УДК 658:512.011; 681.326; 519.713

ХМАРНИЙ СЕРВІС ДЛЯ ТЕСТУВАННЯ І ВЕРИФІКАЦІЇ СИСТЕМ НА КРИСТАЛАХ

ЛИТВИНОВА Є.І., ЄМЕЛЬЯНОВ І.В., ХАХАНОВ І.В.

Описуються методи і реалізація апаратно-програмного забезпечення безумовного паралельного синтезу тестів на основі булевих похідних для логіки, поданої у вигляді чорної скриньки і описаної кубітним покриттям. Наводяться теоретичні основи застосування методів і оцінка їх ефективності для широкого класу цифрових схем, реалізованих в програмованих логічних пристроях. Пропонуються інноваційні методи обчислення логічних похідних і дедуктивного моделювання несправностей для функціональних елементів, описаних кубітним покриттям.

Ключові слова: синтез тестів, проектування і верифікація SoC, кубітне покриття, цифрова схема, моделювання несправностей, булева похідна.

1. Реалізація логічної функціональності на елементах пам'яті

Поняття адресного виконання логічних операцій, реалізованих на елементах пам'яті LUT в програмованих логічних пристроях (PLD), дає потенційну можливість створювати на кристалі тільки адресний простір, максимально технологічний для вбудованого відновлення працездатності всіх компонентів, що беруть участь у формуванні функціональності [1-3]. Актуальність створення адресного простору для компонентів підтверджується розподілом логіки і пам'яті на кристалі, представленим на рис. 1, де після 2020 року на чіпі буде тільки один відсоток логіки і 99 відсотків пам'яті.

Тенденція до збільшення пам'яті дає можливість вбудованого відновлення працездатності відмовлених осередків за рахунок виділених додаткових ресурсів для їх ремонту (spare logic cells). Проблема автономного усунення дефектів (самовідновлення працездатності) логічних елементів пов'язана з відсутністю у них адреси. Але вирішити її можна, якщо зв'язки між елементами логіки зробити гнучкими за допомогою програми опису структури, вміщеної в пам'яті, яка з'єднає логічні компоненти в схему. Крім структури взаємодії елементів, пам'ять повинна містити порядок їх обробки. У разі виникнення дефекту в одному з адресованих логічних елементів система вбудованого тестування відновить його працездатність шляхом переадресації на завідомо справний аналог з ремонтного запасу. Просто вирішується проблема підвищення якості та надійності цифрових систем на кристалах шляхом створення інфраструктури вбудованого тестування, діагностування, оптимізації та відновлення працездатності за рахунок апаратної надмірності і зменшення швидкодії виконання функціональних операцій [2-5].

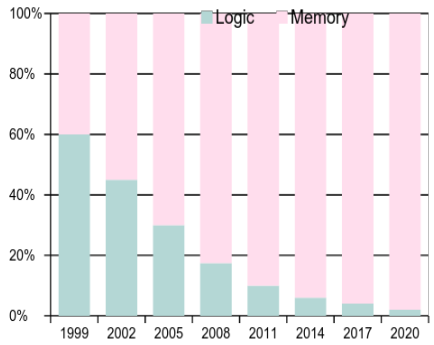

Рис. 1. Розподіл пам'яті і логіки на кристалі

Мета – істотне підвищення швидкодії синтезу тестів і дедуктивної верифікації для blackbox функціональностей логічних компонентів за рахунок використання компактних описів у формі кубітних покриттів і паралельного виконання мінімального числа реєстрових логічних операцій (shift, or, not, nxor).

Задачі:

1) Метод генерації тестів для blackbox функціональностей логічних схем на основі використання кубітних покриттів і паралельного виконання реєстрових логічних операцій (shift, or, not, nxor).

2) Метод обчислення булевих похідних для синтезу тестів на основі використання кубітних покриттів.

3) Метод синтезу тестів на основі використання булевих похідних, поданих векторами в форматі кубітних покриттів.

4) Метод дедуктивного моделювання несправностей для функціональних елементів, поданих векторами кубітних покриттів.

5) Процесор моделювання несправностей і справної поведінки на основі кубітного опису функціональностей, інтегруючий розроблені методи в хмарний або SoC-інфраструктурний сервіс тестування.

Сутність дослідження – розробка методів генерування вхідних тестових послідовностей і оцінки їх якості для функціональних логічних компонентів шляхом паралельного виконання реєстрових логічних операцій (shift, or, not, nxor) над кубітним покриттям і його похідними в структурі процесора кубітного моделювання.

Тестування і діагностування запам'ятовуючих пристроїв реалізується за допомогою спеціальних алгоритмів, що генерують тести: марш; нуль і одиниця, що біжать; логарифмічний розподіл [1-5]. Якщо елементи пам'яті виконують функції логічних схем (reusable logic), то для них слід генерувати тести перевірки функціональностей, не прив'язані до фізики процесів зберігання даних. Тому далі розглядається метод синтезу тестів для функціональних схем, представлених у формі black box, опис яких задається кубітним вектором [6-7].

Ринкова привабливість застосування квантових методів обчислень при створенні комп'ютингових структур в кіберпросторі заснована на використанні кубітних моделей даних, орієнтованих на паралельне вирішення завдань проектування, тестування, дискретної оптимізації [5,8], завдяки збільшенню витрат пам'яті. Не вдаючись в деталі фізичних основ квантової механіки, що стосуються недетермінованої взаємодії атомних часток [9,10], далі використовується поняття кубіта як двійковоговектора для спільного і одночасного завдання булеана станів у дискретній області кіберпростору на основі лінійної суперпозиції унітарних кодів, орієнтованих на паралельне виконання операцій. У теорії квантових обчислень, якашвидко розвивається, вектори станів утворюють квантовий регістр з n кубітів, що формують унітарний або гільбертовий [11] простір H, розмірність якого має ступеневу залежність від числа кубітів .

Мотивація нового підходу для проектування комп'ютингових систем обумовлена появою хмарних сервісів у рамках нової кіберкультури InternetofThings, яка представляє собою спеціалізовані системи, що реалізуються в апаратурі або в програмному продукті. Будь-який компонент функціональності, так само як і структура системи, представляється векторною формою, впорядкованою за адресами, таблицею істинності, що реалізується за допомогою пам'яті. Логічні функції в традиційному виконанні reusable logic не розглядаються. Від цього частково зменшується швидкодія, але, з огляду на те, що 94% SoC-кристала становить пам'ять [2,3], останні 6% будуть імплементуватися в пам'яті, що не критично для більшості хмарних сервісів. Практично для створення ефективних комп'ютингових структур слід використовувати теорію, засновану на обчислювальних компонентах високого рівня абстракції: адресована пам'ять і транзакція.

Особливість організації даних в класичному комп'ютері полягає в адресації біта, байта або іншого компонента. Адресація створює проблему в обробці асоціації елементів множини, які не адресуються та не мають порядку за визначенням. Рішенням може бути процесор, де образ універсуму, з n унітарно кодованих примітивів, використовує суперпозицію для формування булеана $|B(A)|=2^n$ усіх можливих станів [4,12].

Коректність використання прикметника «кубітна» для моделей цифрових пристроїв заснована на порівнянні лінійної і булевої алгебри Кантора з алфавітом $A^k = \{0,1, X, \emptyset\}$. Тут перші два символи – примітиви. Третій визначається суперпозицією: $X=0\cup1$. У гільбертовому просторі лінійна алгебра оперує примітивами $\alpha|0\rangle$, $\beta|1\rangle$ кубіта, третій символ також є суперпозицією двох складових: $|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$. У лінійній алгебрі немає символу, відповідного порожній множині, він є похідною від функції, зворотної по відношенню до суперпозиції. В теорії множин такою операцією є перетин, який дає порожню множину на примітивах: $\emptyset = 0\cap1$. У гільбертовому просторі такою операцією є скалярний добуток або функція Дірака $\langle\alpha|b\rangle$ [11], яка має геометричну інтерпретацію:$\langle\alpha|\beta\rangle = |\alpha|\times|\beta|\times\cos\angle(\alpha, \beta)$. Якщо проекції a і b вектора квантового стану ортогональні, виходить:

$$\langle\alpha|\beta\rangle=|\alpha|\times|\beta|\times\cos\angle(\alpha,\beta) = |\alpha|\times|\beta|\times\cos\angle 90° =0.$$

Скалярний добуток ортогональних векторів дорівнює нулю, що є аналогом символу порожньої множини в алгебрі Кантора. Таблиця відповідності алгебри множин та лінійної алгебри, що зображена нижче, підтверджує властивості ізоморфізму між символами булеана і станами кубіт-вектора:

Boolean $A^k =$	0	1	$X = 0 \cup 1$	$\emptyset = 0 \cap 1$
Qubit $\lvert\psi\rangle =$	$\lvert 0\rangle$	$\lvert 1\rangle$	$\alpha\lvert 0\rangle+\beta\lvert 1\rangle$	$\alpha\lvert 0\rangle\lvert\beta\lvert 1\rangle$

.

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

2. Кубітний метод синтезу тестів

Пропонується метод синтезу тестів, що використовує кубітні вектори або Q-покриття функціональних примітивів цифрових пристроїв, який характеризується компактністю опису даних і паралелізмом виконання логічних операцій. Q-покриття є векторна форма опису поведінки цифрового пристрою, де кожен розряд має адресу, що формується двійковими станами його вхідних змінних:

$$Q=(Q_1,Q_2,...,Q_i,...,Q_n),\ Q_i=\{0,1\},\ Q_i=Q(i),\ i=(X_1X_2,...,X_i,...,X_n).$$

Q-тест є векторна форма неявного завдання тестових послідовностей цифрового пристрою, де координати вектора формують впорядковану послідовність двійкових наборів, що подаються на вхідні змінні за правилом:

$$Q_i = Q(i) = \begin{cases} 1 \to (X_1X_2,...,X_i,...,X_n) = i, \\ 0 \to (X_1X_2,...,X_i,...,X_n) = \emptyset. \end{cases}$$

Іншими словами, якщо координата вектора Q(i)=1, то тестовий набір, складений з двійкових розрядів, які формують десяткову адресу i, подається на входи пристрою. В іншому випадку, при нульовому значенні координати Q(i)=0, такий тестовий набір відсутній.

Абстрагуючись від поняття таблиці істинності, пропонується формальний безумовний алгоритм кубітного синтезу тестів для функціональних примітивів на основі Q-покриття, стосовно раніше розглянутого прикладу:

1) Інвертування всіх розрядів Q-покриття функціонального елемента, що має три вхідні змінні:

Q	1	0	0	1	0	0	1	0
$\bar{Q}$	0	1	1	0	1	1	0	1

2) Логічний зсув номерів розрядів інвертованого кубітного вектора відповідно до послідовності номерів:

$$Q_1(X_1) = 45670123,$$
$$Q_2(X_2) = 23016745,$$
$$Q_3(X_3) = 10325476.$$

Тут діє просте логічне правило: для першої вхідної (молодшої) змінної виконується паралельний обмін даними між сусідніми координатами, для другої вхідної змінної реалізується обмін даними, але вже між сусідніми парами координат, для третьої змінної виконується обмін даними між сусідніми тетрадами координат вектора адрес.

Узагальнення операцій зсуву для функціональності, що містить 4 змінних, представлено на рис. 2.

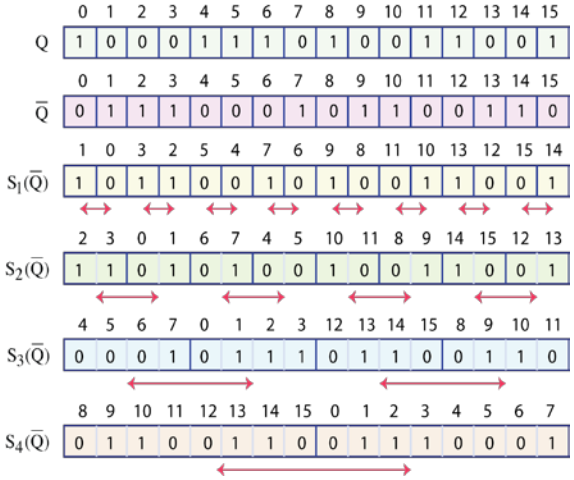

Рис. 2. Операції зсуву для 4-входової функціональності

Збільшення кількості змінних принципово не змінює сутностей зсуву даних: зустрічний зсув двох бітів, пар бітів, тетрад бітів, вісімок, ...

Процедура зустрічного зсуву даних в інвертованому кубітному векторі є найбільш часозатратною в алгоритмі синтезу тестів для функціональних елементів. Тому її швидкодія матиме максимальне значення при апаратній реалізації зсувних операцій.

3) Порівняння за допомогою операції еквівалентності отриманих, в даному випадку, трьох інвертованих і переупорядкованих Q-векторів з вихідним Q-покриттям функціонального елемента:

Q	1	1	0	1	0	1	1	0
$\bar{Q}$	0	0	1	0	1	0	0	1
$S_1(\bar{Q})$	1	0	0	1	0	0	1	0
$S_2(\bar{Q})$	1	0	0	0	0	1	1	0
$S_3(\bar{Q})$	0	0	0	1	0	1	1	0
$T_1 = Q \oplus S_1(\bar{Q})$	1	0	1	1	1	0	1	1
$T_2 = Q \oplus S_2(\bar{Q})$	1	0	1	0	1	1	1	1
$T_3 = Q \oplus S_3(\bar{Q})$	0	0	1	1	1	1	1	1
$T = T_1 \vee T_2 \vee T_3$	1	0	1	1	1	1	1	1

4) Диз'юнкція отриманих трьох векторів формує Q-тест, одиничні координати яких визначають тестові набори для перевірки всіх поодиноких константних несправностей зовнішніх входів і виходів.

З огляду на істотність зустрічних реєстрових зсувів, нижче пропонується універсальний алгоритм отримання перестановок кубітних фрагментів залежно від номера вхідної змінної, що має три вкладених цикли:

1) Завдання і-номера вхідної змінної або кроку 2^i для зустрічного зсуву даних в кубіті: $i = \overline{1, n}$.

2) Формування циклу обробки кубіта з вже заданим кроком 2^i (2,4,8,16…), кратним ступені двійки: $j = \overline{0, 2^n - 1, 2^i}$. Залежно від номера вхідної змінної і формуються такі послідовності індексу j=f(i): [(0,2,4,6…), (0,4,8,12…), (0,8,16,32…)].

3) Задання циклу зустрічного зсуву даних для пари сусідніх груп: $t = j, j + 2^{i-1} - 1$. Значення індексу t=f(i,j): обробка кубіта першої змінної – (0,0; 2,2; 4,4…), другої змінної – (0,1; 4,5; 8,9…), третьої змінної – (0,3; 8,11; 16,19…). Виконання операцій зсуву за допомогою використання буферного регістра B (рис. 3):

$$(B_t = Q_{i,t+j}) \rightarrow (Q_{i,t+j} = Q_{i,t}) \rightarrow (Q_{i,t} = B_t).$$

Кінець алгоритму зсуву даних в регістрах.

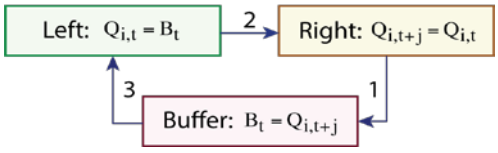

Рис. 3.Зустрічнийзсувсусідніхчастинкубіта

Обчислювальна складність послідовного алгоритму для обробки кубітного покриття функціональності, що має n вхідних змінних, дорівнює $q = 3n2^n$.

Приклад 1. Синтезувати тест для перевірки функціональності, заданої логічним рівнянням:

$$Y = \underline{X}_1X_2\underline{X}_3 \vee X_1\underline{X}_2\underline{X}_3 \vee X_1X_2X_3 \vee \underline{X}_1X_2X_3.$$

Результати виконання алгоритму синтезу тесту за Q-вектором на основі послідовного виконання чотирьох реєстрових логічних операцій (not, shift, nxor, or) представлені у вигляді:

Q	0	0	1	1	1	0	0	1
$\overline{Q}$	1	1	0	0	0	1	1	0
$S_1(\overline{Q})$	1	1	0	0	1	0	0	1
$S_2(\overline{Q})$	0	0	1	1	1	0	0	1
$S_3(\overline{Q})$	0	1	1	0	1	1	0	0
$T_1 = Q \oplus S_1(\overline{Q})$	0	0	0	0	1	1	1	1
$T_2 = Q \oplus S_2(\overline{Q})$	1	1	1	1	1	1	1	1
$T_3 = Q \oplus S_3(\overline{Q})$	1	0	1	0	1	0	1	0
$T = T_1 \vee T_2 \vee T_3$	1	1	1	1	1	1	1	1

На рис. 4 представлений секвенсор синтезу тестів для функціональних елементів, заданих кубітними покриттями. Він містить модуль управління, який розподіляє чотири синхроімпульси, створюючи цикл генерації тесту, що включає 4 паралельні операції, які виконуються послідовно: 0) Початкова стартова операція, що ініціюється сигналом Start, передбачає завантаження в регістр кубітного покриття. 1) Синхроімпульс Clk N активує виконання операції інверсії Not над вмістом регістра кубітного покриття. 2) Синхросигнал Clk S активує виконання операцій зустрічного зсуву над вмістом, в даному випадку трьох, регістрів, отримуючи такі результати: S_1 (not Q), S_2 (not Q), S_3 (not Q). 3) Потім синхросигнал Clk C ініціює виконання паралельних операцій порівняння (в даному випадку для трьох регістрів) отриманих кандидатів в тест з початковим кубітним покриттям: nxor (notQ, S_1), nxor (notQ, S_2), nxor (notQ, S_3). 4) Синхросигнал Clk U активує виконання or-операції над кандидатами в тест, в даному випадку реалізацію логічного об'єднання вмісту трьох регістрів:

$$T = \mathrm{or}[\mathrm{nxor}(\mathrm{notQ}, S_1), \mathrm{nxor}(\mathrm{notQ}, S_2), \mathrm{nxor}(\mathrm{notQ}, S_3)]$$

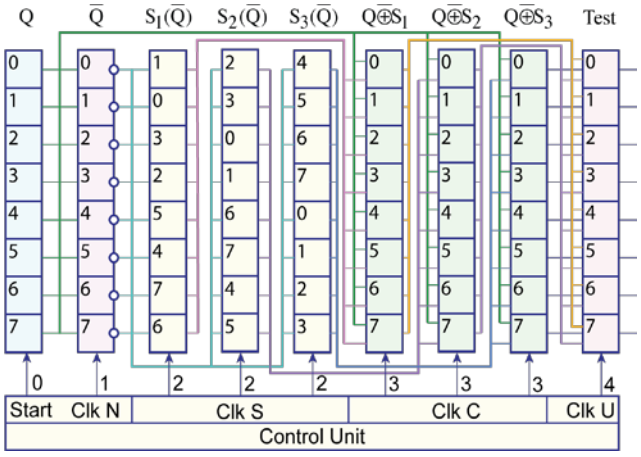

Рис. 4. Секвенсор синтезу тестів

Якщо не економити на апаратурі, то швидкодію тестового генератора можна довести до чотирьох автоматних тактів q=4 паралельного виконання логічних реєстрових операцій, що дає можливість вбудовувати секвенсор в BIST-інфраструктуру цифрових систем на кристалах для online тестування функціональних елементів. При цьому сумарна кількість 2^n- розрядних регістрів дорівнюватиме N(n)=1+1+n+n+1=(2n+3), а загальний обсяг реєстрової пам'яті в бітах буде дорівнювати N(R)=(2n+3) 2^n, де n – число вхідних змінних функціонального елемента.

3. Обчислення булевих похідних для Q-синтезу тестів

Розглянемо метод обчислення булевих похідних по кубітному покриттю для створення умов активізації вхідних змінних при синтезі кубітних тестів. Проведемо аналогію між двома формами булевих функцій для взяття похідних: аналітичною та векторною. Дослідження методу виконаємо на прикладі: $f(x) = \overline{x}_2\overline{x}_3 \vee x_1x_2x_3$.

Питання, що підлягаютьрозв'язанню: 1) Визначення похідних першого порядку за аналітичною та кубітною формами завдання логічної функції. 2) Верифікація отриманих умов активізації шляхом їх моделювання на одній з форм опису функціональності. 3) Синтез тестів активізації змінних логічної функції на основі обчислення похідних.

Кубітний метод взяття булевої похідної:

1) Визначити кубітний вектор функціональності.

2) Виконати xor-операцію для зсунутих назустріч один одному сусідніх частин кубіта: біти, пари, тетради.

3) Результат записати в обидві сусідні частини: біти, пари, тетради:

Y	Y'_1	Y'_2	Y'_3
0	0	0	1
1	1	0	1
0	1	0	1
1	0	0	1
0	0	1	0
0	1	1	0
1	1	1	0
1	0	1	0

Для розглянутого прикладу виконання процедури обчислення похідної за першою змінною має вигляд: {a,b}=a⊕b, (0110,0110)=0101⊕0011. Для отримання похідної за другою змінною величиною слід послідовно xor-скласти сусідні пари кубіт-вектора Y, а загальний результат записати в кожну пару: {a,b}=a⊕b: (00,00)=01⊕01, (11,11)=00⊕11. Для отримання похідної по третій змінній слід послідовно xor-скласти сусідні біти кубіт-вектора Y, а загальний результат записати в кожен біт сусідів: {a,b}=a⊕b: (1,1)=0⊕1, (1,1)=0⊕1, (0,0)= 0⊕0, (0,0)=0⊕0.

Звичайно, що кубіт-похідна за будь-якою вхідною змінною, як вектор, має відносну симетрією рівності підвектора з побудови: похідна першої змінної має симетричну рівність двох тетрад, похідна другої змінної має симетричну рівність

кожних сусідніх пар, похідна третьої змінної має симетричну рівність кожних сусідніх бітів.

Визначити всі похідні для функції трьох змінних: $f(x) = \overline{x}_2\overline{x}_3 \vee x_1x_2x_3$.

Нижче представлений виключно простий процес обчислення похідних на кубітному покритті без розгляду таблиці істинності, що на формальному рівні має такий вигляд:

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

Інтерпретація отриманих кубіт-похідних. Природно, що похідні є функції, задані векторами. Вони можуть бути записані в аналітичній формі (ДНФ) за одиничними значеннями змінних, які формують адреси осередків кубіт-вектора:

$$Y_1' = 011 \vee 111 = \overline{X}_1X_2X_3 \vee X_1X_2X_3 = X_2X_3(\overline{X}_1 \vee X_1) = X_2X_3.$$

$$Y_2' = 000 \vee 010 \vee 100 \vee 101 \vee 110 \vee 111 = \overline{X}_1\overline{X}_3 \vee X_1.$$

$$Y_3' = 000 \vee 001 \vee 100 \vee 101 \vee 110 \vee 111 = \overline{X}_1\overline{X}_2 \vee X_1.$$

Мінімізація булевих функцій, відповідних похідним, приводить до аналітичних виразів, де відсутні змінні, за якими береться похідна. Таким чином, всі результати по обчисленню похідних від трьох форм (аналітична, таблична, векторна) завдання функції є ідентичними. Найбільш технологічним є метод обчислення похідної за кубітним покриттям. Метод має меншу обчислювальну складність в силу компактного представлення функціональності.

Як альтернатива, далі пропонується технологічно простий метод синтезу тестів на основі взяття похідних по кубітних покриттях функціональних елементів.

1) Початкове завдання функціональності кубітним покриттям.

2) Виконання операцій зустрічного зсуву частин кубіт-вектора і подальшого покоординатного хог-підсумовування для отримання векторів похідних для кожної вхідної змінної.

3) Логічне об'єднання векторів похідних, яке формує тест-вектор, що дорівнює за розміром кубітному покриттю.

4) У разі необхідності отримання мінімального тесту вирішується завдання покриття (вже на матриці кубіт-похідних) шляхом знаходження мінімального числа пар одиничних координат кубіт-вектора всіх змінних, де пара одиниць повинна перевіряти поодинокі константні несправності кожного входу.

Рис. 5 ілюструє таблиці процесу отримання тесту відповідно до пунктів алгоритму для функціональності $f(x) = \overline{x}_2\overline{x}_3 \vee x_1x_2x_3$, представленої схемною структурою.

x_1	x_2	x_3	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

→

x_1	x_2	x_3	Y
0	1	1	0
1	1	1	1
1	0	0	1
1	1	0	0
1	0	1	0

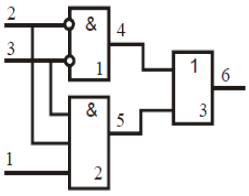

Рис. 5.Отримання тесту для структури булевої функції

З урахуванням виконання пункту 3 алгоритм синтезу дає мінімальний тест перевірки вхідних змінних, який містить 5 наборів, що представлено стовпцем Т, а також продубльовано в явному вигляді правою таблицею. Інтерес представляє той факт, що результат виконання процедури взяття похідної по кубіт-вектору вже містить тест активізації кожної змінної. Об'єднаний тест перевіряє всі константні несправності вхідних змінних, а також може бути використаний для діагностування несправностей, оскільки для істотних входів усі похідні-вектори будуть різними. Фактично взяття похідної за змінною на кубіт-покритті формує Q-тест, не більше і не менше.

4. Дедуктивний аналіз несправностей цифрових структур

Використовується для визначення якості тесту щодо введеного класу несправностей, як правило, одиночних константних. Існує розвинена теорія дедуктивного аналізу [4], орієнтована на паралельну обробку списків несправностей. Пропонується технологічна реалізація дедуктивного моделювання на кубітній формі завдання функціональностей, яка відрізняється від наведеної вище паралелізмом виконання логічних операцій, а також можливістю застосування методу для будь-яких цифрових структур.

Сукупність кубіт-похідних для всіх вхідних змінних, обчислених по кубітному покриттю, являє собою кубітну матрицю для реалізації дедуктивного методу моделювання несправностей. Рядок матриці формує умови для транспортування списків несправностей від зовнішніх входів до виходу за правилом: поодинокі значення створюють об'єднання вхідних списків, а нульові сигнали вказують на входи, списки яких повинні бути відняті з результату об'єднання. Наявність всіх нульових сигналів у рядку створює умови перетину вхідних списків між собою.

Як приклад пропонується побудова дедуктивних формул транспортування списків несправностей

від вхідних змінних до виходу функціональності, заданої вхідними наборами, кубітним покриттям з векторними похідними:

x_1	x_2	x_3	Y	X_1	X_2	X_3
0	0	0	1	0	1	1
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	0	1	0	0
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	1	0	0	0	1	1
1	1	1	1	1	1	1

У загальному випадку формула дедуктивного моделювання логічних функціональностей, представлених у вигляді кубітних векторів, має такий вигляд:

$$L = \vee_{i=1}^{2^n} (x_{i1} \wedge x_{i2} \wedge \dots \wedge x_{ij} \wedge \dots \wedge x_{in}) \wedge (X_{i1} \vee \vee X_{i2} \vee \dots \vee X_{ij} \vee \dots \vee X_{in}),$$

$$L = \vee_{i=1}^{2^n} (x_{i1} \wedge \dots \wedge x_{in}) \wedge (X_{i1} \vee \dots \vee X_{in}),$$

$$L = x \wedge X, x = x_{ij}, X = X_{ij}, i = \overline{1, 2^n}, j = \overline{1, n}.$$

Тут L – список вихідних несправностей; x – матриця вхідних тестових наборів; X – матриця кубітних похідних від кубітного покриття; n – число вхідних змінних.

Алгоритм побудови дедуктивної формули для заданої функціональності включає такі пункти:

1) Завдання кубіт-вектора функціональності.

2) Обчислення кубіт-похідних для вхідних змінних з метою отримання відповідної матриці.

3) Формування аналітичної або матрично-векторної форми обчислення вихідних списків несправностей шляхом логічного множення матриць вхідних тестових впливів і матриці похідних.

Нижче представлений процес обчислення аналітичної і векторної форм для дедуктивного моделювання несправностей логічної функціональності:

$$T = (000,001,010,011,100,101,110,111).$$

$$Q = (011,001,010,100,011,011,011,111)].$$

$$L = (000 \wedge 011) \vee (001 \wedge 001) \vee (010 \wedge 010) \vee \vee (011 \wedge 100) \vee \vee (100 \wedge 011) \vee (101 \wedge 011) \vee (110 \wedge 011) \vee \vee (111 \wedge 111).$$

$$L = (\overline{x}_1\overline{x}_2\overline{x}_3 \wedge \overline{X}_1X_2X_3) \vee (\overline{x}_1\overline{x}_2x_3 \wedge \overline{X}_1\overline{X}_2X_3) \vee \vee (\overline{x}_1x_2\overline{x}_3 \wedge \overline{X}_1X_2\overline{X}_3) \vee \vee (\overline{x}_1x_2x_3 \wedge X_1\overline{X}_2\overline{X}_3) \vee (x_1\overline{x}_2\overline{x}_3 \wedge \overline{X}_1X_2X_3) \vee \vee (x_1\overline{x}_2x_3 \wedge \overline{X}_1X_2X_3) \vee \vee (x_1x_2\overline{x}_3 \wedge \overline{X}_1X_2X_3) \vee (x_1x_2x_3 \wedge X_1X_2X_3).$$

Обчислення векторних форм для дедуктивного моделювання несправностей основних логічних примітивів: or, and, xor представлено в наступній таблиці:

x_1	x_2	$Y^\vee$	$X_1^\vee$	$X_2^\vee$	$Y^\wedge$	$X_1^\wedge$	$X_2^\wedge$	$Y^\oplus$	$X_1^\oplus$	$X_2^\oplus$
0	0	0	1	1	0	0	0	0	1	1
0	1	1	0	1	0	1	0	1	1	1
1	0	1	1	0	0	0	1	1	1	1
1	1	1	0	0	1	1	1	0	1	1

Тут визначені два вектор-стовпці табличного задання вхідних змінних (x_1, x_2), кубіт-вектор функції or – $Y^\vee$, дві похідні $(X_1^\vee, X_2^\vee)$ для кожної вхідної змінної. Далі показані: кубіт-вектор задання функції and і дві колонки похідних, а також кубіт-вектор функції xor і два стовпці похідних по вхідним змінним. Формули дедуктивного моделювання тривіально записуються по рядках:

$$L^\vee = \overline{x}_1\overline{x}_2(X_1 \vee X_2) \vee \overline{x}_1x_2(\overline{X}_1 \vee X_2) \vee x_1\overline{x}_2(X_1 \vee \vee \overline{X}_2) \vee x_1x_2(\overline{X}_1 \vee \overline{X}_2),$$

$$L^\wedge = = \overline{x}_1\overline{x}_2(\overline{X}_1 \vee \overline{X}_2) \vee \vee \overline{x}_1x_2(X_1 \vee \overline{X}_2) \vee x_1\overline{x}_2(\overline{X}_1 \vee X_2) \vee x_1x_2(X_1 \vee X_2),$$

$$L^\oplus = = \overline{x}_1\overline{x}_2(X_1 \vee X_2) \vee \vee \overline{x}_1x_2(X_1 \vee X_2) \vee x_1\overline{x}_2(X_1 \vee X_2) \vee x_1x_2(X_1 \vee X_2) = (X_1 \vee X_2).$$

Тут вхідні змінні x_i з'єднуються між собою знаком кон'юнкції, а похідні – змінні транспортування списків вхідних несправностей, позначені символом X_i, об'єднуються між собою знаком диз'юнкції, відповідно до стану координат стовпців-похідних.

Таким чином, запропонована технологія моделювання несправностей, що заснована на використанні векторних кубітних форм задання функціональностей і похідних, не має аналогів за доступністю розуміння, простотою реалізації та швидкодією. На рис. 6 зображено процесор кубітного моделювання цифрових пристроїв, що включає структури: справного інтерпретативного моделювання,дедуктивного аналізу несправностей, призначеного для оцінки якості тесту і побудови таблиці несправностей, а також модулів тестування і діагностування дефектів на стадіях проектування і експлуатації. Основна відмінність від існуючих рішень полягає у використанні Q-покриття, представленого у формі вектора станів функціональності, що дає можливість істотно підвищити швидкодію моделювання за рахунок виконання паралельних реєстрових операцій.

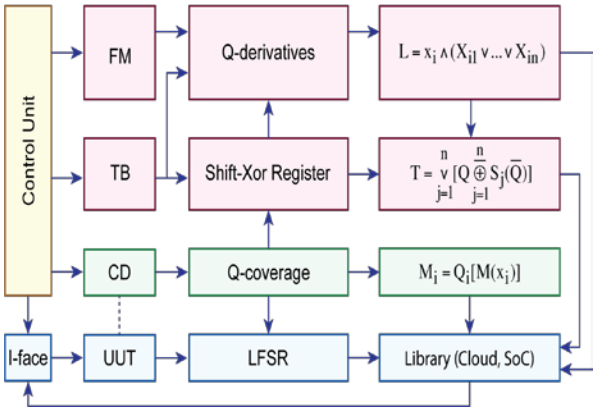

Рис. 6. Процесор кубітного моделювання цифрових пристроїв

Тут: Control Unit – пристрій управління симулятором, який синхронізує роботу блоків справного моделювання та структурних компонентів дедуктивного аналізу несправностей; FM – Fault Matrix, матриця вхідних несправностей розглянутої функціональності цифрового пристрою; TB – Test Bench, упорядкована сукупність вхідних перевіряючих послідовностей, де поточний вхідний набір ідентифікується як x_i; CD – Circuit Description – схемний опис цифрового пристрою, де функціональні елементи представлені кубітними покриттями Q-coverage. Обробка останніх здійснюється блоком справного моделювання $M_i = Q_i[M(x_i)]$, який реалізує адресні транзакції між кубітним покриттям і вектором моделювання M. Результати справного моделювання вхідних наборів формують матрицю GM (Good simulation Matrix), яка записується в Library. Блок Shift-Xor Register формує матрицю похідних в кубітній формі (Q-derivatives), застосовуючи реєстрові операції зсуву і xor. L-блок формування вихідного списку несправностей використовує формулу аналізу вхідного набору і рядки матриці похідних

$$L = x_i \wedge (X_{i1} \vee \ldots \vee X_{in}).$$

Результати дедуктивного аналізу формують списки несправностей, відповідних вхідних наборів, які об'єднуються в DM - Fault Detected Matrix і заносяться до Library. Крім того, T-модуль формує Q-тест і оцінює його якість в матриці одиночних константних несправностей зовнішніх входів і виходів функціональних елементів, які заносяться в Library. Тести для функціональностей разом з матрицями несправностей формують бібліотеку Library = {Signature, Q-coverage, Q-test, Quality, DM, GM}, яка може багаторазово використовуватися як хмарний або вбудований в SoC сервіс для тестування і / або діагностування функціональностей UUT (Unit Under Test) на основі використання інтерфейсу I-face, що підтримує стандарти IEEE 1500 SECT, IP (Internet Protocol). Пошук тестових сервісів у бібліотеці здійснюється за кубітним вектором, попередньо згорнутим в 16-розрядний двійковий код - сигнатуру (Sign), на основі регістра зсуву з лінійними зворотними зв'язками (LFSR. Це дає можливість структурувати бібліотеку для швидкого отримання інформації та проведення тестування в режимі online.

Структура взаємодіючих компонентів хмарного сервісу QuaSim представлена на рис. 7. Квантове або кубітне уявлення моделі цифрового пристрою разом з інтерпретативним симулятором складають ядро системи, інтегрованої у великі дані Інтернету.

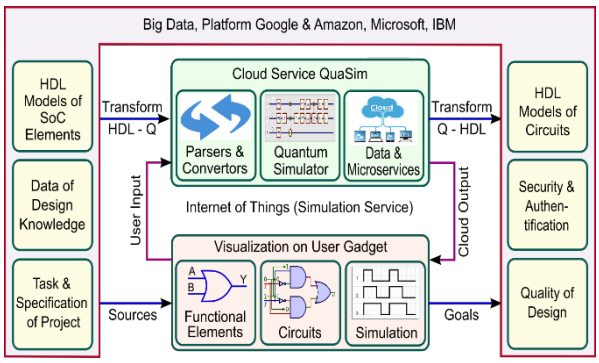

Рис. 7. Хмарний сервіс тестування і моделювання

Це дає можливість використовувати як вихідні дані відкриті специфікації і тестбенчі, описані на мовах VHDL, Verilog. Занурення QuaSim сервісу в інтернет-простір передбачає вивантаження результатів його роботи, пов'язаної з аналізом і синтезом навчальних або ринково орієнтованих проектів у сервіси зберігання даних на платформах Google, Amazon, Microsoft, IBM, Facebook. Демонстрація хмарного інтерфейсу для тестування і верифікації цифрових схем представлена на рис. 8.

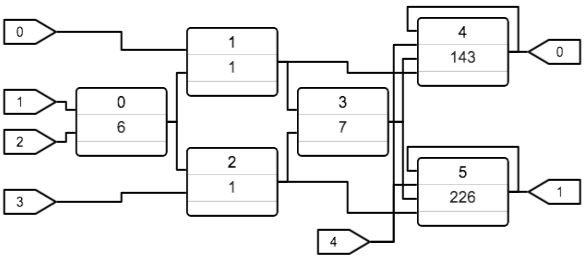

Рис. 8. Скріншот структури з тригерами

Природно, що інтеграція хмарного сервісу з кіберпростором передбачає наявність парсер-мікросервісів для перетворення специфікацій з мов опису апаратури у внутрішню мову QuaSim, а також має існувати і зворотне перетворення даних з кубітного уявлення в стандарти HDL-мов. Парсеризація забезпечує можливість використання відкритих в інтернеті проектів для їх вивчення і порівняння в системі моделювання Q-sim, а також робить доступними внутрішні проектні

рішення QuaSim для всіх бажаючих на ринку освітніх сервісів.

5. Висновки

1) Розроблено метод і секвенсор синтезу тестів для функціональних логічних компонентів, який характеризується паралельним виконанням реєстрових логічних операцій (shift, or, not, nxor) над кубітним вектором і його похідними, що дає можливість істотно зменшити час генерування вхідних наборів і тестування пристрою в режимі embedded online.

2) Розроблено метод обчислення похідних для генерації тестів функціональних компонентів, який характеризується паралельним виконанням реєстрових логічних операцій (shift, or, not, nxor) над кубітним вектором, що дозволяє істотно зменшити час генерування вхідних наборів і тестування пристрою за рахунок апаратної надмірності.

3) Розроблено метод моделювання справної поведінки і несправностей для функціональних компонентів, який характеризується паралельним виконанням реєстрових логічних операцій (shift, or, not, nxor) над кубітним вектором і його похідними, що дає змогу істотно зменшити час верифікації і тестування цифрового пристрою в режимі embedded online.

4) Запропоновано процесор кубітного моделювання цифрових пристроїв імплементувати в SoC або Cloud Service для аналізу справної поведінки і несправностей на основі використання кубітних покриттів функціональних елементів, який відрізняється від відомих реалізацій застосуванням мінімального набору реєстрових логічних операцій і високою швидкодією.

5) Практична значущість досліджень полягає в можливості хмарної реалізації швидкодіючого методу синтезу тестів і моделювання несправностей для функціональних логічних компонентів на основі паралельного виконання реєстрових логічних операцій (shift, or, not, nxor) над кубітним вектором і його похідними, що дає можливість генерувати вхідні набори і оцінювати їх якість в режимі online. Крім того, хмарний мікросервіс синтезу тестів і моделювання дефектів для функціональних логічних компонентів може бути затребуваний для навчальних і наукових цілей в процесах синтезу та аналізу цифрових архітектур.

6) Запропонований метод синтезу тестів для функціональностей на основі кубітного покриття може бути використаний як вбудований BIST-компонент для сервісного обслуговування SoC на основі стандарту граничного сканування IEEE 1500 SECT або як хмарний online сервіс тестування апаратних модулів за допомогою IP-протоколу.

7) Подальші дослідження в даній області будуть спрямовані на створення програмно-апаратних генераторів тестів, симуляторів несправностей, справної поведінки, алгоритмів діагностування та бібліотечних рішень, вбудованих в інфраструктуру кристалів і / або хмарні сервіси, що використовують кубітний опис функціональності логічного компонента.

Література: 1. *Рябцев В.Г., Муамар Д.Н.* Метод и средства визуализации алгоритмов тестов диагностирования запоминающих устройств // Электронное моделирование. 2010. Том 32. № 3. С. 43-52. **2.***Zorian Y.,Shoukourian S.* Test solutions for nanoscale Systems-on-Chip: Algorithms, methods and test infrastructure // Ninth International Conference on Computer Science and Information Technologies Revised Selected Papers, Yerevan, 2013. P. 1-3. **3.** *Tshagharyan G., Harutyunyan G., Shoukourian S. and Zorian Y.*Overview study on fault modeling and test methodology development for FinFET-based memories // 2015 IEEE East-West Design & Test Symposium (EWDTS), Batumi, 2015. P. 1-4. **4.** *Проектирование* и тестирование цифровых систем на кристаллах / В. И. Хаханов и др. Харьков: ХНУРЭ, 2009. 484 с. **5.** *Abramovici M., Breuer M.A. and Friedman A.D.* Digital systems testing and testable design // ComputerSciencePress. 1998. 652 p. **6.** *Кубитные* структуры данных вычислительных устройств / В. И. Хаханов, Ваджеб Гариби, Е. И. Литвинова, А. С. Шкиль // Электронное моделирование. 2015. Т. 37, № 1. С. 76–99. **7.** *Кубитные* технологии анализа и диагностирования цифровых устройств / В. И. Хаханов, Тамер Бани Амер, С. В. Чумаченко, Е. И. Литвинова // Электронное моделирование. 2015. Т. 37, № 3. С. 17–40. **8.** *Автоматизированное* проектирование цифровых устройств / С.С.Бадулин, Ю.М.Барнаулов и др. / Под ред. С.С. Бадулина.М.: Радиоисвязь, 1981. 240 с. **9.** *Michael A. Nielsen & Isaac L.* Chuang. Quantum Computation and Quantum Information. Cambridge University Press. 2010. 676p. **10.***Mikio Nfrfhara*. Quantum Computing. An Overview. Higashi-Osaka: Kinki University, 2010. 53p. **11.** *Курош А.Г.* Курс высшей алгебры. М.: Наука. 1968. 426 с. **12.***Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И.* Структура логического ассоциативного мультипроцессора // Автоматика и телемеханика. 2012. № 10. С. 71-92. **13.** *Molnar L. and Gontean A.*Fault simulation methodes, 2016 // 12th IEEE International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2016. P. 194-197. **14.** *Hadjitheophanous S., Neophytou S. N., Michael M. K.* Scalable parallel fault simulation for shared-memory multiprocessor systems // 2016 IEEE 34th VLSI Test Symposium (VTS), Las Vegas, NV, 2016. P. 1-6. **15.** *Pomeranz Irith, Reddy Sudhakar M.* Aliasing Computation Using Fault Simulation with Fault Dropping // IEEE Trans. on Computers. 1995. P. 139-144. **16.***Ubar R., Kõusaar J., Gorev M. and Devadze S.,* Combinational fault simulation in sequential circuits // 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, 2015. P. 2876-2879. **17.** *Gorev M., Ubar R. and Devadze S.*, Fault simulation with parallel exact critical path tracing in multiple core environment // 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, 2015. P. 1180-1185. **18.** *Pomeranz*

*I.*Fault simulation with test switching for static test compaction // 2014 IEEE 32nd VLSI Test Symposium (VTS), Napa, CA, 2014. P. 1-6. **19.** *Mirkhani S. and Abraham J. A.* EAGLE: A regression model for fault coverage estimation using a simulation based metric // 2014 International Test Conference, Seattle, WA, 2014. P. 1-10. **20.** *Hahanov I., Gharibi W., Iemelianov I., Tamer Bani Amer.* QuaSim – Cloud Service for Digital Circuits Simulation // IEEE 2016 East-West Design & Test Symposium. Yerevan, Armenia. P. 363-370. **21.** *Литвинова Е.И., Хаханов И.В.*Квантовый компьютинг для проектирования цифровых систем// Радиоэлектроника и информатика. 2015. № 4. С. 42-45.

Transliterated bibliography:

1. *Ryabcev V.G., Muamar D.N.* Metod i sredstva vizualizacii algoritmov testov diagnostirovaniya zapominayushchih ustrojstv // Elektronnoe modelirovanie 2010. Tom 32. № 3. S. 43-52.

2.*Zorian Y.,Shoukourian S.* Test solutions for nanoscale Systems-on-Chip: Algorithms, methods and test infrastructure // Ninth International Conference on Computer Science and Information Technologies Revised Selected Papers, Yerevan, 2013, pp. 1-3. doi: 10.1109/CSITechnol.2013.6710371.

3.*Tshagharyan G., Harutyunyan G., Shoukourian S. and Zorian Y.*Overview study on fault modeling and test methodology development for FinFET-based memories // 2015 IEEE East-West Design & Test Symposium (EWDTS), Batumi, 2015. P. 1-4. doi: 10.1109/EWDTS.2015.7493149

4.*Proektirovanie* i testirovanie cifrovyh sistem na kristallah / V. I. Hahanov i dr. Har'kov: HNURE, 2009. 484 s.

5.*Abramovici M., Breuer M.A. and Friedman A.D.* Digital systems testing and testable design // ComputerScience-Press. 1998. 652 p.

6.*Kubitnye* struktury dannyh vychislitel'nyh ustrojstv / V. I. Hahanov, Vadzheb Garibi, E. I. Lit-vinova, A. S. Shkil' // Elektronnoe modelirovanie. 2015. T. 37, № 1. S. 76–99.

7.Kubitnye tekhnologii analiza i diagnostirovaniya cifrovyh ustrojstv / V.I. Hahanov, Tamer Bani Amer, S.V. Chumachenko, E.I. Litvinova // Elektronnoe modelirovanie. 2015. T. 37, № 3. S. 17–40.

8.*Avtomatizirovannoe* proektirovanie cifrovyh ustrojstv / S.S.Badulin, Yu.M.Barnaulov i dr. / Pod red. S.S. Badulina.M.: Radio i svyaz', 1981. 240 s.

9.*MichaelA. Nielsen&IsaacL.*Chuang. Quantum Computation and Quantum Information. Cambridge University Press. 2010. 676p.

10.*Mikio Nfrfhara.* Quantum Computing. An Overview. Higashi-Osaka: Kinki University, 2010. 53p.

11. *Kurosh A.G.* Kurs vysshej algebry. Izd-vo: Moskva: Nauka. 1968. 426 s.

12. *Bondarenko M.F., Hahanov V.I., Litvinova E.I.* Struktura logicheskogo associativnogo mul'tipro-cessora// Avtomatika i telemekhanika. 2012. № 10. S. 71-92.

13.*Molnar L. and Gontean A.* Fault simulation methodes," 2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2016.P. 194-197.

14.*Hadjitheophanous S., Neophytou S. N.,Michael M. K.* Scalable parallel fault simulation for shared-memory multiprocessor systems // 2016 IEEE 34th VLSI Test Symposium (VTS), Las Vegas, NV, 2016.P. 1-6.

15.*Pomeranz Irith, Reddy Sudhakar M.* Aliasing Computation Using Fault Simulation with Fault Dropping // IEEE Trans. on Computers. 1995. P. 139-144.

16.*Ubar R., Kõusaar J., Gorev M. and Devadze S.,* Combinational fault simulation in sequential circuits // 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, 2015. P. 2876-2879.

17.*Gorev M., Ubar R. and Devadze S.,* Fault simulation with parallel exact critical path tracing in multiple core environment // 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, 2015. P. 1180-1185.

18.*Pomeranz I.*Fault simulation with test switching for static test compaction // 2014 IEEE 32nd VLSI Test Symposium (VTS), Napa, CA, 2014. P. 1-6.

19.*Mirkhani S. and Abraham J. A.* EAGLE: A regression model for fault coverage estimation using a simulation based metric // 2014 International Test Conference, Seattle, WA, 2014. P. 1-10.

20.*Hahanov I., Gharibi W., Iemelianov I., Tamer Bani Amer.* QuaSim – Cloud Service for Digital Circuits Simulation // IEEE 2016 East-West Design & Test Symposium. Yerevan, Armenia. P. 363-370.

21.*Litvinova E.I., Hahanov I.V.* Kvantovyj komp'yuting dlya proektirovaniya cifrovyh sistem // Radioelektronika i informatika. 2015. № 4. S. 42-45.

Надійшла до редколегії 11.07.2017

Рецензент: д-р техн. наук, проф. Кривуля Г.Ф.

Литвинова Євгенія Іванівна,д-р техн. наук, проф. кафедри АПОТ ХНУРЕ. Наукові інтереси: проектування та тестування цифрових систем. Хобі: музика. Адреса: Україна, 61166, Харків, пр. Науки, 14, e-mail: litvinova_eugenia@icloud.com.

Ємельянов Ігор Валерійович, науковий співробітник кафедри АПОТ ХНУРЕ. Наукові інтереси: проектування та тестування цифрових систем. Хобі: мандри. Адреса: Україна, 61166, Харків, пр. Науки, 14, e-mail: iyemelyanov@itdelight.com.

Хаханов Іван Володимирович, студент кафедри АПОТ факультету КІУ ХНУРЕ. Науковіінтереси: технічна діагностика цифрових систем, програмуваня. Хобі: горні лижі, англійська мова. Адреса: Україна, 61166, Харьков, пр. Науки, 14, тел. +3805770-21-326,e-mail: ivanhahanov@icloud.com.

Litvinova Evgenia Ivanovna, Dr. of Tech. Sciences, prof., Design Automation Department, NURE. Scientific interests: design and testing of digital systems. Hobbies: music. Address: Ukraine, 61166, Kharkov, Nauki Ave, 14, e-mail: litvinova_eugenia@icloud.com.

Yemelyanov Igor Valerievich, researcher, Design Automation Department, NURE. Scientific interests: design and testing of digital systems. Hobbies: traveling. Address: Ukraine, 61166, Kharkov, Nauki Ave, 14, e-mail: iyemelyanov@itdelight.com.

Hahanov Ivan Vladimirovich, student, Design Automation Department, NURE. Scientific interests: technical diagnostics of digital systems, programming. Hobby: mountain skiing, English. Address: Ukraine, 61166, Kharkov, Nauki Ave., 14, ph. + 3805770-21-326, e-mail: ivanhahanov@icloud.com.