблиці несправностей, визначаються за допомогою векторних* or- and- *операцій, маскованих за рядками вектором експериментальної перевірки* A(T) *всіх асерційних точок*:

$$F(T) = (\underset{Ai=1}{\vee} F_i) \wedge (\overline{\underset{A_i=0}{\vee} F_i}) = (\underset{A_i=1}{\vee} F_i) \wedge (\underset{A_i=0}{\wedge} \overline{F}_i).$$

Вираз є вірним, тому що: 1) Другий співмножник є заперечення диз'юнкції або кон'юнкція заперечень, що означає множення кодів таблиці з їх попередніми запереченням. 2) Перший співмножник орієнтований на пошук кратних дефектів у припущенні, що на одній лінії або змінній можуть бути присутніми одночасно дві протилежних за знаком перевірюваних несправності. Дана формула більшою мірою орієнтована на пошук кратних дефектів в блоках цифрових систем, не пов'язаних з однією лінією. Кратність несправностей у цифровій системі теоретично створює умови для логічного додавання інших комірок стовпчика з метою формування результату у вигляді множини дефектів, які формують заданий вектор експериментальної перевірки, з яких необхідно відняти перевірювані на тесті несправності, які не впливають на формування некоректних реакцій за виходами.

Інтерес становить пошук кратних дефектів на основі мультипроцесора Хасе [60, 63], який орієнтований на вирішення задачі покриття шляхом повного перебору подій, що забезпечують точне покриття вектора експериментальної перевірки стовпцями таблиці несправностей:

$$F(T) = (\underset{i}{\vee} F_i) \oplus A = 0.$$

Тут рішенням є таке поєднання стовпців, що беруть участь у векторній операції логічного додавання, яке в сукупності дає результат, рівний вектору експериментальної перевірки. Оскільки операція часовитратна, то для неї слід використовувати мультипроцесор Хасе, орієнтований на взяття булеана в майже паралельному режимі.

Підводячи підсумок, слід представити модель процесу діагностування цифрових пристроїв, яка містить функціональні перетворювачі, пов'язані з виконанням наступних кроків (рис. 3.2):

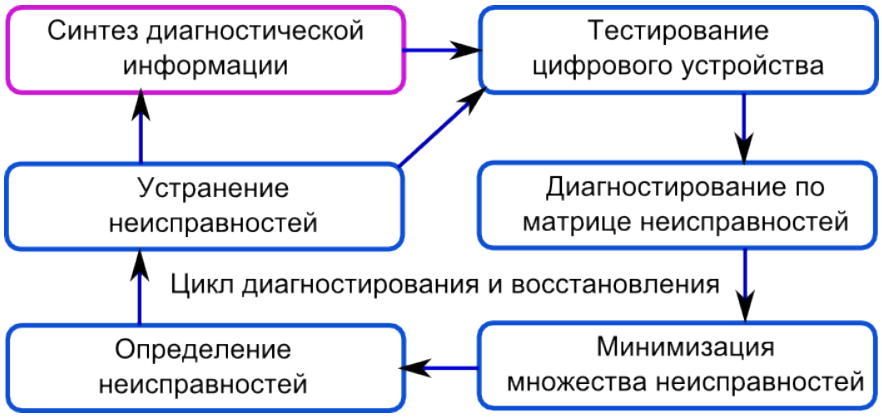


Рисунок 3.2 – Цикл діагностування та ремонту логічних блоків

1. Передпроцесування зводиться до генерування вихідної діагностичної інформації у вигляді тесту діагностування, таблиці несправностей і матриці досяжності цифрової системи.

2. Тестування реального пристрою на основі використання промислового симулятора з метою порівняння фактичних реакцій пристрою з еталонними значеннями за спостережуваними лініями-асерціями, що дає можливість сформувати матрицю вихідних реакцій або вектор експериментальної перевірки у двійковому алфавіті.

3. Обчислення матриці активності графової структури на кожному вхідному тестовому наборі, рівної за розмірністю таблиці несправностей, за допомогою матриці експериментальної перевірки і матриці досяжностей, що дає можливість істотно скоротити область підозрюваних дефектів.

4. Модифікація вмісту таблиці несправностей шляхом її маскування матрицею активності графової структури, з метою визначення тільки тих несправностей, які дійсно формують матрицю експериментальної перевірки в процесі діагностування.

5. Виконання процедури логічного множення над рядками таблиці несправностей для отримання вектора підозрюваних дефектів.

6. Відновлення працездатності цифрового пристрою шляхом переадресації несправних логічних компонентів на їх аналоги з ремонтного запасу і повторення процесу тестового діагностування.

Таким чином, *новизна запропонованого методу діагностування дефектів* полягає у використанні для отримання діагнозу єдиною паралельної операції логічного множення, що в поєднанні зі структурним маскуванням несправностей дає переваги перед аналогами в частині збільшення швидкодії і підвищення глибини діагностування.

3.3 Кубітне моделювання цифрових систем

Розглядаються структури даних, ефективні з точки зору програмної або апаратної реалізації справного інтерпретативного моделювання дискретних систем, описаних у формі кубітних векторів станів виходів примітивів. Для опису цифрової схеми, представленої на рис. 3.3, традиційно використовується структура взаємопов'язаних елементів і кубічні покриття (таблиці істинності) логічного елементів.

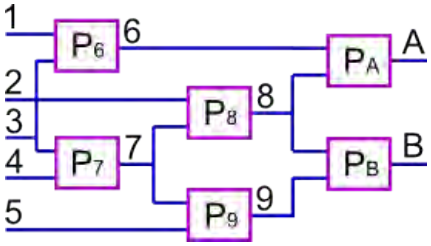


Рисунок 3.3 – Фрагмент цифрової схеми

Мета пропонованого методу кубітного моделювання – замінити таблиці істинності компонентів цифрового пристрою векторами станів виходів. Нехай функціональний примітив з номером має наступну таблицю істинності:

$P_6 =$

X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0

Дане покриття логічного елемента можна трансформувати шляхом унітарного кодування вхідних векторів на основі використання двотактного алфавіту [60, 62, 63, 67]. Символи та їх коди, призначені для опису автоматних змінних, являють собою булеан на універсумі з чотирьох примітивів, що відповідає формату вектора, який містить два кубіта:

B*(Y)={Q=(1000), E=(0100), H=(0010), J=(0001), O={Q,H}=(1010), I={E,J}=(0101), A={Q,E}=(1100), B={H,J}=(0011), S={Q,J}=(1001, P={E,H}=(0110), C={E,H,J}=(1110), F={Q,H,J}=(1011), L={Q,E,J}=(1101), V={Q,E,H}=(1110), Y={Q,E,H,J} =(1111), U=(0000)}.

За допомогою двотактного алфавіту будь-яке покриття функціонального примітиву шляхом кодування вхідних наборів і подальшого об'єднання символів завжди можна представити двома кубами або навіть одним, враховуючи, що куби взаємно інверсних:

$$P_6 = \begin{array}{|c|c|}\hline 00 & 1 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline Q & 1 \\ E & 1 \\ H & 1 \\ J & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline V & 1 \\ J & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1110 & 1 \\ 0001 & 0 \\ \hline \end{array} \rightarrow \boxed{1\ 1\ 1\ 0}$$

Два куба показують не тільки всі рішення, але й інверсію сигналів на виході, що цікаво з позиції активізації всіх логічних шляхів в схемною структурі при синтезі тестів. Наприклад, для зміни стану виходу необхідно створити на входах пару наступних один за одним умов, де в першому такті повинні бути перші три вектори (адреси), а в другому – четвертий вектор, що формується двома вхідними змінними.

Для моделювання справної поведінки досить мати тільки один куб (нульовий або одиничний), оскільки другий завжди є доповненням до першого. Отже, орієнтуючись, наприклад, на одиничний куб, який формує на виході 1, можна прибрати біт стану виходу примітиву, що зменшить розмірність куба або моделі примітиву до кількості адресованих станів елемента, де адреса є вектор, складений з двійкових значень вхідних змінних, за яким визначається стан виходу примітиву.

Кубітне покриття або Q-покриття є векторна інтерпретативна форма завдання функціональності, де значення координати визначає стан виходу функції, відповідне двоичному вхідному слову, формує адресу комірки. Q-покриття одновиходового примітиву завжди представлено двома взаємно інверсними кубами (векторами), розмірність яких дорівнює ступеню двійки від числа вхідних змінних, де одиничне значення координати визначає участь адреси розглянутого біта у формуванні відповідного (0,1) стану виходу примітиву. Кубітні моделі примітивів вимагають створення нової теорії моделювання, прямої і зворотної імплікації, синтезу тестів, моделювання несправностей, пошуку дефектів. Тут і далі представлені основні процедури справного моделювання на основі маніпулювання адресами, неявно представленими в координатах кубів Q-покриття.

Модель для аналізу цифрової системи на основі використання кубітних структур даних може бути описана чотирма компонентами:

$$F =< L, M, X, Q >,$$
$$L = (L_1, L_2, ..., L_j, ..., L_n);$$
$$M = (M_1, M_2, ..., M_j, ..., M_n);$$
$$X = (X_{n_x+1}, X_{n_x+2}, ..., X_{n_x+i}, ..., X_n);$$
$$Q = (Q_{n_x+1}, Q_{n_x+2}, ..., Q_{n_x+i}, ..., Q_n).$$

Тут представлені відповідно: L – вектор ідентифікаторів еквіпотенціальних ліній схеми цифрової системи, який через свою тривіальність може бути виключений з моделі, але при цьому необхідно мати число вхідних змінних пристрою і загальну кількість ліній; M – вектор моделювання станів всіх ліній схеми; X – упорядкована сукупність векторів вхідних змінних кожного примітиву схеми, прив'язаних до номерів виходів, Q – сукупність Q-покриттів примітивів, суворо прив'язаних до номерів виходів і вхідним змінним примітивів; n – число ліній у схемі, n_x – кількість вхідних змінних.

Як приклад кубітного завдання моделі цифрового пристрою $F =< L, M, X, Q >$, поданого на рис. 3.3, нижче приведено варіант структурної таблиці опису схеми для аналізу справної поведінки:

L	1	2	3	4	5	6	7	8	9	A	B
M	1	1	1	1	1	0	1	0	1	1	0
X	.	.	.	.	.	13	34	27	75	68	89
Q	.	.	.	.	.	1	0	1	1	1	1
	.	.	.	.	.	1	1	0	0	0	0
	.	.	.	.	.	1	1	0	0	1	1
	.	.	.	.	.	0	1	0	1	0	1

Метод кубітного моделювання справної поведінки зводиться до визначення значення виходу елемента за адресою, що формується конкатенацією двійкових станів вхідних змінних кожного примітиву цифрової системи $M(Y_i) = Q_i[M(X_{i1} * X_{i2} ... * X_{ij} ... * X_{ik_i})]$. Тут k_i – число вхідних ліній в примітиві з номером i. Оскільки номера невхідних ліній вектора L

однозначно ідентифікують за виходами оброблювані примітиви, то формула моделювання може бути приведена до циклу визначення станів всіх невхідних змінних:

$$M_i = Q_i[M(X_{i1} * X_{i2} ... * X_{ij} ... * X_{ik_i})] = Q_i[M(A_i)],\ i = \overline{n_x + 1, n}.$$

Тут процес моделювання пов'язаний з конкатенованим формуванням адреси біта в кубіті функціональності, який визначає стан примітиву чи невхідної лінії цифрової структури, починаючи з номера $i = n_x + 1$. Якщо змінні створюють не двійкову адресу, то в даному випадку існує можливість формування стану виходу логічного елемента в троїчному алфавіті символом X. Стани виходів формуються ідеально примітивною процедурою обробки кубіта примітиву

$$M_i = Q_i[M(X_i)]$$

на основі простих ітерацій або ітерацій Зейделя [58, 62]. У другому випадку необхідна передпроцесорна процедура ранжування ліній і примітивів схеми, яка дозволяє суттєво зменшити кількість проходів за елементами схеми для досягнення збіжності, коли фіксується рівність станів всіх ліній в двох сусідніх ітераціях. Крім того, ранжування примітивів за рівнями формування виходів дає можливість істотно підвищити швидкодію моделювання за рахунок паралельної обробки функціональних елементів одного рівня. Наприклад, для схеми, зображеної на рис. 3.3, одночасно можна обробляти елементи з номерами 6,7, потім – 8,9 і далі – А, В. У першому випадку, коли використовуються прості ітерації, ранжування не потрібно, але платою за простоту алгоритму моделювання є істотно більша кількість ітеративних проходів за примітивами схеми для досягнення згаданого критерію збіжності. Обчислювальна складність запропонованого Q-методу моделювання на основі кубітних функціональностей визначається процедурами формування

адреси – вхідного вектора, що містить $\mathbf{k_i}$ змінних, для кожного i-го примітиву $[(r+w)\times k_i]$, зчитуванням біта з кубіт-вектора по конкатенованій адресі та записом $(r+w)$ даного біта у вектор моделювання:

$$\eta = \sum_{i=n_x+1}^{n}\{[(r+w)\times k_i]+(r+w)\} = \sum_{i=n_x+1}^{n}[(r+w)\times(k_i+1)] = (r+w)\times\sum_{i=n_x+1}^{n}(k_i+1).$$

Час моделювання одного тест-вектора Q-методом за умови, що цифрова схема, яка складена з 900 чотревходових примітивів, має параметри: r = w = 5ns, $\mathbf{k_i}$= 4, $\mathbf{n_x}$= 100, n=1000, дорівнює 45 мікросекундам:

$$\eta = (r+w)\times\sum_{i=n_x+1}^{n}(k_i+1) = (5+5)\times 900\times(4+1) = 10\times 900\times 5 = 45000\text{ns} = 45\mu\text{s}.$$

Це означає, що швидкодія інтерпретативного Q-методу моделювання дає можливість для даної схеми обробити за одну секунду 22222 вхідних наборів. При цьому цифровий пристрій має істотну перевагу – сервісну функцію online відновлення працездатності в разі відмови примітиву шляхом його переадресації на запасний елемент.

Для синтезу квазіоптимальних структур даних комбінаційного пристрою необхідно використати наступні правила:

1) Ранжована схема цифрового пристрою за структурною глибиною для моделювання за способом Зейделя повинна мати по можливості однотипні примітиви в кожному рівні (шарі) спрацьовування.

2) У кожному рівні бажано мати однакове число примітивів. Тому синтез цифрового пристрою слід орієнтувати на створення прямокутної (матричної) структури однотипних логічних елементів.

3) Реалізація комбінаційних примітивів припускає використання адресованих елементів пам'яті, що мають місце бути в програмованих

логічних пристроях (FPGA, CPLD), широко використовуваних для створення прототипів.

4) Формування для кожного рівня комбінаційного пристрою ремонтних примітивів для відновлення працездатності в режимі online з розрахунку – один запасний елемент на кожен тип компонента, який використовується у рівні.

5) Вартість витрат апаратури для реалізації комбінаційного пристрою, орієнтованого на високу швидкодію, повинна визначатися сумою всіх примітивів, прив'язаних до рівнів комбінаційного пристрою, доповненої лінійкою запасних елементів по одному для кожного шару (за умови існування в кожному шарі однакових примітивів):

$$Q = \sum_{i=\overline{1,n}}^{j=\overline{1,m}} P_{ij} + n.$$

6) Реализація комбіинаційного пристрою, орієнтованого на мінімізацію апаратних витрат, визначається сумою всіх типів примітивів, інваріантних до рівнів комбінаційного пристрою, доповненою лінійкою запасних елементів по одному для кожного типу:

$$Q = \sum_{i=1}^{m} P_i + m.$$

7) Обробка матриці комбінаційних елементів за допомогою процесорної лінійки примітивів, число яких дорівнює потужності максимального рівня або шару в прямокутній структурі, що забезпечує умови для паралельної обробки всіх примітивів в кожному рівні елементів з метою підвищення швидкодії комбінаційного прототипу, реалізованого в PLD.

Таким чином, новизна запропонованого Q-методу інтерпретативного справного моделювання цифрових схем полягає в істотному підвищенні

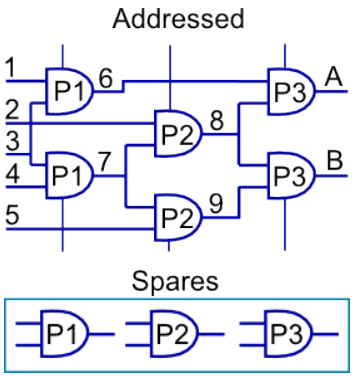
швидкодії і зменшенні обсягів структур даних за рахунок заміни таблиць істинності на Q-покриття, що практично робить розробку конкурентоспроможною з технологіями компілятивного моделювання.

3.4 Відновлення працездатності комбінаційних пристроїв

Нечисленні роботи, присвячені відновленню працездатності логічних схем [68-70], описують дві ідеї. Перша полягає в реконфігурації структури логічних елементів в режимі offline, яка забезпечує можливість заміни кожного з несправних примітивів. Друга створює умови для заміни несправних елементів шляхом використання запасних логічних компонентів і мультиплексорів для переадресації примітивів, що відмовили.

Структури кубітних даних модифікуються убік доповнення рядком типів примітивів $F =< L, M, X, P, Q >, P = (P_1, P_2, ..., P_i, ..., P_m)$, задіяних при синтезі цифрової системи, якщо необхідно в процесі функціонування виконувати ремонт або відновлення працездатності за рахунок введення запасних примітивів, які, так само як і основні, реалізуються на основі елементів пам'яті.

На рис. 3.4 зображено приклад схемної структури з адресованих і трьох запасних елементів. Структури даних, відповідні даній схемі з трьома додатковими елементами, подані тут же:



L	1	2	3	4	5	6	7	8	9	A	B
M	1	1	1	1	1	0	0	1	1	1	0
X	.	.	.	.	.	13	34	27	75	68	89
P	.	.	.	.	.	1	2	3	4	5	6
Q	1	1	0	0	1	1	1	0	1	.	.
	1	1	1	1	0	0	1	1	0	.	.
	1	1	1	1	1	1	1	1	1	.	.
	0	0	0	0	0	0	0	0	0	.	.

Рисунок 3.4 – Приклад схемної структури з адресованих і запасних елементів

Таблиця (див. рис. 3.4) оперує номерами структурних примітивів, що дає можливість замінити будь-який елемент, що відмовив, справним з ремонтного запасу шляхом зміни адресного номера в рядку примітивів P. Ремонтні елементи в цій таблиці починаються зі стовпчика номер 7. У наступній таблиці представлені рядок типів логічних елементів, а також адреси типів цих примітивів, відмічені номерами:

L	1	2	3	4	5	6	7	8	9	A	B
M	1	1	1	1	1	0	0	1	1	1	0
X	.	.	.	.	.	13	34	27	75	68	89
P	.	.	.	.	.	1	1	2	2	3	3
Q	1	0	1	1	0	1	.	.	.	.	.
	1	1	0	1	1	0	.	.	.	.	.
	1	1	1	1	1	1	.	.	.	.	.
	0	0	0	0	0	0	.	.	.	.	.

Така структура даних орієнтована на програмну реалізацію моделювання, а ремонтні примітиви починаються з номера 4. Якщо існує можливість перепрограмування логіки в елементи пам'яті з однаковим числом вхідних змінних, то дану процедуру слід виконувати після фіксації несправного елемента, коли стає відомо – який елемент у структурі та який тип примітиву відмовив. Процедура відновлення працездатності орієнтована на PLD-реалізацію цифрових систем. Якщо кубітні моделі схем не мають запасних примітивів, то відповідний формат таблиць буде мати наступний вигляд:

L	1	2	3	4	5	6	7	8	9	A	B
M	1	1	1	1	1	0	0	1	1	1	0
X	.	.	.	.	.	13	34	27	75	68	89
P	.	.	.	.	.	1	1	2	2	3	3
Q	1	0	1	.	.	.	.	.	.	.	.
	1	1	0	.	.	.	.	.	.	.	.
	1	1	1	.	.	.	.	.	.	.	.
	0	0	0	.	.	.	.	.	.	.	.

Таким чином, кубітні структури даних орієнтовані на компактність опису функціоналів цифрового вироби векторами станів виходів, підвищення швидкодії процедур моделювання за рахунок адресації станів виходів примітивів, а також на відновлення працездатності окремих логічних елементів, завдяки їх реалізації в елементах пам'яті PLD або у формі програмних модулів. Дуже важливо, що в останньому випадку не потрібно зберігати ремонтні примітиви, оскільки запропоновані тут інтерпретативні структури табличних даних спочатку орієнтовані на технологічні зручності усунення дефектів у процесі функціонування прототипу цифрового вироби.

Обробка схеми в кристалі зводиться до визначення адреси, складеного двійковими бітами вектора моделювання, за яким знаходиться значення логічної функції. Кожен примітив має цикл обробки, що містить три процедури:

1) Адресне зчитування номерів вхідних змінних із відповідного стовпця матриці X для формування адреси стану вхідної змінної вектора моделювання: $A = X_{ij},\ i = \overline{1,n};\ j = \overline{1,s_p - 1}$;

2) Формування адреси (двійкового коду) для обчислення логічної функції шляхом конкатенації відповідних станів вхідних змінних у векторі моделювання $A = M(X_{ij}) * M(X_{ir})$;

3) Запис результату виконання логічної функції як стану виходу у відповідний розряд вектору моделювання $M(X_{is_p}) = P[M(X_{ij}) * M(X_{ir})]$.

Процес обробки всіх примітивів схеми в даному випадку є строго послідовним, що являє собою істотне уповільнення процедури формування станів вихідних змінних. Проте зменшення швидкодії можна вважати платою за сервіс вбудованого і автономного відновлення працездатності цифрової структури, який є одним з етапів функціонування інфраструктури обслуговування SoC, представленої на рис. 3.5.

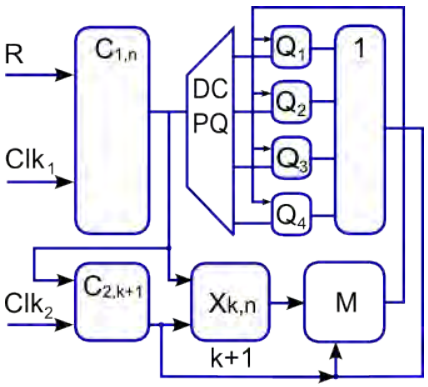


Рисунок 3.5 – Операційна структура комбінаційної схеми

Комбінаційна схема стає операційним пристроєм, де присутні операційний і керуючий автомати. Замінними компонентами в операційному автоматі є типи примітивів – функціональні елементи або структурні примітиви.

Операційний пристрій для реалізації елементно-адресованих комбінаційних схем містить: лічильник обробки поточного примітиву; пам'ять для зберігання типів примітивів, відповідних структурних елементів P; лічильник зчитування номерів вхідних і вихідної змінних поточного примітиву; дешифратор типів примітивів DC; пам'ять для зберігання вектора моделювання M; матрична пам'ять для зберігання номерів входів-виходів структурних примітивів X; лінійка пам'ятей, що реалізують функціональні примітиви P(Q); регістр формування вхідного адресного слова для оброблюваного примітиву RG; логічний елемент Or для комутації результатів обробки функціональних примітивів.

Граф-схема алгоритму управління процесом моделювання структури комбінаційної схеми представлена на рис. 3.6 і містить наступні кроки:

1. Ініціалізація (формування) усіх компонентів (номери та типи елементів, лінії зв'язків для входів і виходів логічних елементів) схемної структури:

$$P = (P_1, P_2, ..., P_i, ..., P_n);\ Q = (Q_1, Q_2, ..., Q_j, ..., Q_m);$$
$$X = [X_{pq}]; p = \overline{1,n};\ q = \overline{1,s_p}.$$

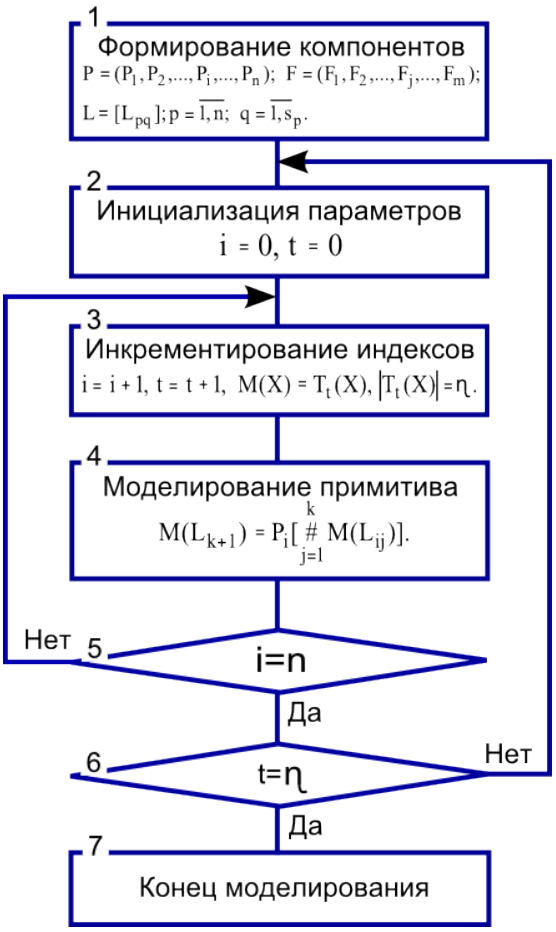


Рисунок 3.6 – Граф-схема алгоритму управління процесом моделювання

2. Ініціалізація параметру оброблюваного примітиву і номеру вхідного набору $i = 0, t = 0$ для його моделювання у двійковому алфавіті $M_r = \{0,1\}$.

3. Інкрементування индексу примітиву, номеру тесту та ініціалізація вхідного тестового (робочого) набору: $i = i + 1,\ t = t + 1,\ M(X) = T_t(X), \left|T_t(X)\right| = \eta.$

4. Конкатенація (#) розрядів слова для формування вхідного впливу $\#_{j=1}^{k} M(X_{ij})$ логічного елементу P_i (типу Q_i) і виконання процедури визначення стану його виходу з наступним записом у відповідну координату вектору моделювання $M(X_{k+1})$:

$$M(X_{k+1}) = \{P_i, Q_i\}[\#_{j=1}^{k} M(X_{ij})].$$

5. Повторення пунктів 3 і 4 з метою отримання станів виходів всіх логічних елементів до виконання умови: i = n.

6. Повторення пунктів 2–4 з метою моделювання всіх вхідних тестових (робочих) наборів, до виконання рівності: $t = \eta$, где η – довжина тесту.

7. Закінчення процесу моделювання цифрового пристрою.

Таким чином, *наукова новизна запропонованої моделі цифрової системи* полягає у введенні в структуру пристрою надлишкових ремонтних компонентів і керуючого автомата, орієнтованого на послідовну обробку комбінаційних примітивів, що дає можливість здійснювати процедуру переадресації примітивів у разі відмови одного з них. Неважко створити аналогічні автомати для паралельної обробки шарів з примітивів ранжованої схеми, що максимально наблизить швидкодію пристрою до його реалізації в кристалах PLD.

3.5 Висновки до розділу 3

Наукова новизна і практична значущість дослідження, заснована на застосуванні кубітних структур, орієнтованих на паралельне обчислення теоретико-множинних по суті даних, формується в наступні кілька пунктів:

1. Удосконалено метод діагностування дефектів цифрових систем за рахунок використання єдиної паралельної операції логічного множення, що в поєднанні зі структурним маскованням несправностей дає переваги перед аналогами в частині компактності представлення даних, збільшення швидкодії і підвищення глибини діагностування.

2. Запропоновано новий Q-метод інтерпретативного справного моделювання цифрових схем, який характеризується використанням компактних Q-покриттів замість таблиць істинності, що дає можливість істотно підвищити швидкодію аналізу за рахунок адресного формування виходів функціональних примітивів і зменшити обсяги структур даних, що практично робить метод конкурентоспроможним з технологіями компілятивного моделювання.

3. Удосконалено модель цифрової системи шляхом доповнення в структуру пристрою надлишкових ремонтних компонентів і керуючого автомата, орієнтованого на послідовну обробку комбінаційних примітивів, що дає можливість здійснювати процедуру переадресації примітивів, що відмовили, в режимі штатного функціонування.

4. Показані приклади використання кубітних структур даних і процедур формування адрес для моделювання цифрових схем та вирішення завдань діагностування шляхом використання векторних паралельних логічних операцій і ремонту несправних модулів на основі адресованих логічних примітивів.

5. Основна інноваційна ідея квантових або кубітних обчислень полягає в переході від обчислювальних процедур над байт-операндом, що визначає у дискретному просторі одне рішення (точку), до квантових паралельних процесів над кубіт-операндом, який одночасно формує булеан рішень.

4 ДІАГНОСТУВАННЯ ТА КОРЕКЦІЯ SOC HDL КОДУ

Пропонується технологія діагностування HDL-моделей систем на кристалах, яка базується на використанні транзакційного графа. Описується метод діагностування, спрямований на зменшення часу виявлення несправностей і пам'яті для зберігання діагностичної матриці за рахунок формування тернарних відносин між тестом, монітором і функціональним компонентом. Вирішуються наступні завдання: розробляється модель цифрової системи у вигляді транзакційного графа і мультідерева таблиць несправностей, а також тернарні матриці активації функціональних компонентів обраного набору моніторів за допомогою тестових послідовностей; розробляється метод аналізу матриці активації з метою виявлення несправних блоків із заданою глибиною і синтезу логічних функцій для подальшого вбудованого апаратного діагностування несправностей.

4.1 TAB модель діагностування несправных компонентів SoC

Мета дослідження – розробка матричної моделі TAB (Tests – Assertions – Blocks) і методу діагностування, що дозволяють зменшити час тестування і обсяг пам'яті для зберігання діагностичної інформації за рахунок формування тернарних відносин (тест – монітор – функціональний компонент) в одній таблиці.

Завдання дослідження: 1) розробка HDL-моделі цифрової системи у формі транзакційного графа для діагностування функціональних блоків з використанням набору асерцій [28, 29, 35, 52, 53, 71, 75-78]; 2) розробка методу аналізу TAB-матриці з метою виявлення мінімального набору несправних блоків [72-76]; 3) Синтез логічних функцій для вбудованої процедури діагностування несправностей [77, 78].

Модель тестування HDL-коду цифрової системи подана наступними xor-відношеннями параметрів <тест – функціональність – несправні блоки B *>:

$$T \oplus B \oplus B^* = 0;$$
$$B^* = T \oplus B = \{T \times A\} \oplus B,$$

які перетворюються у відношення компонентів ТАВ-матриці:

$$M = \{\{T \times A\} \times \{B\}\}, M_{ij} = (T \times A)_i \oplus B_j.$$

Тут координати матриці дорівнюють 1, якщо пара тест-монітор $(T \times A)_i$ перевіряє або активує несправності функционального блоку $B_j \in B$.

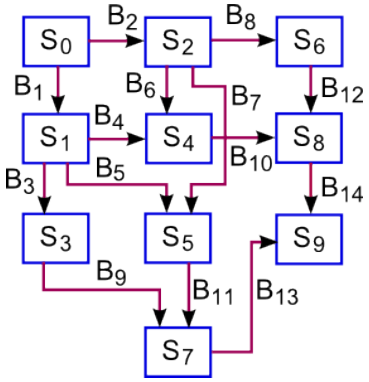
Аналітична модель верифікації з використанням темпоральних асерцій (додаткова спостережуваність операторів або ліній), орієнтована на досягнення заданої глибини діагностування:

$$\Omega = f(G, A, B, S, T),$$
$$G = (A * B) \times S; S = f(T, B);$$
$$A = \{A_1, A_2, ..., A_i, ..., A_h\};$$
$$B = \{B_1, B_2, ..., B_i, ..., B_n\};$$
$$S = \{S_1, S_2, ..., S_i, ..., S_m\};$$
$$T = \{T_1, T_2, ..., T_i, ..., T_k\}.$$

Тут $G = (A * B) \times S$ – функціональність, що зображена CFT графом (Code-Flow Transaction), рис. 4.1; $S = \{S_1, S_2, ..., S_i, ..., S_m\}$ – вершини або оператори коду програми при симуляції тестових сегментів (наборів). Іншими словами, граф може розглядатися як АВС-граф (Assertion Based Coverage Graph). Кожний стан $S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\}$ визначається значеннями суттєвих змінних проекту (булеві, регістрові змінні, пам’ять). Дуги орієнтованого графа подані множиною програмних блоків:

$$B = \{B_1, B_2, ..., B_i, ..., B_n\}; \bigcup_{i=1}^{n} B_i = B; B_i \bigcap_{i \neq j} B_j = \varnothing,$$

де асерції $A_i \in A = \{A_1, A_2, ..., A_i, ..., A_n\}$ можуть бути додані до кожного блоку B_i – послідовність операторів коду, які визначають стан вершини графу $S_i = f(T, B_i)$ у залежності від тестового набору $T = \{T_1, T_2, ..., T_i, ..., T_k\}$. Асерційний монітор, що об'єднує асерції вхідних дуг $A(S_i) = A_{i1} \vee A_{i2} \vee ... \vee A_{ij} \vee ... \vee A_{iq}$ може бути встановлений у кожній вершині.



$$B = (B_1B_3B_9 \vee (B_2B_7 \vee B_1B_5)B_{11})B_{13} \vee$$
$$\vee ((B_1B_4 \vee B_2B_6)B_{10} \vee B_2B_8B_{12})B_{14} =$$
$$= B_1B_3B_9B_{13} \vee B_2B_7B_{11}B_{13} \vee B_1B_5B_{11}B_{13} \vee$$
$$\vee B_1B_4B_{10}B_{14} \vee B_2B_6B_{10}B_{14} \vee B_2B_8B_{12}B_{14}.$$

Рисунок 4.1 – Приклад ABC-графа HDL-кода

Модель HDL-коду, яка подана у вигляді ABC-графа, описує не тільки структуру програмного коду, але також тестові сегменти функціонального покриття, що формуються з використанням програмних блоків, які входять до розглянутої вершини. Остання визначає простір станів змінних, що досягнуте на тесті, і формує функціональне покриття станів змінних, відповідних розглянутій вершині графа $Q = \mathrm{card}S_i^r / \mathrm{card}S_i^p$. У сукупності всі вершини графа мають представляти собою повний простір покриття станів змінних програмного коду, яке визначає якість тесту, дорівняний 100%: $Q = \mathrm{card}\bigcup_{i=1}^{m} S_i^r / \mathrm{card}\bigcup_{i=1}^{m} S_i^p = 1$. Більш того, множина асерцій $<A, S>$, яке існує в графі, дозволяє виконати моніторинг дуг (покриття коду) $B = (B_1, B_2, ..., B_i, ..., B_n)$ та вершин

(функціональне покриття) $S=\{S_1,S_2,...,S_i,...,S_m\}$. Асерції на дугах $B_i \in B$ графа призначені для діагностування функціональних несправностей в програмних блоках. Асерція у вершинах графа $S_i \in S$ несе інформацію про якість тесту та множини асерцій з метою їх покращення або доповнення. АВС-граф дозволяє: 1) оцінювати якість програмного коду шляхом визначення діагнозопрідатності проекту (diagnosability); 2) мінімізувати вартість генерації тестів, діагностування та виправлення функціональних порушень за рахунок використання ассерцій; 3) оптимізувати синтез тестів шляхом покриття всіх дуг і вершин графа мінімальним набором активізованих тестових шляхів. Наприклад, мінімальний тест для наведеного вище АВС-графа містить шість сегментів, які активізують всі наявні дуги і вершини:

$$T = S_0S_1S_3S_7S_9 \vee S_0S_1S_4S_8S_9 \vee S_0S_1S_5S_7S_9 \vee \vee S_0S_2S_4S_8S_9 \vee S_0S_2S_5S_7S_9 \vee S_0S_2S_6S_8S_9.$$

В процесі діагностування тестові сегменти $T=\{T_1, T_2,..., T_r,..., T_k\}$ активізують транзакційний шлях графової моделі, який покриває всі вершини та дуги. Як правило, тестова модель подана декартовим добутком $M = T \times A \times B$, який відповідно має розмірність $Q = k \times h \times n.$ З метою зменшення кількості діагностичних даних окремий монітор або асерційна точка для візуалізації функціональної активізації блоків призначаються кожному тестовому сегменту. Це дозволяє зменшити розмірність матриці до $Q = n \times k$ та зберегти всі особливості тріади відношень $M =< T \times A \times B >$. Пара «тест – монітор» подається трьома можливими формами:

$$< T_i \to A_j >,< \{T_i, T_r\} \to A_j >,< \{T_i\} \to \{A_j, A_s\} > .$$

Метод діагностування функціональних порушень блоків передбачає використання попередньо побудованої ТАВ-матриці (таблиці) $M = [M_{ij}]$, де

рядок є відношення між тестовим сегментом та підмножиною блоків, що активізуються

$$T_i \to A_j \approx (M_{i1}, M_{i2},..., M_{ij},..., M_{in}), M_{ij} = \{0,1\},$$

які спостерігаються монітором A_j. Стовпець таблиці описує відношення між функціональними блоками і тестовими сегментами щодо монітора $M_j = B_j(T_j, A_j)$.

Для діагностування несправних блоків за допомогою процедури тестування визначається реальний вектор асерційної перевірки $A^* = (A_1^*, A_2^*,..., A_i^*,..., A_n^*)$ на тестовому наборі T шляхом формування $A_i^* = f(T_i, B_i)$. Виявлення несправних функціональних блоків ґрунтується на xor-операції між реальним вектором ассерційної перевірки і стовпцем TAB-матриці $A^* \oplus [M_1(B_1) \vee M_2(B_2) \vee ... \vee M_j(B_j) \vee ... \vee M_n(B_n)]$. Несправний блок визначається за допомогою вектора B_j за мінімальною кількістю одиничних координат:

$$B = \min_{j=1,n} [B_j = \sum_{i=1}^{h} (B_{ij} \oplus A_i^*)].$$

Як доповнення до моделі діагностування слід описати деякі важливі властивості TAB-матриці:

$$1)\ M_i = (T_i \times A_j);$$
$$2)\ \overset{m}{\underset{i=1}{\vee}} M_{ij} \rightarrow \overset{n}{\underset{j=1}{\forall}} M_j = 1;$$
$$3)\ M_{ij} \overset{n}{\underset{j=1}{\oplus}} M_{rj} \neq M_{ij};$$
$$4)\ M_{ij} \overset{k}{\underset{i=1}{\oplus}} M_{ir} \neq M_{ij};$$
$$5)\ \log_2 n \leq k \leftrightarrow \log_2 |B| \leq |T|$$
$$6)\ B_j = f(T, A) \rightarrow B \oplus T \oplus A = 0.$$

Властивості означають: 1) Кожен рядок матриці являє собою підмножину декартова добутку теста і монітора. 2) Диз'юнкція всіх рядків матриці дає результат у вигляді вектора, в якому всі координати дорівнюють 1. 3) Всі рядки матриці різні, що виключає надмірність тесту. 4) Всі стовпці матриці різні, що виключає існування еквівалентних несправних блоків. 5) Кількість рядків матриці повинна бути більше двійкового логарифма числа стовпців, що визначає потенційну діагнозопридатність (діагностованість) кожного блоку. 6) Функція діагнозу кожного блоку залежить від повного тесту і моніторів, які повинні бути мінімізовані без погіршення діагнозопридатності.

Нижче наведені 6 тестових сегментів, які активізують шляхи графа щодо асерційної точки S9

$$T = S_0S_1S_3S_7S_9 \vee S_0S_1S_4S_8S_9 \vee S_0S_1S_5S_7S_9 \vee$$
$$\vee S_0S_2S_4S_8S_9 \vee S_0S_2S_5S_7S_9 \vee S_0S_2S_6S_8S_9,$$

при цьому досить просто вивити всі функціональні блоки з можливими порушеннями:

$$B = B_1B_3B_9B_{13} \vee B_2B_7B_{11}B_{13} \vee B_1B_5B_{11}B_{13} \vee$$
$$\vee B_1B_4B_{10}B_{14} \vee B_2B_6B_{10}B_{14} \vee B_2B_8B_{12}B_{14}.$$

Механізм асерцій може бути поданий трьома групами компонентів, які формують логічні вирази для моніторингу програмних і апаратних блоків HDL коду функціональності, які засновані на візуальних точках $\{A_9 \subseteq S_9, A_3 \subseteq S_3, A_6 \subseteq S_6\}$:

$$A_9 = T_1(B_1B_3B_9B_{13}) \vee T_2(B_2B_7B_{11}B_{13}) \vee T_3(B_1B_5B_{11}B_{13}) \vee$$
$$\vee T_4(B_1B_4B_{10}B_{14}) \vee T_5(B_2B_6B_{10}B_{14}) \vee T_6(B_2B_8B_{12}B_{14});$$
$$A_3 = T_1(B_1B_3); A_6 = T_6(B_2B_8).$$

На наступному кроці формуються 6 рядків ТАВ-матриці $M_{ij}(G_1)$ у вигляді відношень між тестовими сегментами та блоками, активізуються:

$M_{ij}(G_1)$	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}
$T_1 \to S_9$	1	.	1	.	.	.	.	.	1	.	.	.	1	.
$T_2 \to S_9$	1	.	.	1	.	.	.	.	.	1	.	.	.	1
$T_3 \to S_9$	1	.	.	.	1	.	.	.	.	.	1	.	1	.
$T_4 \to S_9$	.	1	.	.	.	1	.	.	.	1	.	.	.	1
$T_5 \to S_9$	.	1	.	.	.	.	1	.	.	.	1	.	1	.
$T_6 \to S_9$	.	1	.	.	.	.	.	1	.	.	.	1	.	1
$T_1 \to S_3$	1	.	1	.	.	.	.	.	.	.	.	.	.	.
$T_6 \to S_6$	.	1	.	.	.	.	.	1	.	.	.	.	.	.

ТАВ-матриця шляхів активізації показує існування еквівалентних порушень блоків 3 і 9, 8 і 12 на 6 тестових сегментах з однією асерційною точкою у вершині графа 9. Стовпці 3 і 9, 8 і 12 еквівалентні. Для усунення нерозрізненості двох пар несправних блоків необхідно створити два додаткових монітора у вузлах S3 і S6 для тестових сегментів T1 і T6 відповідно. В результаті три додаткові асерції у вузлах $A = (S_9, S_3, S_6)$ дозволять розрізнити всі несправні блоки програмного HDL-коду. Таким чином, граф дозволяє не тільки синтезувати оптимальний тест, а й визначити мінімальну кількість асерційних моніторів у вузлах графа, необхідну для пошуку несправних блоків із заданою глибиною діагностування.

Процедура діагностування з використанням запропонованої матриці визначається наступним виразом на основі векторної xor-операції між реальними вісьмома асерційними значеннями і стовпцем B TAB-матриці:

$$\{[A_9(T_1,T_2,T_3,T_4,T_5,T_6),A_3(T_1),A_6(T_6)]\oplus B_j=0\}\rightarrow(B_j-\text{failed}).$$

4.2 Діагнозопридатне проектування

Діагнозопридатність визначається відношенням N_d/N кількості виявлених несправних блоків N_d, (якщо немає еквівалентних компонентів або глибина діагностування дорівнює 1), до загальної кількості N HDL-блоків.

Для оцінки витрат E на реалізацію TAB-матричної моделі виявлення функціональних порушень можна використовувати ефективність пари тест-асерція для заданої глибини діагностування. Критерій E функціонально залежить від ставлення між ідеальним $]\log_2 N[\times N$ і реальним $|T|\times|A|\times N$ обсягами необхідної пам'яті або ресурсів (де $|T|$ – довжина тесту, $|A|$ – кількість асерцій) для відповідної TAB-матриці та являє собою відносне значення в інтервалі від 0 до 1:

$$E=\frac{]\log_2 N[\times N}{|T|\times|A|\times N}=\frac{]\log_2 N[}{|T|\times|A|}.$$

Узагальнений критерій якості діагностування залежить від витрат E і діагнозопридатності D:

$$Q=E\times D=\frac{]\log_2 N[}{|T|\times|A|}\times\frac{N_d}{N}.$$

Наприклад, якість діагностування ТАВ-матриці $M_{ij}(G_1)$ до і після додавання двох рядків дорівнює:

$$Q_1[M(6\times 1\times 14)] = \frac{]\log_2 14[}{|6|\times|1|}\times\frac{10}{14} = 0{,}47;$$

$$Q_2[M(8\times 1\times 14)] = \frac{]\log_2 14[}{|8|\times|1|}\times\frac{14}{14} = 0{,}5.$$

Це означає, що розмір першої матриці трохи менше, ніж другий, але діагнозопридатність краще для другого варіанту матриці і в цілому вона є більш переважною. У порівнянні з відомим розв'язком [77], коли кожна комірка матриці містить всі існуючі асерції $|M_{ij}| = |A|$, другий варіант оцінюється наступним низьким значенням:

$$Q_2[M(6\times 3\times 14)] = \frac{]\log_2 14[}{|6|\times|3|}\times\frac{14}{14} = 0{,}2.$$

Таким чином, ТАВ-матриця, що сформована для вибраної пари тест-асерція, дозволяє отримати суттєву перевагу з точки зору скорочення обсягу пам'яті в $|A|-1$ раз при одинаковому значенні діагнозопридатності.

Діагностична якість ТАВ-матриці визначається відношенням кількості бітів, необхідних для ідентифікації (розпізнавання) всіх блоків $]\log_2 N[$, до реальної кількістї бітів коду, що поданий добутком довжини тесту на кількість асерцій $|T|\times|A|$. Якщо перша частина Е критерію якості Q дорівнює 1, що означає - кожен блок з функціональними порушеннями виявляється $N_d = N$, то тест і асерція є оптимальними і обумовлюють найкраще значення критерію якості моделі діагностування Q=1.

Мета аналізу АВС-графа - структурна оцінка розміщення асерційних моніторів, що дозволяє отримати максимальну глибину діагностування

несправних блоків. Діагнозопридатність АВС-графа є функцією, яка залежить від кількості N_n транзитних не кінцевих вершин, у яких існує тільки дві сусідніх дуги, одна з яких є вхідною, інша – вихідною. Такі дуги утворюють шляхи без збіжних і розбіжних розгалужень (N – загальна кількість дуг в графі):

$$D = \frac{N - N_n}{N}.$$

Оцінка визначається кількістю невиявлених або еквівалентних функціональних блоків. Потенційна установка додаткових моніторів для поліпшення діагностованих несправних блоків рекомендується для транзитних вузлів, що містять N_n. Критерій якості діагностування АВС-графа має вигляд:

$$Q = E \times D = \frac{]\log_2 N[}{|T| \times |A|} \times \frac{N - N_n}{N}.$$

Останній вираз визначає деякі практичні правила для синтезу діагнозопридатного HDL-коду: 1) Тест або testbench повинні створювати мінімальну кількість одновимірних шляхів активізації, що покривають всі вузли і дуги АВС-графа. 2) Базова кількість моніторів дорівнює числу кінцевих вершин графа без вихідних дуг. 3) Додатковий монітор може бути розміщений в кожній не кінцевій вершині, яка має одну вхідну і одну вихідну дугу. 4) Паралельні незалежні блоки коду повинні мати n моніторів і один паралельний тест, або один інтегрований монітор і n послідовних тестів. 5) Послідовно з'єднані блоки мають один тест активації для послідовного шляху і n-1 монітор або n тестів і n моніторів. 6) Вузли графа, які мають більше однієї вхідної і вихідної дуги, визначають сприятливі умови для діагностованості поточного фрагменту за допомогою тесту активізації одновимірного шляху

без установки додаткових моніторів. 7) Тестовий набір або testbench повинен забезпечити 100% функціональне покриття для вершин АВС-графа. 8) Критерій якості діагностування як функція, що залежить від структури графа, теста і асерційних моніторів, завжди може бути приведена до близького до 1 значення. Для цього є два альтернативні способи. Перший – збільшення кількості тестових сегментів шляхом активізації нових шляхів для розпізнавання еквівалентних несправних блоків без збільшення числа асерцій, якщо структура графа програми дозволяє потенційні зв'язки. Другий шлях – додавання асерційних моніторів у транзитних вершинах графа. Можливий також третій гібридний варіант, заснований на спільному застосуванні двох вищезгаданих способів.

4.3 Метод багаторівневого діагностування цифрових систем

На рис. 4.2 наведена багаторівнева модель мультидерева B, у якій кожній вершині відповідає компонент цифрової обчислювальної системи. Модель описується тривимірною таблицею (матрицею) активізації функціональних модулів. Вихідні дуги дозволяють перейти на більш низький рівень деталізації в процесі діагностування, коли заміна несправного блоку вимагає значних матеріальних витрат:

$$B = [B_{ij}^{rs}],\ \text{card}B = \sum_{r=1}^{n} \sum_{s=1}^{m_r} \sum_{i=1}^{p_{rs}} \sum_{j=1}^{k_{rs}} B_{ij}^{rs},$$

де n – кількість рівнів діагностування в мультидереві; m_r – кількість функціональних модулів або компонентів на рівні r; k_{rs} (p_{rs}) – кількість компонентів (довжина тесту) в таблиці B^{rs}; $B_{ij}^{rs} = \{0,1\}$ – компонент таблиці активізації, який визначається одиничним значенням, якщо несправна функціональність виявляється тестовим сегментом $T_{i\text{-}A_i}$ відносно

спостережуваного монітору A_i. Кожна вершина-таблиця має деяку кількість вихідних дуг, яка дорівнює числу функціональних компонентів, поданих матрицею активізації.

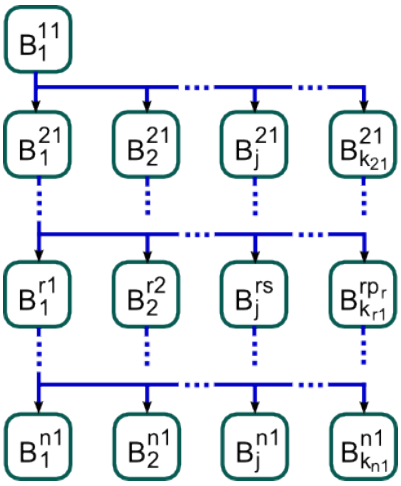


Рисунок 4.2 – Діагностична модель мультидерева

Метод діагностування несправних блоків апаратно-програмної (Hardware-Software) HS-системи, заснований на моделі мультидерева, дозволяє створити універсальний движок у вигляді алгоритму (рис. 4.3, блок 6) для обходу гілок дерева на апріорно заданій глибині:

$$B_j^{rs} \oplus A^{rs} = \begin{cases} 0 \to \{B_j^{r+1,s}, R\}; \\ 1 \to \{B_{j+1,}^{rs}, T\}. \end{cases}$$

Тут векторна xor-операція виконується між стовпцями матриці та вектором асерційної (експериментальної) перевірки A^{rs}, який визначається за допомогою xor-операції над реальним (m) і модельним (g) відгуками функціональності на тестові набори: $A_i^{rs} = m_i^{rs} \oplus g_i^{rs}, i = \overline{1, k_{rs}}$. Якщо всі координати векторної xor-суми $B_j^{rs} \oplus A^{rs} = 0$ дорівнюють нулю, то виконується одно з наступних дій: перехід до матриці активізації більш низького рівня $B_j^{r+1,s}$ або відновлення функціонального блоку $B = B_j^{rs}$.

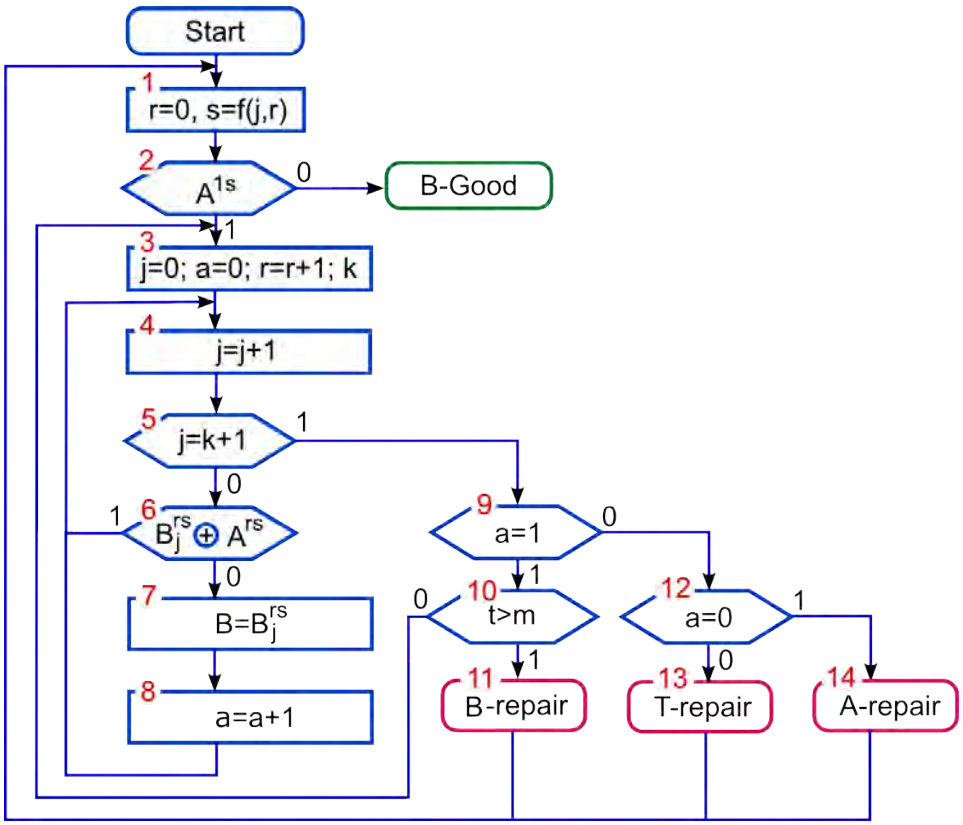


Рисунок 4.3 – Движок для обходу діагностичного мультидерева

Реалізується один із двох можливих шляхів аналізу у залежності від того, який параметр є найбільш важливим: 1) час (t> m, блок 10) – виконується відновлення несправного блоку; 2) матеріальні витрати (t <m) – здійснюється перехід на більш низький рівень для збільшення глибини діагностування несправностей, оскільки заміна більш дрібного блоку зменшує вартість відновлення. Якщо, принаймні, одна координата результуючого вектора xor-суми дорівнює одиниці $B_j^{rs} \oplus A^{rs} = 1$, здійснюється перехід до наступного стовпця матриці. Якщо всі координати асерційного вектора дрівнюють нулю $A^{1s} = 0$, це свідчить про те, що HS-система знаходиться у справному стані. Якщо всі векторні суми при обробці стовпця TAB-матриці не дорівнюють нулю $B_j^{rs} \oplus A^{rs} \neq 0$, то тест, згенерований для виявлення несправностей даного компонента функціональності, повинен бути скорегованим. Якщо більш, ніж одна векторна сума, отримана при обрабці стовпця TAB-матриці

дорівнює нулю $B_j^{rs} \oplus A^{rs} = 0$, це означає, що механізм асерцій, створений для діагностування даного компонента функціональності на представленому тесті, повинен бути доповнений ассерційними моніторами. Таким чином, ТАВ-движок має чотири кінцевих вершини, одна з яких B-good відповідає успішному завершенню тестування. Інші три вершини свідчать про отримання проміжного результату процесу тестування, який необхідно враховувати для збільшення якості тестування та глибини діагностування шляхом формування додаткових ассерцій та/або генерації додаткових тестових сегментів.

Таким чином, граф, наведений на рис. 4.2, дозволяє реалізувати ефективну інфраструктуру сервісного обслуговування для складних технічних систем. Перевагою ТАВ-движка, інваріантного до рівнів ієрархії, є простота підготовки та подання діагностичної інформації у вигляді мінімізованої таблиці активізації функціонального блоку на тестових сегментах.

Технологічна модель інфраструктури вбудованого тестування, діагностування та відновлення несправних блоків (рис. 4.4) включає три компоненти:

1. Тестування блоків (Unit Under Test – UUT) з використанням еталонної моделі (Model Under Test - MUT) для генерації вектора асерційної перевірки (assertion response vector), розмір якого відповідає кількості тестових наборів.

2. Пошук несправних блоків на основі аналізу ТАВ-матриці.

3. Відновлення несправних блоків шляхом заміни їх на справні компоненти з наявного резерву.

Процес-модель вбудованого сервісного обслуговування функціонує у реальному часі і дозволяє підтримувати працездатний стан HS-системи без безпосередньої участі людини. Запропонований алгоритм або ТАВ-движок для аналізу ТАВ-матриці, а також введений критерій якості діагностування дозволяють вирішувати завдання квазі-оптимального покриття програмних і апаратних блоків тестами і асерціями. Модель, що зображена на рис. 4.4, дозволяє забезпечити ефективне сервісне обслуговування складних HS-систем.

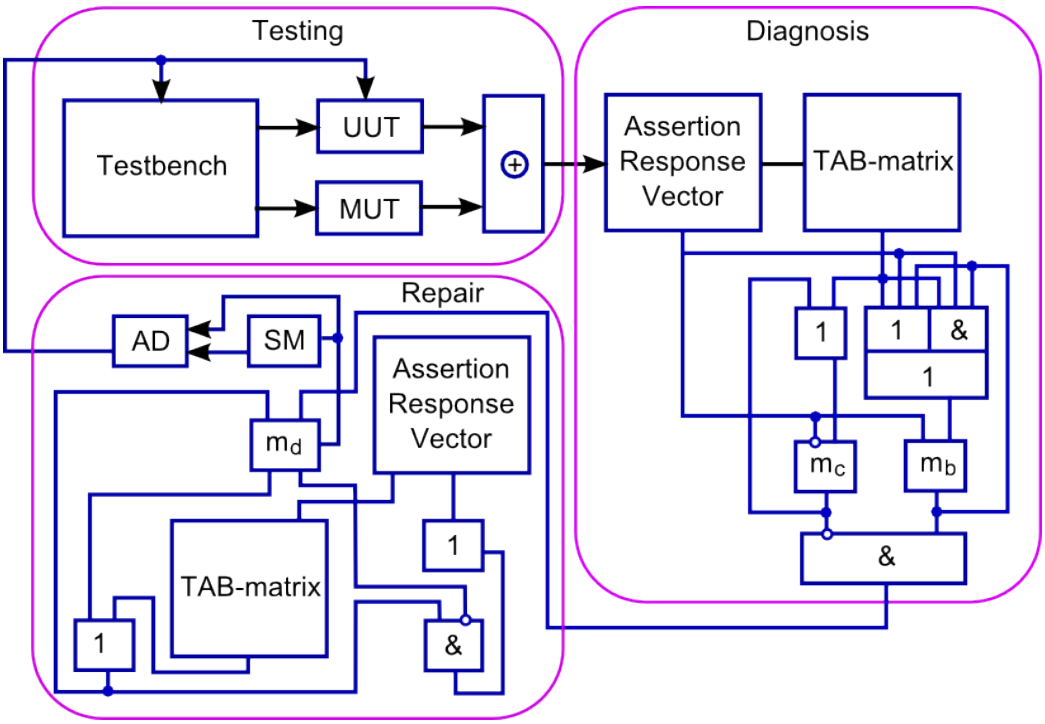


Рисунок 4.4 – Модель вбудованого тестування HS-компонентів

Виграш у часі виходить за рахунок введення в проект додаткової інфраструктури, рис. 4.5, яка дозволяє виконувати вибіркове тестування і діагностування, а також перепрограмування окремих модулів несправних блоків.

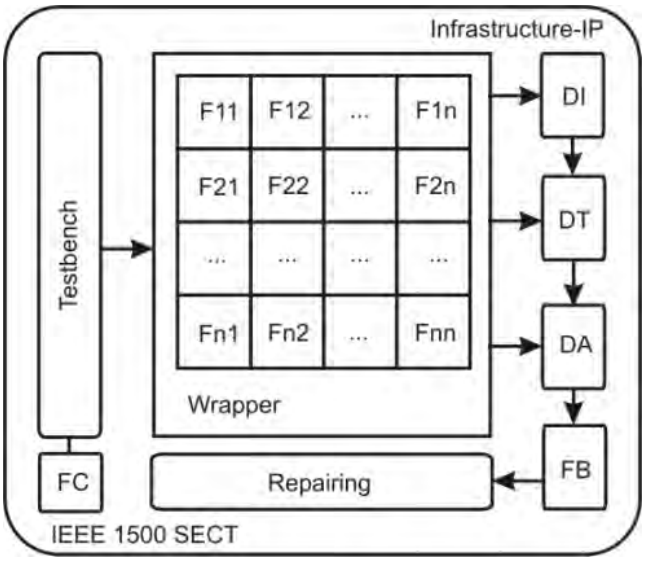


Рисунок 4.5 – Інфраструктура для тестування обчислювальних систем

На рис. 4.5 подані наступні блоки: Testbench – тести для функціональних блоків; FC – функціональне тестове покриття; F – функціональні блоки; DI – діагностична інформація у вигляді таблиць несправностей блоків; DT – методи та засоби діагностування; DA – результати аналізу діагностичної інформації; FB – несправні функціональні модулі; Repairing – відновлення функціональних модулів. Комірка граничного сканування, що представлена на рис. 4.6, забезбечує сервісне обслуговування одного функціонального модуля.

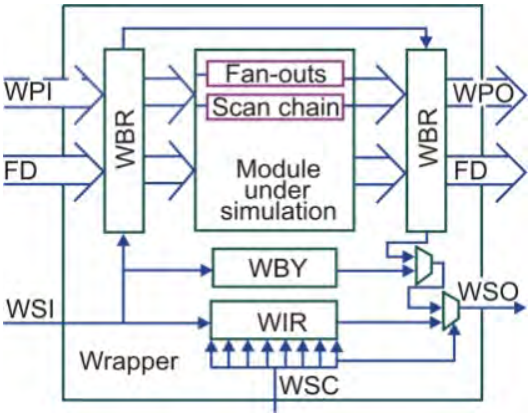


Рис. 4.6. Комірка граничного сканування

4.4 Приклад розв’язання задачі діагностування

Для ілюстрації ефективності запропонованої моделі та методу нижче використовується функціональність у вигляді трьох модулів цифрового фільтра Добеші [72].

Як другий контрольний приклад практичного використання запропонованої моделі активізації та xor-методу аналізу TAB-матриці для пошуку несправних блоків далі представлений синтез діагностичної матриці для графа основного фільтра, рис. 4.7.

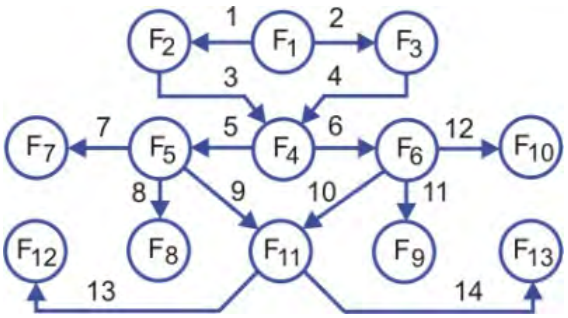


Рисунок 4.7 – Транзакційний граф main-TL

Граф пов’язаний з діагностичноюТАВ-матрицею, де мається 6 активізаційних тестових сегментів та 8 асерцій:

$M_{ij}(TL)$	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}
$T_1 \to F_7$	1	.	1	.	1	.	1	.	.	.	.	.	.	.
$T_2 \to F_8$	.	1	.	1	1	.	.	1	.	.	.	.	.	.
$T_3 \to F_9$	1	.	1	.	.	1	.	.	.	.	1	.	.	.
$T_4 \to F_{10}$	.	1	.	1	.	1	.	.	.	.	.	1	.	.
$T_5 \to F_{12}$	1	.	1	.	1	.	.	.	1	.	.	.	1	.
$T_6 \to F_{13}$	.	1	.	1	.	1	.	.	.	1	.	.	.	1
$T_1 \to F_2$	1	.	.	.	.	.	.	.	.	.	.	.	.	.
$T_2 \to F_3$	.	1	.	.	.	.	.	.	.	.	.	.	.	.

Система діагностичних функцій для апаратної імплементації частини інфраструктури сервісного обслуговування, що відповідна рядкам або моніторам:

$$F_7(T_1) = B_1^1 B_3^1 B_5^1 B_7^1;$$
$$F_8(T_2) = B_2^1 B_4^1 B_5^1 B_8^1;$$
$$F_9(T_3) = B_{11}^1 B_6^1 B_1^1 B_3^1;$$
$$F_{10}(T_4) = B_4^1 B_5^1 B_6^1 B_{12}^1;$$
$$F_{12}(T_5) = B_1^1 B_3^1 B_5^1 B_9^1 B_{13}^1;$$
$$F_{13}(T_6) = B_2^1 B_4^1 B_6^1 B_{10}^1 B_{14}^1;$$
$$F_2(T_1) = B_1^1; F_3(T_2) = B_2^1.$$

Синтез діагностичної матриці для одного модуля дискретного косинусного перетворення з бібліотеки Xilinx у вигляді функціонального покриття показаний у лістингу 4.1.

Лістинг 4.1 – Фрагмент функціонального покриття

```
c0: coverpoint xin
{
bins minus_big={[128:235]};
bins minus_sm={[236:255]};
bins plus_big={[21:127]};
bins plus_sm={[1:20]};
bins zero={0};
}
c1: coverpoint dct_2d
{
bins minus_big={[128:235]};
bins minus_sm={[236:255]};
bins plus_big={[21:127]};
bins plus_sm={[1:20]};
bins zero={0};
bins zero2=(0=>0);
}
endgroup
```

Розроблені також інші 12 модулів транзакційного графа, активізаційні ТАВ-матриці і логічні функції для тестування і виявлення несправностей в модулі дискретного косинусного перетворення. Фрагмент механізму моніторів наведено на лістингу 4.2.

Лістинг 4.2 – Фрагмент коду механізму моніторів

```
sequence first( reg[7:0] a, reg[7:0]b);
 reg[7:0] d;
 (!RST,d=a)
##7 (b==d);
 endsequence
property f(a,b);
 @(posedge CLK)
 // disable iff(RST||$isunknown(a)) first(a,b);
!RST |=> first(a,b);
endproperty
odin:assert property (f(xin,xa7_in))
 // $display("Very good");
else $error("The end, xin =%b,xa7_in=%b", $past(xin, 7),xa7_in);
```

Тестування дискретного косинусного перетворення у середовищі Riviera фірми Aldec дозволило виявити некоректності у семи рядках HDL-моделі:

```
//add_sub1a <= xa7_reg + xa0_reg;//
```

Наступна коректировка коду дозволила отримати наступний фрагмент (лістинг 4.3).

Лістинг 4.3 – Скоригований фрагмент коду

```
add_sub1a <= ({xa7_reg[8],xa7_reg} + {xa0_reg[8],xa0_reg});
add_sub2a <= ({xa6_reg[8],xa6_reg} +{xa1_reg[8],xa1_reg});
add_sub3a <= ({xa5_reg[8],xa5_reg} +{xa2_reg[8],xa2_reg});
add_sub4a <= ({xa4_reg[8],xa4_reg} + {xa3_reg[8],xa3_reg});
end
else if (toggleA == 1'b0)
begin
add_sub1a <= ({xa7_reg[8],xa7_reg} - {xa0_reg[8],xa0_reg});
add_sub2a <= ({xa6_reg[8],xa6_reg} - {xa1_reg[8],xa1_reg});
add_sub3a <= ({xa5_reg[8],xa5_reg} - {xa2_reg[8],xa2_reg});
add_sub4a <= ({xa4_reg[8],xa4_reg} - {xa3_reg[8],xa3_reg});
```

Практична імплементація моделей і методів верифікації інтегрована у середовище моделювання Riviera фірми Aldec Inc, рис. 4.8. Нові асерції і модулі діагностування, додані до системи, дозволяють поліпшити існуючий процес верифікації, що дозволяє на 15% скоротити час розробки цифрового продукту.

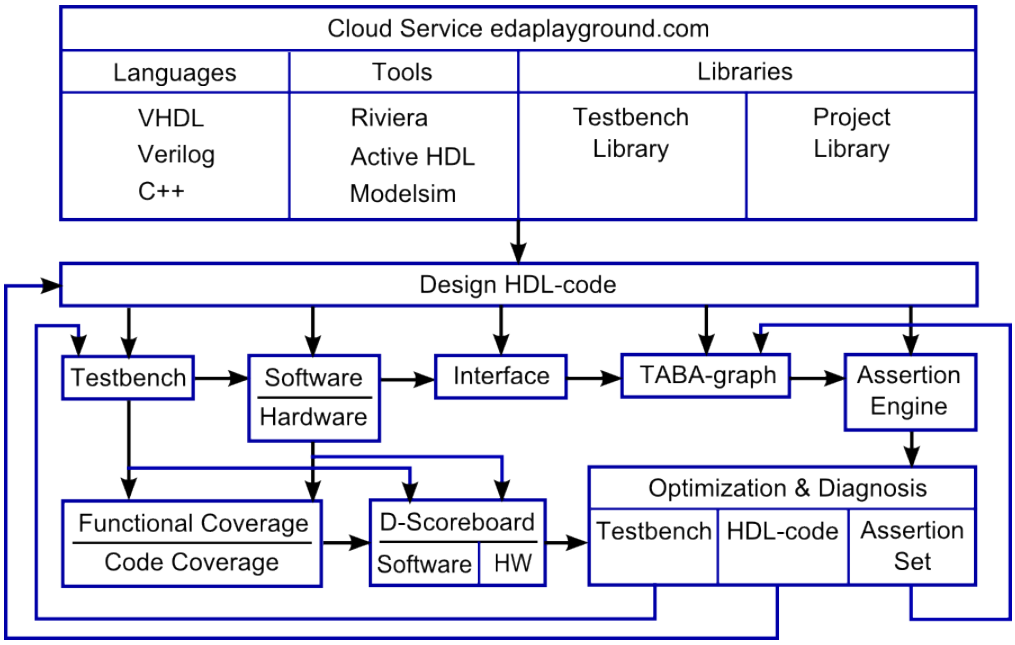


Рисунок 4.8 – Імплементація результатів у систему Riviera

Застосування асерцій дає можливість зменшити довжину testbench і значно скоротити (x3) час проектування (рис. 4.9), яке є найбільш витратним. Механізм ассерцій дозволяє збільшити глибину діагностування функціональних порушень в програмних блоках до рівня 10-20 операторів HDL-коду.

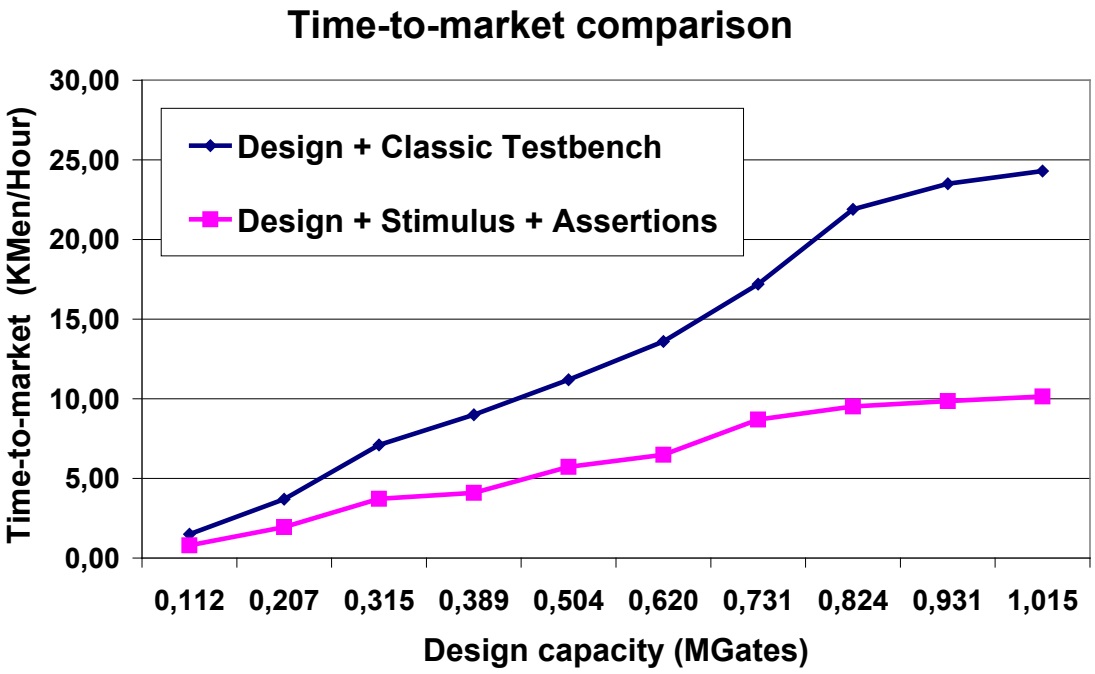


Рисунок 4.9 – Порівняльний аналіз методів верифікації

Завдяки взаємодії засобів моделювання та механізму асерцій, автоматично розміщених всередині HDL-коду, з'являється доступ засобів діагностування до значень всіх внутрішніх сигналів. Це дозволяє швидко визначити місце розташування і тип функціонального порушення, а також скоротити час виявлення помилок при використанні методології проектування зверху вниз. Застосування асерцій для 50 реальних проектів (від 5000 до 5000000 вентилів) дозволило отримати сотні спеціалізованих рішень, включених до бібліотеки верифікаційних шаблонів VTL, які є узагальненням найбільш популярних на ринку EDA (Electronic Design Automation) обмежень темпоральной верифікації для широкого класу цифрових продуктів. Програмна реалізація запропонованої системи аналізу асерцій та діагностування HDL-коду є частиною багатофункціонального інтегрованого середовища Aldec Riviera для моделювання та верифікації HDL-моделей.

Висока продуктивність і технологічність поєднання системи аналізу асерцій і HDL-симулятора компанії Aldec в значній мірі досягається за рахунок інтеграції з внутрішніми компонентами симулятора, в тому числі,

компіляторами HDL-мов. Обробка результатів системи аналізу асерцій забезпечується набором візуальних засобів системи Riviera, що дозволяють полегшити діагностування та усунення функціональних порушень. Модель аналізу асерцій може бути імплементована також в апаратні засоби з певними обмеженнями на підмножину підтримуваних мовних структур. Продукти Riviera, включаючи компоненти темпоральної верифікації асерцій, які дозволяють поліпшити якість проекту на 3-5%, в даний час займають провідні позиції на світовому IT-ринку з кількістю системних інсталяцій 5000 на рік у 200 компаніях і університетах більш ніж 20 країн світу.

4.5 Висновки до розділу 4

1. У розділі представлені інфраструктура і технології аналізу цифрових систем. Запропоновані модель транзакційного графа і метод діагностування цифрових систем на кристаллах, орієнтовані на значне зменшення часу виявлення несправних блоків і пам'яті для зберігання компактної діагностичної матриці, яка описує тернарні відношення у форматі: монітор-орієнтовані тест-сегменти, і призначені для виявлення несправностей функціональних компонентів програмно-апаратних систем.

2. Введено новий критерій якості діагностування, що представляє собою функцію, залежну від структури графа, тестів і асерційних моніторів. Для поліпшення якості діагностування існують два альтернативні шляхи: збільшення набору тестових сегментів для розпізнавання еквівалентних несправних блоків або додавання асерційних моніторів на транзитних вершинах активізаційного графа HDL-коду. Запропонований критерій дозволяє прийняти правильне рішення.

3. Удосконалено ТАВ-алгоритм виявлення функціональних порушень у програмному або апаратному забезпеченні. Він відрізняється від аналогів використанням xor операції, що дозволяє підвищити продуктивність діагностування одиночних і кратних несправних блоків за рахунок паралельного аналізу ТАВ-матриці, застосування граничного сканування на основі стандарту IEEE 1500, а також векторних операцій and, or, xor.

4. Розроблено модель діагностування функціональності системи на кристалі у вигляді мультідерева і метод обходу дерева, імплементований у движок для виявлення несправних блоків із заданою глибиною. Модель і метод дозволяють істотно збільшити продуктивність програмного і апаратного забезпечення інфраструктури IP.

5. Тестова верифікація методу діагностування виконана на трьох реальних прикладах, поданих компонентами SoC фільтра косинусного перетворення, який показав спроможність результатів щодо зменшення часу виявлення несправностей і пам'яті для зберігання діагностичної інформації, а також збільшення глибини діагностування цифрового модуля.

5 «КВАНТОВИЙ» ПРОЦЕСОР ОПТИМАЛЬНОГО ПОКРИТТЯ

Пропонуються кубітні (квантові) структури даних і обчислювальних процесів для суттєвого підвищення швидкодії при вирішенні завдань дискретної оптимізації. Описуються апаратно орієнтовані моделі паралельного (за один цикл) обчислення булеана (множини всіх підмножин) на універсумі з n примітивів для вирішення завдань покриття, мінімізації булевих функцій, стиснення даних, синтезу та аналізу цифрових систем за рахунок реалізації процесорної структури у формі діаграми Хасе.

5.1 Постановка задачі

Мета створення кубіт-процесора – суттєве зменшення часу при вирішенні завдань оптимізації шляхом паралельного обчислення векторних логічних операцій [63, 74, 79] над множиною всіх підмножин від примітивних компонентів за рахунок збільшення пам'яті для зберігання проміжних даних.

Завдання: 1) Визначення структур даних для взяття булеана при вирішенні задачі покриття стовпців матриці $M = \left|M_{ij}\right|, i = \overline{1,m}; j = \overline{1,n}$ одиничними значеннями рядків. Зокрема, при m = n = 8, необхідно виконати паралельно логічну операцію над 256 варіантами всіх можливих поєднань векторів (рядків матриці), що становлять булеан. 2) Система команд процесора повинна включати наступні операції (and, or, xor) над векторами (словами), розмірності m. 3) Розробка архітектури кубіт-процесора для паралельного обчислення $2^n - 1$ варіантов сполучень, спрямованих на оптимальне рішення NP-повної задачі покриття. 4) Реалізація прототипу кубіт-процесора на базі програмованої логіки PLD і верифікація (валідація) апаратного рішення [63, 74, 79-88] на прикладах мінімізації булевих функцій. 5) Приведення інших

практичних завдань дискретної оптимізації до форми задачі покриття для подальшого вирішення на кубіт-процесорі.

Як приклад пропонується вирішити завдання пошуку оптимального одиничного покриття всіх стовпців мінімальним числом рядків матриці M, представленої нижче:

M	1	2	3	4	5	6	7	8
a	1	.	.	.	.	1	.	.
b	.	.	1	.	.	.	1	.
c	1	.	.	.	1	.	1	.
d	.	1	.	1	.	.	.	1
e	.	1	.	.	1	.	.	.
f	1	.	1	.	.	1	1	.
g	.	1	.	1	.	.	.	1
h	.	.	1	.	1	.	.	.

Для цього необхідно зробити перебір всіх 255 сполучень: з восьми по одному рядку, по два, три, чотири, п'ять, шість, сім і вісім. Мінімальна кількість примітивів (рядків), яке формує покриття, є оптимальне рішення. Таких рішень може бути кілька. Діаграма Хасе є компромісна пропозиція, відносно часу і пам'яті, або така стратегія вирішення задачі покриття, коли раніше отриманий результат згодом використовується для створення більш складної суперпозиції. Тому для кожної таблиці покриття, що містить n примітивів (рядків), необхідно генерувати власну мультипроцесорну структуру у формі діаграми Хасе, яка далі має бути використана для майже паралельного рішення NP-повної задачі. Наприклад, для чотирьох рядків таблиці покриття діаграма Хасе – структура мультипроцесора – матиме вигляд, представлений на рис. 5.1.

Оптимальні розв’язки задачі покриття для матриці M, яка генерирує 255 варіантів можливих сполучень, подані рядками у формі ДНФ: $C = fgh \vee efg \vee cdf$. Автомат управління обчислювальним процесом для квантової структури шляхом висхідного аналізу вершин графа заснований на послідовному виконанні наступних кроків:

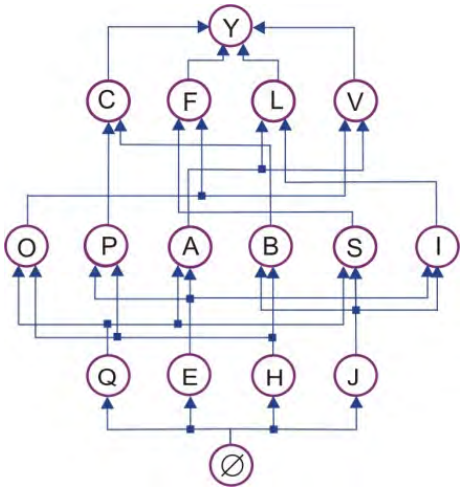


Рисунок 5.1 – Квантова структура обчислювальних процесів

1. Занесення інформації про примітиви в регістри (матриці) $L_i^1 = P_i$ першого рівня з подальшим аналізом якості покриття кожним примітивом в двійковому форматі (1 – є покриття, 0 – немає покриття). Якщо один з примітивів організовує покриття $\bigwedge_{j=1}^{m} L_{ij}^1 = 1$, процес аналізу структури Хасе закінчується. Інакше – виконання переходу (r = r + 1) на наступний рівень графа:

$$L_i^1 = P_i \rightarrow \bigwedge_{j=1}^{m} L_{ij}^1 = \begin{cases} 0 \rightarrow n = n + 1; \\ 1 \rightarrow \text{end}. \end{cases}$$

2. Ініціювання команди обробки чергового (другого) рівня. Послідовне виконання векторних (матричних) операцій (or $L_i^r = L_{ij}^{r-1} \bigwedge_{j=1}^{m} L_{tj}^{r-1}$, and $\bigwedge_{j=1}^{m} L_{ij}^r = 1$) з метою аналізу покриває здатності поєднань примітивних елементів r-рівня. Тут $t = \overline{1,m}, i = \overline{1,m}, r = \overline{1,n}$, n – число ярусів або кількість рядків у таблиці покриття, m – число стовпців в ній. Якщо існує поєднання на розглянутому рівні, що створює повне покриття, яке формує оцінку, рівну 1, то обробка

всіх наступних рівнів процесора не виконується. В іншому випадку виконується перехід для аналізу наступного рівня процесорної структури:

$$L_i^r = L_{ij}^{r-1} \overset{m}{\underset{j=1}{\wedge}} L_{tj}^{r-1} \to \overset{m}{\underset{j=1}{\wedge}} L_{ij}^{r} = \begin{cases} 0 \to r = r + 1; \\ 1 \to \text{end}. \end{cases}$$

Для пошуку оптимального покриття завжди достатньо двох елементів нижнього рівня, що означає – всі операційні вершини мають два регістрових (матричних входи), що істотно зменшує вартість витрат апаратури. Кількість часових тактів для обробки структури процесора в гіршому випадку дорівнює n. Можна створити алгоритм пошуку оптимального покриття шляхом низхідного аналізу вершин графа. У цьому випадку, при знаходженні повного покриття в одному з ярусів необхідний ще один спуск по структурі, щоб переконатися про відсутність в нижньому сусідньому ярусі повного покриття. При позитивній відповіді на дане запитання отримане рішення є оптимальним. Інакше – необхідно виконувати спуск до такого рівня, коли, більш нижній, сусідній ярус не міститиме повного покриття. Вершини процесорної структури можуть мати більше однієї бінарної (унарної) регістрової логічної операції. Тоді необхідно створювати найпростіший дешифратор команд для активізації, наприклад, операцій: and, or, xor, not. Таким чином, переваги кубітного Хасе процесора (Quantum Hasse Processor – QHP) полягають у можливості використовувати не більше, ніж двохвходові схеми векторних логічних операцій (and, or, xor), а значить, в істотному зменшенні вартості за Квайном реалізації процесорних елементів (вершин) і пам'яті за рахунок застосування послідовних обчислень і незначного збільшення часу обробки всіх вершин графа Хасе. Для кожної вершини використовується критерій якості покриття – наявність всіх одиниць в координатах вектора-результата. Якщо критерій якості виконується, то всі інші обчислення можна не проводити, оскільки діаграма Хасе є строго ієрархічна структура по числу поєднань в кожному ярусі. Це означає,

найкраще рішення знаходиться на більш низькому рівні ієрархії. Варіанти одного рівня рівнозначні з реалізації (вартості), тому отримане першим якісне покриття $(Q = \sum_{i=1}^{n} q_i = n)$ є краще рішення, що передбачає зупинку всіх наступних обчислень за стратегією діаграми Хасе. Враховуючи послідовно-паралельну стратегію аналіз вершин графа, час (цикл) обробки всіх примітивів QH-процесора визначається числом рівнів ієрархії (кількістю бітів (примітивів, рядків в таблиці покриття) в кубітної змінної), помноженої на час аналізу однієї вершини: $T = \log_2 2^n \times t = t \times n$. При цьому довжина m рядку таблиці покриття не впливає на оцінку швидкодії. Аналіз вершини включає дві команди: логічну (and, or, xor) і операцію обчислення критерію якості покриття у формі скаляра шляхом застосування функції and до всіх розрядів вектора-результата:

$$m_{ir,j} = M_{i,j} \vee M_{r,j}, (j = \overline{1,n}; \{i \neq r\} = \overline{1,m};);$$
$$m_{ir}^{s} = \wedge m_{ir,j} = \wedge (M_{i,j} \vee M_{r,j})$$

Апаратні витрати на реалізацію QH-процесора залежать від сумарного числа вершин графа Хасе і від кількості бітів (розрядів) в рядку таблиці покриття: $H = 2^n \times k \times m$, де k – коефіцієнт апаратної реалізації (складності) одного розряду бінарної векторної логічної операції і подальшої команди обчислення критерію якості покриття. Таким чином, висока швидкодія розв'язання задачі покриття досягається істотним підвищенням витрат апаратури (в $2^n \times k \times m / k \times m \times n = 2^n / n$ разів у порівнянні з послідовною обробкою графових вершин), яке забезпечує компромісний варіант між повністю паралельною структурою обчислювальних процесів (тут витрати апаратури визначаються числом примітивів в кожній вершині $H = k \times m \times n \times 2^n$, а зростання апаратури складає 2^n разів) і послідовними

обчисленнями однопроцесорного комп'ютера (тут швидкодія обробки графа Хасе дорівнює $T = t \times 2^n$, а апаратні витрати дорівнюють $H^* = k \times m \times n$). Зменшення апаратури у порівеянні зс паралельним варіантом обробки графа складає $Q^H = k \times m \times n \times 2^n / k \times m \times 2^n = n$. Як наслідок істотної апаратної надмірності, зменшення часу аналізу вершин графа в порівнянні з послідовною обробкою структури має наступну оцінку: $Q^T = \frac{t \times 2^n}{t \times n} = \frac{2^n}{n}$.

Таким чином, описані вище апаратно орієнтовані моделі паралельного (за один цикл) обчислення булеана (множини всіх підмножин) на універсумі з n примітивів для вирішення задач покриття, мінімізації булевих функцій, стиснення даних, синтезу та аналізу цифрових систем за рахунок реалізації процесорної структури у формі діаграмі Хасе. Прототип квантового пристрою, реалізованого на основі програмованої логіки для оптимального розв’язання задачі покриття при аналізі кіберпростору (представлений нижче у підрозділі 5.4).

5.2 Метод Windows-декомпозиції для розв’язання задачі покриття

Сутність проблеми полягає у великій розмірності таблиці покриття, наприклад 1000x1000, коли практично неможливо за прийнятний час отримати оптимальне розв’язання задачі покриття, що має експоненціальну складність. Тоді необхідно йти шляхом квазіоптимізації, що дозволяє істотно зменшити часовий чинник і зробити його прийнятним для практики. Нижче пропонується кілька стратегій для зниження часових витрат при обробці таблиць покриття великої розмірності.

1) Метод декомпозиції таблиці покриття на сегменти рядків

$$C = \{C_1, C_2, ..., C_i, ..., C_n\}, C_i = (C_{i1}, C_{i2}, ..., C_{ij}, ..., C_{im}).$$

Кожний сегмент містить m рядків за умови, что розмірність таблиці дорівнює $|C| = n \times m \times k$, а сегмента – $|C_{ij}| = m \times k$. Тут m – кількість рядків в сегменті або число змінних в процесорі Хасе, k – довжина рядка таблиці покриття, яка повинна оброблятися паралельно, n – число сегментів. Операція диз'юнкції виконується над усіма рядками в кожному сегменті, по закінченні якої, розмірність таблиці знижується в m раз. Якщо розмірність отриманої таблиці сегментів більше, ніж кількість входів Хасе процесора, то до неї знову застосовується процедура декомпозиції. Таким чином, вихідна таблиця покриття перетворюється в багаторівневу піраміду, де верхня площадка визначається числом рядків, не більшим, ніж кількість входів процесора Хасе. Наприклад, якщо процесор має 8 змінних для паралельної обробки восьми рядків таблиці покриття, то попередні (нижні) рівні повинні збільшуватися в 8 разів:

Level	m
1	8
2	64
3	512
4	4096
5	32768
...	...

Це означає, що для паралельної обробки таблиці, розмірністю 32768x32768 необхідно виконати всього 5 циклів обробки матриці мультипроцесорною системою Хасе. При цьому на кожному рівні вибираються сегменти, які беруть участь у формуванні покриття, задекларованого на більш високому рівні. Слід мати на увазі, що попередньо виконується сегментування всього простору рядків таблиці покриття до тих пір, поки на останньому рівні не залишиться число сегментів, менше або рівне восьми, наприклад:

Level	m
1	32768
2	4096
3	512
4	64
5	8

На кожному рівні виконується обчислення векторів якості покриття сегментів більш високого рівня на основі застосування векторної or-операції, подано нижче: $C_{ij} = \bigoplus_{r=1}^{k} C_{ijr}$. Наприклад, для обробки таблиці покриття, що мають розмірність 16x16:

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	1	.	.	.	.	.	.	.	1	.	.	.	.	.	.	1
b	.	.	1	.	.	.	1	.	.	.	.	.	1	.	1	.
c	1	.	.	.	1	.	.	.	.	.	1	.	.	.	.	.
d	.	.	.	1	.	.	.	1	.	.	1	.	.	.	.	.
e	.	.	.	.	1	.	.	.	1	.	.	.	.	1	.	.
f	1	.	1	.	.	.	1	.	.	.	.	.	.	1	.	1
g	.	.	.	1	.	.	.	1	.	.	.	.	.	.	.	.
h	.	.	1	.	1	.	.	.	1	.	1	.	1	.	1	.
i	1	.	1	.	.	1	.	.	1	.	.	.	.	.	.	1
j	1	.	.	1	.	.	.	1	.	.	.	1	.	.	.	.
k	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
l	.	1	.	1	.	.	.	.	1	.	1	.	.	.	.	.
m	1	.	.	.	1	1	.	.	.	.	.	.	1	.	.	.
n	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.
p	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.
q	.	.	.	1	.	1	.	1	1	.	.	.	.	.	1	.

необхідно виконати її розбиття на 4 сегменти, де в кожному з них мається 4 вектори:

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	1	.	.	.	.	.	.	.	1	.	.	.	.	.	.	1
b	.	.	1	.	.	.	1	.	.	.	.	.	1	.	1	.
c	1	.	.	.	1	.	.	.	.	.	1	.	.	.	.	.
d	.	.	.	1	.	.	.	1	.	.	1	.	.	.	.	.
e	.	.	.	.	1	.	.	.	1	.	.	.	.	1	.	.
f	1	.	1	.	.	.	1	.	.	.	.	.	.	1	.	1
g	.	.	.	1	.	.	.	1	.	.	.	.	.	.	.	.
h	.	.	1	.	1	.	.	.	1	.	1	.	1	.	1	.
i	.	1	.	.	1	.	.	1	.	.	.	.	.	.	1	
j	.	.	1	.	.	.	1	.	.	.	.	1	.	.	.	.
k	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
l	.	1	.	1	.	.	.	.	1	.	1	.	.	.	.	.
m	.	.	.	1	1	.	.	.	.	.	.	.	1	.	.	.
n	.	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.
p	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.
q	.	.	.	1	.	1	.	1	1	.	.	.	.	.	1	.
	1	.	1	1	1	.	1	1	1	.	1	.	1	.	1	1
	1	.	1	1	1	.	1	1	1	.	1	.	1	1	1	1
	.	1	1	.	.	1	.	1	1	.	1	1	.	.	.	1
	.	.	1	1	1	1	.	1	1	1	1	.	1	.	1	.

Після цього виконується операція диз'юнкції над усіма s-векторами кожного сегмента, а результат заноситься в (s+1)-вектор відповідного сегмента. Всі такі вектори далі використовуються для вирішення задачі покриття, але вже на рівні векторів більш високої ієрархії.

2) Метод декомпозиції таблиці покриття на двомірні сегменти.

Після віконного розбиття виконується побудова (s+1, t+1)-векторів на основі векторної операції диз'юнкції над усіма s-векторами кожного t-вікна, а результат заноситься в (s+1, t+1)-вектор відповідного (s+1, t+1)-вікна. Всі такі вектори далі використовуються для вирішення задачі покриття на більш високому рівні (s+1, t+1)-вектор.

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
a	1	.	.	.	.	.	.	.	1	.	.	.	.	.	.	1	.	.	.	.	.	1	.	.
b	.	.	1	.	.	.	1	.	.	.	.	.	1	.	1	.	.	.	.	.	.	.	.	1
c	1	.	.	.	1	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	1	.	.	.
d	.	.	.	1	.	.	.	1	.	.	1	.	.	.	.	.	1	.	.	.	1	.	.	.
e	.	.	.	.	1	.	.	.	1	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.
f	1	.	1	.	.	.	1	.	.	.	.	.	.	1	.	1	.	.	.	1	.	.	.	.
g	.	.	.	1	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.
h	.	.	1	.	1	.	.	.	1	.	1	.	1	.	1	.	.	.	.	.	.	.	.	1
i	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	1	.	.
j	.	.	.	.	.	.	.	1	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.
k	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.
l	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.
m	.	.	.	.	.	1	.	.	.	.	.	.	1	.	.	.	.	.	.	1	.	.	.	1
n	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	1	.	.
p	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.
q	.	.	.	.	.	1	.	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.	1	.
r	1	.	.	.	.	.	.	1	.	.	1	.	1	.	.	1	.	.	.	.	1	.	.	.
s	.	1	1	.	.	1	.	.	1	.	1	.	1	.	.	.	.	.	1	.	.	.	.	.
t	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
u	.	1	.	.	.	.	.	.	.	.	1	.	.	.	.	1	.	1	.	.	.	.	.	.
v	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
w	1	.	1	.	1	.	.	.	1	.	.	.	1	1	.	.	.	.	1	.	.	.	.	.
x	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
y	.	1	.	.	.	.	.	1	1	.	.	.	1	.	.	.	.	.	.	1	.	.	.	.
q1	1	.	1	1	1	.	1	1	1	.	1	.	1	1	1	1	1	.	.	1	1	1	1	1
q2	.	1	.	.	.	1	.	1	.	.	.	1	1	.	1	1	.	1	1	1	.	1	1	1
q3	1	1	1	.	1	1	.	1	1	1	1	.	1	1	.	1	.	1	1	1	1	.	.	.

Розбиття таблиці на сегменти і на вікна мають переваги, пов'язані зі зменшенням часу вирішення задачі за рахунок розпаралелювання, де всі сегменти (вікна) обробляються одночасно. Недолік визначається отриманням квазіоптимального рішення, яке може відрізнятися від мінімального покриття. Нижче представлені чисельні оцінки витрат і переваг від впровадження двох стратегій запропонованого методу оптимізації:

$$H^r = \frac{2^n}{(2^{n_1} + 2^{n_2} + ... + 2^{n_i} + ... + 2^{n_k}) + 2^{n/u}}$$

$$H^u = \frac{2^n}{[(2^{n_1} + 2^{n_2} + ... + 2^{n_i} + ... + 2^{n_k}) + 2^{n/u}] \times \frac{r}{u}}$$

$$n = 1000; n_1 = n_2 = ... = n_i = ... = n_{10} = 10;\ r = 1000; u = 100 \rightarrow$$

$$H^r = \frac{2^{1000}}{10 \times 2^{100} + 2^{1000/100}} = \frac{2^{1000}}{10 \times 2^{100} + 2^{10}}$$

$$H^u = \frac{2^{1000}}{(10 \times 2^{100} + 2^{1000/100}) \times \frac{1000}{10}} = \frac{2^{1000}}{(10 \times 2^{100} + 2^{10}) \times 100} = \frac{2^{1000}}{1000 \times 2^{100} + 100 \times 2^{10}}$$

Тут перша оцінка – виграш у часових витратах розв'язання задачі покриття при розбитті таблиці на k сегментів, друга, коли додатково і рядок, довжиною r, розбивається на сегменти, довжиною u, де число сегментів дорівнює $\frac{r}{u}$. Виграш формується за рахунок одночасної обробки сегментів або вікон. Після чого задача зводиться до покриття таблиці сегментами. Потім формується точний результат у формі рядків сегментів, що беруть участь в покритті вихідної таблиці. Тут метод орієнтований на паралельну обробку всіх квадрантів (вікон) процесорами Хасе, число яких визначається кількістю квадрантів. Таким чином, платою за високу швидкодію паралельної обробки є апаратна надлишковість, що залежить від числа вікон на найнижчому рівні ієрархії таблиці покриття.

5.3 Апаратна реализація QH-процесора

Для генерації коду моделей мультипроцесора Хасе різної конфігурації за кількістю входів або бітів в кубіті розроблений програмний генератор. На основі завдання вихідних даних: кількості вхідних векторів, їх розрядності та

сучасних технологій [79-93] виконується генерація Verilog-коду. Програма написана з використанням мови Python 2.7, і містить 255 рядків коду.

Структура класів, розроблена за допомогою інструменту ArgoUML, представлена на рис. 5.2. Вона включає основний клас Generator, а також генератор індексів внутрішніх регістрів моделі, який виділений в окремий клас genIndex.

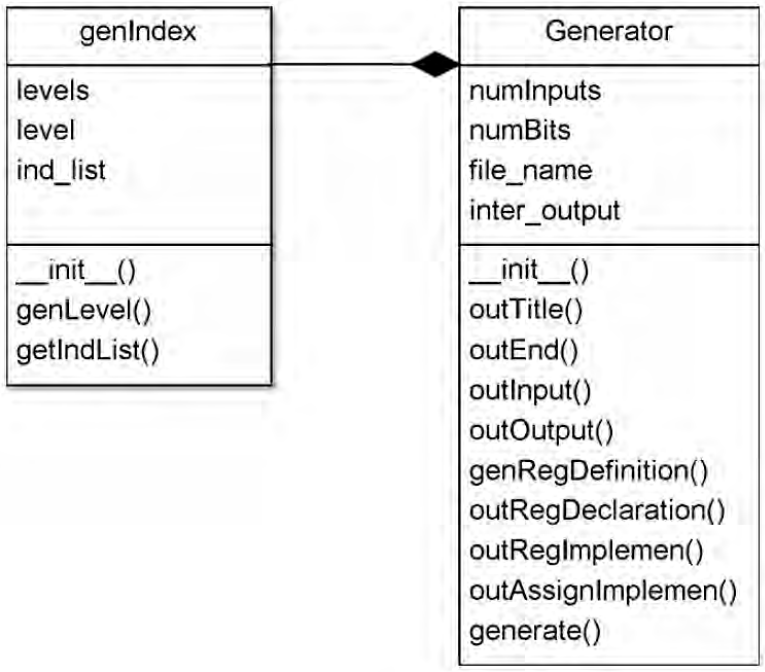


Рисунок 5.2 – Структура класів програми (ArgoUML)

Змінні класу genIndex: levels – кількість рівнів моделі (відповідає числу входів процесора Xace), level – поточний рівень для генерування індексів регістрів, ind_lisc – список строкових значень індексів. Два методи класу: genLevel () і getIndList () генерують і повертають список індексів. Об'єкти класу genIndex використовуються в методах створення опису та реалізації регістрів класу Generator.

Клас Generator містить поля: numInputs – число входів моделі процесора Xace, numBits – розрядність вхідних / вихідних векторів, file_name – ім'я файлу моделі, inter_output – дозволяє при необхідності відключати

генерування виходів ознак для регістрів внутрішніх рівнів. Метод ініціалізації змінних задає їх значення за замовчуванням:

```
def __init__(self, numInputs = 4, numBits = 8, file_name = "model.v",
                    inter_output = True):
```

Будь-який з параметрів може бути змінений при створенні екземпляра класу Generator, наприклад:

```
numInputs = 16
numBits = 8
obj1 = Generator(numInputs, numBits, "model"+str(numInputs)+".v",
False).
```

Розподіл Verilog-моделі на складові зображено на рис. 5.3. Для реалізації кожної частини моделі створено окремий метод класу Generator: outTitle (), outInput (), outOutput (), outRegDeclaration (), outRegImplemen (), outAssignImplemen () і outEnd () відповідно. Метод genRegDefinition (name1, name2, vec = True) викликається з outRegDeclaration (). Метод generate (), використовуючи інші методи класу, виконує генерацію цілої моделі пристрою і запис її у файл.

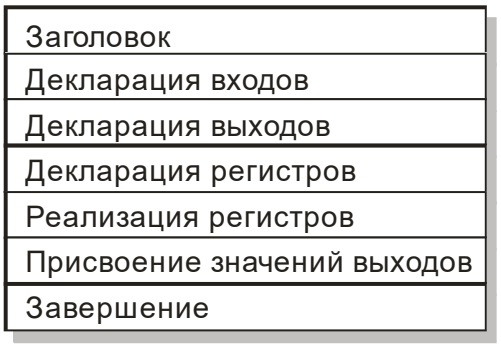


Рисунок 5.3 – Організація Verilog-моделі

Крім того, функції outTitle () і outEnd () відкривають і закривають зовнішній файл:

```
def outTitle(self):
    self.f = open(self.file_name, 'w')
    self.f.write("module device\n  (input clk, rst,\n")

def outEnd(self):
     self.f.write("\nendmodule")
     self.f.close()
```

Табл. 5.1 ілюструє збільшення числа зовнішніх ліній пристрою зі зростанням кількості входів (numInputs). При цьому очевидно, що розрядність вхідних сигналів (numBits) несуттєво впливає на загальне число входів і виходів моделі.

Таблиця 5.1 – Залежність кількості зовнішніх ліній моделі від числа входів (numInputs) та їх розрядності (numBits)

numInputs	numBits							
	4	8	12	16	20	24	28	32
4	31	55	79	103	127	151	175	199
6	83	115	147	179	211	243	275	307
8	279	319	359	399	439	479	519	559
10	1 051	1 099	1 147	1 195	1 243	1 291	1 339	1 387
12	4 127	4 183	4 239	4 295	4 351	4 407	4 463	4 519
14	16 419	16 483	16 547	16 611	16 675	16 739	16 803	16 867
16	65 575	65 647	65 719	65 791	65 863	65 935	66 007	66 079
18	262 187	262 267	262 347	262 427	262 507	262 587	262 667	262 747

Модель пристрою, з конфігурацією з 16 входів, де розрядність входів дорівнює 8 бітам, реалізована з використанням мікросхеми фірми Xilinx серії Spartan-3E: xc3s1200e.

При цьому отримана схема містить 3337 тригерів FF і 3233 4-х входові таблиці перетворення LUT.

```
Logic Utilization:
  Number of Slice Flip Flops:          3,337 out of  17,344   19%
  Number of 4 input LUTs:              3,233 out of  17,344   18%
```

Для 16 входів квантового процесора генератор створений за допомогою мовних засобів Python 2.7 синтезував 288738 рядків (16 кілобайт) коду мови Verilog за 65 секунд на двоядерному процесорі Інтел з тактовою частотою 2,1 Ггц. Виходячи з результатів статичного часового аналізу (рис. 5.4), мінімальний робочий період синхросигналу дорівнює 9.2 ns, що відповідає частоті 108 МГц.

```
Clock clk to Pad
------------+------------+------------------+--------+
            | clk (edge) |                  | Clock  |
Destination |   to PAD   |Internal Clock(s) | Phase  |
------------+------------+------------------+--------+
ol5<0>      |    7.280(R)|clk_BUFGP         |   0.000|
ol5<1>      |    7.512(R)|clk_BUFGP         |   0.000|
ol5<2>      |    6.999(R)|clk_BUFGP         |   0.000|
ol5<3>      |    8.072(R)|clk_BUFGP         |   0.000|
ol5<4>      |    7.866(R)|clk_BUFGP         |   0.000|
ol5<5>      |    8.242(R)|clk_BUFGP         |   0.000|
ol5<6>      |    7.884(R)|clk_BUFGP         |   0.000|
ol5<7>      |    9.186(R)|clk_BUFGP         |   0.000|
ol6         |    7.680(R)|clk_BUFGP         |   0.000|
------------+------------+------------------+--------+

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    4.250|         |         |         |
---------------+---------+---------+---------+---------+
```

Рисунок 5.4 – Часові параметри Хасе процесора на 16 біт

Інтерфейс GUI програми генератора кодів, розроблений для генерування масштабованих моделей Хасе мультипроцесора, наведено на рис. 5.5.

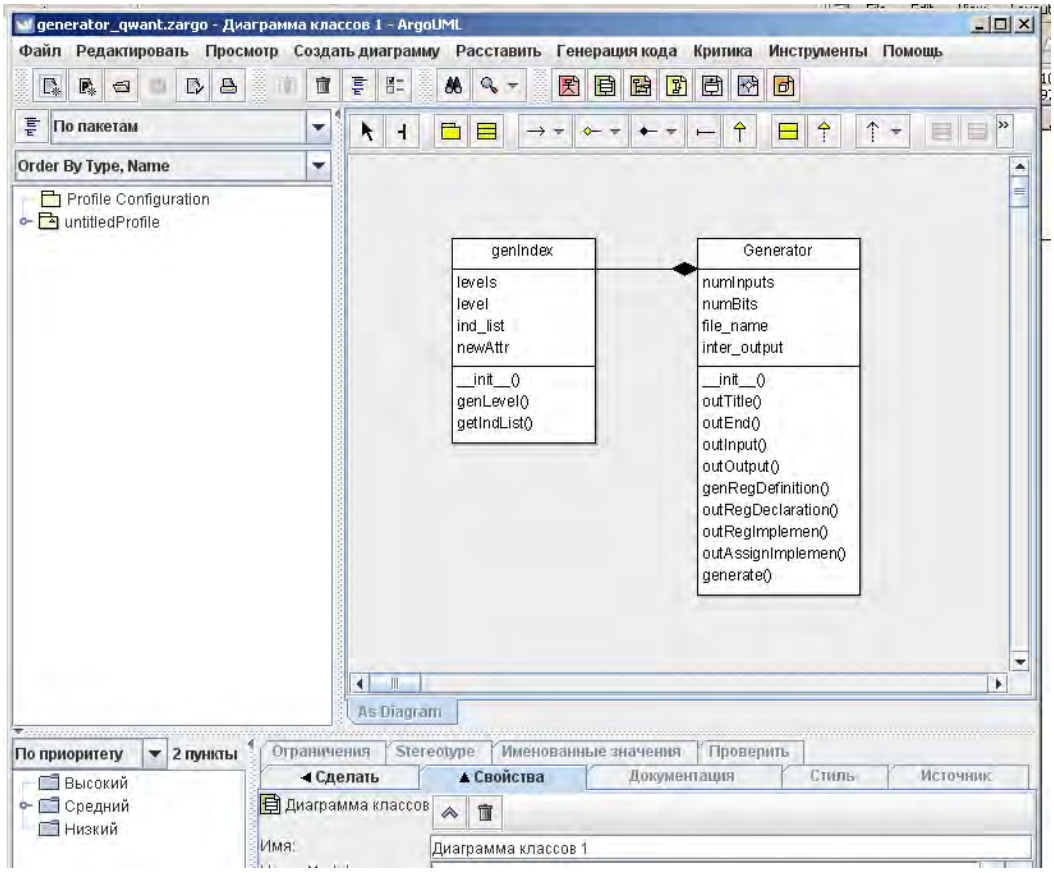
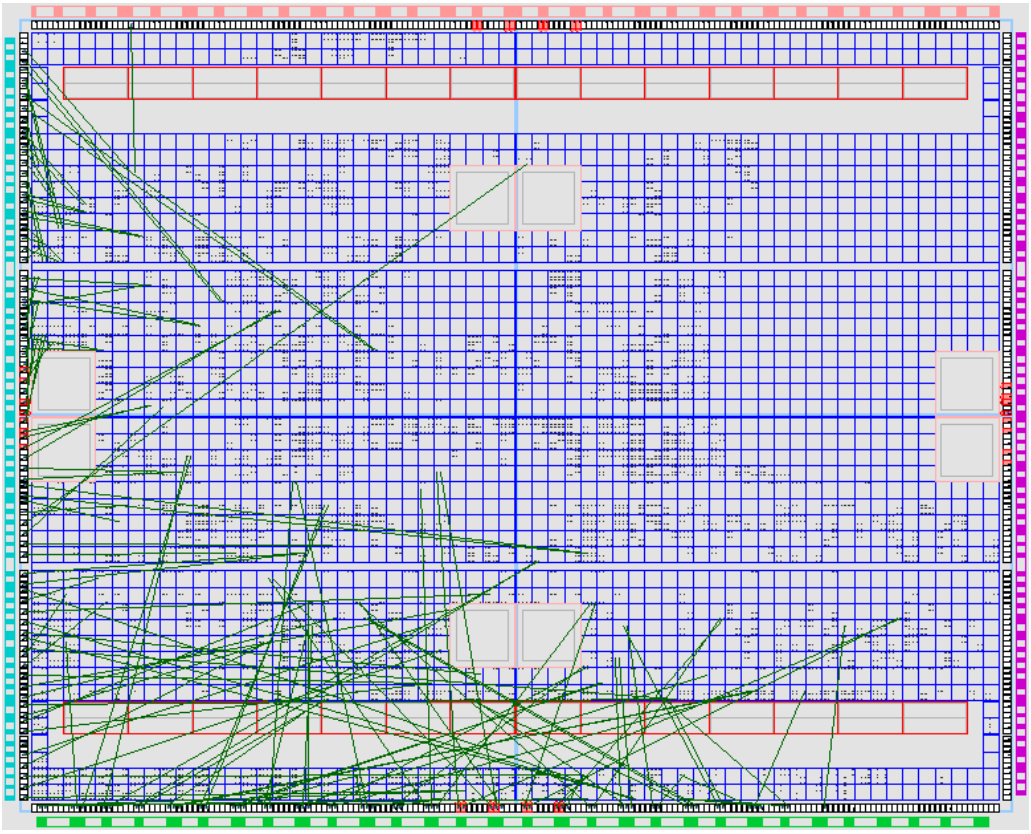


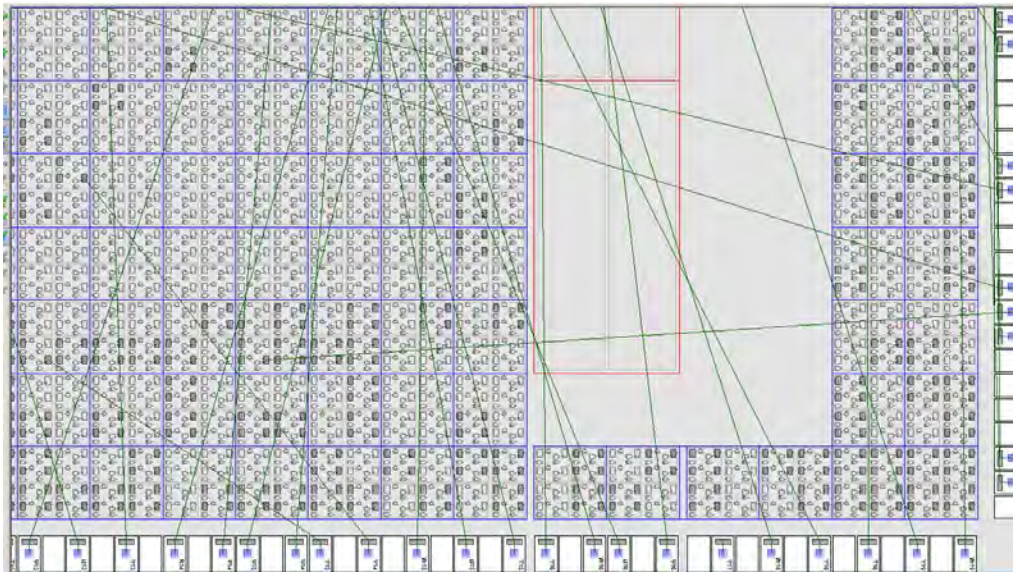
Рисунок 5.5 – Інтерфейс GUI програми генератора кодів

Інтерфейс дозволяє здійснювати автоматичне генерування коду на мові опису апаратури Verilog, задавати і обчислювати основні характеристики проекту, пов'язані з числом входів, довжиною або розрядністю регістрів, а також переходити до фази RTL-синтезу проекту та подальшої його верифікації.

Синтез, мепінг, трасування та розміщення проекту – Хасе процесора – виконана на основі використання відкритого програмного продукту ISE Xilinx. Результат його роботи протягом 5,5 годин на процесорі Intel, тактова частота 2,1 Ггц, зображений у вигляді топології мікросхеми xc3s1200e двома компонентами на рис. 5.6: а – загальний план мікросхеми після виконання процедур Place and Rout; б – збільшений фрагмент кристала, а саме – лівого нижнього кута.



а



б

Рисунок 5.6 – Розміщення пристрою в мікросхемі xc3s1200e:
а – загальний план; б – збільшений фрагмент (лівий ніжній кут) кристала

5.4 Висновки до розділу 5

Наукова новизна. Запропоновано кубітні структури даних і квантова модель обчислювальних процесів на основі використання діаграми Хасе, що дає можливість істотно (x10-x100) підвищити швидкодію при вирішенні задач дискретної оптимізації. Запропоновано квазіоптимальний метод розв'язання задачі покриття для таблиць великої розмірності, який відрізняється від аналогів фазою попереднього розбиття матриці на сегменти або вікна, що дає можливість істотного зменшення часу пошуку покриття шляхом застосування паралельних регістрових логічних операцій над рядками таблиці. Описано апаратно орієнтовані моделі паралельного (за один цикл) обчислення булеана (множини всіх підмножин) на універсумі з n примітивів для вирішення задач покриття, мінімізації булевих функцій, стиснення даних, синтезу та аналізу цифрових систем за рахунок реалізації процесорної структури у формі діаграми Хасе.

Практична значимість. Апаратна імплементація Хасе мультипроцесора в апаратуру кристала Xilinx серії Spartan-3E: xc3s1200e дозволяє істотно зменшити час при вирішенні завдань оптимізації шляхом паралельного обчислення векторних логічних операцій за рахунок збільшення числа процесорних елементів і пам'яті для зберігання проміжних даних. Створено автоматичний генератор коду в середовищі Python, який дає можливість істотно (x3-x5) зменшити час проектування коду на мові Verilog при створенні масштабованих моделей Хасе процесора для їх подальшої імплементації в FPGA-кристали мікросхем компанії Xilinx.

6 ІМПЛЕМЕНТАЦІЯ КУБІТНИХ СТРУКТУР ДАНИХ В ПАРАЛЕЛЬНИЙ ОБЧИСЛЮВАЧ

Подається практична реалізація генератора HDL-коду «квантових» процесорів, що використовують діаграми Хасе для паралельних векторно-логічних (теоретико-множинних) обчислень булеанів, що застосовуються для прискорення вирішення задач, моделювання, верифікації, діагностування. Програмно-апаратна реалізація процесора заснована на використанні мов програмування: C ++, Verilog, Python 2.7 і платформ: Microsoft Windows, X Window (в Unix і Linux) і Macintosh OS X. Генератор HDL-коду дає можливість автоматично синтезувати HDL-коди процесорної структури від 1 до 16 двійкових розрядів для паралельної обробки відповідної кількості вхідних векторів або слів. Верифікація HDL-коду процесора виконана на тестових прикладах задачі покриття, що використовують дві стратегії оптимізації: реверсивний алгоритм для усунення надмірності і розбиття матриці покриттів на частини з метою їх подальшої паралельної обробки процесорами Хасе.

6.1 Опис Verilog-моделі пристрою

Інтерфейс. Модель процесора реалізована за допомогою одного з основних мов розробки ASIC-проектів - мови опису апаратури Verilog.

Інтерфейс пристрою в двох версіях зображено на рис. 6.1. Крім входів синхронізації (clk) і скидання (rst), він містить вхідні вектори даних i_i: m – кількість входів даних, n – розрядність векторів. Вихід out – результат об'єднання всіх векторів вхідних даних. Його розрядність дорівнює розрядності вхідних векторів і дорівнює n. Виходи o_i містять значення ознак відповідного рівня, біт вектора дорівнює 1, якщо вектор у відповідній точці містить всі одиниці, і 0 – в іншому випадку; і – позначає рівень обробки

векторів. Наприклад 8-входовий процесор з 8 розрядними входами даних матиме наступні виходи:

out – 8-бітовий вихід;

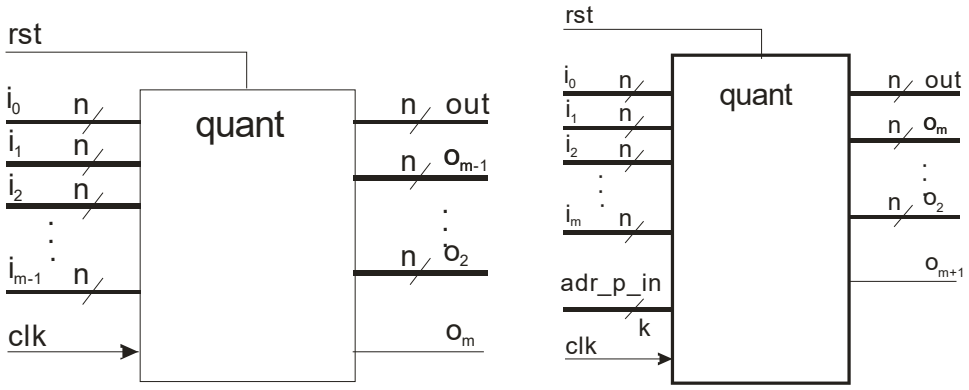
o7 – 8-бітовий вихід ознаки 7-рівня;

o2, o6 – виходи ознак 2-го і 6-го рівнів;

o3, o5 – виходи ознак 3-го і 5-го рівнів;

o4 – виходи ознак 4-го рівня;

o8 – вихід ознаки 8-го рівня.



а б

Рисунок 6.1 – Інтерфейс квантового процесора

Розрядність виходів o_i залежить від числа входів i_m і обчислюється за формулою біноміальних коефіцієнтів

$$\binom{m}{k} = C_m^k = \frac{m!}{k!(m-k)!}, \tag{6.1}$$

де m – кількість входів даних, а де k рівень їх обробки в процесорі. Для m=8…16, параметри виходів наведені в табл. 6.1.

Таблиця 6.1 – Параметри виходів процесора

Уровень	Выход	Разрядность				
		m=8	m=10	m=12	m=14	m=16
2	o2	28	45	66	91	120
3	o3	56	120	220	364	560
4	o4	70	210	495	1001	1820
5	o5	56	252	792	2002	4368
6	o6	28	210	924	3003	8008
7	o7	8	120	792	3432	11440
8	o8	1	45	495	3003	12870
9	o9		10	220	2002	11440
10	o10		1	66	1001	8008
11	o11			12	364	4368
12	o12			1	91	1820
13	o13				14	560
14	o14				1	120
15	o15					16

Як видно з табл. 6.1 кількість виходів значно збільшується з ростом числа вхідних векторів, тому в цьому випадку реалізується інтерфейс (рис. 6.1, б) з мультиплексуванням виходів o_i. У ньому вхід adr_p_in використовується як адреса для вибору необхідного виходу ознак або виводиться тільки частина регістрів ознак.

Інтерфейс пристрою на мові Verilog для 16-входового процесора представлений в лістингу 6.1. За замовчуванням для входів і виходів використовується тип даних wire.

Лістинг 6.1 – Інтерфейс пристрою на мові Verilog для 16-входового процесора

```
// Вариант 1
module device
  (input clk, rst,
   input  [7:0] i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15,
   output [7:0] out, o15,
   output o16);
// Вариант 2
module device
  (input clk, rst,
   input  [7:0] i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15,
   input  [24:0] adr_p,
   output [7:0] out, p_o16,
   output o2, o3, o4, o5, o6, o7, o8, o9, o10, o11, o12, o13, o14, o15, o16,
   output [7:0] o_pr);
```

При збільшенні числа вхідних векторів розмір коду істотно збільшується, що потребує більше часу на моделювання і синтез схеми. Продуктивність систем, що працюють з моделями цифрових пристроїв, істотно підвищується, якщо модель має ієрархічну структуру, тому вигідніше виділити елементарну комірку процесора у вигляді окремого Verilog-модуля (лістинги 6.2, 6.3) і на основі його будувати схему процесора. Вихід регістра – вектора перетинів подано сигналом O, сигнал p_in відповідає регістру ознаки для елементарної комірки, вхід en_p дозволяє відключати або підключати вихід ознаки p. Параметр BITS визначає розрядність вхідних векторів.

Лістинг 6.2 – Елементарна комірка

```
module element
  #(parameter BITS = 8)
   (input rst, clk,
    input [BITS-1:0] I0, I1,
    input en_p,
    output reg [BITS-1:0] O,
    output p);
  reg  p_in;

   always @(posedge clk, posedge rst)
     if (rst)    O <= 'b0;
     else        O <= I0 | I1;

   always @(posedge clk, posedge rst)
     if (rst)     p_in <= 1'b0;
     else         p_in <= &O;

    assign p = (en_p)? p_in: 1'bZ;

endmodule
```

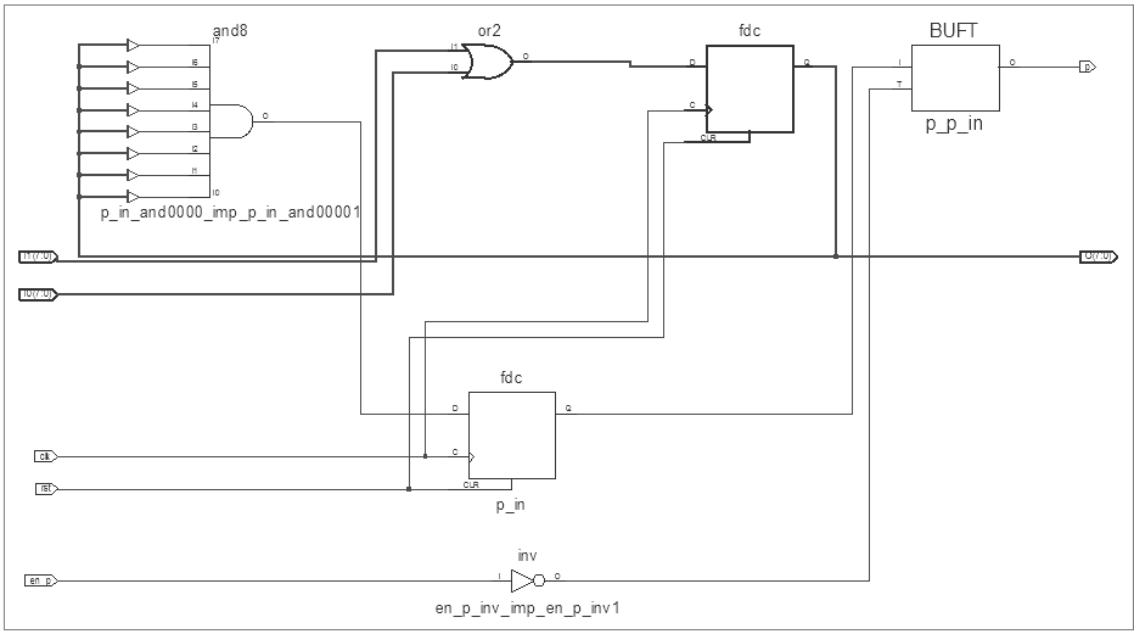


Рисунок 6.2 – Синтез схеми елементарної комірки в Xilinx ISE Design Suite 11.1

Лістинг 6.3 – Елементарна комірка

Фрагмент отчета синтеза элементарной ячейки процессора.

```
=========================================================================
* HDL Synthesis *
=========================================================================
Performing bidirectional port resolution...
Synthesizing Unit <element>.
Related source file is "component.v".
Found 8-bit register for signal <O>.
Found 1-bit tristate buffer for signal <p>.
Found 1-bit register for signal <p_in>.
Summary:
inferred 9 D-type flip-flop(s).
inferred 1 Tristate(s).
Unit <element> synthesized.
=========================================================================
HDL Synthesis Report
Macro Statistics
# Registers : 2
1-bit register : 1
8-bit register : 1
# Tristates : 1
1-bit tristate buffer : 1
```

При використанні елементарної комірки процесора, її підключення виконується так, як показано в лістингу 6.4. Кожна комірка з'єднується з декодером адреси adr_p_in, що визначає чи буде дана ознака в поточний момент подана на вихід o_pr. Декодер заснований на тристабільному буфері.

У цьому фрагменті осередки належать другому рівню процесора і обробляють інформацію з входів i_0 і i_1-i_6. Осередку comp_0_1 відповідає адреса 'd16, це означає, що якщо на вхід adr_p_in подати значення 16, то значення ознаки надійде на зовнішній вихід.

Листинг 6.4 – Фрагмент структурної моделі процесора

```
// COMPONENT LEVEL: 2
assign o_pr = (adr_p_in == 'd16) ? p_0_1 : 'bZ;
element #(8) comp_0_1 (rst, clk, r_0, r_1, r_0_1, p_0_1);

assign o_pr = (adr_p_in == 'd32) ? p_0_2 : 'bZ;
element #(8) comp_0_2 (rst, clk, r_0, r_2, r_0_2, p_0_2);

assign o_pr = (adr_p_in == 'd48) ? p_0_3 : 'bZ;
element #(8) comp_0_3 (rst, clk, r_0, r_3, r_0_3, p_0_3);

assign o_pr = (adr_p_in == 'd64) ? p_0_4 : 'bZ;
element #(8) comp_0_4 (rst, clk, r_0, r_4, r_0_4, p_0_4);

assign o_pr = (adr_p_in == 'd80) ? p_0_5 : 'bZ;
element #(8) comp_0_5 (rst, clk, r_0, r_5, r_0_5, p_0_5);

assign o_pr = (adr_p_in == 'd96) ? p_0_6 : 'bZ;
element #(8) comp_0_6 (rst, clk, r_0, r_6, r_0_6, p_0_6);
```

Присвоювана осередку адреса будується на основі її індексу. Під кожен індекс відводиться необхідна для його зображення кількість розрядів. Наприклад, для 16 вхідних змінних, на кожен індекс записується 4 біти, індекси кодуються справа наліво. Так, для комірки з індексом 0_1 адреса в двійковому вигляді буде виглядати як 0001_0000, що відповідає 16 в десятковій формі.

6.2 Генератор

Для генерації коду моделей пристроїв різної конфігурації розроблено програмний генератор. Створення моделей з великим числом вхідних векторів можливо тільки за допомогою генератора.

Вихідними даними є: кількість вхідних векторів и їх розрядність, на їх основі віконується генерація Verilog-коду. Програма написана з використанням мови Python 2.7. Розроблено дві версії генератора, які реалізують поведінкову і структурну моделі процесора.

6.2.1. *Генератор поведінкової моделі процесора.* Структура поведінкової Verilog-моделі містить: декларацію імені модуля, декларативні частини для оголошення вхідних і вихідних ліній, змінних, відповідних регістрів кожного рівня; код, що описує регістри і значення ознак; завершальну частину, модуль, що закриває.

Для генерування коду розроблений основний клас Generator, а також допоміжний клас genIndex, що генерує індекси внутрішніх регістрів моделі. Структура класів, описана за допомогою інструменту ArgoUML, представлена на рис. 6.3.

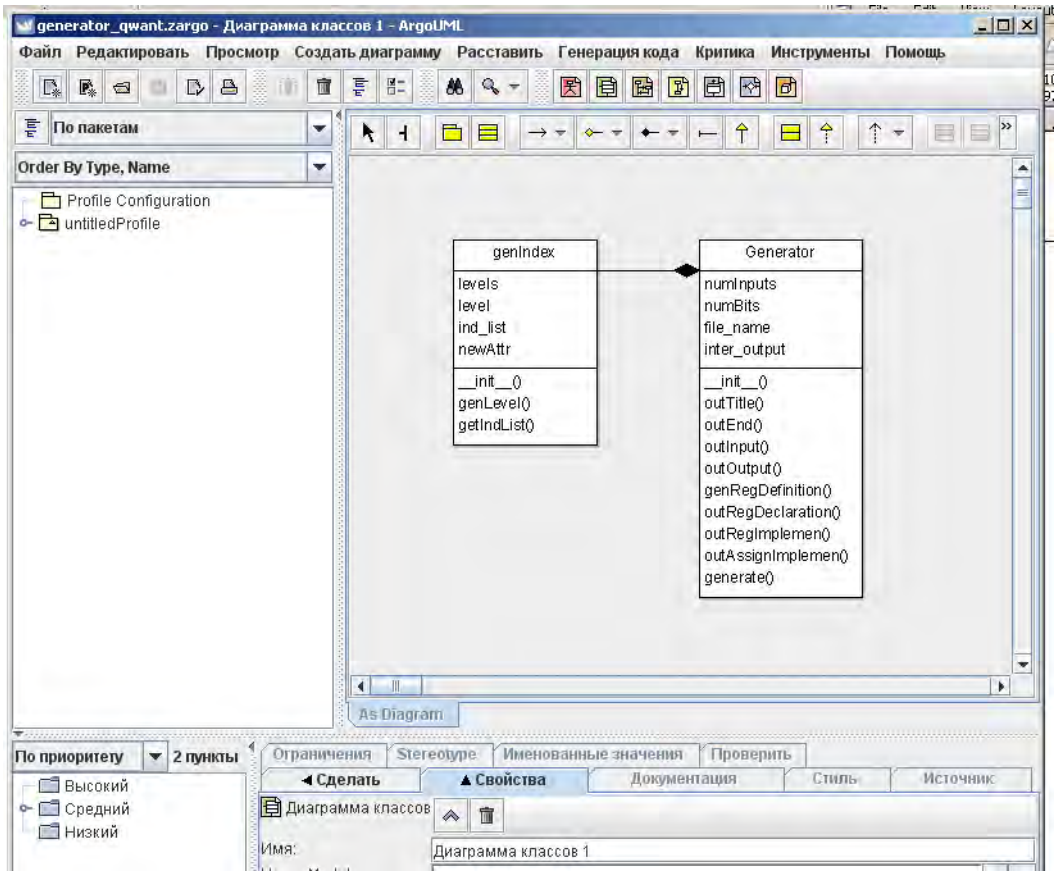


Рисунок 6.3 – Структура класів програми (ArgoUML)

Змінні класу genIndex: levels – кількість рівнів моделі (відповідає числу входів), level - поточний рівень для генерування індексів регістрів, ind_lisc – список строкових значень індексів. Два методи класу genLevel () і getIndList () генерують і повертають список індексів. Об'єкти класу genIndex використовуються в методах створення опису та реалізації регістрів класу Generator.

Для реалізації кожної частини моделі створено окремий метод класу Generator, які описані в табл. 6.2. Клас Generator містить поля: numInputs – число входів моделі, numBits – розрядність вхідних/вихідних векторів, file_name – ім'я файлу моделі, inter_output – дозволяє при необхідності відключати генерування виходів ознак для регістрів внутрішніх рівнів. Метод ініціалізації змінних задає значення змінних за замовчуванням (лістинг 6.5). numInputs = 4, numBits = 8, file_name = "model.v", inter_output = True. Будь-який з параметрів може бути змінений при створенні екземпляра класу Generator, наприклад, таким чином:

```
numInputs = 16
numBits = 8
obj1 = Generator(numInputs, numBits, "model"+str(numInputs)+".v", False).
```

Лістинг 6.5 – Фрагмент класу Generator з функцією ініціалізації полів

```
class Generator:
   """ Model code generator """
   def __init__(self, numInputs = 4, numBits = 8, file_name = "model.v",
                inter_output = True):
      self.numInputs = numInputs
      self.numBits = numBits
      self.file_name = file_name
      self.inter_output = inter_output
      if self.numInputs <= 4 :
         self.coef = 2
      elif self.numInputs <= 8 :
         self.coef = 3
      elif self.numInputs <= 16 :
         self.coef = 4
```

Клас Generator містить також функції outTitle() та outEnd(), які відкривають та закривають зовнішній файл (лістинг 6.6).

Лістинг 6.6 – Функціїї відкриття/закриття зовнішнього файлу

```
def outTitle(self):
    self.f = open(self.file_name, 'w')
    self.f.write("module device\n  (input clk, rst,\n")
def outEnd(self):
self.f.write("\nendmodule")
self.f.close()
```

Таблиця 6.2 – Методи класу Generator поведінкової моделі

Метод	Описание
outTitle()	Генерация заголовока
outInput()	Генерация деклараций входов
outOutput()	Генерация деклараций выходов
outRegDeclaration()	Генерация деклараций переменных для описания регистров
outRegImplemen()	Генерация операторов, реализующих регистры
outAssignImplemen()	Геренация операторов assign для формирования признаков уровней и присвоения значений выходов
outEnd()	Генерация заверашей части кода модели.Завершение
genRegDefinition(name1, name2, vec = True)	Вызывается из outRegDeclaration()
generate()	Является основным методом класса, который используя остальные методы класса выполняет генерацию целой модели устройства и запись ее в файл.

Рис. 6.4 подає повну структуру класів програми генератора поведінкової моделі процесора, що включає реалізацію двох класів Generator і genIndex, а також допоміжні функції для підготовки і формування даних, роботи зі списками індексів регістрів кожного рівня і ознак повноти покриття.

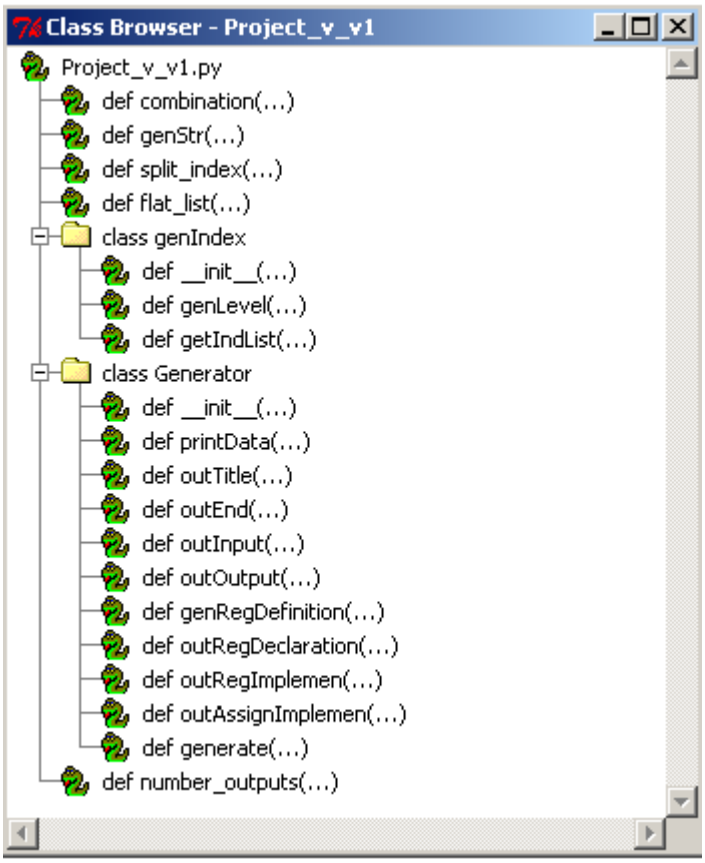


Рисунок 6.4 – Браузер класів з програми Python 2.7 Shell

Генерація індексів регістрів і змінних є досить складним завданням і для цієї мети створений окремий клас genIndex (лістинг 6.7). Для генерації індексів використовуються списки, що належать мові Python. Клас genIndex містить поля: levels - число рівнів моделі, яка відповідає кількості векторів вхідних даних; level - рівень, для якого генеруються індекси, ind_list – список індексів. Метод genLevel () генерує список індексів, а метод getIndList () – повертає список індексів.

У табл. 6.3 показано, як збільшується число зовнішніх ліній пристрою зі збільшенням числа входів (numInputs). При цьому видно, що розрядність вхідних сигналів (numBits) несуттєво впливає на загальне число входів і виходів моделі.

Лістинг 6.7 – Клас генератора індексів змінних

```
class genIndex:
    """ generate list of index for the level """
    def __init__(self, levels = 16, level = 1):
        self.levels = levels
        self. levels = level
        self.ind_list = []
        self.genLevel()

    def genLevel(self):
        for i in xrange(self.levels):
            self.ind_list.append([str(i)])
        if self.level > 1:
            list_t = []
            ii=0
            for i1 in range(self.levels-1):
                ii += 1
                t=[]
                for i2 in xrange(ii, self.levels):
                    t.append(str(i1)+"_"+str(i2))
                #print t
                list_t.append(t)
            #print "list_level 1",list_t
            self.ind_list = list_t
        if self.level > 2:
            iLevel = 3
            while iLevel <= self.level:
                list_t2 = []
                list_t.pop(0)
                i=0
                while (len(list_t) > 0):
                    t = []
                    for i1 in xrange(len(list_t)):
                        list_i1 = list_t[i1]
                        for i2 in xrange(len(list_i1)):
                            s = str(i) + "_" + list_i1[i2]
                            t.append(s)
                    i += 1
                    list_t.pop(0)
                    list_t2.append(t)
                #print "list_level ",iLevel, ":", list_t2
                list_t = list_t2
                iLevel += 1

            self.ind_list = list_t2

    def getIndList(self):
        return self.ind_list
```

Таблиця 6.3 – Залежність кількості зовнішніх ліній моделі від кількості входів (numInputs) та їх розрядності (numBits)

numInputs	numBits							
	4	8	12	16	20	24	28	32
4	31	55	79	103	127	151	175	199
6	83	115	147	179	211	243	275	307
8	279	319	359	399	439	479	519	559
10	1 051	1 099	1 147	1 195	1 243	1 291	1 339	1 387
12	4 127	4 183	4 239	4 295	4 351	4 407	4 463	4 519
14	16 419	16 483	16 547	16 611	16 675	16 739	16 803	16 867
16	65 575	65 647	65 719	65 791	65 863	65 935	66 007	66 079
18	262 187	262 267	262 347	262 427	262 507	262 587	262 667	262 747

Зі збільшенням числа вхідних векторів, значно збільшується розмір моделі, а, отже, і кількість коду для її реалізації, в результаті обробка таких моделей системами синтезу і імлементації займає дуже багато часу. Подібні програми краще працюють з кодом, що складається з декількох модулів. З метою підвищення швидкості виконання операцій синтезу та імплементації розроблена структурна модель процесора, в якому окремий вузол процесора реалізований у вигляді окремого модуля - елементарної комірки. Крім того, в цю модель доданий вхід адреси, що дозволяє вибірково зчитувати інформацію з вузлів-ознак, що дозволило істотно скоротити число зовнішніх виходів пристрою.

6.2.2 *Генератор структурної моделі процесора*. Структурна Verilog-модель складається з областей, які зображені на рис. 6.5. Замість області реалізації регістрів і присвоєння значень виходів, як було в поведінковій моделі, ця модель містить опис реалізації регістрів 1-го рівня за допомогою конструкції always та реалізацію регістрів інших рівнів процесора, включаючи оператор реалізації копії модуля елементарної комірки, оператор assign, що описує тристабільний буфер для підключення виходу ознаки і умовний оператор реалізації мультиплексора вибору комірки. Програма генератора ускладнюється за рахунок додавання декількох методів (структура програми з відлагоджувального засобу для Python 2.7 подана на рис. 6.6. Опис методів класу генератора описані в табл. 6.4. Програма містить 380 рядків коду.

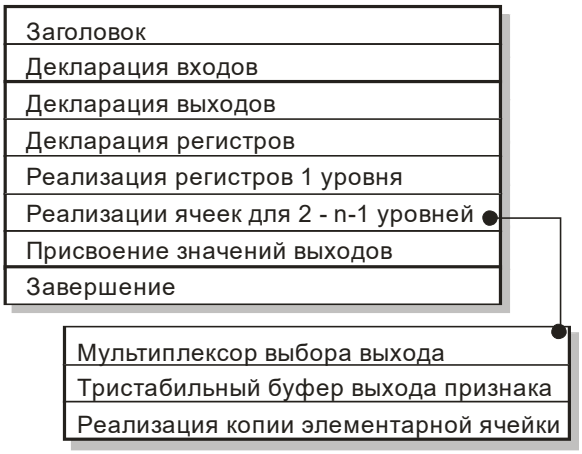


Рисунок 6.5 – Структурна Verilog-модель

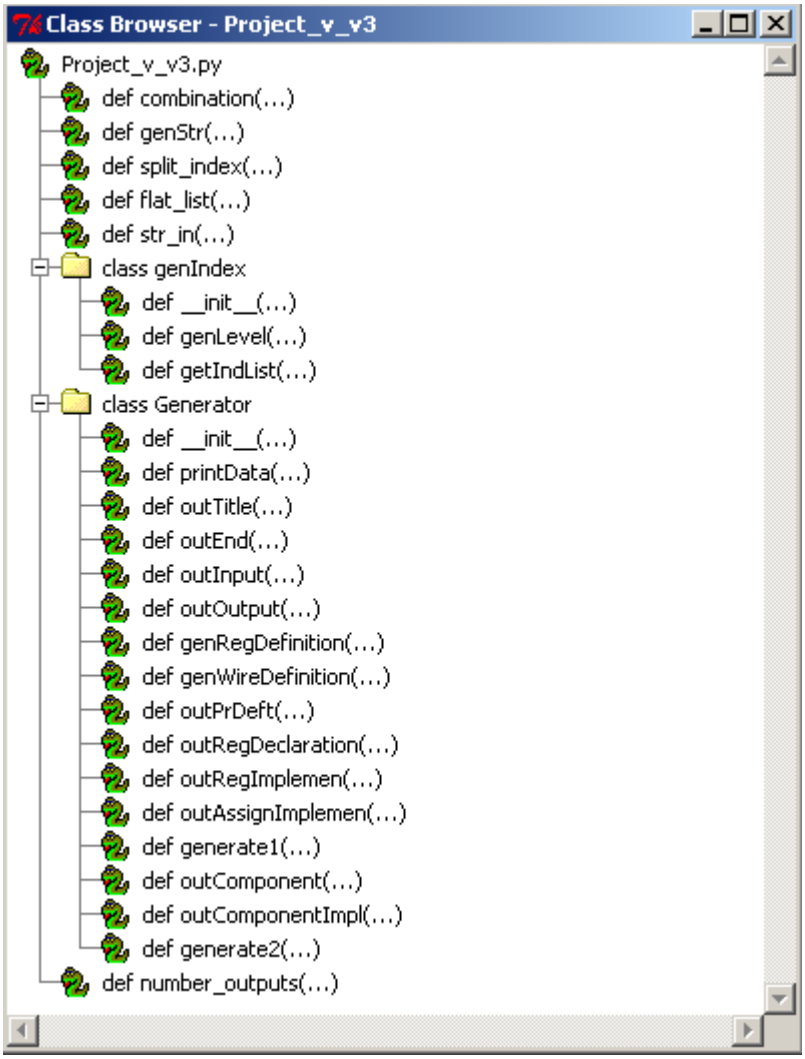


Рисунок 6.6 – Структура програми генератора в браузері класів з програми Python 2.7 Shell

Таблиця 6.4 – Методи класу Generator поведінкової моделі

Метод	Описание
outTitle()	Генерация заголовока
outInput()	Генерация деклараций входов
outOutput()	Генерация деклараций выходов
outRegDeclaration()	Генерация деклараций переменных для описания регистров
outRegImplemen()	Генерация операторов, реализующих регистры
outAssignImplemen()	Геренация операторов assign для формирования признаков уровней и присвоения значений выходов
outEnd()	Генерация заверашей части кода модели.Завершение
genRegDefinition(name1, name2, vec = True	Декларация регистровых переменных, вызывается из функции outRegDeclaration()
genWireDefinition()	Генерация переменных типа данных wire класса цепи, моделирующих тристабильны е буферы
OutComponent()	Генерирует файл с моделью элементарной ячейки процессора
OutComponentImpl()	Генерирует оператор реализации копии модуля элементарной ячекки процессора
printData()	Вывод числа входов процессора и их разрядности
generate2()	Является основным методом класса, который используя остальные методы класса выполняет генерацию целой модели устройства и запись ее в файл.

Присвоювана кожному осередку адреса генерується на основі рядкового подання її індексу за допомогою функції str_in (лістинг 6.8).

Лістинг 6.8 – Функція генерації адреси комірки

```
def str_in(l0, step):
    """ Form integer address value based on element index"""
    i = 0
    l = l0.split('_')
    l.reverse()
    for j in xrange(len(l)):
        i *= 2**step
        i += int(l[j])
        #print 'i', i
    return i
```

Незважаючи на те, що python – це інтерпретативна мова програмування, для програм, створених за допомогою цієї мови, в разі необхідності можна легко отримати виконуваний файл, використовуючи засоби PyInstaller (http://www.pyinstaller.org/) або py2exe (http: //www.py2exe.org/) або подібні до них. Конвертор PyInstaller створює незалежні виконувані програми для операційних систем Windows, Linux, Mac OS X, Solaris і AIX, в той час як py2exe реалізує тільки Windows додатки.

6.2.3 *Графічний інтерфейс користувача генератора.* Для оформлення графічного інтерфейсу користувача була використана відкрита бібліотека tkinter мови python, яка є де-факто стандартом для вирішення подібних задач. Доступність, переносимість, простота отримання, документованість і наявність розширень визначають популярність tkinter для формування GUI в Python-додатках протягом багатьох років. На відміну від більш складних систем, бібліотека tkinter дозволяє відразу ж приступити до роботи з нею, без необхідності попередньо освоювати великі моделі взаємодії класів. Незважаючи на простоту прикладного інтерфейсу бібліотеки tkinter, вона дає можливість додавати нові віджети, написані на мові Python, або підключати додаткові розширення, такі як Pmw, Tix і ttk. Важливою перевагою є переносимість. Сценарій на мові Python, в якому графічний інтерфейс будується за допомогою бібліотеки tkinter, працюватиме без змін на всіх основних сучасних віконних платформах: Microsoft Windows, X Window (в Unix і Linux) і Macintosh OS X. Більш того, зовнішній вигляд створеного інтерфейсу буде звичний для користувачів кожної з цих платформ. Ця особливість розвивалася за мірою того, як бібліотека Tk (на якій заснована tkinter) ставала все більш зрілою. Графічний інтерфейс, реалізований сценарієм Python / tkinter, в Windows виглядає так, як повинен виглядати інтерфейс програми для Windows; в Unix і Linux, забезпечує таке ж взаємодію і на Mac він виглядає так, як повинна виглядати програма Mac.

Бібліотека tkinter є модулем стандартної бібліотеки Python, що поставляється разом з інтерпретатором. Більш того, ув більшість пакетів

установки Python (включаючи стандартний пакет установки Python для Windows, Mac і більшість дистрибутивів Linux) вже включена підтримка tkinter. Завдяки цьому сценарію, написані з використанням модуля tkinter, відразу можуть працювати з більшістю інтерпретаторів Python, не вимагаючи додаткових дій по встановленню. Оскільки задіяна в ній бібліотека Tk використовується також мовами програмування Tcl і Perl (і багатьма іншими), їй приділяється більше уваги і зусиль розробників, ніж іншим наявним інструментарієм.

Вікно графічного інтерфейсу генератора зображено на рис. 6.7. У ньому перед генерацією Verilog-коду можна задавати кількість входів Number of inputs (число вхідних векторів), розрядність вхідних векторів Number of bits, ім'я файлу і шлях для збереження створюваної моделі File name. За замовчуванням число входів і їх розрядність дорівнює 4; модель, що генерується, зберігається в файл model.v. Кнопка Reset скидає введені значення параметрів до заданих за замовчуванням, Generate - генерує Verilog-модель процесора, Cancel – закриває вікно генератора. У програмі використовувалися екземпляри класів мітки Label, поля введення Entry, кнопки Button, PhotoImage для виведення зображення. Всі класи належать бібліотеці Tkiter, яка в свою чергу заснована на бібліотеці Tk ():

```
import sys
from Tkinter import *
root = Tk()
root.title("     Hasse processor model generator")
```

Для упорядкування графічного відображення об'єктів класів у вікні генератора використовується метод сітки – grid. Наступний фрагмент коду являє створення мітки і поля для введення значення параметра Number of inputs та розміщення його в нульовому рядку і колонці сітки.

```
l1 = Label(root, text = 'Number of inputs')
l1.grid(row = 0, column = 0, padx = 20, pady = 20, sticky=W)
num_levels = IntVar(value = 4)
i1 = Entry(root, justify = "center", width = 20, textvariable = num_levels)
i1.config(bd = 3, relief = SUNKEN)
i1.grid(row = 0, column = 1)
```

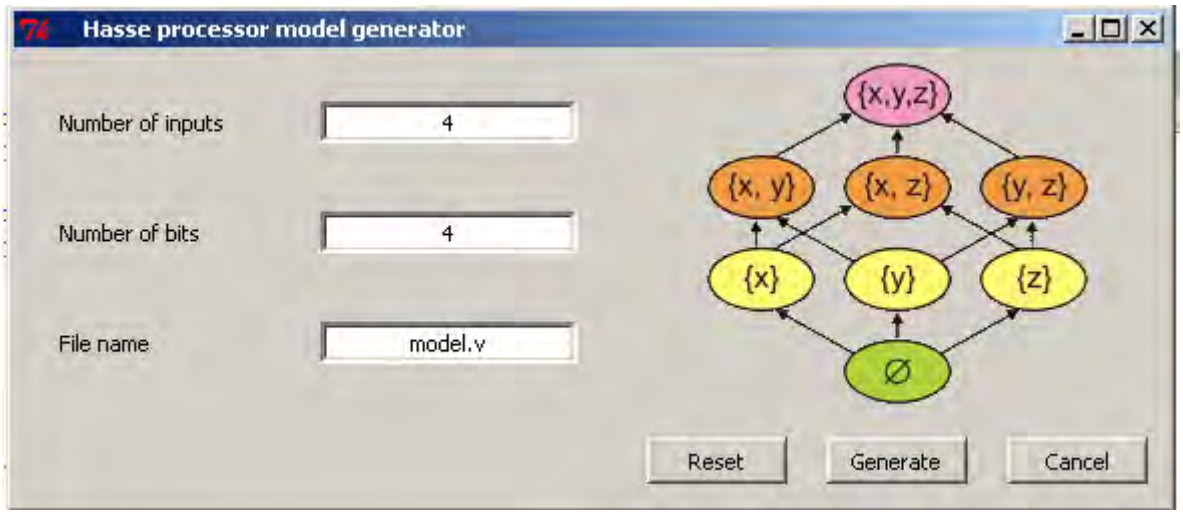


Рисунок 6.7 – Вікно графічного інтерфейсу для Windows

6.3 Синтез та імплементація

Реалізація пристрою на мікросхемах FPGA фірми Xilinx Spartan 3E (xc3s1600e-4-fg320) і Virtex 5 xc5vlx50. Для синтезу та імплементації використовувався програмний пакет фірми Xilinx ISE Design Suite 11.1. Модель пристрою, з конфігурацією 10 входів, розрядність входів 8 біт, реалізована з використанням мікросхеми фірми Xilinx серії Spartan-3E: xc3s1600e. При цьому отримана схема містить 9247 тригера FF і 18062 4-х входові таблиці перетворення LUT, що становить 31% і 61% доступних ресурсів мікросхеми.

Logic Utilization:

Number of Slice Flip Flops: 9,247 out of 29,504 31%

Number of 4 input LUTs: 18,062 out of 29,504 61%.

Виходячи з результатів статичного часового аналізу (табл. 6.5), мінімальний робочий період синхросигналу дорівнює 41 ns, що відповідає частоті 24 МГц.

Таблиця 6.5 – Часові параметри пристрою

Clock clk to Pad

Destination	clk (edge) to PAD	Internal Clock(s)	Clock Phase
o2	13.343(R)	clk_BUFGP	0.000
o3	15.381(R)	clk_BUFGP	0.000
o4	16.611(R)	clk_BUFGP	0.000
o5	19.541(R)	clk_BUFGP	0.000
o6	20.813(R)	clk_BUFGP	0.000
o7	15.613(R)	clk_BUFGP	0.000
o8	12.385(R)	clk_BUFGP	0.000
o9	13.186(R)	clk_BUFGP	0.000
o_pr<0>	40.912(R)	clk_BUFGP	0.000
o_pr<1>	37.572(R)	clk_BUFGP	0.000
o_pr<2>	37.575(R)	clk_BUFGP	0.000
o_pr<3>	37.859(R)	clk_BUFGP	0.000
o_pr<4>	38.169(R)	clk_BUFGP	0.000
o_pr<5>	37.284(R)	clk_BUFGP	0.000
o_pr<6>	38.165(R)	clk_BUFGP	0.000
o_pr<7>	37.858(R)	clk_BUFGP	0.000
out<0>	7.854(R)	clk_BUFGP	0.000
out<1>	8.086(R)	clk_BUFGP	0.000
out<2>	8.088(R)	clk_BUFGP	0.000
out<3>	8.057(R)	clk_BUFGP	0.000
out<4>	7.534(R)	clk_BUFGP	0.000
out<5>	7.465(R)	clk_BUFGP	0.000
out<6>	8.073(R)	clk_BUFGP	0.000
out<7>	8.032(R)	clk_BUFGP	0.000
p_o10<0>	7.716(R)	clk_BUFGP	0.000

Clock to Setup on destination clock clk

Source Clock	Src:Rise|Dest:Rise	Src:Fall|Dest:Rise	Src:Rise|Dest:Fall	Src:Fall|Dest:Fall
clk	4.862			

Для обчислення часових параметрів використовуються результати статичного аналізу, виконувані програмним забезпеченням на етапі Place & Route. Система обчислює затримки шляхів між регістрами (Tclk_to_clk) і від синхровходу до виходу (Tclk_to_pad). Максимальний з цих параметрів визначає мінімальний робочий період синхросигналу, який може бути поданий на вхід пристрою:

Period = max{Tclk_to_clk , Tclk_to_pad_max };

Наприклад, для часових параметрів:

Tclk_to_clk = 4.862 ns
Tclk_to_pad_max = 40.912ns

Мінімальний робочий період синхросигналу буде дорівнювати Period = 40.912 ns, що відповідає частоті Fclk = 24 МГц.

6.4 Формування мінімального покриття двовимірної матриці

Рішення практичної задачі формування двійкового покриття одиничної матриці полягає в пошуку мінімальної комбінації рядків і стовпців, що покривають всі одиничні значення. Пошук покриття невеликої матриці реалізується за допомогою простих методів. Пошук бінарного покриття великої матриці, що містить тисячі рядків і стовпців, вимагає значних витрат часу і засобів.

У даному підрозділі пропонується формувати бінарне покриття двійковій матриці шляхом її попереднього горизонтально-вертикального розбиття на множину підматриць.

Горизонтальне розбиття матриць на подматріци полягає в розподілі матриці M на множину підматриць із заданою кількістю рядків у відповідності з виразом:

$$M(R,S^R) = \{M'_i, M'_{i+1}, \dots M'_{i+n}\}, n = R/S^R$$

де M – оригінальна матриця; R– кількість рядків матриці M; S^R – кількість рядків в підматриці; M'_i – похідна матриця, отримана внаслідок розподілу заданої матриці; i -индекс матриці. Наприклад, матриця M задана як:

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	1	1	1
2	1	0	1	0	0	0	0	1	1	1
3	1	0	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	1	0	0	1	0	0
6	1	0	0	1	1	1	1	1	1	0
7	0	0	0	0	0	1	1	1	1	0
8	1	1	1	1	1	0	0	0	0	0
9	1	1	1	1	0	0	0	1	1	1
10	0	0	0	0	0	1	1	1	1	1

Горизонтальне розбиття матриці на підматриці при значенні спліт-параметру $S^R = 5$ дає результат:

Матриця $M_1^{'}$

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	1	1	1
2	1	0	1	0	0	0	0	1	1	1
3	1	0	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	1	0	0	1	0	0

Матриця $M_2^{'}$

	1	2	3	4	5	6	7	8	9	10
6	1	0	0	1	1	1	1	1	1	0
7	0	0	0	0	0	1	1	1	1	0
8	1	1	1	1	1	0	0	0	0	0
9	1	1	1	1	0	0	0	1	1	1
10	0	0	0	0	0	1	1	1	1	1

$$M = \{M_1^{'}, M_2^{'}\}$$

Аналогічно можна виконати вертикальне розбиття матриці на подматріци із заданою кількістю стовпців.

При вирішенні задачі формування бінарного покриття найбільш ефективним є вертикально-горизонтальне розбиття матриці M на множину підматриць із заданою кількістю рядків і стовпців у відповідності з виразом:

$$M(R, C, S^R, S^C) = \{M_{i,j}^{'}, M_{i,j+1}^{'}, M_{i,j+2}^{'}, M_{i+1,j}^{'}, \dots M_{i+n,j+m}^{'}\}, j = C/C^R,$$

де C – кількість стовпців матриці M; C^R – кількість стовпців підматриці.

Приклад горизонтально-вертикального розбиття матриці M при значенні спліт-параметрів $S^R = 5$ и $S^C = 5$ наведено нижче:

Матриця $M'_{1,1}$

	1	2	3	4	5
1	0	0	0	0	0
2	1	0	1	0	0
3	1	0	1	0	0
4	1	1	1	1	1
5	0	0	0	0	1

Матриця $M'_{2,2}$

	6	7	8	9	10
1	0	1	1	1	1
2	0	0	1	1	1
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	1	0	0

Матриця $M'_{2,1}$

	1	2	3	4	5
6	1	0	0	1	1
7	0	0	0	0	0
8	1	1	1	1	1
9	1	1	1	1	0
10	0	0	0	0	0

Матриця $M'_{1,2}$

	6	7	8	9	10
6	1	1	1	1	0
7	1	1	1	1	0
8	0	0	0	0	0
9	0	0	1	1	1
10	1	1	1	1	1

Розбиття матриць на подматріци дозволяє істотно зменшити час пошуку бінарного покриття за рахунок паралельної обробки підматриць.

Для програмної реалізації задачі побудови бінарного покриття вибрана мова програмування Python, який дозволяє описувати двовимірний масив (матрицю) як вкладений об'єкт (масив масивів) за допомогою класу List (список). Приклад матриці:

	6	7	8	9	10
6	1	1	1	1	0
7	1	1	1	1	0
8	0	0	0	0	0
9	0	0	1	1	1
10	1	1	1	1	1

Спрощений опис матриці на мові Pyton (лістинг 6.9):

Лістинг 6.9 – Спрощений опис матриці

```
matrix = [[1, 1, 1, 1, 0],
          [1, 1, 1, 1, 0],
          [0, 0, 0, 0, 0],
          [0, 0, 1, 1, 1],
          [1, 1, 1, 1, 1]]
```

Фактично матриця складається зі списку лінійних об’єктів (рядків), як показано на лістингу 6.10.

Лістинг 6.10 – Список лінійних об’єктів

```
print line

index      :0
indexes    :[]
values     :[1, 1, 1, 1]
ones count :[1, 1, 1, 1]
```

Члени класу “Line” з унікальним конструктором подані в лістингу 6.11.

Лістинг 6.11 – Члени класу “Line”

```
class Line(object):
…
    def __init__(self, index = 0, indexes = [], values = [], ones_count = []):
        self.index      = index
        self.indexes    = indexes
        self.values     = values
        self.ones_count = ones_count
        if len(self.values) != len(self.ones_count):
            self.ones_count.extend(values)
…
end
```

Лістинг 6.12 містить деякі методи класу Line.

Лістинг 6.12 – Методи класу Line

```
class Line(object):
…
    def ones_qnty(self):
        qnty = 0
        for value in self.values:
            if value:
                qnty += 1
        return qnty

    def binary_or(self, line):
        before_ones_qnty = self.ones_qnty()
        for key, value in enumerate(line.values):
            self_value = self.values[key]
            self.values[key] = self_value | value
```

```
            self.ones_count[key] += value
        if self.ones_qnty() > before_ones_qnty:
            for index in line.indexes:
                self.indexes.append(index)
            self.indexes.append(line.index)
            return True
        return False

    def binary_xor(self, line):
        for key, value in enumerate(line.values):
            self.ones_count[key]    -= value
            if self.ones_count[key] <= 0:
                self.ones_count[key] = 0
                self.values[key] = 0
            else:
                self.values[key]        = 1
        self.indexes.remove(line.index)
…
end
```

Метод «ones_qnty» повертає кількість одиниць в рядку; «Binary_or» застосовує двійкове перетворення or до зазначеного рядку. Булева змінна повертає значення:

– True, якщо рядок модифікований за допомогою двійкового перетворення or;

– False, якщо рядок не модифікований.

Цілочислена змінна «ones_count» зберігає кількість одиниць, отриманих після виконання бінарної операції or.

Метод «binary_xor» виконує операцію xor з обраним рядком. Для кожного елемента значення «value» віднімається з «ones_qnty». Якщо значення «ones_qnty» дорівнює нулю, поле значення відповідного рядка також встановлюється в нуль, в іншому випадку - в одиницю.

Набір матриць подається у вигляді контейнера, який спрощує процедуру маніпулювання складними матрицями (лістинг 6.13).

Лістинг 6.13 – Контейнер

```
class ComplexMatrix(object):

    def __init__(self, matrices):
```

```
        self.matrices   = matrices
        self.columns    = len(self.matrices[0].lines)
        self.rows       = len(self.matrices) / self.columns


    def matrixAtRowColumn(self, row, column):
        return self.matrices[row * self.columns + column]
```

Програмна імплементація вертикально-горизонтального розбиття матриці довільного розміру на множину підматриць з кількістю рядків, заданим користувачем, наведена в лістингу 6.14.

Лістинг 6.14 – Програмна імплементація вертикально-горизонтального розбиття матриці

```
class MatrixUtils(object):
…
    @staticmethod
    def split_array(array, divisions):
        return [array[i:i+divisions] for i in range(0, len(array), divisions)]

    @staticmethod
    def linearSplittedMatricesFromValues(matrix, max_lines = 25):
        splited_lines = MatrixUtils.split_array(matrix.lines, max_lines)
        matrixes = []
        for lines in splited_lines:
            matrixes.append(Matrix(lines))

        return matrixes

…
end
```

Метод «split_matrix» дозволяє розподілити список на множину підсписків (лістинг 6.15):

Лістинг 6.15 – Метод «split_matrix»

```
matrix_to_split = [[0, 0], [0, 1], [1, 1], [1, 0]]
matrices = split_array(matrix_to_split, 2)
>>>Print matrices
[[[0, 0], [0, 1]], [[1, 1], [1, 0]]]
```

Потім список переупорядковується з метою отримання двох матриць (лістинг 6.16).

Лістинг 6.16 – Переупорядкування списку

```
 [[0, 0],
 [0, 1]]

[[1, 1],
 [1, 0]]
```

Програмна імплементація горизонтально-вертикального розбиття матриці на множину підматриць містить метод «ComplexMatrixFromValues», який повертає складну матрицю, отриману з множини підматриць.

Оригінальна матриця розбивається горизонтально і вертикально шляхом виклику внутрішнього методу «rectSplitedMatricesFromValues» (лістинг 6.17).

Лістинг 6.17 – Розбиття матриці горизонтально і вертикально

```
class MatrixUtils(object):
…
    @staticmethod
    def complexMatrixFromValues(values,
                                horiz_max_lines = 10,
                                vert_max_lines = 10):
        matrices = MatrixUtils.rectSplitedMatricesFromValues(values,
                                                             horiz_max_lines,
                                                              vert_max_lines)

        return ComplexMatrix(matrices)
…
end
```

Метод «RectSplitedMatricesFromValues» повертає множину матриць (лістинг 6.18).

Лістинг 5.18 – Повернення множини матриць

```
  @staticmethod
    def    rectSplitedMatricesFromValues(values,    horiz_max_lines    =    10,
vert_max_lines = 10):
        horiz_splitted_lines = []
        for hrz_line in values:
            horiz_splitted_lines.append(MatrixUtils.split_array(hrz_line,
horiz_max_lines))

        result = []
```

```
        final_results = []
        elements_count = len(horiz_splitted_lines[0])
        print 'elements count: ' + str(elements_count)

        for _ in range(elements_count):
            result.append([])

        line_count = 0
        for line in horiz_splitted_lines:
            element_count = 0
            for element in line:
                result[element_count].append(element)
                element_count += 1

            line_count += 1

            if (not (line_count % horiz_max_lines) and line_count):
                final_results.append(result)
                result = []
                for _ in range(elements_count):
                    result.append([])

        print 'result: ' + str(result)
        matrices = []
        mat_counter = 0
        for res in final_results:
            for val in res:
                mat_index = [mat_counter / horiz_max_lines, mat_counter %
vert_max_lines]
                print mat_index

matrices.append(Matrix(MatrixUtils.matrixLinesFromListValues(val), mat_index))
                mat_counter += 1

        return matrices
```

Побудова сигнатури бінарного покриття матриці являє собою оптимізацію високого рівня, зокрема для горизонтально-вертикального розбиття матриці. Ідея полягає в поданні матриці хешем, унікальним по відношенню до кількості одиничних елементів матриці і розділення. Це дозволяє уникнути повторного обчислення двійкового покриття для однієї і тієї ж матриці. Сигнатура матриці подається виразом:

$$H_{i,j} = \{\{R, C\}, \{M_{i,j}\}\}.$$

З матричною сигнатурою асоціюється значення:

$$V_{i,j} = \{\{R_i\}, \{M_{i,j}\}\}$$

Приклад:

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	1	1	1
2	1	0	1	0	0	0	0	1	1	1
3	1	0	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	1	0	0	1	0	0
6	0	0	0	0	0	0	1	1	1	1
7	1	0	1	0	0	0	0	1	1	1
8	1	0	1	0	0	0	0	0	0	0
9	1	1	1	1	1	0	0	0	0	0
10	0	0	0	0	1	0	0	1	0	0

Застосування методу з горизонтальним спліт-параметром $S^R = 5$ і вертикальним спліт-параметром $S^C = 5$ дає результат з чотирьох матриць:

Матриця $M'_{1,1}$

	1	2	3	4	5
1	0	0	0	0	0
2	1	0	1	0	0
3	1	0	1	0	0
4	1	1	1	1	1
5	0	0	0	0	1

Матриця $M'_{1,2}$

	6	7	8	9	10
1	0	1	1	1	1
2	0	0	1	1	1
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	1	0	0

Матриця $M'_{2,1}$

	1	2	3	4	5
6	0	0	0	0	0
7	1	0	1	0	0
8	1	0	1	0	0
9	1	1	1	1	1
10	0	0	0	0	1

Матриця $M'_{2,2}$

	6	7	8	9	10
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	1	0	0

Обчислення хеша для кожної матриці виконується так, як описано нижче:

$$H'_{1,1} = \{\{5,5\}, \{0000010100101001111100001\}\}$$

$$H'_{1,2} = \{\{5,5\}, \{0111100111000000000000100\}\}$$

$$H'_{2,1} = \{\{5,5\}, \{0000010100101001111100001\}\}$$

$$H'_{2,2} = \{\{5,5\}, \{0111100111000000000000100\}\}$$

Результати обчислення хеша:

$$V'_{1,1} = \{\{4\},\{11111\}\}$$
$$V'_{1,2} = \{\{1\},\{01111\}\}$$
$$V'_{2,1} = \{\{9\},\{11111\}\}$$
$$V'_{2,2} = \{\{6\},\{01111\}\}$$

Значення хеша $H'_{1,1}$, $H'_{1,2}$ відповідно дорівнюють $H'_{2,1}, H'_{2,2}$. Якщо відомо бінарне покриття матриці $M'_{1,1}$ можна визначити бінарне покриття матриці $M'_{2,1}$ шляхом обчислення її хеша.

Запропоновані методи спрямовані на розпаралелювання процесу визначення двійкового покриття. Найбільш простим для реалізації є метод горизонтального розбиття. Але при цьому не розглядаються стовпці матриці. Для великих матриць, де кількість стовпців істотно більше кількості рядків, метод горизонтального розбиття може вимагати значних витрат часу внаслідок його лінійної обчислювальної складності.

Метод горизонтально-вертикального розбиття більш складний у реалізації і вимагає більших витрат часу, ніж попередній метод, в тому числі і через необхідність відновлення оригінальної матриці з множини матриць, отриманих в результаті вертикального розбиття. Перевагою даного методу є можливість розпаралелювання процесу вирішення задачі пошуку покриття. Для збільшення швидкості процедури пошуку покриття при обробці підматриць може бути використана замовна логіка. Профайл тест для запропонованих методів представлений в лістингу 6.19, результат – у лістингу 6.20.

Лістинг 6.19 – Профайл тест для методів

```
'''
Created on 17.04.2013

@author: ______
'''
from models.matrix_utils import MatrixUtils
from test_main import test_table_4
import cProfile

def linear_split_test():
    for _ in range(100):
        MatrixUtils.linearSplittedMatricesFromValues(test_table_4, 5)
def rect_split_test():
    for _ in range(100):
        MatrixUtils.rectSplitedMatricesFromValues(test_table_4, 5, 5)
if __name__ == '__main__':
    cProfile.run('rect_split_test()')
    cProfile.run('linear_split_test()')
```

Лістинг 6.20 – Результат

```
  306104 function calls in 1.576 seconds
   Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    1.576    1.576 <string>:1(<module>)
    56000    0.281    0.000    0.399    0.000 line.py:9(__init__)
        1    0.051    0.051    1.576    1.576 main.py:17(rect_split_test)
    11200    0.029    0.000    0.029    0.000 matrix.py:11(__init__)
     8100    0.078    0.000    0.112    0.000 matrix_utils.py:15(split_array)
      100              0.300              0.003              1.525              0.015
matrix_utils.py:43(rectSplitedMatricesFromValues)
    11200              0.404              0.000              0.910              0.000
matrix_utils.py:83(matrixLinesFromListValues)
     8200    0.015    0.000    0.015    0.000 {len}
   145500    0.276    0.000    0.276    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
    56000    0.118    0.000    0.118    0.000 {method 'extend' of 'list' objects}
     9801    0.023    0.000    0.023    0.000 {range}

         188604 function calls in 1.069 seconds

   Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    1.069    1.069 <string>:1(<module>)
     8100    0.035    0.000    0.054    0.000 line.py:9(__init__)
        1    0.047    0.047    1.069    1.069 main.py:12(linear_split_test)
    81950    0.256    0.000    0.256    0.000 matrix.py:11(__init__)
      100    0.059    0.001    0.062    0.001 matrix_utils.py:15(split_array)
      100              0.430              0.004              1.022              0.010
matrix_utils.py:31(linearSplittedMatricesFromValues)
      100    0.053    0.001    0.124    0.001 matrix_utils.py:93(matrixFromListValues)
      100    0.000    0.000    0.000    0.000 {len}
    89950    0.167    0.000    0.167    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
     8100    0.019    0.000    0.019    0.000 {method 'extend' of 'list' objects}
          101    0.002    0.000    0.002    0.000 {range}
```

Метод горизонтального розбиття дозволяє розділити вихідну матрицю на 17 підматриць. Метод горизонтально-вертикального розбиття забезпечує розподіл матриці на 112 підматриць, але його швидкодія на 50% менше, ніж методу горизонтального розбиття.

Алгоритм бінарного покриття матриці описано нижче.

Вибрати перший рядок матриці, яка містить, принаймні, одну одиницю і застосувати двійкову операцію or до даного рядку і інших рядків матриці. Якщо перший рядок містить тільки нульові елементи, її необхідно пропустити і перейти до наступного рядка. Операція повторюється до тих пір, поки не буде знайдено непорожній рядок. Якщо 5 рядків підматриці містять тільки нульові елементи, то підматріця не бере участь в подальших обчисленнях. При кожному виконанні операції or необхідно перевіряти, чи змінює вона вміст рядка. Якщо «так», то зберігається результуючий рядок та індекси обох рядків.

Приклад матриці M1:

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	1	0	0	0
4	0	1	0	0	0
5	0	0	1	0	0

$l1\ empty \rightarrow l1\ skipped$

$l2\ empty \rightarrow l2\ skipped$

$l3^{\square} \cup l4 \rightarrow l3^{\square} = 01000 \rightarrow l4\ rejected$

$l3 \cup l5 \rightarrow l3^{1}\ saved$

Рядки $\{l_1,l_2\}$ пропущені, оскільки вони містять тільки нульові елементи.

Матриця M_2:

	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	0	0
3	1	1	0	0	0
4	0	0	1	1	0
5	1	1	0	0	0

$l1 \cup l2 \rightarrow l1' = 01001\ not\ changed \rightarrow l2\ rejected$

$l1' \cup l3 \rightarrow l1^1 = 11001\ changed \rightarrow l3\ saved$

$l1^1 \cup l4 \rightarrow l1^2 = 11111\ changed \rightarrow l4\ saved$

Матриця M3:

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

$l1\ empty \rightarrow l1\ skipped$

$l2\ empty \rightarrow l2\ skipped$

$l3\ empty \rightarrow l3\ skipped$

$l4\ empty \rightarrow l4\ skipped$

$l5\ empty \rightarrow l5\ skipped$

При виконанні кожної операції or необхідно переконатися в тому, що результуючий рядок не заповнений тільки одиничними елементами. У цьому випадку операція or більше не виконується.

Приклад:

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	1	1
7	1	0	0	0	1	1	0	1	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

$$l1 \cup l2 \rightarrow l1^1 = 1101000000 \rightarrow l2\ saved$$

$$l1^1 \cup l3 \rightarrow l1^2 = 1101100000 \rightarrow l3\ saved$$

$$l1^2 \cup l4 \rightarrow l1^2 = 1101100000 \rightarrow l4\ rejected$$

$$l1^2 \cup l5 \rightarrow l1^3 = 1111100000 \rightarrow l5\ saved$$

$$l1^3 \cup l6 \rightarrow l1^4 = 1111101011 \rightarrow l6\ saved$$

$$l1^4 \cup l7 \rightarrow l1^5 = 1111111111 \rightarrow l7\ saved$$

$$l1^5 = \{l1, l2, l3, l5, l6, l7\}$$

Застосування зворотного алгоритму і бінарної операції or до першого рядку здійснюється шляхом виконання трьох кроків.

1. Сортувати рядки за спаданням кількості одиничних елементів в них.

Приклад матриці M1:

	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	0	0	0	0	0
2	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Відсортована матриця M_{1S}:

	1	2	3	4	5	6	7	8	9	10
5	0	0	0	0	0	0	1	1	1	1
2	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

2. Виконати бінарну операцію or для першого та кожного наступного рядка до тих пір, поки не вийде повне покриття або не буде досягнутий кінець матриці (якщо досягнуто кінець матриці, але повне покриття не отримано, припинити обчислення):

$$l5 \cup l2 \rightarrow l5^1 = 0011001111 \rightarrow l2\ saved$$
$$l5^1 \cup l3 \rightarrow l5^2 = 0011111111 \rightarrow l3\ saved$$
$$l5^2 \cup l4 \rightarrow l5^3 = 0111111111 \rightarrow l4\ saved$$
$$l5^3 \cup l1 \rightarrow l5^4 = 1111111111 \rightarrow l1\ saved$$
$$l5^4 = \{l5, l2, l3, l4, l1\}$$

3. Видилити з множини $l5^4$ по одному рядку, починаючи з передостаннього. Кожен раз, після видалення рядку перевірити, чи не досягнуто повне бінарне покриття. Даний крок алгоритму виконується рекурсивно або реверсивно. Двійкове покриття може бути оптимізовано шляхом видалення надлишкових рядків.

$$ls \cup l2 \rightarrow ls^1 = 0011001111 \rightarrow l2\ saved$$
$$ls^1 \cup l3 \rightarrow ls^2 = 0011111111 \rightarrow l3\ saved$$
$$ls^2 \cup l4 \rightarrow ls^3 = 0111111111 \rightarrow l4\ saved\ (line\ to\ delete)$$
$$ls^3 \cup l1 \rightarrow ls^4 = 1111111111 \rightarrow l1\ saved$$
$$ls^4 = \{l5, l2, l3, l4, l1\}$$

Результат:

$$ls \cup l2 \rightarrow ls^1 = 0011001111 \rightarrow l2\ saved$$
$$ls^1 \cup l3 \rightarrow ls^2 = 0011111111 \rightarrow l3\ saved$$
$$ls^3 \cup l1 \rightarrow ls^4 = 1111111111 \rightarrow l1\ saved$$
$$ls^4 = \{l5, l2, l3, l1\}$$

Бінарне покриття без рядка $l4$ повне, рядок $l4$ буде виключений з остаточного покриття. Рядок $l4$ відзначається як первинно надлишковий. Тепер, коли знайдений один надлишковий рядок, алгоритм застосовується рекурсивно, поки не будуть видалені всі надлишкові рядки. Після того, як не залишиться надлишкових рядків для первинного рядка $l4$, алгоритм застосовується для наступного первинного рядка. У розглянутому прикладі наступним основним рядком є $l3$. Після того, як алгоритм буде застосований для всіх первинних рядків, здійснюється порівняння знайденої кількості надлишкових рядків і вибирається найкращий результат.

Застосування алгоритму для первинного рядка $l3$:

$$l5 \cup l2 \rightarrow ls^1 = 0011001111 \rightarrow l2\ saved$$
$$ls^1 \cup l3 \rightarrow ls^2 = 0011111111 \rightarrow l3\ saved(line\ to\ delete)$$
$$ls^3 \cup l1 \rightarrow ls^4 = 1111111111 \rightarrow l1\ saved$$
$$ls^4 = \{l5, l2, l3, l1\}$$

Результат:

$$ls \cup l2 \rightarrow ls^1 = 0011001111 \rightarrow l2\ saved$$
$$ls^3 \cup l1 \rightarrow ls^4 = 1111001111 \rightarrow l1\ saved$$
$$ls^4 = \{l5, l2, l1\}$$

Видалення рядку $l3$ призводить до того, що бінарне покриття стає неповним. Рядок $l3$ не може бути видаленим.

Застосування алгоритму до рядку $l2$:

$$l5 \cup l2 \rightarrow l5^1 = 0011001111 \rightarrow l2\ saved(line\ to\ delete)$$

$$l5^1 \cup l3 \rightarrow l5^2 = 0011111111 \rightarrow l3\ saved$$

$$l5^3 \cup l1 \rightarrow l5^4 = 1111111111 \rightarrow l1\ saved$$

$$l5^4 = \{l5, l3, l1\}$$

Результат:

$$l5^1 \cup l3 \rightarrow l5^2 = 0000001111 \rightarrow l3\ saved$$

$$l5^3 \cup l1 \rightarrow l5^4 = 1111001111 \rightarrow l1\ saved$$

$$l5^4 = \{l5, l1\}$$

Вилучення рядку $l2$ призводить до того, що бінарне покриття стає неповним. Рядок $l2$ не може бути вилученим.

Наступний метод реалізує бінарну операцію or і алгоритм для першого рядка матриці. На першому кроці вибирається первинний рядок, який не повинен містити тільки нульові елементи (не повинен бути порожнім). Якщо матриця містить тільки порожні рядки, вона пропускається і повертається порожній рядок (лістинг 6.21).

Лістинг 6.21 – Реалізація для першого рядка матриці

```
class Matrix(object):
…
    def binary_or_for_first_line_with_full_check(self):
        if not len(self.lines):
            return Line()
        first_line = self.lines[0].deepcopy()
        iter_lines = iter(self.lines)
        next(iter_lines)

        try:
            passedLoop = False
            while first_line.is_full_of_zeros():
                first_line = next(iter_lines)
                passedLoop = True
            if passedLoop:
                first_line = first_line.deepcopy()
        except StopIteration:
```

```
            return first_line

        for line in iter_lines:
            first_line.binary_or(line)
            if first_line.is_full_of_ones():
                break

        return first_line
…
end
```

Імплементація рекурсивного методу і двійкової операції or для першого рядка. Статичний метод «applyMatrixRecurtion» класу «Algorithm» приймає як аргумент матричний об'єкт «Matrix». Повернене значення - список рядків об'єктів типу «Line» (лістинг 6.22).

Лістинг 6.22 – Імплементація рекурсивного методу

```
class Algorithm(object):
…
    @staticmethod
    def applyMatrixRecurtion(matrix):
        matrix.sort_by_ones_qnty()
        matrix.updateLinesMapping()

        return matrix.binary_full_check_reverse()
…
end
```

Клас «Matrix» реализує описаний алгоритм (лістинг 6.23).

Листинг 6.23 – Реалізація алгоритму за допомогою класу «Matrix»

```
class Matrix(object):
…
    def binary_full_check_reverse(self):
        line = self.binary_or_for_first_line_with_full_check()
        result_lines = []

        for value in line.indexes:
            newLine = line.deepcopy()
            newLine.binary_xor(self.lineForIndex(value))
            result_lines.append(newLine)

        return result_lines
…
end
```

Приклади вхідної матриці та вихідних значень наведено нижче (лістинги 6.24-26).

Лістинг 6.24 – Вхідна матриця

```
test_table = [
             [0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0], #0
             [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #1
             [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0], #2
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #3
             [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #4
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], #5
             [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #6
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #7
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], #8
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #9
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0], #10
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #11
             [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0], #12
             [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], #13
             [0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], #14
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], #15
             [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #16
             [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], #17
             [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1], #18
             ]
```

Лістинг 6.25 – Код генерації вихідних значень.

```
if __name__ == '__main__':
    lines = Algorithm.applyMatrixRecurtion(
                      MatrixUtils.matrixFromListValues(test_table))
    for line in lines:
        print line
```

Лістинг 6.26 – Вихідні значення

```
index      :0
indexes    :[18, 1, 14, 2, 12, 13, 6, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
ones count :[1, 3, 3, 2, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1]
index      :0
indexes    :[1, 14, 2, 12, 13, 6, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
ones count :[1, 2, 3, 1, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 0]
index      :0
indexes    :[18, 14, 2, 12, 13, 6, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
ones count :[1, 3, 3, 2, 2, 3, 0, 1, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1]
index      :0
indexes    :[18, 1, 2, 12, 13, 6, 8, 10, 15, 17]
```

```
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
ones count :[1, 2, 3, 2, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 12, 13, 6, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1]
ones count :[1, 3, 2, 2, 3, 3, 1, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 13, 6, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1]
ones count :[1, 3, 3, 1, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 12, 6, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1]
ones count :[1, 3, 2, 2, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 12, 13, 8, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
ones count :[1, 3, 3, 2, 3, 3, 1, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 12, 13, 6, 10, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1]
ones count :[1, 3, 3, 2, 3, 3, 1, 1, 1, 1, 1, 2, 0, 1, 1, 1, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 12, 13, 6, 8, 15, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1]
ones count :[1, 3, 3, 2, 3, 3, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 12, 13, 6, 8, 10, 17]
values     :[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
ones count :[1, 3, 3, 2, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1]
index      :0
indexes    :[18, 1, 14, 2, 12, 13, 6, 8, 10, 15]
values     :[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
ones count :[0, 3, 3, 2, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1]
```

Порівняння методів здійснюється за допомогою процедур оцінки продуктивності алгоритмів (лістинг 6.27).

Лістинг 6.27 – Оцінка продуктивності алгоритмів

```
def recurtion_test():
    for _ in range(100):

Algorithm.applyMatrixRecurtion(MatrixUtils.matrixFromListValues(test_table))

def first_line_or_test():
    for _ in range(100):

Algorithm.applyMatrixFirstLineOR(MatrixUtils.matrixFromListValues(test_table))


if __name__ == '__main__':
    cProfile.run('first_line_or_test()')
    cProfile.run('recurtion_test()')
```

Результати застосування методу горизонтального розбиття матриці для першого рядка (лістинг 6.28).

Лістинг 6.28 – Застосування методу горизонтального розбиття

```
21105 function calls in 0.146 seconds
Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.146    0.146 <string>:1(<module>)
      100    0.001    0.000    0.103    0.001 algorithm.py:15(applyMatrixFirstLineOR)
     5500    0.026    0.000    0.026    0.000 line.py:19(ones_qnty)
     1800    0.011    0.000    0.021    0.000 line.py:29(is_full_of_ones)
      100    0.001    0.000    0.001    0.000 line.py:34(is_full_of_zeros)
     1800    0.041    0.000    0.063    0.000 line.py:39(binary_or)
      100    0.002    0.000    0.005    0.000 line.py:78(deepcopy)
     2000    0.014    0.000    0.022    0.000 line.py:9(__init__)
        1    0.002    0.002    0.146    0.146 main.py:27(first_line_or_test)
      100    0.000    0.000    0.000    0.000 matrix.py:11(__init__)
      100              0.011               0.000              0.102              0.001
matrix.py:73(binary_or_for_first_line_with_full_check)
      100    0.017    0.000    0.042    0.000 matrix_utils.py:93(matrixFromListValues)
      100    0.000    0.000    0.000    0.000 {iter}
     5900    0.012    0.000    0.012    0.000 {len}
     3000    0.006    0.000    0.006    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
      301    0.002    0.000    0.002    0.000 {method 'extend' of 'list' objects}
      100    0.000    0.000    0.000    0.000 {next}
        1    0.000    0.000    0.000    0.000 {range}
```

Результати застосування рекурсивного методу (лістинг 6.29):

Лістинг 6.29 – Застосування рекурсивного методу

```
596943 function calls in 3.720 seconds
Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.720    3.720 <string>:1(<module>)
      100    0.002    0.000    3.678    0.037 algorithm.py:52(applyMatrixRecurtion)
   341792    1.177    0.000    1.177    0.000 line.py:19(ones_qnty)
    41000    0.257    0.000    0.493    0.000 line.py:29(is_full_of_ones)
      100    0.000    0.000    0.001    0.000 line.py:34(is_full_of_zeros)
    41000    0.845    0.000    1.160    0.000 line.py:39(binary_or)
     1100    0.021    0.000    0.024    0.000 line.py:57(binary_xor)
     1200    0.020    0.000    0.043    0.000 line.py:78(deepcopy)
     3100    0.024    0.000    0.036    0.000 line.py:9(__init__)
        1    0.003    0.003    3.720    3.720 main.py:23(recurtion_test)
      100    0.012    0.000    1.949    0.019 matrix.py:102(binary_full_check_reverse)
      100    0.000    0.000    0.000    0.000 matrix.py:11(__init__)
      100    0.005    0.000    1.688    0.017 matrix.py:18(sort_by_ones_qnty)
      100    0.039    0.000    0.039    0.000 matrix.py:39(updateLinesMapping)
     1100    0.003    0.000    0.003    0.000 matrix.py:45(lineForIndex)
```

```
      100              0.210              0.002              1.869              0.019
matrix.py:73(binary_or_for_first_line_with_full_check)
   109346    0.720    0.000    1.426    0.000 matrix_utils.py:110(reverseLines)
      100    0.013    0.000    0.038    0.000 matrix_utils.py:93(matrixFromListValues)
      100    0.000    0.000    0.000    0.000 {iter}
    47300    0.089    0.000    0.089    0.000 {len}
     4200    0.009    0.000    0.009    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler'
objects}
     3601    0.008    0.000    0.008    0.000 {method 'extend' of 'list' objects}
     1100    0.003    0.000    0.003    0.000 {method 'remove' of 'list' objects}
      100    0.257    0.003    1.683    0.017 {method 'sort' of 'list' objects}
      100    0.000    0.000    0.000    0.000 {next}
        1    0.000    0.000    0.000    0.000 {range}
```

Рекурсивний метод з реверсом характеризується більш низькою швидкодією і більш високою точністю за рахунок застосування бінарної xor рекурсії.

6.5 Висновки до розділу 6

Виконана програмно-апаратна реалізація процесора, яка заснована на використанні мов програмування: C ++, Verilog, Python 2.7 і платформ: Microsoft Windows, X Window (в Unix і Linux) і Macintosh OS X. Генератор HDL-коду дає можливість автоматично синтезувати HDL-коди процесорної структури від 1 до 16 двійкових розрядів для паралельної обробки відповідної кількості вхідних векторів або слів.

Верифікація HDL-коду процесора виконана на тестових прикладах задачі покриття, що використовують дві стратегії оптимізації: реверсивний алгоритм для усунення надмірності і розбиття матриці покриттів на частини з метою їх подальшої паралельної обробки процесорами Xace.

Продуктивність розглянутих двох методів залежить від розподілу одиничних елементів в матриці, інформація про який надзвичайно важлива для великих матриць. Однак найчастіше такої інформації немає. В цьому випадку обидва розглянутих підходи можуть бути використані для логічної оптимізації та збільшення швидкодії другого методу.

Найкращим розв’язком є поєднання двох розглянутих методів: сортування матриці і подальший її розподіл на підматриці; застосування реверсивного алгоритму рекурсивного методу для підматриць, що містять більше одного одиничного елемента, і простого алгоритму горизонтального розбиття для решти матриць; перехід до пункту 2 першого рекурсивного методу.

7 ІНФРАСТРУКТУРА ЗАХИСТУ І СЕРВІСНОГО ОБСЛУГОВУВАННЯ ВІРТУАЛЬНИХ КІБЕРСИСТЕМ

Пропонується математичний апарат створення інфраструктури реальних або віртуальних програмно-апаратних телекомунікаційних та інформаційних кібернетичних систем (КС), орієнтований на захист від несанкціонованого доступу до сервісів, визначених у специфікації системи, шляхом проникнення через легальні інтерфейси взаємодії компонентів, що володіють уразливостями. Інфраструктура захисних сервісів створюється разом з кіберсистемою і супроводжує останню протягом всього життєвого циклу, обслуговуючи всі наступні модифікації КС, і постійно вдосконалюючись шляхом підвищення інтелекту через поповнення в часі бібліотек конструктивних і деструктивних компонентів.

7.1 Тенденції захисту кіберпростору

Корпоративні мережі, як і персональні комп'ютери, а також окремі сервіси (програмні продукти) йдуть у «хмари» кіберпростору, які мають яскраво виражену тенденцію до розшарування інтернету за спеціалізованими сервісами, рис. 7.1. Якщо сьогодні 4000000000 користувачів з'єднуються в інтернеті (1 zettabytes = $10^{21} = 2^{70}$ байт) за допомогою 50 мільярдів гаджетів, то через п'ять років кожен активний користувач буде мати не менше 10 пристроїв для зв'язку з кіберпростором. Стає неможливим використання персональних комп'ютерів без мультиплікування інформації на всі пристрої. Але навіть процедура простого копіювання вимагає все більше непродуктивного часу на сервісне обслуговування систем і проектів, яке досягає 50% за наявністю декількох ідентичних за функціями пристроїв або серверів. Непрофесійне сервісне обслуговування останніх створює проблеми надійності збереження даних, а також несанкціонованого доступу до них. Виникає також проблема віддаленого доступу до фізичних пристроїв при

переміщенні користувачів у просторі, коли важко знайти необхідні сервіси та інформацію серед гаджетів, залишених вдома або в офісі. Економічний чинник ефективного використання придбаних додатків, розміщених в гаджетах і персональних комп'ютерах, змушує користувача відмовлятися від їх покупки на користь майже безкоштовної оренди сервісів на хмарах. Все згадане вище є суттєвими аргументами і незаперечними доказами неминучого переходу або результату всього людства у кіберпростір віртуальних мереж і комп'ютерів, що розташовуються в професійно надійних хмарах сервісів.

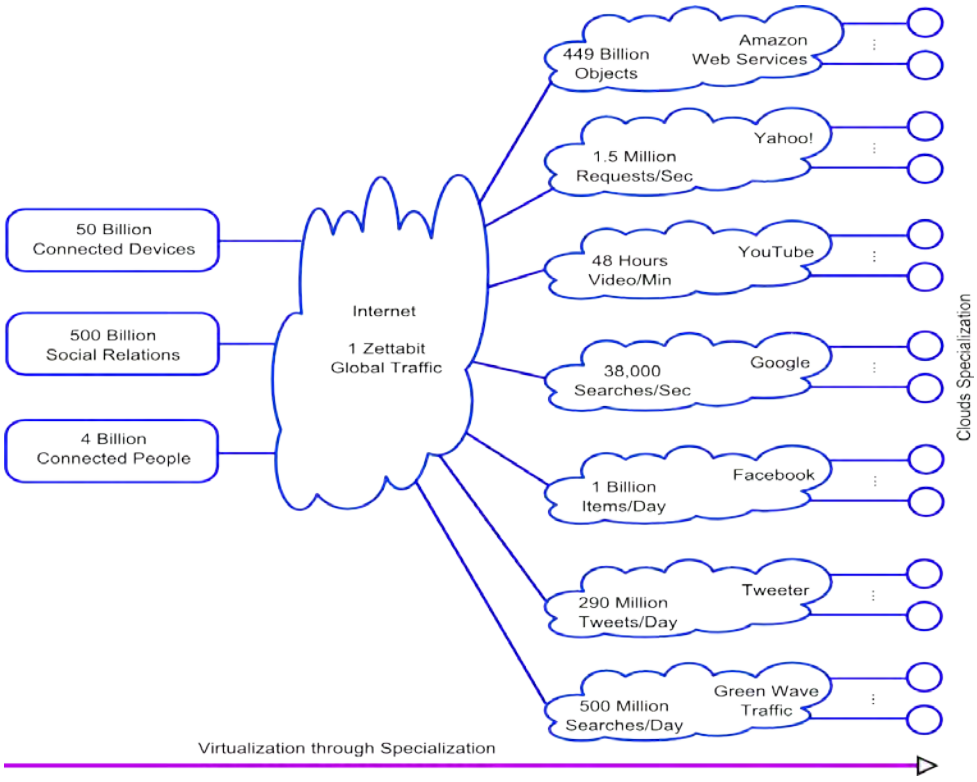


Рисунок 7.1 – Віртуалізація реального світу

Переваги віртуального світу полягають в тому, що мікро-комірки і макро-мережі в хмарах інваріантні по відношенню до незліченних гаджетів кожного користувача або корпорації. Хмарні компоненти знімають практично всі згадані вище проблеми надійності, безпеки, сервісного обслуговування і практично не мають недоліків. У зв'язку з глобальним переходом корпорацій і користувачів в хмари надзвичайно актуальною і

ринково привабливою стає проблема захисту інформації і компонентів кіберпростору від несанкціонованого доступу, деструктивних проникнень, вірусів [63, 94-102]. Необхідно створювати надійну, тестопридатність і захищену від несанкціонованих проникнень інфраструктуру кіберпростору і його компонентів (віртуальні персональні комп'ютери і корпоративні мережі) за аналогією з існуючими сьогодні рішеннями в реальному кібернетичному світі.

Поняття, що визначаються словами «проникнення» і «вразливість», є взаємно доповнюючими один одного. Якщо є вразливість об'єкта або процесу, то в неї, як в дірку, можливе проникнення деструктивності, яка вписується в функціональність кіберсистеми. Вірно і зворотне, якщо зафіксовано проникнення, то воно сталося внаслідок наявності в системі уразливості (дірки). Проблема захисту кіберсистеми від несанкціонованого доступу полягає в «неможливості» розрізнити деструктивність від «конструктивності» або валідного користувача. Тим не менше, існують методики, технології, програмні засоби і системи, здатні ефективно вирішувати питання захисту корпоративного або персонального кіберпростору з наперед заданою вірогідністю проникнення. Існуючі публікації [63, 94-102] за даним напрямком оперують такими термінами. Тест проникнень [95, 96, 101, 102] – сукупність зовнішніх і внутрішніх деструктивних впливів, спрямованих на виявлення вразливостей доступу до сервісів КС шляхом моделювання або аналізу проникнень на моделі кіберсистеми.

Якість тесту визначається його повнотою у відсотках відносно перевірки всіх можливих типів вразливостей, що генеруються вручну або автоматично для кожної конкретної кіберсистеми.

Результат тестування реальної системи (System Under Penetration Test – SUPT) формує кількісну оцінку вразливості, а також список структурних вразливостей наперед заданих типів, виявлених в процесі тестового експерименту.

Якщо процес тестування зафіксував непорожній список деструктивних (вразливостей) [97-100], то необхідно виконувати діагностування на основі використання тестів діагностування з метою визначення місця причини та виду уразливості з наперед заданою глибиною пошуку деструктивностей. Після точного визначення всіх вразливостей виконуються процедури їх усунення шляхом часткової або повної реконструкції кіберсистеми на основі використання перевірених бібліотечних структурних рішень.

Всі процедури, описані вище, використовують три бібліотеки: 1) негативну, що описує всі можливі типи вразливостей; 2) позитивну, де кожній уразливості ставиться у відповідність вірне програмно-апаратне рішення, що усуває деструктивність; 3) неперевірені рішення, які складають потенціал «інтелекту» КС, що довизначається в процесі експлуатації кіберсистеми. Всі три бібліотеки необхідно поповнювати як у процесі проектування КС, так і на стадії експлуатації у реальному часі.

Завдання інфраструктури захисного сервісу кіберсистеми:

1) Синтез (дедуктивної) моделі КС для тестування, діагностування та відновлення невразливості кіберсистеми.

2) Генерування тестів перевірки та діагностування вразливостей, близьких до 100% повноті.

3) Створення алгоритмів пошуку вразливостей з наперед заданою глибиною діагностування.

4) Створення генераторів тестів перевірки та діагностування вразливостей, близьких до 100% повноті.

5) Тестопридатності проектування (модифікації) невразливих кіберсистем, «вільних» від вразливостей на поточний момент розвитку технологічної та математичної культури.

6) Розробка вбудованої інфраструктури захисного сервісу для кіберсистем, орієнтованих на моніторинг, тестування, діагностування та відновлення невразливості в реальному масштабі часу в процесі експлуатації.

7) Розробка спеціалізованих маршрутів (алгоритмів, планів) моніторингу, тестування, діагностування та відновлення невразливості КС в реальному масштабі часу в процесі експлуатації.

8) Верифікація інфраструктурних тестопригодності рішень, розроблених для реальних КС.

Об'єкт тестування – кібернетична система взаємодіючих програмно-апаратних, телекомунікаційних, інформаційних компонентів, орієнтована на надання якісних сервісів через стандартні інтерфейси санкціонованою користувачеві в реальному масштабі часу. Всі типи вразливостей (проникнень) не виводять об'єкт тестування за кордону заданої функціональності кіберсистеми, представленої булевою функцією:

$$Y = f(X_1, X_2, ..., X_i, ..., X_n),\ X_i, Y \in \{0,1\}.$$

Тому модель вразливостей накладається на графову структуру функціональніх модулів, что мають вхідні і вихідні транзакційні змінні. Транзакційній граф зображений дугами - функціональностямі (сервісамі) з моніторамі (асерціями), а також вершинами, що формують стани киберсистеми, за допомогою змінніх, пам'яті, інтерфейсніх портів введення-виведення інформації, пріймачів, терміналів, комп'ютерів: $F = (A * B) \times S$, де $S = \{S_1, S_2, ..., S_i, ..., S_m\}$ – вершини або стани КС при моделюванні тестових сегментів. Кожний стан $S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\}$ визначається значеннями істотних змінних КС (змінні, пам’ять, термінали, комп’ютери). Орієнтовані дуги графа є функціональні блоки: $B = (B_1, B_2, ..., B_i, ..., B_n)$, $\bigcup_{i=1}^{n} B_i = B$; $\bigcap_{i=1}^{n} B_i = \varnothing$, де кожному з них може бути поставлена у відповідність асерція $A_i \in A = \{A_1, A_2, ..., A_i, ..., A_n\}$ для моніторингу функціональностей у часі і у просторі.

Існують базові технології тестування безпеки кіберсистем: OSSTMM – The Open Source Security Methodology Manual; NIST Guideline on Network Security Testing; ISACA Switzerland – Testing IT Systems Security With Tiger Teams; Draft Guideline on Network Security Testing; NIST Special Publication 800-26 Security Self-Assessment Guide for Information Technology Systems; Cybersecurity Vulnerability Assessment Methodologies (Cybersecurity VAMs); Information Systems Security Assessment Framework, OISSG.

Функція мети подана підвищенням ефективності сервісного обслуговування на основі стандартів тестування, граничного сканування і спеціальних технологій діагностування та відновлення невразливості КС, яка визначається мінімальним значенням рівня вразливості, часу відновлення працездатності T і нефункціональної програмно-апаратної надмірності H:

$$E = F(L, T, H) = \min[\frac{1}{3}(L + T + H)],$$
$$Y = (1 - P)^{n};$$
$$L = 1 - Y^{(1-k)} = 1 - (1 - P)^{n(1-k)};$$
$$T = \frac{(1 - k) \times H^{s}}{H^{s} + H^{a}}; \; H = \frac{H^{a}}{H^{s} + H^{a}},$$

де L – доповнення до рівня невразливості Y, яке залежить від тестопригодності КС k, ймовірності P існування вразливостей і числа невиявлених деструктиву n. Час тестування і діагностування залежить від тестопригодності архітектури k, помноженої на число структурних компонентів інфраструктури, віднесеного до загальної кількості елементів КС. Надмірність знаходиться у залежності від структурної складності тестопридатної надбудови, поділеної на програмно-апаратну складність КС. Надмірність інфраструктури забезпечує задану глибину діагностування вразливостей за час, визначений замовником.

7.2 Математичний апарат інфраструктури захисного сервісу

Тут розглядаються метрика, алгебра, структури даних і моделі оцінювання якості взаємодії процесів, явищ, об'єктів і компонентів в кіберпросторі та кіберсистемі, необхідні при створенні ефективних рушіїв для обчислювальних процедур аналізу даних у процесах тестування проникнень і відновлення невразливості.

Критерії взаємодії об'єктів тестування грунтуються на використанні бета-метрики вимірювання відстаней в кіберпросторі. Кіберпростір – дискретний векторно-логічний простір – сукупність взаємодіючих за відповідною метрикою інформаційних процесів і явищ, що описані векторами логічних змінних і що використовують як носій комп'ютерні системи та мережі. Метрика – спосіб вимірювання відстані у просторі між компонентами процесів або явищ, описаних векторами логічних змінних. Відстань (булева похідна, ступінь зміни, відмінності або близькості) в кіберпросторі визначається xor-відношенням векторів (матриць), які позначають компоненти процесу або явища, що відрізняє його від кодової відстані за Хемінгом. Процедури порівняння, вимірювання, оцінювання, розпізнавання, тестування, діагностування, оперують xor-відношенням об'єктів або їх компонентів. Компонент простору представлений k-мірним вектором $a = (a_1, a_2, ..., a_j, ..., a_k)$, $a_j \in \{0,1\}$, де кожна його координата визначена у двійковому алфавіті, 0 – «неправда», 1 – «істина». Нуль-вектор є k-мірний кортеж, все координати якого дорівнюють нулю: $a_j = 0, j = \overline{1,k}$.

Метрика β кібернетичного простору визначається рівністю

$$\beta = \bigoplus_{i=1}^{n} d_i = 0,$$

яка формує нуль-вектор для xor-суми відстаней d_i між ненульовим і кінцевим числом об'єктів, замкнутих у цикл. Тут n – кількість відстаней між компонентами (векторами) простору, складовими цикл $D=(d_1,d_2,...,d_i,...,d_n)$, d_i – є вектор відстані, відповідний ребру циклу, що з'єднує два компоненти (вектора) a,b простору, який далі позначається без індексу як $d(a,b)$. Відстань між двома об’єктами a і b є похідний вектор: $d(a,b)=(a_j \oplus b_j)_1^k$. Векторному значенню відстані відповідає норма (скаляр), що визначається кодовою відстанню за Хемінгом між двома векторами у вигляді числа одиниць вектора d(a,b). Метрика β векторного логічного двійкового простору є рівна нуль-вектору xor-сума відстаней між кінцевим числом вершин графа, що утворюють цикл. Тепер можна дати більш формальне визначення кіберпростору, як векторно-логічне, нормоване β-метрикой, де xor-сума відстаней між кінцевим числом точок циклу дорівнює нуль-вектору. Визначення метрики через відношення дозволяє скоротити систему аксіом (рефлексивності, симетричності і транзитивності, трикутного замикання) з трьох до однієї і поширити її дію на скільки завгодно складні структури n-мірного логічного простору. Класичне завдання метрики для визначення взаємодії однієї, двох і трьох точок у векторному логічному просторі, є окремим випадком β-метрики при $i=1,2,3$ відповідно:

$$M \subset \beta = \begin{cases} d_1 = 0 \leftrightarrow a = b; \\ d_1 \oplus d_2 = 0 \leftrightarrow d(a,b) = d(b,a); \\ d_1 \oplus d_2 \oplus d_3 = 0 \leftrightarrow d(a,b) \oplus d(b,c) = d(a,c). \end{cases}$$

Векторно-логічний транзитивний трикутник має повну аналогію чисельному виміру відстані в метричному М-просторі, який задається системою аксіом, що визначає взаємодію однієї, двох і трьох точок в будь-якому просторі:

$$M = \begin{cases} d(a,b) = 0 \leftrightarrow a = b; \\ d(a,b) = d(b,a); \\ d(a,b) + d(b,c) \geq d(a,c). \end{cases}$$

Специфіка аксіоми трикутника (метричного) М-простору полягає в чисельному (скалярному) порівнянні відстаней трьох об'єктів. При цьому інтервальна невизначеність відповіді – дві сторони трикутника можуть бути більше або рівні третьій – малопридатна для визначення точної довжини останньої сторони. Бета-метрика усуває даний недолік і виключає невизначеність бінарного відношення детермінованих процесів або явищ. Третя сторона трикутника у векторному логічному просторі визначається двійковим вектором-відстанню між двома вершинами шляхом обчислення xor-суми відстаней двох інших сторін трикутника:

$$d(a,b) \oplus d(b,c) = d(a,c) \rightarrow d(a,b) \oplus d(b,c) \oplus d(a,c) = 0 .$$

Метрика β кібернетичного багатозначного векторно-логічного простору, є вектор, рівний значенню $\varnothing$ за всіма координатами, отриманий шляхом застосування симетричної різниці відстаней між кінцевим числом точок, що утворюють цикл:

$$\beta = \overset{n}{\underset{i=1}{\Delta}} d_i = \varnothing .$$

Тут кожна координата вектора, відповідного об'єкту, визначена в алфавіті, що становить булеан на універсумі примітивів потужністю p: $a_j = \{\alpha_1, \alpha_2, ..., \alpha_r, ..., \alpha_m\}, m = 2^p$.

На основі введеної метрики аналізу кіберпростору вводяться критерії оцінювання взаємодії кінцевого числа об'єктів між собою. Скалярний

критерій взаємодії двох об'єктів (процесів) в дискретному булевому просторі, поданих k-мірними багатозначними векторами

$$m = (m_1, m_2, ..., m_j, ..., m_k),\ m_j \in \{0,1,x\};\ A = (A_1, A_2, ..., A_j, ..., A_k),\ A_j \in \{0,1,x\},$$

необхідний для порівняння і подальшого вибору, кращого в деякому розумінні, розв'язку. Ступінь приналежності m-вектора до A-вектору позначається як $\mu(m \in A)$, непринадлежності – $\overline{\mu}(m \in A)$. Існує 5 типів теоретико-множинної взаємодії двох векторів:

$$1)\ m = A;\ 2)\ m \subset A;\ 3)\ A \subset m;\ 4)\ m \cap A \neq \{m, A, \varnothing\};\ 5)\ m \cap A = \varnothing.$$

Мета скалярного критерію – оцінити будь-яке з вказаних взаємодій інтервальною оцінкою [0,1] шляхом спільного використання трьох параметрів: кодової відстані $d(m, A)$ і двох функцій неприналежності $\overline{\mu}(m \in A) = 1 - \mu(m \in A)$, $\overline{\mu}(A \in m) = 1 - \mu(A \in m)$:

$$Q = \frac{1}{3}\left[\frac{1}{k} d(m, A) + [1 - \mu(m \in A)] + [1 - \mu(A \in m)]\right],$$

$$d(m, A) = \text{card}\,(m_i \overset{k}{\underset{i=1}{\cap}} A_i = \varnothing);$$

$$\mu(m \in A) = 2^{c-a};$$

$$\mu(A \in m) = 2^{c-b};$$

$$a = \text{card}\,(A_i = x), i = \overline{1,k};$$

$$b = \text{card}\,(m_i = x), i = \overline{1,k};$$

$$c = \text{card}\,(m_i \overset{k}{\underset{i=1}{\cap}} A_i = x).$$

Тут $d(m,A)=\mathrm{card}\,(m_i \underset{i=1}{\overset{k}{\cap}} A_i=\varnothing)$ – потужність або кількість порожніх координатних перетинів двох взаємодіючих векторів, що складають відстань за Хемінгом; $\mu(m\in A)=2^{c-a}$ ($\mu(A\in m)=2^{c-b}$) – відношення загального для m і A простору до простору вектора A(m), що формує зазначену функцію приналежності. Операції координатного перетину (and), симетричної різниці (xor) визначені для символів алфавіту Кантора $A=\{0,1,x=\{0,1\},\varnothing\}$, що кодируються векторами (01,10,11,00) відповідно:

$\cap$	0	1	x	$\varnothing$
0	0	$\varnothing$	0	$\varnothing$
1	$\varnothing$	1	1	$\varnothing$
x	0	1	x	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

$\wedge$	01	10	11	00
01	01	00	01	00
10	00	10	10	00
11	01	10	11	00
00	00	00	00	00

Δ	0	1	x	$\varnothing$
0	$\varnothing$	x	1	0
1	x	$\varnothing$	0	1
x	1	0	$\varnothing$	x
$\varnothing$	0	1	x	$\varnothing$

$\oplus$	0	1	x	$\varnothing$
0	00	11	10	01
1	11	00	01	10
x	10	01	00	11
$\varnothing$	01	10	11	00

Нормування параметрів критерію (кодової відстані і функцій неналежності) дозволяє оцінити рівень взаємодії векторів в чисельному інтервалі [0,1]. З урахуванням ізоморфізму теоретико-множинних і логічних операцій, критерій якості можна трансформувати до вигляду:

$$Q=\frac{1}{3}\left[\frac{1}{k}d(m,A)+[1-\mu(m\in A)]+[1-\mu(A\in m)]\right],$$

$$d(m,A)=\mathrm{card}\,(m_i \underset{i=1}{\overset{k}{\oplus}} A_i=U);$$

$$\mu(m\in A)=\mathrm{card}\,(A_i=U)-\mathrm{card}\,(m_i \underset{i=1}{\overset{k}{\wedge}} A_i=U);$$

$$\mu(A\in m)=\mathrm{card}\,(m_i=U)-\mathrm{card}\,(m_i \underset{i=1}{\overset{k}{\wedge}} A_i=U);$$

$$U=\begin{cases}1\leftarrow\{m_i,A_i\}\in\{0,1\};\\ x\leftarrow\{m_i,A_i\}\in\{0,1,x\}.\end{cases}$$

Якщо вектори m і A – двійкові за всіма координатам, то змінна U=1 і обчислення проводяться за правилами двійкової $\oplus$-операції. Якщо вектори

m і A визначені в троїчному алфавіті, то змінна U=x ініціює обчислення на основі використання теоретико-множинної операції симетричної різниці Δ. Перший компонент $\frac{1}{n}d(m,A)$ критерію формує ступінь неспівпадання k-мірних векторів у вигляді кодової відстані за Хемінгом, віднесеного до довжини вектора, шляхом виконання операції xor над усіма координатами; другий і третій компоненти $[1-\mu(m\in A)]+[1-\mu(A\in m)]$ визначають ступені неналежності результату кон'юнкції до простору кожного з двох взаємодіючих векторів. Якщо такі міри дорівнюють нулю $\frac{1}{n}d(m,A)=0,\ [1-\mu(m\in A)]=0,\ [1-\mu(A\in m)]=0$, то об'єкти ідентичні один одному. Поняття приналежності і неналежності є взаємодоповнюючими, але в даному випадку більш технологічно обчислювати неналежність, оскільки загальноприйнятим в літературі є поняття нульової розбіжності об'єктів, що свідчить про їх повну ідентичність. Даний критерій працює в інтервалі [0,1]. Повне співпадання двох об'єктів $d(m,A)=0,\ \mu(m\in A)=1,\ \mu(A\in m)=1$ характеризується нульовою оцінкою критерию $Q=\frac{1}{3}\left[\frac{1}{k}0+[1-1]+[1-1]\right]=0.$ Протилежним варіантом оцінювання максимальне неспівпадіння двох об’єктів: $d(m,A)=k,\ \mu(m\in A)=0,\ \mu(A\in m)=0$, яке визначається оцінкою взаємодії: $Q=\frac{1}{3}\left[\frac{1}{k}k+[1-0]+[1-0]\right]=1.$ Якщо параметри взаємодії дорівнюють $d(m,A)=0,\ \mu(m\in A)=\frac{1}{2},\ \mu(A\in m)=\frac{1}{2}$, то критерій буде мати наступну оцінку: $Q=\frac{1}{3}\left[\frac{1}{k}0+[1-\frac{1}{2}]+[1-\frac{1}{2}]\right]=\frac{1}{3}.$ Взаємодія (перетин) двох векторів: A = (XXX1X) и m = (XX0X0) дає спільний простір, що дорівнює (XX010) = {00010, 01010, 10010, 11010}. Критерій якості взаємодії при

параметрах $d(m,A)=0,\ \mu(m\in A)=\frac{1}{2},\ \mu(A\in m)=\frac{1}{4}$ буде мати наступну оцінку:

$$Q=\frac{1}{3}\left[\frac{1}{k}0+[1-\frac{1}{2}]+[1-\frac{1}{4}]\right]=\frac{1}{4}.$$

Гідність введеного критерію (неналежності, відмінності) полягає в лінійності зміни його чисельного значення від 0 до 1 в міру збільшення «відстані» від повного співпадіння двох об'єктів до максимально можливого, коли кодова відстань одно $d(m,A)=k.$

Критерій може бути використаний в задачах відстеження цілі, руху за заданим маршрутом, тестування і діагностування функціональних порушень і вразливостей, пошуку, розпізнавання та прийняття рішень. Критерій якості Q, використовуваний для виконання регуляторної функції при оцінюванні взаємодії об'єктів в реальному масштабі часу, необхідно мінімізувати. Тим не менш, скалярна оцінка має тільки інтегральні властивості взаємодії двох об'єктів, що дозволяє здійснювати порівняння декількох відстаней, частіше міри близькості одного об'єкта по відношенню до кінцевої множини інших. Недоліком інтегральної оцінки є неоднозначність її приведення до вихідного векторного еквіваленту, як і будь-якого іншого функціонального відношення: пряма імплікація однозначна, зворотна – багатозначна. Тому повна картина аналізу взаємодії об'єктів повинна містити не тільки інтегральний скалярний критерій Q, а й результат їх векторного відношення

$$Q(m,A)=m\oplus A,$$

який більш інформативний для подальшої корекції напрямку вирішення завдань синтезу або аналізу процесів взаємодії в рамках існуючої системи. Як отримати векторний критерій якості взаємодії двох об'єктів? Формула скалярного критерію якості після проведення векторних операцій використовує процедури обчислення трьох компонентів: кодова відстань, що визначається числом одиниць в координатах результуючого вектора,

отриманого на основі xor-операції $d(m,A)=m\oplus A$ і дві функції приналежності $\mu=\mu(m\in A)\vee\mu(A\in m)=(A\wedge\overline{m\wedge A})\vee(m\wedge\overline{m\wedge A})$, які в сукупності також визначаються xor-операцією, у загальному випадку на замкнутому теоретико-множинному алфавіті:

$$\mu=(A\wedge\overline{m\wedge A})\vee(m\wedge\overline{m\wedge A})=[A\wedge(\overline{m}\vee\overline{A})]\vee[m\wedge(\overline{m}\vee\overline{A})]=$$
$$=[(A\wedge\overline{m})\vee(A\wedge\overline{A})\vee(m\wedge\overline{m})\vee(m\wedge\overline{A})]=$$
$$=(A\wedge\overline{m})\vee(m\wedge\overline{A})=m\oplus A.$$

Логічне об'єднання двох векторних функцій, які формують кодову відстань і взаємну належність один одному, дає, природно, шуканий результат: $Q=d(m,A)\vee[\mu(m\in A)\vee\mu(A\in m)]=(m\oplus A)\vee(m\oplus A)=m\oplus A.$

Це означає, що за суттю взаємодія будь-яких об'єктів в кіберпросторі визначається виконанням симетричної різниці в багатозначному алфавіті (xor-операції у двійковому):

Δ	0	1	x	∅
0	∅	x	1	0
1	x	∅	0	1
x	1	0	∅	x
∅	0	1	x	∅

0=01; 1=10
x=11; ∅=00 →

⊕	0	1	x	∅
0	00	11	10	01
1	11	00	01	10
x	10	01	00	11
∅	01	10	11	00

Але при кодуванні символів алфавіту двійковими векторами-примітивами, операція симетричної різниці між символами в координатах векторів перетворюється на xor-операцію двійкових векторів. Інші логічні операції при формуванні векторної оцінки взаємодії об'єктів в кіберпросторі, згідно з наведеними вище формулами, не використовуються. Як приклад нижче запропоновані процедури виконання операції симетричної різниці і xor над двома формами об'єктів, представленими у вигляді символів алфавіту Кантора і двійкових кодів:

m =	x	x	x	x	1	0	1	0
A =	1	0	0	x	x	x	1	0
Δ =	0	1	1	∅	0	1	∅	∅
m =	11	11	11	11	10	01	10	01
A =	10	01	01	11	11	11	10	01
⊕ =	01	10	10	00	01	10	00	00

Другий приклад ілюструє обчислення взаємодії векторів в двотактному алфавіті опису автоматних змінних $B^2(Y)$ у форматах символьного і двійкового опису координат:

m =	Y	A	B	S	P	L	E	Q
A =	H	S	J	L	E	L	F	C
Δ =	L	B	H	E	H	∅	Y	Y
m =	1111	1100	0011	1001	0110	1101	0100	1000
A =	0010	1001	0001	1101	0100	1101	1011	0111
⊕ =	1101	0101	0010	0100	0010	0000	1111	1111

Тут є цікавим факт, що в кубітному форматі опису символьних змінних теоретико-множинні, в загальному випадку послідовно виконуються, операції над елементами множин замінюються паралельними операціями, що істотно підвищує швидкодію обчислювальних процесів аналізу моделей за рахунок відповідного збільшення обсягу пам'яті. Для створення кубітних структур даних обчислювальних процесів необхідно визначити: 1) універсум примітивів (процесів або явищ) з подальшим їх унітарним кодуванням в межах кубіта; 2) компактну систему (структуру) відношень (функціональних), які задають поведінку об'єкта; 3) послідовність обробки компонентів структури на основі паралельного виконання векторних логічних операцій, які заміняють теоретико-множинні, послідовні у часі, обчислювальні процедури.

Дві форми (скалярна і векторна) існування критерію якості $q = \{Q, Q(m, A)\}$ спрямовані на вибір кращого розв’язку (для користувача) і деталізацію відмінностей між об'єктами (для комп'ютера) відповідно. Чисельний еквівалент зручний для людини, яка не здатна оперувати

лінгвістичними (багатозначними) змінними при оцінці взаємодії об'єктів, поданих векторами. До того ж, дві однакові чисельні оцінки не означають ідентичності двох відстаней при взаємодії трьох об'єктів у просторі. Наприклад: d(a,b) = 0011 = 2, d(a,c) = 1100 = 2, при a = 0000, b = 1100, c = 0011. Тому до скалярної оцінки необхідно мати векторний еквівалент критерію якості взаємодії, який показує структуру подібності та відмінності за всіма параметрами (змінними) векторів.

Обчислити критерій – значить визначити ступінь належність чи неналежність даного процесу або явища, в тому числі, до деякого класу об'єктів. Така класифікація, шляхом порівняння аналізованого об'єкта з сімейством, але представленим у формі одного узагальненого вектора, дає можливість суттєво підвищити швидкодію завдань аналізу структур даних. Для цього необхідно створювати ієрархічні формати структур даних, орієнтовані на компактне представлення спеціальним чином закодованих об'єктів. При поданні об'єкта кіберпростору сукупністю теоретико-множинних або кубітних змінних структура вектора ділиться на сегменти, відповідні кубітів. Кубітна змінна (кубіт) – сукупність n двійкових розрядів, необхідних для унітарного кодування n примітивів і булеана породжених символів. Форми подання вектора кубітних змінних: символьна і/або кубітно-двійкова, орієнтовані на паралельне виконання теоретико-множинних операцій $(\cap,\cup,\tilde{m})$ за допомогою алгебри векторної логіки $(\wedge,\vee,\overline{m})$. Приклади таких операцій у згаданих форматах мають вигляд:

m =	Y	A	B	S	P	L	E	Q
A =	H	S	J	L	E	L	F	C
∩ =	H	Q	J	S	E	L	∅	∅
m =	1111	1100	0011	1001	0110	1101	0100	1000
A =	0010	1001	0001	1101	0100	1101	1011	0111
∨ =	0010	1000	0001	1001	0100	1101	0000	0000
m =	Y	A	B	S	P	L	E	Q
$\tilde{m}$ =	∅	B	A	P	S	H	F	C
m =	1111	1100	0011	1001	0110	1101	0100	1000
$\overline{m}$ =	0000	0011	1100	0110	1001	0010	1011	0111

При аналізі кубітно-двійкових форм зображення об'єктів з метою визначення відстаней між ними необхідно враховувати: 1) Кодова відстань формується за наявністю хоча б одного кубіта, рівного нулю за всіма його координатами. 2) В іншому випадку обчислюються функції приналежності на підставі підрахунку загального числа одиниць, отриманого при виконанні векторної операції кон'юнкції, віднесених до кількості одиниць кожного з векторів, що відповідають двом різним об'єктам кіберпростору. 3) Xor-сума відстаней об'єктів, що становлять цикл, дорівнює вектору, складеному з нульових кубітів. 4) Xor-сума всіх примітивів кубіта дорівнює вектору, який має всі одиничні координати. 5) Формування багатозначних сигнатур на основі кубітних структур даних може істотно розширити сферу застосування апарату xor-поліномів з нелінійними зворотними зв'язками. 6) Неструктурована множина примітивів, що самоорганізується в процесі моделювання або вирішення конкретного завдання істотно зменшує обсяг моделей і час їх створення. 7) Реалізація дерева класифікації і процедур його аналізу значно скорочує обсяг структур даних, а також час вирішення відповідних завдань. Приклад такого дерева зображено на рис. 7.2, яке, завдяки бінарності, виконує класифікацію (спуск по дереву) за мінімальне число кроків обчислювальної процедури.

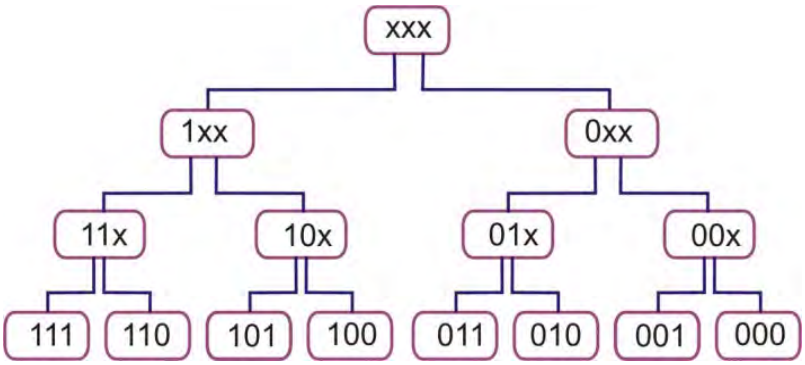


Рисунок 7.2 – Приклад класифікаційного бінарного дерева

Процедура класифікації: 1) Аналіз і-го розряду вхідного вектора m:

$$P = \begin{cases} P^0 \leftarrow m_i \oplus A_i = 0; \\ P^1 \leftarrow m_i \oplus A_i = 1. \end{cases}$$

для вибору лівої чи правої гілки вершини дерева. Тут множина А визначає узагальнені коди-сигнатури, а також кінцеві вершини дерева. 2) Аналіз закінчується позитивно, якщо оброблені всі розряди вхідного вектора, який ідентифікований існуючим аналогом в бібліотеці. В іншому випадку об'єкт не може бути ідентифікований в межах системи, яка повинна бути розширена. 3) Якщо результат аналізу має неоднозначність по відношенню до 0 і 1, то об'єкт ідентифікується вже не примітивом, а класом (підкласом). Час виконання процедури класифікації визначається виразом: $T = \log_2 N$, що є заслугою надлишкових вершин, які дозволяють систему з N відношень (нижній рівень кодів) подати у вигляді дерева.

Таким чином, запропонована модифікована модель критерію скалярної і векторної якості оцінювання бінарних відношень, яка відрізняється використанням функції неналежність і кодової відстані Хемінга, що забезпечує лінійність зміни чисельного значення критерію від 0 до 1 в міру збільшення «відстані» від повного збігу двох об'єктів до максимально можливого, коли кодова відстань дорівнює $d(m, A) = k.$ Критерій може бути використаний при оцінюванні взаємодії об'єктів у реальному масштабі часу в задачах тестування, діагностування функціональних порушень, уразливостей.

7.3 Дедуктивний метод пошуку уразливостей в КС

Основна ідея дедуктивного методу полягає в аналізі зіставлення вхідних і вихідних даних кіберсистеми з метою виявити деструктивні проникнення або уразливості шляхом виконання процедур порівняння між

наперед штатними (функціональними) режимами і ситуаціями, що викликають підозру. Для імплементації методу в інфраструктуру захисних сервісів необхідно мати графову модель логіки функціонування кіберсистеми, яка досить просто може бути трансформована до системи логічних рівнянь, придатної для дедуктивного аналізу. Далі пропонується модель дедуктивно-паралельного синхронного аналізу вразливостей (проникнень) кіберсистеми (об'єкта), яка дозволяє за одну ітерацію обробки структури обчислити всі деструктиви, що перевіряються на тест-векторі. Мета дедуктивного аналізу - визначити якість синтезованого тесту відносно повноти покриття їм вразливостей, а також побудувати таблицю перевірки тестовими наборами усіх виявлених вразливостей КС для виконання процедур діагностування. Така модель заснована на рішенні рівняння:

$$L = T \oplus F, \tag{7.1}$$

Де $F = (F_{m+1}, F_{m+2}, ..., F_i, ...F_n)\,(i = \overline{m+1, n})$ – сукупність функцій справної (коректної) поведінки КС; m – число входів; $Y_i = F_i(X_{i1}, ..., X_{ij}, ..., X_{in_i})$ – n_i-входовий і-й елемент схеми, що реалізує F_i для визначення стану лінії (виходу) Y_i на тест-векторі T_t; тут X_{ij} – j-й вхід і-го елементу; тест $T = (T_1, T_2, ..., T_t, ..., T_k)$ – упорядкована сукупність двійкових векторів, довизначення у процесі справного моделювання на множині вхідних, внутішніх та вихідних ліній, що об’єднана в матрицю

$$T = [T_{ti}] = \begin{bmatrix} T_{11}, T_{12}, ..., T_{1i}, ..., T_{1n} \\ ..... \\ T_{t1}, T_{t2}, ..., T_{ti}, ..., T_{tn} \\ ..... \\ T_{k1}, T_{k2}, ..., T_{ki}, ..., T_{kn} \end{bmatrix}, \tag{7.2}$$

невхідна координата якої визначається моделюванням функції $T_{ti} = Y_i = F_i(X_{i1},...,X_{ij},...,X_{in_i})$ на тест-векторі T_t; $L = (L_1, L_2,..., L_t,..., L_k)$ – множина дедуктивних схем або моделей, що визначаються выразом (7.1), де $L_t = (L_{t1}, L_{t2},..., L_{ti},..., L_{tn})$;

$$L_{ti} = T_t \oplus F_i \tag{7.3}$$

– дедуктивна функція (ДФ) паралельного моделювання несправностей на тест-векторі T_t, що відповідає справному елементу F_i, яка дає можливість обчислювати список вхідних проникнень, транспортованих на вихід елемента F_i [100].

Поняття синхронності введеної моделі (7.1) визначається умовою: $\Delta t = (t_{j+1} - t_j) >> \tau >> \tau_i$, коли інтервал часу між зміною вхідних наборів $(t_{j+1} - t_j)$, які подаються на КС, набагато більше максимальної затримки системи τ і елементу τ_i. Це дозволяє виключити час як несуттєвий параметр [8], що використовується в технологіях моделювання та синтезу тестів.

У загальному випадку, коли функція КС представлена таблицею істинності, застосування формули (7.1) дозволяє отримати для заданого тест-вектора T_t таблицю транспортування вразливостей (проникнень), за якою можна записати ДФ моделювання деструктиву. Приклади отримання таких функцій подані у наступному вигляді (перший доданок – тест-вектор, другий і результат – таблиці істинності та транспортування вразливостей):

X_1	X_2	Y_1
0	1	0

$\oplus$

X_1	X_2	Y_1
0	0	0
0	1	0
1	0	1
1	1	1

$=$

X_1	X_2	L_1
0	1	0
0	0	0
1	1	1
1	0	1

$L_1 = X_1X_2 \vee X_1\overline{X}_2;$

X_1	X_2	Y_2
1	1	1

$\oplus$

X_1	X_2	Y_2
0	0	0
0	1	0
1	0	0
1	1	1

$=$

X_1	X_2	L_2
1	1	1
1	0	1
0	1	1
0	0	0

$L_2 = X_1X_2 \vee X_1\overline{X}_2 \vee \overline{X}_1X_2.$

Тут дедуктивні функції L_1, L_2 записані у вигляді диз'юнктивної нормальної форми за констітуентами одиниці таблиць транспортування деструктивів.

З урахуванням розбиття тесту на складові вектори рівняння (7.1) отримання ДФ для $T_t \in T$ набуває наступного вигляду: $L_t = T_t \oplus F$. Якщо функціональний опис КС зображено компонентами (примітивами), що формують стани всіх ліній (з'єднань) КС, то як формули перетворення справної моделі примітиву F_i на тест-векторі T_t в дедуктивну функцію L_{ti} постає наступний вираз:

$$L_{ti} = T_t \oplus F_i = f_{ti}[(X_{i1} \oplus T_{t1}),(X_{i2} \oplus T_{t2}),...,(X_{ij} \oplus T_{tj}),...,(X_{in_i} \oplus T_{tn_i})] \oplus T_{ti}, \quad (7.4)$$

який є основою дедуктивного аналізу деструктивних порушень КС [95, 98].

Приклад. Отримати дедуктивні функції паралельного моделювання вразливостей на вичерпному тесті для базису функціональних елементів And, Or, Not. З урахуванням виразу (7.4) виконуються наступні перетворення для функції And:

$$L_{and}[T=(00,01,10,11),F=(X_1\wedge X_2)]=$$
$$=L\{(\bar{x}_1\bar{x}_2\vee\bar{x}_1x_2\vee x_1\bar{x}_2\vee x_1x_2)\wedge[(X_1\oplus T_{t1}\wedge X_2\oplus T_{t2})\oplus T_{t3})]\}=$$
$$=(\bar{x}_1\bar{x}_2)\{[(X_1\oplus 0)\wedge(X_2\oplus 0)]\oplus 0\}\vee(\bar{x}_1x_2)\{[(X_1\oplus 0)\wedge(X_2\oplus 1)]\oplus 0\}\vee$$
$$\vee(x_1\bar{x}_2)\{[(X_1\oplus 1)\wedge(X_2\oplus 0)]\oplus 0\}\vee(x_1x_2)\{[(X_1\oplus 1)\wedge(X_2\oplus 1)]\oplus 1\}=$$
$$=(\bar{x}_1\bar{x}_2)(X_1\wedge X_2)\vee(\bar{x}_1x_2)(X_1\wedge\bar{X}_2)\vee(x_1\bar{x}_2)(\bar{X}_1\wedge X_2)\vee(x_1x_2)(X_1\vee X_2).$$

Аналогічно виконуються обчислення для функції Or:

$$L_{or}[T=(00,01,10,11),F=(X_1\vee X_2)]=$$
$$=L\{(\bar{x}_1\bar{x}_2\vee\bar{x}_1x_2\vee x_1\bar{x}_2\vee x_1x_2)\wedge[(X_1\oplus T_{t1}\vee X_2\oplus T_{t2})\oplus T_{t3})]\}=$$
$$=(\bar{x}_1\bar{x}_2)\{[(X_1\oplus 0)\vee(X_2\oplus 0)]\oplus 0\}\vee(\bar{x}_1x_2)\{[(X_1\oplus 0)\vee(X_2\oplus 1)]\oplus 1\}\vee$$
$$\vee(x_1\bar{x}_2)\{[(X_1\oplus 1)\vee(X_2\oplus 0)]\oplus 1\}\vee(x_1x_2)\{[(X_1\oplus 1)\vee(X_2\oplus 1)]\oplus 1\}=$$
$$=(\bar{x}_1\bar{x}_2)(X_1\vee X_2)\vee(\bar{x}_1x_2)(\bar{X}_1\wedge X_2)\vee(x_1\bar{x}_2)(X_1\wedge\bar{X}_2)\vee(x_1x_2)(X_1\wedge X_2).$$

Тут $T_t=(T_{t1},T_{t2},T_{t3}),(t=\overline{1,4})$ – тест-вектор, що має 3 координати, де остання з них визначає стан виходуа двохвходового елементуа And (Or). У наступному перетворенні $T_t=(T_{t1},T_{t2}),(t=\overline{1,2})$ – тест-вектор, який має 2 координати, де друга – стан виходу інвертора:

$$L_{not}[T=(0,1),F=\bar{X}_1]=L\{(\bar{x}_1\vee x_1)[(\overline{X_1\oplus T_{t1}})\oplus T_{t2}]\}=$$
$$=\bar{x}_1[(\overline{X_1\oplus 0})\oplus 1]\vee x_1[(\overline{X_1\oplus 1})\oplus 0]=\bar{x}_1\bar{\bar{X}}_1\vee x_1\bar{\bar{X}}_1=\bar{x}_1X_1\vee x_1X_1.$$

Останній вираз ілюструє інваріантність інверсії до вхідного набору для транспортування вразливостей. Вона трансформується в повторювач. Тому дана функція не фігурує на виходах дедуктивних елементів. Спільна апаратурна реалізація ДФ для решти двухвходових елементів And, Or на вичерпному тесті подана універсальним функціональним примітивом (рис. 7.4) дедуктивно-паралельного аналізу несправностей.

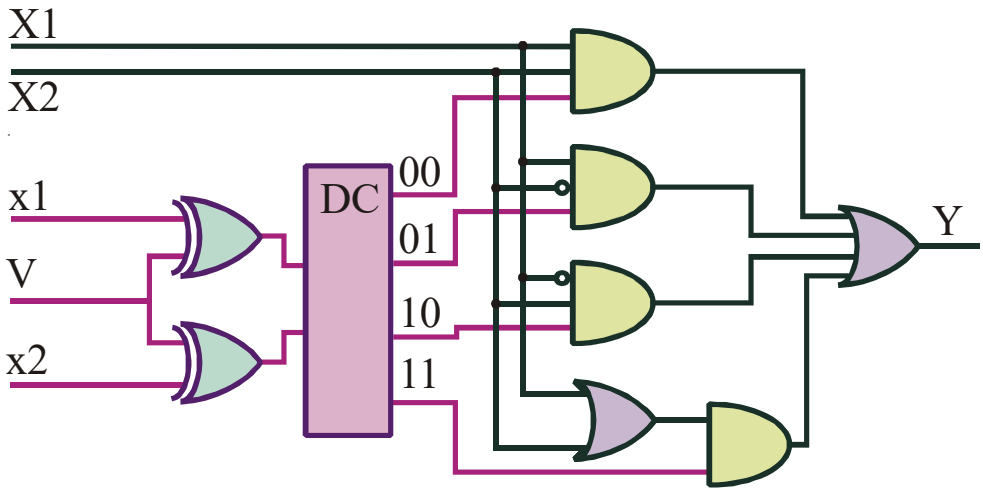


Рисунок 7.4 – Симулятор несправних примітивів

У симуляторі зображені булеві (x1, x2) і регістрові (X1, X2) для кодування вразливостей входи, змінна вибору типу справної функції (AND, OR), вихідна реєстрова змінна Y. Стани двійкових входів x1, x2 і змінна вибору елемента визначають одну з чотирьох дедуктивних функцій для отримання вектора Y перевіряються несправностей.

Для ілюстрації паралельного моделювання вхідних 8-розрядних векторів вразливостей з метою отримання на виході Y множини перевірюваних деструктивів для логічних елементів 2And, 2Or використовується наступна таблиця:

(V, x1, x2) =	000	100	011	111	010	110
X1(RG)	01110001	01110001	10110110	00111011	00101010	10111001
X2(RG)	01111000	01111000	10110101	00110100	10111001	00101010
Y(RG)	01110000	01111001	10110111	00110000	10010001	10010001

Застосування такого симулятора дає можливість трансформувати функціональну модель F коректної поведінки КС в дедуктивну L, яка інваріантна у сенсі універсальності тестовим наборам і не припускає в процесі моделювання використовувати модель F. Тому симулятор, як апаратна модель ДФ, є ефективним двигуном дедуктивно-паралельного моделювання КС, що підвищує швидкодію аналізу кіберсистем в 10 - 1000 разів у порівнянні з програмною реалізацією. Але при цьому співвідношення

обсягів моделей коректного моделювання та аналізу вразливостей становить 1:10. Підхід апаратного аналізу деструктиву спрямований на розширення функціональних можливостей вбудованих засобів моделювання, які можна зберігати на хмарі і постійно ними користуватися для верифікації інфраструктури захисту КС. Обчислювальна складність обробки проекту, що складається з n компонентів, дорівнює $Q = (2n^2\tau)/W$, где τ – час виконання регістрової операції (And, Or, Not); W - розрядність регістра.

Для апаратної реалізації дедуктивно-паралельного моделювання на основі запропонованого симулятора може бути використана обчислювальна структура, що зображена на рис. 7.5. Особливість схемної реалізації полягає у спільному виконанні двох операцій: однобітових – для емуляції функцій логічних елементів And, Or і паралельної – для обробки багаторозрядних векторів несправностей шляхом виконання операцій логічного множення, заперечення і складання.

Функціональне призначення основних блоків (пам’ять і процесор): 1. $M = [M_{ij}]$ – квадратична матриця моделювання деструктивних проникнень (ДП), де i,j =1,q; q – загальне число ліній в оброблюваної КС. 2. Вектори збереження станів коректного моделювання, визначені в моменти часу t-1 і t, необхідні для формування дедуктивних функцій примітивів. 3. Модуль пам'яті для зберігання опису КС у вигляді структури логічних елементів. 4. Буферні регістри, розмірністю q, для зберігання операндів і виконання регістрових паралельних операцій над векторами ДП, що зчитані з матриці M. 5. Блок коректного моделювання для визначення двійкового стану виходу чергового оброблюваного логічного елемента. 6. Дедуктивно-паралельний симулятор, що обробляє за один такт дві регістрових змінних X1, X2 з метою визначення вектора ДП, що транспортуються на вихід логічного елемента Y.

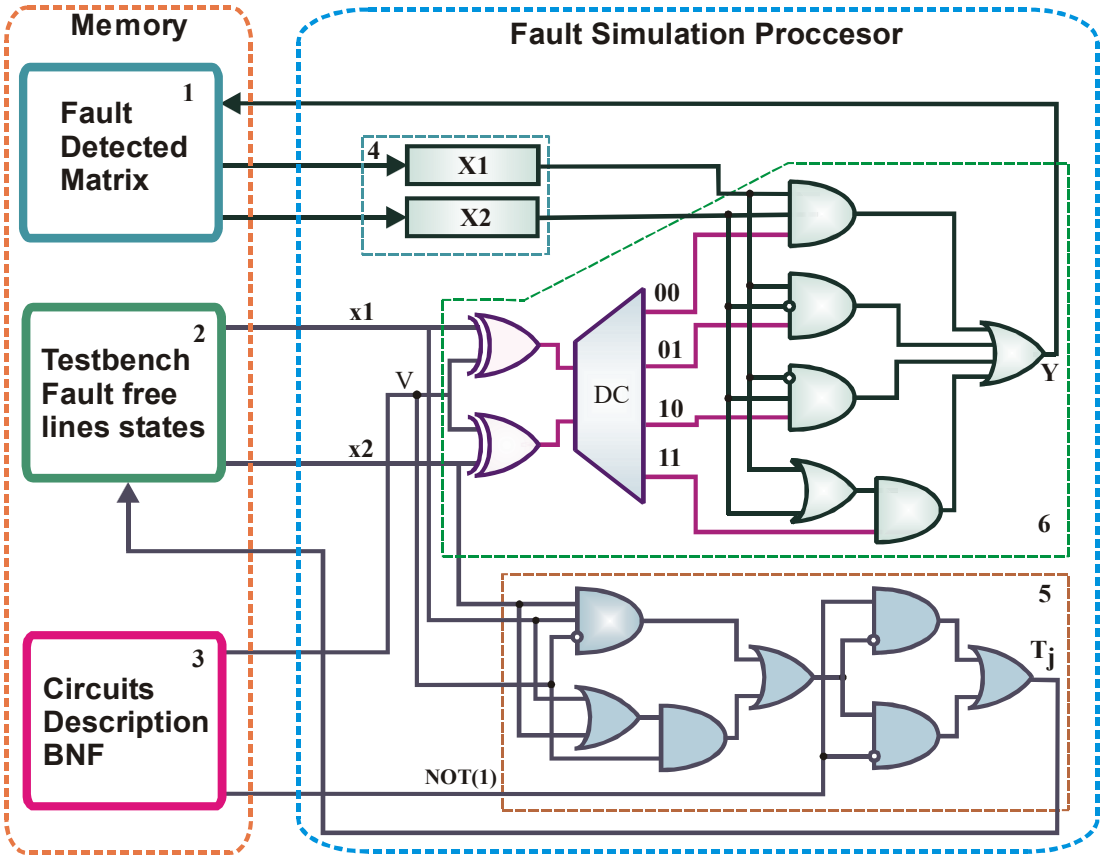


Рисунок 7.5 – HVS-структура апаратного моделювання

Перевага запропонованої структури моделювання ДП. 1. Суттєве зменшення кількості модельованих ДП, обумовлених тільки числом збіжних розгалужень, яке становить до 20% від загального числа ліній. 2. Зниження обсягу пам'яті, необхідного для зберігання матриці модельованих ДП. 3. Простота реалізації Hardware Vulnerability Simulator (HVS) в апаратному виконанні, що дозволяє на порядок збільшити швидкодію моделювання ДП. 4. Використання HVS як першої фази дедуктивно-топологічного методу, який ґрунтується на результаті обробки збіжних розгалужень для швидкодіючого аналізу деревовидних структур.

Маршрут моделювання КС з попередніми розбиттям моделі пристрою на дві структурні організації (збіжні розгалуження і деревовидні підграфи) зображено на рис. 7.6.

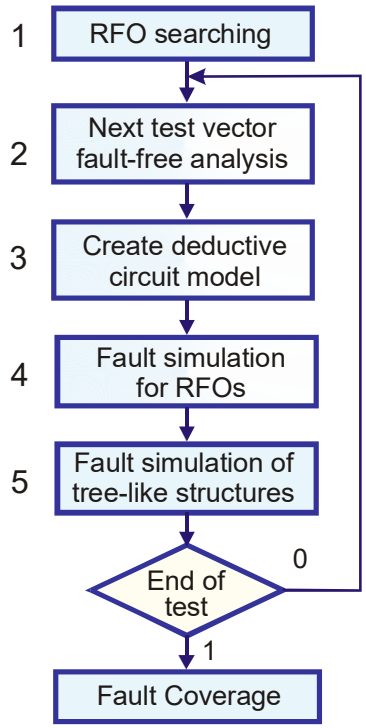


Рисунок 7.6 – Модель процесу дедуктивно-паралельного моделювання

Підсумки запропонованої технології моделювання з попереднім розбиттям КС на збіжні розгалуження і деревовидні підграфи. Дедуктивно-паралельний аналіз ДП на основі їх зворотного простежування несправностей вимагає практично лінійних витрат пам'яті і часу, що залежать від числа ліній КС. Витрати часу для обробки збіжних розгалужень мають квадратичну залежність від їх числа: $Q = (r^2 / W) + n_r + n_p + (n - r - r^0)$. Тут (r^2 / W) – час моделювання ДП r збіжних розгалужень, число яких визначається как $r = 0.2 \times n$; $n_r = n$ – час реконфігурування примітивів схеми на вхідном наборі; $n_p = n$ – час пошуку підграфів ліній, що відповідають неперівірюваним збіжним розгалуженням; $(n - r - r^0) = n - 0.2 \times n - 0.4 \times n = 0.4 \times n$ – час виконання суперпозиції рішень на множині ліній схеми без збіжних розгалужень і попередників для неперевірюваних збіжних розгалужень. Враховуючи фактичні значення параметрів у функції від числа ліній, можна отримати оцінку швидкодії дедуктивно-паралельного методу [94, 98, 99]:

$$Q = [(0.2 \times n)^2 / W] + n + n + (n - 0.2 \times n - 0.4 \times n) = [(0.2 \times n)^2 / W] + 2.4 \times n).$$

Таким чином, виграш за швидкодією запропонованого методу тим більше, чим менше відсоток збіжних розгалужень у схемі КС.

Для порівняння паралельний алгоритм має обчислювальну складність ність Ср, яка визначається функціональною залежністю від числа нееквівалентних ДП (b), довжини комп'ютерного слова (W), кількості еквівалентних вентилів (G): $C_p = (b^2 / W) \times G^3$. Дедуктивний алгоритм має відміни у формулі оцінки швидкодії: $C_d = b^2 \times Q \times G^2 \big|_{Q=G} = b^2 G^3$, де Q – середнє число активізованих ДП вентилей. Дедуктивно-паралельный метод без розбиття схеми має швидкодію, що визначається виразом: $C_{dp} = G^2 + (b^2 / W) \times G^2$. Перший доданок задає час справного моделювання, другий – час аналізу ДП, лінії якого не ранжовані. Для комбінаційної ранжированої схеми швидкодія методу має оцінку $C^r_{dp} = G + (b^2 / W) \times G.$ Швидкодія дедуктивно-паралельного методу вище паралельного і дедуктивного $(C^r_{dp} << \{C_p, C_d\})$, завдяки розділенню фаз справного та несправного моделювань.

Запропонована технологія програмно-апаратного дедуктивно-паралельного моделювання ДП орієнтована на створення моделей дедуктивних примітивів компонентів і зв'язків КС з метою тестування вразливостей (проникнень). Представлена структурна модель апаратного симулятора і пристрою моделювання в цілому, які орієнтовані на істотне підвищення швидкодії засобів моделювання КС великої розмірності шляхом розподілу функцій коректного аналізу та обчислення списків перевірюваних вразливостей на тестових наборах.

Метод дедуктивно-паралельного моделювання дає можливість оцінювати якість (повноту) запропонованих тестів, а також визначати всі потенційно можливі місця існування вразливостей з метою їх подальшого усунення.

7.4 Модель процесів тестування уразливостей і проникнень

Пропонуються технологічні та ефективні процес-моделі і методи діагностування вразливостей або функціональних порушень в програмних та/або апаратних компонентах кіберсистеми. Використовуються регістрові (векторні) або матричні (табличні) структури даних, які орієнтовані на паралельне виконання логічних операцій при пошуку вразливих компонентів КС.

Проблема синтезу або аналізу компонентів довільної системи та інфраструктури сервісів може бути сформульована у вигляді взаємодії (симетричної різниці – аналог xor-операції на булеані) в кібернетичному просторі її моделі F з вхідними впливами T і реакціями L:

$$f(F,T,L) = \varnothing \rightarrow F \Delta T \Delta L = \varnothing .$$

Кіберпростір – сукупність взаємодіючих за метрикою інформаційних процесів і явищ, що використовують як носія комп'ютерні системи та мережі. Зокрема, компонент простору представлений k-мірним (кортежем) вектором $a = (a_1, a_2, ..., a_j, ..., a_k)$, $a_j = \{0,1\}$ у двійковому алфавиті. Нуль-вектор є k-мірний кортеж, всі координаты якого дорівнюють нулю: $a_j = 0, j = \overline{1,k}$.

Метрика β кібернетичного (двійкового) простору визначається ріністю, яка формує нуль-вектор для xor-суми відстаней d_i між ненульовим і кінцевим числом точок (об’єктів), що замкнені у цикл:

$$\beta = \bigoplus_{i=1}^{n} d_i = 0.$$

Инакше: метрика β векторного логічного двійкового простору дорівнює нуль-вектору xor-суми відстаней між кінцевим числом точок

(вершин) графа, що утворюють цикл. Сума n-мірних двійкових векторів, які задають координати точок циклу дорівнює нуль-вектору. Дане визначення метрики оперує відносинами, що дозволяє скоротити систему аксіом з трьох до однієї і поширити її дію на будь-які конструкції n-мірного кіберпростору.

Метрика β кібернетичного багатозначного простору, де кожна координата вектору (об'єкту) визначена в алфавіті, що складає булеан на універсумі примітивів потужності p: $a_j = \{\alpha_1, \alpha_2, ..., \alpha_r, ..., \alpha_m\}, m = 2^p$, є дорівняна $\varnothing$-вектору (за всіма координатами) симетрична різниця відстаней між кінцевою кількістю точок, які утворюють цикл:

$$\beta = \underset{i=1}{\overset{n}{\Delta}} d_i = \varnothing.$$

Рівність пустому вектору симетричної різниці покоординатної теоретико-множинної взаємодії підкреслює рівнозначність компонентів (відстаней), що формують рівняння.

Введена метрика становить не тільки теоретичний інтерес, але має і практичну спрямованість на узагальнення і класифікацію задач тестування шляхом створення моделі xor-відносин на множині з чотирьох основних компонентів. Процедури синтезу тестів, моделювання вразливостей і їх діагностування можна звести до xor-відносинам на графі (рис. 7.7) повної взаємодії чотирьох вершин (функціональність, кіберсистеми (Unit Under Test), тест, уразливості) $G = \{F, U, T, L\}$.

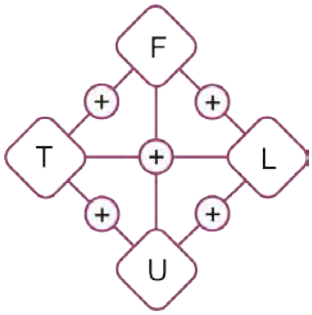


Рисунок 7.7 – Граф взаємодії компонентів тестування

Такий граф породжує чотири базових трикутники, які формують 12 практично корисних тріад відносин, формулюють завдання тестування:

$T \oplus F \oplus L = 0$	$T \oplus L \oplus U = 0$	$T \oplus F \oplus U = 0$	$F \oplus L \oplus U = 0$
1) $T = F \oplus L$	4) $T = L \oplus U$	7) $T = F \oplus U$	10) $F = L \oplus U$
2) $F = T \oplus L$	5) $L = T \oplus U$	8) $F = T \oplus U$	11) $L = F \oplus U$
3) $L = T \oplus F$	6) $U = T \oplus L$	9) $U = T \oplus F$	12) $U = F \oplus L$

Введення вершини U в граф взаємодії компонентів тестування розширює функціональні можливості моделі, з'являються нові властивості отриманої системи. Введення в структуру нової вершини повинно мати вагомі аргументи на користь її доцільності. Що стосується представленого на рис. 7.7 графа, змістовно все формульно описані задачі можна класифікувати в групи наступним чином.

Група 1 – теоретичні експерименти (на моделі функціональності), без кіберсистеми: 1) Синтез тесту за моделлю функціональності для заданого списку вразливостей. 2) Побудова моделі функціональності на основі заданого тесту і списку вразливостей. 3) Моделювання вразливостей функціональності на заданому тесті.

Група 2 – реальні експерименти (на кіберсистемі), без моделі функціональності: 4) Синтез тесту шляхом фізичної емуляції вразливостей в КС. 5) Визначення списку вразливостей пристрою при виконанні діагностичного експерименту. 6) Верифікація тесту і вразливостей в експерименті на реальній КС.

Група 3 – тестові експерименти (верифікація), без вразливостей: 7) Синтез тесту шляхом порівняння результатів моделювання моделі і реальної КС. 8) Синтез функціональності за реальною КС і заданим тестом. 9) Верифікація тесту і моделі функціональності відносно реальної КС з існуючими уразливостями.

Група 4 – експерименти в процесі функціонування, на робочих впливах: 10) Перевірка правильності поведінки реальної КС на існуючих або заданих уразливостях. 11) Перевірка працездатності пристрою відносно існуючої моделі в процесі функціонування. 12) Верифікація функціональності і списку вразливостей відносно поведінки реального КС.

Найбільш популярними задачами з перерахованого вище списку є: 1, 3, 5, 8, 9. Можна ввести й іншу класифікацію типів задач, яка дає можливість визначити на графі $G = (F, U, T, L)$ всі концептуальні шляхи вирішення цільових проблем: синтезу тестів, визначення моделі функціональності, генерування моделі вразливостей і проектування КС:

$$
\begin{array}{llll}
1)\ T = F \oplus L; & 4)\ F = T \oplus L; & 7)\ L = T \oplus F; & 10)\ U = T \oplus L; \\
2)\ T = U \oplus L; & 5)\ F = U \oplus L; & 8)\ L = T \oplus U; & 11)\ U = T \oplus F; \\
3)\ T = F \oplus U; & 6)\ F = T \oplus U; & 9)\ L = F \oplus U; & 12)\ U = F \oplus L.
\end{array}
$$

Всі конструкції, подані у відносинах, мають чудову властивість оборотності. Компонент, який вираховується за допомогою двох інших, може бути використаний як аргумент для визначення будь-якого з двох вихідних. Тому тут можна говорити про транзитивну оборотність кожної тріади відносин на повному графі, коли за двома будь-якими компонентами завжди і однозначно можна відновити або визначити третій. При цьому формат подання кожного компонента повинен бути однаковим за формою і розмірністю (вектори, матриці). На основі запропонованої метрики і моделей тестування далі розглянуті більш докладно методи діагностування вразливостей або функціональних порушень.

7.5 Граф-метод пошуку функціональних порушень в КС

Використовується рівняння простору

$$f(F,T,L,U) = 0 \rightarrow F \oplus T \oplus L \oplus U = 0,$$

яке трансформується до вигляду $L = (T \oplus F) \oplus (T \oplus U)$. Діагностування уразливостей (функціональних порушень) зводиться до порівняння результатів модельного $(T \oplus F)$ і натурного $(T \oplus U)$ експериментів, який формує список функціональних порушень L, присутніх в об'єкті діагностування. Формула-модель процесу пошуку блоку F_i з функціональними порушеннями зводиться до вибору pрозв'язку шляхом визначення xor-взаємодії між трьома компонентами:

$$L = F_i \leftarrow [(T \oplus F_i) \overset{p}{\underset{i=1}{\oplus}} (T \oplus U_i)] = 0.$$

Аналітична модель верифікації HDL-коду з використанням механізму ассерцій (додаткових ліній спостереження) орієнтована на досягнення заданої глибини діагностування і подана у наступному вигляді:

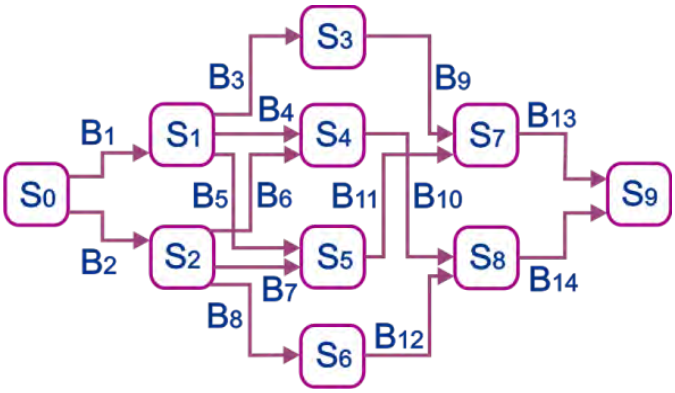
$$\begin{array}{ll} M = f(F,A,B,S,T,L), & F = (A*B)\times S;\ S = f(T,B); \\ A = \{A_1,A_2,...,A_i,...,A_n\}; & B = \{B_1,B_2,...,B_i,...,B_n\}; \\ S = \{S_1,S_2,...,S_i,...,S_m\}; & S_i = \{S_{i1},S_{i2},...,S_{ij},...,S_{ip}\}; \\ T = \{T_1,T_2,...,T_i,...,T_k\}; & L = \{L_1,L_2,...,L_i,...,L_n\}. \end{array}$$

Тут $F = (A*B)\times S$ – функциональность, представленная графом (рис. 7.8) транзакцій програмних блоків (Code-Flow Transaction Graph – CFTG), де $S = \{S_1,S_2,...,S_i,...,S_m\}$ – вершини або стани програмного продукту при моделюванні тестових сегментів. Інакше граф можна ідентифікувати як

АВС-граф. Кожний стан $S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\}$ визначається значеннями істотних змінних КС (змінні, пам'ять, термінали, комп'ютери). Орієнтовані дуги графа зображені сукупністю функціональних блоків:

$$B = (B_1, B_2, ..., B_i, ..., B_n), \ \bigcup_{i=1}^{n} B_i = B; \ \bigcap_{i=1}^{n} B_i = \varnothing,$$

де кожному з них може бути поставлена у відповідність асерція $A_i \in A = \{A_1, A_2, ..., A_i, ..., A_n\}$. Кожна дуга B_i – група операторів коду – формує стан вершини $S_i = f(T, B_i)$ у залежності від тесту $T = \{T_1, T_2, ..., T_i, ..., T_k\}$. Кожній вершині може бути поставлений асерційний монітор, що поєднує асерції вхідних у вершину дуг $A(S_i) = A_{i1} \vee A_{i2} \vee ... \vee A_{ij} \vee ... \vee A_{in}$. Вершина може мати більш однієї вхідяної (вихідної) дуги. Множество блоків з функціональними порушеннями подано списком $L = \{L_1, L_2, ..., L_i, ..., L_n\}$.



$$B = (B_1B_3B_9 \vee (B_2B_7 \vee B_1B_5)B_{11})B_{13} \vee$$
$$\vee ((B_1B_4 \vee B_2B_6)B_{10} \vee B_2B_8B_{12})B_{14} =$$
$$= B_1B_3B_9B_{13} \vee B_2B_7B_{11}B_{13} \vee B_1B_5B_{11}B_{13} \vee$$
$$\vee B_1B_4B_{10}B_{14} \vee B_2B_6B_{10}B_{14} \vee B_2B_8B_{12}B_{14}.$$

Рисунок 7.8 – Приклад АВС-графа для HDL-кода

Модель HDL-коду, подана у формі АВС-графа, відображає не тільки структуру програмного коду, а й тестові зрізи функціональних покриттів, що формуються за допомогою програмних блоків, які входять у розглянуту вершину. Остання визначає відношення між досягнутим на тесті простором змінних і потенційно можливим, яке формує функціональне покриття як потужність стану і-вершини графа $Q = \mathrm{card}C_i^r / \mathrm{card}C_i^p$. У сукупності всі вершини графа повинні складати повне покриття простору станів змінних КС, який формирує якість тесту, рівну 1 (100%): $Q = \mathrm{card}\bigcup_{i=1}^{m} C_i^r / \mathrm{card}\bigcup_{i=1}^{m} C_i^p = 1$. Крім того, механізм асерцій $<A, C>$, існуючий на графі, дозволяє виконати моніторинг дуг (function-coverage) $A = \{A_1, A_2, ..., A_i, ..., A_n\}$ і вершин (state-coverage) $C = \{C_1, C_2, ..., C_i, ..., C_m\}$. Ассерції на дугах відповідають за діагностування функціональних порушень в КС блоках. Асерції на вершинах графа дають додаткову інформацію про якість тестів (асерцій) з метою їх поліпшення або доповнення. Транзакційний граф КС блоків дає можливість: 1) використовувати апарат тестопридатного проектування для оцінки якості КС; 2) оцінити витрати на створення тестів, діагностування та виправлення функціональних порушень; 3) оптимізувати синтез тесту шляхом вирішення задачі покриття мінімальною кількістю активізованих шляхів всіх дуг (вершин). Наприклад, мінімальний тест для наведеного АВС-графа має шість сегментів, які активізують всі існуючі шляхи:

$$T = S_0S_1S_3S_7S_9 \vee S_0S_1S_4S_8S_9 \vee S_0S_1S_5S_7S_9 \vee$$
$$\vee S_0S_2S_4S_8S_9 \vee S_0S_2S_5S_7S_9 \vee S_0S_2S_6S_8S_9.$$

Тесту можна поставити у відповідність наступну матрицю активізації програмних блоків:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1

Матриця активізації ілюструє факт нерозрізненості на тесті функціональних порушень в блоках 3 і 9, 8 і 12, які складають два класи еквівалентностей за наявністю однієї асерції (монітора) у вершині 9. Для усунення такої нерозрізненості необхідно створити два додаткових монітора в вершинах 3 і 6. В результаті, 3 ассерції на вершинах $A=(A_3,A_6,A_9)$, дають можливість розрізнити між собою всі блоки програмного коду. Таким чином, граф дозволяє не тільки синтезувати оптимальний тест, але і визначати мінімальне число асерційних моніторів тут і далі в вершинах для пошуку несправних блоків із заданою глибиною діагностування.

Збільшення числа асерційних моніторів призводить до модифікації таблиці активізації. Інакше, на заданому тесті і механізмі асерцій необхідно однозначно вирішувати задачу діагностування функціональних порушень КС з глибиною до функціонального блоку. При цьому число асерцій і тестових сегментів має бути мінімально допустимим для кодової ідентифікації всіх блоків:

$$\left|T\right|+\left|A\right|\geq \log_2\left|B\right| = \mathrm{card}T+\mathrm{card}A\geq \log_2 \mathrm{card}B\,.$$

Спочатку кількість асерційних моніторів дорівнює числу тестових сегментів. Таблиця активізації функціональних модулів дозволяє виконувати ідентифікацію блоків коду з функціональними порушеннями на узагальненому векторі експериментальної перевірки (асерційного моніторингу):

$$V = (V_1, V_2, ..., V_i, ..., V_n), V_i = \{0,1\}, V_i = T_i \oplus B_j, \forall j(B_{ij} = 1).$$

Координата вектора $V_i = T_i \oplus B_j = 1$ идентифікує факт «падіння» тест-сегменту на підмножині активізованих їм блоків. У відповідності з вектором V, заданим на таблиці активізації з урахуванням наведеного вище правила обчислення його координат:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	V
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.	0
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1	1
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.	0
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.	0
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1	1

будується логічна функція функціональних порушень КС, яка спрощується на основі використання координат вектора експериментальної перевірки V:

$$B = (\overline{T}_1 \vee B_1 \vee B_3 \vee B_9 \vee B_{13}) \wedge (\overline{T}_2 \vee B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (\overline{T}_3 \vee B_1 \vee B_5 \vee B_{11} \vee B_{13}) \wedge (\overline{T}_4 \vee B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (\overline{T}_5 \vee B_2 \vee B_7 \vee B_{11} \vee B_{13}) \wedge (\overline{T}_6 \vee B_2 \vee B_8 \vee B_{12} \vee B_{14});$$
$$\{V, T\} = (010101) \rightarrow$$
$$B = (0 \vee B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge (0 \vee B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\vee (0 \vee B_2 \vee B_8 \vee B_{12} \vee B_{14}) =$$
$$= (B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge (B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\vee (B_2 \vee B_8 \vee B_{12} \vee B_{14}) =$$
$$= B_1B_2 \vee B_4B_2 \vee ... \vee B_3B_6B_{12} \vee ... \vee B_{14}.$$

Після перетворення кон'юнктивної нормальної форми (КНФ) до диз'юнктивної нормальної форми отримані терми містять всі можливі розв’язки у вигляді покриття одиничних координат вектора експериментальної перевірки одиночними або кратними функціональними

порушеннями КС блоків. Вибір кращого розв’язку пов'язаний з визначенням терму ДНФ мінімальної довжини. У даному прикладі оптимальним рішенням є терм, що містить один блок $B = B_{14}$, який своїм функціональним порушенням покриває три одиниці у векторі експериментальної перевірки. Даний факт також очевидний з порівняння двох останніх стовпців матриці активізації B.

Інший апаратно орієнтований аналітичний розв’язок пов'язаний з синтезом багатовиходової комбінаційної схеми – дешифратора за матрицею активізації програмних блоків:

$$B_1 = T_1T_2T_3\overline{T}_4\overline{T}_5\overline{T}_6; B_2 = \overline{T}_1\overline{T}_2\overline{T}_3T_4T_5T_6;$$
$$B_3 = \overline{T}_1T_2T_3T_4T_5T_6; \ \ldots B_{14} = \overline{T}_1T_2\overline{T}_3T_4\overline{T}_5T_6.$$

Такий пристрій має число входів, рівне кількості тестових сегментів, а вихідних ліній, – рівне числу КС блоків. При подачі на входи дешифратора вектора експериментальної перевірки формується одиничне значення на одному з його виходів. При цьому номер виходу відповідає блоку, який має функціональні порушення. Така схема становить інтерес для створення замкнутої в цикл інфраструктури тестування і виправлення функціональних порушень, де адреса уразливого блоку може бути використана для його автоматичної заміни на альтернативний модуль з існуючої бібліотеки діверсних рішень.

7.6 Організація кіберфізичного процесу сервісного обслуговування та захисту

Технологічна модель інфраструктури вбудованого тестування, діагностування та відновлення невразливості (рис. 7.9) має три компоненти: 1. Тестування КС (Unit Under Test (UUT)) з використанням еталонної моделі (Model Under Test (MUT)) для формування вектора експериментальної

перевірки m_a, розмірність якого відповідає числу тестових наборів. 2. Пошук дефектів на основі аналізу таблиці вразливостей А. 3. Відновлення невразливості КС за допомогою заміни вразливих блоків на компоненти з Spare Good Primitives.

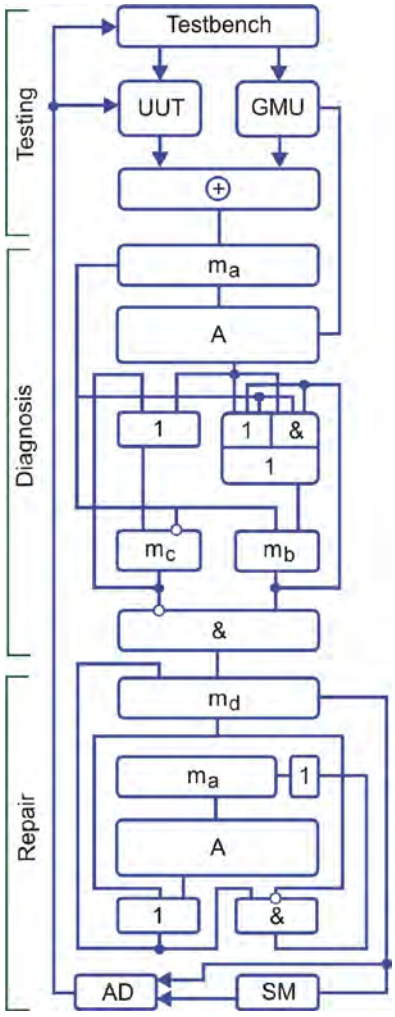


Рисунок 7.9 – Модель вбудованого тестування компонентів КС

Процес-модель вбудованого сервісного обслуговування працює в реальному масштабі часу і дозволяє підтримувати в працездатному стані без втручання людини КС, що є доцільним рішенням у разі застосування технологій, пов'язаних з дистанційною експлуатацією виробу.

Запропоновані процес-моделі аналізу асоціативних таблиць, а також введені критерії якості логічних рішень дозволяють вирішувати завдання квазіоптимального покриття, діагностування вразливостей програмних і (або)

апаратних блоків. Модель векторних обчислень стала основою для розробки спеціалізованої мультипроцесорної архітектури, орієнтованої на пошук, розпізнавання і прийняття рішень на основі використання структур асоціативних таблиць.

Таким чином, представлена на малюнку граф-схема дає можливість ефективно здійснювати сервісне обслуговування як завгодно складної киберсистеми. Переваги такого движка, інваріантного до рівнів ієрархії, полягають у простоті підготовки та подання діагностичної інформації у вигляді мінімізованої таблиці активізації функціональних блоків або деструктивних компонентів (вразливостей) на тестових сегментах.

В останньому випадку ефект (зменшення години) отриманий за рахунок введення додаткової інфраструктури, наведеної вище у розділі 4 на рис. 4.5, до функціональності проекту, що дозволяє вибірково здійснювати тестування и діагностування, а також перепрограмувати окремі модулі в разі фіксації вразливостей. Тут зображені блоки мають зміст відповідно: Testbench – тести перевірки функціональних блоків, FC – функціональне покриття тесту, F – функціональні блоки КС, DI – діагностична інформація, DT – методи та засоби діагностування, DA – результати аналізу процесу діагностування, FB – вразливі функціональні модулі, Repairing – відновлення працездатності функціональних модулів.

Сервісне обслуговування окремої комірки функціональності здійснюється за допомогою осередку граничного сканування, поданої вище у розділі 4 на рис. 4.6.

7.7 Висновки до розділу 7

Розроблено модель, метод і виконана їх програмно-апаратна реалізація, орієнтовані на істотне зменшення часу тестування і витрат пам'яті для зберігання матриці діагностування шляхом формування тернарних відносин (тест - монітор - функціональний компонент) з метою швидкого виявлення уразливостей та їх усунення.

1. Представлена структурна модель відншень на множині з чотирьох основних компонентів тестування і діагностики (функціональність, КС, тест, уразливості), яка характеризується повною xor-взаємодією всіх вершин графа і транзитивною оборотністю кожної тріади відношень, що дозволяє визначити і класифікувати шляхи вирішення практичних задач, включаючи синтез тестів, моделювання та пошук вразливостей.

2. Запропоновано удосконалену процес-модель визначення вразливостей в КС, яка відрізняється використанням xor-операції, що дає можливість підвищити швидкодію діагностування одиночних або кратних ФН на основі паралельного аналізу таблиці ФН за допомогою логічних векторних операцій and, or, xor.

3. Розроблено структурну модель у вигляді мультидерева і метод (движок) його швидкого обходу, які відрізняються інваріантністю до рівнів ієрархії КС для діагностування вразливостей на тестових сегментах, що дає можливість ефективно, в реальному масштабі часу, здійснювати сервісне обслуговування як завгодно складної киберсистеми.

Оскільки методи діагностування вразливостей для КС є одним з найбільш істотних компонентів інфраструктури сервісного обслуговування проектованих і функціонуючих КС, можуть використовуватись векторно-логічні методи діагностування уразливостей [94], основна мета яких – визначення місця причини та виду уразливості на структурі КС шляхом аналізу таблиць вразливостей (функціональних порушень – ФН), побудованих в процесі моделювання всіх можливих деструктивів.

Застосування 3D-методу діагностування уразливостей мотивовано ринковою привабливістю матричного методу пошуку ФН в компонентах КС, як найтехнологічнішого, який орієнтований на паралельну обробку даних, що дає можливість істотно зменшити час діагностичного обслуговування при виникненні вразливостей (проникнень) [94].

8 РЕВЕРСИВНИЙ РЕГІСТР ЗСУВУ

Пропонується опис корисної моделі, яка належить до автоматики і обчислювальної техніки і може бути використана в пристроях діагностування і статистичної обробки інформації.

8.1 Огляд моделей

Відомий реверсивний регістр зсуву (авт. св. СССР №1642527 МПК G11C 19/00, Бюл.14, 1991), що містить n-розрядний регістр з CKIR-тригерів, перший і другий елементи АБО, перший і другий елементи АБО-НІ, перший, другий, третій і четвертий елементи І, третій і четвертий елементи АБО на розряд (в першому розряді тільки третій елемент АБО, в останньому розряді тільки четвертий елемент АБО) для перетворення унітарного двійкового коду в просторовий, комбінаційний двійковий суматор SM кількості одиниць неущільненного двійкового коду, два дешифратори DC1 і DC2 для перетворення позиційного двійкового коду кількості одиниць в унітарний код одиниць, п’ятий і шостий елементи АБО для управління синхронізацією відповідно першого і другого дешифраторів DC1 і DC2, входи обнуління і синхронізації регістра зсуву, прямий і інверсний інформаційні входи та прямий інформаційний вихід реверсивного регістра зсуву при зсуві праворуч, прямий і інверсний інформаційні входи та прямий інформаційний вихід реверсивного регістра зсуву при зсуві ліворуч.

Такий реверсивний регістр зсуву забезпечує зсув коду як ліворуч, так і праворуч за n тактів, ущільнення двійкового коду як ліворуч, так і праворуч за один такт, але має великі апаратні витрати (10 елементів на розряд), що зумовлює велику складність схеми.

Найбільш близьким по сукупності ознак до патенту, що заявляється, є реверсивний регістр зсуву (патент України на корисну модель № 83310 МПК (2013.01) G11C 19/00), що містить n-розрядний регістр стану з CDR-

тригерів, перший елемент АБО, перший і другий елементи І з трьома входами в кожному розряді, другий і третій елементи АБО на розряд (в першому розряді тільки другий елемент АБО, в останньому розряді тільки третій елемент АБО) для перетворення унітарного двійкового коду в просторовий, комбінаційний двійковий суматор SM кількості одиниць неущільненного двійкового коду, два дешифратори DC1 і DC2 для перетворення позиційного двійкового коду кількості одиниць в унітарний код одиниць, четвертий і п’ятий елементи АБО для управління синхронізацією відповідно першого і другого дешифраторів DC1 і DC2, входи обнуління і синхронізації, прямий інформаційний вхід та прямий інформаційний вихід реверсивного регістра зсуву при зсуві праворуч, вхід управління режимом – зсув/ущільнення, входи управління напрямом зсуву – праворуч або ліворуч, прямий інформаційний вхід та прямий інформаційний вихід реверсивного регістра зсуву при зсуві ліворуч.

Такий реверсивний регістр зсуву забезпечує зсув коду як ліворуч, так і праворуч за n тактів, ущільнення двійкового коду як ліворуч, так і праворуч за один такт, але має великі апаратні витрати (6 елементів на розряд) і велику кількість входів в елементи І, що зумовлює велику складність схеми.

8.2 Інноваціна пропозиція

В основу винаходу поставлена задача створення такого реверсивного регістра зсуву , в якому нове схемне рішення дозволило б при збереженні кількості мікрооперацій і швидкодії зменшити його апаратні витрати за рахунок зменшення кількості елементів АБО і кількості входів в логічні елементи І.

Ця задача вирішена наступним чином. В реверсивному регістрі зсуву, який складається з групи n тригерів стану, першого елемента АБО, першого і другого елементів І в кожному розряді, комбінаційного двійкового суматора SM кількості одиниць неущільненного двійкового колу, першого DC1 і

другого дешифраторів DC2 для перетворення позиційного двійкового коду кількості одиниць в унітарний код одиниць, другого елемента АБО в кожному розряді, крім останнього, третього елемента АБО в кожному розряді, крім першого, входів обнуління і синхронізації регістра зсуву з'єднаних відповідно з входами обнуління і синхронізації усіх тригерів стану, прямого інформаційного входу реверсивного регістра зсуву, що з'єднаний з першим входом першого елемента І в першому розряді, перший вхід першого елемента І в кожному розряді, крім першого, з'єднаний з прямим виходом тригера попереднього розряду, прямий вихід тригера стану останнього розряду є виходом реверсивного регістра при зсуві праворуч, входу управління ущільненням коду ліворуч, другі входи перших елементів І в кожному розряді з'єднані з шиною зсув праворуч, другі входи других елементів І в кожному розряді з'єднані з шиною зсув ліворуч , прямого інформаційного входу реверсивного регістра зсуву при зсуві ліворуч, що з'єднаний з першим входом другого елемента І в останньому розряді, перший вхід другого елемента І в кожному розряді, крім останнього, з'єднаний з прямим виходом тригера наступного розряду, входу управління ущільненням коду праворуч, прямий вихід тригера стану першого розряду є виходом реверсивного регістра зсуву при зсуві ліворуч. в якому прямі виходи тригерів усіх розрядів з'єднані з відповідними входами комбінаційного двійкового суматора SM кількості одиниць неущільненного двійкового коду, виходи якого з'єднані з відповідними входами дешифраторів DC1 і DC2 , виходи других схем АБО в кожному розряді, крім останнього, з'єднані відповідно з третім входом першого елемента АБО у даному розряді, другий вхід другого елемента АБО в кожному розряді, крім останнього і передостаннього , з'єднано з виходом другого елемента АБО в наступному розряді, другий вхід другого елемента АБО в передостанньому розряді з'єднано з виходом n дешифратора DC1 і з третім входом першого елемента АБО в останньому розряді, виходи унітарного коду першого дешифратора DC1 з'єднані з відповідним першим входом другого елемента АБО в кожному розряді, крім

останнього, виходи унітарного коду другого дешифратора DC2 з'єднані з відповідним першим входом третього елемента АБО в кожному розряді, крім першого, вихід кількості одиниць n другого дешифратора DC2 з'єднано з четвертим входом першого елемента АБО в першому розряді і з другим входом третього елемента АБО в другому розряді, другий вхід третього елемента АБО в кожному розряді, крім першого і другого, з'єднано з виходом третього елемента АБО в попередньому розряді, виходи першого і другого елементів I в кожному розряді з'єднані відповідно з першим і другим входами першого елемента АБО в цьому розряді, вихід першого елемента АБО в усіх розрядах з'єднано з D-входами CDR-тригерів, згідно з **запропонованим** рішенням використані двовходові перший і другий елементи I в кожному розряді, а інверсні входи управління першого DC1 і другого дешифраторів DC2 з'єднано відповідно з входами ущільнення ліворуч і ущільнення праворуч.

Таким чином, завдяки вилученню третіх входів в перших і других елементах I усіх розрядів, вилученню четвертого і п'ятого елементів АБО та завдяки новим зв'язкам створено реверсивний регістр зсуву зі спрощеною схемою, що дозволяє зменшити його апаратні витрати при збереженні швидкодії і функціональних можливостей.

На рис. 8.1 зображена структурна схема реверсивного регістра зсуву і ущільнення кодів для узагальненого випадку кількості розрядів n .

Реверсивний регістр зсуву містить в кожному розряді CDR-тригер 1, перший 2 елемент АБО, перший 3 і другий 4 елементи I, комбінаційний 5 двійковий суматор SM кількості одиниць неущільненного двійкового коду, перший 6 дешифратор DC1 для перетворення позиційного двійкового коду кількості одиниць в унітарний код при ущільненні ліворуч, другий 7 дешифратор DC2 для перетворення позиційного двійкового коду кількості одиниць в унітарний код при ущільненні праворуч, в усіх розрядах, крім останнього, ланцюг других 8 схем АБО для перетворення унітарного двійкового коду в ущільнений при ущільненні ліворуч, в усіх розрядах, крім

першого, ланцюг третіх 9 схем АБО для перетворення унітарного двійкового коду в ущільнений при ущільненні праворуч, вхід 10 обнуління регістра зсуву, вхід 11 синхронізації регістра зсуву, прямий 12 інформаційний вхід регістра зсуву при зсуві праворуч, прямий 13 інформаційний вихід регістра зсуву при зсуві праворуч, вхід 14 управління режимом роботи реверсивного регістра зсуву зсув/ущільнення ліворуч (1/0), вхід 15 управління зсувом праворуч, вхід 16 управління зсувом ліворуч, прямий 17 інформаційний вхід регістра зсуву при зсуві ліворуч, прямий 18 інформаційний вихід регістра зсуву при зсуві ліворуч, вхід 19 управління режимом роботи реверсивного регістра зсуву зсув/ущільнення праворуч (1/0).

8.3 Робота пристрою

Пристрій працює наступним чином.

В залежності від значення сигналу на вході 15 управління режимом роботи реверсивного регістра зсуву зсув праворуч/ущільнення (1/0) і значення сигналів зсув ліворуч/ущільнення (1/0) на вході 16, зсув/ущільнення ліворуч 14 (1/0), зсув/ущільнення праворуч 19 (1/0), та значення сигналу синхронізації на вході 11 подача сигналу/відсутність сигналу (1/0), регістр працює в чотирьох режимах роботи: зсув послідовного двійкового коду праворуч, зсув послідовного двійкового коду ліворуч, ущільнення двійкового коду праворуч, ущільнення двійкового коду ліворуч.

Заповнення n-розрядного регістра зсуву послідовним кодом виконується за час дії n сигналів зсуву на вході синхронізації 11 і присутності відповідних розрядів послідовного коду від розряду n до n-1,n-2,…,2,1 на послідовному вході 12 реверсивного регістра зсуву і значеннях сигналів 1 і 0 відповідно на входах зсув праворуч 15 і зсув ліворуч 16, і значенні сигналу 1 на входах зсув/ущільнення ліворуч 14 і зсув/ущільнення праворуч 19 відповідно.

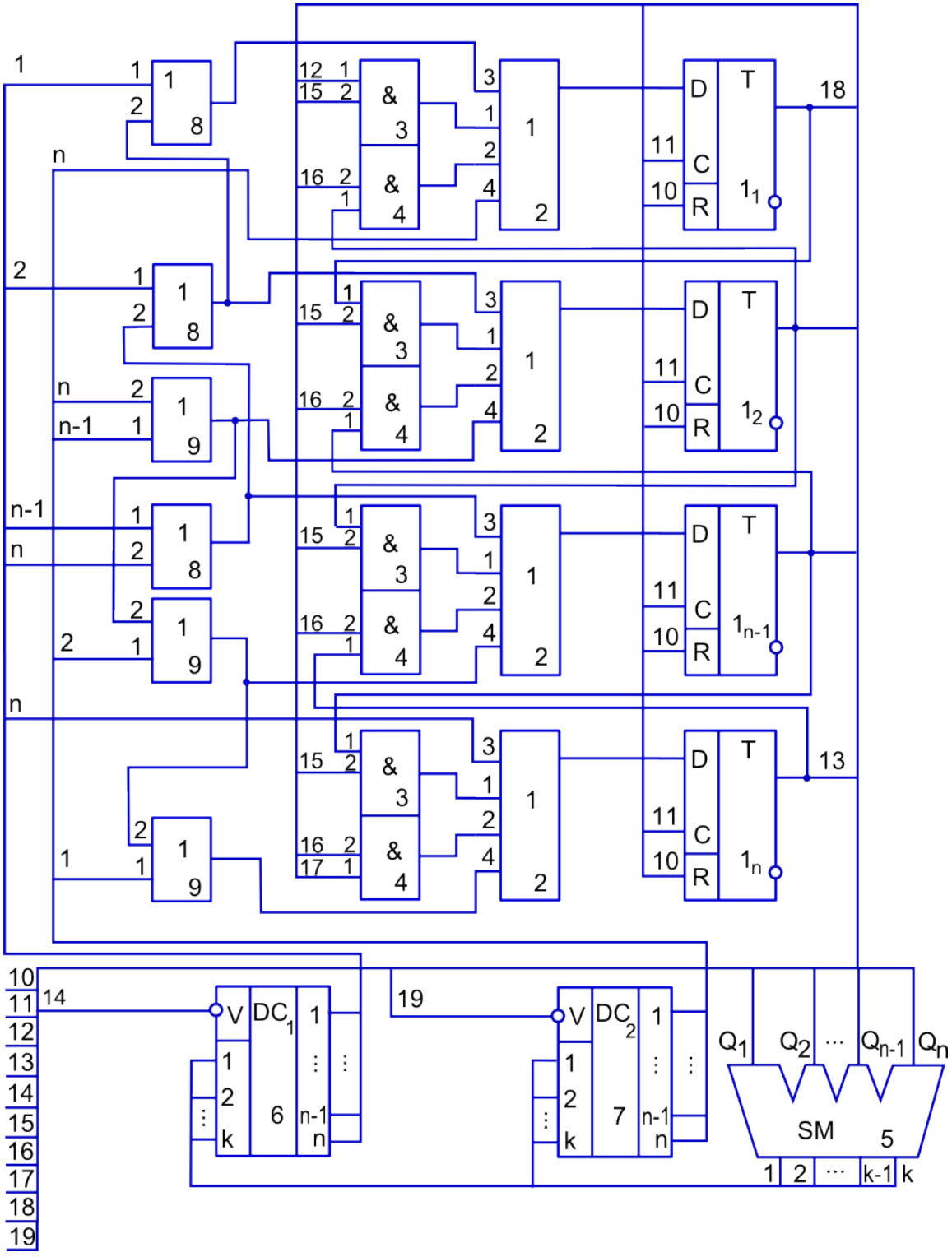


Рисунок 8.1 – Реверсивний регістр зсуву

Наприклад, після запису в регістр зсуву при n=8 коду 01001010 цей код буде зберігатися в регістрі зсуву подалі при відсутності імпульсів синхронізації на вході 11. При нульових значеннях сигналів на входах управління напрямом зсуву 15 і 16 регістр працює в режимі ущільнення двійкового коду ліворуч при сигналах 0 на вході 14 і 1 на вході 19.

Логічний зв'язок між прямим виходом тригера попереднього розряду і інформаційним D-входом тригера розряду, що розглядається, при цьому відсутній. Позиційний двійковий код кількості одиниць і з виходу комбінаційного суматора 5 за допомогою дешифратора 6 перетворюється у сигнал одиниці на виході і, який далі подається на перший вхід другої 8 схеми АБО розряду і і далі через ланцюг других 8 схем АБО на другі входи других 8 схем АБО в розрядах і, і-1, і-2, і-3, …,2, 1.

Після подачі одного імпульсу на вхід синхронізації 11 в регістр зсуву буде записано двійковий код 11100000 ущільнений ліворуч, який через вихід 18 може бути виведено з регістра зсуву зсувом ліворуч.

При нульових значеннях сигналів на входах управління напрямом зсуву 15 і 16 регістр працює в режимі ущільнення двійкового коду праворуч при сигналах 1 на вході 14 і 0 на вході 19.

Логічний зв'язок між прямим виходом тригера наступного розряду і інформаційним D-входом тригера розряду, що розглядається ,. при цьому відсутній. Позиційний двійковий код кількості одиниць і з виходу комбінаційного суматора 5 за допомогою дешифратора 7 перетворюється у сигнал одиниці на виході і , який далі подається на перший вхід третьої 9 схеми АБО розряду і і далі через ланцюг третіх 9 схем АБО на другі входи третіх 9 схем АБО в розрядах і, і+1, і-+2, і+3, … n - 1, n .

Наприклад, після запису в регістр зсуву при n=8 коду 01001010, переключення регістра потім в режим ущільнення праворуч і подачі одного сигналу на вхід синхронізації 11 одержимо в регістрі зсуву код 00000111, який через вихід 13 може бути виведено з регістра зсуву зсувом праворуч.

8.4 Висновки до розділу 8

Швидкодія регістра зсуву в режимі ущільнення коду не залежить від кількості розрядів коду і дорівнює одному такту.

Запропонований регістр зсуву має малі апаратні витрати і може бути використаний в пристроях діагностування і статистичної обробки інформації.

Даний регістр використовується в віртуальних обчислювачах аналізу великих даних для підвищення швидкодії в процедурах точного пошуку інформації за рахунок високого паралелізму векторно-логічних процедур, що повністю вилучають арифметичні операції.

9 АЛГЕБРА І СТРУКТУРИ МУЛЬТИПРОЦЕСОРА ДЛЯ АНАЛІЗУ ВЕЛИКИХ ДАНИХ

Пропонується алгебраїчна система, орієнтована на математичну підтримку моделей і методів векторного алгебрологічного аналізу кібернетичного простору при пошуку, розпізнаванні та прийнятті рішень [35, 58, 61, 103-106]. Розробляються структури даних і методологія синтезу паралельних мультипроцесорів для підвищення швидкодії аналізу великих даних.

9.1 Інфраструктура алгебрологічних перетворень

Інфраструктура векторних алгебрологічних перетворень в просторі VLAS (Vector Logical Algebra Space) подана на рис. 9.1. Вхідна інформація може бути задана матрицями і булевими змінними. Попередньо виконується векторизація змінних, а потім обробка отриманої асоціації. Вихідна інформація подана у вигляді вектора m, матриці A, а також булевої змінної Y, отриманої в результаті девекторизації асоціативної впорядкованої послідовності C. Для аналізу матричних або векторних даних, при заданих вхідних умовах, що подають матрицю або вектор, далі запропонована алгебраїчна матрична структура.

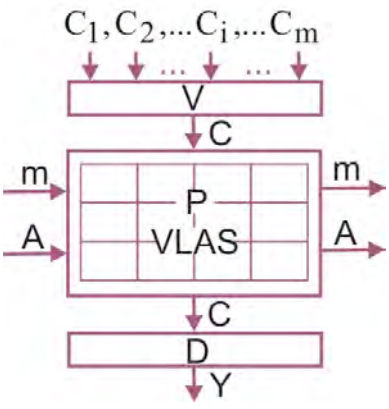


Рисунок 9.1 - Інфраструктура алгебраїчних перетворень

Алгебраїчні структури регулюють векторно-матричні перетворення в дискретному векторному булевому просторі для аналізу інформації на основі логічних операцій над асоціативними даними. Представлена алгебра орієнтована на обробку пар (матриця - матриця).

9.2 Система законів алгебри матричної логіки

Елементами є множини n-мірних двійкових векторів A,

$$A = \{a_s \mid a_s = (a_{s1}, a_{s2}, ..., a_{si}, ..., a_{sn}), a_{si} \in \{0,1\}, s = \overline{1,p}, i = \overline{1,n}\} \qquad (9.1)$$

зображених у матричній формі:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix}. \qquad (9.2)$$

Потужність множини A задається за умовою кількістю векторів-рядків: $|A| = p$. Таким чином, можна розглядати як (9.2) матрицю розміру $p \times n$ у формі (9.2).

Диз'юнкція і кон'юнкція сукупності матриць A і B визначається як покомпонентне виконання логічних операцій:

$$A \vee B = \{a_{ij} \vee b_{ij} \mid a_{ij} \in A, b_{ij} \in B, i = \overline{1,p}, j = \overline{1,n}\}, \qquad (9.3)$$

$$AB = \{a_{ij} b_{ij} \mid a_{ij} \in A, b_{ij} \in B, i = \overline{1,p}, j = \overline{1,n}\}. \qquad (9.4)$$

1. Комутативний закон:

$$A \vee B = B \vee A;\ \ A \wedge B = B \wedge A. \tag{9.5}$$

2. Асоціативний закон:

$$(A \vee B) \vee C = A \vee (B \vee C), \tag{9.6}$$

$$(AB)C = A(BC). \tag{9.7}$$

Д о в е д е н н я. Асоціативність диз'юнкції (кон'юнкції) матриць A, B і C можна відповідно зобразити у наступному вигляді. Доцільно виконати перетворення окремо лівої і правої частин рівності (9.6), яке підлягає доведенню. Тоді з лівої частини (9.6) випливає:

$$(A \vee B) \vee C = \left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \vee \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \right) \vee$$

$$\vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11} \vee b_{11} & a_{12} \vee b_{12} & \dots & a_{1i} \vee b_{1i} & \dots & a_{1n} \vee b_{1n} \\ a_{21} \vee b_{21} & a_{22} \vee b_{22} & \dots & a_{2i} \vee b_{2i} & \dots & a_{2n} \vee b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} \vee b_{s1} & a_{s2} \vee b_{s2} & \dots & a_{si} \vee b_{si} & \dots & a_{sn} \vee b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} \vee b_{p1} & a_{p2} \vee b_{p2} & \dots & a_{pi} \vee b_{pi} & \dots & a_{pn} \vee b_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} (a_{11} \vee b_{11}) \vee c_{11} & (a_{12} \vee b_{12}) \vee c_{12} & \dots & (a_{1i} \vee b_{1i}) \vee c_{1i} & \dots & (a_{1n} \vee b_{1n}) \vee c_{1n} \\ (a_{21} \vee b_{21}) \vee c_{21} & (a_{22} \vee b_{22}) \vee c_{22} & \dots & (a_{2i} \vee b_{2i}) \vee c_{2i} & \dots & (a_{2n} \vee b_{2n}) \vee c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{s1} \vee b_{s1}) \vee c_{s1} & (a_{s2} \vee b_{s2}) \vee c_{s2} & \dots & (a_{si} \vee b_{si}) \vee c_{si} & \dots & (a_{sn} \vee b_{sn}) \vee c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{p1} \vee b_{p1}) \vee c_{p1} & (a_{p2} \vee b_{p2}) \vee c_{p2} & \dots & (a_{pi} \vee b_{pi}) \vee c_{pi} & \dots & (a_{pn} \vee b_{pn}) \vee c_{pn} \end{bmatrix} \quad (9.8)$$

Права частина рівності (9.6) перетворюється до вигляду:

$$A \vee (B \vee C) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \vee$$

$$\vee \left(\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} \right) =$$

$$= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \vee \begin{bmatrix} b_{11} \vee c_{11} & b_{12} \vee c_{12} & \dots & b_{1i} \vee c_{1i} & \dots & b_{1n} \vee c_{1n} \\ b_{21} \vee c_{21} & b_{22} \vee c_{22} & \dots & b_{2i} \vee c_{2i} & \dots & b_{2n} \vee c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} \vee c_{s1} & b_{s2} \vee c_{s2} & \dots & b_{si} \vee c_{si} & \dots & b_{sn} \vee c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} \vee c_{p1} & b_{p2} \vee c_{p2} & \dots & b_{pi} \vee c_{pi} & \dots & b_{pn} \vee c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11} \vee (b_{11} \vee c_{11}) & a_{12} \vee (b_{12} \vee c_{12}) & \dots & a_{1i} \vee (b_{1i} \vee c_{1i}) & \dots & a_{1n} \vee (b_{1n} \vee c_{1n}) \\ a_{21} \vee (b_{21} \vee c_{21}) & a_{22} \vee (b_{22} \vee c_{22}) & \dots & a_{2i} \vee (b_{2i} \vee c_{2i}) & \dots & a_{2n} \vee (b_{2n} \vee c_{2n}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} \vee (b_{s1} \vee c_{s1}) & a_{s2} \vee (b_{s2} \vee c_{s2}) & \dots & a_{si} \vee (b_{si} \vee c_{si}) & \dots & a_{sn} \vee (b_{sn} \vee c_{sn}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} \vee (b_{p1} \vee c_{p1}) & a_{p2} \vee (b_{p2} \vee c_{p2}) & \dots & a_{pi} \vee (b_{pi} \vee c_{pi}) & \dots & a_{pn} \vee (b_{pn} \vee c_{pn}) \end{bmatrix} \quad (9.9)$$

Справедливість асоціативного закону у формі (9.6) для диз'юнкції випливає з поелементної рівності матриць (9.8) і (9.9), що можна продемонструвати за допомогою таблиці істинності 9.1:

Таблиця 9.1 – Доведення асоціативного закону ув формі (9.6)

$$(a_{ij} \vee b_{ij}) \vee c_{ij} = a_{ij} \vee (b_{ij} \vee c_{ij})$$

№	a_{ij}	b_{ij}	c_{ij}	$a_{ij} \vee b_{ij}$	$(a_{ij} \vee b_{ij}) \vee c_{ij}$	$b_{ij} \vee c_{ij}$	$a_{ij} \vee (b_{ij} \vee c_{ij})$
0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	1
2	0	1	0	1	1	1	1
3	0	1	1	1	1	1	1
4	1	0	0	1	1	0	1
5	1	0	1	1	1	1	1
6	1	1	0	1	1	1	1
7	1	1	1	1	1	1	1

Доведення асоціативності кон'юнкції в формі (9.7) проводиться аналогічно:

$$(AB)C = \left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \right) \wedge$$

$$\wedge \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1i}b_{1i} & \dots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2i}b_{2i} & \dots & a_{2n}b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1}b_{s1} & a_{s2}b_{s2} & \dots & a_{si}b_{si} & \dots & a_{sn}b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1}b_{p1} & a_{p2}b_{p2} & \dots & a_{pi}b_{pi} & \dots & a_{pn}b_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} (a_{11}b_{11})c_{11} & (a_{12}b_{12})c_{12} & \dots & (a_{1i}b_{1i})c_{1i} & \dots & (a_{1n}b_{1n})c_{1n} \\ (a_{21}b_{21})c_{21} & (a_{22}b_{22})c_{22} & \dots & (a_{2i}b_{2i})c_{2i} & \dots & (a_{2n}b_{2n})c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{s1}b_{s1})c_{s1} & (a_{s2}b_{s2})c_{s2} & \dots & (a_{si}b_{si})c_{si} & \dots & (a_{sn}b_{sn})c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{p1}b_{p1})c_{p1} & (a_{p2}b_{p2})c_{p2} & \dots & (a_{pi}b_{pi})c_{pi} & \dots & (a_{pn}b_{pn})c_{pn} \end{bmatrix}. \quad (9.10)$$

Права частина (9.7) перетворюється до вигляду:

$$A(BC) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \wedge$$

$$\wedge \left(\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \wedge \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} \right) =$$

$$= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \cdot \begin{bmatrix} b_{11}c_{11} & b_{12}c_{12} & \dots & b_{1i}c_{1i} & \dots & b_{1n}c_{1n} \\ b_{21}c_{21} & b_{22}c_{22} & \dots & b_{2i}c_{2i} & \dots & b_{2n}c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1}c_{s1} & b_{s2}c_{s2} & \dots & b_{si}c_{si} & \dots & b_{sn}c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1}c_{p1} & b_{p2}c_{p2} & \dots & b_{pi}c_{pi} & \dots & b_{pn}c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}(b_{11}c_{11}) & a_{12}(b_{12}c_{12}) & \dots & a_{1i}(b_{1i}c_{1i}) & \dots & a_{1n}(b_{1n}c_{1n}) \\ a_{21}(b_{21}c_{21}) & a_{22}(b_{22}c_{22}) & \dots & a_{2i}(b_{2i}c_{2i}) & \dots & a_{2n}(b_{2n}c_{2n}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1}(b_{s1}c_{s1}) & a_{s2}(b_{s2}c_{s2}) & \dots & a_{si}(b_{si}c_{si}) & \dots & a_{sn}(b_{sn}c_{sn}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1}(b_{p1}c_{p1}) & a_{p2}(b_{p2}c_{p2}) & \dots & a_{pi}(b_{pi}c_{pi}) & \dots & a_{pn}(b_{pn}c_{pn}) \end{bmatrix}. \quad (9.11)$$

Справедливість асоціативного закону у формі (9.7) для кон'юнкції випливає також з покомпонентної рівності матриць з (9.10) і (9.11), що можна продемонструвати за допомогою таблиці істинності 9.2.

Таблиця 9.2 - Доведення асоціативного закона в формі (9.7)

$(a_{ij}b_{ij})c_{ij} = a_{ij}(b_{ij}c_{ij})$

№	a_{ij}	b_{ij}	c_{ij}	$a_{ij}b_{ij}$	$(a_{ij}b_{ij})c_{ij}$	$b_{ij}c_{ij}$	$a_{ij}(b_{ij}c_{ij})$
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
2	0	1	0	0	0	0	0
3	0	1	1	0	0	1	0
4	1	0	0	0	0	0	0
5	1	0	1	0	0	0	0
6	1	1	0	1	0	0	0
7	1	1	1	1	1	1	1

3. Дистрибутивний закон:

$$(A \vee B)C = AC \vee BC, \quad (9.12)$$

$$AB \vee C = (A \vee C)(B \vee C). \quad (9.13)$$

Д о в е д е н н я. В дистрибутивному законі беруть участь три матриці розміру pxn, тобто сумісні. Виконання рівності (9.12) можна продемонструвати шляхом послідовного приведення лівої його частини до вигляду:

$$(A \vee B)C = \left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \vee \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \right) \wedge$$

$$\wedge \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11} \vee b_{11} & a_{12} \vee b_{12} & \dots & a_{1i} \vee b_{1i} & \dots & a_{1n} \vee b_{1n} \\ a_{21} \vee b_{21} & a_{22} \vee b_{22} & \dots & a_{2i} \vee b_{2i} & \dots & a_{2n} \vee b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} \vee b_{s1} & a_{s2} \vee b_{s2} & \dots & a_{si} \vee b_{si} & \dots & a_{sn} \vee b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} \vee b_{p1} & a_{p2} \vee b_{p2} & \dots & a_{pi} \vee b_{pi} & \dots & a_{pn} \vee b_{pn} \end{bmatrix} \cdot \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} (a_{11} \vee b_{11})c_{11} & (a_{12} \vee b_{12})c_{12} & \dots & (a_{1i} \vee b_{1i})c_{1i} & \dots & (a_{1n} \vee b_{1n})c_{1n} \\ (a_{21} \vee b_{21})c_{21} & (a_{22} \vee b_{22})c_{22} & \dots & (a_{2i} \vee b_{2i})c_{2i} & \dots & (a_{2n} \vee b_{2n})c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{s1} \vee b_{s1})c_{s1} & (a_{s2} \vee b_{s2})c_{s2} & \dots & (a_{si} \vee b_{si})c_{si} & \dots & (a_{sn} \vee b_{sn})c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{p1} \vee b_{p1})c_{p1} & (a_{p2} \vee b_{p2})c_{p2} & \dots & (a_{pi} \vee b_{pi})c_{pi} & \dots & (a_{pn} \vee b_{pn})c_{pn} \end{bmatrix}. \quad (9.14)$$

Далі права частина (9.12) перетворюється як:

$$AC \vee BC = \left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \cdot \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} \right) \vee$$

$$\vee \left(\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \cdot \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} \right) =$$

$$= \begin{bmatrix} a_{11}c_{11} & a_{12}c_{12} & \dots & a_{1i}c_{1i} & \dots & a_{1n}c_{1n} \\ a_{21}c_{21} & a_{22}c_{22} & \dots & a_{2i}c_{2i} & \dots & a_{2n}c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1}c_{s1} & a_{s2}c_{s2} & \dots & a_{si}c_{si} & \dots & a_{sn}c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1}c_{p1} & a_{p2}c_{p2} & \dots & a_{pi}c_{pi} & \dots & a_{pn}c_{pn} \end{bmatrix} \vee \begin{bmatrix} b_{11}c_{11} & b_{12}c_{12} & \dots & b_{1i}c_{1i} & \dots & b_{1n}c_{1n} \\ b_{21}c_{21} & b_{22}c_{22} & \dots & b_{2i}c_{2i} & \dots & b_{2n}c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1}c_{s1} & b_{s2}c_{s2} & \dots & b_{si}c_{si} & \dots & b_{sn}c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1}c_{p1} & b_{p2}c_{p2} & \dots & b_{pi}c_{pi} & \dots & b_{pn}c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}c_{11} \vee b_{11}c_{11} & a_{12}c_{12} \vee b_{12}c_{12} & \dots & a_{1i}c_{1i} \vee b_{1i}c_{1i} & \dots & a_{1n}c_{1n} \vee b_{1n}c_{1n} \\ a_{21}c_{21} \vee b_{21}c_{21} & a_{22}c_{22} \vee b_{22}c_{22} & \dots & a_{2i}c_{2i} \vee b_{2i}c_{2i} & \dots & a_{2n}c_{2n} \vee b_{2n}c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1}c_{s1} \vee b_{s1}c_{s1} & a_{s2}c_{s2} \vee b_{s2}c_{s2} & \dots & a_{si}c_{si} \vee b_{si}c_{si} & \dots & a_{sn}c_{sn} \vee b_{sn}c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1}c_{p1} \vee b_{p1}c_{p1} & a_{p2}c_{p2} \vee b_{p2}c_{p2} & \dots & a_{pi}c_{pi} \vee b_{pi}c_{pi} & \dots & a_{pn}c_{pn} \vee b_{pn}c_{pn} \end{bmatrix}. \quad (9.15)$$

Аналогічно ліву частину (9.13) можна привести до вигляду:

$$AB \vee C = \left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \right) \vee$$

$$\vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1i}b_{1i} & \dots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2i}b_{2i} & \dots & a_{2n}b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1}b_{s1} & a_{s2}b_{s2} & \dots & a_{si}b_{si} & \dots & a_{sn}b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1}b_{p1} & a_{p2}b_{p2} & \dots & a_{pi}b_{pi} & \dots & a_{pn}b_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}b_{11} \vee c_{11} & a_{12}b_{12} \vee c_{12} & \dots & a_{1i}b_{1i} \vee c_{1i} & \dots & a_{1n}b_{1n} \vee c_{1n} \\ a_{21}b_{21} \vee c_{21} & a_{22}b_{22} \vee c_{22} & \dots & a_{2i}b_{2i} \vee c_{2i} & \dots & a_{2n}b_{2n} \vee c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1}b_{s1} \vee c_{s1} & a_{s2}b_{s2} \vee c_{s2} & \dots & a_{si}b_{si} \vee c_{si} & \dots & a_{sn}b_{sn} \vee c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1}b_{p1} \vee c_{p1} & a_{p2}b_{p2} \vee c_{p2} & \dots & a_{pi}b_{pi} \vee c_{pi} & \dots & a_{pn}b_{pn} \vee c_{pn} \end{bmatrix}. \quad (9.16)$$

З правої частини (9.13) виходить матриця:

$$(A \vee C)(B \vee C) =$$

$$= \left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} \right) \wedge$$

$$\wedge \left(\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1i} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2i} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{si} & \dots & b_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pi} & \dots & b_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix} \right) =$$

$$= \begin{bmatrix} a_{11} \vee c_{11} & a_{12} \vee c_{12} & \dots & a_{1i} \vee c_{1i} & \dots & a_{1n} \vee c_{1n} \\ a_{21} \vee c_{21} & a_{22} \vee c_{22} & \dots & a_{2i} \vee c_{2i} & \dots & a_{2n} \vee c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} \vee c_{s1} & a_{s2} \vee c_{s2} & \dots & a_{si} \vee c_{si} & \dots & a_{sn} \vee c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} \vee c_{p1} & a_{p2} \vee c_{p2} & \dots & a_{pi} \vee c_{pi} & \dots & a_{pn} \vee c_{pn} \end{bmatrix} \wedge$$

$$\wedge \begin{bmatrix} b_{11} \vee c_{11} & b_{12} \vee c_{12} & \dots & b_{1i} \vee c_{1i} & \dots & b_{1n} \vee c_{1n} \\ b_{21} \vee c_{21} & b_{22} \vee c_{22} & \dots & b_{2i} \vee c_{2i} & \dots & b_{2n} \vee c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{s1} \vee c_{s1} & b_{s2} \vee c_{s2} & \dots & b_{si} \vee c_{si} & \dots & b_{sn} \vee c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_{p1} \vee c_{p1} & b_{p2} \vee c_{p2} & \dots & b_{pi} \vee c_{pi} & \dots & b_{pn} \vee c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} (a_{11} \vee c_{11})(b_{11} \vee c_{11}) & (a_{12} \vee c_{12})(b_{12} \vee c_{12}) & \dots & (a_{1i} \vee c_{1i})(b_{1i} \vee c_{1i}) & \dots & (a_{1n} \vee c_{1n})(b_{1n} \vee c_{1n}) \\ (a_{21} \vee c_{21})(b_{21} \vee c_{21}) & (a_{22} \vee c_{22})(b_{22} \vee c_{22}) & \dots & (a_{2i} \vee c_{2i})(b_{2i} \vee c_{2i}) & \dots & (a_{2n} \vee c_{2n})(b_{2n} \vee c_{2n}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{s1} \vee c_{s1})(b_{s1} \vee c_{s1}) & (a_{s2} \vee c_{s2})(b_{s2} \vee c_{s2}) & \dots & (a_{si} \vee c_{si})(b_{si} \vee c_{si}) & \dots & (a_{sn} \vee c_{sn})(b_{sn} \vee c_{sn}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a_{p1} \vee c_{p1})(b_{p1} \vee c_{p1}) & (a_{p2} \vee c_{p2})(b_{p2} \vee c_{p2}) & \dots & (a_{pi} \vee c_{pi})(b_{pi} \vee c_{pi}) & \dots & (a_{pn} \vee c_{pn})(b_{pn} \vee c_{pn}) \end{bmatrix} \quad (9.17)$$

Покомпонентне порівняння матриць (9.14) і (9.15), а також (9.16) і (9.17) повинно задовільнювати формулам:

$$(a_{ij} \vee b_{ij})c_{ij} = a_{ij}c_{ij} \vee b_{ij}c_{ij}, \quad (9.18)$$

$$a_{ij}b_{ij} \vee c_{ij} = (a_{ij} \vee c_{ij})(b_{ij} \vee c_{ij}), \quad (9.19)$$

де $i = \overline{1,p}, j = \overline{1,n}$. Область визначення для (9.18), (9.19) обмежується стандартними $2^3 = 8$ двійковими наборами. Доведення рівностей (9.18) і (9.19) подається у вигляді таблиць істинності 9.3, 9.4.

Таблиця 9.3 – Доведення дистрибутивного закону в формі (9.12)

$$(a_{ij} \vee b_{ij})c_{ij} = a_{ij}c_{ij} \vee b_{ij}c_{ij}, i = \overline{1,p}, j = \overline{1,n}$$

№	a_{ij}	b_{ij}	c_{ij}	$a_{ij} \vee b_{ij}$	$(a_{ij} \vee b_{ij})c_{ij}$	$a_{ij}c_{ij}$	$b_{ij}c_{ij}$	$a_{ij}c_{ij} \vee b_{ij}c_{ij}$
0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
2	0	1	0	1	0	0	0	0
3	0	1	1	1	1	0	1	1
4	1	0	0	1	0	0	0	0
5	1	0	1	1	1	1	0	1
6	1	1	0	1	0	0	0	0
7	1	1	1	1	1	1	1	1

Таблица 9.4 – Доведення дистрибутивного закону в формі (9.13)

$$a_{ij}b_{ij} \vee c_{ij} = (a_{ij} \vee c_{ij})(b_{ij} \vee c_{ij}), i = \overline{1,p}, j = \overline{1,n}$$

№	a_{ij}	b_{ij}	c_{ij}	$a_{ij}b_{ij}$	$a_{ij}b_{ij} \vee c_{ij}$	$a_{ij} \vee c_{ij}$	$b_{ij} \vee c_{ij}$	$(a_{ij} \vee c_{ij})(b_{ij} \vee c_{ij})$
0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	1	1
2	0	1	0	0	0	0	1	0
3	0	1	1	0	1	1	1	1
4	1	0	0	0	0	1	0	0
5	1	0	1	0	1	1	1	1
6	1	1	0	1	1	1	1	1
7	1	1	1	1	1	1	1	1

Таким чином, справедливість дистрибутивного закону встановлена.

4. Дії з константами. Константами виступають нульова та одинична матриці: M^0 и M^1:

$$M \vee M^0 = M,\ M \cdot M^0 = M^0, \tag{9.20}$$

$$M \vee M^1 = M^1,\ M \cdot M^1 = M, \tag{9.21}$$

$$\overline{M}^0 = M^1,\ \overline{M}^1 = M^0. \tag{9.22}$$

5. Закон протиріччя:

$$M \cdot \overline{M} = M^0. \tag{9.23}$$

6. Закон виключеного третього:

$$M \vee \overline{M} = M^1. \tag{9.24}$$

7. Закон елімінації (поглинання):

$$A(A \vee B) = A, \; A \vee AB = A. \tag{9.25}$$

Д о в е д е н н я. Застосування операцій з константами та дистрибутивного закону до перетворення (9.25) дає:

$$A(A \vee B) \overset{(9.20)}{=} (A \vee M^0)(A \vee B) \overset{(9.13)}{=} A \vee M^0 B \overset{(9.20)}{=} A \vee M^0 \overset{(9.20)}{=} A,$$

$$A \vee AB \overset{(9.21)}{=} AM^1 \vee AB \overset{(9.20)}{=} A(M^1 \vee B) \overset{(9.21)}{=} AM^1 \overset{(9.21)}{=} A.$$

8. Закон зклеювання:

$$(A \vee B)(A \vee \overline{B}) = A, \; AB \vee A\overline{B} = A. \tag{9.26}$$

Д о в е д е н н я. На підставі дистрибутивного закону, законів виключеного третього, протиріччя та дій з константами з рівностей (9.26) випливає:

$$(A \vee B)(A \vee \overline{B}) \overset{(9.13)}{=} A \vee B\overline{B} \overset{(9.23)}{=} A \vee M^0 \overset{(9.20)}{=} A,$$

$$AB \vee A\overline{B} \overset{(9.12)}{=} A(B \vee \overline{B}) \overset{(9.24)}{=} A \cdot M^1 \overset{(9.21)}{=} A.$$

9. Закон Блейка-Порецького:

$$A(\overline{A} \vee B) = AB,\ A \vee \overline{A}B = A \vee B, \tag{9.27}$$

$$\overline{A}(A \vee B) = \overline{A}B,\ \overline{A} \vee AB = \overline{A} \vee B. \tag{9.28}$$

Д о в е д е н н я. Слід скористатися дистрибутивниым законом. Тоді на прикладі першої рівності (9.27) можна отримати:

$$A(\overline{A} \vee B) \overset{(9.12)}{=} A\overline{A} \vee AB \overset{(9.23)}{=} M^0 \vee AB \overset{(9.20)}{=} AB.$$

Інші формулювання закону Порецького доводяться аналогічно.

10. Закон идемпотентності:

$$A \vee A = A, A \cdot A = A.$$

11. Закон інволюції: $\overline{\overline{A}} = A$.

12. Закон Де Моргана:

$$\overline{A \vee C} = \overline{A} \cdot \overline{C}, \tag{9.29}$$

$$\overline{AC} = \overline{A} \vee \overline{C}. \tag{9.30}$$

Д о в е д е н н я. На підставі покомпонентних операцій над елементами матриць справедливість формул (9.29), (9.30) можна довести за допомогою таблиць істинності. Тодігда перетворення (9.29) дає:

$\overline{A \vee C} =$

$$= \overline{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix} \vee \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix}} =$$

$$= \overline{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} & a_{s2} & \dots & a_{si} & \dots & a_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pi} & \dots & a_{pn} \end{bmatrix}} \cdot \overline{\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1i} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2i} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{s1} & c_{s2} & \dots & c_{si} & \dots & c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p1} & c_{p2} & \dots & c_{pi} & \dots & c_{pn} \end{bmatrix}},$$

$$\overline{\begin{bmatrix} a_{11} \vee c_{11} & a_{12} \vee c_{12} & \dots & a_{1i} \vee c_{1i} & \dots & a_{1n} \vee c_{1n} \\ a_{21} \vee c_{21} & a_{22} \vee c_{22} & \dots & a_{2i} \vee c_{2i} & \dots & a_{2n} \vee c_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{s1} \vee c_{s1} & a_{s2} \vee c_{s2} & \dots & a_{si} \vee c_{si} & \dots & a_{sn} \vee c_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} \vee c_{p1} & a_{p2} \vee c_{p2} & \dots & a_{pi} \vee c_{pi} & \dots & a_{pn} \vee c_{pn} \end{bmatrix}} =$$

$$= \begin{bmatrix} \bar{a}_{11} & \bar{a}_{12} & \dots & \bar{a}_{1i} & \dots & \bar{a}_{1n} \\ \bar{a}_{21} & \bar{a}_{22} & \dots & \bar{a}_{2i} & \dots & \bar{a}_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \bar{a}_{s1} & \bar{a}_{s2} & \dots & \bar{a}_{si} & \dots & \bar{a}_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \bar{a}_{p1} & \bar{a}_{p2} & \dots & \bar{a}_{pi} & \dots & \bar{a}_{pn} \end{bmatrix} \cdot \begin{bmatrix} \bar{c}_{11} & \bar{c}_{12} & \dots & \bar{c}_{1i} & \dots & \bar{c}_{1n} \\ \bar{c}_{21} & \bar{c}_{22} & \dots & \bar{c}_{2i} & \dots & \bar{c}_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \bar{c}_{s1} & \bar{c}_{s2} & \dots & \bar{c}_{si} & \dots & \bar{c}_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \bar{c}_{p1} & \bar{c}_{p2} & \dots & \bar{c}_{pi} & \dots & \bar{c}_{pn} \end{bmatrix},$$

$$\begin{bmatrix} \overline{a_{11} \vee c_{11}} & \overline{a_{12} \vee c_{12}} & \dots & \overline{a_{1i} \vee c_{1i}} & \dots & \overline{a_{1n} \vee c_{1n}} \\ a_{21} \vee c_{21} & a_{22} \vee c_{22} & \dots & a_{2i} \vee c_{2i} & \dots & a_{2n} \vee c_{2n} \\ \overline{\dots} & \overline{\dots} & \dots & \overline{\dots} & \dots & \overline{\dots} \\ a_{s1} \vee c_{s1} & a_{s2} \vee c_{s2} & \dots & a_{si} \vee c_{si} & \dots & a_{sn} \vee c_{sn} \\ \overline{\dots} & \overline{\dots} & \dots & \overline{\dots} & \dots & \overline{\dots} \\ a_{p1} \vee c_{p1} & a_{p2} \vee c_{p2} & \dots & a_{pi} \vee c_{pi} & \dots & a_{pn} \vee c_{pn} \end{bmatrix} =$$

$$= \begin{bmatrix} \bar{a}_{11}\bar{c}_{11} & \bar{a}_{12}\bar{c}_{12} & \dots & \bar{a}_{1i}\bar{c}_{1i} & \dots & \bar{a}_{1n}\bar{c}_{1n} \\ \bar{a}_{21}\bar{c}_{21} & \bar{a}_{22}\bar{c}_{22} & \dots & \bar{a}_{2i}\bar{c}_{2i} & \dots & \bar{a}_{2n}\bar{c}_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \bar{a}_{s1}\bar{c}_{s1} & \bar{a}_{s2}\bar{c}_{s2} & \dots & \bar{a}_{si}\bar{c}_{si} & \dots & \bar{a}_{sn}\bar{c}_{sn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \bar{a}_{p1}\bar{c}_{p1} & \bar{a}_{p2}\bar{c}_{p2} & \dots & \bar{a}_{pi}\bar{c}_{pi} & \dots & \bar{a}_{pn}\bar{c}_{pn} \end{bmatrix}.$$

Для формули (9.30) перетворення виконуються аналогічно. Поелементне порівняння матриць приводить до перевірки рівностей

$$\overline{a_{ij} \vee c_{ij}} = \bar{a}_{ij} \cdot \bar{c}_{ij},\ \overline{a_{ij}c_{ij}} = \bar{a}_{ij} \vee \bar{c}_{ij},$$

за допомогою таблиць істинності, які будуть включати $2^2 = 4$ стандартних двійкових наборів. Тоді покомпонентне виконання логічних операцій визначається наступним чином (табл. 9.5, 9.6):

Таблиця 9.5 – Перевірка закону Де Моргана в формі (9.29)

$$\overline{a_{ij} \vee c_{ij}} = \bar{a}_{ij} \cdot \bar{c}_{ij}, i = \overline{1,p}, j = \overline{1,n}$$

N	a_{ij}	c_{ij}	$a_{ij} \vee c_{ij}$	$\overline{a_{ij} \vee c_{ij}}$	$\bar{a}_{ij}$	$\bar{c}_{ij}$	$\bar{a}_{ij} \cdot \bar{c}_{ij}$
0	0	0	0	1	1	1	1
1	0	1	1	0	1	0	0
2	1	0	1	0	0	1	0
3	1	1	1	0	0	0	0

Таблица 9.6 – Перевірка закону Де Моргана в формі (9.30)

$$\overline{a_{ij}c_{ij}} = \bar{a}_{ij} \vee \bar{c}_{ij}, i = \overline{1,p}, j = \overline{1,n}$$

N	a_{ij}	c_{ij}	$a_{ij}c_{ij}$	$\overline{a_{ij}c_{ij}}$	$\bar{a}_{ij}$	$\bar{c}_{ij}$	$\bar{a}_{ij} \vee \bar{c}_{ij}$
0	0	0	0	1	1	1	1
1	0	1	0	1	1	0	1
2	1	0	0	1	0	1	1
3	1	1	1	0	0	0	0

Таким чином, встановлено, що для алгебри матричної логіки система законів аналогічна законам булевої алгебри.

9.3 Мультипроцесор паралельного аналізу великих даних

Далі подані моделі кібер-вибору, що важливо при розгляді практично орієнтованих задач - знайти краще рішення з кінцевого числа варіантів взаємодії m⊕Ai для його подальшої реалізації без арифметичних операцій, які на один два порядки знижують швидкодію. Нехай, наприклад є два вектора AB, відносно яких необхідно виконати операції: a = m⊕A, b = m⊕B, щоб оцінити близькість кожного з них до вектор-запиту m:

A =	1	0	1	0	0	1	0	1
B =	0	1	0	0	1	0	1	1
m =	1	1	1	1	1	1	0	0
$\mu_a(m \in A) = m \oplus A$	0	1	0	1	1	0	0	1
$\mu_b(m \in B) = m \oplus B$	1	0	1	1	0	1	1	1
$a = slc(\mu_a)$	1	1	1	1	0	0	0	0
$b = slc(\mu_b)$	1	1	1	1	1	1	0	0

A =	0	0	0	1	0	0	0	1
B =	0	1	0	0	1	1	0	0
m =	1	1	1	1	1	1	0	0
$\mu_a(m \in A) = m \oplus A$	1	1	1	0	1	1	0	1
$\mu_b(m \in B) = m \oplus B$	1	0	1	1	0	0	0	0
$a = slc(\mu_a)$	1	1	1	1	1	1	0	0
$b = slc(\mu_b)$	1	1	1	0	0	0	0	0

A =	1	0	1	0	0	1	1	1
B =	0	1	0	0	1	0	0	1
m =	1	1	1	1	1	1	0	0
$\mu_a(m \in A) = m \oplus A$	0	1	0	1	1	0	1	1
$\mu_b(m \in B) = m \oplus B$	1	0	1	1	0	1	0	1
$a = slc(\mu_a)$	1	1	1	1	1	0	0	0
$b = slc(\mu_b)$	1	1	1	1	1	0	0	0

Далі пропонується проста і доступна для розуміння і реалізації структура паралельного обчислення кращого варіанту на основі нечисельних порівняння двох альтернативних векторів a і b, отриманих на основі використання однотактної операції slc - зрушення всіх одиниць вліво з ущільненням [63]. Після паралельного зсуву за один такт всіх одиниць в регістрах векторного критерію якості, що оцінюють взаємодії об'єктів в кіберпросторі, теоретично можливі три варіанти співвідношення одиниць, представлених нижче (взаємодія раніше отриманих векторів a і b):

a =	1	1	1	1	0	0	0	0
b =	1	1	1	1	1	1	0	0
$a \wedge b$	1	1	1	1	0	0	0	0
$q^b = (a \wedge b) \oplus b$	0	0	0	0	1	1	0	0
$q^a = (a \wedge b) \oplus a$	0	0	0	0	0	0	0	0
$Q^b = \bigvee_{i=1}^{k} q_i^b = 1$								
$Q^a = \bigvee_{i=1}^{k} q_i^a = 0$	w	i	n	n	e	r		

a =	1	1	1	1	1	1	0	0
b =	1	1	1	0	0	0	0	0
$a \wedge b$	1	1	1	0	0	0	0	0
$q^b = (a \wedge b) \oplus b$	0	0	0	0	0	0	0	0
$q^a = (a \wedge b) \oplus a$	0	0	0	1	1	1	0	0
$Q^b = \bigvee_{i=1}^{k} q_i^b = 0$	w	i	n	n	e	r		
$Q^a = \bigvee_{i=1}^{k} q_i^a = 1$								

a =	1	1	1	1	1	0	0	0
b =	1	1	1	1	1	0	0	0
$a \wedge b$	1	1	1	1	1	0	0	0
$q^b = (a \wedge b) \oplus b$	0	0	0	0	0	0	0	0
$q^a = (a \wedge b) \oplus a$	0	0	0	0	0	0	0	0
$Q^b = \bigvee_{i=1}^{k} q_i^b = 0$	w	i	n	n	e	r		
$Q^a = \bigvee_{i=1}^{k} q_i^a = 1$	w	i	n	n	e	r		

Пояснення: Нульове значення Q-критерію означає кращу альтернативу з розглянутих, яка використовується далі для порівняння з іншими оцінками або як кінцевий варіант вирішення проблеми. Логічна структура для реалізації кібер-вибору має наступний вигляд:

$$\left\langle \begin{matrix} Q^b = \bigvee\limits_{i=1}^{k} q_i^b \\ Q^a = \bigvee\limits_{i=1}^{k} q_i^a \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} q^b = (a \wedge b) \oplus b \\ q^a = (a \wedge b) \oplus a \end{matrix} \right\rangle \leftarrow \langle a \wedge b \rangle \leftarrow \left\langle \begin{matrix} a \\ b \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} a = \mathrm{slc}(\mu_a) \\ b = \mathrm{slc}(\mu_b) \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} \mu_a(m \in A) = m \oplus A \\ \mu_b(m \in B) = m \oplus B \end{matrix} \right\rangle \leftarrow \left\langle \begin{matrix} m \\ A \\ B \end{matrix} \right\rangle$$

Регістрові змінні a і b, що позначають вектори стислих вліво одиничних значень, об'єднуються і інвертуються для одночасного виконання xor-операцій. Потім результати у вигляді станів регістрів подаються на входи двох логічних елементів or, які формують вже стани двох булевих змінних, які створюють три поєднання: 00, 01, 10. Нульове значення однієї з двох змінних означає найкращий розв’язок, який необхідно вибрати. Два нульових стани означають, що обидва рішення рівнозначні за рівнем переваги. Одинична комбінація булевих змінних неможлива. Схемна реалізація кібер-вибору з двох альтернатив зображена на рис. 9.2.

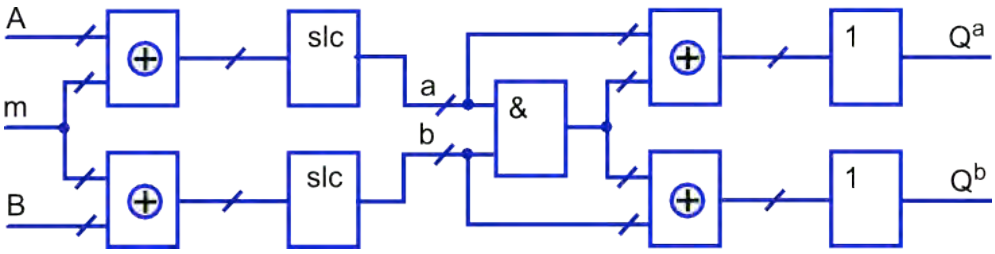


Рисунок 9.2 - Схемна реалізація вибору оптимального розв’язку

Якщо врахувати, що вибір кращого розв’язку повинен ідентифікуватися логічним сигналом 1 (замість 0), що відповідає максимальному значенню функції належності, визначеної раніше, тоді схемна структура для визначення кращого варіанту з двох альтернатив може

бути незначно модифікована за допомогою додаткових інверторів на виходах логічних or елементів, що формують $\{Q^a, Q^b\} \in Q$. У цьому випадку на виходах цифрової структури можливі наступні пари станів: 10, 01 і 11, де вибір рішення (a або b) здійснюється за одиничним значенням одного з виходів. Пара сигналів 00 на зовнішніх виходах схеми з інверторами неможлива. Таким чином, точний пошук замовленої інформації в big data можна і потрібно виконувати на основі тільки логічних операцій and, or, not, xor, slc без використання арифметичних функцій, що дозволяє проектувати швидкодіючі векторно-логічні фізичні та/або віртуальні мультипроцесори для суттєвого зменшення часу виконання сервісів хмарними додатками. Обчислювальна процедура пошуку кращого варіанту з двох можливих зводиться до паралельного виконання 4-х векторних операцій і однієї логічної, в результаті яких на одному або декількох виходах формується значення нуля, яке ідентифікує кращий розв’язок:

$$Q^a = \underset{i}{\vee}\{[slc(m \oplus A) \wedge slc(m \oplus B)] \oplus A\}, \quad Q^b = \underset{i}{\vee}\{[slc(m \oplus A) \wedge slc(m \oplus B)] \oplus B\}.$$

Запропонована дискретна булева метрика, векторні критерії якості xor-взаємодії об'єктів в кіберпросторі, нечисельне ранжування отриманих критеріїв для вибору об'єкта за запитом дають підстави вважати про ринкове впровадження програмно-апаратної реалізації метрики і заснованої на ній обчислювальної структури в хмарні сервіси аналізу big data.

Векторно-логічний SIMD-мультипроцесор характеризується відсутністю арифметичних операцій, паралельним обчисленням відстані між запитом та інформаційними квантами, а також одночасним визначенням кращого з можливих n-рішень за мінімумом функції приналежності, що дає можливість на порядок підвищити швидкодію максимально точного пошуку даних в big data. Його структура зображена на рис. 9.3, де подані тільки логічні примітиви для виконання векторних і булевих (бітових) операцій.

Процесор працює таким чином: вектор запит m, що складається з k-бітів, взаємодіє з xor-функцією з матрицею M, що має n рядків або векторів. В результаті виконання xor-операцій формуються n функцій приналежності, що визначають ступінь близькості або відстань між запитом і кожним вектор-рядком матриці M. Для оцінювання відстаней і вибору кращої (мінімальної) взаємодії виконується реєстрова операція slc, яка здійснює ущільнення всіх одиниць із зсувом вліво за один автоматний такт, що дає можливість оцінювати мінімальну відстань $m \oplus M_i$ номером біта, в якому знаходиться права крайня одиниця. Для визначення номера вектора-рядку, що формує мінімум функції приналежності, здійснюється паралельна поразрядна операція логічного множення над усіма векторами, що містять зсунуті вліво поодинокі значення, що дає можливість обчислити вектор з мінімальним числом одиниць A_{min}. Останній використовується для визначення номера або індексу вектор-рядка матриці M, що має краще значення функції приналежності, шляхом виконання векторної xor-операції між A_{min} і всіма зсунутими функціями належності A_i $(i = 1, n)$. Внаслідок цього формуються вектори q_i $(i = 1, n)$, біти яких визначають вхідні значення кожного з n логічних елементів or. Вихід кожного or-елемента дорівнює одиниці, якщо існує хоча б одне одиничне значення в результатах порівняння $A_i \oplus A_{min}$. Якщо таких одиниць немає, то мінімальна відстань між $m \oplus M_i$ ідентифікується 0-станом одного або, можливо, декількох виходів Q_i $(i = 1, n)$. Аналітична модель пошуку оптимального розв'язку в кіберпросторі по вектор-запросу оперує п'ятьма паралельними логічними операціями, що виконуються послідовно:

$$Q^i = \underset{j=1,k}{\vee}\{[\underset{p=1,n}{\wedge}\ \underset{s=1,n}{slc}(m \underset{r=1,n}{\oplus} M_i)] \underset{i=1,n}{\oplus} M_i\}.$$

Структурна модель векторного логічного процесора, відповідна аналітичної моделі формування оптимального розв'язку, має вигляд, зображений на рис. 9.3.

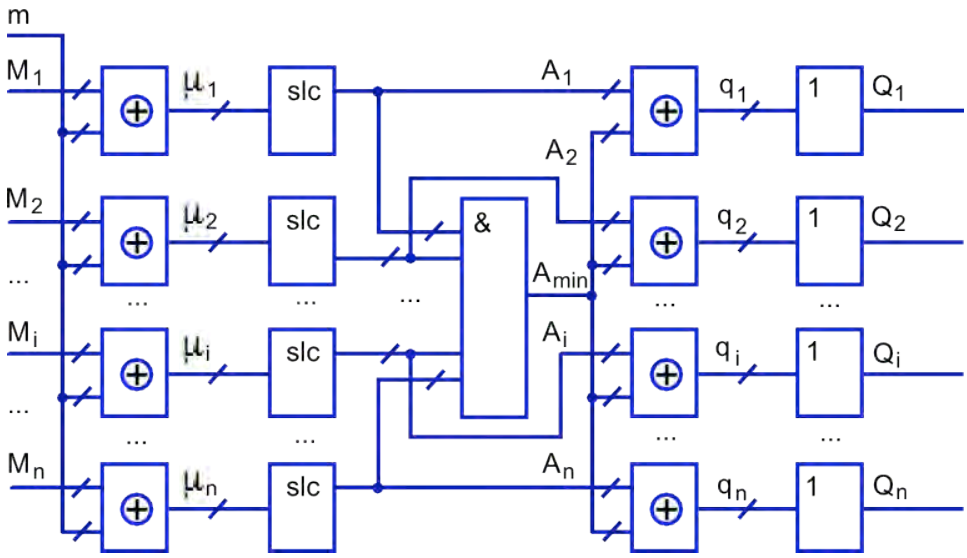


Рисунок 9.3 - Векторно-логічний мультипроцесор

Структурна модель векторного логічного процесора. Три приклади обчислення станів виходів векторно-логічного процесора, як реакції на запит m для матриці M, що складається з двох вектор-рядків, подані нижче. Перша фаза ілюструє формування векторів одиничних значень, зсунутих вліво: відповідна аналітичної моделі формування оптимального розв'язку, має вигляд, зображений на рис. 9.3 $\{A_1, A_2\} \in A$:

$M_1 =$	1	0	1	0	0	1	0	1
$M_2 =$	0	1	0	0	1	0	1	1
$m =$	1	1	1	1	1	1	0	0
$\mu_1(m \in M_1) = m \oplus M_1$	0	1	0	1	1	0	0	1
$\mu_2(m \in M_2) = m \oplus M_2$	1	0	1	1	0	1	1	1
$A_1 = slc(\mu_1)$	1	1	1	1	0	0	0	0
$A_2 = slc(\mu_2)$	1	1	1	1	1	1	0	0

$M_1 =$	0	0	0	1	0	0	0	1
$M_2 =$	0	1	0	0	1	1	0	0
$m =$	1	1	1	1	1	1	0	0
$\mu_1(m \in M_1) = m \oplus M_1$	1	1	1	0	1	1	0	1
$\mu_2(m \in M_2) = m \oplus M_2$	1	0	1	1	0	0	0	0
$A_1 = slc(\mu_1)$	1	1	1	1	1	1	0	0
$A_2 = slc(\mu_2)$	1	1	1	0	0	0	0	0

$M_1 =$	1	0	1	0	0	1	1	1
$M_2 =$	0	1	0	0	1	0	0	1
$m =$	1	1	1	1	1	1	0	0
$\mu_1(m \in M_1) = m \oplus M_1$	0	1	0	1	1	0	1	1
$\mu_2(m \in M_2) = m \oplus M_2$	1	0	1	1	0	1	0	1
$A_1 = slc(\mu_1)$	1	1	1	1	1	0	0	0
$A_2 = slc(\mu_2)$	1	1	1	1	1	0	0	0

Друга фаза ілюструє формування станів виходів процесора: $\{Q_1, Q_2\} \in Q$, де в першому і другому випадках існує тільки один

«переможець» з нульом на виході, а в третьому випадку - обидві вектор-рядки матриці M є оптимальними розв’язками для запросу m:

$A_1 =$	1	1	1	1	0	0	0	0	$A_1 =$	1	1	1	1	1	1	0	0	$A_1 =$	1	1	1	1	1	0	0	0
$A_2 =$	1	1	1	1	1	1	0	0	$A_2 =$	1	1	1	0	0	0	0	0	$A_2 =$	1	1	1	1	1	0	0	0
$A_{min} = M_1 \wedge M_2$	1	1	1	1	0	0	0	0	$A_{min} = M_1 \wedge M_2$	1	1	1	0	0	0	0	0	$A_{min} = M_1 \wedge M_2$	1	1	1	1	1	0	0	0
$q_2 = (M_1 \wedge M_2) \oplus M_2$	0	0	0	0	1	1	0	0	$q_2 = (M_1 \wedge M_2) \oplus M_2$	0	0	0	0	0	0	0	0	$q_2 = (M_1 \wedge M_2) \oplus M_2$	0	0	0	0	0	0	0	0
$q_1 = (M_1 \wedge M_2) \oplus M_1$	0	0	0	0	0	0	0	0	$q_1 = (M_1 \wedge M_2) \oplus M_1$	0	0	0	1	1	1	0	0	$q_1 = (M_1 \wedge M_2) \oplus M_1$	0	0	0	0	0	0	0	0
$Q_2 = \overset{k}{\underset{i=1}{\vee}} q_i^2 = 1$									$Q_2 = \overset{k}{\underset{i=1}{\vee}} q_i^2 = 0$	w	i	n	n	e	r			$Q_2 = \overset{k}{\underset{i=1}{\vee}} q_i^2 = 0$	w	i	n	n	e	r		
$Q_1 = \overset{k}{\underset{i=1}{\vee}} q_i^1 = 0$	w	i	n	n	e	r			$Q_1 = \overset{k}{\underset{i=1}{\vee}} q_i^1 = 1$									$Q_1 = \overset{k}{\underset{i=1}{\vee}} q_i^1 = 0$	w	i	n	n	e	r		

Інтерес становить формування запитів в багатозначному алфавіті (наприклад, Кантора) опису змінних взаємодіючих векторів. На перший погляд, існують проблеми підрахунку відстані між запитом та інформаційним компонентом кіберпростору з недвійковим кодуванням. Але якщо багатозначні символи булеана примітивів, що беруть участь у формуванні векторів, позначити двійковими кодами (0 = 10, 1 = 01, X = 11, ∅= 00), то відстань «запит - компонент» можна оцінювати за допомогою раніше описаної процедури зсуву всіх одиниць вліво з ущільненням:

$M_1 =$	1	0	1	0	0	1	0	1	$M_1 =$	0	0	0	0	0	0	0	1	$M_1 =$	X	0	1	X	0	X	1	X
$M_2 =$	0	1	0	0	1	0	1	1	$M_2 =$	0	1	1	1	1	1	0	0	$M_2 =$	0	X	X	0	X	0	X	1
$m =$	1	X	1	1	X	1	X	0	$m =$	X	X	1	1	X	1	X	X	$m =$	X	X	1	1	X	X	0	0
$\mu_1(m \in M_1) = m \oplus M_1$	∅	1	∅	X	1	∅	1	X	$\mu_1(m \in M_1) = m \oplus M_1$	1	1	X	X	1	X	1	0	$\mu_1(m \in M_1) = m \oplus M_1$	∅	1	∅	0	1	∅	X	1
$\mu_2(m \in M_2) = m \oplus M_2$	X	0	X	X	0	X	0	X	$\mu_2(m \in M_2) = m \oplus M_2$	1	0	∅	∅	0	∅	1	1	$\mu_2(m \in M_2) = m \oplus M_2$	1	∅	0	X	∅	1	1	X
$A_1 = slc(\mu_1)$	11	11	11	1	0	0	0	0	$A_1 = slc(\mu_1)$	11	11	11	11	11	11	1	0	$A_1 = slc(\mu_1)$	11	11	11	0	0	0	0	0
$A_2 = slc(\mu_2)$	11	11	11	11	11	11	1	0	$A_2 = slc(\mu_2)$	11	11	1	0	0	0	0	0	$A_2 = slc(\mu_2)$	11	11	11	11	0	0	0	0

Переможцем в трьох, наведених вище, номинаціях становяться відповідно: M_1, M_2, M_1, як такі, що мають мінімальне число одиниць або максимальну кількість нульових координат. Таким чином, не існує принципових обмежень для оцінювання взаємодії об'єктів в кіберпросторі шляхом використання нечисельних метрик, що виключають арифметичні операції. Більш того, всі відстані в інформаційному світі можна вимірювати

за допомогою xor-операції або симетричної різниці, які забезпечують вибір кращого розв’язку на основі векторно-логічних критеріїв якості взаємодії.

Наступний приклад ілюструє роботу мультипроцесора в багатозначному алфавіті опису логічних змінних, орієнтованому на компетентнісне рейтингування учнівської молоді. Нехай є група студентів, яка отримала сесійні оцінки з восьми іспитів у метриці A, B, C, D, які кодуються відповідними векторами: 1000, 1100, 1110, 1111. Необхідно визначити кращого студента, який інтегрально отримав максимальні бали за сесію. Результати обчислень подані нижче:

$M_1 =$	A	B	D	D	B	A	C	C
$M_2 =$	C	C	D	A	B	B	A	D
$M_3 =$	B	C	C	B	A	A	C	D
$m =$	A	B	C	A	A	A	A	C
$\mu_1(m \in M_1) = m \oplus M_1$	0000	0000	0001	0111	0100	0000	0110	0000
$\mu_2(m \in M_2) = m \oplus M_2$	0110	0010	0001	0000	0100	0100	0000	0001
$\mu_3(m \in M_3) = m \oplus M_3$	0100	0010	0000	0100	0000	0000	0110	0001
$A_1 = slc(\mu_1)$	1111	111						
$A_2 = slc(\mu_2)$	1111	111						
$A_3 = slc(\mu_3)$	1111	11						
$M_3 =$	w	i	n	n	e	r		

$M_1 =$	A	B	D	D	B	A	C	C
$M_2 =$	C	C	D	A	B	B	A	D
$M_3 =$	B	C	C	B	A	A	C	D
$m =$	A	A	A	A	A	A	A	A
$\mu_1(m \in M_1) = m \oplus M_1$	0000	0100	0111	0111	0100	0000	0110	0110
$\mu_2(m \in M_2) = m \oplus M_2$	0110	0110	0111	0000	0100	0100	0000	0111
$\mu_3(m \in M_3) = m \oplus M_3$	0100	0110	0110	0100	0000	0000	0110	0111
$A_1 = slc(\mu_1)$	1111	1111	1111	1				
$A_2 = slc(\mu_2)$	1111	1111	1111	1				
$A_3 = slc(\mu_3)$	1111	1111	111					
$M_3 =$	w	i	n	n	e	r		

Тут в лівій таблиці вектор-еталон m приведений до найкращих фактичних оцінок, отриманих студентами з кожного іспиту. Права таблиця оперує вектором-еталоном з теоретично можливими кращими (A = 1000) значеннями тестування знань. В обох випадках інтегральний критерій якості сесії визначає кращим студента під номером 3. Для кодування всіх п'яти градацій болонської метрики оцінювання знань пропонується використовувати і нульову комбінацію: A = 0000, B = 1000, C = 1100, D = 1110, E = 1111. У цьому випадку дві наступні таблиці дають аналогічний попередньому результат вибору кращого студента за підсумками сесії:

$M_1 =$	A	B	D	D	B	A	C	C
$M_2 =$	C	C	D	A	B	B	A	D
$M_3 =$	B	C	C	B	A	A	C	D
$m =$	A	B	C	A	A	A	A	C
$\mu_1(m \in M_1) = m \oplus M_1$	0000	0000	0010	1110	1000	0000	1100	0000
$\mu_2(m \in M_2) = m \oplus M_2$	1100	0100	0010	0000	1000	1000	0000	0010
$\mu_3(m \in M_3) = m \oplus M_3$	1000	0100	0000	1000	0000	0000	1100	0100
$A_1 = slc(\mu_1)$	1111	111						
$A_2 = slc(\mu_2)$	1111	111						
$A_3 = slc(\mu_3)$	1111	11						
$M_3 =$	w	i	n	n	e	r		

$M_1 =$	A	B	D	D	B	A	C	C
$M_2 =$	C	C	D	A	B	B	A	D
$M_3 =$	B	C	C	B	A	A	C	D
$m =$	A	A	A	A	A	A	A	A
$\mu_1(m \in M_1) = m \oplus M_1$	0000	1000	1110	1110	1000	0000	1100	1100
$\mu_2(m \in M_2) = m \oplus M_2$	1100	1100	1110	0000	1000	1000	0000	1110
$\mu_3(m \in M_3) = m \oplus M_3$	1000	1100	1100	1000	0000	0000	1100	1110
$A_1 = slc(\mu_1)$	1111	1111	1111					
$A_2 = slc(\mu_2)$	1111	1111	1111					
$A_3 = slc(\mu_3)$	1111	1111	111					
$M_3 =$	w	i	n	n	e	r		

Якщо примітиви оцінок нерівнозначні за вагою в метриці порівняння, то їх не можна позначати унітарними кодами, які виконують лише роль ідентифікаторів елементів в універсальній множині рівних примітивів. Тому для позначення ваг в кодах оцінок був використаний фактор кількості одиниць. Тим не менш, подальші дії, орієнтовані на визначення інтегральної якості знань студентів для вибору кращого з них шляхом порівняння з ідеальним результатом, не пов'язані з якими-небудь арифметичними операціями, а використовують тільки логічні процедури. Стратегічно завдання лінійної обчислювальної складності вирішувалася в рамках наступного формулювання - знайти студента, який має мінімальну відстань до заздалегідь відомого ідеального результату, у вигляді вектор-еталона m. Альтернативна стратегія передбачає пошук кращого з n студентів шляхом послідовного порівняння кожного з них один з одним, на що буде витрачено значно більше часу, оскільки обчислювальна складність такої процедури - $(n^2/2)-n$. Для ранжування всіх студентів відносно ідеального результату необхідно виконати $(n^2/2)-n$ векторно-логічних операцій. Після визначення на кожному кроці кращого студента відповідний йому рядок матриці M (екзаменаційної компетенції академічної групи учнів) слід вилучити з подальшого розгляду шляхом занесення в її розряди одиничних значень сигналів.

Для ефективної роботи логічного мультипроцесора необхідно сформувати M-матрицю (рис. 9.4) можливих варіантів вирішення проблеми,

яка, зокрема, може бути продуктом застосування пошукової системи Google (Hadoop) до кіберпростору Internet (big data), використовуваної для грубої і широкої вибірки, коли кількість знайдених інформаційних фрагментів досягає сотень або тисяч варіантів. Потім настає черга функціонування мультипроцесора, що формує точний розв’язок за запитом m, яке має бути збережено в структурованій, спеціалізованій частини кіберпростору для подальшого багаторазового використання. Тому входом і виходом логічного мультипроцесора слід вважати форми кіберпростору: Internet of Things, Big Data, Cyber-Physical Systems. Ринкова привабливість запропонованого мультипроцесора полягає в можливості його використання для: підвищення якості та швидкодії пошукових процедур в big data, створення вбудованих автоматичних, автономних систем діагностування та відновлення працездатності, засобів цілевказівки і розпізнавання образів. Типовою для кіберфізічних систем, що використовують інформаційний простір, є функціональність, коли за запитом виникають багатоальтернативні варіанти його виконання у векторно-логічній формі опису компонентів кіберпростору (суб'єктів, процесів або явищ), які необхідні для управління соціальними, біологічними та неприродними виробничо-технологічними процесами без участі людини.

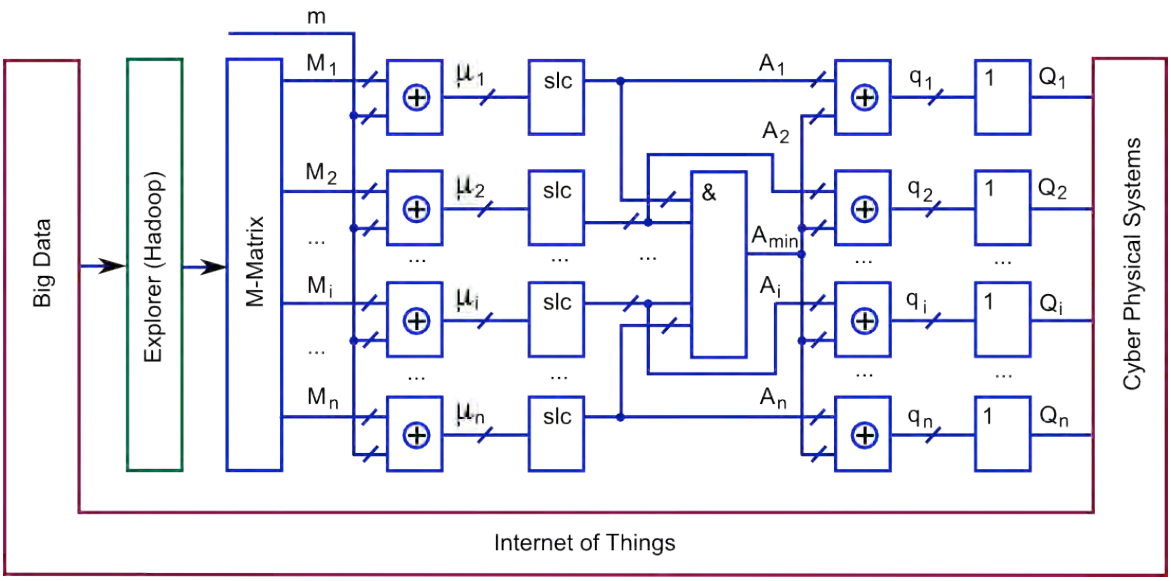


Рисунок 9.4 - Взаємодія мультипроцесора з кіберпростором

Перспективний напрямок майбутнього дослідження - «Образні транзакції великих даних» - «Big Data Image Transactions». Образна культура мислення, спілкування або транзакцій: переклад компонентів опису об'єкта з часу у простір. Чи можливо таке? «Червоний (100), синій (010), зелений (001)» - записано в часі, якщо слова поєднати, замінивши їх образами і виключивши час, вийде принципово новий, але вже згенерований образ білого кольору (111). «Мама (100) мила (010) раму (001)», - якщо позначити слова примітивами, то їх суперпозиція дає код паралельного образу (111), який легко уявити фотографією жінки зі щіткою, що миє вікно. Художник Леонардо да Вінчі створює паралельний образ Мони Лізи шляхом суперпозиції послідовних візуальних фрагментів. Але результат не має параметру часу і тим він цінний. Якщо розбити його картину на суперпозиційні фрагменти, то вона втратить свою привабливість. Джоконду неможливо описати словами! Будь-яка картина краще її текстуального опису. Тим не менш, на ринку електронних технологій вже існують автоматичні програмні додатки як прямого синтезу «вербальний опис - картина», так і зворотнього аналізу «картина - вербальний опис». Тут можна починати з голосового та/або мануального синтезу та аналізу простих геометричних фігур (трикутник, квадрат, коло).

Ще один напрямок дослідження - «Образний транзакційний процесор», «Image Transactions Processor». Мається на увазі процесор, який створює білатеральну симетрію «образ - транзакція - образ», яка розбивається на два типи транзактору: «вербальний (мануальноий, голосовий) опис - транзакція-синтез - образ» і «образ - транзакція-аналіз - вербальний (мануальний, голосовий) опис».

Мова послідовних символів, використовувана людством, недосконала за своїми часовими витратами, пов'язаними з транзакціями інформації між приймачем і передавачем. Тому вчені змушені сьогодні констатувати факт, що ринку необхідний дізраптор нової культури паралельного, суперпозіційного, образного мислення, спілкування, генерування, прийому-

передачі та сприйняття інформації і дійсності без параметра часу. Слід перекладати кіберпростір в паралельні образи для здійснення транзакцій на основі квантових структур даних (властивості: суперпозиція, переплутування, паралелізм). Цим можна істотно заощадити пам'ять, час навчання, прийому-передачі та сприйняття інформації в безпосередньо взаємодіючих парах: «людина - комп'ютер», «комп'ютер - комп'ютер», «людина - людина» без традиційних інтерфейсів (клавіатура, голос, тактильність). Історія знає аналоги у формі ієрогліфів, клинопису, настінних малюнків, де наші пращури намагалися прибрати неістотність часу при описі фактів минулого дійсності, щоб мінімізувати час нашої уваги на пізнання сутності, а не на процес, який вже не дуже цікавий. Образ ефективніше для сприйняття, ніж вербальний опис, тому PR-акції завжди оперують фотографіями з короткими слоганами. Образне мислення - компресія або стиснення процесу в одну фотографічну мить або явище. Фільм - в картину, слова і пропозиції - в образ. Послідовність логічних елементів в один інформаційний «квантовий» примітив - кубіт.

9.4 Висновки до розділу 9

Запропоновано алгебрологічну систему, що включає матричну структуру, яка призначена для використання в інфраструктурі векторного логічного аналізу, заснованого на застосуванні мультипроцесорного персонального віртуального кіберкомпьютера.

Матрична алгебра логіки - сукупність носія та сигнатури, де носій складається з двійкових таблиць, розмірністю pxn, а сигнатура задає набір логічних операцій І-АБО-НІ над компонентами носія. На відміну від відомих операцій в теорії матриць, логічні множення, додавання в матричній алгебрі логіки виконуються покомпонентно. На основі введених операцій виконано доведення основних тотожностей і законів матричної логіки, за аналогією із системою аксіом і тотожностей булевої алгебри.

Практична значимість запропонованих результатів досліджень полягає в орієнтації введеної алгебри на ефективний аналіз матричних логічних процедур за допомогою мультипроцесорних віртуальних обчислювальних структур.

10 ІНШІ РЕЗУЛЬТАТИ, ОТРИМАНІ ТА РЕАЛІЗОВАНІ В РАМКАХ НДР

10.1 Результативність виконання НДР

Таблиця 10.1 – Кількісні показники виконання НДР

№ з/п	Критерії	Заплановано (відповідно до запиту)	Виконано (за резуль-татами НДР)	% викона-ння
		кількість	кількість	%
1.	**Публікації колективу виконавців НДР:**			
	1.1. Статті у журналах, що входять до наукометричних баз даних (*).	5	8	160%
	1.2. Публікації в матеріалах конференцій, що входять до наукометричних баз даних (*).	22	40	182%
	1.3. Статті у журналах, що включені до переліку наукових фахових видань України.	25	32	145%
	1.4. Публікації у матеріалах конференцій, тезах доповідей та виданнях, що не включені до переліку наукових фахових видань України.	38	68	179%
	1.5. Монографії, опубліковані за рішенням Вченої ради ВНЗ (наукової установи).	Не заплановано	1	100% перевиконання
	1.6. Підручники, навчальні посібники з грифом МОН України.	Не заплановано	–	–
	1.7. Навчальні посібники без грифу МОН України.	Не заплановано	1	100% перевиконання
	1.8. Словники, довідники.	Не заплановано	–	–
2.	**Підготовка наукових кадрів:**			
	2.1. Захищено докторських дисертацій за тематикою НДР.	1	– З пошукачем Гузь О.О. з Донецької автодорожньої академії втрачено зв’язок	–
	2.2. Подано до розгляду у спеціалізовану вчену раду докторських дисертацій за тематикою НДР.	Не заплановано	2	–
	2.3. Захищено кандидатських дисертацій за тематикою НДР.	3	2	67%**
	2.4. Подано до розгляду у спеціалізовану вчену раду кандидатських дисертацій за	Не заплановано	2	200% перевиконання

№ з/п	Критерії	Заплановано (відповідно до запиту)	Виконано (за резуль-татами НДР)	% викона-ння
		кількість	кількість	%
	тематикою НДР. 2.5. Захищено магістерських робіт за тематикою НДР.	6	24	400%
3.	**Охоронні документи на об'єкти права інтелектуальної власності, які створено за тематикою НДР:**	1	2	200%
	3.1. Отримано патентів (свідоцтв авторського права) України.	Не заплановано	–	–
	3.2. Подано заявок на отримання патенту України.	Не заплановано	–	–
	3.3. Отримано патентів (свідоцтв авторського права) інших держав.	Не заплановано	–	–
	3.4. Подано заявок на отримання патенту інших держав.			
4.	**Участь з оплатою у виконанні НДР:**			
	4.1. Студентів.	Не заплановано	–	–
	4.2. Молодих учених та аспірантів.	1	2	200%

** – показник з захисту кандидатських дисертацій можна вважати виконаним у повному обсязі, оскільки одну з поданих до захисту робіт вже рекомендовано до захисту та опубліковано об'яву.

10.2 Монографії та навчальні посібники

Монографії:

Хаханов В.І. Smart Cyber University. 2014. – 256 с. (підготовлено до друку).

Підручники та посібники:

Проектування та тестування цифрових систем на кристалах / В.І. Хаханов, Є.І. Литвинова, С.В. Чумаченко – Харків: ХНУРЕ.– 2014.– 484 с. (підготовлено до друку).

10.3 Патенти

Отримано 2 патенти:

1) Патент на корисну модель № 83310 «Реверсивний регістр зсуву». Какурін М.Я., Хаханов В.І., Литвинова Є.І., Вареца В.В., Макаренко Г.М. 10.09.2013.

2) Патент на корисну модель № 90665. «Реверсивний регістр зсуву». Какурін М.Я., Хаханов В.І., Литвинова Є.І., Вареца В.В., Макаренко Г.М. 10.06.2014.

10.4 Наукові статті

1. Гузь О.А., Хаханов В.И., Мурад Али А., Baghdad Ammar Avni Abbas, Хаханова И.В. Метрика и критерии анализа киберпространства // АСУ и приборы автоматики. 2011. Вып. 156. С.90-98.

[http://www.ewdtest.com/ri/asu_2011/ASU_156_2011.pdf]

2. Гузь О.А., Хаханов В.И., Мурад Али А., Литвинова Е.И., Хаханова И.В. Квантовые модели вычислительных процессов // Радиоэлектроника и информатика. 2011. №3. С. 40-60.

3. Гузь О.А., Хаханов В.И., Мурад Али А., Литвинова Е.И., Хаханова И.В., Baghdad Ammar Avni Abbas. Квантовые модели данных и вычислительных процессов // Радіоелектронні і комп'ютерні системи. №6(58). 2012.С. 47-52.

4. Хаханов В.И., Чумаченко С.В., Гузь О.А., Литвинова Е.И. Инфраструктура диагностирования программно-аппаратных систем // Радіоелектроніка. Інформатика. Управління. № 1 (26). 2012. С. 134-140.

5. Горобец О.А., Чумаченко С.В., Хаханов В.И., Мурад Али Аббас. Генерирование булеана для синтеза квантового процессора // АСУ и приборы автоматики. 2011.Вып. 157. С.4-16.

[http://www.ewdtest.com/ri/asu_2011/ASU_157_2011.pdf]

6. Хаханов В.И., Чумаченко С.В., Литвинова Е.И., Гузь О.А. Инфраструктура диагностирования программно-аппаратных систем // Прогресивні інформаційні технології. 2012. С. 134-140.

7. Aleksandr Mischenko, Vladimir Hahanov, Svetlana Chumachenko. Testing and Diagnosis of Bad Messages in Individual Cyberspace // Radioelektroniks and informatics. 2012. № 1. P. 9-16.

8. Hahanova A.V., Barannik V.V. The Method of Binary Structures Compression on Basis of Cascade Encoding in Telecommunication Systems // Radioelektroniks and informatics. 2012. № 1. P. 41-44.

9. Murad Ali Abbas, Горобец А.А., Скоробогатый М., Хаханов В.И., Чумаченко С.В. Модели анализа эффективности вычислительных структур // Радиоэлектроника и информатика. 2012. №3. С. 4-11.

10. Шахов Д.В., Гузь О.А., Хаханов В.И., Бондаренко М.Ф., Энглези И.П., Убар Р., Лобур М.В., Меликян В., Дохов А.И., Бодянский Е.В., Филатов В.А. Тевяшев А.Д., Ткаченко В.Ф., Филиппенко О.И., Саатчян А.Г., Чумаченко С.В., Литвинова Е.И., Хаханова И.В., Белоус Н.В., Полетайкин А.Н., Чугуров И.Н. Зеленая волна – облако мониторинга и управления дорожным движением (Green Wave Traffic on Cloud) // АСУ и приборы автоматики. 2012. Вып. 160. С.4-21.

[http://www.ewdtest.com/ri/asu_2012/ASU_160_2012.pdf]

11. Хаханов В.И., Anders Carlsson, Чумаченко С.В. Инфраструктура PENTESTING и управления уязвимостью // АСУ и приборы автоматики. 2012. Вып. 160. С.36-54.

[http://www.ewdtest.com/ri/asu_2012/ASU_160_2012.pdf]

12. Бутенко С.А., Хаханов В.И., Anders Carlsson, Чумаченко С.В. Модели управления уязвимостью // АСУ и приборы автоматики. 2012. Вып. 161. С. 10-24. [http://www.ewdtest.com/ri/asu_2012/ASU_161_2012.pdf]

13. Мизь В.А., Хаханова А.В. Анализ систем автоматизированного мониторинга автомобильного транспорта и управления дорожным движением //

АСУ и приборы автоматики. 2012. Вып. 161. С.25-31. [http://www.ewdtest.com/ri/asu_2012/ASU_161_2012.pdf]

14. Хаханов В.И., Энглези И.П., Литвинова Е.И., Чумаченко С.В., Гузь О.А., Хаханова А.В. Облачная инфраструктура мониторинга и управления дорожным движением // Радіоелектронні і комп'ютерні системи. №5(64). 2013. С. 106-111.

15. Зиарманд А.Н**.** Теоретическая суть проекта «SMART ROADS» // АСУ и приборы автоматики. 2013. Вып. 162. С.28-34. [http://www.ewdtest.com/ri/asu_2013/ASU_163_2013.pdf]

16. Хаханов В.И., Baghdadi Ammar Awni Abbas, Чумаченко С.В., Шкиль А.С. Квантовые структуры для тестирования цифровых устройств // АСУ и приборы автоматики. 2013. Вып. 163. С.4-17. [http://www.ewdtest.com/ri/asu_2013/ASU_163_2013.pdf]

17. Хаханов В.И., Чумаченко С.В., Литвинова Е.И. Состояние IT-рынка Украины (аналитический обзор) // АСУ и приборы автоматики. 2013. Вып. 163. С.22-46. [http://www.ewdtest.com/ri/asu_2013/ASU_163_2013.pdf]

18. Хаханов В.И., Чумаченко С.В., Литвинова Е.И., Мищенко А.С. Развитие киберпространства и информационная безопасность // Радіоелектроніка, інформатика, управління. 2013. №1. С. 151-157.

19. Хаханов В.И., Baghdadi Ammar Awni Abbas, Литвинова Е.И., Хаханова И.В. Квантовые структуры данных вычислительных систем // АСУ и приборы автоматики. 2013. Вып. 164. С.4-17.

[http://www.ewdtest.com/ri/asu_2013/ASU_164_2013.pdf]

20. Багдади Аммар Авни Аббас (Baghdadi Ammar Awni Abbas), Хаханов В.И., Литвинова Е.И., Бутенко С.А., Чумаченко С.В. Квантовые модели диагностирования цифровых систем // Радиоэлектроника и информатика. 2013. № 2. С. 35-43. [http://www.ewdtest.com/ri/ri_2013_2/RI_13_2.pdf]

21. Hahanova I.V. Method of Pentest Synthesis and Vulnerability Detection // Radioelektroniks and informatics. 2012. № 4. P.68-73. [http://www.ewdtest.com/ri/ri_2012_4/Maket_RI_2012_4.pdf]

22. Багдади Аммар Авни Аббас (Baghdadi Ammar Awni Abbas), Хаханов В.И., Чумаченко С.В., Меликян В. Кубитные технологии анализа и диагностирования цифровых устройств // Радиоэлектроника и информатика. 2013. №3. С.4-15.

[http://www.ewdtest.com/ri/ri_2013_3/RI_13_3-new-ri.pdf]

23. Врублевский Н.Н., Хаханов В.И., Baghdadi Ammar Awni Abbas, Литвинова Е.И., Хаханова И.В. Кубитные структуры данных вычислительных устройств // АСУ и приборы автоматики. 2013. Вып. 164. С.4-19.

24. Хаханов В.И., Литвинова Е.И., Чумаченко С.В., Филиппенко О.И. Интеллектуальное облако управления движением (Smart Cloud Traffic Control) // Радиоэлектроника и информатика. 2013. № 2. С. 67-76.

25. Зайченко С.А., Лештаева И.Б., Варченко В.Г., Зенович И. Модель гетерогенного покрытия для эффективной функциональной верификации цифровых систем // АСУ и приборы автоматики. 2013. Вып. 165. С.4-14.

26. Хаханов В.И., Baghdadi Ammar Awni Abbas, Чумаченко С.В., Шкиль А.С., Меликян Вазген. Кубитные технологии анализа и диагностирования цифровых устройств // Радиоэлектроника и информатика. 2013. № 3. С. 36-45.

27. Зайченко С.А, Александров В.И., Белоус В.В., Березин Н.П. Методы ранжирования тестов для моделей цифровых систем по гетерогенным метрикам покрытия // Радиоэлектроника и информатика. 2013. № 4. С. 51-60.

28. Зайченко С.А. Baghdadi Ammar Awni Abbas, Хаханов В.И., Литвинова Е.И. Диагностирование HDL-моделей систем на кристаллах // Радиоэлектроника и информатика. 2013. № 4. С. 60-68.

29. Vladimir Hahanov, Eugeniya Litvinova,Vladimir Obrizan, Igor Yemelyanov. Matrix-Model for Diagnosing SoC HDL-Code // Radioelectronics & Informatics. 2013. № 1. P. 12-19.

30. Кривуля, Г. Ф. Анализ корректности продукционных правил в системах нечеткого логического вывода с использованием квантовых моделей / Г. Ф. Кривуля, А. С. Шкиль, Д. Е. Кучеренко // АСУ и приборы автоматики. 2013. Вып. 165. С. 42-53.

31. Хаханов В.И., Обризан В.И., Мищенко А.С., Филиппенко И.В. Киберфизические системы как технологии киберуправления (аналитический обзор) // Радиоэлектроника и інформатика. 2014. №1. С. 38-43.

32. Ларченко Л.В., Вареца В.В., Ларченко Б.Д., Макаренко А.Н.. Сравнительная оценка быстродействия регистров-компакторов синхронного и асинхронного типов // АСУ и приборы автоматики. Вып. 166. 2014. 6 с.

33. *Хаханов В.И., Ваджеб Гариби, Литвинова Е.И., Шкиль А.С. Кубитные структуры данных вычислительных устройств // Электронное моделирование. 2014. №6. 24 с.

34. *Murad Ali Abbas, Hahanov V., Litvinova E., Gharibi W. Qubit models for SoC Synthesis // Parallel and cloud computing. USA. 2012. Vol.1. Issue 1. P. 16-20.
[http://www.vkingpub.com/UploadFiles/2014-06/373/20140623105356306 58.pdf]

35. *Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И. Структура логического ассоциативного мультиплексора // Автоматика и телемеханика. Россия. 2012. С. 71-92.

36. *Шахов Д.В., Хаханов В.И., Меликян В.Ш, Саатчян А.Г. «Зеленая волна» – облако мониторинга и управления дорожным движением // Армения. Вестник «Информационные технологии, электроника, радиотехника». Вып. 16. 2013. С. 53-60.

37. *Krivoulya, G. F. Analysis of production rules in expert systems of diagnosis / G. F. Krivoulya, A. S. Shkil, D. Ye. Kucherenko // Automatic control and computer sciences. 2013. Vol. 47 (No. 6). P. 331-341.

38. *Мищенко А., Емельянов И., Тамер Бани Амер, Хаханов В., Чумаченко С., Литвинова Е. Облачное управление физическими и кадровыми ресурсами (Cloud-driven Cyber Managing Resources) // Australian science review. Австралія. 2014. 21 с.

39. *Мизь В., Зиарманд А., Хаханов В., Гариби Ваджеб, Чумаченко С., Литвинова Е. Интернет-слой для киберуправления транспортом (Internet-

driven Cyber Control of Traffic) // Австралія. Australian science review. Австралія. 2014. 12 с.

40. *Багдади А.А., Хаханов В.И., Меликян В.Ш., Литвинова Е.И. Квантовое моделирование и тестирование вычислительных устройств // Вестник государственного инженерного университета Армении. 2014. Вып.17. №1. С.9-19.

10.5 Тези доповідей на конференціях

1. Степанова Ю.В., Пащенко А., Мостовая К.Л. (научный руководитель – д.т.н., проф. Литвинова Е.И.). Модели генерации тестов для функциональных элементов // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 17-19 апреля 2012. Ч. 5. С. 40-41.

2. Василенко В.А., Щербин Д.А. (научный руководитель – д.т.н., проф. Литвинова Е.И.). Синтез тестов $F \oplus L$ методом // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С.42-43.

3. Батуров М.А., Бражников А.Н. (научный руководитель – д.т.н., проф. Литвинова Е.И.). Аппарат булевых производных для синтеза тестов // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С. 44-45.

4. Карпинский А.М., Максимов М.М. (научный руководитель – д.т.н., проф. Хаханов В.И.) Структуры данных кубитные вычислительные процессы // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С.46-47.

5. Пырлык А.Т., Сандркин Д.Л. (научный руководитель – д.т.н., проф. Хаханов В.И.) Квантовый процессор для дискретной оптимизации // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С.48-49.

6. Адамов А.С., Хаханова Ю.В. (научный руководитель – д.т.н., проф. Чумаченко С.В.) Функциональная модель индивидуального киберпространства пользователя // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч.5. 17-19 апреля 2012. С.50-51.

7. Мищенко А.С., Зацарный А.В., Гончаров Е.В. (научный руководитель – д.т.н., проф. Чумаченко С.В.). Комплексная система защиты индивидуального киберпространства // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С. 62-63.

8. Адамов А.С., Щербин Д.А. (научный руководитель – д.т.н., проф. Чумаченко С.В.). Модель поиска вредоносного кода в программных продуктах // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С.52-53.

9. Дементьев С.П. (научный руководитель – д.т.н., проф. Кривуля Г.Ф.). Задача отражения результатов логического синтеза // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 17-19 апреля 2012. Ч. 5. С.19-20.

10. Лобода А.В., Стец С.А. (научный руководитель – к.т.н., доцент Сыревич Е.Е.) Проектирование «умного дома» с помощью беспроводных сетей XBEE // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С.23-24.

11. Лобода А.В., Стец С.А. (научный руководитель – к.т.н., доцент Сыревич Е.Е.). Сбор информации о состоянии жилой комнаты с использованием беспроводных технологий // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 17-19 апреля 2012. Ч. 5. С.27-28.

12. Шеремет Е.В., Арефьев А.А. (научный руководитель – к.т.н., доцент Сыревич Е.Е.). Система управления освещением // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 17-19 апреля 2012. С.31-32.

13. Мова А.Ю., Усиченко Р.И. (научный руководитель – к.т.н., доцент Бабич А.В.). Введение диагностического узла в модель обратной связи RTCP для централизованной архитектуры ВКС // Материалы XVI Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 17-19 апреля 2012. Ч 5. С.73-74.

14. Mova A., Usichenko R., Babich A. Extended procedures of RTCP data transmission in the videoconference with the centralized // Матеріали XI Міжнародної конференції TCSET 2012, присвяченої 60-річчю заснування радіотехнічного факультету у Львівській політехніці.21-24 лютого. 2012. Львів – Славське. С.367.

15. Syrevitch Ye., Loboda A., Stets S. Implementation of discontinuous heating for heat-saving in «Intelligent Home» // Матеріали XI Міжнародної конференції TCSET 2012, присвяченої 60-річчю заснування радіотехнічного факультету у Львівській політехніці. 21-24 лютого 2012. Львів – Славське. С.357.

16. Гузь О.А. Хаханов В.И., Литвинова Е.И., Чумаченко С.В. Кубит-процессор для задач оптимального покрытия // Материалы международной научно-технической конференции «Информационные системы и технологии». Морское-Харьков. 22-29 сентября 2012. С.72-73.

17. Бондаренко М.Ф., Хаханов В.И. Квантовые технологии реализации мозгоподобных вычислительных структур акад. В.М. Глушкова // Матеріали XIX Міжнародної конференції з автоматичного управління. Київ. 26-28 вересня 2012. С. 32-33.

18. *Gorobets A., Chugurov I., Scherbin D. Chumachenko S. Dijkstra Algorithm for cyber structures analysis // Матеріали XII Міжнародної науково-технічної конференції CADSM 2013 «Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці». 19-23 лютого. 2013. Львів – Поляна. С.61-65.

19. *Shakhov D. Hahanov V., Gharibi W., Lobur M., Litvinova E., Chumachenko S., Saatchyan A., Guz O., Filippenko O., Poletaykin A. Cloud

«Green Wave traffic monitoring and control» // Матеріали XII Міжнародної науково-технічної конференції CADSM 2013 «Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці». 19-23 лютого 2013. Львів – Поляна. С.120-126.

20. *Jerbarov D., Abeid A.M. Hahanov V., Hayford A., Ahmetoglu A.H., Oghumu S. Pentesting and vulnerability diagnosis // Матеріали XII Міжнародної науково-технічної конференції CADSM 2013 «Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці». 19-23 лютого 2013. Львів – Поляна. С.127-131.

21. Мизь В., Щербин Д., Хаханов В., Чумаченко С. Облачное управление дорожным движением // П’ята Міжнародна науково-практична конференція «Сучасні інформаційні та інноваційні технології на транспорті (MINTT-2013)». Херсон. 28 – 30 травня 2013.С. 41-43.

22. Дементьев С.П., Хаханов В.И. Интеллектуальная система «Инфраструктура – транспорт – облако» // Матеріали XIX Всеукраїнської науково-практичної конференції «Інноваційний потенціал української науки – XXI сторіччя». Т.2. 2013. Запоріжжя. С.79-84.

23. Мищенко А.С., Чумаченко С.В. Оптимизация параллельных вычислительных структур // Матеріали XIX Всеукраїнської науково-практичної конференції «Інноваційний потенціал української науки – XXI сторіччя». Т.2. 2013. Запоріжжя. С.84-88.

24. Мищенко А.С., Дахири Фарид. Защита информации и компонентов киберпространства в облаках // Материалы третьей международной научной конференции студентов и молодых ученых «Сучасні інформаційні технології 2013».25-26 квітня 2013. Одеса. С. 31-32.

25. Дементьев С.П., Хаханов В.И., Чумаченко С.В. Модель процессов тестирования уязвимостей и проникновений // Тези доповідей Четвертої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації». 23-25 квітня 2013. Вінниця. С. 97-100.

26. Мищенко А.С., Хаханов В.И., Чумаченко С.В. Векторно-логический метод диагностирования уязвимостей // Тези доповідей Четвертої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації». 23-25 квітня 2013. Вінниця. С. 101-104.

27. Dahiri F., Dementiev S. Hahanov V.I., Chumachenko S.V., Litvinova E.I. Intellection Traffic Control on Cloud // HPC – UA Conference «Parallel and Distributed Computing Systems». Kharkiv, March 13-14, 2013. P. 130-142.

28. Гузь О.А. Бондаренко М.Ф., Хаханов В.И., Энглези И.П., Лобур М.В., Чумаченко С.В., Литвинова Е.И. Облако мониторинга и управления дорожным движением – зеленая волна // Материалы международного научного симпозиума «Наука в жизни современного человека».Одесса. Февраль 2013. С. 80-100.

29. Хаханов В.И., Чумаченко С.В., Литвинова Е.И., Хаханова А.В. Метрика качества дорожной инфрастуктуры // Материалы Международной научной конференции “Интеллектуальные системы принятия решений и проблемы вычислительного интеллекта (ISDMCI’2013)”. 20–24 мая 2013. Евпатория. С. 313-315.

30. *Мищенко А.С. Чумаченко С.В. Критерий качества оценивания бинарных отношений // Материалы конференции "Информатика, математика, автоматика". 22-27 апреля 2013. Сумы. С. 68.

31. Белоус В.В., Скоробогатий М.В. Электронный идентификатор транспорта для облака мониторинга и управления // Матеріали XIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса. 29 квітня 2013. С. 135-136.

32. Василенко В., Белоус В. (научный руководитель – д.т.н., проф. Чумаченко С.В.) Реализация алгоритма Дейкстры для структуры соединений процессорных примитивов типа G2 // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.7-8.

33. Врублевский Н.Н., Мусиенко Б.Г. (научный руководитель – д.т.н., проф. Литвинова Е.И.) Кубит-процессор оптимального покрытия // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.9-10.

34. Галаган С., Скоробогатый М. (научный руководитель – д.т.н., проф. Чумаченко С.В.) Реализация алгоритма Дейкстры для структуры соединений процессорных примитивов типа G3 // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.11-12.

35. Крохмаль И.А.. Сенченко М.И. (научный руководитель – д.т.н., проф. Хаханов В.И.). Кубитные, квантовые модели данных и вычислительных процессов // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.15-16.

36. Мизь В.А. (научный руководитель – ас. Обризан В.И.). Анализ методов распознавания движения в реальном времени // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 22-24 апреля 2013. Ч. 5. С.17-18.

37. Мищенко А.С., Чугуров И.И. Научный руководитель – д.т.н., проф. Чумаченко С.В. Модификация алгоритма Дейкстры для определения средней стоимости межсоединений вычислительной архитектуры // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.19-20.

38. Батуров М., Гниденко В. (научный руководитель – д.т.н., проф. Хаханов В.И.) Инфраструктура тестирования вредоносных компонентов // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч.5. 22-24 апреля 2013. С.54-55.

39. Горобец А.А., Дахири Фарид. (научный руководитель – д.т.н., проф. Хаханов В.И.) Автоматная модель взаимодействия облака и транспортных

средств // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.60-61.

40. Дементьев С., Ломова Ю. (научный руководитель – д.т.н., проф. Хаханов В.И.) Диагностирование функциональных нарушений на основе использования матрицы активизации // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.62-63.

41. Емельянов И., Котляров А. (научный руководитель – д.т.н., проф. Хаханов В.И.) Телеметрический модуль «Sherlock» для управления мобильными объектами // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч.5. 22-24 апреля 2013. С.64-65.

42. Зацарный А., Проценко Е. (научный руководитель – д.т.н., проф. Хаханов В.И.) Типы взаимодействия объектов индивидуального киберпространства пользователя и интегральная оценка // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.66-67.

43. Мизь В., Бояджан А. (научный руководитель – д.т.н., проф. Хаханов В.И.) Интеллектуальная система «Инфраструктура-транспорт-облако» // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.72-73.

44. Приймак А.С., Межевич А. (научный руководитель – д.т.н., проф. Хаханов В.И.) Организация связей «облако-автомобиль» и «облако-инфраструктура» // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.80-81.

45. Пузанов М.В., Полетайкин А.Н. (научный руководитель – д.т.н., проф. Хаханов В.И.) . Автоматизированная система контроля городского дорожного движения «клубок» // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.82-83.

46. Филиппенко И.О., Короленко А. (научный руководитель – д.т.н., проф. Хаханов В.И.). Структура автомобильного блока CAR-ID // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.90-91.

47. Щербин Д., Копейка Н. Средства мониторинга и управления дорожным движением // Материалы XVII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 22-24 апреля 2013. С.98-99.

48. Хаханов В.И., Энглези И.П., Гузь О.А., Полетайкин А.Н. Современные инфраструктурные средства управления дорожным движением в крупных городах // Матеріали III Міжнародної науково-практичної конференції «Проблеми підвищення рівня безпеки, комфорту та культури дорожнього руху». ХНАДУ. 16-17 квітня 2013. С. 208-210.

49. Белоус В.В. Научный руководитель – к.т.н., доцент Зайченко С.А. Методы уменьшения энергопотребления FPGA-устройств // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 14-16 апреля 2014. Ч. 5. С.5-6.

50. Варченко В.Г. Научный руководитель – к.т.н., доцент Зайченко С.А. Проверка ограничений проектов цифровых систем на основе топологического анализа RTL-модели // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 2014. часть 5. 14-16 апреля. С.7-8.

51. Березин Н. П. Научный руководитель – к.т.н., доцент Зайченко С.А. Топологическая верификация правил проектирования для многослойных FPGA-устройств Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.9-10.

52. Зенович И.А. Научный руководитель – к.т.н., доцент Зайченко С.А. Система управления многофазной статической верификацией проектов систем на кристалле // Материалы XVIII Международного молодежного

форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.11-12.

53. Ткачук А.С. Научный руководитель – доц. Кулак Э.Н. Контроллер LCD монитора для отладочного макета SPARTAN 3E // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке Ч. 5. 14-16 апреля 2014. С.13-14.

54. Титаренко В.Н., Кричко П.В. (Научный руководитель – д.т.н., проф. Хаханова И.В. Перспективы развития встраиваемых систем // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.15-16.

55. Авак Кингсли Эдем. Научный руководитель – д.т.н., проф. Кривуля Г.Ф. Использование стандарта PCI Express" для схемной реализации "умного" дома // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.17-18.

56. Черкасов С.И. Научный руководитель – д.т.н., проф. Кривуля Г.Ф. Оценка готовности компьютерной системы // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.19-20.

57. Демидчук М.А., Раков К.В. Научный руководитель – к.т.н., проф. Немченко В.П. Тестирование конформности сетевых протоколов // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014.. С.21-22.

58. Угримова О.К. Научный руководитель – доц. Шкиль А.С. Анализ продукционных правил в системах логического вывода // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке Ч. 5. 14-16 апреля 2014. С.23-24.

59. Серокурова А. С. Научный руководитель – доц. Шкиль А.С. Структурно-функциональные методы поиска ошибок проектирования в языковых моделях цифровых устройств // Материалы XVIII Международного моло-

дежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.25-26.

60. Дахири Ф.Н. Научный руководитель – проф. Хаханов В.И. Виртуальный квантовый компьютер для решения задач кубического покрытия Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.42-43.

61. Зиарманд А.Н., Чугуров И.И. Научный руководитель – д-р техн. наук, проф. В.И. Хаханов. Формальная модель киберсистемы // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С. 44-45.

62. Бояджян А.Г., Ломова Ю.В. Научный руководитель – д-р техн. наук, проф. В.И. Хаханов. Структура ICTC-системы // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.46-47.

63. Мизь В.А., Соколов А.С. Научный руководитель – д-р техн. наук, проф. В.И. Хаханов. Инновационные облачные сервисы для управления дорожным движением // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 2014. Ч. 5. 14-16 апреля. С.48-49.

64. Бояджян А.Г., Ломова Ю.В. Научный руководитель – доц. каф. АПВТ Хаханова А.В. Облачные сервисы GOOGLE // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.50-51.

65. Степанова Ю.В. Научный руководитель – к.т.н., проф. Немченко В.П. Оптимизация энергопотребления и ускорение передачи сообщений на основе протокола zigbee // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.64-65.

66. Скоробогатый М.В. Научный руководитель – д.т.н., проф. Чумаченко С.В. Анализ квантовых алгоритмов для решения высокопараллельных за-

дач // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.54-55.

67. Мельник Н.А. Научный руководитель – д.т.н., проф. Чумаченко С.В. Инфраструктура облачных сервисов для мобильных платформ // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.56-57.

68. Батуров М.А. Научный руководитель – д.т.н., проф. Литвинова Е.И. Дедуктивный метод поиска неисправностей // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.58-59.

69. Бутенко С.О., Косилов О.В. Научный руководитель – д.т.н., проф. Литвинова Е.И. Математический аппарат метода повышения тестопригодности критических систем управления // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». Ч. 5. 14-16 апреля 2014. С.60-61.

70. Шеремет Е. В. Научный руководитель – д.т.н., проф. Кривуля Г.Ф. Інструментальні засоби штучного інтелекту для діагностики комп'ютерних систем // Материалы XVIII Международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». 2014. часть 5. 14-16 апреля. С.66-67.

71. *Sherbin D., Gharibi W., Hahanov V., Baghdad Ammar Avni Abbas, Litvinova E.I., Modeling of digital systems // Матеріали XIII Міжнародної конференції TCSET 2014. 25 лютого -1 березня. 2014. Львів – Славське. С.77.

72. *Dementiev S., Hahanov V., Chumachenko S., Hahanova A. Quantum method for diagnosing digital systems // Матеріали XIII Міжнародної конференції TCSET 2014.25 лютого -1 березня. 2014. Львів – Славське. С.78

73. *Miz V., Ziarmand A., Hahanova A., Priymak A., Hahanov V. Cloud service for traffic control // Матеріали XIII Міжнародної конференції TCSET 2014.25 лютого -1 березня. 2014. Львів – Славське. С.557.

74. *Vladimir Miz, Aleksey Zhalilo, Artur Ziarmand, Vladimir Hahanov, Wajeb Gharibi, Abramova L.S., Svetlana Chumachenko, Eugenia Litvinova, Anna Hahanova, Vladimir Rustinov, Cyber Physical System – Smart Cloud Traffic Control // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.49-66.

75. *Ivan Hahanov, Daria Krulevska, Anastasya Yerchenko, Alexander Mishchenko, Dmitry Shcherbin, Aleksey Priymak Vladimir Hahanov, Wajeb Gharibi, Kudin A.P., Ngene Cristopher (Nigeria), Tiekura Yeve (Côte d'Ivoire) Cyber Physical Social Systems – Future of Ukraine // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.67-81.

76. *Farid Dahiri, Anastasiya Hahanova, Baghdadi Ammar Awni Abbas (Baghdad University). Qubit Method for Diagnosing Digital Systems // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.93-96.

77. *Sergey Zaychenko, Valeria Varchenko Vladimir Hahanov. Method for Diagnosing SoC HDL-code // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.97-102.

78. *Volodymyr Miz, Vladimir Hahanov. Smart traffic light in terms of the Cognitive road traffic management system (CTMS) based on the Internet of Things // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.103-107.

79. *Emelyanov Igor, Hahanova Irina, Tamer Bani Amer. Qubit Modeling Digital Systems // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.246-248.

80. *Yulia Hahanova, Armen Bayadzhan. Repair of Combinational Units // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.249-251.

81.*Alexander Adamov, Vladimir Hahanov, Anders Carlsson. Discovering New Indicators for Botnet Traffic Detection // Proc. of IEEE East-West Design and Test Symposium. Kiev. 26-29 September 2014. P.281-285.

82. Хаханов В.И. Киберфизическая система, облачное управление транспортом // Конференция AI Ukraine-2014. Харьков. 25 октября 2014.

83. *Murad Ali Abbas Hahanov V., Litvinova E., Chumachenko S. Synthesis of qubit models for logic function // Proc. of XxvII Conference on Design of Circuits and integrated systems. P. 20-21. Avignon, France, Nov. 28-30th 2012. (Hahanov V., Chumachenko S.)

84. *Hahanova A.V. Barannik V.V., Krivonos V.N. Coding Tangible Component of Transforms to Provide Accessibility and Integrity of Video Data // Proc. of IEEE East-West Design and Test Symposium. IEEE. USA. Kharkov. 14-17 September 2012. P.475 - 478.

85. *Hahanova A. Barannik V., Krasnoruckiy A. The Positional Structural-Weight Coding of the Binary View of Transformants // Proc. of IEEE East-West Design and Test Symposium. IEEE. USA. Kharkov. 14-17 September 2012. P.525 - 528.

86. *Hahanov, W. Gharibi, K.L. Man, E. Litvinova, S. Chumachenko, O. Guz VIntelligent Road Control and Monitoring // Proc. of the 2013 International Conference on Future Information and Communication Engineering - ICFICE2013, Shenyang, China, June, 2013.

87. *Хаханов В.И., Чумаченко С.В., Филиппенко И.В., Хаханова А.В. Infrastructure of PenTestsng and vulerability management // Материалы IX Международной конференции «Стратегия качества в промышленности и образовании». Технический университет-Варна. Болгария. 31 мая – 07 июня 2013. С.456-458.

88. *Guz O., Hahanov V., Gharibi W., Baghdadi Ammar Awni Abbas, Chumachenko S., Litvinova E. Cloud traffic monitoring and control // Proc. of the 2013 IEEE 7th International conference on intelligent data acquisition and advanced computing systems (IDAACS). Berlin, Germany. September 12-14. 2013. P. 244-248.

89. *Wajeb Gharibi, Hahanov V.I., Anders Carlsson, Hahanova I.V., Filippenko I.V. Quantum Technology for Analysis and Testing Computing Sys-

tems // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P.52-56.

90. *Ziarmand A.N., Arefjev A., Hahanov V.I., Guz O.A., Ngene Christopher Umerah. Cloud Traffic Control System // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P.72-76.

91. *Miz V.A., Shcherbin D., Litvinova E.I., Englesy I.P. Cloud Infrastructure for Car Service // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P.77-84.

92. *Volodymyr Miz, Volodymyr Hahanov. Quantum Computing Approach for Shortest Route Finding // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P.85-87.

93. *Dementiev S., Baghdadi Ammar Awni Abbas, Hahanov V.I., Palanichamy Manikandan, Litvinova E.I. Quantum Modeling and Repairing Digital Systems // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P.88-93.

94. *Priymak A., Maksimov M., Hahanov V.I., Hahanova I.V., Litvinova E.I., Elena Fomina, Tiecoura Yves, Malek Jehad Mohammad Jararweh. Quantum Models for Description of Digital Systems // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P.94-101.

95. *Artur Ziarmand. Smart Road Infrastructure // Proc. of IEEE East-West Design and Test Symposium. Rostov-on-Don. 27-30 September 2013. P. 430-434.

96. *Gorobets A., Chugurov I., Scherbin D., Chumachenko S. Dijkstra Algorithm for cyber structures analysis // Матеріали XII Міжнародної науково-технічної конференції CADSM 2013 «Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці». 19-23 лютого 2013. Львів – Поляна. С.61-65.

97. *Shakhov D., Hahanov V., Gharibi W., Lobur M., Litvinova E., Chumachenko S., Saatchyan A., Guz O., Filippenko O., Poletaykin A. Cloud «Green Wave traffic monitoring and control» // Матеріали XII Міжнародної науково-технічної конференції CADSM 2013 «Досвід розробки та застосування

приладо-технологічних САПР в мікроелектроніці». 19-23 лютого 2013. Львів – Поляна. С.120-126.

98. *Jerbarov D., Abeid A.M., Hahanov V., Hayford A., Ahmetoglu A.H., Oghumu S. Pentesting and vulnerability diagnosis // Матеріали XII Міжнародної науково-технічної конференції CADSM 2013 «Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці». 19-23 лютого 2013. Львів – Поляна. С.127-131.

99. *Hahanov V.I., Hyduke S., Gharibi W., Litvinova E.I., Chumachenko S., Hahanova I.V. Quantum Models and Method for Analysis and Testing Computing Systems // Proc. of the 11th International Conference on Information Technology – ITNG 2014, 7-9 April. 2014. Las Vegas, Nevada. P.66.

100. *Hahanov V.I., Litvinova E.I., Gharibi W., Chumachenko S. Cyber Physical System – iCloud Traffic Control // Proc. of the 11th International Conference on Computer Modelling and Simulation – UKSim - AMSS 2014, 26-28 March. 2014. Cambridge, United Kingdom. P.158-161.

101. *Hahanov V.I., Litvinova E.I., Gharibi W., Chumachenko S. iCloud Traffic Control and Monitoring // Proc. of the 16th International Conference on Information Technology – ITNG 2014, 7-9 April. 2014. Las Vegas, Nevada. P.65.

102. Hahanov V.I. Пленарный доклад на конференции “Digital System Technology: Education, Research, and Industrial Aspects”. Тегеран. Иран. 6-10 февраля 2014.

103. *Hahanov V., Ka Lok Man, Baghdad Ammar Avni Abbas, Litvinova E.I., Chumachenko S., Jihyeok Ahn, Kyung Ki Kim. Tab-model for multilevel diagnosis and repair of HDL SoC // Proc. ISOCC. 2014. P. 181-182.

104. *Hahanov V., Litvinova E.I., Chumachenko S., Wajeb Gharibi. Qubit data structures for analyzing computing systems // Proc. Third International conference of data mining and knowledge management. November 7-8. 2014. Dubai. UAE. P. 72-81.

105. *Хаханов В., Гариби В., Чумаченко С., Литвинова Є. Кіберфізична система хмарного управління транспортом // II Международный научный

конгресс “Фундаментальные и прикладные исследования в Америке, Европе и Азии”. 2014. 12 с.

106. *Хаханов В., Тамер Бані Амер, Чумаченко С., Литвинова Є. Розумне управління ресурсами // II Международный научный конгресс “Фундаментальные и прикладные исследования в Америке, Европе и Азии”. 2014. 21 с.

107. *Хаханов В.И., Bahdadi Ammar Awni Abbas, Чумаченко С.В. Кубитные стуктуры данных вычислительных устройств // Материалы 10й российской конференции с международным участием «Новые информационные технологи в исследовании сложных структур». Томск. 2014. С. 57-58.

108. *Хаханов В.И., Матросова А.Ю., Bahdadi Ammar Awni Abbas, Литвинова Е.И. Кубитные технологии анализа и диагностирования цифровых устройств // Материалы 10й российской конференции с международным участием «Новые информационные технологи в исследовании сложных структур». Томск. 2014. С. 58-59.

10.6 Договори з зацікавленими організаціями

Укладено угоду про співробітництво з Технологічним університетом Блекінге (м. Карлскрона, Швеція, 2012 р.).

Укладено угоду про співробітництво з Норвезьким університетом науки та технологій (м. Тронхейм, Норвегія, 2013 р.)

10.7 Організація і проведення конференцій та семінарів

З 14 по 17 вересня 2012 р. у Харкові за ініціативою Харківського національного університету радіоелектроніки, за підтримки Академії наук прикладної радіоелектроніки, Харківського містського голови та Харківської областної держадміністрації, а також за фінансової підтримки Талліннського технологічного університету (Естонія), IT-компаній Aldec, Synopsys,

Лабораторія Касперського, DataArt Lab пройшов ювілейний 10-й Міжнародний науковий симпозіум «IEEE East-West Design & Test Symposium – 2012». Технічним спонсором EWDTS'2012 є комп'ютерне співтовариство IEEE Computer Society Test Technology Technical Council (TTTC). Рейтинг симпозіуму – № 6 серед прем'єр-конференцій планети з комп'ютерної інженерії. Мета симпозіуму – розширення міжнародного співробітництва та обмін досвідом між провідними вченими Західної та Східної Європи, Північної Америки та інших країн в області автоматизації проектування, тестування та верифікації електронних компонентів і систем. До участі в симпозіумі було прийнято 137 доповідей від 296 авторів із 30 країн світу (Алжир, Вірменія, Білорусь, Бразилія, Кот Дівуар, Китай, Естонія, Ефіопія, Франція, Німеччина, Індія, Іран, Італія, Лівія, Мексика, Нігерія, Палестина , Польща, Португалія, Румунія, Росія, Саудівська Аравія, Сінгапур, Південна Африка, Сирія, Швеція, Туреччина, Великобританія, Україна, США). Кількість учасників – 103. Кількість пленарних та замовних доповідей – 12. Кількість університетів і компаній – 102. Кількість міст – 79.

З 27 по 30 вересня 2013 р. в Ростові-на-Дону за ініціативою Харківського національного університету радіоелектроніки, спільно з Донським державним технічним університетом (ДДТУ), за підтримки Академії наук прикладної радіоелектроніки, а також за фінансової підтримки Талліннського технологічного університету (Естонія), IT-компаній Aldec, Synopsys, Лабораторія Касперського, DataArt Lab, Agilent Technologies пройшов 11-й Міжнародний науковий симпозіум «IEEE East - West Design & Test Symposium - 2013». Технічним спонсором EWDTS'2013 є комп'ютерне співтовариство IEEE Computer Society Test Technology Technical Council (TTTC). Рейтинг симпозіуму – № 6 серед прем'єр-конференцій планети з комп'ютерної інженерії. Мета симпозіуму – розширення міжнародного співробітництва та обмін досвідом між провідними вченими Західної та Східної Європи, Північної Америки та інших країн в області автоматизації проектування, тестування та верифікації електронних компонентів і систем. Оргкомітет симпозіуму провів вели-

ку підготовчу роботу, сповістивши понад 400 учасників та організацій України, країн СНД і далекого зарубіжжя, а також з розробки та підтримки сайту симпозіуму http://ewdtest.com/, доступ до якого через мережу інтернет щорічно збільшує кількість потенційних учасників. До участі у симпозіумі було прийнято 110 доповідей від 305 авторів з 23 країн світу (Алжир, Вірменія, Білорусь, Бразилія, Чехія, Кот Дівуар, Китай, Естонія, Індія, Іран, Ірак, Йорданія, Мексика, Нігерія, Норвегія, Португалія, Росія, Саудівська Аравія, Швеція, Туреччина, Україна, США, В'єтнам). Кількість учасників, які прибули на симпозіум – 84. Кількість пленарних та замовних доповідей – 7. Кількість університетів і компаній – 73. Кількість міст - 48. Представлено 81 доповідь від 94 авторів з 10 країн (Вірменія, Білорусія, Бразилія, Чехія, Індія, Іран, Португалія, Росія, Україна, США). Були видані програма і матеріали симпозіуму (Proceedings) обсягом 503 сторінки формату А4. EWDTS-2013 привернув увагу преси та телебачення – радіо «Ростов», офіційне видання ДДТУ «Донський технічний», телестудія ДДТУ, програма «Дзеркало науки» Харківського Обласного телебачення, які висвітлювали роботу симпозіуму.

З 26 по 29 вересня 2014 р. в Києві за ініціативою Харківського національного університету радіоелектроніки на базі Національного педагогічного університету ім. М.П. Драгоманова та при підтримці Академії наук прикладної радіоелектроніки, Талліннського технологічного університету (Естонія) пройшов 12-й Міжнародний науковий симпозіум «IEEE East-West Design & Test Symposium - 2014». Кількість університетів і компаній – 69, серед них: ХНУРЕ (Харків), Донськой державний технічний університет (Ростов-на-Дону, Росія), МІЕТ (Москва), НТУ ХПІ (Харків), НТУУ КПІ (Київ), НАУ «ХАІ» (Харків), Південно-Федеральний університет (Ростов-на-Дону, Росія), Єреванський інженерний університет (Вірменія, Єреван), Стеленбоський університет (ПАР), Чеський технічний університет у Празі (Чеська Республіка), Науково-виробнича компанія «Радій» (Кіровогад, Україна), Одеський національний політехнічний університет (Одеса, Україна), Вятський державний університет (Вятка, Росія), Московський державний університет ім. М.В. Ло-

моносова (Москва, Росія), Томський державний університет (Томськ, Росія), ІТ-компанії Aldec, Synopsys, Російський дослідницький центр Хуавей (Москва, Росія), Концерн «Сузірря» (Воронеж, Росія). Видано збірник статей матеріалів, тез (додається) – були видані матеріали симпозіуму (Proceedings) обсягом 361 сторінка формату А4.

При кафедрі Автоматизації проектування обчислювальної техніки ХНУРЕ діє щотижневий науковий семінар «Актуальні проблеми кіберпростору», який об’єднує дослідників, викладачів, студентів та аспірантів кафедри, університету, а також зацікавлених вчених університетів Харкова та України. У межах семінару обговорюються питання, пов’язані з розвитком квантових обчислень, движків кіберпростору, хмаровими обчисленнями, захистом інформації у кіберпросторі, персональним віртуальним кіберкомп’ютером. За період виконання НДР проведено засідання 93 семінарів.

10.8 Виконані інноваційні проекти та проекти, що виконуються

1) Виконується проект «Curricula Development for new speciality Master of Engineering in MEMS/NEMS Design" (MastMEMS)» TEMPUS сумісно з університетом «Львівська політехніка», Київським національним університетом, Технічним університетом м. Лодзь (Польща), Ліонським університетом (Франція), Університетом м. Ільмінау (Німеччина), Університетом м. Павія (Італія) на 2012 – 2015 рр. – керівники: д.т.н., проф. Хаханов В.І., д.т.н., проф. Чумаченко С.В. Обсяг фінансування ХНУРЕ – 153 тис. Євро.

2) Завершено проект «SEIDA» за рахунок Євросоюзу (BAITSE «Baltic Academic IT Security Exchange») сумісно зі Швецією (куратор проекту), Естонією, Латвією – автори: д.т.н., проф. Хаханов В.І., викладач, аспірант Адамов О.С., учасники – д.т.н., проф. Чумаченко С.В. Термін: 01.10.2011 – 31.05.2014. Сумісні наукові дослідження, проведення конференцій та участь у конференціях з доповідями, розробка монографії, розробка курсу лекції та

лабораторного практикуму з захисту інформації, обмін студентами та викладачами з метою стажування.

3) Розпочато виконання проекту ТЕМПУС ENGENSEC, Blekinge Institute of Technology (BTH), Karlskrona, Sweden, 2014-2016 рр. (виконавці від кафедри АПОТ – ас. Адамов А.С., ст.. викл. Обрізан В.І.), координатор від України – каф. ТКС ХНУРЕ.

10.9 Отримані гранти

Таблиця 10.2 – Гранти, отримані у межах НДР

Тип гранту	Кількість грантів
На стажування/підвищення кваліфікації: **2012 рік** **1) 1 грант** на магістерську підготовку в університеті м. Ааллен (Німеччина), студент 5 курсу Башмаков В., лютий-вересень 2013. **2013 рік** **1) 1 грант** на магістерську підготовку в університеті м. Ааллен (Німеччина), студент 5 курсу Башмаков В., лютий 2012 -вересень 2013. **2) 1 грант** на магістерське стажування в Ліонському університеті (Франція) – магістрант Мізь В., липень 2013 **3) 1 грант** на магістерське стажування в Лодзинському політехнічному університеті (Польща) – магістрант Дахірі Фарід, липень 2013. **2014 рік** **1) 1 грант** на магістерське стажування (переддипломна дослідницька практика) у палійському університеті (Італія) - магістрант Дахірі Фарід, 28.02-28.03.2014. **2) 2 гранти** на інтенсивні технологічні магістерські курси в Ліонському університеті (Франція) – магістранти Білоус В., Скоробогатий М.., травень 2014. **3) 4 гранти** на інтенсивні технологічні магістерські курси в Лодзинському політехнічному університеті (Польща) – магістранти Мусієнко Б., Марченко Б., Мороз К., Гурєєв Б. 30.10-23.11.2014 **4) 1 грант** на стажування в Лодзинському політехнічному університеті (Польща) – д.т.н., проф. Литвинова Є.І., листопад 2014. **5) 2 гранти** на участь у тренінгу «Courses modules developers training “Train the Trainer”» з отриманням сертифікатів, 20–24 жовтня 2014 Wiesbaden, Germany (ст. викладач каф. АПОТ Обрізан В.І., асс. каф. АПОТ Адамов О.С.) **6) 1 грант** на участь у тренінгу Название: Internet of Things and Smart Cities Ph.D. School 2014 (Школа для аспірантів за темою "Інтернет речей та розумні міста". Леричі, Італія. Організатор - Університет Парми. 8-13 вересня 2014 – асп. каф. АПОТ Мізь В.	**15**

Тип гранту	Кількість грантів
На організацію науково-навчальних заходів: **<u>2012 рік</u>** **1) 2 гранти** від Талліннського технологічного університету на 27.03.2012-31.03.2012 – д-р техн. наук, проф. Хаханов В.І. на опонування дисертації; д-р техн. наук, проф. Чумаченко С.В. для узгодження питань з підготовки сумісного дослідницького проекту між ХНУРЕ та Талліннським технологічним університетом. **2) 3 гранти** на організацію 10-го Міжнародного наукового симпозіуму «IEEE East-West Design & Test Symposium - 2012», що пройшов 14 - 17 вересня 2012 у Харкові за ініціативою Харківського національного університету радіоелектроніки та за підтримкою Академії наук прикладної радіоелектроники, а також при фінансовій підтримці Талліннського технологічного університету (Естония), IT-компаній Aldec, Synopsys, Лабораторія Касперського, DataArt Lab. **3) 2 гранти** на участь у нараді університетів-партнерів щодо виконання проекту ТЕМПУС 9-10.12.2012 – д-р техн. наук, проф. Хаханов В.І.; д-р техн. наук, проф. Чумаченко С.В. **<u>2013 рік</u>** **1) 3 гранти** на організацію 11-го Міжнародного наукового симпозіуму «IEEE East-West Design & Test Symposium - 2013», що пройшов 27 - 30 вересня 2013 у Ростові-на-Дону за ініціативою Харківського національного університету радіоелектроніки та за підтримкою Академії наук прикладної радіоелектроники, а також при фінансовій підтримці Талліннського технологічного університету (Естония), IT-компаній Aldec, Synopsys, Лабораторія Касперського, DataArt Lab, Agilent Technologies. **2) 2 гранти** на участь у нараді університетів-партнерів щодо виконання проекту ТЕМПУС 19-21.10.2013 – д-р техн. наук, проф. Хаханов В.І.; д-р техн. наук, проф. Чумаченко С.В. **3) 2 гранти** до Норвезького університету науки і технологій 12-15.06.2013 для опонування дистертації та узгодження питань наукво-технічного співробітництва – д-р техн. наук, проф. Хаханов В.І.; д-р техн. наук, проф. Чумаченко С.В. **4) 1 грант** до Талліннського технологічного університету для узгодження питань науково-технічного співробітництва 16-18.06.2013 – д-р техн. наук, проф. Чумаченко С.В. **<u>2014 рік</u>** **1) 1 грант** на організацію 12-го Міжнародного наукового симпозіуму «IEEE East-West Design & Test Symposium - 2014», що відбувся 26-29 вересня 2014 р. у Києві за ініціативою Харківського національного університету радіоелектроніки та за підтримки Академії наук прикладної радіоелектроники, а також при фінансовій підтримці IT-компанії. **2) 2 гранти** на участь у нараді університетів-партнерів щодо виконання проекту ТЕМПУС MastMst 12-19.11.2014 – д-р техн. наук, проф. Хаханов В.І.; д-р техн. наук, проф. Чумаченко С.В. **3) 1 грант** на участь у нараді університетів-партнерів щодо виконання проекту ТЕМПУС ENGENSEC Blekinge Institute of Technology (BTH), Karlskrona, Sweden, 24-26 березня 2014 – ас. каф. АПОТ Адамов О.С. **4) 1 грант** на участь у нараді університетів-партнерів щодо виконання	**20**

Тип гранту	Кількість грантів
проекту ТЕМПУС ENGENSEC Saint Petersburg State University of Telecommunications (SUT) (Saint Petersburg, Russia) 30 June – 2 July 2014 – асс. каф. АПОТ Адамов О.С.	
На участь у конференції/олімпіаді: **2012 рік** **1) 2 гранти** на участь у конференції «Design and Circuits and Integrated Systems – 2012», Франція, г. Авіньон (д.т.н., проф. Хаханов В.І., д.т.н., проф. Чумаченко С.В.). **2) 3 гранти** на участь у нараді робочої групи проекту BAITSE «Baltic Academic IT Security Exchange» 27.05.2012-03.06.2012, Карлскрона, Швеція (д.т.н., проф. Хаханов В.І., д.т.н., проф. Чумаченко С.В., ас. Каф. АПОТ Адамов О.С.) **3) 3 гранти** на участь у нараді робочої групи проекту BAITSE «Baltic Academic IT Security Exchange» 12-14.12.2012, Рига, Латвія (д.т.н., проф. Хаханов В.І., Чумаченко С.В., ас. каф. АПОТ Адамов О.С.). **2013 рік** **1) 1 грант** на участь в олімпіаді з мікроелектроніки, м. Єреван, Вірменія, жовтень 2012 (магістр гр. СКСм-13-1 Скоробогатий М.) **2) 1 грант** на участь у нараді робочої групи проекту BAITSE «Baltic Academic IT Security Exchange» 22-26.04.2013, Вроцлав, Польща (ас. каф. АПОТ Адамов О.С.) **3) 1 грант** на участь у нараді робочої групи проекту BAITSE «Baltic Academic IT Security Exchange» 08-12.10.2013, Судак, Україна (ас. каф. АПОТ Адамов О.С.) **2014 рік** **1) 3 гранти** на участь у роботі семінару “Design&Test” (Тегеран, Іран) – д-р техн. наук, проф. Хаханов В.І.; д-р техн. наук, проф. Чумаченко С.В., д.т.н., проф. Литвинова Є.І. 03.02.2014 – 11.02.2014. **2) 4 гранти** на участь у Міждународній молодіжній конференції та конкурсі наукових робіт «Cyber Security for the Next Generation» (тур Росія+СНД), Россия, Москва, ЗАО «Лабораторія Касперського», д.т.н., проф. Хаханов В.І., д.т.н., проф. Чумаченко С.В., ас. каф. АПОТ Адамов О.С., асп. Зіарманд А.Н., 19.02.2014 – 22.02.2014. **3) 3 гранти** на участь у конференції “IEEE UKSim2014 16th International Conference on Modelling and Simulation”, Великобританія, м. Кембридж, Кембриджський університет – д-р техн. наук, проф. Хаханов В.І., Чумаченко С.В., Литвинова Є.І. 24.03.2014 – 31.03.2014. **4) 3 гранти на** участь у конференції “Information Technology – New Generation (ITNG)”, США, г. Лас Вегас, Університет Лас Вегаса, – д-р техн. наук, проф. Хаханов В.І.; д-р техн. наук, проф. Чумаченко С.В., д.т.н., проф. Литвинова Є.І. 05.04.2014 – 14.04.2014. **5) 1 грант** на участь в олімпіаді з мікроелектроніки, м. Єреван, Вірменія, жовтень 2014 (студент гр. КІ-14-4 Бояджян А.) **6) 3 гранти** на участь у конференції “Third International Conference of Data Mining & Knowledge Management Process, CDKP-2014”, ОАЕ, м. Дубай, приймаюча сторона: Vel Tech University, Chennai, India – д-р техн. наук, проф. Хаханов В.І., Чумаченко С.В., Литвинова Є.І. 3-10.11.2014.	**28**
РАЗОМ:	**63**

10.10 Теми захищених та поданих дисертацій

1) Аспірант Мурад Алі Аббас «Квантові моделі обчислювальних процесів для тестування цифрових систем на кристалах». – Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 – комп'ютерні системи и компоненти (науковий керівник – д.т.н., проф. Хаханов В.І.). Захист – 19.12.2012.

2) Аспірант Альмадхоун С.М.М. Методи пошуку помилок проектування в моделях цифрових пристроїв на мовах опису апаратури. – Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.12 – системи автоматизації проектувальних робіт. (наук. керівник – к.т.н., доц. Шкиль О.С.). Захист – 26.11.2013.

3) Аспірант Кучеренко Д.Ю. Моделі та методи підвищення якості експертного діагностування комп'ютерних систем з використанням продукційних правил. – Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти (наук. керівник – к.т.н., доц. Шкиль О.С.). Подано до вченої ради 19.11.2014.

4) Аспірант Багдаді Аммар Авні Аббас. Кубітні моделі і методи аналізу та діагностування цифрових пристроїв. – Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти (наук. керівник – д.т.н., проф. Хаханов В.І.). Подано до вченої ради 23.12.2014.

10.11 Виставки

Таблиця 10.3 – Участь у виставках

П.І.Б. розроблювачів експонату	Назва експонату	Назва виставки	Дата й місце проведення виставки	Нагороди
2012				
Короленко А.А., Филиппенко И.О.	Эхо процессор	16-й Международный молодежный форум	Харьков, 17-19.04.2012. ХНУРЭ	
Короленко А.А., Филиппенко И.О.	Система оптической связи	16-й Международный молодежный форум	Харьков, 17-19.04.2012. ХНУРЭ	
Короленко А.А., Филиппенко И.О.	Ламповый гитарный усилитель	16-й Международный молодежный форум	Харьков, 17-19.04.2012. ХНУРЭ	
Короленко А.А., Филиппенко И.О.	GPRS система слежения за подвижным объектом	16-й Международный молодежный форум	Харьков, 17-19.04.2012. ХНУРЭ	
2013				
Бояджян А.Г., Гольцев Д.А., Емельянов И.В.	Системы электронной коммерции в современных условиях развития бизнеса	17-й Международный молодежный форум	Харьков, 22-24.04.2013 ХНУРЭ	
Хаханов В.И., Чумаченко С.В.	Curricula Development for New Specialization: Master of Engineering in Microsystems Design 530785-TEMPUS-1-2012-1-PL-TEMPUS-JPCR	Национальная выставка-презентация «Инноватика в современном образовании» в Выставочном центре «Киев-ЭкспоПлаза»	Киев 22-24.10.2013. ЭкспоПлаза	Диплом "За активну участь і представлення освітніх інноваційних технологій"
Хаханов В.И., Чумаченко С.В.	Топологическая верификация правил проектирования для многокристальных FPGA-устройств	Национальная выставка-презентация «Инноватика в современном образовании» в Выставочном центре «Киев-ЭкспоПлаза»	Киев 22-24.10.2013. ЭкспоПлаза	Диплом «За активну участь і представлення освітніх інноваційних технологій»
Мизь В.А.	Cloud Green Traffic Control – интеллектуальное облако управление инфраструктурой дорожного движения	Международный форум «Kharkiv-IT». Харьков, Пятихатки	Харьков, 26.11.2013	Победитель конкурса, I место, диплом, золотая медаль
2014				
Зайченко С.А., Лештаев П.В., Степанов А.И., Парченко П.В., Сухомлинов Д.А., Гуреев Б.Н., Подлинный А.С., Лукашенко О.С., Александров В.И.	ALINT-PRO	Design Automation Conference (DAC'14)	США, Сан-Франциско, 1-5.06.2014	
Хаханов В.И., Зайченко А.С.,	Topological Verification Rules	Выставка научных разработок ХНУРЭ	ХНУРЭ, 21.12.2014	

П.І.Б. розроблювачів експонату	Назва експонату	Назва виставки	Дата й місце проведення виставки	Нагороди
Гуреев Б.Н. (Kharkov National University of Radioelectronics Aldec Inc)	For Multi-Chip FPGA-Devices			
Хаханов В.И., Зиарманд А.Н.	Киберфизическая система «Green Wave Traffic Control»	Выставка научных разработок ХНУРЭ	ХНУРЭ, 21.12.2014	
Хаханов В.И., Чумаченко С.В. Мусиенко Б.	TEMPUS MastMst - европейский образовательный проект (магистерские стажировки, моделирование встроенных микросистем).	Выставка научных разработок ХНУРЭ	ХНУРЭ, 21.12.2014	

10.12 Теми захищених магістерських робіт за тематикою НДР

1. Палюх С.В. Методи та спеціалізовані програмні засоби керування комп'ютерами в локальній мережі. Гр. СКСм-11-1. Керівник – проф. каф. АПОТ Кривуля Г.Ф.

2. Приймак О.С. Квантові моделі обчислювальних процесів. Гр. СКСм-11-1. Керівник – проф. каф. АПОТ Хаханов В.І.

3. Приходченко Р.С. Методи ефективного проміжсесійного кодування в бездротових мережах стандарту 802.11. Гр. СКСм-11-1. Керівник – проф. каф. АПОТ Немченко В.П.

4. Сухановська А.В. Методи управління комплексом "розумний дім" з збільшенною захищенністю даних. Гр. СКСм-11-1. Керівник – доц. каф. АПОТ Кулак Е.М.

5. Бражников А.М. Розробка спеціалізованої комп’ютерної моделі аналізу компетентності користувачів в комп'ютерних системах. Гр. СКСм-12-1. Керівник – доц. каф.АПОТ Шкіль О.С.

6. Кокулюк А.А. Програмно-апаратні інтерфейси для спеціалізованих комп'ютерних промислових систем. Гр. СКСм-12-1. Керівник – проф. каф. АПОТ Кривуля Г.Ф.

7. Куповець Н.С. Методи проектування тривимірних графічних об'єктів. Гр. СКСм-12-1. Керівник – проф. каф. АПОТ Хаханова І.В.

8. Недяк А.А. Моделі та методи аналізу конформності мережевих протоколів нової генерації. Гр. СКСм-12-1. Керівник – проф. каф.АПОТ Немченко В.П.

9. Філіппенко І.О. Спеціалізована комп'ютерна система мультимедійного стрілецького тренажору. Гр. СКСм-12-1. Керівник – проф. каф. АПОТ Хаханов В.І.

10. Шеремет Є.В. Спеціалізовані програмно-апаратні комп’ютерні засоби забезпечення енергобезпеки комплексу "Розумний будинок". Гр. СКСм-12-1. Керівник – проф. каф. АПОТ Кривуля Г.Ф.

11. Білоус В.В. Методи зменшення енергоспоживання для FPGA – пристроїв. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Чумаченко С.В.

12. Бутенко С.О. Моделі та методи обробки цифрових сигналів. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Литвинова Є.І.

13. Дахірі Ф.Н. Віртуальний квантовий комп'ютер для розв'язання задач покриття. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Хаханов В.І.

14. Демидчук М.О. Аналіз конформності мережевих протоколів на базі графічних моделей. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Немченко В.П.

15. Косилов О.В. Моделі та методи аналізу продуктивності та масштабованості сервісів обміну повідомленнями. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Литвинова Є.І.

16. Крічко П.В. Проектування вбудованих систем на основі контролера Picoblaze. Гр. СКСм-13-1. Керівник – проф. каф.АПОТ Хаханова І.В.

17. Мельник М.О. Інфраструктура хмарових сервісів для мобільних платформ. Гр. СКСм-13-1. Керівник – проф. каф.АПОТ Чумаченко С.В.

18. Раков К.В. Моделі та методи аналізу мережевих протоколів на базі мереж Петрі. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Немченко В.П.

19. Сірокурова А.С. Структурно-функціональні методи пошуку помилок проектування у мовних моделях цифрових пристроїв. Гр. СКСм-13-1. Керівник – доц. каф. АПОТ Шкіль О.С.

20. Скоробогатий М.В. Реалізація квантового алгоритму пошуку оптимальних шляхів на графі у віртуальному комп'ютері. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Чумаченко С.В.

21. Степанова Ю.В. Оптимізація роботи бездротової сенсорної мережі на базі технології ZigBee. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Немченко В.П.

22. Титаренко В.М. Мікроконтролерна реалізація вимірювальних приладів. Гр. СКСм-13-1. Керівник – проф. каф. АПОТ Хаханова І.В.

23. Угрімова О.К. Аналіз продукційних правил в системах логічного виводу. Гр. СКСм-13-1. Керівник – доц. каф. АПОТ Шкіль О.С.

24. Черкасов С.І. Інструментальні засоби штучного інтелекту для діагностики комп'ютерних систем. Гр. СКСм-13-1. Керівник – проф. каф.АПОТ Кривуля Г.Ф.

10.13 Залучення позабюджетних коштів

Госпдоговори виконані та, що виконуються (назва, замовник, загальна сума, доля, що виконана власними силами конкретні результати отримані по договору). Загальний обсяг позабюджетних робіт у 2012-2014 рр. – 30000 грн.

Таблиця 10.4 – Залучення позабюджетних коштів

№ п/п	Назва	Замовник	Загальна сума	Доля, що виконана власними силами	Конкретні результати, отримані по договору
1	Госпдоговірна НДР про надання консультаційних послуг	Сєвєродонецьке НВО «Імпульс»	Обсяг фінансування: 15000 грив.	100%	Надання консультаційних послуг

У межах підтримки фінансування НДР кафедрою автоматизації проектування обчислювальної техніки було залучено кошти Європроекту «Curricula Development for New Specialization: Master of Engineering in Microsystems Design 530785-TEMPUS-1-2012-1-PL-TEMPUS-JPCR» для оснащення лабораторії проектування мікросистем загальною сумою 11900.00 Євро:

№ п/п	Найменування обладнання	Інвентарний №	Кількість	Введено до експлуатації
1	Комп'ютер персональний Apple Macbook Pro MD101PL/A	8471300000	1	11.2014
2	Проектор Epson EH-TW5200	8528691000	2	11.2014
3	Сервер HP ProLiant ML 350e Gen8: Model 686778-425	8471410000	1	11.2014
4	Сервер Apple Mac mini with OS X	8471410000	1	11.2014
5	Станція графічна HP Z1 PN: WM427EA	8528594000	1	11.2014
6	Комп'ютер All0in-One Apple iMac 21,5'' MD093	8471410000	3	11.2014

10.14 Розробки

Таблиця 10.5 – Розробки за тематикою виконаної НДР, що впроваджені за межами ХНУРЕ

№ з/п	Назва та автори розробки	Важливі показники, які характеризують рівень отриманого наукового результату; переваги над аналогами, економічний, соціальний ефект	Місце впровадження (назва організації, відомча належність, адреса)	Дата акту впровадження	Практичні результати, які отримано від впровадження (обладнання, обсяг отриманих коштів, налагоджено співпрацю для подальшої роботи тощо)
1	Методи функціонального тестування критичних систем керування	Розроблено математичний апарат методу підвищення контролепридатності критичних систем управління, відмінною рисою якого є побудова логічних елементів	ЧАО Сєвєродонецьке НВО «Імпульс»	Акт про впровадження 12.03.2012	Програмно-апаратні засоби тестування цифрового обладнання захисту, побудованого з використанням функціональних елементів з арифметичними операціями, що

№ з/п	Назва та автори розробки	Важливі показники, які характеризують рівень отриманого наукового результату; переваги над аналогами, економічний, соціальний ефект	Місце впровадження (назва організації, відомча належність, адреса)	Дата акту впровадження	Практичні результати, які отримано від впровадження (обладнання, обсяг отриманих коштів, налагоджено співпрацю для подальшої роботи тощо)
		алгоритмів захисту без використання традиційної бінарної арифметики за рахунок застосування багатомірного (багаторозрядного) зображення вхідних, вихідних даних та процесів їх обробки			використовуються при проведенні лабораторних робіт, пов'язаних з синтезом цифрових систем на кристалах, що імплементується у програмовану логіку, у процесі вивчення курсів «Проектування цифрових систем на ПЛІС», «Основи автоматизації проектування».
2	Методи пошуку помилок проектування в моделях цифрових пристроїв	Розроблена методика проведення діагности-чного експерименту при верифікації HDL-коду, що дозволяє автоматично застосовувати алгоритми пошуку дефектів в залежності від форми специфікації і стилю написання HDL-моделі. Запропонований метод зворотного відстеження для графової моделі HDL-коду дозволив значно підвищити глибину пошуку помилок проектування (до операторів HDL-коду) в умо-	ООО «Альдек-КТС»	Довідка про впровадження № 32/13 від 15.05.2013	Використання зазначених методів у системах верифікації HDL-моделей при автоматизованому проектуванні цифрових пристроїв, які подані на мовах опису апаратури, а саме: у навчальному процесі кафедри АПОТ при викладанні навчальних дисциплін: 1) «Логічне моделювання» для бакалаврів напрямку «Комп'ютерна інженерія» у лекційному матеріалі по темі

№ з/п	Назва та автори розробки	Важливі показники, які характеризують рівень отриманого наукового результату; переваги над аналогами, економічний, соціальний ефект	Місце впровадження (назва організації, відомча належність, адреса)	Дата акту впровадження	Практичні результати, які отримано від впровадження (обладнання, обсяг отриманих коштів, налагоджено співпрацю для подальшої роботи тощо)
		вах неповної специфікації на проектований пристрій і значно скоротити загальний час верифікації HDL-моделі. Застосування розробленої методики дозволило на 20-30% скоротити час на пошук помилок проектування HDL-моделей і підвищити глибину пошуку до помилкового HDL-оператора.			«Верифікація моделей цифрових пристроїв» та у лабораторних роботах по темі «Побудова мовних моделей цифрових пристроїв». 2) «Основи комп'ютерної діагностики» для бакалаврів напрямку «Комп'ютерна інженерія» у лекційному матеріалі та практичних заняттях по темі «Методи пошуку дефектів у цифрових пристроях». 3) «Теорія проектування СКС» для магістрів спеціальності «Спеціалізовані комп'ютерні системи» у лекційному матеріалі по темі «Системи автоматизованого проектування цифрових пристроїв».
3	Моделі та методи підвищення якості експертного діагностування комп'ютерних	Отримання достовірних результатів діагностування за рахунок перевірки бази знань експертної системи	ПАТ «УРБП Шлях»	Довідка про впровадження від 11.09.2014	

№ з/п	Назва та автори розробки	Важливі показники, які характеризують рівень отриманого наукового результату; переваги над аналогами, економічний, соціальний ефект	Місце впровадження (назва організації, відомча належність, адреса)	Дата акту впровадження	Практичні результати, які отримано від впровадження (обладнання, обсяг отриманих коштів, налагоджено співпрацю для подальшої роботи тощо)
	систем	діагностування, що подана у вигляді продукційних правил, на логічну коректність. Достовірність результатів експертного діагностування при визначенні технічного стану корпоративної комп'ютерної мережі.			
4	Кубітні моделі та методи аналізу і діагностування цифрових пристроїв	За рахунок апаратної та структурної надлишковості на 50% збільшено швидкодію інтерпретативного моделювання, на 5% підвищено вихід придатної продукції, на 12% збільшено глибину діагностування несправних функціональних блоків та на 15% зменшено час відладки HDL-коду у процесі проектування цифрових систем на кристалах.	Алдек-КТС	Довідка про впровадження від 02.12.2014	

1) «Інфраструктура вбудованого ремонту апаратних компонентів цифрових систем на кристалах». Автори: д.т.н., проф. каф. АПОТ, декан факультету КІУ Хаханов В.І., д.т.н., проф. каф. АПОТ Чумаченко С.В., д.т.н., проф.

каф. АПОТ Литвинова Є.І., асп. каф. АПОТ Мурад Алі Аббас, асп. каф. АПОТ Горобець О.О.

Основні характеристики, суть розробки: *Наукова новізна результатів* – нова апаратно-орієнтована модель паралельного обчислення булеана, яка характеризується використанням процесорної Хассе-структури; нова автоматна модель комбінаційного пристрою, що характеризується можливістю автономного та вбудованого відновлення працездатності компонентів логічних пристроїв за рахунок переадресації дефектних примітивів; удосконалена кубітна модель даних, яка відрізняється нечисельним поданням розрядів у двійковому векторі; удосконалений суперпозиційний метод синтезу кубів функціональностей, що відрізняється формою отримання компактного покриття; удосконалений метод оцінювання ефективності обчислювальних структур, який відрізняється застосуванням модифікованого алгоритму Дейкстра для пошуку найкоротших шляхів між вершинами графової моделі функціональних блоків. *Практична значимість* – кубітні моделі та квантові методи тестування і ремонту доведено до програмно-апаратної реалізації в інфраструктурі вбудованого сервісного обслуговування функціональних компонентів систем на кристалах, що дозволило створити ефективні маршрути синтезу тестів та діагностування функціональних модулів SoC.

2) «Хмарове цифрове керування дорожнім рухом (Smart Cloud Traffic Control)». Автори: д-р техн. наук, проф. Хаханов В.І., д-р техн. наук, проф. Литвинова Є.І., д-р техн. наук, проф. Чумаченко С.В., асп. Зіарманд А.Н., асп. Мізь В.О., доц. каф. ТКС Філіпенко О.І., студент гр. КІ-12-4 Чугуров І.І.

Основні характеристики, суть розробки: Сутність розробки полягає у створенні інтелектуальної інфраструктури дорожнього руху (ІІДР) – хмарного сервісу моніторингу інфраструктури і керування дорожнім рухом у реальному масштабі часу «Зелена хвиля» на основі розробки віртуальної інфраструктури дорожнього руху, інтегрованої з вуличними дорожніми контролерами, засобами радіочастотної ідентифікації автомобілів з метою підвищення якості й безпеки пересування транспортних засобів, мінімізації часових і ма-

теріальних витрат при виконанні заданих маршрутів. *Наукова новизна* визначається системною інтеграцією хмари моніторингу і керування, блоків радіочастотної ідентифікації транспорту, а також засобів моніторингу і керування дорожньої інфраструктури, що дає можливість автоматизувати процеси оптимального керування транспортними засобами і дорожнім рухом у режимі реального часу для рішення соціальних, гуманітарних, економічних і екологічних проблем. *Інноваційна привабливість* розробки полягає у тому, що запропонована інтелектуальна система (інфраструктура, транспорт, хмара) моніторингу і керування дорожнім рухом відрізняється від існуючої структурною інтеграцією трьох взаємозалежних інтерактивних компонентів: 1) Існуючі сервіси електронної картографії із засобами радіолокації і радіонавігації, 2) Новий хмарний сервіс моніторингу і керування дорожнім рухом на основі дорожніх контролерів, 3) Удосконалені засоби радіочастотної ідентифікації автомобіля і доступу до хмарних сервісів для комфортного й безпечного пересування за маршрутами, оптимізації часових і матеріальних витрат.

3) «Методи пошуку помилок проектування в моделях цифрових пристроїв». Автори: д-р техн. наук, проф. Кривуля Г.Ф., канд. техн. наук, доц. Шкиль О.С., асп. Альмадхоун С.М.М., асп. Кучеренко Д.Ю.

Розроблена методика проведення діагностичного експерименту при верифікації HDL-коду, що дозволяє автоматично застосовувати алгоритми пошуку дефектів в залежності від форми специфікації і стилю написання HDL-моделі. Запропонований метод зворотного відстеження для графової моделі HDL-коду дозволив значно підвищити глибину пошуку помилок проектування (до операторів HDL-коду) в умовах неповної специфікації на проектований пристрій і значно скоротити загальний час верифікації HDL-моделі. Застосування розробленої методики дозволило на 20-30% скоротити час на пошук помилок проектування HDL-моделей і підвищити глибину пошуку до помилкового HDL-оператора. Використання зазначених методів у системах верифікації HDL-моделей при автоматизованому проектуванні цифрових пристроїв, які подані на мовах опису апаратури, а саме: у навчальному про-

цесі кафедри АПОТ при викладанні навчальних дисциплін: «Логічне моделювання» для бакалаврів напрямку «Комп’ютерна інженерія» у лекційному матеріалі по темі «Верифікація моделей цифрових пристроїв» та у лабораторних роботах по темі «Побудова мовних моделей цифрових пристроїв»; «Основи комп’ютерної діагностики» для бакалаврів напрямку «Комп’ютерна інженерія» у лекційному матеріалі та практичних заняттях по темі «Методи пошуку дефектів у цифрових пристроях»; «Теорія проектування СКС» для магістрів спеціальності «Спеціалізовані комп’ютерні системи» у лекційному матеріалі по темі «Системи автоматизованого проектування цифрових пристроїв».

4) «Застосування квантових обчислень для пошуку найкоротшого шляху (Q-route)». Автори: д-р техн. наук, проф. Хаханов В.І., асп. Мізь В.О.

Запропоновано новий підхід знаходження найкоротшого маршруту на графі доріг. Цей підхід заснований на теорії графів і використовує основні принципи квантових обчислень. Однією з численних переваг квантових обчислень є висока швидкодія при виконанні паралельних операцій. Запропонований підхід дозволяє знаходити декілька можливих шляхів одночасно і вибирати оптимальний маршрут на основі спеціальних метрик і параметрів. Інноваційна привабливість розробки полягає в тому, що запропонована інтелектуальна система пошуку найкоротшого маршруту відрізняється від існуючої застосуванням квантових обчислень.

10.15 Використання результатів НДР у навчальному процесі

Отримані результати НДР використовуються у навчальному процесі ХНУРЕ (довідки про впровадження від 10.09.2012, 20.05.2013, 10.09.2014) при викладанні курсів «Дискретна математика», «Спеціальні розділи дискретної математики», а саме: удосконалений метод оцінювання ефективності обчислювальних структур, який відрізняється від аналогів застосуванням графової моделі міжз'єднань функціональних блоків; удосконалений алгоритм Дейкстра; застосування метрики і критеріїв взаємодії об'єктів у кіберпросторі для підви-

щення швидкодії розв’язання задач дискретної оптимізації; нова апаратно-орієнтована модель паралельного обчислення булеана, яка характеризується використанням процесорної Хасе-структури; Хасе-структура обчислювального процесу; оцінка топології з'єднань компонентів цифрової системи за допомогою модифікованого алгоритму Дейкстра. Моделі та методи аналізу продукційних правил на логічну коректність і програма «Production rules analyzer» використовуються у навчальних дисциплінах: «Моделі та засоби діагностування ПЗ» у лекційному матеріалі на тему «Експертне діагностування ПЗ»; «Комп’ютерні системи на базі нечіткої логіки» у лекційному матеріалі – «Нечітка логіка у експертних системах діагностування» та у лабораторних роботах «Побудова системи підтримки прийняття рішення на базі нечіткої логіки»; «Теорія проектування СКС» у лекційному матеріалі – «Експертні системи діагностування» та у лабораторних роботах – «Побудова експертних систем». Програмно-апаратні засоби тестування, діагностування та відновлення працездатності функціональних блоків використовуються при проведенні лабораторних робіт з курсів: "Проектування цифрових систем на ПЛІС", "Квантові обчислення", в курсовому, дипломному проектуванні та при виконанні магістерських атестаційних робіт.

Підготовлено до друку підручник: Проектування та тестування цифрових систем на кристалах / В.І. Хаханов, Є.І. Литвинова, С.В. Чумаченко – Харків: ХНУРЕ.– 2014.– 484 с.

10.16 Дипломи, нагороди, довідки та акти впроваджень

СПРАВКА

о внедрении инфраструктуры встроенного ремонта аппаратных компонентов цифровых систем на кристаллах в научно-исследовательскую и производственную деятельность компании ЧАО «Северодонецкое «НПО «Импульс»

В результате экспериментальных исследований и проверки работоспособности инфраструктуры встроенного ремонта аппаратных компонентов цифровых систем на кристаллах, разработанного на кафедре автоматизированного проектирования вычислительной техники Харьковского национального университета радиоэлектроники при участии (30%) соискателя Мурада установлено, что разработанные средства имеют следующие характеристики:

1. Кубитная модель данных, которая отличается нечисленным представлением разрядов в двоичном векторе для существенного повышения быстродействия векторных логических операций при решении задач дискретной оптимизации.
2. Аппаратно-ориентированная модель параллельного вычисления булеана, которая характеризуется использованием процессорной Хассе-структуры для решения задач покрытия и минимизации функциональных структур.
3. Автоматная модель комбинационного устройства, которая характеризуется возможностью автономного и встроенного восстановления работоспособности компонентов логических устройств за счет переадресации дефектных примитивов.
4. Методы оценивания эффективности вычислительных структур, которые отличаются применением модифицированного алгоритма Дейкстра для поиска кратчайших путей между вершинами графовой модели функциональных блоков.
5. Аппаратный прототип квантового вычислителя на основе программируемой логики, который позволяет на порядок повысить быстродействие оптимального поиска покрытий в задачах дискретной оптимизации.
6. Верификация моделей, методов и архитектуры квантового вычислителя позволяет использовать их в качестве компонентов инфраструктуры сервисного обслуживания и встроенного ремонта цифровых систем для существенного (5%) повышения качества SoC.

Обоснованность подтверждается результатами экспериментальных исследований, обработкой нескольких реальных цифровых проектов. Результаты экспериментов подтверждают 5% повышение качества SoC по сравнению с существующими академическими аналогами.

Программно-аппаратные средства в виде аппаратных моделей и встроенной инфраструктуры ремонта являются валидными и используются в качестве методологического дополнения при проведении проектно-конструкторских работ, связанных с созданием цифровых систем на кристаллах, имплементируемых в программируемую логику.

Заведующий отделом, ученый секретарь ЧАО «Северодонецкое НПО «Импульс»,

кандидат технических наук _____ В.А. Ларгин

17.08.2012

Подпись Ларгина В.А.
удостоверяю
Зав. отд. [illegible] Н.И. Брилева
27/08-2012г.

ХНУРЕ
Вхідний № 01/27-1889
05 09 2012 р.

ALDEC
The Design Verification Company

ООО «Алдек-Харьков»
пр.Ленина 18/9, оф. 1
61166, г.Харьков, Украина

ЕГРПОУ 33817534
Р/с 26006033205400 в АКИБ «УкрСиббанк», МФО 351005
Тел.: (+38 057) 760-47-25

Исх. номер 17/12 / от 27.08.2012г.

СПРАВКА

о внедрении инфраструктуры встроенного ремонта аппаратных компонентов цифровых систем на кристаллах в научно-исследовательскую и производственную деятельность ООО «Алдек-Харьков»

В результате экспериментальных исследований и проверки работоспособности инфраструктуры встроенного ремонта аппаратных компонентов цифровых систем на кристаллах, разработанного на кафедре автоматизированного проектирования вычислительной техники Харьковского национального университета радиоэлектроники при участии (30%) соискателя Мурада установлено, что разработанные средства имеют следующие функциональности и преимущества:

1. Усовершенствованная кубитная модель данных, которая отличается нечисленным представлением разрядов в двоичном векторе для существенного повышения быстродействия векторных логических операций при дискретной оптимизации.
2. Новая аппаратно-ориентированная модель параллельного вычисления булеана, которая характеризуется использованием процессорной Хассе-структуры для решения задач покрытия и минимизации функциональных структур.
3. Новая автоматная модель комбинационного устройства, которая характеризуется возможностью автономного и встроенного восстановления работоспособности компонентов логических устройств за счет переадресации дефектных примитивов.
4. Усовершенствованные методы оценивания эффективности вычислительных структур, которые отличаются применением модифицированного алгоритма Дейкстра для поиска кратчайших путей между вершинами графовой модели функциональных блоков.
5. Аппаратный прототип квантового вычислителя основе программируемой логики, который позволяет существенно (x10-x100) повысить быстродействие оптимального поиска покрытий в задачах дискретной оптимизации.
6. Верификация моделей, методов и архитектуры квантового вычислителя позволяет использовать их в качестве компонентов инфраструктуры сервисного обслуживания и встроенного ремонта цифровых систем для существенного (5%) повышения качества SoC.

Обоснованность подтверждается результатами экспериментальных исследований, обработкой 9 реальных цифровых проектов. Результаты экспериментов подтверждают достаточную эффективность для (5%) повышения качества SoC по сравнению с существующими академическими аналогами.

Программно-аппаратные средства в виде аппаратных моделей и встроенной инфраструктуры ремонта, являются валидными и используются в качестве методологического дополнения к системе Riviera при проведении проектных работ, связанных с созданием цифровых систем на кристаллах, имплементируемых в программируемую логику.

УКРАЇНА ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ "АЛДЕК-ХАРКІВ" №33817534

27.08.2012г. Директор ООО «Алдек-Харьков», к.т.н. Зайченко С.А.

УТВЕРЖДАЮ

проректор по научной работе ХНУРЭ,
_____ проф. Сляпченко Н.И.

АКТ

10.09.2012 г. Харьков

О внедрении инфраструктуры встроенного ремонта аппаратных компонентов цифровых систем на кристаллах в учебный процесс ХНУРЭ. Составлен комиссией, созданной на основании распоряжения проректора в составе:

Председатель комиссии: Чумаченко С.В., и.о. зав. каф. АПВТ
Члены комиссии: Хаханова И.В., профессор каф. АПВТ
Кулак Э.Н., доцент каф. АПВТ

В период с 03.09.2012 по 08.09.2012 года комиссия провела работу по определению внедрения в учебный процесс моделей, методов и программно-аппаратных средств инфраструктуры встроенного тестирования аппаратных компонентов цифровых систем на кристаллах, разработанных на кафедре автоматизированного проектирования вычислительной техники Харьковского национального университета радиоэлектроники при участии (30%) соискателя Мурад Али Абас.

Комиссией установлено, что предложенные аппаратно-ориентированная модель параллельного вычисления булеана, автоматная модель комбинационного устройства, кубитная модель данных, суперпозиционный метод синтеза кубов функциональностей и метод оценивания эффективности вычислительных структур реализованны в программно-аппаратных средствах встроенного ремонта цифровых систем на кристаллах, выполняющих следующие функции:

1. Кубитное форматирование данных с нечисленным представлением разрядов в двоичном векторе для повышения быстродействия векторных логических операций при решении задач дискретной оптимизации.
2. Параллельное вычисление булеана на основе использования процессорной Хассе-структуры для решения задач покрытия и минимизации функциональных структур.
3. Автономное и встроенное восстановление работоспособности компонентов логических устройств за счет переадресации дефектных примитивов.
4. Оценивание эффективности вычислительных структур на основе модифицированного алгоритма Дейкстра для поиска кратчайших путей между вершинами графовой модели функциональных блоков.
5. Существенное (x10-x100) повышение быстродействия оптимального поиска покрытий в задачах дискретной оптимизации, а также улччшение (на 5%) качества SoC за счет встроенного ремонта цифровых систем.

Программно-аппаратные средства тестирования и ремонта используются при проведении лабораторных работ, связанных с синтезом цифровых систем на кристаллах, имплементируемых в программируемую логику, в процессе изучения курсов: "Проектирование цифровых систем на ПЛИС", "Модели и методы обеспечения качества проектирования цифровых систем".

Председатель комиссии: Чумаченко С.В., и.о. зав. каф. АПВТ
Члены комиссии: Хаханова И.В., профессор каф. АПВТ
Кулак Э.Н., доцент каф. АПВТ

ALDEC
The Design Verification Company

ТОВ «Алдек-КТС»
пр.Леніна 18/9, оф. 1
61166, м.Харків, Україна

ЕДРПОУ 36374067
Р/р 26008154763 в АТ ХОД «Райффайзен Банк Аваль»,
МФО 380805
Тел.: (+38 057) 760-47-25

Исх. номер 32/13 от 15.05.13 г.

СПРАВКА

о внедрении результатов диссертационной работы
Альмадхоуна Самера М..
" Методы поиска ошибок проектирования в моделях цифровых устройств на языках описания аппаратуры»"

Методы поиска ошибок проектирования в моделях цифровых устройств, представленных на языках описания аппаратуры, разработанные в Харьковском Национальном Университете Радиоэлектроники при участии аспиранта Альмадхоуна Самера М. (30%) позволяют достигнуть снижения трудоемкости верификации HDL-моделей цифровых устройств функционально-блочного уровня при автоматизированном проектировании.

Разработанная методика проведения диагностического эксперимента при верификации HDL-кода позволила автоматически применять различные алгоритмы поиска дефектов в зависимости от формы спецификации и стиля написания HDL-модели. Предложенный метод обратного отслеживания для графовой модели HDL-кода позволил значительно повысить глубину поиска ошибок проектирования (до операторов HDL-кода) в условиях неполной спецификации на проектируемое устройство и значительно сократиnь общее время верификации HDL-модели.

Разработанная на основе результатов исследования методика проведения диагностического эксПеримента используется на предприятии ООО "Алдек-КТС" при автоматизированном проектированбиии цифровых устройств с использованием САПР Active-HDL и системы имитационного моделирования Riviera™ компании Aldec Inc. (USA). Применение разработанной методики позволило на 20-30% процентов сократить время на поиск ошибок проектирования. HDL-моделях и повысить глубину поиска до ошибочного HDL-оператора.

С Уважением,
Директор ООО «Алдек-КТС», к.т.н. Зайченко С. О.

15.05.2013р.

«ЗАТВЕРДЖУЮ»
Проректор ХНУРЕ з НПР
_____ Ясена П. С.
"20" травня 2013 р.

С П Р А В К А

про впровадження в навчальний процес ХНУРЕ результатів дисертаційної роботи Альмадхоуна С.М.. "«Методи пошуку помилок проектування в моделях цифрових пристроїв на мовах опису апаратури», представленої на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.12 – системи автоматизації проектувальних робіт

Комісія у складі: зав.каф. АПОТ проф. Чумаченко С.В., проф. каф. АПОТ Кривулі Г.Ф., доц. каф. АПОТ Кулак Е.М. розглянула матеріали дисертаційної роботи Альмадхоуна С.М, які використовуються в навчальному процесі кафедри АПОТ ХНУРЕ, і прийшла до наступного висновку.

Розроблені у дисертаційній роботі методи пошуку помилок проектування у HDL-моделях цифрових пристроїв дозволяють суттєво скороти час на верифікацію HDL-моделей, особливо в умовах неповної специфікації. Методи зворотного відслідковування дозволяють суттєво підвищити глибину пошуку помилок проектування до помилкового HDL-оператора. Зазначені методи використовуються в системах верифікації HDL-моделей при автоматизованому проектуванні цифрових пристроїв, які подані на мовах опису апаратури.

У навчальному процесі кафедри АПОТ ХНУРЕ результати дисертаційної роботи Альмадхоуна С.М. використовуються у таких навчальних дисциплінах.

1. У навчальній дисципліні «Логічне моделювання» для бакалаврів напрямку «Комп'ютерна інженерія» у лекційному матеріалі по темі «Верифікація моделей цифрових пристроїв» та у лабораторних роботах по темі «Побудова мовних моделей цифрових пристроїв».

2. У навчальній дисципліні для бакалаврів напрямку «Комп'ютерна інженерія» «Основи комп'ютерної діагностики» у лекційному матеріалі та практичних заняттях по темі «Методи пошуку дефектів у цифрових пристроях».

3. У навчальній дисципліни «Теорія проектування СКС» для магістрів спеціальності «Спеціалізовані комп'ютерні системи» у лекційному матеріалі по темі «Системи автоматизованого проектування цифрових пристроїв»

Зав.каф. АПОТ проф .. Чумаченко С.В.,
Проф.. каф. АПОТ Кривуля Г.Ф.,
Доц. каф. АПОТ Кулак Е.М

ALDEC
The Design Verification Company

ТОВ «Алдек-КТС»
вул. Космічна 23а
61145, м.Харків, Україна

ЕДРПОУ 36374067
Р/р 26008154763 в АТ ХОД
«Райффайзен Банк Аваль», МФО 380805
Тел.: (+38 057) 760-47-25

СПРАВКА

о внедрении кубитных моделей и методов анализа и диагностирования цифровых устройств в научно-исследовательскую и производственную деятельность
ООО «Алдек–КТС»

Внедрение в научно-исследовательскую и производственную деятельность компании ООО «Алдек–КТС» кубитных моделей и методов анализа и диагностирования цифровых устройств, разработанных на кафедре автоматизированного проектирования вычислительной техники Харьковского национального университета радиоэлектроники при участии (30%) соискателя Багхдади Аммар Авни Аббас позволило повысить быстродействие программных и аппаратных средств анализа цифровых устройств. Установлено, что использование кубитных покрытий функциональных элементов, методов адресного параллельного моделирования, встроенного диагностирования и ремонта позволило значительно повысить выход годной продукции, что обусловлено использованием следующих результатов:

1. Новые кубитные модели описания цифровых систем и компонентов, которые характеризуются компактностью описания таблиц истинности в форме Q-покрытий, позволяют повысить быстродействие программных и аппаратных средств интерпретативного моделирования вычислительных устройств за счет адресной реализации анализа логических примитивов.

2. Новая матричная модель кубитных примитивов для реализации комбинационных схем, которая характеризуется адресным объединением Q-покрытий на элементах памяти, соединенных в цифровую схему с помощью вектора состояний линий, дает возможность восстанавливать работоспособность отказавших логических примитивов путем их переадресации на запасные компоненты при достаточно высоком быстродействии функционирования вычислительного устройства.

3. Новая автоматная MQT-модель цифрового устройства, которая характеризуется использованием только адресуемых структур памяти и операции транзакции для программной и аппаратной реализации комбинационных и последовательностных функциональностей. Это дает возможность создавать быстродействующие и надежные вычислители для проектирования сервисов киберпространства на основе параллельных логических операций и ремонта неисправных адресуемых функциональных примитивов.

4. Новый Q-метод интерпретативного исправного моделирования цифровых схем, который характеризуется использованием компактных Q-покрытий вместо таблиц истинности, позволяет существенно повысить быстродействие анализа схемы за счет адресного формирования выходов функциональных примитивов и уменьшить объем структур данных.

5. Новый критерий качества диагностирования, который учитывает структуру графа, тестов и ассерционных мониторов, позволяет повысить диагнозопригодность проекта за счет увеличения тестовых сегментов для распознавания эквивалентных неисправных блоков или

ALDEC
The Design Verification Company

ТОВ «Алдек-КТС»
вул. Космічна 23а
61145, м.Харків, Україна

ЕДРПОУ 36374067
Р/р 26008154763 в АТ ХОД
«Райффайзен Банк Аваль», МФО 380805
Тел.: (+38 057) 760-47-25

добавления ассерционных мониторов на транзитных вершинах активизированного графа HDL-кода.

Предложенные компоненты инфраструктуры сервисного обслуживания цифровых систем на кристаллах использованы при разработке трех реальных цифровых проектов. Результаты экспериментов показывают, что за счет аппаратной и структурной избыточности на 50% увеличено быстродействие интерпретативного моделирования, на 5% повышен выход годной продукции, на 12% увеличена глубина диагностирования неисправных функциональных блоков и на 15% уменьшено время отладки HDL-кода в процессе проектирования цифровых систем на кристаллах.

Разработанные модели и методы верификации интегрированы в состав методологического обеспечения среды моделирования Riviera фирмы Aldec Inc. И используются при проведении проектных работ, связанных с созданием цифровых систем на кристаллах, имплементируемых в программируемую логику.

С Уважением,
Директор ООО «Алдек-КТС»: _____ / Зайченко С.О. /

02.12.2014р.

«ЗАТВЕРДЖУЮ»
Проректор ХНУРЕ з НПР
_____Лєсна Н. С.
"10" 09 2014 р.

СПРАВКА

про впровадження в навчальний процес ХНУРЕ результатів дисертаційної роботи Кучеренко Д. Ю. «Моделі та методи підвищення якості експертного діагностування комп'ютерних систем з використанням продукційних правил», представленої на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти

Комісія у складі: зав. каф. АПОТ проф. Чумаченко С. В., проф. каф. АПОТ Кривулі Г. Ф., доц. каф. АПОТ Кулак Е. М. розглянула матеріали дисертаційної роботи Кучеренко Д. Ю., які використовуються в навчальному процесі кафедри АПОТ ХНУРЕ, і прийшла до наступного висновку.

Розроблені у дисертаційній роботі моделі та методи аналізу продукційних правил на логічну коректність дозволяють суттєво підвищити достовірність результатів експертного діагностування комп'ютерних систем. Розроблена програма «Production rules analyzer», в основі якої лежать розроблені методи, може бути використана в системах експертного діагностування під час підготовки та аналізу бази знань, що представлена у вигляді продукційних правил.

У навчальному процесі кафедри АПОТ ХНУРЕ результати дисертаційної роботи Кучеренко Д. Ю. використовуються у таких навчальних дисциплінах.

1. У навчальній дисципліні «Моделі та засоби діагностування ПЗ» для бакалаврів напрямку «Комп'ютерна інженерія» у лекційному матеріалі на тему «Експертне діагностування ПЗ».
2. У навчальній дисципліни «Комп'ютерні системи на базі нечіткої логіки» для спеціалістів спеціальності «Спеціалізовані комп'ютерні системи» у лекційному матеріалі на тему «Нечітка логіка у експертних системах діагностування» та у лабораторних роботах на тему «Побудова системи підтримки прийняття рішення на базі нечіткої логіки».
3. У навчальній дисципліни «Теорія проектування СКС» для магістрів та спеціалістів спеціальності «Спеціалізовані комп'ютерні системи» у лекційному матеріалі на тему «Експертні системи діагностування» та у лабораторних роботах на тему «Побудова експертних систем».

Зав. каф. АПОТ проф . Чумаченко С. В.

Проф. каф. АПОТ Кривуля Г. Ф.

Доц. каф. АПОТ Кулак Е. М.

Украина, 61093, г. Харьков, пер. Верховский, 13
тел./факс: (057) 714-95-59, 714-98-19, 714-29-89
E-mail: way@way.kharkov.ua
http://www.way.kharkov.ua

Частное акционерное общество
ПУТЬ
Украинское ремонтно-строительное предприятие

СПРАВКА

о внедрении результатов диссертационной работы
Кучеренко Дарии Ефимовны
«Модели и методы повышения качества экспертного диагностирования компьютерных систем с использованием продукционных правил»

Модели и методы повышения качества экспертного диагностирования компьютерных систем, разработанные в Харьковском национальном университете радиоэлектроники при участии аспирантки Кучеренко Д. Е. (30%), позволяют получить достоверные результаты диагностирования за счет проверки базы знаний экспертной системы диагностирования, представленной в виде продукционных правил, на логическую корректность.

Усовершенствованная модель представления диагностических признаков компьютерной сети в виде лингвистических переменных позволила использовать единый подход к построению базы знаний эксперта, а предложенные методы проверки базы продукционных правил на логическую корректность – повысить достоверность результатов экспертного диагностирования при определении технического состояния корпоративной компьютерной сети (ККС).

Результаты диссертационного исследования используются на предприятии ЧАО «УРСП Путь» при экспертном оценивании качества ККС. Предложенная модель объекта диагностирования позволяет формализовать процесс выбора и оценивания диагностических признаков, характеризующих текущее состояние сети. Использование экспертной системы диагностирования позволило администратору сети получить объективную интегральную характеристику качества её работы и сократить время проведения диагностического эксперимента.

С уважением,

Директор ЧАО «УРСП Путь»: _____ / Коваленко А.А. /

"11" сентября 2014 г.


THE INNOVATION PROJECTS COMPETITION
it
KHARKIV
WINNER
2013
it
KHARKIV
Лучший проект
в области облачных вычислений
Cloud Green Traffic Control – интеллектуальное облако
управления инфраструктурой дорожного движения
Мизь Владимир Александрович

ДИПЛОМ

н а г о р о д ж у є т ь с я

фіналіст Всеукраїнського конкурсу
студентських наукових робіт за напрямком

«Інформатика, обчислювальна техніка
та автоматизація»

Білоус Василь Віталійович

що зайняв ІІ місце за наукову роботу

«Моделі аналізу ефективності
обчислювальних структур»

Голова конкурсної комісії
Проректор Севастопольського
національного технічного
університету — В.О. Крамарь

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ * СЕВАСТОПОЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ * Код 02070973

м. Севастополь – 2013 р.

Гр №6

ДИПЛОМ

нагороджується

фіналіст Всеукраїнського конкурсу
студентських наукових робіт за напрямком

«Інформатика, обчислювальна техніка
та автоматизація»

Скоробогатий Михайло Володимирович

що зайняв ІІ місце за наукову роботу

«Моделі аналізу ефективності
обчислювальних структур»

Голова конкурсної комісії
Проректор Севастопольського
національного технічного
університету В.О. Крамарь

м. Севастополь – 2013 р.

Гр №6

Київський університет імені Бориса Грінченка
1874

Microsoft

SITE
SOCIETY FOR INFORMATION TECHNOLOGY AND TEACHER EDUCATION

This certifies that

Zyarmand Arthur

Has completed the Microsoft Technology Enriched Instruction

FACULTY DEVELOPMENT WORKSHOP

And is recognized as a

MICROSOFT FACULTY FELLOW

Viktor Ogneviuk
Rector of Borys Grinchenko Kyiv University
Doctor of Philosophy, Professor
Full Member of the National Academy
of Pedagogical Sciences of Ukraine

James Garner Ptaszynski, Ph.D.
Senior Director, World Wide Higher Education
Microsoft

Michael Searson, Ph.D.
Executive Director
School for Global Education
& Innovation, Kean University
President, SITE


ENGENSEC
Educating the Next generation
experts in Cyber Security
Certificate
given to
Oleksandr Adamov
For completition of the following training sessions:
Train-the-Trainer, Project Management, Intercultural Competence,
Introduction to Distant Learning Systems
part of
ENGENSEC Programme
Budenheim, Germany – October 2014
Bundeskriminalamt
Heiko Held
BKA – Unit KJ 32

IT-KHARKIV

КОНКУРС ІННОВАЦІЙНИХ ПРОЕКТІВ

INNOVATION PROJECTS COMPETITION

it KHARKIV

ДИПЛОМ

НАГОРОДЖУЄТЬСЯ

Чумаченко Світлана Вікторівна
завідувач кафедри

Литвинова Євгенія Іванівна
професор кафедри

Зіарманд Артур Нісарович
асистент кафедри
Кафедра автоматизації проектування обчислювальної техніки
Харківського національного університету радіоелектроніки

ЗА 3 МІСЦЕ У НОМІНАЦІЇ

Кращий проект в області геоінформаційних систем та інформаційного забезпечення транспорту

ПРОЕКТ

Створення системи «Розумні дороги» для організації інфраструктури дорожнього руху (Smart Roads Infrastructure)

Голова організаційного комітету конкурсу інноваційних проектів «IT-KHARKIV» — *А.В. Моченков*

it-kharkiv.com.ua

28 листопада 2013 р.


Technically Co-Sponsored by
IEEE
IEEE computer society
tttc
EAST – WEST
DESIGN & TEST
IEEE CS TTTC
Diploma
is awarded to
Artur Ziarmand
For the best Regular Paper on EWDTS'13 Symposium and outstanding contribution in Design & Test
Chairs of EWDTS'13
Yervant Zorian
Vladimir Hahanov
EWDTS
2013
Rostov-on-Don, Russia, 27-30 September, 2013

Kaspersky® Academy

CyberSecurity

for the Next Generation

International Student Conference

DIPLOMA

Artur Ziaemand

Best presentation

in Russia and CIS Round of "CyberSecurity for the Next Generation",
the international student conference on computer security issues

21 February, 2014
Moscow

Eugene Kaspersky,
CEO Kaspersky Lab
Chairman of the Conference

KASPERSKY lab

Kaspersky® Academy

CyberSecurity

for the Next Generation

International Student Conference

CERTIFICATE

Artur Ziarmand

participated in Russia and CIS Round 2014 of "CyberSecurity for the Next Generation", the international student conference on computer security issues held in Kaspersky Lab Office on 19–21 February 2014

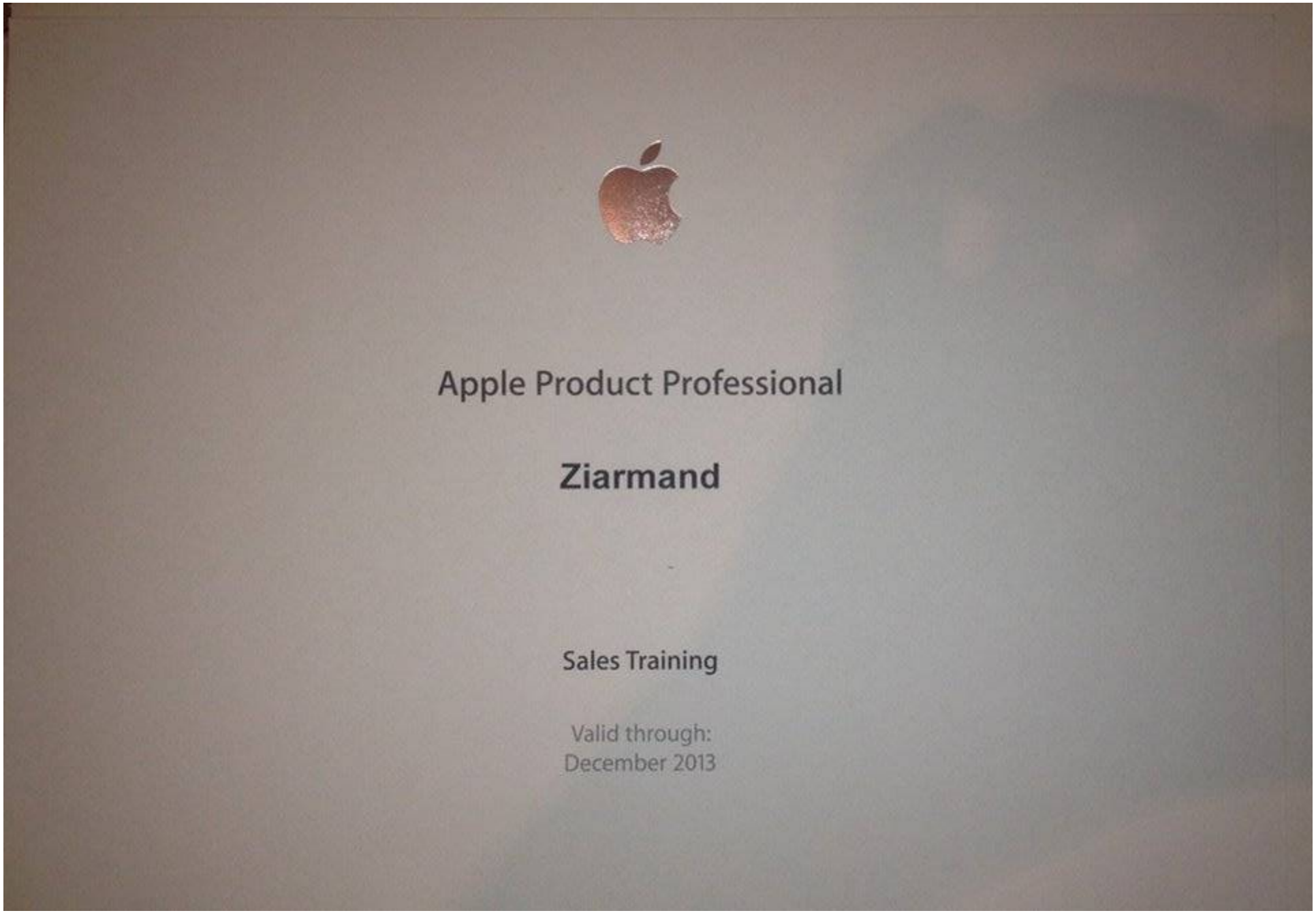
and submitted the research paper:
iCloud Traffic Control

Moscow, 2014

Eugene Kaspersky,
CEO Kaspersky Lab
Chairman of the Conference

KASPERSKY lab


IEEE
IEEE computer society
tttc
EAST – WEST DESIGN & TEST IEEE CS TTTC
Diploma
is awarded to
Artur Ziarmand
For the best Regular Paper on EWDTS'14 Symposium and outstanding contribution in Design & Test
Chairs of EWDTS'14
Yervant Zorian
Vladimir Hahanov
IEEE EAST-WEST DESIGN & TEST INTERNATIONAL SYMPOSIUM
EWDTS
2014
Kiev, Ukraine, 29 September, 2014


Apple Product Professional
Ziarmand
Sales Training
Valid through:
December 2013

IT-KHARKIV

КОНКУРС ІННОВАЦІЙНИХ ПРОЕКТІВ

INNOVATION PROJECTS COMPETITION

it KHARKIV

ДИПЛОМ

НАГОРОДЖУЮТЬСЯ

Хаханов Володимир Іванович
декан факультету комп'ютерної інженерії та управління
Харківського національного університету радіоелектроніки

Енглезі Ірина Павлівна
ректор Донецької академії автомобільного транспорту

Мізь Володимир Олександрович
аспірант кафедри автоматизації проектування обчислювальної техніки
Харківського національного університету радіоелектроніки

ЗА ПЕРЕМОГУ У НОМІНАЦІЇ

Кращий проект в області хмарних обчислень

ПРОЕКТ

Інтелектуальна хмара управління інфраструктурою дорожнього руху (Cloud Green Traffic Control)

Голова організаційного комітету конкурсу інноваційних проектів «IT-KHARKIV»

А.В. Моченков

it-kharkiv.com.ua

28 листопада 2013 р.

ВИСНОВКИ

Існуючі програмні продукти та публікації практично не пропонують асоціативно-логічних технологій пошуку, розпізнавання та прийняття рішень в дискретно інформаційному просторі [63], що складається з big data. Практично всі вони використовують універсальну систему команд сучасного коштовного процесора з математичним співпроцесором. У той же час, апаратні спеціалізовані засоби логічного аналізу, які є їх прототипами [118], як правило, орієнтовані на побітову або невекторную обробку інформації.

Запропонований новий підхід векторно-логічної обробки великих даних з повним виключенням арифметичних операцій, що впливають на швидкодію і апаратну складність, може бути ефективно реалізований на основі використання як сучасних мультипроцесорних цифрових систем на кристалах, так і за допомогою віртуальних паралельних процесорів, що функціонують під егідою кіберфізичних систем або хмарних сервісів-фільтрів. Фактична реалізація підходу заснована на пропозиції інноваційних моделей і методів, які використовують ідею векторно-логічної метрики кіберпростору:

1. Метрика аналізу кіберпростору (big data) за допомогою віртуальних процесорів, яка характеризується використанням єдиної логічної xor-операції для визначення кібер-відстані шляхом циклічного замикання не менше одного об'єкта, що дає можливість на порядок підвищити швидкодію аналізу big data і підрахунок нечисельних структурних критеріїв якості взаємодії інформаційних об'єктів на основі використання векторних логічних операцій для точного пошуку, розпізнавання образів і прийняття рішень.

2. Нова структурно-цифрова модель віртуального процесора для точного пошуку інформації, яка характеризується використанням векторних функцій приналежності запиту до інформаційних компонентів big data і подальшого їх порівняння для визначення мінімальної відстані, що дає

можливість вибрати оптимальний розв’язок при істотному зменшенні часу пошуку за рахунок повного виключення арифметичних операцій.

3. Нова автоматна модель віртуального процесора, яка характеризується транзакційною взаємодією компонентів пам'яті, виконуючих роль комбінаційних і послідовних елементів, реалізованих у формі кубітних або «квантових» примітивів, що дає можливість створювати паралельні віртуальні комп'ютери для ефективного вирішення завдань аналізу big data без наявності арифметичних команд і забезпечувати високу швидкодію хмарно-орієнтованих процесорів. Удосконалене визначення адресно-автоматної моделі віртуального комп'ютера <MQT> як організація структури функціональних примітивів пам'яті без гальванічних або дротових зв'язків, на яких визначені адресні транзакції даних в часі і просторі для аналізу об'єктів в кіберпросторі big data. Запропоновано Q-метод синтезу обчислювальних пристроїв, який характеризується використанням векторно-кубітної форми компактного опису структурних компонентів, що дає можливість суттєво підвищити швидкодію адресно-орієнтованого моделювання логічних елементів і зменшити розмірність пам'яті для зберігання структуру даних цифрового пристрою.

4. Нова кіберінформаційна модель аналізу big data, що використовує засоби хмарних сервісів, кіберфізичні системи, паралельні віртуальні мультипроцесори з мінімальним набором векторно-логічних операцій для точного пошуку інформації на основі запропонованої булеанової метрики і векторно-логічних критеріїв якості, що дає можливість поступової класифікації та впорядкування хаотичних даних big data в масштабах кіберекосистеми планети.

5. Практична значимість запропонованих моделей полягає в необхідності реструктуризації кіберпростору шляхом заміни концепції аморфних big data на семантично класифікуовану інформаційну інфраструктуру корисних даних, призначених для управління кіберфізичними процесами. У зв'язку з цим запропоновано напрями

формування технологічної культури big data для поступового підвищення рівня корисної інформації від 0,4% до 10% шляхом компетентнісної інфраструктуризації кіберпростору великих даних.

6. Нова модель векторно-логічного SIMD-мультипроцесора, який характеризується відсутністю арифметичних операцій, паралельним обчисленням відстані між запитом та інформаційними квантами, а також одночасним визначенням кращого з можливих n-рішень за мінімумом функції приналежності, що дає можливість на порядок підвищити швидкодію максимально точного пошуку даних в big data.

Основна інноваційна наукова та ринково-орієнтована ідея виконаного дослідження полягає в створенні структур даних, метриці, автомата і прототипу віртуального процесора для синтезу кіберфізичних систем: Smart Cloud Traffic Control, Smart Cyber University, Smart Big Data Analytics, які при їх функціонуванні вилучають присутність людини.

Майбутні дослідження спрямовані на проектування big data driven cyber physical systems, які орієнтовані на постійну метрико-семантичну реструктуризацію кіберпростору з метою зручного витягання знань, а також на перетворення соціальних відносин неприродного світу шляхом передачі управління від людини до хмарних сервісів.

СПИСОК ПОСИЛАНЬ

1. *Дуплий В.А.* Квантовая информация, кубиты и квантовые алгоритмы / С.А. Дуплий, В.В. Калашников, Е.А. Маслов // Вісник Харківського університету. – №657. – 2005. – С. 99-104.

2. *Нильсен М.* Квантовые вычисления и квантовая информация / М. Нильсен, И. Чанг. Пер. с англ. – М.: Мир, 2006. – 824 с.

3. *Бекман И.Н.* Лекции по информатике // Электронный ресурс http://www.twirpx.com/file/602601/

4. *Benenti G., Casati G., Strini G.* Principles of Quantum Computation and Information. Volume 1: Basic Concepts. World Scientific. – 2004. – 256 p.

5. *Vedral V., Plenio M.B.* Basics of Quantum Computation. 1998. 28 p. Электронный ресурс: arXiv:quant-ph/9802065 v1 25 Feb 1998 http://www.tfp.uni-karlsruhe.de/~cuevas/Lehre/SS04/9802065.pdf

6. *Rosinger E.E.* Basics of Quantum Computation (Part 1). – 2004. – 87 p. Электронный ресурс: arXiv:quant-ph/0407064 v1 8 Jul 2004 http://chaos.swarthmore.edu/courses/TSG/2004d.pdf

7. *Stenholm S., Suominen K.-A.* Quantum approach to informatics. A John Wiley & Sons, Inc., Publication. – 2005. – 249 p.

8. *Imai Hiroshi, Hayashi Masahito.* Quantum Computation and Information. From Theory to Experiment. Springer. – 2006. – 234 p.

9. *Nielsen M.A., Chuang I.L.* Quantum Computation and Quantum Information. Cambridge University Press. – 2010. – 710 p.

10. *Mikio Nakahara.* Quantum computing: an overview // Mathematical Aspects of Quantum Computing. – 2007. – 53p. http://www.worldscibooks.com/physics/6851.html

11. *Whitney M.G.* Practical Fault Tolerance for Quantum Circuits. A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate Division of the University of California, Berkeley. – 2009. – 206 p.

12. *DiVincenzo D.P.* The Physical Implementation of Quantum Computation // IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA. 9 p. Электронный ресурс: arXiv:quant-ph/0002077 v3 13 Apr 2000. http://www.unifiedfieldtheories.com/0002077_DiVincenzo_Phys_Imp.pdf

13. *Svore K.M., Terhal B.M., DiVincenzo D.P.* Local Fault-Tolerant Quantum Computation // Электронный ресурс: arXiv:quant-ph/0410047v2 6 Jun 2005. http://research.microsoft.com/pubs/143764/local2005.pdf

14. *Бейтсон Г.* Шаги в направлении экологии разума / Г. Бейтсон. М.: КомКнига. – 2005. – 248 с.

15. *Валиев К.А.* Квантовые компьютеры: надежды и реальность / К.А.Валиев, А.А.Кокин. Ижевск: РХД. – 2001. – 352 с.

16. *Feinstein D.D.Y.* Computer-Aided-Design Methods for Emerging Quantum Computing Technologies / D.D.Y. Feinstein. BiblioLabsII. – 2011. – 184 p.

17. *Hayes J.P.* Testing for Missing-Gate Faults in Reversible Circuits / John P. Hayes, Ilia Polian, Bernd Becker // Proc. of Asian Test Symposium. Taiwain. November, 2004.

18. *Матюшкин И.В.* Квантовые клеточные автоматы / И.В. Матюшкин // Электронный научный журнал «ИССЛЕДОВАНО В РОССИИ». – 2011. – С. 367-392. Электронный ресурсhttp://zhurnal.ape.relarn.ru/articles/2011/029.pdf

19. *Feinstein D.Y.* Partially Redundant Logic Detection Using Symbolic Equivalence Checking in Reversible and Irreversible Logic Circuits /D.Y. Feinstein, M.A. Thornton, D.M. Miller // Design, Automation and Test in Europe, DATE ’08. – 2008. – P. 1378 – 1381.

20. *Nayeem N.M.* Online Fault Detection in Reversible Logic / N.M. Nayeem, J.E. Rice // Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). – 2011. – P.426-434.

27. *Golubitsky O.* A Study of Optimal 4-bit Reversible Toffoli Circuits and Their Synthesis / O. Golubitsky, D. Maslov // IEEE Transactions on Computers. – 2011. – P. 1-14.

28. *Основы* технической диагностики / Под. ред. П.П.Пархоменко. М.: Энергия, 1976. – 460 с.

29. *Пархоменко П.П.* Основы технической диагностики (Оптимизация алгоритмов диагностирования, аппаратурные средства) / П.П. Пархоменко, Е.С. Согомонян. Под ред. П.П. Пархоменко. М.: Энергия, 1981. – 320 с.

31. *Novak O.* Handbook of testing electronic systems / O. Novak, E. Gramatova, R.Ubar. Czech University Publishing House. – 2005. – 402 p.

32. *Автоматизация* диагностирования электронных устройств / Ю.В. Малышенко и др./ Под ред. В.П. Чипулиса. М.: Энергоатомиздат, 1986. – 216с.

35. *Хаханов В.И., Хаханова И.В., Литвинова Е.И., Гузь О.А.* Проектирование и верификация цифровых систем на кристаллах. Verilog & System Verilog: Харьков. – Новое слово, 2010. – 528с.

36. *Электронный* ресурс: http://www.scrigroup.com/limba/engleza/92/The-Design-Flow-and-Fault-Mode51775.php

37. *Андрюхин А.И.* Параллельная генерация тестов для МОП-структур на переключательном уровне / А.И. Андрюхин // Наукові праці ДонНТУ. Серія “Інформатика, кібернетика та обчислювальна техніка. – Вип. 11(164). 2010. – С. 75-78.

38. *Harris I.G.* Hardware-Software Covalidation: Fault Models and Test Generation / Ian G. Harris // Design and Test of Computers. Vol. 20, Num. 4. July-August 2003. 12 p. Электронный ресурс:

http://www.ics.uci.edu/~harris/pubdir/hldvt01hwsw.pdf

39. *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка VHDL. Харьков: ХНУРЭ, 2003. – 492 с.

40. *Надежность* технических систем / Под ред. И.А.Ушакова. М.: 1985. – 512 с.

41. *Stanisavljevi M.* Reliability of Nanoscale Circuits and Systems / M. Stanisavljevi, M. Schmid, Y. Leblebici. Springer, 2011. – 240 p.

42. *Fan X.* Fault diagnosis of VLSI designs: cell internal faults and volume diagnosis throughput / Xiaoxin Fan // PhD (Doctor of Philosophy) thesis, University of Iowa. – 2012. – 134 p.

43. *Pomeranz I.* Transition Path Delay Faults: A New Path Delay Fault Model for Small and Large Delay Defects / I. Pomeranz, S.M. Reddy // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2008. – Vol.16, No.1. – P. 98–107.

44. *Pomeranz I.* Selection of a Fault Model for Fault Diagnosis Based on Unique Responses / I. Pomeranz, S.M. Reddy // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2010. – Vol.18, No.11. P. 1533-1543.

45. *Vanitha K.* Implementation of an integrated FPGA based automatic test equipment and test generation for digital circuits / K. Vanitha, C.A. Sathiya Moorthy // Intern. Conf. on Information Communication and Embedded Systems (ICICES). – 2013. – P. 741-746.

46. *Niemann H.* Fault tolerant control based on active fault diagnosis / H. Niemann // Proc. of the 2005 American Control Conference. – 2005. – Vol. 3. – P. 2224-2229.

47. *Abramovici M.* BIST-Based Delay-Fault Testing in FPGAs / Miron Abramovici, Charles E. Stroud // Journal of Electronic Testing: Theory & Applications. – 2003. – Vol. 19, No. 5. – P. 549-558.

48. *Ulbricht M.* A new hierarchical built-in self-test with on-chip diagnosis for VLIW processors / Markus Ulbricht, Mario Scholzel, Tobias Koal, Heinrich Theodor Vierhaus // Design and Diagnostics of Electronic Circuits & Systems (DDECS). – April 2011. – P.143-146.

49. *Elm M.* BISD: Scan-based Built-In self-diagnosis / M. Elm, H.-J. Wunderlich // Design, Automation & Test in Europe Conference & Exhibition (DATE). – March 2010. – P. 1243-1248.

50. *Huang Yu-Jen.* A built-in self-test scheme for the post-bond test of TSVs in 3D ICs / Yu-Jen Huang, Jin-Fu Li, Ji-Jan Chen, Ding-Ming Kwai, Yung-Fa Chou, Cheng-Wen Wu // VLSI Test Symposium (VTS). – May 2011. – P.20-25.

51. *Shianling Wu.* Logic BIST Architecture Using Staggered Launch-on-Shift for Testing Designs Containing Asynchronous Clock Domains / Wu Shianling, Laung-Terng Wang, Yu Lizhen, H. Furukawa, Wen Xiaoqing, W.-B. Jone, N.A. Touba, Zhao Feifei, Liu Jinsong, Hao-Jan Chao, Li Fangfang, Jiang Zhigang // Defect and Fault Tolerance in VLSI Systems (DFT). – Oct. 2010. – P. 358-366.

52. *Da Silva F.* The Core Test Wrapper Handbook. Rationale and Application of IEEE Std. 1500™ / F. Da Silva, T. McLaurin, T. Waayers. Springer. – 2006. – XXIX. 276 p.

53. *Marinissen E.J.* Guest Editors' Introduction: The Status of IEEE Std 1500 / E.J. Marinissen, Yervant Zorian // IEEE Design & Test of Computers. – 2009. – No26(1). – P.6-7.

54. *Elm M.* Scan Chain Organization for Embedded Diagnosis / M. Elm, H.-J. Wunderlich // Design, Automation and Test in Europe, DATE '08. – 2008. – P. 468–473.

55. *Michael A. Nielsen & Isaac L. Chuang.* Quantum Computation and Quantum Information. Cambridge University Press. – 2010. – 676p.

56. *Stig Stenholm, Kalle-Antti Suominen*. Quantum approach to informatics. John Wiley & Sons, Inc., 2005. – 249p.

57. *Mark G. Whitney.* Practical Fault Tolerance for Quantum Circuits. PhD dissertation. University of California, Berkeley. 2009. 229p.

58. *Горбатов В.А.* Основы дискретной математики. – М. : Высшая школа. – 1986. – 311 с.

59. *Hahanov V.I., Litvinova E.I., Chumachenko S.V., Baghdadi Ammar Awni Abbas, Eshetie Abebech, Mandefro.* Qubit Model for solving the coverage problem // Proc. of IEEE East-West Design and Test Symposium. IEEE. USA. Kharkov. 14-17 September 2012. P.142 - 144.

60. *Хаханов В.И., Мурад Али Аббас, Литвинова Е.И., Гузь О.А., Хаханова И.В.* Квантовые модели вычислительных процессов // Радиоэлектроника и информатика. – 2011. – №3. – С.35-40.

61. *Проектирование* и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. – Харьков: ХНУРЭ. – 2009.– 484 с.

62. *Хаханов В.И.* Техническая диагностика цифровых и микропроцессорных структур. – К.: ИСИО.– 1995. – 242 с.

63. *Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И.* Структура логического ассоциативного мультипроцессора // Автоматика и телемеханика. – 2012. – № 10. – С. 71-92.

64. *Борисовец Б. Э. , Шаршунов С. Г.* Общая модель и синтез тестов для механизмов управления межрегистровым обменом данными в микропроцессорах // Автоматика и телемеханика. – 1992, № 8. – С. 142–149.

65. *Koal T., Scheit D., Vierhaus H.T.* A comprehensive scheme for logic self repair // Conf. Proc. on Signal Processing Algorithms, Architectures, Arrangements, and Applications.– 2009.– P. -18.

67. *Vladimir Hahanov, Alexander Barkalov and Marian Adamsky.* Infrastructure intellectual property for SoC simulation and diagnosis service. Springer, 2011. – P. 289-330.

68. *Хаханов В.И., Литвинова Е.И., Хаханова И.В., Murad Ali Abbas.* Инфраструктура встроенного восстановления логических PLD-схем // Радиоэлектроника и информатика. 2012. №2. С. 54-57.

69. *Hahanov V., Litvinova E., Gharibi W., Murad Ali Abbas*. Qubit models for SoC Synthesis Parallel and cloud computing. – USA. – 2012. – Vol.1. – Iss 1. P. 16-20.

70. *Hahanov V.I., Litvinova E.I., Chumachenko S.V., Baghdadi Ammar Awni Abbas, Eshetie Abebech, Mandefro*. Qubit Model for solving the coverage problem // Proc. of IEEE East-West Design and Test Symposium. – IEEE. USA. – Kharkov. 2012. P.142 - 144.

71. *Benso A.* IEEE Standard 1500 Compliance Verification for Embedded Cores / A. Benso, S. Di Carlo, P. Prinetto, Y. Zorian // IEEE Transactions on Very Large Scale Integration (VLSI) Systems.– April, 2008.– Vol.16, No.4.– P. 397-407.

72. *Хаханов В.И.* Логический ассоциативный вычислитель / В.И. Хаханов, Е.И. Литвинова, С.В. Чумаченко, О.А. Гузь // Электронное моделирование.– 2011.– № 1.– С. 73-90.

73. *Бондаренко М.Ф.* Инфраструктура мозгоподобных вычислительных процессов / М.Ф. Бондаренко, О.А. Гузь, В.И. Хаханов, Ю.П. Шабанов-Кушнаренко.– Харьков: Новое Слово.– 2010.– 160 с.

74. *Хаханов В.И.* VHDL + Verilog = Синтез за минуты / В.И. Хаханов, И.В. Хаханова. – Харьков: СМИТ.– 2007.– 264 с.

75. *IEEE* Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture IEEE Std 1149.7-2009.

76. *Ubar R.* Block-Level Fault Model-Free Debug and Diagnosis in Digital Systems / R. Ubar, S. Kostin, J. Raik // 12th Euromicro Conference DSD '09. – 2009.– P. 229-232.

77. *Ngene Christopher Umerah.* A diagnostic model for detecting functional violation in HDL-code of SoC / Ngene Christopher Umerah, V. Hahanov // Proc. of IEEE East-West Design and Test Symposium.– Sevastopol, Ukraine.– 19-20 September.– 2011.– P. 299-302.

78. *Baghdadi Ammar Awni Abbas*. Диагностирование HDL-моделей систем на кристаллах / Baghdadi Ammar Awni Abbas, В.И. Хаханов, Е.И. Литвинова, С.А. Зайченко // Радиоэлектроника и информатика.– 2013.– №4.– С.64-72.

79. *Bergeron, Janick*. Writing testbenches: functional verification of HDL models. – Boston: Kluwer Academic Publishers, 2001. – 354 с.

80. *Samir Palnitkar*. Verilog HDL. A guide tp digital design and synthesis.– SunSoft Press.– 1996.– 396p.

81. *Bhasker, J.* Verilog HDL Synthesis. Practical Primer. – Allentown: Star Galaxy Publishing, 1998. – 215 p.

82. *Donald E. Thomas. Philip R. Moorby.* The Verilog Hardware Description Language.– New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers.– 2002– 404 p.

83. *James M. Lee.* VERILOG QUICKSTART. A Practical Guide to Simulation and Synthesis in Verilog. .– New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers.– 2002– 378 p

84. *Janick Bergeron, Eduard Cerny, Alan Hunter, Andrew Nightingale*. Verification Methodology Manual for SystemVerilog.– New York: Springer.– 2005. – 338 p.

85. *Bhasker J.* Verilog HDL, Third Edition.– Star Galaxy Publishing.– 2005.– 568 p.

86. *IEEE* Std 1364.1-2002. IEEE Standard for Verilog® Register Transfer Level Synthesis – IEEE Computer Society Sponsored by the Design Automation Standards Committee.– Published by The Institute of Electrical and Electronics Engineers, Inc.– New York.– 2002.– 109 p.

87. *IEEE* Std 1364-1995. IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language– IEEE Standard Verilog Hardware Description Language.– IEEE Computer Society Sponsored by the Design Automation Standards Committee. – Published by The Institute of Electrical and Electronics Engineers, Inc.– 345 East 47th Street, New York, NY 10017-2394, USA – 1996.– 675 p.

88. *IEEE* Std 1364-2001. (Revision of IEEE Std 1364-1995).– IEEE Standard Verilog Hardware Description Language. – IEEE Computer Society Sponsored by the Design Automation Standards Committee. – Published by The Institute of Electrical and Electronics Engineers, Inc.– 3 Park Avenue, New York, NY 10016-5997, USA. – 2001.– 791p.

89. http://www.xilinx.com

90. http://www.xilinx.com/products/design-tools/ise-design-suite/

91. http://www.python.org

92. *Kenneth A.* Lambert. Fundamentals of Python: From First Programs Through Data Structures. – Boston : Course Technology, 2010. – 945p.

93. *Лутц М.* Программирование на Python, том I, 4-е издание. – СПб.: Символ-Плюс, 2011. – 992 с.

94. *Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S.* Information analysis infrastructure for diagnosis // Information an international interdisciplinary journal. 2011. Japan. Vol.14. № 7. P. 2419-2433.

95. *Bishop M.* About Penetration Testing // IEEE Security & Privacy.– 2007.– Vol. 5, Iss. 6.– P. 84 – 87.

96. *Mainka C., Somorovsky J., Schwenk J.* Penetration Testing Tool for Web Services Security // IEEE Eighth World Congress on Services.– 2012.– P. 163 – 170.

97. *Salas P.A.P., Padmanabhan Krishnan, Ross K.J.* Model-based Security Vulnerability Testing // 18th Australian Software Engineering Conference.– 2007.– P. 284 – 296.

98. *Bau Jason, Bursztein Elie, Gupta Divij, Mitchell John.* State of the Art: Automated Black-Box Web Application Vulnerability Testing // 2010 IEEE Symposium on Security and Privacy.– 2010.– P. 332 – 345.

99. *Shahriar H., Zulkernine M.* Automatic Testing of Program Security Vulnerabilities // 33rd Annual IEEE International Computer Software and Applications Conference.– 2009.–Vol. 2.– P. 550 – 555.

100. *Sedaghat S., Adibniya F., Sarram M.-A.* The investigation of vulnerability test in application software // International Conference on the Current Trends in Information Technology (CTIT).– 2009.– P.1-5.

101. *Wilhelm T.* Professional Penetration Testing.– Syngress.–2009.– 524 p.

102. *Shakeel A., Heriyanto T.* BackTrack 4: Assuring Security by Penetration Testing.– Packt Publishing.– 2011.– 392 p.

103. *Brown S.D., Vranesic Z.G.* Fundamentals of digital logic with VHDL design. USA: McGraw-Hill Companies, 2000. 840p. www.mhhe.com/brawnvranesic.

104. *Ben Bennetts*. DFT course. http://www.ben@dft..co.uk

105. *Math* resources: http://www.math.com/

106. *Хаханов В.И., Хаханова И.В., Литвинова Е.И., Гузь О.А.* Проектирование и верификация цифровых систем на кристаллах. – Харьков: Новое слово. – 2010. – 528с.

107. *Hasan Alkhatib, Paolo Faraboschi, Eitan Frachtenberg, Hironori Kasahara, Danny Lange, Phil Laplante, Arif Merchant, Dejan Milojicic, Karsten Schwan*. IEEE CS 2022 Report. – IEEE Computer Society. – 2014. – 163 p.

108. [http://www.tssonline.ru/articles2/fix-corp/rost-obema-informatsii--realii-tsifrovoy-vselennoy#sthash.rpNOdQLF.dpuf]

109. *Mayer-Schönberger V.* Big Data: A Revolution that Will Transform How We Live, Work / V. Mayer-Schönberger, K. Cukie / Виктор Майер-Шенбергер, Кеннет Кукьер. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим.– Изд-во: Манн, Иванов и Фербер.– 2013.– 240 с.

110. *Demchenko Y., de Laat C., Membrey P.* Defining architecture components of the Big Data Ecosystem // International Conference on Collaboration Technologies and Systems (CTS).– 2014.– P. 104 – 112.

111. *Grolinger K., Hayes M., Higashino W.A., L'Heureux A., Allison D.S., Capretz M.A.M.* Challenges for MapReduce in Big Data // IEEE World Congress on Services (SERVICES). – 2014.– P. 182 – 189.

112. Lichen Zhang. A framework to specify big data driven complex cyber physical control systems // International Conference on Information and Automation (ICIA).– 2014.– P. 548 – 553.

113. *Lichen Zhang.* Designing big data driven cyber physical systems based on AADL // International Conference on Systems, Man and Cybernetics (SMC).– 2014.– P. 3072 – 3077.

114. *Michalik P., Stofa J., Zolotova I.* Concept definition for Big Data architecture in the education system // 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI).– 2014.– P. 331 – 334.

115. Munoz M. Space systems modeling using the Architecture Analysis & Design Language (AADL) // International Symposium on Software Reliability Engineering Workshops (ISSREW).– 2013.– P. 97 – 98.

116. *Курош А.Г.* Курс высшей алгебры. Изд-во: Москва: Наука, 1968. – 426с.

117. *Ariane Hellinger, Ariane Hellinger, Heinrich Seeger.* Cyber-Physical Systems. Driving force for innovation in mobility, health, energy and production. Acatech. National Academy of Science and Engineering. – 2011. – 48p.

118. *Vladimir Hahanov, Wajeb Gharibi, Kudin A.P., Ivan Hahanov, Ngene Cristopher (Nigeria), Tiekura Yeve (Côte d'Ivoire), Daria Krulevska, Anastasya Yerchenko, Alexander Mishchenko, Dmitry Shcherbin, Aleksey Priymak.* Cyber Physical Social Systems – Future of Ukraine // Proceedings of 12th IEEE EWDT Symposium, Kiev, Ukraine, September 26-29, 2014. – P. 67-81.

119. *Han Hu, Yonggang Wen, Tat-Seng Chua, Xuelong LiP.* Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. IEEE Explore: 2014. ISSN: 2169-3536. – P. 652 – 687.

120. *Fadi H. Gebara, H. Peter Hofstee, and Kevin J. Nowka,* IBM Research–Austin. Second-Generation Big Data Systems. IEEE Computer magazine. 2015, January. – P. 36-41.