УДК 681.326:519.713

В.И. Хаханов, д-р техн. наук, **Тамер Бани Амер**, аспирант,
С.В. Чумаченко, д-р техн. наук, **Е.И. Литвинова**, д-р. техн. наук
Харьковский национальный университет радиоэлектроники
(Украина, 61166, Харьков, пр. Ленина, 14,
тел. (057) 7021326, e-mail: hahanov@kture.kharkov.ua)

Кубитные технологии анализа и диагностирования цифровых устройств

Предложены технология и примеры реализации кубитных моделей, методов и алгоритмов повышения быстродействия существующих программных и аппаратных средств анализа цифровых вычислительных устройств в результате увеличения размерности структур данных и памяти для одновременного хранения обрабатываемых состояний. Представлены результаты исследований моделей и методов диагностирования цифровых систем, моделирования восстановления работоспособности отказавших примитивов.

Запропоновано теорію та приклади реалізації кубітних моделей, методів і алгоритмів для підвищення швидкодії існуючих програмних і апаратних засобів аналізу та синтезу цифрових обчислювальних пристроїв в результаті збільшення розмірності структур даних і пам'яті для одночасного збереження оброблюваних станів. Наведено результати досліджень моделей і методів діагностування цифрових систем, моделювання відновлення працездатності дефектних примітивів.

Ключевые слова: цифровые кубитные структуры, моделирование, диагностирование и ремонт цифровых систем.

Эволюция киберпространства планеты условно делится на следующие периоды: 1) 1980—1990 гг. — формирование парка персональных компьютеров; 2) 1990—2000 гг. — внедрение Интернет-технологий в производственные процессы и быт человека; 3) 2000—2010 гг. — повышение качества жизни в результате внедрения мобильных устройств и облачных сервисов; 4) 2010—2015 гг. — создание цифровой инфраструктуры мониторинга, управления и взаимодействия стационарных и движущихся объектов (воздушный, морской, наземный транспорт и роботы), создание глобальной цифровой инфраструктуры киберпространства, где все процессы и явления идентифицируются во времени и пространстве и становятся интеллектуальными.

В связи с развитием параллельных вычислений с неупорядоченными данными в последние годы для создания облачных Интернет-сервисов все

чаще применяются кубитные структуры (КС), являющиеся удачной альтернативой существующим затратным по времени классическим моделям последовательной обработки теоретико-множественных структур в результате существенного расширения памяти [1]. В настоящее время это вполне допустимо, поскольку рынок нано-электронных технологий предоставляет разработчикам цифровых систем до десяти миллиардов вентилей на кристалле размерностью 2 × 2 см при толщине пластины 5 микрон. При этом современные технологии допускают создание пакета, или «сэндвича», содержащего до семи кристаллов. Практически беспроводное соединение таких пластин основано на технологической возможности сверления до 10 тысяч сквозных отверстий на одном квадратном сантиметре.

С появлением трехмерных FinFETs транзисторов и на их основе — 3D-технологий возникают новые возможности для создания более быстродействующих объемных цифровых систем (ЦС) при сокращении задержек параллельных вычислительных устройств [2—7]. Поэтому следует использовать модели и методы, предназначенные для создания быстродействующих средств параллельного решения практических задач. Учитывая дискретность и многозначность алфавитов описания информационных процессов, свойство параллелизма, заложенное в квантовых вычислениях, можно использовать при создании эффективных и интеллектуальных вычислителей для киберпространства, облачных структур и сервисов Интернета, а также для повышения надежности цифровых устройств (ЦУ) тестирования и моделирования дискретных систем на кристаллах.

Не рассматривая физические основы квантовой механики, касающиеся недетерминированного взаимодействия атомных частиц, будем использовать понятие КС как векторной формы совместного или одновременного задания булеана состояний в конечной и дискретной областях киберпространства, ориентированного на параллелизм и суперпозицию обработки предлагаемых кубитных моделей и методов.

Квантовые эмуляторы в классических компьютерах достаточно эффективно применяются при решении оптимизационных задач, связанных с полным перебором вариантов решений на основе теории множеств [1, 8]. Множество элементов в компьютере всегда является упорядоченным, поскольку каждый бит, байт или другой компонент имеет свой адрес. Поэтому все теоретико-множественные операции сводятся к полному перебору адресов примитивных элементов. Адресный порядок структур данных подходит для задач, где компоненты моделей можно строго ранжировать, что дает возможность выполнять их анализ за одну итерацию. Для неупорядоченных структур (например, множество всех подмножеств) использование классической модели памяти приводит к увеличению вре-

мени анализа ассоциации равных по рангу примитивов или к неэффективности обработки ассоциативных групп.

Вместо строгого порядка можно использовать для неупорядоченных данных процессор, где элементарной ячейкой является образ или шаблон универсума из n примитивов, который генерирует $Q = 2^n$ всех возможных состояний такой ячейки в виде булеана или множества всех подмножеств. При создании такой ячейки используется унитарное позиционное кодирование состояний примитивов, которое с помощью суперпозиции последних образует универсум примитивов. Такое кодирование состояний формирует в пределе булеан или множество всех подмножеств [8, 9].

n-Кубит есть векторная форма унитарного кодирования универсума из n примитивов для задания булеана состояний 2^{2^n} с помощью 2^n двоичных переменных. Например, если $n = 2$, то 2-кубит задает 16 состояний с помощью четырех переменных. Если $n = 1$, то кубит задает четыре состояния на универсуме из двух примитивов с помощью двух двоичных переменных (00, 01, 10, 11) [1]. При этом допускается суперпозиция (одновременное существование) в векторе 2^n состояний. Кубит (n-кубит) дает возможность использовать логические операции вместо теоретико-множественных для существенного ускорения процессов синтеза и анализа дискретных систем. Далее кубит отождествляется с n-кубитом или вектором, если это не мешает пониманию излагаемого материала. Поскольку квантовые вычисления связаны с анализом КС, будем использовать определение «квантовый» для идентификации технологий, в которых используются три свойства квантовой механики: параллелизм обработки, суперпозиция состояний, перепутывание.

Свойство перепутывание (entanglement) в классической квантовой физике поясняется так: «две разделенные частицы одного кванта знают друг о друге все» — парадокс переплетения [1]. Интерпретация данного свойства в многозначной алгебре Кантора — «два взаимно-дополняющих до универсума символа, знают друг о друге все независимо от расстояния». Любое изменение состояния одного из них мгновенно приводит к изменению другого. Они связаны между собой отношением дополнения и являются идеальными «приемо-передатчиками» по операции дополнения. Это относится также к кубитам, которые представляют собой векторную форму унитарного кодирования и взаимодействия символов универсума. Можно обобщить взаимную «дополнительность» двух частиц свойства перепутывания на большее число взаимосвязанных компонентов. Два из трех символов, образующих транзитивное замыкание по симметрической разности $A1 \Delta A2 \Delta A3 = \varnothing$, знают о третьем все. В общем случае $n-1$ из n символов, образующих циклическое замыкание по симметрической раз-

ности A1 Δ A2 Δ…Δ An = ∅, определяют состояние символа с номером n, т.е. циклическое, по симметрической разности, взаимодействие объектов позволяет определить состояние любого из них, если известны значения $n-1$ компонента.

Кубитный метод диагностирования ЦС представляет собой метод диагностирования функциональных нарушений и константных неисправностей в программных или аппаратных блоках, в которых используются кубитные, или многозначные, структуры данных для задания диагностической информации [10]. Метод позволяет существенно уменьшить вычислительную сложность процессов моделирования и диагностирования с помощью введения параллельных логических операций над матричными данными. Предлагаемый кубитный метод моделирования ЦУ с восстановлением работоспособности компонентов ЦС в режиме онлайн имеет существенно более высокое быстродействие в результате адресной реализации процедуры обработки функциональных примитивов, заданных кубитными векторами состояний выходов.

Модель объекта диагностирования представлена в форме графа ЦС, имеющей функциональные элементы, соединенные линиями связей, среди которых есть ассерции — точки наблюдения, или мониторинга, необходимые для верификации, тестирования и диагностирования неисправностей [2]. Диагностическая информация представлена следующими компонентами:

тест диагностирования неисправностей заданного класса (рассматриваются одиночные константные дефекты $\{\equiv 0, \equiv 1\}$ линий схемы);

таблица неисправностей [6], строки которой заданы векторами проверяемых на каждом тестовом наборе дефектов, привязанных к линиям схемы;

матрица достижимостей, определяющая достижимость каждой ассерционной точки со стороны множества предшествующих линий [8];

матрица состояния ассерционного механизма, или матрица экспериментальной проверки, задающая состояние каждой ассерции на тестовых наборах путем сравнения эталонной реакции в данной точке с реальным сигналом в процессе выполнения диагностического эксперимента [2, 7].

Базовая модель диагностирования ЦУ, дискретного процесса или явления представлена компонентами, создающими четыре измерения в пространстве признаков:

$$D_b = < S, A, F, T >, \quad V_b = (|S| \times |A| \times |F| \times |T|), \quad S^* = f(S, A, T),$$
$$D = \{< S, A >, < F, T >\}; \quad V = (|S| \times |A|) + (|F| \times |T|), \quad A^* = g(T, A),$$
$$V_b >> V; \quad F^* = h(S, A, F, T).$$

При этом в модели объем диагностической информации V формируется декартовым произведением (мощностей) четырех компонентов: S — структура объекта; A — механизм ассерций или мониторинга; F — совокупность неисправностей или модулей, подверженных функциональным нарушениям; T — тестовые наборы или сегменты для диагностирования неисправностей или совокупности упомянутых модулей.

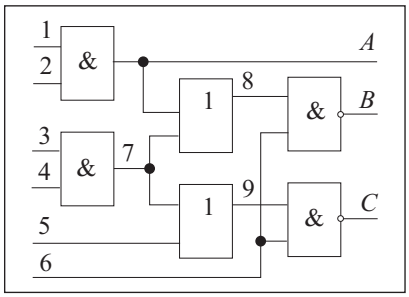

Рис. 1. Фрагмент цифровой схемы

Существенно уменьшить объем диагностической информации можно понижением размерности пространства признаков посредством разделения базовой модели на два непересекающихся подмножества: $< S, A >$ и $< F, T >$. В этом случае оценка объема диагностической информации становится не мультипликативной, а аддитивной по отношению к мощности полученных в результате разбиения подмножеств без какого-либо уменьшения глубины диагностирования. Первый компонент модели диагностирования представлен матрицей достижимостей, которая позволяет минимизировать возможные дефекты (маску) при сравнении истинных и реальных результатов моделирования выходных сигналов на каждом тестовом наборе или сегменте. Число строк такой матрицы равно числу наблюдаемых выходов, или ассерций.

С помощью метода диагностирования создается двоичная матрица структурной активизации неисправностей, которая представляет собой маску для существенного уменьшения множества предполагаемых дефектов при совместном анализе таблицы неисправностей. При этом символы одиночных константных дефектов $\{0,1,X,\varnothing\}$, $X=\{0,1\}$ в ячейках таблицы неисправностей [6] кодируются соответствующими состояниями кубита (10, 01, 11, 00) многозначного алфавита Кантора $A^k=\{0,1,X,\varnothing\}$, что дает возможность исключить из вычислительных процессов теоретико-множественные процедуры, заменив их векторными логическими операциями.

На рис. 1 представлен фрагмент цифровой схемы, с помощью которого рассмотрим суть предлагаемого метода. Ассерционные точки A, B, C являются точками наблюдения за состоянием всех линий схемы в процессе тестирования (выполнения диагностического эксперимента) посредством подачи пяти тестовых воздействий, заданных в таблице неисправностей F (T) (табл. 1), координаты которой заданы проверяемыми на тест-векторах неисправностями 0 и 1, а также координатами: $\varnothing$ (.) — отсутствие проверяемых дефектов и X — проверка на линии константы 0 и 1 одновременно. В правой части табл. 1 представлена матрица состояний

ассерционного механизма в виде результатов сравнения эталонной и реальной реакций ЦУ на тестовые наборы, где 1 означает несовпадение, 0 — совпадение упомянутых реакций.

В табл. 1 не учитывается структура схемы для повышения глубины диагностирования на основе вычисления реальной матрицы состояний ассерционного механизма, которая совместно с матрицей достижимостей создает структурную маску, минимизирующую множество предполагаемых дефектов. Для фрагмента цифровой схемы, представленной на рис. 1, матрица достижимостей имеет следующий вид:
Здесь выходы-ассерции A, B, C являются мониторами технического состояния объекта диагностирования. Каждый из них может иметь два значения, $A_{ij} = \{0,1\}$, которые определяют матрицу экспериментальной проверки $A = |A_{ij}|$ путем сравнения эталонных $T = |T_{ij}|$ и реальных $U = |U_{ij}|$ состояний наблюдаемых или выходных линий, $A_{ij} = T_{ij} \oplus U_{ij}$, формирующих маску возможных дефектов с помощью следующего выражения:

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

$S = \|S_{ij}\|$	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

Каждый тест-вектор (тест-сегмент) активизирует собственную структуру возможных дефектов, функционально зависящую от маски, ассерций (состояния наблюдаемых выходов) и тестовых наборов: $S = f(S, A, T_i)$. Если предположить, что в матрице $S = |S_{ij}|$ состояния ассерционных выходов на первом тест-векторе составляют $A_{1A} = 0$, $A_{1B} = 1$, $A_{1C} = 1$, где значение 1 идентифицирует проявление дефекта в устройстве, то маска возможных

Таблица 1

$F(T)$	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

дефектов, согласно функционалу $S_1 = S(T_1) = (\underset{A_{1j}=1}{\vee} S_{1j}) \wedge (\overline{\underset{A_{1j}=0}{\vee} S_{1j}})$, будет иметь следующий вид:

$$S_1 = S(T_1) = (S_2 \vee S_3) \wedge (\overline{S_1 1}) =$$

$$= (111101110010 \vee 001111101001) \wedge (\overline{110000000100}) =$$

$$= (111111111011) \wedge (001111111011) = (001111111011).$$

Полученная маска накладывается на первую строку табл. 1, что определяет множество предполагаемых дефектов $F_i = T_i \wedge S|_{i=1} \rightarrow F_1 = T_1 \wedge S_1$, формирующих ассерционную выходную реакцию $A_{1(A,B,C)} = (011)$ устройства на первый тест-вектор:

F	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

В соответствии с предложенной процедурой получения маски одной строки выполняется построение матрицы структурной активизации неисправностей $S(T)$ с помощью таблицы экспериментальной проверки $A = |A_{ij}|$, задающей состояния ассерционного механизма в процессе выполнения тестирования $S(T) = S \otimes A$:

$S = \lvert S_{ij} \rvert$	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

$\overset{\otimes}{\rightarrow}$

$A = \lvert A_{ij} \rvert$	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
$T1$	0	0	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

=

С целью формирования структур данных, удобных для компьютерной обработки, необходимо перевести символы табл. 1 в двухразрядные коды в

соответствии с правилами ▷-кодирования: $\triangleright = \{0 = 10; 1 = 01; X = 11; \varnothing = 00\}$. Применение этих правил дает следующий 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

$\xrightarrow{\triangleright}$

$\xrightarrow{\triangleright}$

$F(T)$	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

После получения структурной матрицы $S(T)$, предназначенной маскировать реальные дефекты в таблице неисправностей и ее кодированной формы, необходимо выполнить #-суперпозицию двух матриц: $F(T) = S(T) \# F(T)$. Для этого необходимо выполнить #-операции над одноименными координатами: $F_{ij} = \overline{F}_j \leftarrow (F_j = 00) \vee (S_{ij} = 0)$, что означает модификацию кодов координат, указанных в табл. 1, при выполнении заданных условий. В противном случае данная операция сводится к инверсии ячеек матрицы кодов неисправностей, маскируемых нулевыми сигналами структурной матрицы активизации, а также всех нулевых кодов табл. 1.

Таблица истинности данной #-операции в символьном и кодированном виде,

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

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

,

скорректирована относительно инверсии состояния 00 в 11 при единичном значении сигнала активизации неисправности, поскольку такой код (00) означает наличие в схеме на линии пустого множества проверяемых дефектов, что невозможно. Кроме того, код 00 блокирует все вычисления конъюнкции по столбцу, превращая результат в 00. Инверсия кода дает возможность не маскировать знаки при логическом умножении действительно присутствующих дефектов. При этом предполагается, что тест-вектором невозможно проверить на одной линии схемы дефекты разных знаков.

Выполнение процедуры суперпозиции между структурной матрицей и кодированной табл. 1 по правилу $F(T) = S(T) \# F(T)$ дает следующий результат:

$S(T)$	1	2	3	4	5	6	7	8	9	A	B	C
$T1$	0	0	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

$\xrightarrow{\#}$

$F(T)$	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

$\xrightarrow{\#}$ =

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

=

На заключительном этапе диагностирования выполняется единственная векторная операция логического умножения всех строк кодированной модифицированной табл. 1:

$$F(T) = (\underset{A_i=1}{\vee} F_i) \wedge (\overline{\underset{A_i=0}{\vee} F_i}) = (\underset{A_i=1}{\wedge} F_i) \wedge (\overline{\underset{A_i=0}{\vee} F_i}) = (\underset{A_i=1}{\wedge} F_i) \wedge (\underset{A_i=0}{\wedge} \overline{F_i}) = (\overset{n}{\underset{i=1}{\wedge}} F_i).$$

Это дает возможность точно определить все дефекты, присутствующие в объекте диагностирования, которые представлены в двух последних строках кодированной табл. 1: $F = \{3^1, 6^1, A^0\}$.

Теоретическое доказательство матричного диагностирования одиночных и кратных дефектов представим в виде двух теорем.

Теорема 1. Одиночные константные дефекты цифровой схемы, заданные кубитами на тестовых наборах многозначной таблицы неисправностей, определяются с помощью векторной and-операции, маскируемой по строкам матрицей (вектором) экспериментальной проверки $A=|A_{ij}|$ всех ассерционных точек:

$$F(T)=(\underset{A_i=1}{\vee} F_i)\wedge(\overline{\underset{A_i=0}{\vee} F_i})=(\underset{A_i=1}{\wedge} F_i)\wedge(\overline{\underset{A_i=0}{\vee} F_i})=(\underset{A_i=1}{\wedge} F_i)\wedge(\underset{A_i=0}{\wedge} \overline{F_i})=(\overset{n}{\underset{i=1}{\wedge}} F_i).$$

Данное выражение справедливо, так как:

1) второй сомножитель — отрицание дизъюнкции — есть конъюнкция отрицаний, что означает умножение кодов таблицы с их предварительным отрицанием;

2) первый сомножитель ориентирован на поиск непротиворечивых дефектов, поэтому вместо него используем $(\underset{A_i=1}{\wedge} F_i)$.

Действительно, на одной линии, или переменной, не могут присутствовать одновременно две противоположные по знаку проверяемые неисправности. Поэтому в базовой формуле дизъюнкция дефектов $(\underset{A_i=1}{\vee} F_i)$ в большей степени ориентирована на поиск кратных неисправностей, но не связанных с одной линией. Кратность противоречивых дефектов на одной линии, равно как и инверсия пустого множества неисправностей, теоретически создает условия беспрепятственного умножения других ячеек столбца для формирования на каждой линии результата в виде дефекта одного знака или пустого множества неисправностей.

Теорема 2. Кратные константные дефекты цифровой схемы, заданные кубитами на тестовых наборах многозначной таблицы неисправностей, определяются с помощью векторных or- and- операций, маскируемых по строкам вектором экспериментальной проверки $A(T)$ всех ассерционных точек:

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

Выражение является верным, так как:

1) второй сомножитель есть отрицание дизъюнкции или конъюнкция отрицаний, что означает умножение кодов таблицы с их предварительным отрицанием;

2) первый сомножитель ориентирован на поиск кратных дефектов в предположении, что на одной линии или переменной могут присутствовать одновременно две противоположные по знаку проверяемые неисправности.

Данная формула в большей степени ориентирована на поиск кратных дефектов в блоках ЦС, не связанных с одной линией. Кратность неисправ-

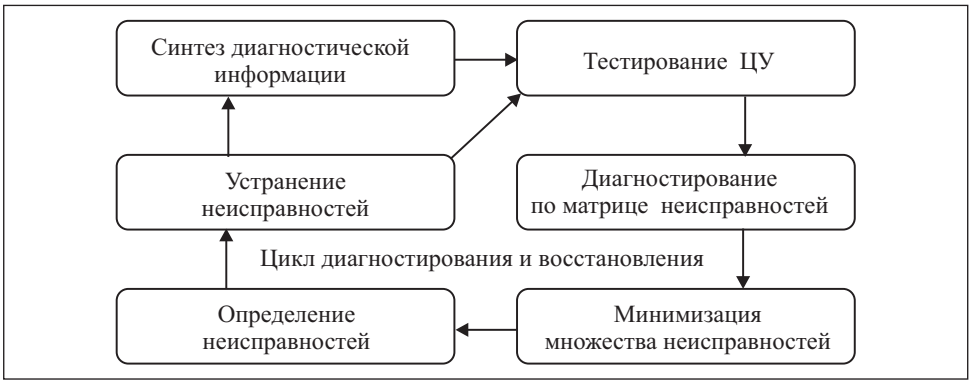

Рис. 2. Цикл диагностирования и ремонта логических блоков

ностей в ЦС теоретически создает условия для логического сложения других ячеек столбца для формирования результата в виде множества дефектов, формирующих заданный вектор экспериментальной проверки, из которых необходимо вычесть проверяемые на тесте неисправности, не оказывающие влияния на формирование некорректных реакций по выходам.

Интерес представляет поиск кратных дефектов на основе мультипроцессора Хассе [4, 5], ориентированного на решение задачи покрытия посредством полного перебора событий, обеспечивающих точное покрытие вектора экспериментальной проверки столбцами таблицы неисправностей:

$$F(T) = (\vee_i F_i) \oplus A = 0.$$

В данном случае решением является такое сочетание столбцов, участвующих в векторной операции логического сложения, которое в совокупности дает результат, равный вектору экспериментальной проверки. Поскольку операция времязатратная, для нее следует использовать мультипроцессор Хассе, ориентированный на взятие булеана в почти параллельном режиме.

На рис. 2 представлена модель процесса диагностирования ЦУ, которая содержит функциональные преобразователи, связанные с выполнением следующих шагов.

1. Генерирование исходной диагностической информации в виде теста диагностирования, таблицы неисправностей и матрицы достижимостей цифровой системы.

2. Тестирование реального устройства на основе использования промышленного симулятора для сравнения фактических реакций устройства с эталонными значениями по наблюдаемым линиям-ассерциям, что дает возможность сформировать матрицу выходных реакций или вектор экспериментальной проверки в двоичном алфавите.

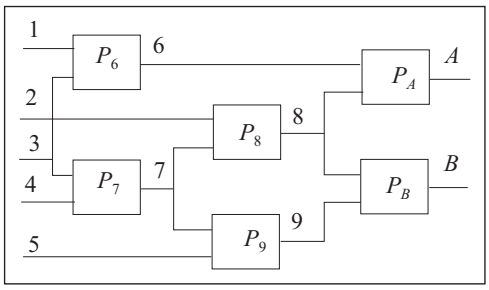

Рис. 3. Фрагмент цифровой схемы

3. Вычисление матрицы активности графовой структуры на каждом входном тестовом наборе, равной по размерности таблице неисправностей, с помощью матрицы экспериментальной проверки и матрицы достижимостей, что дает возможность существенно сократить область предполагаемых дефектов.

4. Модификация содержимого таблицы неисправностей посредством маскирования ее матрицей активности графовой структуры для определения только тех неисправностей, которые действительно формируют матрицу экспериментальной проверки в процессе диагностирования.

5. Выполнение процедуры логического умножения над строками таблицы неисправностей для получения вектора подозреваемых дефектов.

6. Восстановление работоспособности ЦУ с помощью переадресации неисправных логических компонентов на их аналоги из ремонтного запаса и повторение процесса тестового диагностирования.

Таким образом, новизна предложенного метода диагностирования дефектов заключается в использовании для получения диагноза единственной параллельной операции логического умножения, что в сочетании со структурным маскированием неисправностей дает возможность увеличить быстродействие и повысить глубину диагностирования.

Кубитное моделирование ЦС. Рассмотрим структуры данных, используемые для программной или аппаратной реализации исправного интерпретативного моделирования дискретных систем, описанных в форме кубитных векторов состояний выходов примитивов. Для описания цифровой схемы, представленной на рис. 3, традиционно используется структура взаимосвязанных элементов и кубические покрытия (таблицы истинности) логических элементов.

С помощью предлагаемого метода кубитного моделирования можно заменить таблицы истинности компонентов ЦУ векторами состояний выходов. Пусть функциональный примитив с номером P_6 имеет следующую таблицу истинности:

$P_6 =$

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

Данное покрытие логического элемента можно трансформировать унитарным кодированием входных векторов на основе двухтактного алфавита [4—7]. Символы и их коды, предназначенные для описания автоматных переменных, представляют собой булеан на универсуме из четырех примитивов, что соответствует формату вектора, содержащего два кубита:

$$B^*(Y) = \{Q = (1000),\ E = (0100),\ H = (0010),\ J = (0001),$$

$$O = \{Q, H\} = (1010),\ I = \{E, J\} = (0101),\ A = \{Q, E\} = (1100),$$

$$B = \{H, J\} = (0011),\ S = \{Q, J\} = (1001),\ P = \{E, H\} = (0110),$$

$$C = \{E, H, J\} = (1110),\ F = \{Q, H, J\} = (1011),\ L = \{Q, E, J\} = (1101),$$

$$V = \{Q, E, H\} = (1110),\ Y = \{Q, E, H, J\} = (1111),\ U = (0000)\}.$$

С помощью двухтактного алфавита любое покрытие функционального примитива кодированием входных наборов и последующего объединения символов можно представить двумя кубами или даже одним, учитывая, что кубы взаимно инверсны:

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

Два куба показывают не только все решения, но и инверсию сигналов на выходе, что интересно с позиции активизации всех логических путей в схемной структуре при синтезе тестов. Например, для изменения состояния выхода необходимо создать на входах пару следующих одно за другим условий, где в первом такте должны быть первые три вектора (адреса), а во втором — четвертый вектор, формируемый двумя входными переменными. Для моделирования исправного поведения достаточно иметь один куб (нулевой или единичный), так как второй всегда является дополнением к первому. Следовательно, ориентируясь, например, на единичный куб, формирующий на выходе единицу, можно исключить бит состояния выхода примитива, что уменьшит размерность куба или модели примитива до числа адресуемых состояний элемента, где адрес есть вектор, составленный из двоичных значений входных переменных, по которому определяется состояние выхода примитива.

Кубитное покрытие, или Q-покрытие, есть векторная интерпретативная форма задания функциональности, где значение координаты определяет состояние выхода функции, соответствующее двоичному входному слову, формирующему адрес ячейки. Q-покрытие одновыходового прими-

тива всегда представлено двумя взаимно инверсными кубами (векторами), размерность которых равна степени двойки от числа входных переменных, где единичное значение координаты определяет участие адреса рассматриваемого бита в формировании соответствующего (0, 1) состояния выхода примитива. Кубитные модели примитивов требуют создания новой теории моделирования, прямой и обратной импликации, синтеза тестов, моделирования неисправностей, поиска дефектов.

Рассмотрим основные процедуры исправного моделирования на основе манипулирования адресами, неявно представленными в координатах кубов Q-покрытия. Модель для анализа ЦС на основе кубитных структур данных может быть описана четырьмя компонентами:

$$F =< L, M, X, Q >,$$

$$L = (L_1, L_2, ..., L_j, ..., L_n),$$

$$M = (M_1, M_2, ..., M_j, ..., M_n),$$

$$X = (X_{n_x+1}, X_{n_x+2}, ..., X_{n_x+i}, ..., X_n),$$

$$Q = (Q_{n_x+1}, Q_{n_x+2}, ..., Q_{n_x+i}, ..., Q_n).$$

Здесь L — вектор идентификаторов эквипотенциальных линий схемы ЦС, который ввиду тривиальности может быть исключен из модели, но при этом необходимо знать число входных переменных ЦУ и общее число линий; M — вектор моделирования состояний всех линий схемы; X — упорядоченная совокупность векторов входных переменных каждого примитива схемы, привязанных к номерам выходов; Q — совокупность Q-покрытий примитивов, строго привязанных к номерам выходов и входным переменным примитивов; n — число линий в схеме; n_x — число входных переменных.

В качестве примера кубитного задания модели ЦУ $F =< L, M, X, Q >$ (см. рис. 3), рассмотрим вариант структурной таблицы описания схемы для анализа исправного поведения (табл. 2).

Таблица 2

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

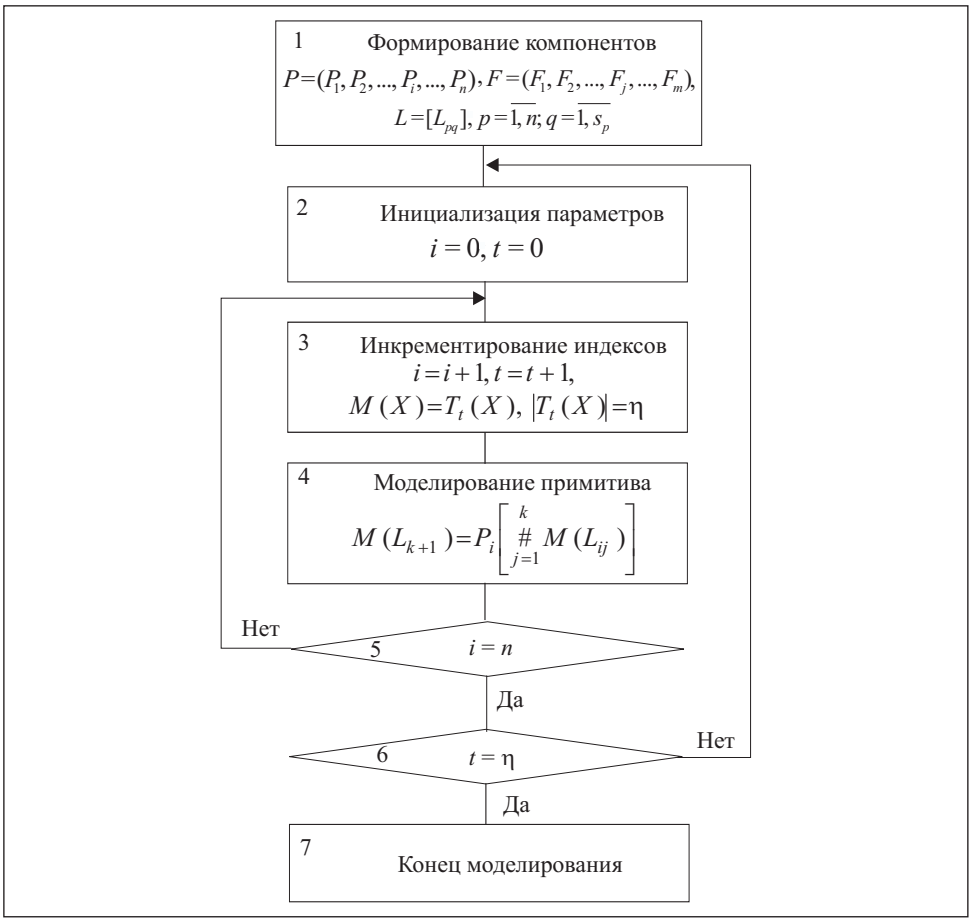

Рис. 4. Граф-схема алгоритма управления процессом моделирования

Метод кубитного моделирования исправного поведения сводится к определению значения выхода элемента по адресу, формируемому конкатенацией двоичных состояний входных переменных каждого примитива ЦС, $M(Y_i)=Q_i[M(X_{i1}* X_{i2}...* X_{ij}...* X_{ik_i})]$, где k_i — число входных линий в примитиве с номером i. Поскольку номера невходных линий вектора L однозначно идентифицируют по выходам обрабатываемые примитивы, формула моделирования может быть приведена к циклу определения состояний всех невходных переменных:

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

При этом процесс моделирования связан с конкатенированным формированием адреса бита в кубите функциональности, который определяет сос-

тояние примитива, или невходной линии цифровой структуры, начиная с номера $i = n_x + 1$. Если переменные создают не двоичный адрес, то в данном случае существует возможность формирования состояния выхода логического элемента в троичном алфавите символом *X*. Состояния выходов формируются идеально примитивной процедурой обработки кубита примитива $M_i = Q_i[M(X_i)]$ на основе простых итераций или итераций Зейделя [6, 8]. В последнем случае необходима препроцессорная процедура ранжирования линий и примитивов схемы, которая позволяет существенно уменьшить число проходов по элементам схемы для достижения сходимости, когда фиксируется равенство состояний всех линий в двух соседних итерациях. Кроме того, ранжирование примитивов по уровням формирования выходов дает возможность существенно повысить быстродействие моделирования в результате параллельной обработки функциональных элементов одного уровня. Например, для схемы, представленной на рис. 4, можно одновременно обрабатывать элементы с номерами 6, 7, затем — 8, 9 и далее — *A*, *B*. В первом случае, когда используются простые итерации, ранжирования не требуется, но платой за простоту алгоритма моделирования является существенно большее число итеративных проходов по примитивам схемы для достижения упомянутого критерия сходимости.

Вычислительная сложность предложенного *Q*-метода моделирования на основе кубитных функциональностей определяется процедурами формирования адреса — входного вектора, содержащего k_i переменных для каждого *i*-го примитива $[(r+w)\times k_i]$, считыванием бита из кубит-вектора по конкатенированному адресу и записью $(r+w)$ данного бита в вектор моделирования:

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

Время моделирования одного тест-вектора *Q*-методом при условии, что цифровая схема, составленная из 900 четырехвходовых примитивов, имеет параметры: $r = w = 5$ нс, $k_i = 4$, $n_x = 100$, $n = 1000$, составляет

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

Это означает, что быстродействие интерпретативного *Q*-метода моделирования позволяет для данной схемы за одну секунду обработать 22 222 входных набора. При этом ЦУ имеет существенное преимущество — сервисную функцию онлайн-восстановления работоспособности — в случае отказа примитива переадресации его на запасной элемент.

Для синтеза квазиоптимальных структур данных комбинационного устройства требуется выполнение следующих правил:

1. Наличие в ранжированной схеме ЦУ при моделировании способом Зейделя по возможности однотипных примитивов в каждом уровне (слое) срабатывания.

2. В каждом уровне желательно наличие одинакового числа примитивов. Поэтому синтез ЦУ следует ориентировать на создание прямоугольной (матричной) структуры однотипных логических элементов.

3. При реализации комбинационных примитивов необходимо использование адресуемых элементов памяти, применяемых в программируемых логических устройствах (FPGA, CPLD).

4. Формирование для каждого уровня комбинационного устройства ремонтных примитивов для восстановления работоспособности в режиме онлайн из расчета один запасной элемент на каждый тип компонента, используемый в уровне.

5. Стоимость аппаратурных затрат для реализации комбинационного устройства, ориентированного на высокое быстродействие, определяется суммой всех примитивов, привязанных к уровням комбинационного устройства, дополненной линейкой запасных элементов по одному для каждого слоя (при условии существования в каждом слое одинаковых примитивов): $Q = \sum_{i=\overline{1,n}}^{j=\overline{1,m}} P_{ij} + n.$

6. Реализация комбинационного устройства, ориентированного на минимизацию аппаратных затрат, определяется суммой всех типов примитивов, инвариантных к уровням комбинационного устройства, дополненной линейкой запасных элементов по одному для каждого типа: $Q = \sum_{i=\overline{1}}^{m} P_i + m.$

7. Обработка матрицы комбинационных элементов с помощью процессорной линейки примитивов, число которых равно мощности максимального уровня или слоя в прямоугольной структуре, что обеспечивает условия для параллельной обработки всех примитивов в каждом уровне элементов для повышения быстродействия комбинационного прототипа, реализуемого в PLD.

Таким образом, предложенный *Q*-метод интерпретативного исправного моделирования цифровых схем позволяет существенно повысить быстродействие и уменьшить объемы структур данных посредством замены таблиц истинности *Q*-покрытиями, что практически делает его конкурентоспособным относительно технологий компилятивного моделирования.

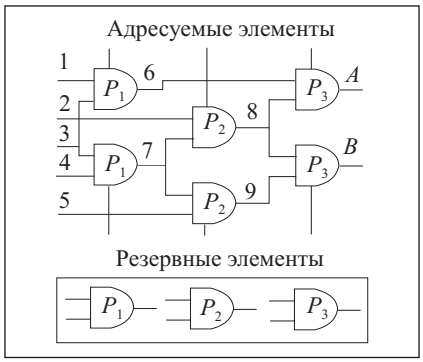

Рис. 5. Пример схемной структуры из адресуемых и запасных элементов

Восстановление работоспособности комбинационных устройств. В немногочисленных работах, посвященных восстановлению работоспособности логических схем [9, 11—14], описаны две идеи. Первая состоит в реконфигурации структуры логических элементов в режиме офлайн, обеспечивающей возможность замены каждого из неисправных примитивов, вторая — заключается в создании условий замены неисправных элементов с помощью использования запасных логических компонентов и мультиплексоров для переадресации отказавших примитивов.

Структуры кубитных данных модифицируются посредством добавления строки типов примитивов $F =< L, M, X, P, Q >, P = (P_1, P_2, ..., P_i, ..., P_m)$, задействованных при синтезе ЦС, если необходимо в процессе функционирования выполнять ремонт или восстановление работоспособности с помощью введения запасных примитивов, которые, так же как и основные, реализуются на основе элементов памяти. На рис. 5 приведен пример схемной структуры из адресуемых и трех запасных элементов. Соответствующие структуры данных с тремя дополнительными элементами представлены в табл. 3. Номера структурных примитивов дают возможность заменить любой отказавший элемент исправным из ремонтного запаса, изменив адресный номер в строке примитивов *P*. Ремонтные элементы в табл. 3 начинаются со столбца 7.

В табл. 4 представлены строка *L* типов логических элементов и адреса типов примитивов *P*. Данная структура ориентирована на программную реализацию моделирования, а ремонтные примитивы начинаются с номера 4. Если существует возможность перепрограммирования логики в элементе

Таблица 3

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	.	.

памяти с одинаковым числом входных переменных, то данную процедуру следует выполнять после фиксации неисправного элемента, когда становится известно, какой элемент в структуре и какой тип примитива отказал. Процедура восстановления работоспособности ориентирована на PLD-реализацию ЦС. При этом элементы памяти являются адресуемыми и каждый из них реализует логическую функцию, где входные двоичные наборы — это адреса ячеек, в которых записаны состояния выходов таблицы истинности. Если кубитные модели схем не имеют запасных примитивов, то получаем формат табл. 5.

Таким образом, кубитные структуры данных позволяют обеспечивать компактность описания функционалов ЦУ векторами состояний выходов, повышение быстродействия процедур моделирования вследствие адресации состояний выходов примитивов, а также восстановление работоспособности отдельных логических элементов при их реализации в элементах памяти PLD или в форме программных модулей. Следует заметить, что при этом не требуется хранить ремонтные примитивы, так как в предложенных интерпретативных структурах табличных данных изначально учтены технологические удобства устранения дефектов посредством пере-

Таблица 4

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	.	.	.	.	.

Таблица 5

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	.	.	.	.	.	.	.	.

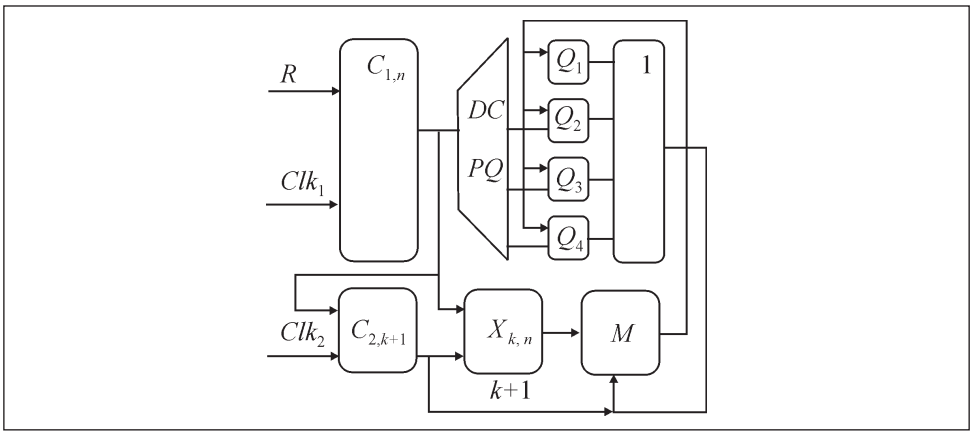

Рис. 6. Операционная структура комбинационной схемы

программирования компонентов PLD в процессе функционирования прототипа ЦУ.

Обработка схемы в кристалле сводится к определению адреса, составленного двоичными битами вектора моделирования, по которому находится значение логической функции. Каждому примитиву соответствует цикл обработки, состоящий из трех процедур:

1) адресное считывание номеров входных переменных из соответствующего столбца матрицы X для формирования адреса состояния входной переменной вектора моделирования: $A = X_{ij}$, $i = \overline{1,n}$, $j = \overline{1,s_p - 1}$;

2) формирование адреса (двоичного кода) для вычисления логической функции посредством конкатенации соответствующих состояний входных переменных в векторе моделирования: $A = M(X_{ij}) * M(X_{ir})$;

3) запись результата выполнения логической функции как состояния выхода в соответствующий разряд вектора моделирования: $M(X_{is_p}) = P[M(X_{ij}) * M(X_{ir})]$.

Процесс обработки всех примитивов схемы — строго последовательный, чем объясняется существенное замедление процедуры формирования состояний выходных переменных. Однако при этом обеспечивается встроенное и автономное восстановление работоспособности ЦС, что является одним из этапов функционирования инфраструктуры обслуживания SoC (рис. 6). На этом этапе комбинационная схема представляет собой операционное устройство с операционным и управляющим автоматами. Заменяемыми компонентами в операционном автомате являются типы примитивов — функциональные элементы, или структурные примитивы.

Операционное устройство для реализации элементно-адресуемых комбинационных схем содержит следующие элементы: счетчик обработ-

ки текущего примитива C_1; память для хранения типов примитивов, соответствующих структурным элементам P; счетчик считывания номеров входных и выходной переменных текущего примитива C_2; дешифратор типов примитивов DC; память для хранения вектора моделирования M; матричная память для хранения номеров входов-выходов структурных примитивов X; линейка памятей, реализующих функциональные примитивы $P(Q)$; регистр формирования входного адресного слова для обрабатываемого примитива RG; логический элемент Or для коммутации результатов обработки функциональных примитивов.

Граф-схема алгоритма управления процессом моделирования структуры комбинационной схемы (см. рис. 4) содержит следующие шаги:

1. Инициализация (формирование) всех компонентов (номера и типы элементов, линии связей для входов и выходов логических элементов) схемной структуры: $\underline{P} = (P_1, P_2, ..., P_i, ..., P_n)$, $Q = (Q_1, Q_2, ..., Q_j, ..., Q_m)$, $X = [X_{pq}]$, $p = \overline{1, n}$, $q = \overline{1, s_p}$.

2. Инициализация параметра обрабатываемого примитива и номера входного набора $i = 0$, $t = 0$ для его моделирования в двоичном алфавите: $M_r = \{0, 1\}$.

3. Инкрементирование индекса примитива, номера теста и инициализация входного тестового (рабочего) набора: $i = i + 1$, $t = t + 1$, $M(X) = T_t(X)$, $|T_t(X)| = \eta$.

4. Конкатенация (#) разрядов слова для формирования входного воздействия $\overset{k}{\underset{j=1}{\#}} M(X_{ij})$ логического элемента P_i (типа Q_i) и выполнение процедуры определения состояния его выхода с последующей записью в соответствующую координату вектора моделирования: $M(X_{k+1})$:

$$M(X_{k+1}) = \{P_i, Q_i\}\left[\overset{k}{\underset{j=1}{\#}} M(X_{ij})\right].$$

5. Повторение пунктов 3 и 4 для получения состояний выходов всех логических элементов до выполнения условия $i = n$.

6. Повторение пунктов 2—4 для моделирования всех входных тестовых (рабочих) наборов до выполнения равенства $t = \eta$, где η — длина теста.

7. Окончание процесса моделирования ЦУ.

Таким образом, введение в структуру ЦУ избыточных ремонтных компонентов и управляющего автомата, ориентированного на последовательную обработку комбинационных примитивов, дает возможность осуществлять процедуру переадресации примитивов в случае отказа одного из них. Нетрудно создать аналогичные автоматы для параллельной обработки

слоев из примитивов ранжированной схемы, что максимально приблизит быстродействие устройства к его реализации в кристаллах PLD.

Выводы

1. Метод диагностирования дефектов ЦС, усовершенствованный в результате использования единственной параллельной операции логического умножения, в сочетании со структурным маскированием неисправностей обеспечивает компактность представления данных, увеличение быстродействия и глубины диагностирования.

2. Предложенный *Q*-метод интерпретативного исправного моделирования ЦС с использованием компактных *Q*-покрытий вместо таблиц истинности дает возможность существенно повысить быстродействие анализа посредством адресного формирования выходов функциональных примитивов и уменьшить объемы структур данных, что позволяет считать его конкурентоспособным в области технологии компилятивного моделирования.

3. Модель ЦС, усовершенствованная с помощью добавления в структуру устройства избыточных ремонтных компонентов и управляющего автомата, ориентированного на последовательную обработку комбинационных примитивов, дает возможность осуществлять процедуру переадресации отказавших примитивов в режиме штатного функционирования.

4. Основная инновационная идея квантовых, или кубитных, вычислений заключается в переходе от вычислительных процедур над байт-операндом, определяющим в дискретном пространстве одно решение (точку), к квантовым параллельным процессам над кубит-операндом, одновременно формирующим булеан решений.

СПИСОК ЛИТЕРАТУРЫ

1. *Nielsen M.A., Chuang I.L.* Quantum Computation and Quantum Information. — Cambridge University Press, 2010. — 676 p.
2. *Хаханов В.И.*, *Литвинова Е.И., Гузь О.А*. Проектирование и тестирование цифровых систем на кристаллах. — Харьков, ХНУРЭ, 2009. — 484 с.
3. *Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S*. Information analysis infrastructure for diagnosis // Inform. an int. interdisciplinary journal. — 2011. — Vol. 14, № 7. — P. 2419—2433.
4. *Хаханов В.И., Мурад Али А., Литвинова Е.И. и др.* Квантовые модели вычислительных процессов // Радиоэлектроника и информатика. — 2011. — № 3. — С. 35—40.
5. *Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И.* Структура логического ассоциативного мультипроцессора// Автоматика и телемеханика. — 2012. — № 10. — С. 71—92.
6. *Хаханов В.И.* Техническая диагностика цифровых и микропроцессорных структур.— Киев: ИСИО, 1995. — 242 с.
7. *Hahanov V., Barkalov A., Adamsky M.* Infrastructure intellectual property for SoC simulation and diagnosis service. — Springer, 2011. — P. 289—330.

8. *Горбатов В.А.* Основы дискретной математики. — М. : Высш. школа,1986.— 311 с.
9. *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.
10. *Хаханов В.И., Багхдади Аммар Авни Аббас, Литвинова Е.И., Шкиль А.С.* Кубитные структуры данных вычислительных устройств // Электрон. моделирование.— 2015. — **37**, № 1.— С. 49—76.
11. *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.
12. *Hahanov V.I., Litvinova E.I., Chumachenko S.V. et al.* Qubit Model for solving the coverage problem // Proc. of IEEE East-West Design and Test Symposium. — Kharkov, 2012. — P. 142—144.
13. *Чжен Г., Мэннинг Е., Метц Г.* Диагностика отказов цифровых вычислительных систем. — М. : Мир, 1972. — 230 с.
14. *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. 13—18.

V.I. Hahanov, Tamer Bani Amer, S.V. Chumachenko, E.I. Litvinova

QUBIT TECHNOLOGIES OF ANALYSIS
AND DIAGNOSIS OF DIGITAL DEVICES

Technologies and examples of realization of qubit models, methods and algorithms have been proposed for increasing speed of response of existing software and hardware for analysis of digital computing devices as a result of increasing the dimension of structures of the data and memory for simultaneous storage of states under processing. The results of investigation of models and methods for diagnosing digital systems of modeling the reduction of serviceability of primitives removed from service.

K e y w o r d s: digital qubit structures, modeling, diagnosis and repair of digital systems.

REFERENCES

1. Nielsen, M.A. and Chuang, I.L. (2010), Quantum computation and quantum information, Cambridge University Press, Cambridge, U.K.
2. Hahanov, V.I., Litvinova, E.I. and Guz, O.A. (2009), *Proektirovanie i testirovanie tsifrovykh sistem na kristallakh* [Design and testing of digital systems on crystals], KhNRE, Kharkov, Ukraine.
3. Hahanov, V., Wajeb, Gharibi, Litvinova, E. and Chumachenko, S. (2011), "Information analysis infrastructure for diagnosis", *Inform. int. interdisciplinary journal,* Vol. 14, no. 7, pp. 2419-2433.
4. Hahanov, V.I., Murad, Ali A., Litvinova, E.I., Guz, O.A. and Hahanova I.V. (2011), "Quantum models of computation proceses", *Radioelektronika i informatika,* no. 3, pp. 35-40.
5. Bondarenko, M.F., Hahanov, V.I. and Litvinova, E.I. (2012), "Structure of logical associative multiprocessor", *Avtomatika i telemekhanika*, no. 10, pp. 71-92.
6. Hahanov, V.I. (1995), *Tekhnicheskaya diagnostika tsifrovykh i mikroprotsessornykh struktur* [Technical diagnostics of digital and microprocessor structures], ISIO, Kiev, Ukraine.
7. Hahanov, V., Barkalov, A. and Adamsky, M. (2011), "Infrastructure intellectual property for SoC simulation and diagnosis service", *Springer*, pp. 289-330.
8. Gorbatov V.A., (1986), *Osnovy diskretnoi matematiki* [Principles of higher mathematics], Vysshaya shkola, Moscow, Russia.

9. Hahanov, V.I., Litvinova, E.I., Hahanova, I.V. and Murad, Ali Abbas (2012), "Infrastructure of built-in restoration of logical PLD-circuits", *Radioelektronika i informatika*, no. 2, pp. 54-57.
10. Hahanov, V.I., Baghdadi Ammar Awni Abbas, Litvinova, E.I., Shkil, O.S. (2015), "Qubit data structures of computing devices", *Elektronnoe modelirovanie*, Vol. 37, no. 1, pp. 49-76.
11. Hahanov, V., Litvinova, E., Gharibi, W. and Murad Ali Abbas (2012), "Qubit models for SoC synthesis parallel and cloud computing", *USA,* Vol.1, iss. 1, pp. 16-20.
12. Hahanov, V.I., Litvinova, E.I., Chumachenko, S.V., Baghdadi Ammar Awni Abbas and Eshetie Abebech, Mandefro (2012), "Qubit model for solving the coverage problem", *Proc. of IEEE East-West Design and Test Symposium,* Kharkov, 2012, pp.142-144.
13. Chzhen, G., Manning, E. and Metts, G. (1972), *Dignostika otkazov tsifrovykh vychislitelnykh sistem* [Diagnostics of failures of digital computation systems], Mir, Moscow, Russia.
14. Koal, T., Scheit, D. and Vierhaus, H.T. (2009), "A comprehensive scheme for logic self repair", *Proc. Conf. on Signal Processing Algorithms, Architectures, Arrangements, and Applications*, pp. 13-18.

Поступила 16.10.13;
после доработки 19.12.13

ХАХАНОВ Владимир Иванович, д-р техн. наук, декан факультета компьютерной инженерии и управления, профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1978 г. окончил Харьковский ин-т радиоэлектроники. Область научных исследований — проектирование и тестирование вычислительных систем, сетей и программных продуктов.

ТАМЕР Бани Амер, аспирант кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. Область научных исследований — проектирование и тестирование вычислительных систем.

ЧУМАЧЕНКО Светлана Викторовна, д-р техн. наук, профессор, зав. кафедрой автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1991 г. окончила Харьковский госуниверситет. Область научных исследований — дискретная математика, моделирование вычислительных систем.

ЛИТВИНОВА Евгения Ивановна, д-р техн. наук, профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1985 г. окончила Харьковский ин-т радиоэлектроники. Область научных исследований — проектирование и тестирование цифровых систем и сетей на кристаллах.